**NXP Semiconductors**
Reference Manual

Document identifier: MCXE31XRM
Rev. 3, 2026-05-26

# MCXE31x Reference Manual

Supports MCXE315, MCXE316, MCXE317, and MCXE31B

# Contents

Chapter 1 About This Manual..... 7

Chapter 2 Introduction..... 14

Chapter 3 Memory Map..... 23

Chapter 4 Signal Multiplexing..... 27

Chapter 5 Cortex-M7 Overview..... 39

Chapter 6 Miscellaneous Control Module (MCM)..... 43

Chapter 7 Miscellaneous System Control Module (MSCM).....62

Chapter 8 Virtualization Wrapper (VIRT_WRAPPER)..... 94

Chapter 9 System Integration Unit Lite2 (SIUL2).....113

Chapter 10 Touch Sensing Pin Coupling (TSPC).....232

Chapter 11 Crossbar Switch (AXBS)..... 240

Chapter 12 Peripheral Bridge (AIPS_Lite)..... 252

Chapter 13 Direct Memory Access Multiplexer (DMAMUX)..... 254

Chapter 14 Enhanced Direct Memory Access (eDMA).....264

Chapter 15 Interrupt Monitor (INTM).....365

Chapter 16 Semaphores2 (SEMA42)..... 374

Chapter 17 Crossbar Integrity Checker (XBIC).....383

Chapter 18 Extended Resource Domain Controller (XRDC)..... 397

Chapter 19 Memory and Memory Interfaces..... 472

Chapter 20 Embedded Flash Memory (c40asf)..... 477

Chapter 21 Flash Memory Controller (PFLASH)..... 538

Chapter 22 RAM Controller (PRAMC)..... 562

Chapter 23 Clocking..... 569

Chapter 24 Clock Generation Module (MC_CGM)..... 637

Chapter 25 Fast Internal RC Oscillator (FIRC)..... 731

Chapter 26 Slow Internal RC Oscillator (SIRC)..... 734

Chapter 27 Fast Crystal Oscillator Digital Controller (FXOSC)..... 737

Chapter 28 Slow Crystal Oscillator Digital Controller (SXOSC)..... 744

Chapter 29 PLL Digital Interface (PLLDIG)..... 750

Chapter 30 Reset Overview..... 766

Chapter 31 Boot Overview..... 788

Chapter 32 Reset Generation Module (MC_RGM)..... 814

Chapter 33 Power-on Reset Watchdog (POR_WDG)..... 842

Chapter 34 Security Overview..... 845

Chapter 35 EdgeLock Secure Enclave (HSE_B)..... 847

Chapter 36 Device Configuration Format (DCF) records..... 867

Chapter 37 Device Configuration Module General-Purpose Registers (DCM_GPR)..... 872

Chapter 38 Device Configuration Module (DCM)..... 953

Chapter 39 Messaging Unit (MU)..... 995

Chapter 40 Power Management..... 1124

Chapter 41 Power Management Controller (PMC for MCXE31B)..... 1144

Chapter 42 Power Management Controller (PMC for MCXE315, MCXE316, and MCXE317)..... 1156

Chapter 43 Mode Entry Module (MC_ME)..... 1169

Chapter 44 Power Control Unit (MC_PCU)..... 1275

Chapter 45 Wakeup Unit (WKPU)..... 1278

Chapter 46 Safety Overview..... 1304

Chapter 47 Error Injection Module (EIM)..... 1314

Chapter 48 Error Reporting Module (ERM)..... 1370

Chapter 49 Fault Collection and Control Unit (FCCU)..... 1401

Chapter 50 Self-test General-Purpose Registers (SELFTEST_GPR)..... 1457

Chapter 51 Self-Test Control Unit (STCU2)..... 1461

Chapter 52 Register Protection (REG_PROT)..... 1506

Chapter 53 Clock Monitoring Unit – Frequency Check (CMU_FC)..... 1518

Chapter 54 Clock Monitoring Unit – Frequency Meter (CMU_FM)..... 1534

Chapter 55 Cyclic Redundancy Check (CRC)..... 1542

Chapter 56 Power Conversion and Motor Control (PCMC)..... 1553

Chapter 57 Analog-to-Digital Converter (ADC)....1570

Chapter 58 Low Power Comparator (LPCMP)....1733

Chapter 59 Logic Control Unit (LCU)....1774

Chapter 60 Enhanced Modular IO Subsystem (eMIOS)....1850

Chapter 61 Body Cross-triggering Unit (BCTU)....1937

Chapter 62 Trigger MUX (TRGMUX)....1976

Chapter 63 Software Watchdog Timer (SWT)....2038

Chapter 64 System Timer Module (STM)....2055

Chapter 65 Periodic Interrupt Timer (PIT)....2064

Chapter 66 Real Time Clock (RTC)....2088

Chapter 67 Low Power Serial Peripheral Interface (LPSPI)....2101

Chapter 68 Low Power Inter-Integrated Circuit (LPI2C)....2155

Chapter 69 Flexible I/O (FlexIO)....2218

Chapter 70 CAN (FlexCAN)....2297

Chapter 71 Synchronous Audio Interface (SAI)....2469

Chapter 72 Ethernet Media Access Controller (EMAC)....2519

Chapter 73 Low Power Universal Asynchronous Receiver/Transmitter (LPUART)....3109

Chapter 74 Quad Serial Peripheral Interface (QuadSPI)....3162

Chapter 75 Debug Subsystem....3251

Chapter 76 JTAG Controller (JTAGC)..... 3294

Chapter 77 JTAG Data Communication (JDC)..... 3303

Chapter 78 Temperature Sensor (TEMPSENSE).....3312

Appendix A Release notes.....3319

Legal information..... 3357

# Chapter 1
# About This Manual

## 1.1 Audience

This reference manual (RM) is intended for system software, hardware developers, and applications programmers who need to develop products using this chip. It assumes that its users understand operating systems, microprocessor system design, and basic principles of software and hardware.

## 1.2 Organization

This manual has two main sets of chapters.

- Chapters in the first set contain information that applies to all components on the chip.
- Chapters in the second set are organized into functional groupings that detail particular areas of functionality.
  - Examples of these groupings are clocking, timers, and communication interfaces.
  - Each grouping includes chapters that provide a technical description of individual modules.

### 1.2.1 Attachments

This manual includes key information in the files attached to it. For example, memory map and I/O details. Use the content in these attachments in conjunction with this manual's content.

**NOTE**

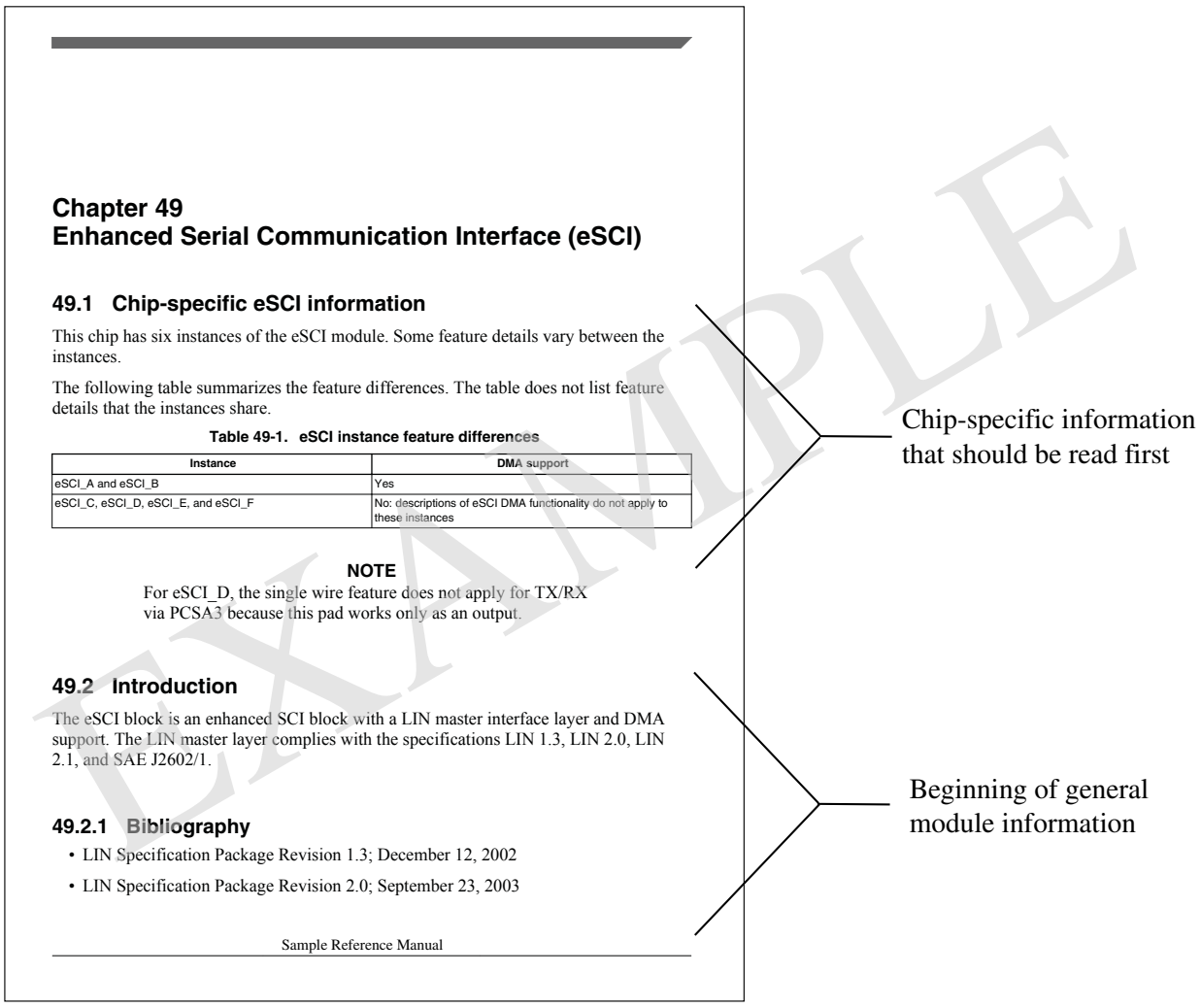
Select the paperclip icon on the left side of the PDF window to see the list of attachments.

## 1.3 Module descriptions

Each module chapter has two main parts:

- The first section, *chip-specific [module name]* information, provides details such as the number of module instances on the chip and connections between that module and the other ones. Read this section *first* because its content is crucial for understanding the information in the other sections of the chapter.
- The subsequent sections provide general information about the module, including its signals, registers, and functional description.

The following figure shows you an example of this demarcation.



Chapter 49
Enhanced Serial Communication Interface (eSCI)

49.1 Chip-specific eSCI information

This chip has six instances of the eSCI module. Some feature details vary between the instances.

The following table summarizes the feature differences. The table does not list feature details that the instances share.

Table 49-1. eSCI instance feature differences

| Instance | DMA support |
|---|---|
| eSCI_A and eSCI_B | Yes |
| eSCI_C, eSCI_D, eSCI_E, and eSCI_F | No: descriptions of eSCI DMA functionality do not apply to these instances |

NOTE
For eSCI_D, the single wire feature does not apply for TX/RX via PCSA3 because this pad works only as an output.

49.2 Introduction

The eSCI block is an enhanced SCI block with a LIN master interface layer and DMA support. The LIN master layer complies with the specifications LIN 1.3, LIN 2.0, LIN 2.1, and SAE J2602/1.

49.2.1 Bibliography

- LIN Specification Package Revision 1.3; December 12, 2002
- LIN Specification Package Revision 2.0; September 23, 2003

Sample Reference Manual



EXAMPLE

Figure 1. Example of chapter chip-specific information and general module information

### 1.3.1 Chip-specific information that clarifies content in the same chapter

The following figure shows an example of chip-specific information that clarifies general module information presented later in the chapter. In this case, the chip-specific register reset values supercede the reset values that appear in the register diagram.

**Chapter 9**
**System Integration Unit Lite2 (SIUL2)**

### 9.1 Chip-specific SIUL2 information

#### 9.1.1 Feature configurations

In this device, the SIUL2_0 module instance does not support the following features described in the generic description:

- Interrupts
- DMA channels

#### 9.1.2 Notes for IMCR

Out of reset, PA_00, PA_04, and PA_05 pads have JTAG input functionality selected by default. It should be disabled in the corresponding IMCR registers (IMCR61, IMCR60, and IMCR50 respectively) in order to use other functionality such as GPIO.

### 9.2 Introduction

#### 9.2.1 Overview

The System Integration Unit Lite2 provides control over all the electrical pin controls and ports with 16 bits of bidirectional, general-purpose input and output signals. One of the most important functions of the SIUL2 is to enable the user to select the functions and electrical characteristics that appear on external device pins. It also controls the multiplexing of internal signals from one module to another and controls chip I/O. It supports as many as 32 external interrupts with trigger event configuration. The following figure is the block diagram of SIUL2 and its interfaces to other system components.

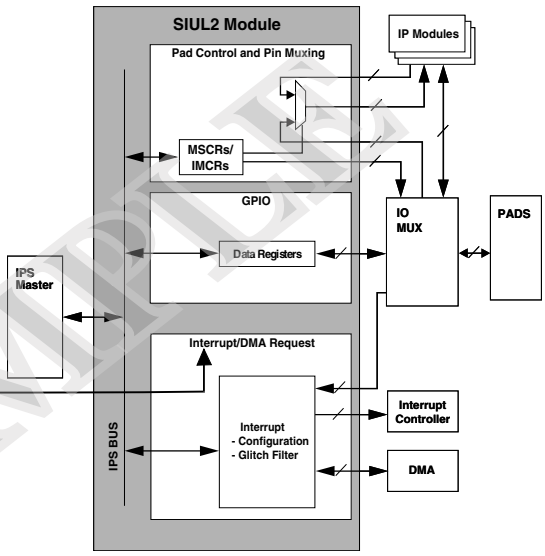



**Figure 23. System Integration Unit Lite2 block diagram**

This module provides dedicated pad control to general-purpose pads that can be configured as either inputs or outputs. The SIUL2 module provides registers that enable user software to read values from GPIO pads configured as inputs, and write values to GPIO pads configured as outputs:

- When configured as output, you can write to an internal register to control the state driven on the associated output pad.
- When configured as input, you can detect the state of the associated pad by reading the value from an internal register.
- When configured as input and output, the pad value can be read back, which can be used as a method of checking if the written value appeared on the pad.

To assist software development, GPIO data registers can be accessed using various mechanisms. These differing mechanisms allow support for port access or for bit manipulation without the need to use read-modify-write operations:

- Access to two 16-bit ports in one access
- Read/write access to a single bit
- A 16-bit port write with a bit mask, using single 32-bit access.

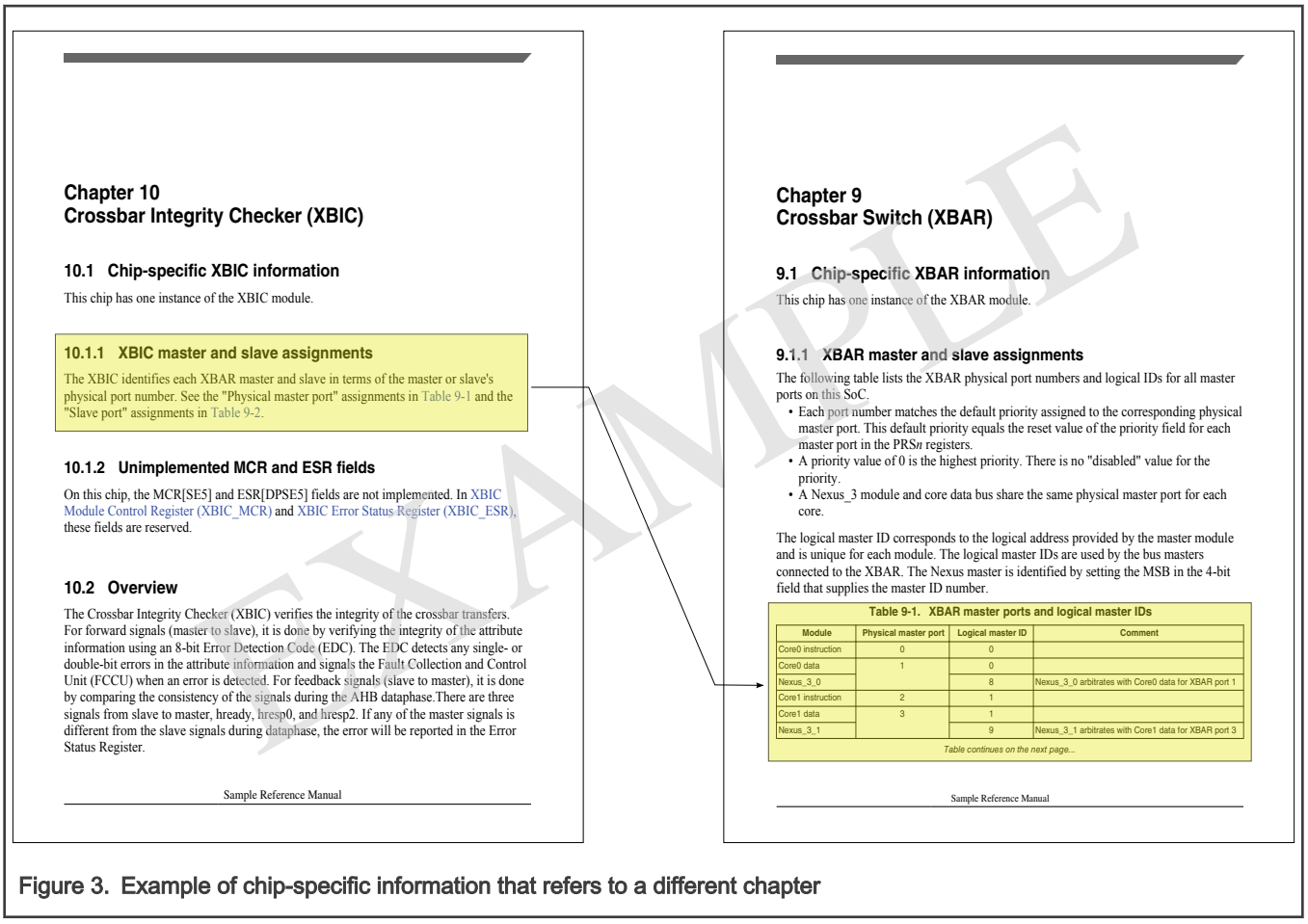


EXAMPLE

Figure 2. Example of chip-specific information that clarifies content in the same chapter

## 1.3.2 Chip-specific information that refers to a different chapter

Related chip-specific information may be provided in different chapters of the manual. The following figure shows an example of two such connected pieces of information. In this case, read both before you proceed.



Chapter 10
Crossbar Integrity Checker (XBIC)

10.1 Chip-specific XBIC information

This chip has one instance of the XBIC module.

10.1.1 XBIC master and slave assignments

The XBIC identifies each XBAR master and slave in terms of the master or slave's physical port number. See the "Physical master port" assignments in Table 9-1 and the "Slave port" assignments in Table 9-2.

10.1.2 Unimplemented MCR and ESR fields

On this chip, the MCR[SE5] and ESR[DPSE5] fields are not implemented. In XBIC Module Control Register (XBIC_MCR) and XBIC Error Status Register (XBIC_ESR), these fields are reserved.

10.2 Overview

The Crossbar Integrity Checker (XBIC) verifies the integrity of the crossbar transfers. For forward signals (master to slave), it is done by verifying the integrity of the attribute information using an 8-bit Error Detection Code (EDC). The EDC detects any single- or double-bit errors in the attribute information and signals the Fault Collection and Control Unit (FCCU) when an error is detected. For feedback signals (slave to master), it is done by comparing the consistency of the signals during the AHB dataphase.There are three signals from slave to master, hready, hresp0, and hresp2. If any of the master signals is different from the slave signals during dataphase, the error will be reported in the Error Status Register.

Sample Reference Manual

Chapter 9
Crossbar Switch (XBAR)

9.1 Chip-specific XBAR information

This chip has one instance of the XBAR module.

9.1.1 XBAR master and slave assignments

The following table lists the XBAR physical port numbers and logical IDs for all master ports on this SoC.

- Each port number matches the default priority assigned to the corresponding physical master port. This default priority equals the reset value of the priority field for each master port in the PRS*n* registers.
- A priority value of 0 is the highest priority. There is no "disabled" value for the priority.
- A Nexus_3 module and core data bus share the same physical master port for each core.

The logical master ID corresponds to the logical address provided by the master module and is unique for each module. The logical master IDs are used by the bus masters connected to the XBAR. The Nexus master is identified by setting the MSB in the 4-bit field that supplies the master ID number.

Table 9-1. XBAR master ports and logical master IDs

| Module | Physical master port | Logical master ID | Comment |
|---|---|---|---|
| Core0 instruction | 0 | 0 | |
| Core0 data | 1 | 0 | |
| Nexus_3_0 | | 8 | Nexus_3_0 arbitrates with Core0 data for XBAR port 1 |
| Core1 instruction | 2 | 1 | |
| Core1 data | 3 | 1 | |
| Nexus_3_1 | | 9 | Nexus_3_1 arbitrates with Core1 data for XBAR port 3 |

*Table continues on the next page...*

Sample Reference Manual

Figure 3. Example of chip-specific information that refers to a different chapter

## 1.4 Register descriptions

Module chapters present register information in the following:

- Memory maps, which contain:
  - An offset from the module's base address
  - The mnemonic and name of each register
  - The width of each register (in bits)
  - The reset value of each register
- Register figures
- Field-description tables
- Associated text

The following figure shows register figure conventions used throughout the manual.

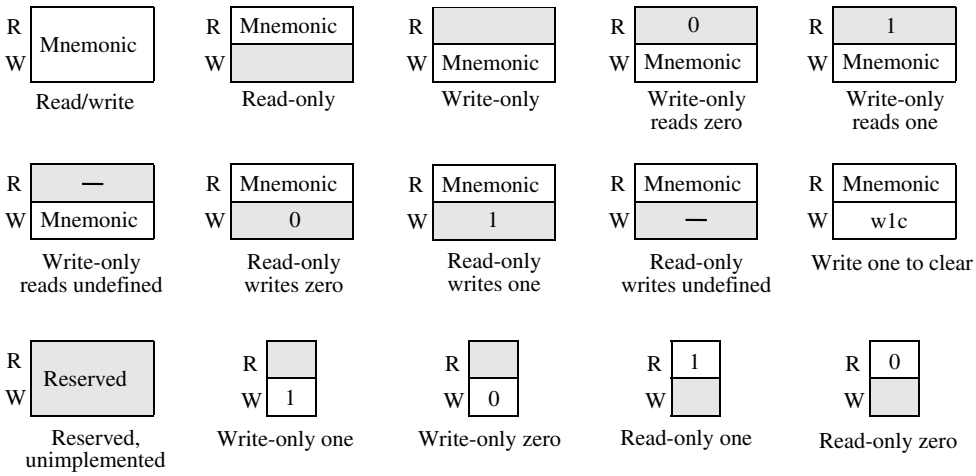



**Figure 4. Register figure conventions**

**NOTE**

Reset values of reserved locations documented in this manual are subject to change and must not be used for diagnostic purposes.

## 1.5 Conventions

### 1.5.1 Notes and cautions

Specific information is provided as part of notes and cautions throughout this manual.

**NOTE**

Emphasizes information that deserves extra attention.

**CAUTION**

Informs you of situations that could lead to highly undesirable outcomes—such as damage to the chip or irreversible malfunction.

### 1.5.2 Numbering systems

The following suffixes identify different numbering systems:

**Table 1. Numbering systems**

| This suffix | Identifies a |
|---|---|
| b | Binary number. For example, the binary equivalent of the number 5 is mentioned as 101b. In some cases, *0b* is prefixed to binary numbers. |
| d | Decimal number. Decimal numbers are followed by this suffix only when there is a possibility of confusion. In general, decimal numbers are used without a suffix. |
| h | Hexadecimal number. For example, the hexadecimal equivalent of the number 60 is mentioned as 3Ch. In some cases, *0x* is prefixed to hexadecimal numbers. |

### 1.5.3 Typographic notation

The following typographic notations are used throughout this document:

Table 2. Typographic notation

| Example | Description |
|---|---|
| *x* and other italicized text | The italized, lowercase *x* is used as a placeholder for replaceable numbers. In general, italicized text is used for titles of publications and for emphasis. Additionally, italics could be used for metasymbols in syntax descriptions. Plain lowercase letters are used as placeholders for single letters and numbers. |
| `code font` | Fixed-width font (such as Courier) used for code. It is used for a letter, word, or phrase that you want the user to type. For example, "Type `Read` and press Enter."<br>This type of font is also used for instruction mnemonics, directives, symbols, subcommands, parameters, operators, computer-language elements, code listings, commands that appear in running text, and for sample code. Instruction mnemonics and directives in text and tables are mentioned in all caps; for example, BSR. |
| SR[SCM] | A mnemonic in square brackets represents the name of a register field. This example refers to the Scaling Mode (SCM) field in the Status Register (SR). |
| REVNO[6:4], XAD[7:0] | Numbers in brackets that are separated by a colon represent either:<br>• A subset of a register's named field<br>For example, REVNO[6:4] refers to bits 6-4 that are part of the COREREV field occupying bits 6-0 of the REVNO register.<br>• A continuous range of individual signals of a bus<br>For example, XAD[7:0] refers to signals 7-0 of the XAD bus. |
| MOD.REG | A period separates the elements of a hierarchy: subsystem.module.register. For example:<br>• SWT.TO means that the TO register is located in the SWT module.<br>• SMU.XRDC.CR means that the CR register is located in the XRDC module within the SMU subsystem. |

### 1.5.4 Special terms

The following terms have special meanings.

Table 3. Special terms

| Term | Meaning |
|---|---|
| Asserted | Refers to the state of a signal as follows:<br>• An active-high signal is asserted when high (1).<br>• An active-low signal is asserted when low (0). |
| Deasserted | Refers to the state of a signal as follows:<br>• An active-high signal is deasserted when low (0).<br>• An active-low signal is deasserted when high (1).<br>In some cases, deasserted signals are described as *negated*. |
| Reserved | Refers to a memory space, register, field, or programming setting. Writes to a reserved location can result in unpredictable functionality or behavior. You must: |

*Table continues on the next page...*

Table 3. Special terms (continued)

| Term | Meaning |
|---|---|
| | • Not modify the default value of a reserved programming setting, such as the reset value of a reserved register field.<br>• Consider undefined locations in memory to be reserved.<br>• Not use the reset values of reserved locations documented in this manual for diagnostic purposes. They subject to change. |
| Write 1 to clear (w1c) | Refers to the access type of a register field that is used to clear the field by writing the value 1 to it. |
| Undefined (u) | Refers to undefined reset values |

## 1.6 Editorial changes

Each new release of this document includes editorial improvements such as:

- Spelling
- Grammar
- Punctuation
- Voice
- Tense
- Capitalization
- Formatting
- Punctuation
- Presentation
- Navigation

## 1.7 Glossary

The "Glossary" section provides a list of selected terms and definitions used in a chapter. The glossary, if present, is placed at the end of the chapter.

A glossary contains:

- NXP-specific terms, symbols, and notations, with definitions.
- Terms, symbols, and notations that can have different meanings in different contexts. The glossary defines the way they are used in the chapter—especially if the meaning is different from the meaning in standard references.

A glossary does not contain:

- Field mnemonics
- Register mnemonics
- Module mnemonics
- Signal names
- Industry-standard terms

# Chapter 2
# Introduction

## 2.1 Overview

The MCXE 31 product series further extends the highly-scalable portfolio of MCX E family in the commercial and industrial industries with the Arm Cortex-M7 core at higher frequency, more memory, SIL-2 rating and advanced security module. With a focus on industrial environment robustness, the MCXE31 product series devices are well suited to a wide range of applications in electrical harsh environments, and are optimized for cost-sensitive applications offering new, space saving package options. The MCX E31 series offers a broad range of memory, peripherals and performance options. Devices in this series share common peripherals and pin-out, allowing developers to migrate easily within a chip series or among other chip series to take advantage of more memory or feature integration.

**CAUTION**

MCXE315/MCXE316/MCXE317 and MCXE31B specific information is preliminary until these devices are qualified and may change without notice.

## 2.2 MCXE31 Series Introduction

**NOTE**

MCXE31 series has only a single ARM core and does not support lockstep mode. Any content in this manual related to multi-core or lockstep mode or dual-core interconnection must be ignored.

The MCXE31 series comes to market together with easy-to-use enablement software, application specific software, and various development tools, supported by broad third parties in different development phases.

The portfolio scalability, future proof feature like advanced security, as well as software/tool/third-party development support allows developers to standardize on the MCXE31 series for their end product platforms, maximizing hardware and software reuse, and reducing time-to-market.

Following are the general features of the MCXE31 series chips:

- 32-bit Arm Cortex-M7 core with IEEE-754 compliant SPFPU, executing up to 160 MHz
- Scalable memory footprints up to 4 MB flash memory and up to 512 KB SRAM
- Precision mixed-signal capability with low power comparators (LPCMP) and multiple 12-bit ADCs
- Powerful timers for a broad range of applications including motor control, lighting control and body applications
- Serial communication interfaces such as Serial communication interfaces including LPUART, LPSPI, LPI2C, FlexCAN with ISOCAN-FD support, Ethernet and QuadSPI. FXIO configuration allows other communication options including SENT.
- SESIP level 2 compliant EdgeLock Secure Enclave (HSE_B)
- Power supply (3.0 – 5.5 V) with fully functional flash memory program/erase/read operations
- Functional safety compliance with IEC 61508 SIL 2 hardware integrity and SIL 3 systematic capability (features depend on device specification)
  - Multiple internal watchdogs
  - Voltage monitors
  - Clock monitors
  - Memory protection
  - Data transport checks
  - ECC on memories

— Cyclic redundancy checking

- Ambient operation temperature range: –40°C to 135°C
- Junction temperature range: –40°C to 135°C

## 2.3 Feature summary

The MCX E31 product family includes the Arm Cortex-M7 core features described in the following table.

**Table 4. MCXE 31 chip's feature summary**

| Feature | Inclusions |
|---|---|
| Core and architecture | • Singel Arm Cortex-M7 core running up to 160 MHz<br>• Arm core based on the Armv7 architecture and ThumbR-2 ISA with 2.14 DMIPS/MHz<br>• 8 KB data and 8 KB instruction cache for optimizing wait state execution from memories<br>• 96 KB Tightly Coupled Memory associated with each core<br>• On-core MPU for dynamic task protection (16 regions)<br>• SPFPU, IEEE 754 compliant<br>• Harvard bus architecture implementing dedicated instruction and data path<br>• 5-stage pipeline with branch speculation<br>• XRDC integrated with a crossbar switch to provide memory and peripheral protection<br>• DSP<br>• I/O protection (VIRT_WRAPPER)<br>• ETM supporting instruction trace<br>• Arm third-party ecosystem support: software and tools to help minimize development time and cost |
| DMA | • Up to 2x64-channel DMAMUX<br>• eDMA with up to 32 channels<br>• Complex data transfers performed with minimal intervention from a host processor<br>• Programmable support for scatter-gather DMA processing |
| System and power management | • Support for simplified power modes (Run and Standby)<br>• Support for clock gating of unused modules; specific peripherals continue to work in low-power modes<br>• Support for external ballast transistor to generate core supply<br>• Fully independent CPU and peripheral clocking scheme<br>• Rapid start-up from a 48 MHz FIRC |

*Table continues on the next page...*

Table 4. MCXE 31 chip's feature summary (continued)

| Feature | Inclusions |
|---|---|
| | • Various low-power oscillators such as the 32 kHz SIRC and an external 32 kHz crystal support SXOSC<br>• PMC with LVD and selectable trip points<br>• Support for multiple power modes<br>• NMI |
| Memory and memory interfaces | • Up to 4 MB program flash memory, up to 128 KB data flash memory, and up to 512 KB SRAM, all with an ECC<br>• 4-bit/8-bit QuadSPI |
| Clocks | • External 8 MHz–40 MHz crystal oscillator or resonator<br>• External 32 kHz crystal oscillator<br>• Internal clock references<br>— 48 MHz FIRC ±5%<br>— 32 kHz SIRC ±10%<br>• Up to 960 MHz PLL for divided system clock operation |
| Security and integrity | • EdgeLock Secure Enclave (HSE_B)<br>— Upgradable Firmware delivered by NXP, to be programmed by the user<br>• Security ciphers:<br>— Symmetric: AES-128/192/256<br>— Cipher modes: ECB, CBC, CMAC, GMAC, CTR, OFB, CCM, and GCM<br>— Asymmetric: RSA (up to 4096 bits) and ECC (up to 521 bits)<br>— Hash: Miyaguchi-Preneel, SHA-2/SHA-3 (up to 512 bits)<br>— Number of keys is configurable and controlled by HSE FW<br>— Random number generator<br>• Security use case supported:<br>— Secure boot<br>— Secure communication<br>— Component protection<br>— Secure storage<br>— Key exchange |
| Safety IEC61508 | • Classification up to SIL-2<br>• ERM and EIM support<br>• WDOG with an independent clock source |

*Table continues on the next page...*

Table 4. MCXE 31 chip's feature summary (continued)

| Feature | Inclusions |
|---|---|
| | • Voltage monitors<br>• Bandgap voltage available as ADC input<br>• External clock source monitoring using an independent reference<br>• PLL lock and loss-of-lock protection<br>• XRDC<br>• ECC on code flash memory, data flash memory, and system RAM<br>• ADC self-test feature<br>• Internal analog monitoring of all supplies available<br>• CRC generation module<br>• FCCU failure output |
| Analog | • 12-bit ADC<br>— Up to 72 external analog inputs<br>— 1 µs conversion time<br>— Internal bandgap voltage reference channel, supporting automatic compare and an optional hardware trigger<br>— Up to five internal reference inputs<br>— Automatic compare with interrupt<br>— Self-test and self-calibration scheme<br>• ACMP with an internal 8-bit DAC as a reference<br>— ACMP with both positive and negative inputs, separately selectable interrupts on rising and falling comparator output<br>— Ability to cross trigger the timers from both the ADC and ACMP outputs<br>• Temperature Sensor (TempSense) with output measurable by ADC. |
| Timers | • 16-bit eMIOS timers, offering up to 72 standard channels<br>— Input capture, output compare, and PWM modes<br>— Fault input support with global fault control<br>— Multiple features such as deadtime insertion, configurable polarity, quadrature decoding, and so on<br>• 32-bit PITs, with four channels, for raising interrupts and triggering DMA channels<br>• 32-bit RTC<br>• Motor control and power conversion using combination of eMIOS, LCU , BCTU, and ADC<br>• Up to one STMs, with four channels each |

*Table continues on the next page...*

Table 4. MCXE 31 chip's feature summary (continued)

| Feature | Inclusions |
|---|---|
| Communications | • LPSPI supporting DMA with full-duplex or single-wire bidirectional communication in Master or Slave mode<br>• FlexIO, with an option to configure it as different communication peripherals, offering support for SENT<br>• LPI2C modules with DMA support, low-power availability, master or slave support, and system management bus<br>• LPUART with DMA support, having an optional 13-bit break, full-duplex non-return- to-zero (NRZ), low-power availability and supports LIN protocol versions 1.3, 2.0, 2.1, 2.2A, and SAE J2602 with using SW LIN driver<br>• FlexCAN modules with ISOCAN-FD and DMA support<br>• SAI capable of supporting stereo audio channels<br>• EMAC complex (10/100 Ethernet) that supports 1588 timers, MII/RMII interface, AVB, and TSN support |
| Debug | • DWT, with four configurable comparators as hardware watchpoints<br>• SWO-synchronous trace data support<br>• ITM with software and hardware trace, plus time stamping<br>• FPB with an ability to patch code and data from code space to system space<br>• Trace of all execution units and bus masters made available through an Arm TPIU over GPIO pins; a very low bandwidth trace option also available via the SWO<br>• Embedded trace FIFO (ETF)—a dedicated trace buffer available for each of the core masters, allowing data to be captured internally before being optionally routed to external trace pins<br>• SWV—trace capability providing displays of reads, writes, exceptions, PC samples, and print |
| I/O and package | • Up to 145 GPIO pins<br>• Up to 142 GPIO pins with interrupt functionality<br>• Up to 59 GPIO pins with wakeup capability<br>• Pseudo open drain support on LPUART, FlexIO, LPI2C<br>• Package options of 172 MAX QFP, 100 MAX QFP, 48 LQFP, and 100 LQFP |

## 2.4 Block diagram

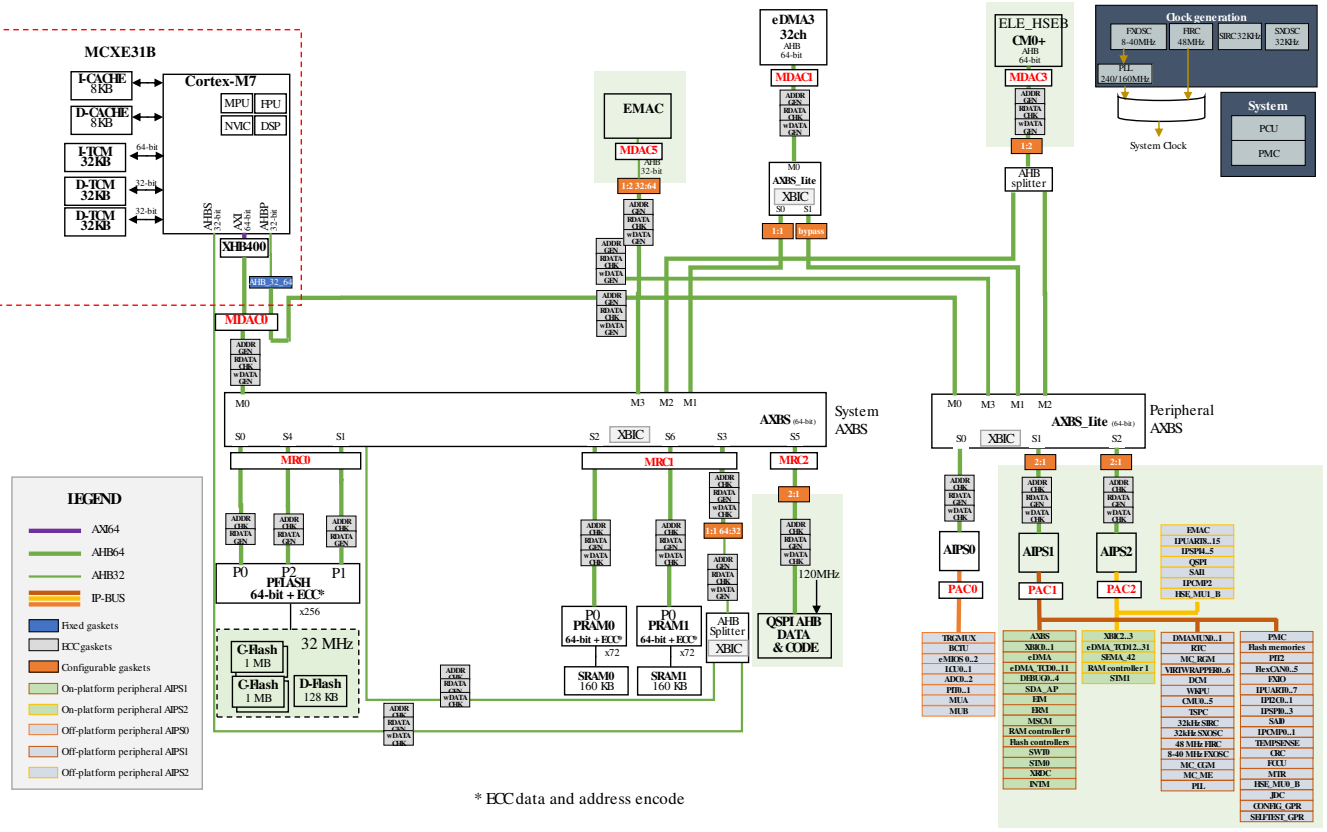



Figure 5. Block diagram - MCXE31B

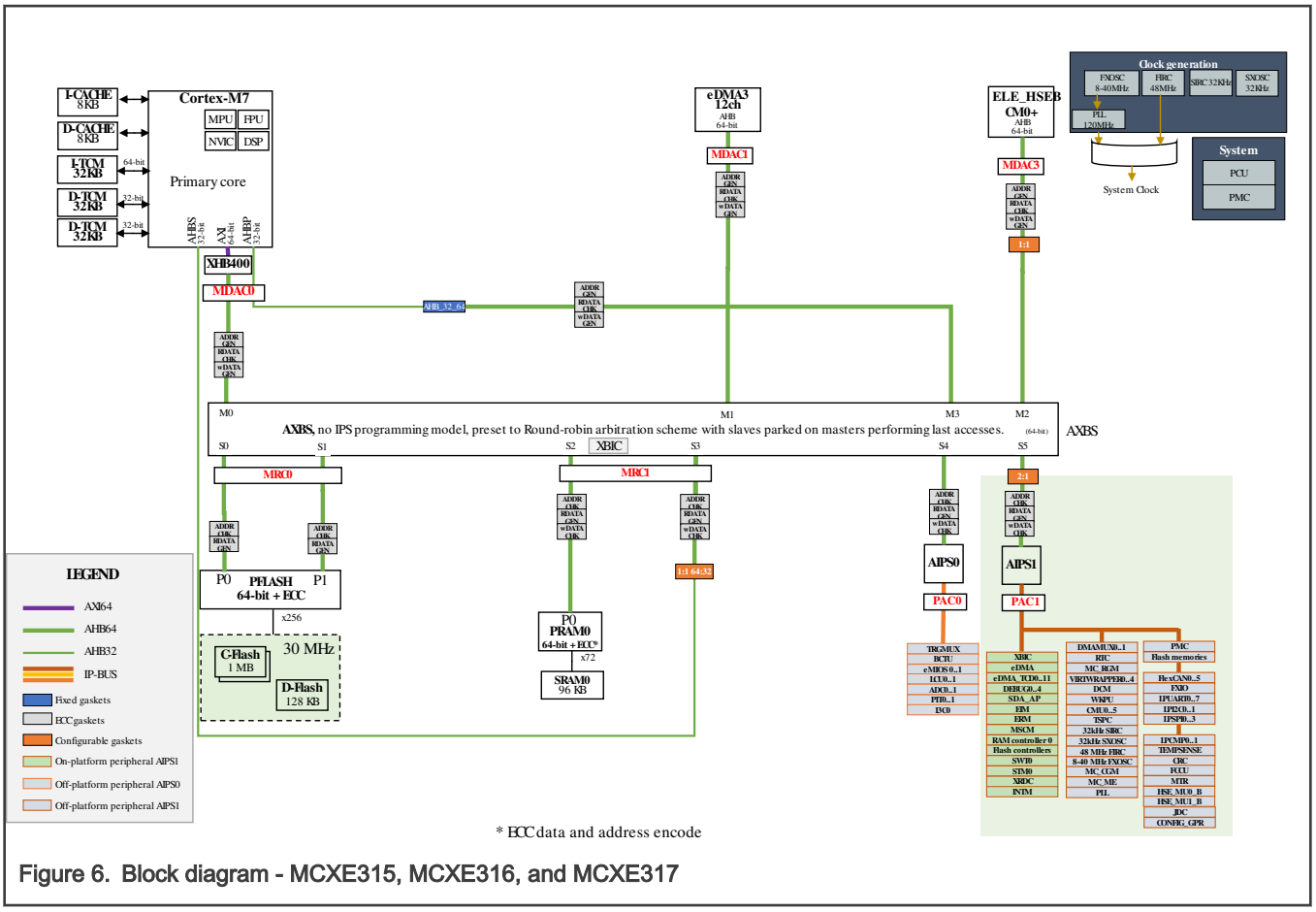



Figure 6. Block diagram - MCXE315, MCXE316, and MCXE317

## 2.5 Feature comparison

The following table compares some of the prominent features related to memory and package options of these chips from the MCXE31 family/product series:

- MCXE315
- MCXE316
- MCXE317
- MCXE31B

Table 5. MCXE31 chip's feature comparison

| Feature | Chip | | | |
|---|---|---|---|---|
| | MCXE315 | MCXE316 | MCXE317 | MCXE31B |
| Core | 1 x M7 | | | |
| Safety/SIL | 2 | | | |
| Program flash memory | 512 KB | 1 MB | 2 MB | 4 MB |
| Data flash memory (KB) | 64 | | 128 | |
| Total RAM (KB) | 112KB (incl. 96KB TCM) | 128KB (incl. 96KB TCM) | 192KB (incl. 96KB TCM) | 512KB (including 96KB TCM) |

*Table continues on the next page...*

Table 5. MCXE31 chip's feature comparison (continued)

| Feature | Chip | | | |
|---|---|---|---|---|
| | MCXE315 | MCXE316 | MCXE317 | MCXE31B |
| Standby RAM | 16 KB | 32 KB | | |
| Security | EdgeLock Secure Enclave (HSE_B) | | | |
| Frequency (MHz) | 120 | | | 160 |
| DMA channels | 12 | | | 32 |
| FlexCAN instances | 3 | | 6 | 6 |
| EMAC instances | — | | | 1 |
| SAI instances | — | | | 2 |
| LPUART instances | 4 | | 8 | 16 |
| LPSPI instances | 4 | | | 6 |
| $I^2C$ instances | 2 | | | |
| FlexIO (incl. SENT support) channels | 16 | | 32 | |
| QuadSPI instances | — | | | 1[1] |
| ADC instances | 2 | | | 3 |
| LPCMP instances | 1 | | 2 | 3 |
| PIT instances | 2 | | | 3 |
| SWT instances | 1 | | | 1 |
| STM instances | 1 | | | 2 |
| LCU instances | 2 | | | |
| BCTU instances | 1 | | | |
| TRGMUX instances | 1 | | | |
| eMIOS instances | 2 | | | 3 |
| RTC instances | 1 | | | |
| 172-HDQFP package | No | | Yes | |
| 100-HDQFP package | Yes | | | No |
| 48-pin LQFP package | Yes | | No | |
| 100-pin LQFP package | No | | Yes | |

1. 4-bit data width

## 2.6 Glossary

**AES** Advanced encryption standard

**SIL** SIL/Safety integrity Level. This is a risk classification scheme as defined by IEC 61508 functional safety standard for industrial

| | |
|---|---|
| AVB | Audio video bridging |
| CBC | Cipher block chaining |
| CCM | Counter with CBC MAC (Cipher block chaining message authentication code) |
| CMAC | Cipher-based message authentication code |
| CTR | Counter-based block cipher mode |
| DSP | Digital signal processor |
| DWT | Debug watchpoint and trace |
| ECB | Electronic code book |
| ECC | Elliptic curve cryptography/ Error code correction |
| ETM | Embedded trace macrocell |
| ETF | Embedded trace FIFO |
| FPB | Flash patch and breakpoint unit |
| GCM | Galois/Counter mode, an encryption algorithm |
| GMAC | Galois message authentication code |
| GPIO | General purpose input/output |
| ITM | Instrumentation trace macrocell |
| ISOCAN-FD | ISO 11898-1 compliant CAN with FD (Flexible datarate) |
| LVD | Low voltage detection |
| NMI | Non-maskable interrupt |
| OFB | Output feedback based block cipher mode |
| PIL | Processor-in-the-loop |
| PLL | Phase locked loop oscillator |
| PWM | Pulse width modulation |
| SDK | Software development kit |
| SENT | Single edge nibble transmission |
| SWV | Serial wire viewer |
| SPFPU | Single precision floating point unit |
| SWO | Serial wire output |
| TPIU | Trace port interface unit |
| TSN | Time sensitive networking |
| WDOG | Windowed watchdog |

# Chapter 3
# Memory Map

## 3.1 Introduction

This chip contains various memories and memory-mapped peripherals that are placed in a 32-bit contiguous memory space, and this chapter describes the memory and peripheral locations within that memory space.

For high-level chip memory map details, see the memory map file attached to this document.

## 3.2 SRAM memory map

A total of 512 KB RAM is available on MCXE31B product series, and it is partitioned as follows:

- ITCMs (64 KB)
- D0TCMs (64 KB)
- D1TCMs (64 KB)
- PRAMC_0 (160 KB)
- PRAMC_1 (160 KB)

PRAMC_0 controls the first 160 KB of RAM that is implemented as follows:

- 32 KB of memory supports content retention in Standby mode
- 128 KB of memory is available only in Run mode

PRAMC_1 controls the second 160 KB of RAM that is implemented as follows: :

- 32 KB of memory is available only in Run mode
- 128 KB of memory is available only in Run mode

The memory map file attached to this document provides a complete architectural address space definition for various sections that the RAM is partitioned into and across the MCXE31 product series. Based on the physical sizes of the memories and peripherals, the actual address regions used may be smaller.

In addition to ITCM, DTCM, and SRAM, a particular section of the RAM is associated with the Arm Cortex-M7 (CM7) internal cache. This chip includes 8 KB of I-cache RAM and 8 KB of D-cache RAM for each Cortex-M7 core.

The cache attributes in the MCXE31 cache scheme are defined for a given address range and core, and not according to the physical RAM or flash memory block.

## 3.3 Access-related details of the memory types used in this chip

The Cortex-M7 core can access these memories sequentially:

- ITCM
- DTCM
- I-cache
- D-cache

ITCM and DTCM can be accessed via 32-bit AHBS interface by any master, e.g., different Cortex-M7 cores, eDMA, etc to bootload instructions in ITCM. EMAC is another master that can access DTCM. See 'Block diagram' in the 'Introduction' chapter for details on the transaction path.

Access to SRAM beyond the RAM available on the chip terminates the bus cycle with an error followed by an appropriate response in the requesting bus master.

## 3.4 Considerations related to TCM's implementation

You must first initialize TCM (ITCM and DTCM) and system RAMs by 64-bit writes before performing read accesses. The system RAM can be initialized using eDMA and core. The ITCM initialization can be performed only by core using either direct or back-door accesses. The DTCM can be initialized also by 32-bit writes performed either using core's direct and back-door accesses as or eDMA. These writes are required to set up the initial ECC code words after chip power-on reset.

Each Cortex-M7 core is equipped with a 32 KB ITCM and 64 KB DTCM with a zero wait-state access.

See table 'Memory ECC initialization summary' in chapter 'Memory and Memory Interfaces' for details on memory ECC initialization.

## 3.5 Flash memory map

For details, see the memory map file attached to this document.

## 3.6 AIPS-Lite memory map

You can access the peripheral memory map via a crossbar slave port. The next table shows the three regions associated with peripheral space.

**Table 6. Regions associated with peripheral space**

| Address of region | Region description |
|---|---|
| 4000_0000h–401F_FFFFh | This 2048 KB region (AIPS_Lite_0) is partitioned into 128 spaces, each 16 KB in size, having 32 on-platform and 96 off-platform spaces. AIPS_Lite generates unique module enables for all the 32 on-platform spaces. |
| 4020_0000h–403F_FFFFh | This 2048 KB region (AIPS_Lite_1) is partitioned into 128 spaces, each 16 KB in size, having 32 on-platform and 96 off-platform spaces. AIPS_Lite generates unique module enables for all the 32 on-platform spaces. |
| 4040_0000h–405F_FFFFh | This 2048 KB region (AIPS_Lite_2) is partitioned into 128 spaces, each 16 KB in size, having 32 on-platform and 96 off-platform spaces. AIPS_Lite generates unique module enables for all the 32 on-platform spaces. |

Modules that are disabled via their clock gate control fields in the MC_CGM registers disable the associated AIPS_Lite slots. Access to any address within an unimplemented or disabled peripheral bridge slot results in a transfer error termination.

Multiple instances of same peripherals are connected to different bridges on the interconnect. For details, see the memory map file attached to this document.

## 3.7 Serialization of memory operations

In particular cases, you must complete the process of writing to a peripheral before the subsequent action occurs. Examples of such situations include:

- Exiting an interrupt service routine
- Changing a mode
- Configuring a function

In these situations, you must perform a read-after-write sequence to achieve the required serialization of memory operations. The following table provides this sequence.

Table 7. Read-after-write sequence for serialization of memory operations

| Step | Action |
|---|---|
| 1 | Write to the associated peripheral register. |
| 2 | Read the register to verify the write process. |
| 3 | Continue with the subsequent operations. |

## 3.8 PPB memory map

PPB is a part of the defined Arm bus architecture and provides access to specific processor-local modules. You can access these modules only through the core, and not through other system masters.

Table 8. PPB memory map

| Starting hex address | Ending hex address | Size (KB) | Module |
|---|---|---|---|
| E000_0000 | E000_0FFF | 4 | ITM |
| E000_1000 | E000_1FFF | | DWT |
| E000_2000 | E000_2FFF | | FPB |
| E000_3000 | E000_DFFF | 44 | — |
| E000_E000 | E000_EFFF | 4 | SCS |
| E000_F000 | E003_FFFF | 196 | Reserved |
| E004_0000 | E004_0FFF | 4 | TPIU |
| E004_1000 | E004_1FFF | | ETM |
| E004_2000 | E004_2FFF | | CTI |
| E004_3000 | E004_3FFF | | — |
| E004_4000 | E004_4FFF | | |
| E004_5000 | E004_5FFF | | |
| E004_6000 | E007_FFFF | 232 | |
| E008_0000 | E008_0FFF | 4 | MCM |
| E008_1000 | E008_1FFF | | — |
| E008_2000 | E008_2FFF | | |
| E008_3000 | E00F_EFFF | 496 | |
| E00F_F000 | E00F_FFFF | 4 | Cortex-M7 PPB ROM table |

## 3.9 Glossary

| | |
|---|---|
| **CTI** | Cross trigger interface |
| **DTCM** | Data tightly coupled memory |
| **D-cache** | Data cache |
| **DWT** | Debug watchpoint and trace |
| **ETM** | Embedded trace macrocell |

| | |
|---|---|
| **FPB** | Flash patch and breakpoints |
| **ITCM** | Instruction tightly coupled memory |
| **I-cache** | Instruction cache |
| **ITM** | Instrumentation trace macrocells |
| **PPB** | Private peripheral bus |
| **SCS** | System control space |
| **SRAM** | Static random access memory |
| **TPIU** | Trace port interface unit |

# Chapter 4
# Signal Multiplexing

## 4.1 Introduction

The signal multiplexing enables the sharing of single pad for multiple functions.

The signal multiplexing unit comprises control signals from SIUL2 and pad interface logic. The signal multiplexing unit consists of several individual sub-units, each handling the signal multiplexing of one pad.

The "SIUL2 Multiplexed Signal Configuration Register (MSCR)" controls the module specific pad settings (pull-up etc) and the signal present on the external pin. See SIUL2_MSCR for the description of control signals. The pad attributes may vary depending on the pad type.

For the pad attributes of each pad type and their reset values per port, see the IOMUX file attached to this document. The pads specific to the packages and their multiplexing is also documented in the IOMUX file attached to this document.

**NOTE**

The input functions for the protocols which are not to be used should be appropriated configured as 'disabled low'/'disabled high' with appropriate SIUL2.IMCR configurations corresponding for that function.

## 4.2 Pad description

Following figure shows the basic representation of a GPIO pad.

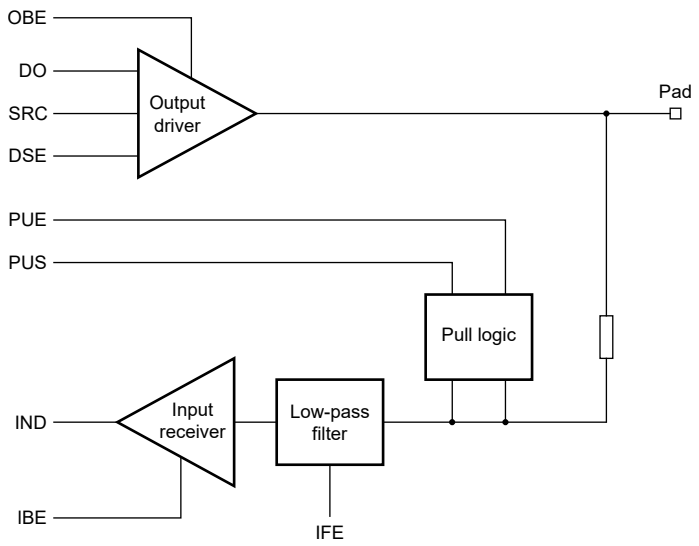



**Figure 7. GPIO pad representation**

**Table 9. Pad Signal description**

| Signal name | Direction | Descriprtion |
|---|---|---|
| Pad | I/O | I/O to external world |
| DO | I | Data coming from the core into the pad |
| OBE | I | Enable output driver |

*Table continues on the next page...*

Table 9. Pad Signal description (continued)

| Signal name | Direction | Descriprtion |
|---|---|---|
| PUE | I | 0: Disable internal pullup or pulldown resistor 1: Enable internal pullup or pulldown resistor |
| PUS | I | 0: Enable internal pulldown resistor if pue is set 1: Enable internal pullup resistor if pue is set |
| IBE | I | Enable input receiver |
| IND | O | Data coming out of the pad into the core |
| SRC | I | Slew Rate Control |
| PKE | I | Pad keeping enable |
| IFE | I | Input filter enable |
| DSE | I | Drive Strength enable |

Table 10. Input buffer enable

| IBE | Pad | IND | Description |
|---|---|---|---|
| 0 | X | 0 | Input buffer disabled, ind gets low |
| 1 | 0/1 | 0/1 | Input buffer enabled, ind = pad |

Table 11. Output buffer enable

| OBE | DO | Pad | Description |
|---|---|---|---|
| 0 | X | Z | Output buffer disabled, pad hi-Z (If not configured as input) |
| 1 | 0/1 | 0/1 | Output buffer enable, pad = do |

Table 12. Input filter enable

| IFE | Description |
|---|---|
| 1 | Input filter enabled |
| 0 | Input filter disabled |

Table 13. Pull up/Pull down

| PUE | PUS | Pad | Description |
|---|---|---|---|
| 0 | X | - | Weak pull disabled. Pad retains previous state |
| 1 | 0 | 0 | Weak pull down enabled |
| 1 | 1 | 1 | Weak pull up enabled |

Table 14. Slew rate control

| SRC | Description |
|---|---|
| 0 | Slew rate enabled |
| 1 | Slew rate disabled |

Table 15. Drive strength enable

| DSE | Description |
|---|---|
| 1 | Drive strength supported |
| 0 | Drive strength not supported |

Table 16. Pad keeping enable

| PKE | Description |
|---|---|
| 0 | Pad keeping disabled |
| 1 | Pad keeping enabled |

**NOTE**

The default state of GPIO pins on a reset event is high-Z. The high-Z state might settle to active high or active low at chip depending on the supply, temperature and other factors. Hence, it is recommended to use external pulls to ensure safe inactive state in event of a reset.

## 4.3 Functional description

The signal multiplexing architectural implementation is as shown in the following figure.

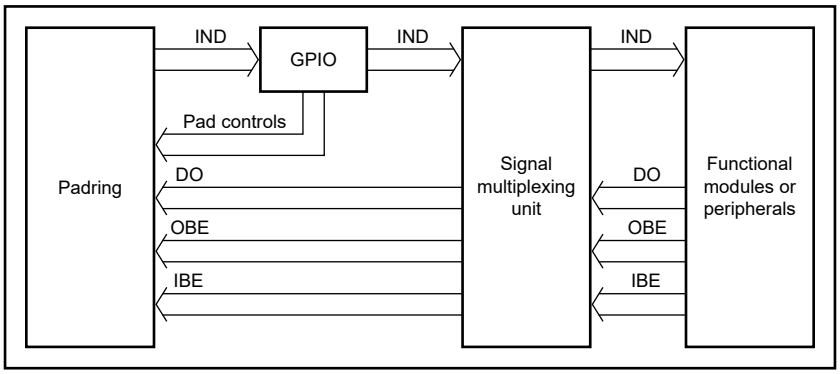



Figure 8. Signal Multiplexing

## 4.4 Signal Multiplexing sheet

IO Signal Description Input Multiplexing sheet(s) attached to the Reference Manual contains information on pins/balls of this device.

The 'IO Signal Table' and 'Input Muxing' tabs in the sheet correspond to the signal multiplexing information. The 'IO Signal Table' consists of all the pin muxing details and the 'Input Muxing' specifies the priority for the input muxing where an input path is driven by more than one pad.

### 4.4.1 IO Signal Table

Following is an example snippet of IO Signal Table. For selecting any functionality, the pad MSCR register (refer to SIUL2_MSCRn) needs to be configured accordingly.

| Port | CR | SSS | Function | Module | Description | I/O Direction | Pad Type | MCXE314_172mqf | I/O Power Domain | Pad State During Destructive Reset | Pad State After Destructive Reset | Pad State During Functional Reset | Pad State After Functional Reset | Pad State After Selftest | Pad State After FCCU Error | I/O Power Segment | MSCR | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | OBE[31] | IBE[19] | INV[17] | SMC[16] | DSE[14] | PUE[13] | PUS[11] | IDS[8] | IF[3:0] | SMC[5] | SSS[2:0] |
| PTA0 | SIUL_MSCR0 | 0000_0000 | GPIO[0] | SIUL | GPIO[0] | I/O | GPIO-STANDARD | 137 | VDD_HV_A | Hi-Z | Hi-Z | Hi-Z | Hi-Z | Hi-Z | Hi-Z | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | - | - | 000 |
| PTA0 | | 0001_0001 | LPSPI4_PCS2 | LPSPI4 | Peripheral Chip Select 2 | O | | | | | | | | | | | | | | | | | | | | | |
| PTA0 | | 0010_0010 | eMIOS_0_CH[17]_Y | eMIOS_0 | eMIOS Channel | O | | | | | | | | | | | | | | | | | | | | | |
| PTA0 | | 0011_0011 | LCU0_OUT4 | LCU0 | LCU Output | O | | | | | | | | | | | | | | | | | | | | | |
| PTA0 | | 0100_0100 | FXIO_D2 | FXIO | FlexIO Bi-directional Shift/timer I/O | O | | | | | | | | | | | | | | | | | | | | | |
| PTA0 | | 0101_0101 | eMIOS_1_CH[0]_X | eMIOS_1 | eMIOS Channel | O | | | | | | | | | | | | | | | | | | | | | |
| PTA0 | | 0110_0110 | LPSPI0_PCS7 | LPSPI0 | Peripheral Chip Select 7 | O | | | | | | | | | | | | | | | | | | | | | |
| PTA0 | | 0111_0111 | TRGMUX_OUT3 | TRGMUX | Trigger Mux Output | O | | | | | | | | | | | | | | | | | | | | | |
| PTA0 | - | - | ADC0_S8 | ADC0 | ADC Standard Input | I | | | | | | | | | | | | | | | | | | | | | |
| PTA0 | | | CMP1_IN0 | CMP1 | Comparator Input Signal | I | | | | | | | | | | | | | | | | | | | | | |
| PTA0 | SIUL_IMCR528 | 0000_0000 | EIRQ[0] | SIUL | External Interrupt | I | | | | | | | | | | | | | | | | | | | | | |
| PTA0 | SIUL_IMCR576 | 0000_0000 | eMIOS_0_CH[17]_Y | eMIOS_0 | eMIOS Channel | I | | | | | | | | | | | | | | | | | | | | | |
| PTA0 | SIUL_IMCR592 | 0000_0000 | eMIOS_1_CH[0]_X | eMIOS_1 | eMIOS Channel | I | | | | | | | | | | | | | | | | | | | | | |
| PTA0 | SIUL_IMCR640 | 0000_0000 | FXIO_D2 | FXIO | FlexIO Bi-directional Shift/timer I/O | I | | | | | | | | | | | | | | | | | | | | | |
| PTA0 | SIUL_IMCR736 | 0000_0000 | LPSPI0_PCS7 | LPSPI0 | Peripheral Chip Select 7 | I | | | | | | | | | | | | | | | | | | | | | |
| PTA0 | SIUL_IMCR768 | 0000_0000 | LPSPI4_PCS2 | LPSPI4 | Peripheral Chip Select 2 | I | | | | | | | | | | | | | | | | | | | | | |
| PTA0 | SIUL_IMCR832 | 0000_0000 | LPUART0_CTS | LPUART0 | Clear To Send (bar) | I | | | | | | | | | | | | | | | | | | | | | |

**Figure 9. IO signal table snippet**

The columns of the above figure are described below:

- Port: This field in IO Signal Table specifies the PAD names of the device.
- CR(Control Register): This field specifies the name of MSCR corresponding to the Port field. 'On this device, there are eight port groups (PTA, PTB, PTC, PTD, PTE, PTF and PTG) that are controlled by SIUL2_MSCRn registers. The below table shows the mapping of ports with respect to SIUL2_MSCRn registers.

**Table 17. Port/MSCR mapping**

| Port | SIUL2_MSCRn index |
|---|---|
| PortA[0-31] | MSCR0 - MSCR31 |
| PortB[0-31] | MSCR32 - MSCR63 |
| PortC[0-31] | MSCR64 - MSCR95 |
| PortD[0-31] | MSCR96 - MSCR127 |
| PortE[0-31] | MSCR128 - MSCR159 |
| PortF[0-31] | MSCR160 - MSCR191 |
| PortG[0-27] | MSCR192 - MSCR219 |

See SIUL2_MSCR/IMCR (See following Input Muxing section section for details on IMCR) for description of MSCR/IMCR fields.

- SSS: This field specifies the ALT mode of operation as per MSCR[Mux_mode]. Not all pins support all pin muxing slots. Unimplemented pin muxing slots are reserved. The corresponding pin is configured in the following pin muxing slot as follows:
  - 000: Alternative 0 (GPIO)
  - 001: Alternative 1 (chip-specific)
  - 010: Alternative 2 (chip-specific)
  - 011: Alternative 3 (chip-specific)
  - 100: Alternative 4 (chip-specific)
  - 101: Alternative 5 (chip-specific)
  - 110: Alternative 6 (chip-specific)
  - 111: Alternative 7 (chip-specific)

**NOTE**

The analog functionalities are specified with '-' in this field.

- Function: This field specifies the functionality of the pad as per the corresponding ALT mode specified by SSS field.
- Module: The Module field contains the module which is governing the pad for the ALT mode.

- Description: This field mentions a short description of pad functionality.
- Direction: This field specifies the direction (Input, Output or Input/Output) of the pad for the concerned functionality.
- Pad Type: This field mentions the pad type of the corresponding pad.
  - GPIO-Standard:
    - Switching up to 10MHz
    - High drive-strengh not supported.
    - Slew-rate control not supported.
  - GPIO-Standard plus:
    - Switching up to 25MHz
    - Supports high drive-strengh.
    - Slew-rate control not supported.
  - GPIO-Medium:
    - Switching up to 50MHz
    - Supports high drive-strengh.
    - Supports slew-rate control.
  - GPIO-Fast:
    - Switching up to 120MHz
    - Supports high drive-strengh.
    - Supports slew-rate control.
- The next columns specify the pin number in the supported packages for the device.
- I/O Power Domain: This field refers to the power domain of associated pad (VDD_HV_A / VDD_HV_B).
- Pad State during/after Reset: These fields represent the pad states during and post different device resets.
- MSCR: This field specifies the default MSCR value for corresponding pad. Refer SIUL2_MSCRn for description of MSCR fields.
- The next two columns specify the reset value and the configurable bit fields of MSCR corresponding to pad.

### 4.4.2 Input muxing table

As the same function can be multiplexed to several pads by configuring their respective IMCRs, there is priority input muxing. In case of same input being driven from multiple pads, the one with highest priority (1 being the highest) will drive the input. Following is a snippet of Input Muxing Table.

| Destination Instance | Destination Function | CR Instance | Input CR# | Input SSS | Source Instance | Source Signal | MCXE31x_172mqfp_DC |
|---|---|---|---|---|---|---|---|
| CAN0 | CAN0_RX | SIUL | SIUL_IMCR512 | 0000_0000 | - | disable low | |
| | | | | 0000_0001 | IO_PAD | PTC2 | 50 |
| | | | | 0000_0010 | IO_PAD | PTA6 | 102 |
| | | | | 0000_0011 | IO_PAD | PTB0 | 95 |
| | | | | 0000_0100 | IO_PAD | PTA28 | 30 |
| | | | | 0000_0101 | IO_PAD | PTF21 | - |

Figure 10. Input muxing table snippet

The columns of the figure are briefly described below:

- Destination Instance: This field contains the instance name of the input path to where the signal will propagate from padring.
- Destination Function: This field mentions the function name of the input path.

- Input SSS: This field specifies the IMCR[Mux_mode] value corresponding to the pad specified in source signal column.
- Source Instance: This field specifies the source pad type. A blank is mentioned for the default source when no pad is driving the input path.
- Source Signal: This field mentions the pad name. A ‘disable low’/’disable high’ specifies the signal behavior when none of the pads are driving the input path.
- The next columns specify the pin number in the supported packages for the device.

### 4.4.3 MSCR/IMCR description/explanation

#### MSCR assignments

The Implemented SIUL2 Multiplexed Single Configuration Register details is provided in the I/O Signal Description Table attached as excel.

Example:

If user wants to configure PTB0 as CAN0RX and PTB1 as CAN0TX, then the following configuration can be used in SIUL2 registers:

```
//.CAN0.RX.(PTB0):
SIUL2.IMCR0.B.SSS.=.0b011;.//.Select.CAN0-RX
SIUL2.MSCR32.B.IBE.=.1;....//.Enable.the.input.buffer

//.CAN0.TX.(PTB1):
SIUL2.MSCR33.B.SSS.=.0b101;.//.Select.CAN0-TX
SIUL2.MSCR33.B.OBE.=.1;.//.Enable.the.output.buffer
```

MSCR bit fields correspond to pin/pad basis, these are independent of muxing implemented on that specific pin/pad. As an example, pad PTA31 has SSS, SMC, DSE, PUS, PKE, INV, IBE and OBE are implemented with the reset value 0 and SRC is implemented with reset value 1.

In the 'IO Signal Table' tab of the IOMUX file attached, the MSCR register bits shows ‘0’, ‘1’, and ‘-’ for state of all bits associated with different ports. ‘0’ or ’1’ represents the reset state and ‘-’ represents the bit is not supported on the respective port.

**NOTE**

The GPIOs 38 and 39 are used for direct connections and FXOSC.

### 4.4.4 Pinout diagrams

See IO Signal Description Input Multiplexing sheet(s) attached to the Reference Manual for pinout diagrams corresponding to available packages.

## 4.5 Pin States

The tables in the upcoming sections mention the state of pins of the device under various conditions/event like Functional Reset, Destructive Reset, Power Up condition, during Selftest Phase etc. The details of the Reset events can be found in the Reset overview Chapter. The Details of Selftest can be found in the STCU chapter.

### 4.5.1 Pin numbers

The pin numbers which have been referenced to the tables are explicitly mentioned here

- RESET_B: PTA5 = GPIO5. The pad state tables below assume RESET_B configured as Reset pin (See UTEST DCF record 'dcf_client_reset_pad_dedicated' in the DCF clients file attached to this document for Reset pin control as dedicated/muxed GPIO'.
- JTAG_TMS : PTA4 = GPIO4
- JTAG_TCK : PTC4 = GPIO68
- JTAG_TDI : PTC5 = GPIO69
- JTAG_TDO : PTA10 = GPIO10
- TRACE_ETM_CLKOUT : PTG6 = GPIO[198] (Slow pad trace), PTC2=GPIO[66] (Fast pad trace)
- TRACE_ETM_D0 : PTG7 = GPIO[199] (Slow pad trace), PTD7=GPIO[103] (Fast pad trace)
- TRACE_ETM_D1 : PTG15 = GPIO[207] (Slow pad trace), PTD12=GPIO[108] (Fast pad trace)
- TRACE_ETM_D2 : PTG16 = GPIO[208] (Slow pad trace), PTD11=GPIO[107] (Fast pad trace)
- TRACE_ETM_D3 : PTF31 = GPIO[191] (Slow pad trace), PTD10=GPIO[106] (Fast pad trace)
- TRACE_ETM_D4 : PTG17 = GPIO[209] (Slow pad trace)
- TRACE_ETM_D5 : PTF28 = GPIO[188] (Slow pad trace)
- TRACE_ETM_D[6:15] : PTG18:PTG27 = GPIO[210]:GPIO[219] (Slow pad trace)
- CLKOUT_STANDBY : PTA12 = GPIO12
- FCCU_ERR[0:1] : PTA2:PTA3 = GPIO[2]:GPIO[3]

### 4.5.2 Power up

This table mentions the pin behavior on power up condition of the device.

**Table 18. Power up**

| Pin function | POWER UP until SW comes up[1] |
|---|---|
| RESET_B | LOW |
| JTAG_TMS,JTAG_TCK,JTAG_TDI,JTAG_TDO as JTAG | PULLED Values:<br>(TMS = HIGH, TCK = LOW, TDI =HIGH)<br>TDO: HIGH Z |
| ETM_TRACE | HIGH Z |
| FCCU_ERR | HIGH Z |
| CLKOUT_STANDBY | HIGH Z |
| GPIOs | HIGH Z |
| EXTAL | HIGH Z till extal is connected |
| XTAL | HIGH Z till xtal is connected |

1. The IO pad states are undefined until Supply rises sufficiently to enable the POR circuits.

### 4.5.3 Destructive Reset

This table mentions the Pin behavior when any Destructive Reset event is triggered.

Table 19. Destructive Reset

| Pins | ON Destructive Reset until SW comes up |
|---|---|
| RESET_B | LOW |
| TMS,TCK,TDI,TDO as JTAG | PULLED Values:<br>(TMS = HIGH TCK = LOW, TDI =HIGH)<br>TDO: HIGH Z |
| ETM_TRACE (Only on 17trace Pins) | HIGH Z |
| FCCU_ERR | HIGH Z |
| CLKOUT_STANDBY | Clkout expose if CLKOUT_STANDBY is configured to be Enabled<br>Else HIGH Z |
| GPIOs | HIGH Z |

### 4.5.4 Functional Reset including Functional entry sequence

This table mentions the Pin behavior when any functional Reset event is triggered. Also, when the FCCU reaction is configured as Functional Reset , then also the same behavior is achieved.

Table 20. Functional Reset including Functional entry sequence

| Pins | Functional Reset Entry | Reset until SW comes up |
|---|---|---|
| RESET_B | LOW | LOW |
| TMS,TCK,TDI,TDO as JTAG | TMS,TCK,TDI: PULLED Values if configured as JTAG<br>(TMS = HIGH TCK = LOW, TDI =HIGH)<br>HIGH Z if configured as GPIO<br>TDO: HIGH Z | PULLED Values<br>(TMS = HIGH TCK = LOW, TDI =HIGH)<br>TDO: HIGH Z |
| ETM_TRACE | Trace if MDMAPCTL[DBGRSTSLOWPAD] or MDMAPCTL[DBGRSTFASTPAD] is configured to be Enabled<br>Else HIGH Z | Trace if MDMAPCTL[DBGRSTSLOWPAD] or MDMAPCTL[DBGRSTFASTPAD] is configured to be Enabled<br>Else HIGH Z |
| FCCU_ERR | ERR from FCCU if UTEST_MISC[FCCU_EOUT_DEDICATED] is configured to be Enabled<br>Else HIGH Z | ERR from FCCU if UTEST_MISC[FCCU_EOUT_DEDICATED] is configured to be Enabled.<br>Else HIGH Z |
| CLKOUT_STANDBY | Clkout expose if DCMRWP1[CLKOUT_STANDBY] is configured to be Enabled<br>Else HIGH Z | Clkout expose if DCMRWP1[CLKOUT_STANDBY] is configured to be Enabled<br>Else HIGH Z |

*Table continues on the next page...*

Table 20. Functional Reset including Functional entry sequence (continued)

| Pins | Functional Reset Entry | Reset until SW comes up |
|---|---|---|
| GPIOs | HIGH Z | HIGH Z |

### 4.5.5 SELFTEST(MC_RGM.ERCTRL[ERASSERT] configured as 1)

Table 21. SELFTEST(MC_RGM.ERCTRL[ERASSERT] configured as 1)

| Pins | Selftest | Reset until SW comes up |
|---|---|---|
| RESET_B | LOW if configured as Reset Pin<br>High Z if configured as GPIO | LOW if configured as Reset Pin<br>High Z if configured as GPIO |
| TMS,TCK,TDI,TDO as JTAG | PULLED Values if configured as JTAG<br>(TMS = HIGH, TCK = LOW, TDI = HIGH)<br>TDO: HIGH Z | PULLED Values<br>(TMS = HIGH, TCK = LOW, TDI = HIGH)<br>TDO: HIGH Z |
| ETM_TRACE | Trace if MDMAPCTL[DBGRSTSLOWPAD] or MDMAPCTL[DBGRSTFASTPAD] is configured to be Enabled.<br>Else HIGH Z | Trace if MDMAPCTL[DBGRSTSLOWPAD] or MDMAPCTL[DBGRSTFASTPAD] is configured to be Enabled.<br>Else HIGH Z |
| FCCU_ERR | FCCU Error State if UTEST_MISC[FCCU_EOUT_DEDICATED] is Enabled and DCMRWD2[EOUT_STAT_DUR_STEST] is Enabled<br>Else HIGH Z | FCCU Error State if UTEST_MISC[FCCU_EOUT_DEDICATED] is Enabled and DCMRWD2[EOUT_STAT_DUR_STEST] is Enabled<br>Else HIGH Z |
| CLKOUT_STANDBY | Clkout expose if DCMRWP1[CLKOUT_STANDBY] is configured to be Enabled<br>Else HIGH Z | Clkout expose if DCMRWP1[CLKOUT_STANDBY] is configured to be Enabled<br>Else HIGH Z |
| GPIOs | HIGH Z | HIGH Z |

### 4.5.6 SELFTEST (RGM_ERCTRL[ERASSERT]=0 and RGM_FRBE[ST_DONE] =0)

Table 22. SELFTEST (RGM_ERCTRL[ERASSERT]=0 and RGM_FRBE[ST_DONE] =0)

| Pins | Selftest | Reset after Selftest until SW comes up |
|---|---|---|
| RESET_B | Same state as before selftest | Same state as before selftest |
| TMS,TCK,TDI,TDO as JTAG | Same state as before selftest | PULLED Values<br>(TMS = HIGH, TCK = LOW, TDI = HIGH)<br>TDO: No change |
| ETM_TRACE | Same state as before selftest | Same state as before selftest |

*Table continues on the next page...*

Table 22. SELFTEST (RGM_ERCTRL[ERASSERT]=0 and RGM_FRBE[ST_DONE] =0) (continued)

| Pins | Selftest | Reset after Selftest until SW comes up |
|---|---|---|
| FCCU_ERR | FCCU Error State if UTEST_MISC[FCCU_EOUT_DEDICATED] is Enabled and DCMRWD2[EOUT_STAT_DUR_STEST] is Enabled<br>Else No change | FCCU Error State if UTEST_MISC[FCCU_EOUT_DEDICATED] is Enabled and DCMRWD2[EOUT_STAT_DUR_STEST] is Enabled<br>Else No change |
| CLKOUT_STANDBY | Same state as before selftest | Same state as before selftest |
| GPIOs | Same state as before selftest | Same state as before selftest |

### 4.5.7 FCCU fault in RUN mode when the fault reaction is not Reset

**NOTE**

For any GPIO PAD If SMC=1 then the pad would retain it's state during the fault mode else if SMC=0 then the pad would become high Z during the fault mode.

Table 23. FCCU fault in RUN mode, fault reaction not Reset

| Pins | ON FCCU Fault |
|---|---|
| RESET_B | If RESET_B is assigned as a dedicated PAD then no change in the state of the PAD else if RESET_B is configured as GPIO then it would behave as per the SMC configuration.<br>• Case 1: If dcf_client_reset_pad_dedicated[reset pad dedicated] is configured as 1'b1, the pad functions as per the reset state machine.<br>• Case 2: If dcf_client_reset_pad_dedicated[reset pad dedicated] is configured as 1'b0 and SIUL2.MSCR5[SMC] is configured as 1'b1, the pad functions as per the configured protocol status, the protocol being configured via SIUL2.MSCR5[SSS].<br>• Case 3: If dcf_client_reset_pad_dedicated[reset pad dedicated] is configured as 1'b0 and SIUL2.MSCR5[SMC] is configured as 1'b0, the pin gets tristated (high-Z). |
| TMS, TCK, TDI, TDO as JTAG | If the PAD is configured as a JTAG PAD then no change and the pad would continue performing the operation else if it is a GPIO PAD then it would behave as per the SMC configuration.<br>• Case 1: If the pad is configured for JTAG mode and SMC bit of SIUL2's MSCR register of the corresponding pin is configured as 1'b1, the pad functions as per the JTAGC state machine or protocol.<br>• Case 2: If the pad is configured for non-JTAG mode and SMC bit of SIUL2's MSCR register of the corresponding pin is configured as 1'b0, the pins get tristated (high-Z). |
| ETM_TRACE | If the PAD is configured as a dedicated PAD then no change and the pad would continue performing the operation else if it is a GPIO PAD then it would behave as per the SMC configuration.<br>• Case 1: If the pad is configured for ETM_TRACE functionality and SMC bit of SIUL2's MSCR register of the corresponding pin is configured as 1'b1, the pad functions as per the ETM trace control logic.<br>• Case 2: If the pad is configured for non-ETM_TRACE mode and SMC bit of SIUL2's MSCR register of the corresponding pin is configured as 1'b0, the pins get tristated (high-Z). |

*Table continues on the next page...*

Table 23. FCCU fault in RUN mode, fault reaction not Reset (continued)

| Pins | ON FCCU Fault |
|---|---|
| FCCU_ERR | If the PAD is configured as a dedicated PAD by programming the UTEST_MISC[FCCU_EOUT_DEDICATED] as well as Eout indication is enabled then it would be indicating error state else if it is a GPIO PAD then it would behave as per the SMC configuration.<br>• Case 1: If dcf_client_utest_misc[FCCU_EOUT_DEDICATED] is configured as 1'b1, the FCCU_ERR pins indicate the error state.<br>• Case 2: If dcf_client_utest_misc[FCCU_EOUT_DEDICATED] is configured as 1'b0 and SMC bit of the SIUL2's corresponding MSCR register is configured as 1'b1, the pin retains its state.<br>• Case 3: If dcf_client_utest_misc[FCCU_EOUT_DEDICATED] is configured as 1'b0 and SIUL2.MSCR5[SMC] is configured as 1'b0, the pin gets tristated (high-Z). |
| CLKOUT_STANDBY | If the PAD is configured as a dedicated PAD then no change and the pad would continue performing the operation else if it is a GPIO PAD then it would behave as per the SMC configuration.<br>• Case 1: If the pad is configured for CLKOUT_STANDBY functionality and SMC bit of SIUL2's MSCR register of the corresponding pin is configured as 1'b1, the pin continues functioning as the CLKOUT_STANDBY pin.<br>• Case 2: If the SMC bit of SIUL2's MSCR register of the corresponding pin is configured as 1'b0, the pin gets tristated (high-Z). |
| GPIOs | If the PAD is configured as a dedicated PAD then no change and the pad would continue performing the operation else if it is a GPIO PAD then it would behave as per the SMC configuration.<br>• Case 1: If the SMC bit of SIUL2's MSCR register of the corresponding pin is configured as 1'b1, the pin continues its operation.<br>• Case 2: If the SMC bit of SIUL2's MSCR register of the corresponding pin is configured as 1'b0, the pins get tristated (high-Z). |

### 4.5.8 STANDBY ( DCMRWF1[STANDBY_IO_CONFIG] must be configured as 1)

- Please refer to Padkeeping Section for steps to configure the various pad states during standby

Table 24. STANDBY

| Pins | Standby until SW comes up |
|---|---|
| RESET_B | Alive / HighZ /LOW/HIGH (Depends on Software configuration before entering STANDBY) |
| TMS,TCK,TDI,TDO as JTAG | HighZ /LOW/HIGH (Depends on Software configuration before entering STANDBY) |
| ETM_TRACE | HighZ /LOW/HIGH (Depends on Software configuration before entering STANDBY) |
| FCUU_ERR | HighZ /LOW/HIGH (Depends on Software configuration before entering STANDBY) |
| CLKOUT_STANDBY | Alive / HighZ /LOW/HIGH (Depends on Software configuration before entering STANDBY) |
| GPIOs | HighZ /LOW/HIGH (Depends on Software configuration before entering STANDBY) |

## 4.6 Glossary

| | |
|---|---|
| **DSE** | Drive strength enable |
| **EXTAL** | External crystal input |
| **GPIO** | General purpose input/output |
| **IBE** | Input buffer enable |
| **INV** | Invert enable |
| **JTAG_TMS** | JTAG test mode select |
| **JTAG_TCK** | JTAG test clock input |
| **JTAG_TDI** | JTAG test data input |
| **JTAG_TDO** | JTAG test data output |
| **OBE** | Output buffer enable |
| **PUS** | Pullup and pulldown select |
| **PKE** | Pad keeping enable |
| **SMC** | Safe mode control |
| **SRC** | Slew rate control |
| **SSS** | Solid-state storage |
| **XTAL** | External crystal output |

# Chapter 5
# Cortex-M7 Overview

## 5.1 Introduction

Cortex-M7 is a high-performance embedded processor intended for deeply embedded applications that require fast interrupt response features. The configuration of the processor is based on little-endian format, and you must compile the execution testbench tests in this format too.

### 5.1.1 Features

Cortex-M7 provides:

- Low interrupt latency
- Low-cost debug
- Backwards compatibility with existing Cortex-M profile processors
- In-order superscalar pipeline
- Dual-issue support for load/load and load/store instruction pairs to multiple memory interfaces
- An MPU that you could configure to protect regions of memory
- A NVIC
- A debug and trace unit (CoreSight components)
- Floating-point arithmetic functionality, with support for single-precision arithmetic
- The ability to perform speculative load from any Normal type memory space through its AXIM bus, if D-cache is enabled
- Several memory interfaces that include:
  - Harvard architecture-based instruction and data caches, and an AXIM interface
  - A dedicated low-latency AHBP interface
  - A 64-bit AXI AMBA4 memory interface with a 8 KB instruction cache and an 8 KB data cache for efficient access to external resources. The instruction and data caches are ECC protected.
  - A 32-bit AHBS for interfacing with slaves such as DMA
  - 64-bit and 32-bit memory interfaces for the connection to local Tightly Coupled Memories called ITCM and DTCM.

### 5.1.2 Related information

For detailed information on:

- Cortex-M7 processor, see Arm Cortex-M7 Processor Technical Reference Manual.
- Cortex-M7 peripherals and control, see Arm Cortex-M7 Devices Generic User Guide.
- System memory map, see the memory map file attached to this document.

### 5.1.3 Buses, interconnects, and interfaces

This table discusses Cortex-M7 buses and their associated interconnects and interfaces.

Table 25. Buses and associated information

| Bus name | Description |
|---|---|
| AXIM | Using the XHB400 module, this bus is first translated to the AHB-Lite bus that interfaces with the AXBS crossbar switch providing high-bandwidth access to on-chip memories and peripherals. |
| AHBP | This bus connects to the AXBS crossbar switch providing high-bandwidth access to on-chip peripherals. |
| PPB | This bus provides access to these modules:<br>• Arm modules such as NVIC, ETM, ITM, DWT, and ROM tables<br>• Miscellaneous Control Module (MCM) |

**NOTE**

MCXE31 AHBP bus is enabled after reset. Therefore, accesses of all cores to on-chip peripherals are performed exclusively through this bus.

### 5.1.4 Core configuration

This table describes Cortex-M7 parameter settings.

Table 26. Core configuration

| Parameter | Configuration[1] |
|---|---|
| FPU | Single precision |
| DSP extension instructions | • Single cycle 16/32-bit MAC<br>• Single cycle dual 16-bit MAC<br>• 8/16-bit SIMD arithmetic<br>• Hardware divide (2-12 cycles) |
| Armv8-M security extensions | Not implemented |
| I-cache | Implemented |
| D-cache | Implemented |
| Caches ECC | Implemented |
| On core MPU region | 16 |
| Number of IRQs | 240 |
| IRQ priority width configuration | 4 (16 interrupt priority levels) |
| Debug (breakpoint/watchpoint) | Full comparator set: 4 DWT and 8 FPB comparators |
| Internal trace support | ITM and DWT trace functionality implemented |
| ETM support | Instruction and data ETM interface implemented |
| CTI | Implemented |
| WIC support | Not implemented |
| Dual-redundant core (lock-step) CPU functionality | Not implemented |

*Table continues on the next page...*

Table 26. Core configuration (continued)

| Parameter | Configuration[1] |
|---|---|
| RAR | All asynchronously reset |
| I-cache size | 8 KB |
| D-cache size | 8 KB |
| ITCM | 32 KB for CM7_0 |
| DTCM0 | 32 KB for CM7_0 |
| DTCM1 | 32 KB for CM7_0 |

1. Armv7-M (Harvard architecture), six-stage superscalar plus branch prediction

## 5.2 Speculative accesses

The Arm Cortex M7 processor can issue speculative read accesses that may access any location within the complete memory address range. This behavior can be controlled, but not disabled. Corresponding effects must be considered for a proper operation of your system.

Speculative accesses do not cause any processor faults. The processor is aware whether an access is speculative, and ignores any error response signaled by the system due to the speculative access. However, the system that is integrating the processor cannot distinguish speculative accesses from non-speculative accesses.

Addresses used by speculative accesses are not validated against the memory map of the device, and may attempt to also access non-existing memory regions or hardware elements having side effects. For details about corresponding behavior of the Arm Cortex M7 processor see Related information. Important processing aspects are listed in section “Memory Model” within the Generic User Guide.

Speculative accesses can result in improved performance when the related memory regions are properly characterized. It is imperative to properly setup the attributes within the Memory Protection Unit (MPU) to avoid any unwanted impact of a speculative access. Examples for possible, corresponding issues are usually the result of side effects unknown to the processor:

- Speculative access to an uninitialized RAM memory location that causes a double bit error {the related fault processing by the FCCU is performed independently from the processor}.
- Speculative access to a peripheral that causes an unwanted operation; for example, a FIFO read {the corresponding data may be removed from the FIFO without being processed}.
- Speculative access to a peripheral that is not clocked, powered down, or cannot respond {the access may not be terminated, resulting an access that is stalled, system blockage}.
- Speculative access to an address range that causes an unexpected error being reported; for example, a read-while-write error of the embedded flash (indicated by setting MCRS[RWE]) during a flash erase or program operation {which may hide real errors}.

Speculative accesses can be controlled by a proper assignment of memory regions within the MPU:

- Speculative instruction fetches are never made to memory addresses in an Execute Never region.
- Speculative data reads are never made to memory addresses marked as non-accessible in the MPU.
- Speculative cache line-fills are never made to non-cacheable memory addresses.
- Speculative data reads and speculative cache line-fills are never made to memory addresses in a region having a Device or Strongly-ordered attribute.
- Speculative reads are never made on the AHBP interface.
- Speculative writes are never made.

Memory regions mapped to a TCM are always treated as Normal Memory (equivalent to the MPU attribute) and are therefore always subject to speculation. Related issues can be avoided by properly initializing any TCM memory before a corresponding access may occur.

When no speculative accesses should be initiated to a memory region, it is recommended to set all of the following attributes within the MPU for this region: Device or Strongly-ordered, and Execute Never. These attributes are often also used for address ranges associated with peripherals.

Unwanted processing of side effects caused by a speculative access can also be inhibited by disabling the related events while a speculative access may occur. As an example, the FCCU can be enabled after the ECC of the RAM memories has been initialized. As a second example, read accesses to a Flash block can be inhibited by configuring an MPU region while it is being erased.

## 5.3 Debug facilities

This chip has extensive debug capabilities such as run control and tracing. It includes the standard Arm debug port that supports the JTAG and SWD interfaces.

## 5.4 Vector fetch behavior on Cortex-M7

In Cortex-M7, the vector fetches are looked up into the I-Cache. if the vector table is located in a region of memory that is cacheable, any load or store to the vector must be treated as self-modifying code and cache maintenance instructions should be used to synchronize the updates to the data and instruction caches. The Cortex-M7 Device Generic User Guide chapter 'Cache maintenance design hints and tips' specifies a recommendation for synchronization of the D-Cache and I-Cache.

If cache maintenance is to be avoided each time when the vector table gets updated, then the vector table must be allocated in the ITCM or DTCM, as those are non-cacheable regions. Alternatively, the I-Cache must be enabled after the vector table has been initialized.

## 5.5 TCM retry

TCM retry is disabled and software should disable the TCM retry bit at startup by programming the relevant core's CM7_ITCMCR[RETEN] and CM7_DTCMR[RETEN] fields to disable state.

## 5.6 Glossary

| | |
|---|---|
| **AHBP** | AHB-lite peripheral |
| **AHBS** | AHB-slave port |
| **AXIM** | Advanced extensible interface master |
| **DWT** | Data watchpoint and trace unit |
| **ETM** | Embedded trace macrocell |
| **FPU** | Floating point unit |
| **FPB** | Flash patch and breakpoint |
| **ITM** | Instrumentation trace macrocell |
| **MAC** | Multiplier accumulator (refers to a multiplier accumulator unit as well as multiplier accumulator operation) |
| **NVIC** | Nested vectored interrupt controller |
| **RCCU** | Redundancy control checking unit |
| **RAR** | Reset-all-registers |
| **SIMD** | Single instruction multiple data |

# Chapter 6
# Miscellaneous Control Module (MCM)

## 6.1 Chip-specific MCM information

### 6.1.1 MCM instances and configuration

This chip has one MCM instance.

Table 27. Memories

| Memory / Variant | MCXE315/MCXE316/MCXE317/MCXE31B (single core) |
|---|---|
| DTCM | 64 KB |
| ITCM | 32 KB |
| Icache | 8 KB |
| Dcache | 8 KB |

## 6.2 Overview

MCM provides miscellaneous control functions and contains local memory descriptors for the Cortex-M7 core. For more information about core-related registers, see the Cortex-M7 core overview chapter.

**NOTE**

The terminology in this chapter has been updated to align with Arm's AMBA AHB Protocol Specification, as shown in the table below.

Table 28. Updated terms

| Updated term | Deprecated term |
|---|---|
| Manager | Master |
| Subordinate | Slave |

### 6.2.1 Features

The MCM includes the following features:

- Program-visible information on the platform configuration and revision
- Floating Point Exception monitor and interrupt control
- Local memory descriptors:
  - ITCM
  - D0TCM
  - D1TCM
  - I-cache
  - D-cache

## 6.3 Functional description

### 6.3.1 Interrupts

MCM generates an interrupt if any of the following are true:

- FPU input denormal interrupt is enabled (FIDCE) and an input is denormalized (FIDC).
- FPU inexact interrupt is enabled (FIXCE) and a number is inexact (FIXC).
- FPU underflow interrupt is enabled (FUFCE) and an underflow occurs (FUFC).
- FPU overflow interrupt is enabled (FOFCE) and an overflow occurs (FOFC).
- FPU divide-by-zero interrupt is enabled (FDZCE) and a divide-by-zero occurs (FDZC).
- FPU invalid operation interrupt is enabled (FIOCE) and an invalid operation occurs (FIOC).
- Write abort interrupt is enabled (WABE) and a write abort occurs (CORTEX-M7 WABORTS INDICATOR).

#### Determining interrupt source

To determine the exact source of the interrupt for Cortex-M7 core, qualify the interrupt status flags with the corresponding interrupt enable fields.

- MCM_ISCR[31:16] && MCM_ISCR[15:0]
- Search the result for asserted flags, which indicate the exact interrupt sources.

## 6.4 Memory map and register descriptions

The memory map and register descriptions below describe the registers using byte addresses.

**NOTE**

The following actions result in bus errors:

- Writing to read-only registers at 0x0.
- Reading from or writing to an address from offset 480h and above.
- Accessing any MCM register while in User mode. These registers are only accessible while in Supervisor mode.

### 6.4.1 MCM register descriptions

#### 6.4.1.1 MCM memory map

MCM_0 base address: E008_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | SoC-defined Platform Revision (PLREV) | 16 | R | 0000h |
| 2h | Processor Core Type (PCT) | 16 | R | AC70h |
| Ch | Core Platform Control (CPCR) | 32 | RW | 0000_0200h |
| 10h | Interrupt Status and Control (ISCR) | 32 | RW | 0000_0000h |
| 30h | Process Identifier (PID) | 8 | RW | 00h |
| 400h | Local Memory Descriptor 0 (LMEM_DESC_0) | 32 | R | 8606_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 404h - 408h | Local Memory Descriptor a (LMEM_DESC_1 - LMEM_DESC_2) | 32 | R | 8604_2000h |
| 40Ch | Local Memory Descriptor 3 (LMEM_DESC_3) | 32 | R | 8426_4000h |
| 410h | Local Memory Descriptor 4 (LMEM_DESC_4) | 32 | R | 8444_6000h |

### 6.4.1.2 SoC-defined Platform Revision (PLREV)

#### Offset

| Register | Offset |
|---|---|
| PLREV | 0h |

#### Function

Specifies a chip-defined platform revision number. A platform input signal defines the state of this register; it can only be read from the IPS programming model. Any attempted write is ignored.

#### Diagram

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

#### Fields

| Field | Function |
|---|---|
| 15-0<br>PLREV | Defines the software-visible revision number, specified by a platform input signal. |

### 6.4.1.3 Processor Core Type (PCT)

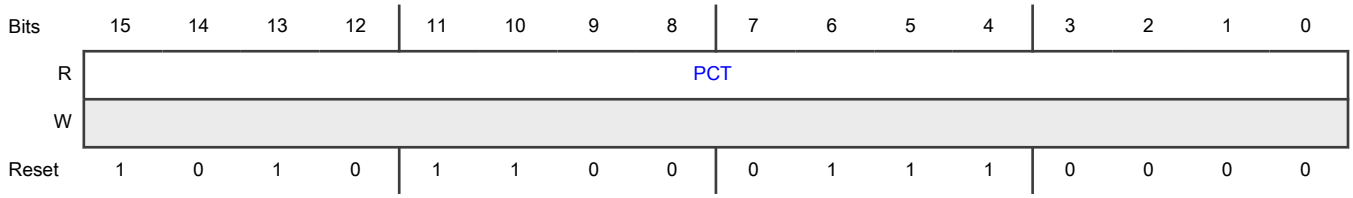
#### Offset

| Register | Offset |
|---|---|
| PCT | 2h |

### Function

Specifies the architecture of the processor core within the platform on the chip. A module input signal defines the state of this register, which can only be read from the IPS programming model. Any attempted write is ignored.

### Diagram




### Fields

| Field | Function |
|---|---|
| 15-0<br>PCT | Core Complex Identifier<br>Identifies the core complex. This MCM design supports the Arm Cortex M7 core.<br>1010_1100_0111_0000b - Arm Cortex-M7 |

## 6.4.1.4 Core Platform Control (CPCR)

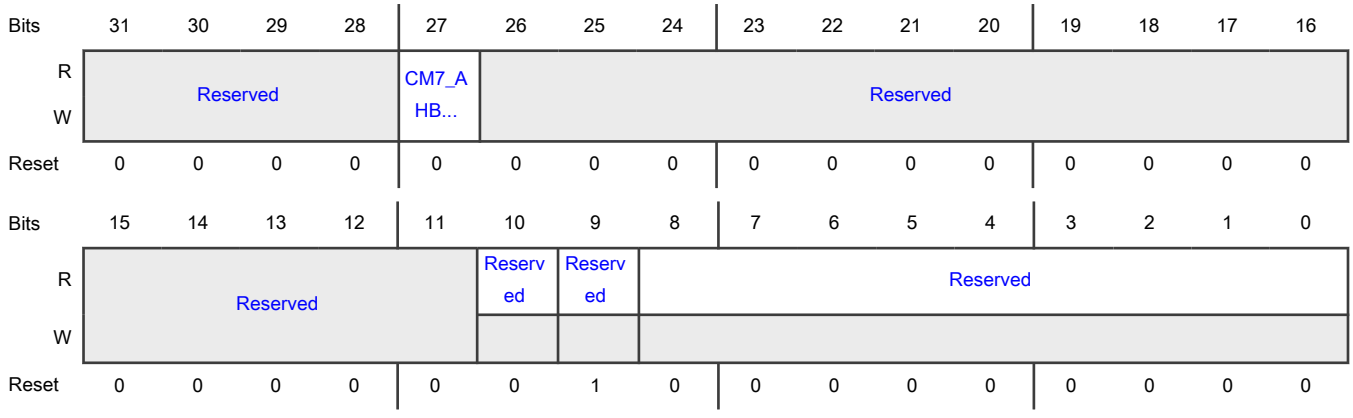
### Offset

| Register | Offset |
|---|---|
| CPCR | Ch |

### Function

Defines the arbitration and protection schemes for the two system RAM arrays.

### Diagram



**Fields**

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27<br>CM7_AHBSPRI | AHB Subordinate Priority<br>Indicates the access priority on the AHBS port of the Cortex-M7.<br>**NOTE**<br>This setting has no effect unless enabled by AHBSCR[CTL][1] of the Cortex-M7 core.<br>0b - Uses a round-robin arbitration scheme<br>1b - AHB-subordinate access has priority over a core access |
| 26-11<br>— | Reserved |
| 10<br>— | Reserved |
| 9<br>— | Reserved |
| 8-0<br>— | Reserved |

1. For more information see Cortex-M7 documentation: Arm Cortex-M7 Processor Technical Reference Manual at www.arm.com.

### 6.4.1.5 Interrupt Status and Control (ISCR)

**Offset**

| Register | Offset |
|---|---|
| ISCR | 10h |

**Function**

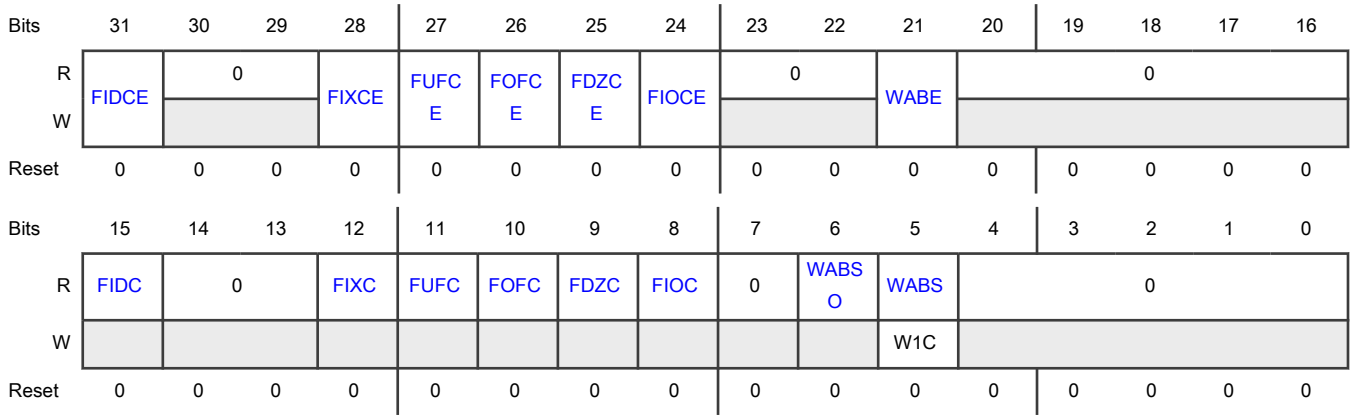
Defines the configuration and reports the status for a number of core-related interrupt exception conditions. It includes:

- Enable and status fields associated with the core's floating-point exceptions
- Bus errors associated with the core's cache write buffer

The individual event indicators are first qualified with their exception enables, and then logically summed to form an interrupt request sent to the core's NVIC.

Bits 15-8 are read-only indicator flags based on the processor's FPSCR register. Attempted writes to these fields are ignored. When these flags are 1, they retain this value until software clears the corresponding FPSCR field. For more information see Cortex-M7 documentation: Arm Cortex-M7 Processor Technical Reference Manual at www.arm.com.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>FIDCE | FPU Input Denormal Interrupt Enable<br>0b - Disable<br>1b - Enable |
| 30-29<br>— | Reserved |
| 28<br>FIXCE | FPU Inexact Interrupt Enable<br>0b - Disable<br>1b - Enable |
| 27<br>FUFCE | FPU Underflow Interrupt Enable<br>0b - Disable<br>1b - Enable |
| 26<br>FOFCE | FPU Overflow Interrupt Enable<br>0b - Disable<br>1b - Enable |
| 25<br>FDZCE | FPU Divide-by-Zero Interrupt Enable<br>0b - Disable<br>1b - Enable |
| 24<br>FIOCE | FPU Invalid Operation Interrupt Enable<br>0b - Disable<br>1b - Enable |
| 23-22 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 21<br>WABE | TCM Write Abort Interrupt Enable<br>0b - Disable<br>1b - Enable |
| 20-16<br>— | Reserved |
| 15<br>FIDC | FPU Input Denormal Interrupt Status<br>Indicates that an input denormalized number has been detected in the processor's FPU. This field is a copy of the core's FPSCR[IDC] field. When this field is 1, it retains this value until software clears the FPSCR[IDC] field.<br>0b - No interrupt<br>1b - Interrupt occurred |
| 14-13<br>— | Reserved |
| 12<br>FIXC | FPU Inexact Interrupt Status<br>Indicates that an inexact number has been detected in the processor's FPU. This field is a copy of the core's FPSCR[IXC] field. When this field is 1, it retains this value until software clears the FPSCR[IXC] field.<br>0b - No interrupt<br>1b - Interrupt occurred |
| 11<br>FUFC | FPU Underflow Interrupt Status<br>Indicates that an underflow has been detected in the processor's FPU. This field is a copy of the core's FPSCR[UFC] field. When this field is 1, it retains this value until software clears the FPSCR[UFC] field.<br>0b - No interrupt<br>1b - Interrupt occurred |
| 10<br>FOFC | FPU Overflow Interrupt Status<br>Indicates that an overflow has been detected in the processor's FPU. This field is a copy of the core's FPSCR[OFC] field. When this field is 1, it retains this value until software clears the FPSCR[OFC] field.<br>0b - No interrupt<br>1b - Interrupt occurred |
| 9<br>FDZC | FPU Divide-by-Zero Interrupt Status<br>Indicates that a divide-by-zero operation has been detected in the processor's FPU. This field is a copy of the core's FPSCR[DZC] field. When this field is 1, it retains this value until software clears the FPSCR[DZC] field. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No interrupt<br>1b - Interrupt occurred |
| 8<br>FIOC | FPU Invalid Operation Interrupt Status<br>Indicates that an illegal operation has been detected in the processor's FPU. This field is a copy of the core's FPSCR[IOC] field. When this field is 1, it retains this value until software clears the FPSCR[IOC] field.<br>0b - No interrupt<br>1b - Interrupt occurred |
| 7<br>— | Reserved |
| 6<br>WABSO | Write Abort on Subordinate Overrun<br>The overrun conditions are reported only if WABE=1.<br>0b - No write abort overrun<br>1b - Write abort overrun occurred |
| 5<br>WABS | Write Abort on Subordinate<br>Indicates when a write abort has occurred on the AHBS interface.<br>0b - No write abort occurred on AHBS interface<br>1b - Write abort occurred on AHBS interface |
| 4-0<br>— | Reserved |

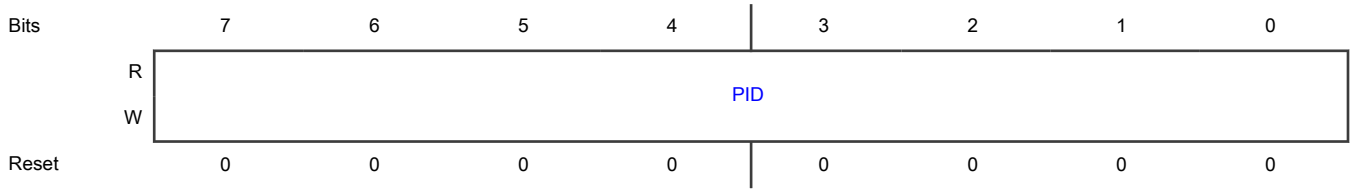
### 6.4.1.6 Process Identifier (PID)

**Offset**

| Register | Offset |
|---|---|
| PID | 30h |

**Function**
Contains the CPU process ID.

### Diagram




### Fields

| Field | Function |
|---|---|
| 7-0<br>PID | Process Identifier<br>Identifies the CPU process. |

## 6.4.1.7 Local Memory Descriptor 0 (LMEM_DESC_0)

### Offset

| Register | Offset |
|---|---|
| LMEM_DESC_0 | 400h |

### Function

**NOTE**

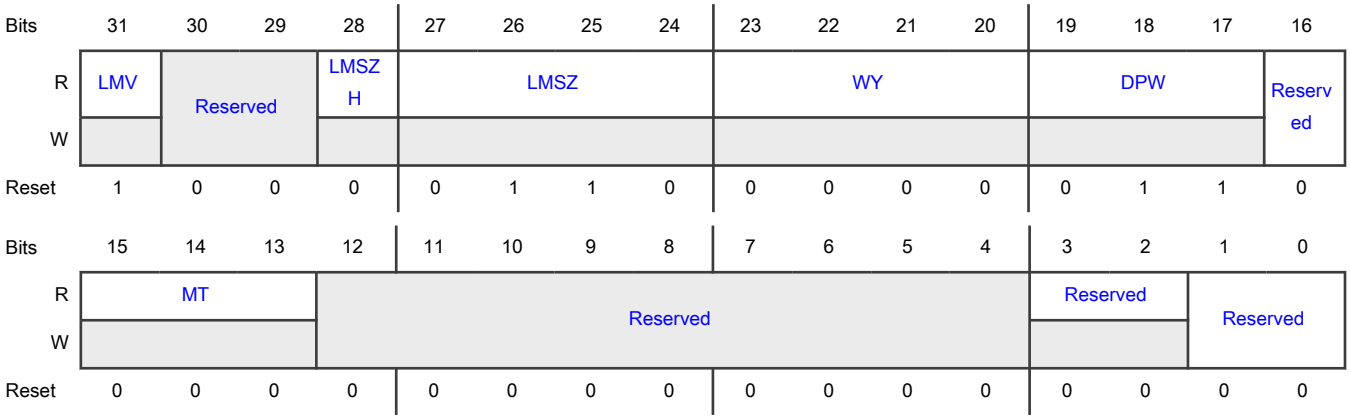
The DESC_a registers map to the LMEMs in this way:

- DESC_0: ITCM
- DESC_1: D0TCM
- DESC_2: D1TCM
- DESC_3: I-cache
- DESC_4: D-cache

**NOTE**

You can read and write to the reserved fields (instead of read as zero and write ignored). Writing to any of these fields has no functional impact.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LMV | Local Memory Valid<br>This read-only field defines the validity (presence) of the local memory.<br>0b - LMEM*n* not present<br>1b - LMEM*n* present |
| 30-29<br>— | Reserved |
| 28<br>LMSZH | LMEM Size Hole<br>Used for local memories that are not fully populated (that is, local memories that include a memory "hole" in the upper 25 % of the address range).<br>0b - LMEM*n* is a power-of-2 capacity<br>1b - LMEM*n* is not a power-of-2, with capacity of 0.75 × LMSZ |
| 27-24<br>LMSZ | Local Memory Size<br>0000b - 0 KB<br>0001b - 1 KB<br>0010b - 2 KB<br>0011b - 4 KB<br>0100b - 8 KB<br>0101b - 16 KB<br>0110b - 32 KB<br>0111b - 64 KB<br>1000b - 128 KB<br>1001b - 256 KB |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1010b - 512 KB<br>1011b - 1024 KB<br>1100b - 2048 KB<br>1101b - 4096 KB<br>1110b - 8192 KB<br>1111b - 16384 KB |
| 23-20<br>WY | Level 1 Cache Ways<br>0000b - No cache<br>0010b - 2-way set associative<br>0100b - 4-way set associative |
| 19-17<br>DPW | Data Path Width<br>Defines the LMEM*n* data path width, which is the width of the local memory.<br>000b-001b - Reserved<br>010b - 32 bits<br>011b - 64 bits<br>100b-111b - Reserved |
| 16<br>— | Reserved |
| 15-13<br>MT | Memory Type<br>000b - ITCM<br>001b - DTCM<br>010b - I-cache<br>011b - D-cache |
| 12-4<br>— | Reserved |
| 3-2<br>— | Reserved |
| 1-0<br>— | Reserved |

### 6.4.1.8 Local Memory Descriptor a (LMEM_DESC_1 - LMEM_DESC_2)

#### Offset

| Register | Offset |
|---|---|
| LMEM_DESC_1 | 404h |
| LMEM_DESC_2 | 408h |

#### Function

**NOTE**

The DESC_a registers map to the LMEMs in this way:

- DESC_0: ITCM
- DESC_1: D0TCM
- DESC_2: D1TCM
- DESC_3: I-cache
- DESC_4: D-cache

**NOTE**

You can read and write to the reserved fields (instead of read as zero and write ignored). Writing to any of these fields has no functional impact.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LMV | Reserved | | LMSZH | LMSZ | | | | WY | | | | DPW | | | Reserved |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

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

#### Fields

| Field | Function |
|---|---|
| 31<br>LMV | Local Memory Valid<br>This read-only field defines the validity (presence) of the local memory.<br>0b - LMEM*n* not present<br>1b - LMEM*n* present |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 30-29<br>— | Reserved |
| 28<br>LMSZH | LMEM Size Hole<br>Used for local memories that are not fully populated (that is, local memories that include a memory "hole" in the upper 25 % of the address range).<br>0b - LMEM*n* is a power-of-2 capacity<br>1b - LMEM*n* is not a power-of-2, with capacity of 0.75 × LMSZ |
| 27-24<br>LMSZ | Local Memory Size<br>0000b - 0 KB<br>0001b - 1 KB<br>0010b - 2 KB<br>0011b - 4 KB<br>0100b - 8 KB<br>0101b - 16 KB<br>0110b - 32 KB<br>0111b - 64 KB<br>1000b - 128 KB<br>1001b - 256 KB<br>1010b - 512 KB<br>1011b - 1024 KB<br>1100b - 2048 KB<br>1101b - 4096 KB<br>1110b - 8192 KB<br>1111b - 16384 KB |
| 23-20<br>WY | Level 1 Cache Ways<br>0000b - No cache<br>0010b - 2-way set associative<br>0100b - 4-way set associative |
| 19-17<br>DPW | Data Path Width<br>Defines the LMEM*n* data path width, which is the width of the local memory.<br>000b-001b - Reserved<br>010b - 32 bits |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 011b - 64 bits<br>100b-111b - Reserved |
| 16<br>— | Reserved |
| 15-13<br>MT | Memory Type<br>000b - ITCM<br>001b - DTCM<br>010b - I-cache<br>011b - D-cache |
| 12-4<br>— | Reserved |
| 3-2<br>— | Reserved |
| 1-0<br>— | Reserved |

#### 6.4.1.9 Local Memory Descriptor 3 (LMEM_DESC_3)

**Offset**

| Register | Offset |
|---|---|
| LMEM_DESC_3 | 40Ch |

**Function**

**NOTE**

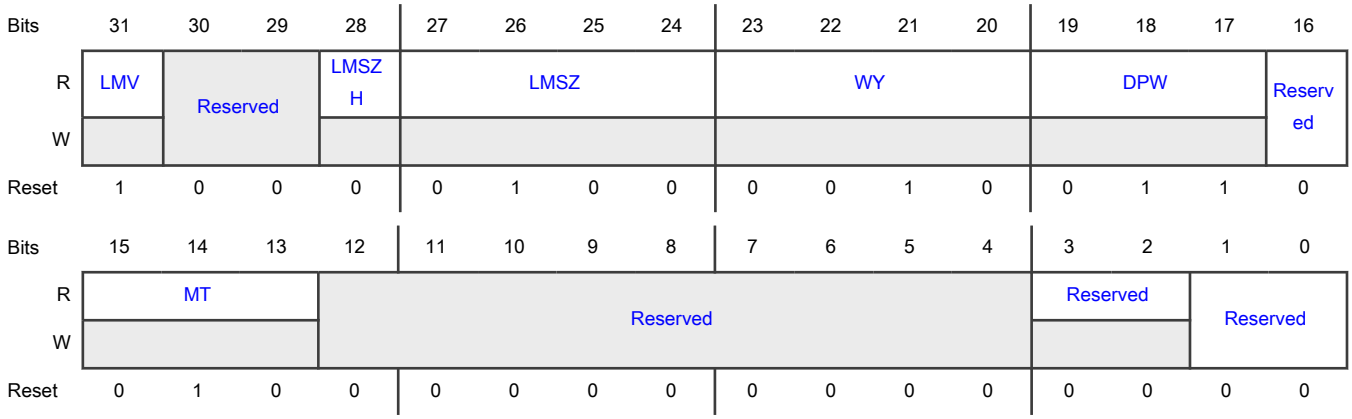
The DESC_a registers map to the LMEMs in this way:

- DESC_0: ITCM
- DESC_1: D0TCM
- DESC_2: D1TCM
- DESC_3: I-cache
- DESC_4: D-cache

**NOTE**

You can read and write to the reserved fields (instead of read as zero and write ignored). Writing to any of these fields has no functional impact.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LMV | Local Memory Valid<br>This read-only field defines the validity (presence) of the local memory.<br>0b - LMEM*n* not present<br>1b - LMEM*n* present |
| 30-29<br>— | Reserved |
| 28<br>LMSZH | LMEM Size Hole<br>Used for local memories that are not fully populated (that is, local memories that include a memory "hole" in the upper 25 % of the address range).<br>0b - LMEM*n* is a power-of-2 capacity<br>1b - LMEM*n* is not a power-of-2, with capacity of 0.75 × LMSZ |
| 27-24<br>LMSZ | Local Memory Size<br>0000b - 0 KB<br>0001b - 1 KB<br>0010b - 2 KB<br>0011b - 4 KB<br>0100b - 8 KB<br>0101b - 16 KB<br>0110b - 32 KB<br>0111b - 64 KB<br>1000b - 128 KB<br>1001b - 256 KB |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1010b - 512 KB<br>1011b - 1024 KB<br>1100b - 2048 KB<br>1101b - 4096 KB<br>1110b - 8192 KB<br>1111b - 16384 KB |
| 23-20<br>WY | Level 1 Cache Ways<br>0000b - No cache<br>0010b - 2-way set associative<br>0100b - 4-way set associative |
| 19-17<br>DPW | Data Path Width<br>Defines the LMEM*n* data path width, which is the width of the local memory.<br>000b-001b - Reserved<br>010b - 32 bits<br>011b - 64 bits<br>100b-111b - Reserved |
| 16<br>— | Reserved |
| 15-13<br>MT | Memory Type<br>000b - ITCM<br>001b - DTCM<br>010b - I-cache<br>011b - D-cache |
| 12-4<br>— | Reserved |
| 3-2<br>— | Reserved |
| 1-0<br>— | Reserved |

### 6.4.1.10 Local Memory Descriptor 4 (LMEM_DESC_4)

#### Offset

| Register | Offset |
|---|---|
| LMEM_DESC_4 | 410h |

#### Function

**NOTE**

The DESC_a registers map to the LMEMs in this way:

- DESC_0: ITCM
- DESC_1: D0TCM
- DESC_2: D1TCM
- DESC_3: I-cache
- DESC_4: D-cache

**NOTE**

You can read and write to the reserved fields (instead of read as zero and write ignored). Writing to any of these fields has no functional impact.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LMV | Reserved | | LMSZH | LMSZ | | | | WY | | | | DPW | | | Reserved |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |

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

#### Fields

| Field | Function |
|---|---|
| 31<br>LMV | Local Memory Valid<br>This read-only field defines the validity (presence) of the local memory.<br>0b - LMEM*n* not present<br>1b - LMEM*n* present |
| 30-29 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 28<br>LMSZH | LMEM Size Hole<br>Used for local memories that are not fully populated (that is, local memories that include a memory "hole" in the upper 25 % of the address range).<br>0b - LMEM*n* is a power-of-2 capacity<br>1b - LMEM*n* is not a power-of-2, with capacity of 0.75 × LMSZ |
| 27-24<br>LMSZ | Local Memory Size<br>0000b - 0 KB<br>0001b - 1 KB<br>0010b - 2 KB<br>0011b - 4 KB<br>0100b - 8 KB<br>0101b - 16 KB<br>0110b - 32 KB<br>0111b - 64 KB<br>1000b - 128 KB<br>1001b - 256 KB<br>1010b - 512 KB<br>1011b - 1024 KB<br>1100b - 2048 KB<br>1101b - 4096 KB<br>1110b - 8192 KB<br>1111b - 16384 KB |
| 23-20<br>WY | Level 1 Cache Ways<br>0000b - No cache<br>0010b - 2-way set associative<br>0100b - 4-way set associative |
| 19-17<br>DPW | Data Path Width<br>Defines the LMEM*n* data path width, which is the width of the local memory.<br>000b-001b - Reserved<br>010b - 32 bits<br>011b - 64 bits<br>100b-111b - Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 16<br>— | Reserved |
| 15-13<br>MT | Memory Type<br>000b - ITCM<br>001b - DTCM<br>010b - I-cache<br>011b - D-cache |
| 12-4<br>— | Reserved |
| 3-2<br>— | Reserved |
| 1-0<br>— | Reserved |

## 6.5 Glossary

| | |
|---|---|
| **ITCM** | Instruction Tightly-Coupled Memory |
| **DTCM** | Data Tightly Coupled Memory |
| **I-cache** | Instruction Cache Memory |
| **D-cache** | Data Cache Memory |

# Chapter 7
# Miscellaneous System Control Module (MSCM)

## 7.1 Chip-specific MSCM information

### 7.1.1 MSCM instance

This chip has one MSCM instance.

**NOTE**

The XN_CTRL register is used to restrict execution from SRAM, including TCMs and their backdoors, which will be permanent until next device reset, while still allowing data R/W.

XN_CTRL register is reserved for MCXE31B.

## 7.2 Introduction

MSCM contains registers for:

- CPU configuration
- On-chip memory control
- Interrupt router control
- Message-based interrupt configuration
- Virtual management

### 7.2.1 Features

- Software-accessible processor core configuration information
- Support for interrupt router control
- Support for message-based interrupt configuration

## 7.3 Functional description

MSCM provides information of the system cores and can identify the core that is running currently.

### 7.3.1 MSI routing

MSIs are interrupts that are indirectly broadcast to a target core by writing configuration bits in MSCM. These MSIs can be initiated by one core targeting another core in the system (known as core-to-core interrupts). These MSIs are initiated via writes to Interrupt Router CPn Interrupt Generation (IRCP0IGR0 - IRCP1IGR3) and managed through Interrupt Router CPn Interrupt Status (IRCP0ISR0 - IRCP1ISR3), where *n* indicates the logical core number (0-1) and *m* represents the interrupt number (0-3). CM7_0 core can support up to four outstanding core-to-core interrupts.

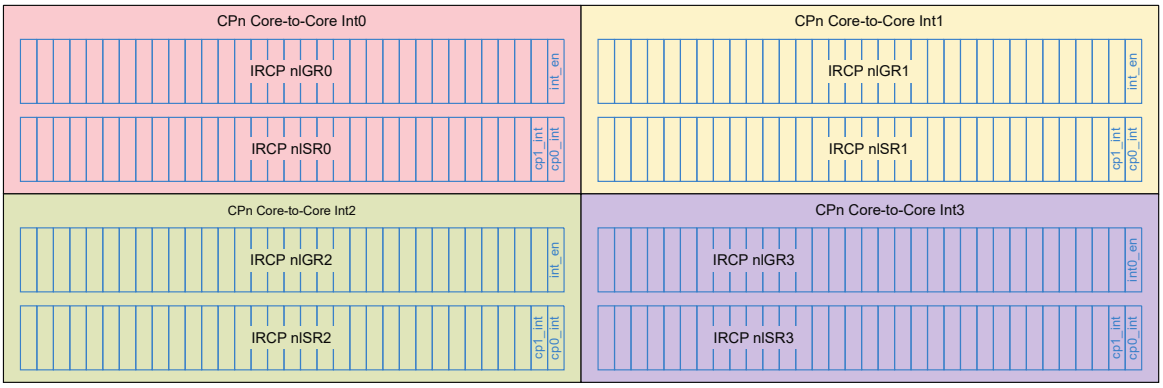



Figure 11. IRCP*n*IGR*m*/IRCP*n*ISR*m* pairs for one core

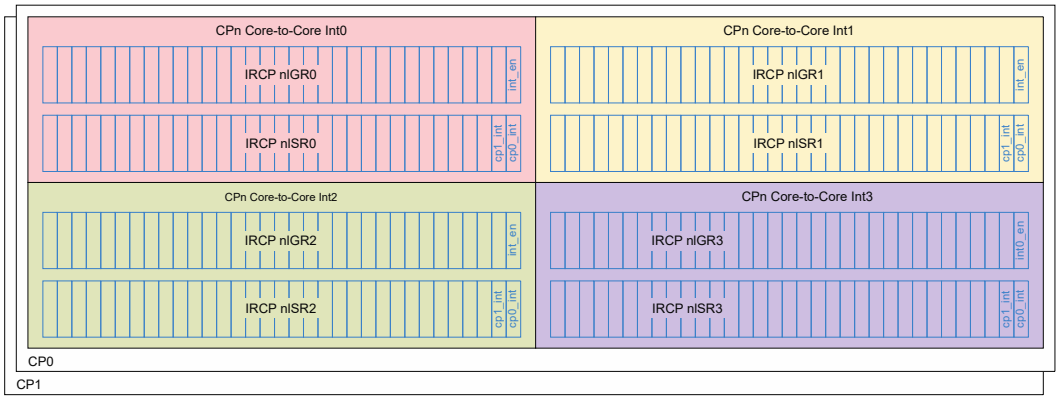



Figure 12. IRCP*n*IGR*m*/IRCP*n*ISR*m* pairs per core

### 7.3.1.1 Core-to-core MSIs

The next figure depicts the sequence for initiating a core-to-core MSI, in which *m* represents the initiating core, *n* represents the target core, and *x* indicates the MSI number. CP*m* writes to IRCP*n*IGR*x* to initiate an MSI. The outstanding MSI that CP*m* initiates, targeting CP*n*, is reflected in the corresponding bit-mapped field in IRCP*n*ISR*x*.

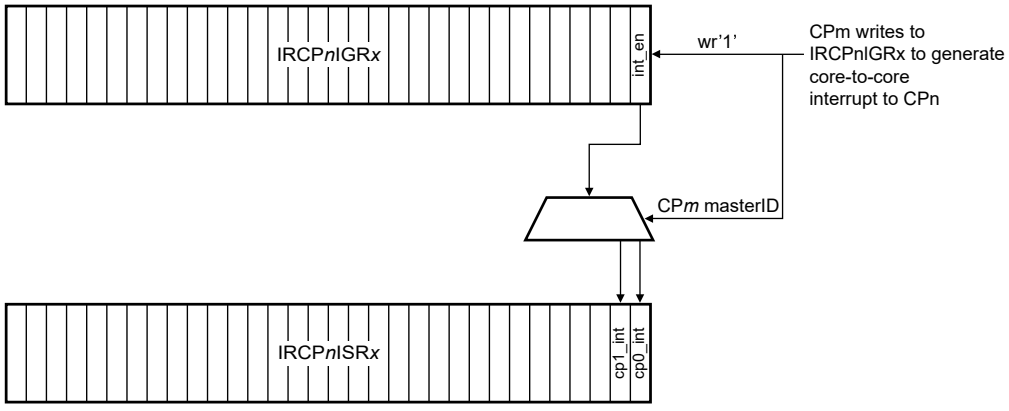



Figure 13. Initiating a core-to-core MSI via IRCP*n*IGR*x*/IRCP*n*ISR*x*

## 7.3.2 Interrupt steering and semaphores

#### 7.3.2.1 Interrupt handling overview

The interrupt handling mechanisms of the CM7 cores are very similar. These cores have a NVIC tightly coupled to the processor core. The real-time performance of the cores means the NVIC directly provides an appropriate interrupt vector, in the form of the starting instruction address for the interrupt service routine, to the core. These core architectural features directly translate into a faster ISR entry and exit capabilities, coupled with an improved runtime performance. See the Arm modules and Arm core technical reference manuals for details.

In this architecture, a total of 240 IRQs are supported, where this parameter is defined by the realistic limits of the NVIC implementation, both in terms of silicon size and supported frequency of operation. These 240 IRQs are split into a total of four directed requests and 236 shared peripheral requests. Unless noted otherwise, let the directed requests be defined as IRQ[3:0] and the shared peripheral requests as IRQ[239:4]. See the interrupt map file attached to this document for details.

#### 7.3.2.2 MSCM interrupt router functional description

As described in MSCM register descriptions, the interrupt routing registers enable the steering of requests to the processor cores.

## 7.4 Memory map and register definition

**NOTE**

The CP1 registers shown in the MSCM memory map section (from offset value 220h to 23Ch) are unavailable for the MCXE315, MCXE316, and MCXE317 variants.

### 7.4.1 Core configuration registers

These read-only registers contain data that defines the core setup for this chip. You can access the registers using 32-bit read references; other access sizes terminate with an error. Attempted write accesses to the read-only core configuration registers also terminate with an error.

The core configuration portion of the MSCM programming model map is organized based on the logical core number, and not on any type of physical port number. The following table shows how the configuration is partitioned.

**Table 29. MSCM core configuration partitioning**

| Offset address range | Purpose |
|---|---|
| 0h–018h | Defines the generic core x configuration information. Only the CM7 cores on the chip can access this region in either User or Privileged mode; reads by non-core bus masters (including the debugger) are treated as read as zero (RAZ) accesses. Write attempts are not permitted and terminate with a system bus error. |
| 020h–038h | Defines the configuration information for core 0 (CP0). Any bus master can access this region in either User or Privileged mode. Write attempts are not permitted and terminate with a system bus error. |

**NOTE**

Attempted accesses to reserved locations are not permitted and terminate with a system bus error.

### 7.4.2 Shared peripheral interrupt (SPI) routing

The SPI router portion of MSCM provides a set of memory-mapped registers defining the interrupt routing for all the SPIs on this chip.

### 7.4.3 MSCM register descriptions

#### 7.4.3.1 MSCM memory map

MSCM.MSCM base address: 4026_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Processor X Type (CPXTYPE) | 32 | R | See section |
| 4h | Processor X Number (CPXNUM) | 32 | R | See section |
| 8h | Processor X Revision (CPXREV) | 32 | R | See section |
| Ch | Processor X Configuration 0 (CPXCFG0) | 32 | R | See section |
| 10h | Processor X Configuration 1 (CPXCFG1) | 32 | R | See section |
| 14h | Processor X Configuration 2 (CPXCFG2) | 32 | R | See section |
| 18h | Processor x Configuration 3 (CPXCFG3) | 32 | R | 0000_000Bh |
| 20h | Processor 0 Type (CP0TYPE) | 32 | R | 434D_3730h |
| 24h | Processor 0 Number (CP0NUM) | 32 | R | See section |
| 28h | Processor 0 Count (CP0REV) | 32 | R | See section |
| 2Ch | Processor 0 Configuration 0 (CP0CFG0) | 32 | R | 0502_0504h |
| 30h | Processor 0 Configuration 1 (CP0CFG1) | 32 | R | 0000_0000h |
| 34h | Processor 0 Configuration 2 (CP0CFG2) | 32 | R | See section |
| 38h | Processor 0 Configuration 3 (CP0CFG3) | 32 | R | 0000_000Bh |
| 200h | Interrupt Router CP0 Interrupt Status (IRCP0ISR0) | 32 | RW | 0000_0000h |
| 204h | Interrupt Router CP0 Interrupt Generation (IRCP0IGR0) | 32 | RW | 0000_0000h |
| 208h | Interrupt Router CP0 Interrupt Status (IRCP0ISR1) | 32 | RW | 0000_0000h |
| 20Ch | Interrupt Router CP0 Interrupt Generation (IRCP0IGR1) | 32 | RW | 0000_0000h |
| 210h | Interrupt Router CP0 Interrupt Status (IRCP0ISR2) | 32 | RW | 0000_0000h |
| 214h | Interrupt Router CP0 Interrupt Generation (IRCP0IGR2) | 32 | RW | 0000_0000h |
| 218h | Interrupt Router CP0 Interrupt Status (IRCP0ISR3) | 32 | RW | 0000_0000h |
| 21Ch | Interrupt Router CP0 Interrupt Generation (IRCP0IGR3) | 32 | RW | 0000_0000h |
| 220h | Interrupt Router CP1 Interrupt Status (IRCP1ISR0) | 32 | RW | 0000_0000h |
| 224h | Interrupt Router CP1 Interrupt Generation (IRCP1IGR0) | 32 | RW | 0000_0000h |
| 228h | Interrupt Router CP1 Interrupt Status (IRCP1ISR1) | 32 | RW | 0000_0000h |
| 22Ch | Interrupt Router CP1 Interrupt Generation (IRCP1IGR1) | 32 | RW | 0000_0000h |
| 230h | Interrupt Router CP1 Interrupt Status (IRCP1ISR2) | 32 | RW | 0000_0000h |
| 234h | Interrupt Router CP1 Interrupt Generation (IRCP1IGR2) | 32 | RW | 0000_0000h |
| 238h | Interrupt Router CP1 Interrupt Status (IRCP1ISR3) | 32 | RW | 0000_0000h |
| 23Ch | Interrupt Router CP1 Interrupt Generation (IRCP1IGR3) | 32 | RW | 0000_0000h |
| 400h | Interrupt Router Configuration (IRCPCFG) | 32 | RW | 0000_0000h |
| 600h | Enable Interconnect Error Detection (ENEDC) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 700h | AHB Gasket Configuration (IAHBCFGREG) | 32 | RW | 0000_0000h |
| 880h - A5Eh | Interrupt Router Shared Peripheral Routing Control (IRSPRC0 - IRSPRC239) | 16 | RW | 0003h |

### 7.4.3.2 Processor X Type (CPXTYPE)

#### Offset

| Register | Offset |
|---|---|
| CPXTYPE | 0h |

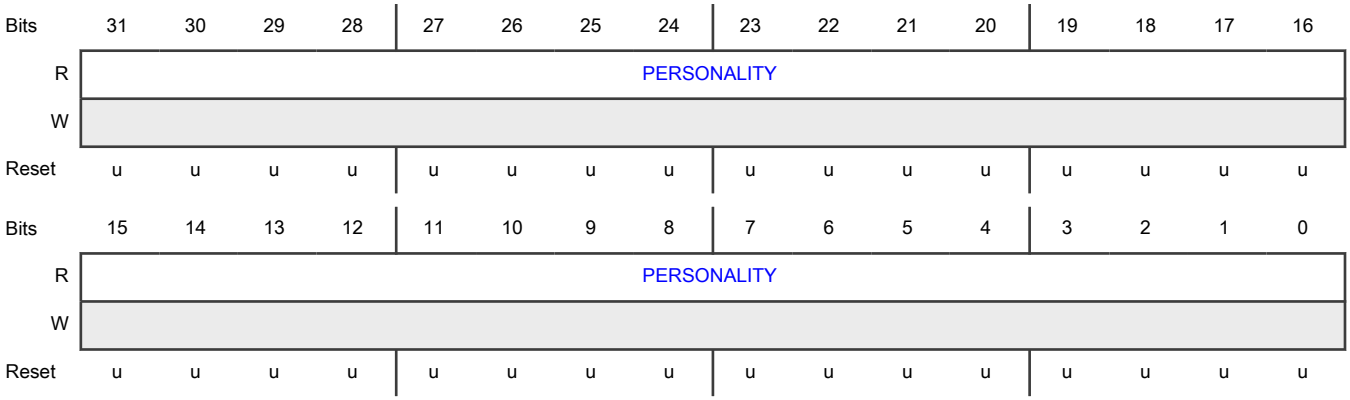
#### Function

Provides a CPU-specific response indicating the personality of the core making the access. The 32-bit response includes four ASCII characters defining the CPU type (Cortex-M7 cores) along with a byte defining the logical revision number and a byte defining the instance number of the core.

A read from Cortex-M7 returns the appropriate processor information. Reads from any other bus master return all 0s and attempted write accesses terminate with an error.

Access: User or privileged read-only

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-0<br>PERSONALITY | Personality of CPx<br>Defines the processor personality for CP*x*. |

| Field | Function |
|---|---|
| | |

| Processor | Personality |
|---|---|
| CP*x* = Cortex-M7_0 | 43_4D_37_30h |

### 7.4.3.3 Processor X Number (CPXNUM)

#### Offset

| Register | Offset |
|---|---|
| CPXNUM | 4h |

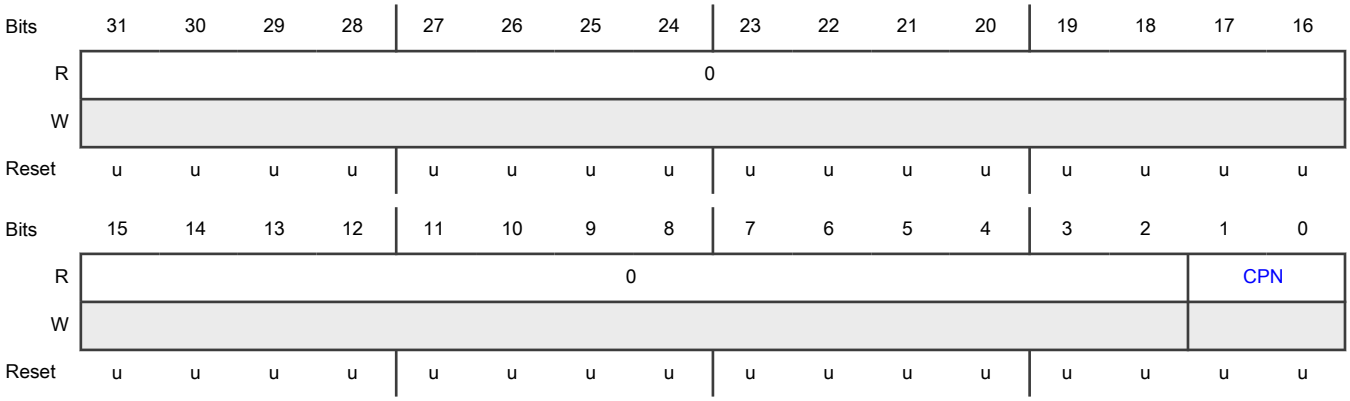
#### Function

Provides a CPU-specific response indicating the logical processor number of the core making the access.

A read from the Cortex-M7 cores returns the appropriate processor information. Reads from any other bus master return all 0s and attempted write accesses terminate with an error.

Access: User or privileged read-only

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1-0<br>CPN | Processor Number<br>This zero-filled word defines the logical processor number for CP*x*. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>CPN in MSCM indicates only the on-platform cores and not the HSE core. CPN = 0 represents Cortex-M7_0 . Processor X Number (CPXNUM).<br>00b - Cortex-M7 core 0<br>01b - Reserved |

### 7.4.3.4 Processor X Revision (CPXREV)

#### Offset

| Register | Offset |
|---|---|
| CPXREV | 8h |

#### Function

Provides a CPU-specific response indicating the logical revision number of the core.

A read from the Cortex-M7 cores returns the appropriate processor information. Reads from any other bus master return all 0s and attempted write accesses terminate with an error.

Access: User or privileged read-only

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0 | Processor Revision |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| RYPZ | Defines the processor revision for CP*x*.<br>For the Cortex-M7 cores in this chip, RYPZ = 12h corresponding to the r1p2 core release. |

### 7.4.3.5 Processor X Configuration 0 (CPXCFG0)

#### Offset

| Register | Offset |
|---|---|
| CPXCFG0 | Ch |

#### Function

Provides a CPU-specific response detailing configuration information. In this case, it is information on Level 1 (L1) cache, if present.

A read from Cortex-M7 returns the appropriate processor information. Reads from any other bus master return all 0s and attempted write accesses terminate with an error.

Access: User or privileged read-only

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ICSZ | | | | | | | | ICWY | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCSZ | | | | | | | | DCWY | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

#### Fields

| Field | Function |
|---|---|
| 31-24<br>ICSZ | Level 1 Instruction Cache Size<br>Provides an encoded value of the instruction cache size. The capacity of the memory is derived using the formula $2^{(8+SZ)}$ and expressed as bytes. Here, ICSZ is a non-zero value and ICSZ = 0 indicates that the memory is not present.<br>For the Cortex-M7 cores in this chip, ICSZ = 5h (8 KB). |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 23-16<br>ICWY | L1 Instruction Cache Ways<br>Provides the number of cache ways for the instruction cache.<br>For Cortex-M7 cores in this chip, ICWY = 2h (2-way set-associative). |
| 15-8<br>DCSZ | L1 Data Cache Size<br>Provides an encoded value of the data cache size.<br>The capacity of the memory is derived using the formula $2^{(8+SZ)}$ and expressed as bytes. Here, DCSZ is a non-zero value and DCSZ = 0 indicates that the memory is not present.<br>For Cortex-M7 cores in this chip, DCSZ = 5h (8 KB). |
| 7-0<br>DCWY | L1 Data Cache Ways<br>Provides the number of cache ways for the data cache.<br>For the Cortex-M7 cores in this chip, DCWY = 4h (4-way set-associative). |

### 7.4.3.6 Processor X Configuration 1 (CPXCFG1)

#### Offset

| Register | Offset |
|---|---|
| CPXCFG1 | 10h |

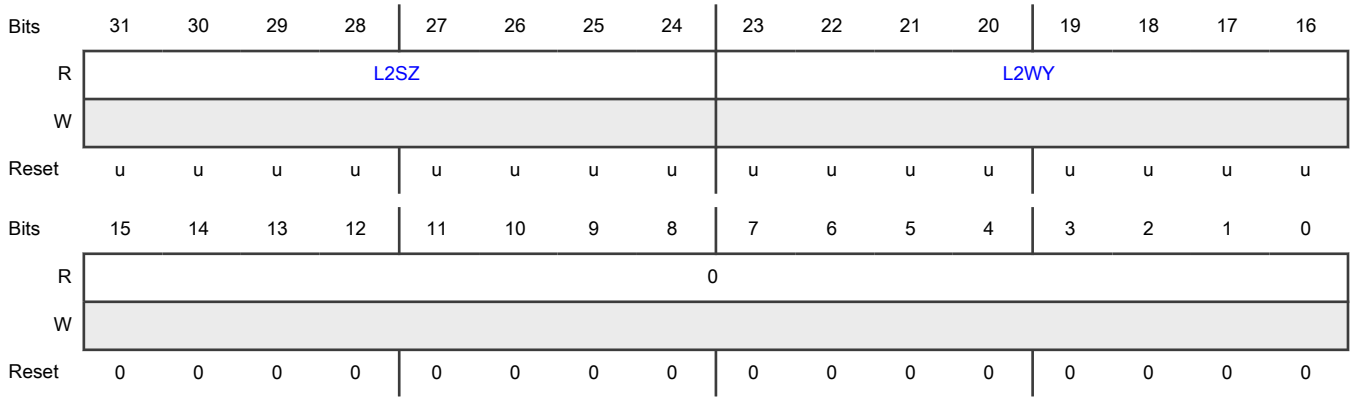
#### Function

Provides a CPU-specific response detailing configuration information. In this case, it is information on Level 2 (L2) cache, if present.

A read from the Cortex-M7 cores returns the appropriate processor information. Reads from any other bus master return all 0s and attempted write accesses terminate with an error.

Access: User or privileged read-only

#### Diagram



### Fields

| Field | Function |
|---|---|
| 31-24<br>L2SZ | L2 Cache Size<br>Provides an encoded value of the L2 cache size. The capacity of the memory is derived using the formula $2^{(8+SZ)}$ and expressed as bytes. Here, L2SZ is a non-zero value, and L2SZ = 0 indicates that the memory is not present.<br>For the Cortex-M7 cores in this chip, L2SZ = 0h (not present). |
| 23-16<br>L2WY | L2 Cache Ways<br>Provides the number of cache ways for the L2 cache.<br>For the Cortex-M7 cores in this chip, L2WY = 0h (not present). |
| 15-0<br>— | Reserved |

## 7.4.3.7 Processor X Configuration 2 (CPXCFG2)

### Offset

| Register | Offset |
|---|---|
| CPXCFG2 | 14h |

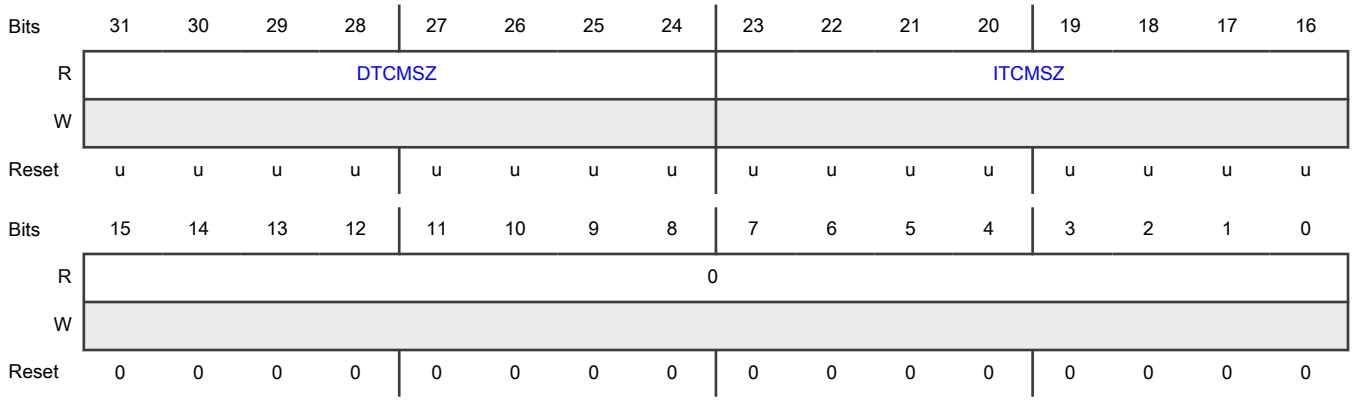
### Function

Provides a CPU-specific response detailing configuration information. In this case, it is information on tightly coupled local memories, if present.

A read from the Cortex-M7 cores returns the appropriate processor information. Reads from any other bus master return all 0s and attempted write accesses terminate with an error.

Access: User or privileged read-only

### Diagram



**Fields**

| Field | Function |
|---|---|
| 31-24<br>DTCMSZ | Tightly Coupled Data Memory Size<br>Provides an encoded value of the tightly coupled local data memory size. The capacity of the memory is derived using the formula $2^{(8+SZ)}$ and expressed as bytes. Here, DTCMSZ is a non-zero value and DTCMSZ = 0 indicates that the memory is not present.<br>For the Cortex-M7 cores in this chip:<br>• DTCMSZ = 08 in Decoupled mode (64 KB)<br>• DTCMSZ = 09 for Cortex-M7_0 in Lockstep mode (128 KB) |
| 23-16<br>ITCMSZ | Instruction Tightly Coupled Memory Size<br>Provides an encoded value of the tightly coupled local instruction memory size. The capacity of the memory is derived using the formula $2^{(8+SZ)}$ and expressed as bytes. Here, ITCMSZ is a non-zero value, and ITCMSZ = 0 indicates that the memory is not present.<br>For the Cortex-M7 cores in this chip:<br>• ITCMSZ = 07 in Decoupled mode (32 KB)<br>• ITCMSZ = 08 for Cortex-M7_0 in Lockstep mode (64 KB) |
| 15-0<br>— | Reserved |

### 7.4.3.8 Processor x Configuration 3 (CPXCFG3)

**Offset**

| Register | Offset |
|---|---|
| CPXCFG3 | 18h |

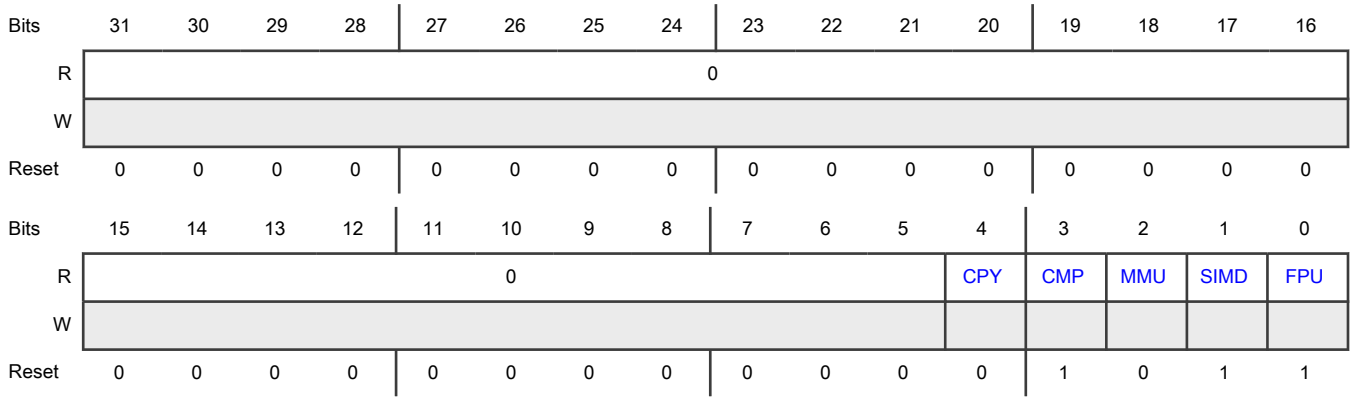
**Function**

Provides a CPU-specific response detailing configuration information. In this case, it is information on processor options.

A privileged read from Cortex-M7 returns the appropriate processor information. Reads from any other bus master return all 0s and attempted write accesses terminate with an error.

Access: User or privileged read-only

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-5<br>— | Reserved |
| 4<br>CPY | Cryptography<br>Indicates if the cryptography extensions are supported in the core.<br>For the Cortex-M7 cores in this chip, CPY = 0h.<br>0b - Not supported<br>1b - Supported |
| 3<br>CMP | Core Memory Protection Unit<br>Indicates if the core memory protection hardware is included in this core.<br>For the Cortex-M7 cores in this chip, CMP = 1h.<br>0b - Not included<br>1b - Included |
| 2<br>MMU | Memory Management Unit<br>Indicates if virtual management capabilities are supported in this core.<br>For the Cortex-M7 cores in this chip, MMU = 0h.<br>0b - Not supported<br>1b - Supported |
| 1<br>SIMD | SIMD/NEON Instruction Support<br>Indicates if the instruction set extensions supporting SIMD and/or NEON capabilities are included in the processor.<br>For the Cortex-M7 cores in this chip, SIMD = 1h.<br>0b - Not included |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Included |
| 0<br>FPU | Floating Point Unit<br>Indicates if hardware support for floating point capabilities is provided in the processor.<br>For the Cortex-M7 cores in this chip, FPU = 1h.<br>0b - Not provided<br>1b - Provided |

### 7.4.3.9 Processor 0 Type (CP0TYPE)

#### Offset

| Register | Offset |
|---|---|
| CP0TYPE | 20h |

#### Function

Defines the configuration information for processor 0 (CP0). It has the same field definitions and functionality as provided in Processor X Type (CPXTYPE).

A read from any bus master returns the appropriate processor information and attempted write accesses terminate with an error.

Access: User or privileged read-only

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PERSONALITY | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset [1] | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |

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

1. u

#### Fields

| Field | Function |
|---|---|
| 31-0<br>PERSONALITY | Processor Personality<br>This field defines the processor personality for CP0.<br>For CP0–Cortex-M7 core 0, personality = 43_4D_37_30h. |

### 7.4.3.10 Processor 0 Number (CP0NUM)

#### Offset

| Register | Offset |
|---|---|
| CP0NUM | 24h |

#### Function

Defines the configuration information for processor 0 (CP0). It has the same field definitions and functionality as provided in Processor X Number (CPXNUM).

A privileged read from any bus master returns the appropriate processor information and attempted write accesses terminate with an error.

Access: User or privileged read-only

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1-0<br>CPN | Processor Number<br>This zero-filled word defines the logical processor number for CP0.<br>For processor Cortex-M7 core 0, processor number = 0. |

### 7.4.3.11 Processor 0 Count (CP0REV)

#### Offset

| Register | Offset |
|---|---|
| CP0REV | 28h |

#### Function

Defines the configuration information for processor 0 (CP0). It has the same field definitions and functionality as provided in Processor X Revision (CPXREV).

A read from any bus master returns the appropriate processor information and attempted write accesses terminate with an error.

Access: User or privileged read-only

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>RYPZ | Processor Revision<br>Defines the processor revision for CP0.<br>For the Cortex-M7 processor, RYPZ = 12h corresponding to the r1p2 core release. |

### 7.4.3.12 Processor 0 Configuration 0 (CP0CFG0)

#### Offset

| Register | Offset |
|---|---|
| CP0CFG0 | 2Ch |

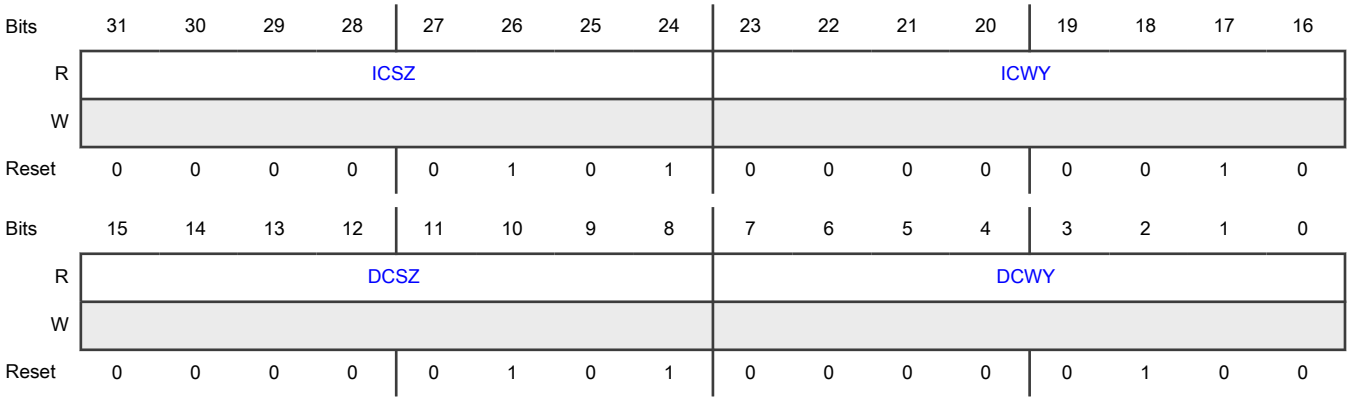
### Function

Defines the configuration information for processor 0 (CP0). It has the same field definitions and functionality as provided in Processor X Configuration 0 (CPXCFG0).

A read from any bus master returns the appropriate processor information and attempted write accesses terminate with an error.

Access: User or privileged read-only

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>ICSZ | Level 1 Instruction Cache Size<br>Provides an encoded value of the instruction cache size. The capacity of the memory is derived using the formula $2^{(8+SZ)}$ and expressed as bytes. Here, ICSZ is a non-zero value, and ICSZ = 0 indicates that the memory is not present.<br>For the Cortex-M7 cores in this chip, ICSZ = 5h (8 KB). |
| 23-16<br>ICWY | L1 Instruction Cache Ways<br>Provides the number of cache ways for the instruction cache.<br>For the Cortex-M7 cores in this chip, ICWY = 2h (2-way set-associative). |
| 15-8<br>DCSZ | L1 Data Cache Size<br>Provides an encoded value of the data cache size.<br>The capacity of the memory is derived using the formula $2^{(8+SZ)}$ and expressed as bytes. Here, DCSZ is a non-zero value, and DCSZ = 0 indicates that the memory is not present.<br>For Cortex-M7 cores in this chip, DCSZ = 5h (8 KB). |
| 7-0<br>DCWY | L1 Data Cache Ways<br>Provides the number of cache ways for the data cache.<br>For the Cortex-M7 cores in this chip, DCWY = 4h (4-way set-associative). |

### 7.4.3.13 Processor 0 Configuration 1 (CP0CFG1)

#### Offset

| Register | Offset |
|---|---|
| CP0CFG1 | 30h |

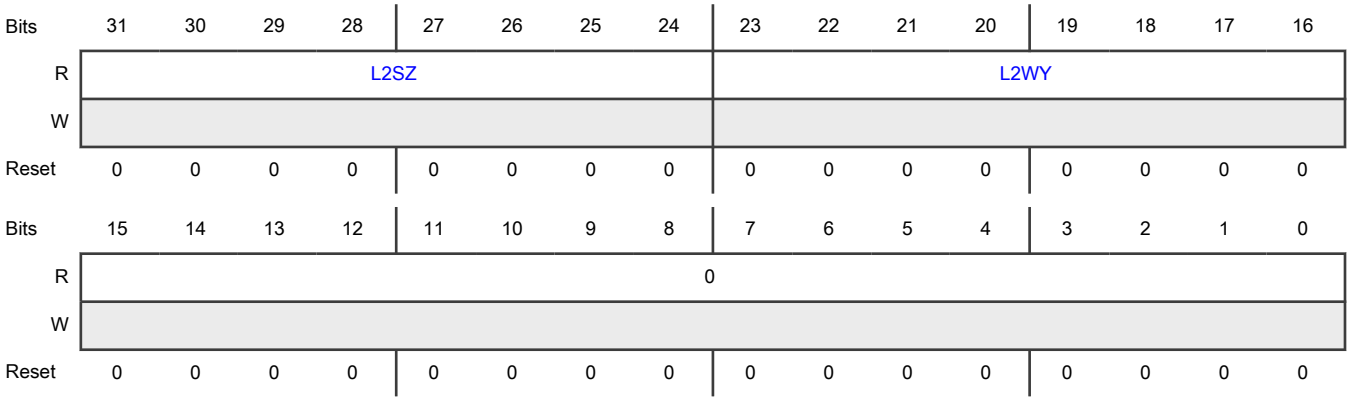
#### Function

Defines the configuration information for processor 0 (CP0). It has the same field definitions and functionality as provided in Processor X Configuration 1 (CPXCFG1).

A privileged read from any bus master returns the appropriate processor information and attempted write accesses terminate with an error.

Access: User or privileged read-only

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-24<br>L2SZ | L2 Cache Size<br>Provides an encoded value of the L2 cache size.<br>The capacity of the memory is derived using the formula $2^{(8+SZ)}$ and expressed as bytes. Here, L2SZ is a non-zero value, and L2SZ = 0 indicates that the memory is not present.<br>For the Cortex-M7 cores in this chip, L2SZ = 0h (not present). |
| 23-16<br>L2WY | L2 Cache Ways<br>Provides the number of cache ways for the L2 cache.<br>For the Cortex-M7 cores in this chip, L2WY = 0h (not present). |
| 15-0<br>— | Reserved |

### 7.4.3.14 Processor 0 Configuration 2 (CP0CFG2)

#### Offset

| Register | Offset |
|---|---|
| CP0CFG2 | 34h |

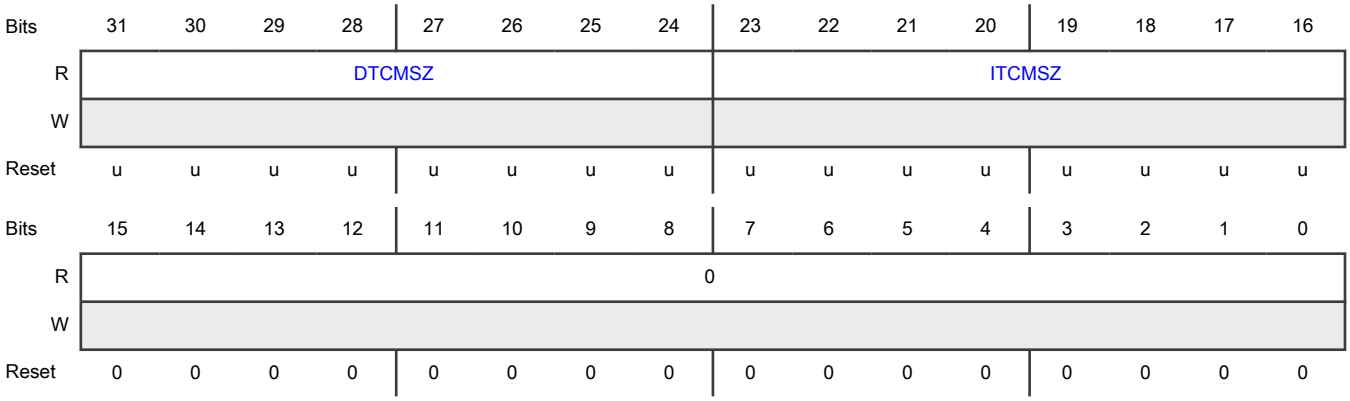
#### Function

Defines the configuration information for processor 0 (CP0). It has the same field definitions and functionality as provided in Processor X Configuration 2 (CPXCFG2).

A privileged read from any bus master returns the appropriate processor information and attempted write accesses terminate with an error.

Access: User or privileged read-only

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-24<br>DTCMSZ | Tightly Coupled Data Memory Size<br>Provides an encoded value of the tightly coupled local data memory size.<br>The capacity of the memory is derived using the formula $2^{(8+SZ)}$ and expressed as bytes. Here, TMLSZ is a non-zero value, and TMLSZ = 0 indicates that the memory is not present.<br>For the Cortex-M7 cores in this chip:<br>• DTCMSZ = 8h in Decoupled mode (64 KB)<br>• DTCMSZ = 9h in Lockstep mode (128 KB) |
| 23-16<br>ITCMSZ | Instruction Tightly Coupled Memory Size<br>Provides an encoded value of the tightly coupled local instruction memory size.<br>The capacity of the memory is derived using the formula $2^{(8+SZ)}$ and expressed as bytes. Here, TMUSZ is a non-zero value, and TMUSZ = 0 indicates that the memory is not present. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | For the Cortex-M7 cores in this chip:<br>• ITCMSZ = 7h in Decoupled mode (32 KB)<br>• ITCMSZ = 8h in Lockstep mode (64 KB) |
| 15-0<br>— | Reserved |

### 7.4.3.15 Processor 0 Configuration 3 (CP0CFG3)

#### Offset

| Register | Offset |
|---|---|
| CP0CFG3 | 38h |

#### Function

Defines the configuration information for processor 0 (CP0). It has the same field definitions and functionality as provided in Processor x Configuration 3 (CPXCFG3).

A privileged read from any bus master returns the appropriate processor information and attempted write accesses terminate with an error.

Access: User or privileged read-only

#### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | CPY | CMP | MMU | SIMD | FPU |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

#### Fields

| Field | Function |
|---|---|
| 31-5<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 4<br>CPY | Cryptography<br>Indicates if the cryptography extensions are supported in the core.<br>For the Cortex-M7 cores in this chip, CPY = 0h.<br>0b - Not supported<br>1b - Supported |
| 3<br>CMP | Core Memory Protection Unit<br>Indicates if the core memory protection hardware is included in this core.<br>For the Cortex-M7 cores in this chip, CMP = 1h.<br>0b - Not included<br>1b - Included |
| 2<br>MMU | Memory Management Unit<br>Indicates if virtual management capabilities are supported in this core.<br>For the Cortex-M7 cores in this chip, MMU = 0h.<br>0b - Not supported<br>1b - Supported |
| 1<br>SIMD | SIMD/NEON Instruction Support<br>Indicates if the instruction set extensions supporting SIMD and/or NEON capabilities are included in the processor.<br>For the Cortex-M7 cores in this chip, SIMD = 1h.<br>0b - Not included<br>1b - Included |
| 0<br>FPU | Floating Point Unit<br>Indicates if hardware support for floating point capabilities is provided in the processor.<br>For the Cortex-M7 cores in this chip, FPU = 1h.<br>0b - Not provided<br>1b - Provided |

#### 7.4.3.16 Interrupt Router CPn Interrupt Status (IRCP0ISR0 - IRCP1ISR3)

**Offset**

For n = 0 to 1; m = 0 to 3:

| Register | Offset |
|---|---|
| IRCPnISRm | 200h + (n × 20h) + (m × 8h) |

### Function

Provides an interrupt bit map, where each bit defines the state of a unique MSI based on the initiating core. An MSI interrupt clears in an interrupt service routine by writing 1 to the appropriate field in IRCP*n*ISR*m*.

In this discussion, CP*m* represents the initiating core and CP*n* represents the target core for a core-to-core interrupt. For more information on interrupt source mapping, see the interrupt map file attached to this document.

For read access:

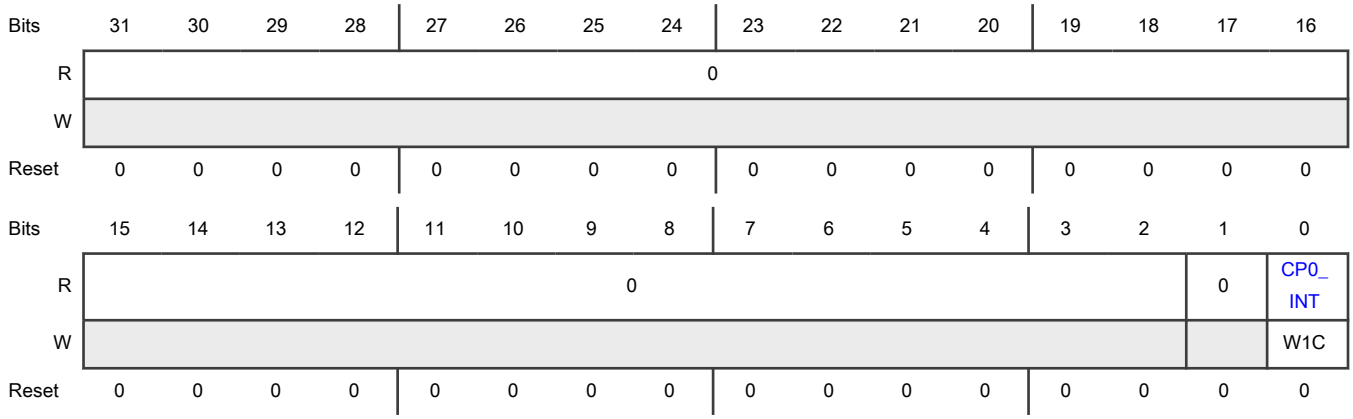
- Reads to IRCP*n*ISR*m* are only accessible in Privileged mode using 32-bit (word) accesses.
- Privileged 32-bit read accesses from non-core (and non-debugger) bus masters are treated as RAZ.
- Attempted accesses in User mode or the ones using a size other than 32 bits are not permitted. They terminate with an error.
- When CP*n* requests to read IRCP*n*ISR*m*, MSCM returns the entire content of IRCP*n*ISR*m*.
- When a trusted core, as indicated by Interrupt Router Configuration (IRCPCFG), requests to read IRCP*n*ISR*m*, MSCM returns the entire content of IRCP*n*ISR*m*.
- When the debugger requests to read IRCP*n*ISR*m*, MSCM returns the entire content of IRCP*n*ISR*m*.
- When CP*m* requests to read IRCP*n*ISR*m*, MSCM returns the value of the corresponding status, CP*m*_INT, while not exposing all the other pending interrupts that the cores initiated.
- When CP*m* requests to read IRCP*n*ISR*m*, MSCM returns the value of the corresponding status, CP*m*_INT, in bit position 0, reflecting how CP*m* set the MSI when it wrote to IRCP*n*IGR*m*. All the other fields on the returned read value are zero-filled.

For write access:

- Writes to IRCP*n*ISR*m* are only accessible in Privileged mode using 32-bit (word) accesses.
- Attempted accesses in User mode or the ones using a size other than 32 bits are not permitted. They terminate with an error.
- Writes to IRCP*n*ISR*m* follow the Write 1 to Clear (W1C) protocol, whereby writing 1 causes the corresponding field to become 0, and writing 0 is ignored.
- The target core, CP*n*, has full access to write to all the fields of IRCP*n*ISR*m*.
- A trusted core, as indicated by Interrupt Router Configuration (IRCPCFG), has full access to write to all the fields of IRCP*n*ISR*m*.
- When CP*m* is different from CP*n*, the W1C action by CP*m* only clears IRCP*n*ISR*m*[CP*m*_INT].
- The CP*m* field must present W1C in bit position 0 to clear its corresponding interrupt. Write data bits 1-31 that CP*m* presents are ignored.
- Privileged write accesses from the non-core (and non-debugger) bus masters are treated as Writes Ignored (WI).
- Privileged write accesses from the debugger are treated as WI.

Access: Privileged mode only

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>CP0_INT | CP0-to-CPn Interrupt<br>Generates a directed interrupt initiated by core 0 targeting core *n*, if the appropriate interrupt routing bit is enabled. The interrupt is negated when the target core, a trusted core, or core 0 writes 1 to clear the field.<br>0b - No interrupt asserted to CPn<br>1b - Interrupt to CPn asserted |

## 7.4.3.17 Interrupt Router CPn Interrupt Generation (IRCP0IGR0 - IRCP1IGR3)

### Offset

For n = 0 to 1; m = 0 to 3:

| Register | Offset |
|---|---|
| IRCPnIGRm | 204h + (n × 20h) + (m × 8h) |

### Function

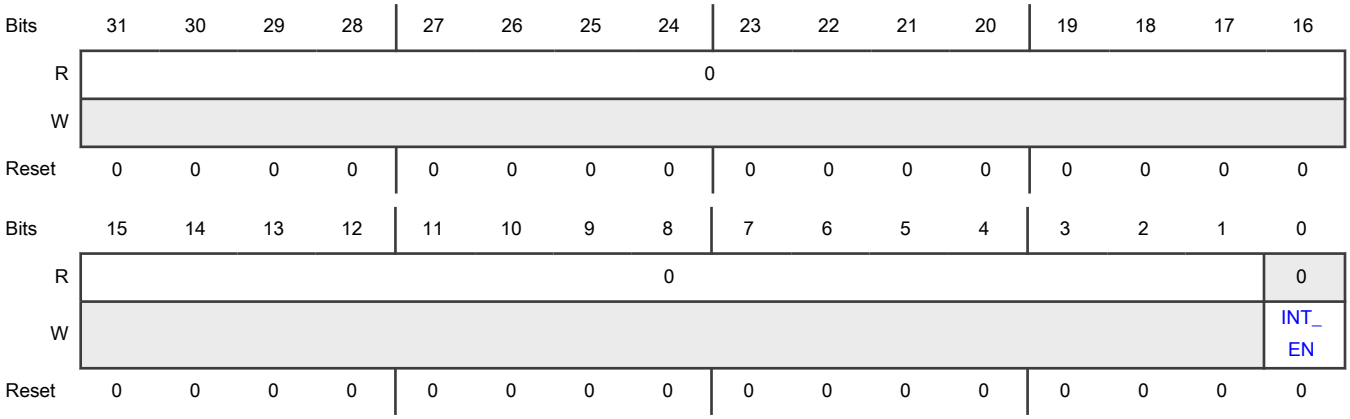
Provides a mechanism for cores to initiate an MSI to another core in the system.

Privileged, 32-bit accesses from the:

- Cortex-M7 cores are treated as RAZ/W.
- Debugger are treated as RAZ/WI.
- Non-core (and non-debugger) bus masters are treated as RAZ/WI.

Access: Privileged mode only. Attempted accesses in User mode or the ones using a size other than 32 bits are not permitted and terminate with an error.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>INT_EN | Interrupt Enable<br>Initiates a core-to-core interrupt targeting CP*n*, if CP*m* writes to this field. See MSI routing. |

## 7.4.3.18 Interrupt Router Configuration (IRCPCFG)

### Offset

| Register | Offset |
|---|---|
| IRCPCFG | 400h |

### Function

Provides a mechanism to designate specific cores in the system as trusted. These trusted cores are allowed to access and manage outstanding MSIs.

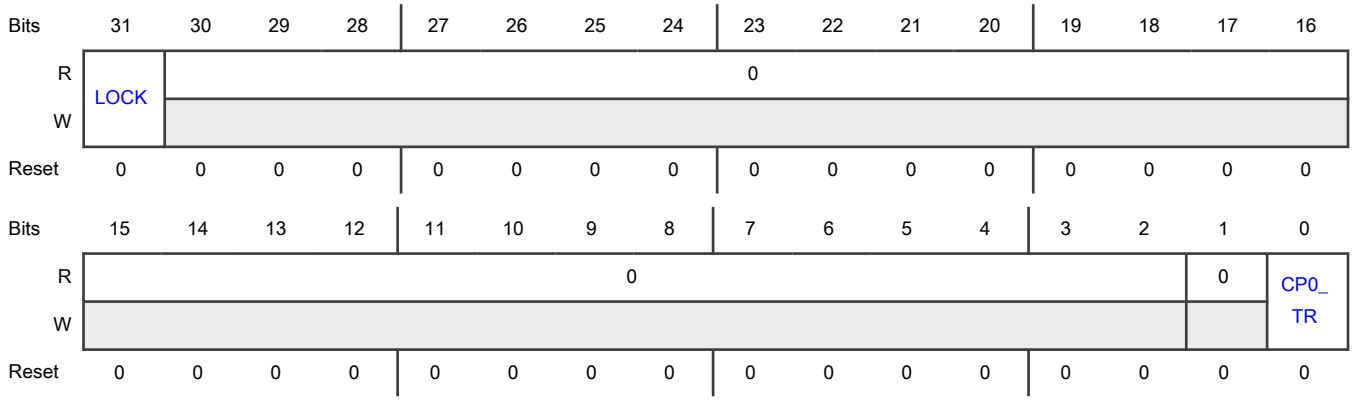
Privileged, 32-bit accesses from the:

- Cortex-M7 cores are treated as R/W.
- Debugger are treated as R/WI.
- Non-core (and non-debugger) bus masters are treated as RAZ/WI.

Attempted accesses in User mode or the ones using a size other than 32 bits are not permitted. They terminate with an error.

Access: Privileged mode only

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LOCK | Lock<br>Provides a locking mechanism that can be used to limit the ability to write to the register. After you write 1 to this field, it remains 1 until the next reset.<br>0b - Register can be written by any privileged write<br>1b - Register is locked (read-only) until the next reset |
| 30-2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>CP0_TR | CP0 as Trusted Core<br>Indicates if CP0 is a trusted core with access to read the full contents of IRCP*n*ISR*m*.<br>0b - Not trusted<br>1b - Trusted |

## 7.4.3.19 Enable Interconnect Error Detection (ENEDC)

### Offset

| Register | Offset |
|---|---|
| ENEDC | 600h |

### Function

Enables interconnect error detection.

For more information, see the FCCU file attached to this document.

Access: Privileged mode only

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | EN_ADD_... | EN_WR_C... | EN_ADD_... | EN_WR_A... | EN_ADD_... | EN_WR_A... | EN_ADD_... | EN_WR_A... | EN_ADD_... | Reserved | EN_ADD_... | EN_WR_T... |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EN_ADD_... | EN_WR_P... | EN_ADD_... | EN_WR_P... | EN_ADD_... | EN_ADD_... | EN_ADD_... | EN_RD_T... | Reserved | | EN_RD_E... | EN_RD_H... | Reserved | EN_RD_E... | EN_RD_C... | EN_RD_C... |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27<br>EN_ADD_CM7_0_TCM | Enable Address Check Cortex-M7_0_TCM<br>Enables or disables the address check for Cortex-M7_0_TCM backdoor path.<br>0b - Disabled<br>1b - Enabled |
| 26<br>EN_WR_CM7_0_TCM | Enable Write Data Check Cortex-M7_0_TCM<br>Enables or disables the write data check for Cortex-M7_0_TCM backdoor path.<br>0b - Disabled<br>1b - Enabled |
| 25<br>EN_ADD_AIPS2 | Enable Address Check AIPS2<br>Enables or disables the AIPS2 address check.<br>0b - Disabled<br>1b - Enabled |
| 24<br>EN_WR_AIPS2 | Enable Write Data Check AIPS2<br>Enables and disables the write data check for AIPS2.<br>0b - Disabled<br>1b - Enabled |
| 23<br>EN_ADD_AIPS1 | Enable Address Check AIPS1<br>Enables or disables the address check for AIPS1 path.<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 22<br>EN_WR_AIPS1 | Enable Write Data Check AIPS1<br>Enables or disables the write data check for AIPS1 path.<br>0b - Disabled<br>1b - Enabled |
| 21<br>EN_ADD_AIPS0 | Enable Address Check AIPS0<br>Enables or disables the address check for AIPS0 path.<br>0b - Disabled<br>1b - Enabled |
| 20<br>EN_WR_AIPS0 | Enable Write Data Check AIPS0<br>Enables or disables the write data check for AIPS0 path.<br>0b - Disabled<br>1b - Enabled |
| 19<br>EN_ADD_QSPI | Enable Address Check QuadSPI<br>Enables or disables the address check for QuadSPI path.<br>0b - Disabled<br>1b - Enabled |
| 18<br>— | Reserved |
| 17<br>EN_ADD_TCM | Enable Address Check TCM<br>Enables or disables the address check for TCM 64-bit path.<br>0b - Disabled<br>1b - Enabled |
| 16<br>EN_WR_TCM | Enable Write Data Check TCM<br>Enables or disables the write data check for TCM 64-bit path.<br>0b - Disabled<br>1b - Enabled |
| 15<br>EN_ADD_PRAM1 | Enable Address Check PRAM1<br>Enables or disables the address check for PRAM1 path.<br>0b - Disabled<br>1b - Enabled |
| 14 | Enable Write Data Check PRAM1 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| EN_WR_PRAM1 | Enables or disables the write data check for PRAM1.<br>0b - Disabled<br>1b - Enabled |
| 13<br>EN_ADD_PRAM0 | Enable Address Check PRAM0<br>Enables or disables the address check for PRAM0 path.<br>0b - Disabled<br>1b - Enabled |
| 12<br>EN_WR_PRAM0 | Enable Write Data Check PRAM0<br>Enables or disables the write data check for PRAM0 path.<br>0b - Disabled<br>1b - Enabled |
| 11<br>EN_ADD_PFLASH_PORT2 | Enable Address Check P_FLASH_PORT2<br>Enables or disables the address check for P_FLASH_PORT2 path.<br>0b - Disabled<br>1b - Enabled |
| 10<br>EN_ADD_PFLASH_PORT1 | Enable Address Check P_FLASH_PORT1<br>Enables or disables the address check for P_FLASH_PORT1 path.<br>0b - Disabled<br>1b - Enabled |
| 9<br>EN_ADD_PFLASH_PORT0 | Enable Address Check P_FLASH_PORT0<br>Enables or disables the address check for P_FLASH_PORT0 path.<br>0b - Disabled<br>1b - Enabled |
| 8<br>EN_RD_TCM | Enable Read Data Check TCM<br>Enables or disables the read data check for TCM 32-bit path.<br>0b - Disabled<br>1b - Enabled |
| 7-6<br>— | Reserved |
| 5<br>EN_RD_EMAC | Enable Read Data Check EMAC<br>Enables or disables read data check for EMAC path. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |
| 4<br>EN_RD_HSE | Enable Read Data Check HSE<br>Enables or disables the read data check for HSE path.<br>0b - Disabled<br>1b - Enabled |
| 3<br>— | Reserved |
| 2<br>EN_RD_EDMA | Enable Read Data Check eDMA<br>Enables or disables the read data check for eDMA path.<br>0b - Disabled<br>1b - Enabled |
| 1<br>EN_RD_CM7_0_AHBP | Enable Read Data Check Cortex-M7_0_AHBP<br>Enables or disables the read data check for Cortex-M7_0_AHBP path.<br>0b - Disabled<br>1b - Enabled |
| 0<br>EN_RD_CM7_0_AHBM | Enable Read Data Check Cortex-M7_0_AHBM<br>Enables or disables the read data check for Cortex-M7_0_AHBM path.<br>0b - Disabled<br>1b - Enabled |

### 7.4.3.20 AHB Gasket Configuration (IAHBCFGREG)

**Offset**

| Register | Offset |
|---|---|
| IAHBCFGREG | 700h |

**Function**

Controls the functional configuration of the AHB gaskets located on the platform.

Access: Privileged mode only

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | AIPS2_D... | 0 | | | AIPS1_D... | 0 | | | QSPI_DI... | 0 | | | TCM_DIS... |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | HSE_DIS... | 0 | | | DMA_AXB... | 0 | | | DMA_AXB... | 0 | | | EMAC_DI... |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |
| 28<br>AIPS2_DIS_WR_OPT | Determines whether write burst optimizations in the AIPS2 AHB gasket are enabled or disabled.<br>Enabling optimization allows performance improvements during burst writes. Disabling optimization is required only if you expect an early write burst termination from a master.<br>0b - Enabled<br>1b - Disabled |
| 27-25<br>— | Reserved |
| 24<br>AIPS1_DIS_WR_OPT | Determines whether write burst optimizations in the AIPS1 AHB gasket are enabled or disabled.<br>Enabling optimization allows performance improvements during burst writes. Disabling optimization is required only if you expect an early write burst termination from a master.<br>0b - Enabled<br>1b - Disabled |
| 23-21<br>— | Reserved |
| 20<br>QSPI_DIS_WR_OPT | Determines whether write burst optimizations in the QuadSPI AHB gasket are enabled or disabled.<br>Enabling optimization allows performance improvements during burst writes. Disabling optimization is required only if you expect an early write burst termination from a master.<br>0b - Enabled<br>1b - Disabled |
| 19-17 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 16<br>TCM_DIS_WR_OPT | Determines whether write burst optimizations in the TCM AHB gasket are enabled or disabled.<br>Enabling optimization allows performance improvements during burst writes. Disabling optimization is required only if you expect an early write burst termination from a master.<br>0b - Enabled<br>1b - Disabled |
| 15-13<br>— | Reserved |
| 12<br>HSE_DIS_WR_OPT | Determines whether write burst optimizations in the HSE AHB gasket are enabled or disabled.<br>Enabling optimization allows performance improvements during burst writes. Disabling optimization is required only if you expect an early write burst termination from a master.<br>0b - Enabled<br>1b - Disabled |
| 11-9<br>— | Reserved |
| 8<br>DMA_AXBS_S1_DIS_WR_OPT | Determines whether write burst optimizations in the DMA AXBS S1 AHB gasket are enabled or disabled.<br>Enabling optimization allows performance improvements during burst writes. Disabling optimization is required only if you expect an early write burst termination from a master.<br>0b - Enabled<br>1b - Disabled |
| 7-5<br>— | Reserved |
| 4<br>DMA_AXBS_S0_DIS_WR_OPT | Determines whether write burst optimizations in the DMA AXBS S0 AHB gasket are enabled or disabled.<br>Enabling optimization allows performance improvements during burst writes. Disabling optimization is required only if you expect an early write burst termination from a master.<br>0b - Enabled<br>1b - Disabled |
| 3-1<br>— | Reserved |
| 0<br>EMAC_DIS_WR_OPT | Determines whether write burst optimizations in the EMAC AHB gasket are enabled or disabled.<br>Enabling optimization allows performance improvements during burst writes. Disabling optimization is required only if you expect an early write burst termination from a master. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Enabled<br>1b - Disabled |

### 7.4.3.21 Interrupt Router Shared Peripheral Routing Control (IRSPRC0 - IRSPRC239)

#### Offset

For n = 0 to 239:

| Register | Offset |
|---|---|
| IRSPRCn | 880h + (n × 2h) |

#### Function

Provides an array of 16-bit registers, where each register defines the routing control for the corresponding interrupt request. Starting from IRQ = 0 (first on-platform interrupt vector). See the interrupt map file attached to this document for details.

For this chip, each interrupt request can be either routed to a subset or to all the cores using the bit-mapped fields in IRSPRC*n*. If all the CP*x*E*n* fields are cleared, the interrupt request is disabled. Each routing control halfword can be locked by writing 1 to the LOCK field.

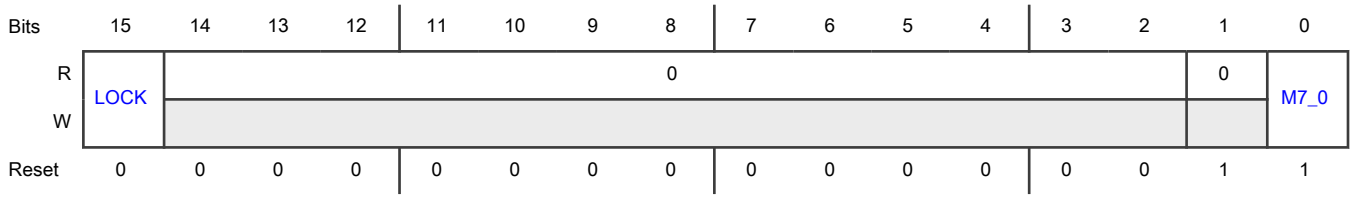
Privileged accesses from non-core (and non-debug) bus masters are treated as RAZ/WI, and any attempted User mode reference terminates with an error. Attempted accesses using a size other than a 16-bit halfword also terminate with an error.

If you write 1 to all the CP*x*En bits, all the cores service the interrupt. It is up to software to ensure that no conflicts arise from this setup either via the interrupt handler or through programming core level interrupt routing (the NVIC/GIC).

Reads and writes to this register beyond IRSPRC207 lead to unpredictable results.

Access: Privileged mode only

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 15<br>LOCK | Lock<br>Provides a mechanism to lock the routing of the corresponding interrupt request. After you write 1 to this field, attempted writes to IRSPRC*n* are ignored until the next reset writes 0 to the field. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Writes to IRSPRCn allowed<br>1b - Writes to IRSPRCn ignored |
| 14-2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>M7_0 | Enable Cortex-M7_0 Interrupt Steering<br>Enables or disables the corresponding interrupt request to route to Cortex-M7_0.<br>0b - Routing disabled<br>1b - Routing enabled |

## 7.5 Glossary

| | |
|---|---|
| **GIC** | Generic interrupt controller |
| **IRQs** | Interrupt requests |
| **ISR** | Interrupt service routine |
| **MSI** | Message signal interface |
| **NVIC** | Nested vector interrupt controller |

# Chapter 8
# Virtualization Wrapper (VIRT_WRAPPER)

## 8.1 Chip-specific VIRT_WRAPPER information

### 8.1.1 System overview of PDAC scheme

This figure shows the interaction of XRDC (containing PDACs), related to different cores (domain IDs), with VIRT_WRAPPER.

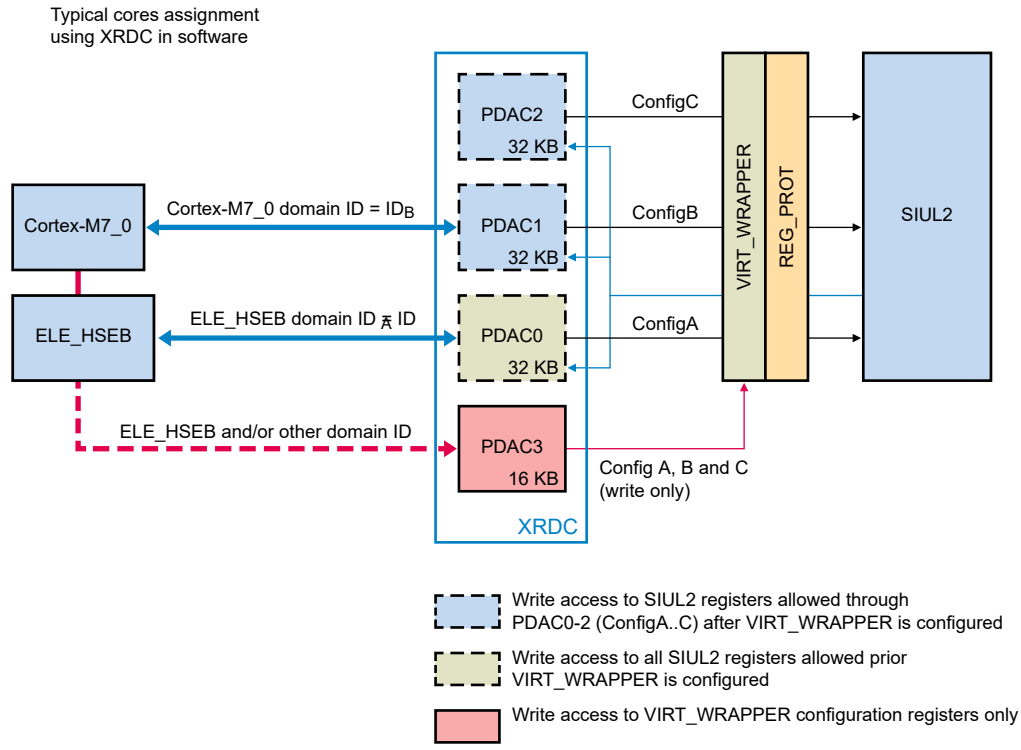



Figure 14. System overview of PDAC scheme

### 8.1.2 Initial VIRT_WRAPPER operation

Initially, VIRT_WRAPPER:

- Protects the "ConfigB-C" paths to SIUL2. You must configure VIRT_WRAPPER to define the SIUL2 R/W control registers that you can access through the "ConfigB-C" VIRT_WRAPPER access paths.
- Does not protect the direct path to SIUL2. It allows write accesses to all the SIUL2 R/W control registers. If PDAC0 is set to allow write access only to master core, for example ELE_HSEB core (variant supported by the chip), then other masters cannot program the SIUL2 register. However, if these non-ELE_HSEB masters program SIUL2 registers using other PDAC slots, a transfer error occurs. Any core can act as master core.

### 8.1.3 Additional VIRT_WRAPPER details

See the "SIUL2 memory map overview and protection" table, later in this chapter, for an overview of the SIUL2 registers and their protection attributes.

By default, any core can access SIUL2 registers through PDAC0. This process is called ANY_MASTER. However, after XRDC configuration, ELE_HSEB:

- Assigns PDACs to the respective cores or domain IDs

- Locks the XRDC configuration
- Programs VIRT_WRAPPER registers for pad assignments

ELE_HSEB accesses PDAC0 solely for the pads having 11b as the value in their corresponding registers. Also, ELE_HSEB retains the same value for the pads that it stores. In case ELE_HSEB needs to take control of some pins, it can still configure PDAC*n* to be accessible not only from a specific core master, but also from ELE_HSEB, before it allows other cores to execute.

PDAC3 protects the IPS register portal of VIRT_WRAPPER configuration registers that any master can access but only via the memory slot assigned to PDAC3. The configuration via IPS register portal can be locked to prevent any further changes until the next functional reset. PDAC3 implements this locking.

The protection information of the parallel port output registers is inherited from the protection information of the included pads, according to these rules:

- When all the pads assigned to a GPIO port share the same protection information, the corresponding parallel port output register for this port inherits the same protection information.
- When at least one of the pads assigned to a GPIO port has different protection information than the other pads assigned to this GPIO port, access to the corresponding parallel port output register for this port is disabled for all the masters. Because of the aforementioned protection group mapping, the data bits of a register encode the protection control information related to one full GPIO port, or a chunk of 16 pads.
- You can assign the following SIUL2 registers to individual PDAC control:
  - DISR0
  - DIRER0
  - DIRSR0
  - IREER0
  - IFEER0
  - IFER0
  - IFMCR0–31
  - IFCPR

  Then, you can access all these registers only through the assigned PDAC because the VIRT_WRAPPER_REG_C1039_1024 register controls the SIUL2 interrupt registers.
- You cannot access the SIUL2 R/W control registers configured through PDAC3 to be accessible through the ConfigA VIRT_WRAPPER access path, using the ConfigB VIRT_WRAPPER access path.
- You cannot access any of the SIUL2 R/W control registers through ConfigB-C VIRT_WRAPPER access paths prior to VIRT_WRAPPER configuration. By default, any core can access SIUL2 registers through PDAC0 after reset.
- The same PDAC that accesses the main SIUL2 registers is used to access the SIUL2-mirrored registers and soft-lock-bit registers. This is because these registers are a part of the same 16 KB space.
- Access to mirrored SIUL2 registers sets the bit in Soft lock bit as an inherent property of register protection. See the "Register Protection" chapter for details on the soft lock bit behavior, when accessing the mirrored region.

## 8.2 Introduction

Virtualization refers to the various techniques, methods, or approaches of creating a virtual (rather than actual) version of something, such as a virtual hardware platform, operating system (OS), storage device, or network resources. The term "hardware virtualization" refers to the creation of a virtual machine that acts like a real computer with an operating system. The underlying hardware resources separate the software executed on these virtual machines (named "guest program"). In many cases, the specifically modified guest programs are required to run in such a virtual environment.

There are different types of hardware virtualization. One of them is para-virtualization. The para-virtualization is a non-simulated hardware environment. However, a guest program is executed in its own isolated domain, as if it is running on a separate

system. Such a behavior is especially beneficial for the software targeted toward functional safety, because it allows freedom of interference for certain aspects of this software.

The hardware-assisted virtualization is a way of improving the efficiency of hardware virtualization. It involves employing specially designed CPUs and hardware components that help improve the performance of a guest environment. VIRT_WRAPPER, described in this chapter, is such a hardware component.

### 8.2.1 Overview

VIRT_WRAPPER is an extension of the Register Protection (REG_PROT) wrapper methodology. It works in parallel with an existing REG_PROT wrapper by virtualizing registers within the virtualized module. In this chip, SIUL2 is the virtualized module. Virtualized module is the module instance associated with VIRT_WRAPPER. Virtualization is implemented by protecting (for example, granting or inhibiting) the access to a register within the virtualized module, dependent on the specific criteria. It also virtualizes registers within REG_PROT when associated with registers within the virtualized module.

Registers within virtualized module or the REG_PROT wrapper are virtualized by granting, or inhibiting the access to different PDAC (Peripheral domain access control (note that PDAC and PDAC slot are interchangeable terms.))s as encoded within virtualization pad assignments in VIRT_WRAPPER configuration registers. For this purpose, the virtualization information is programmed into VIRT_WRAPPER configuration registers that specify the grouping of registers within virtualized module. Upon an access to virtualized module through the peripheral interface, the grouping information is exercised to identify whether the corresponding transaction should be granted, or inhibited. A corresponding signal is forwarded to the REG_PROT wrapper that then grants, or inhibits the transaction. The following figure depicts the usage of VIRT_WRAPPER in combination with the REG_PROT wrapper and virtualized module.

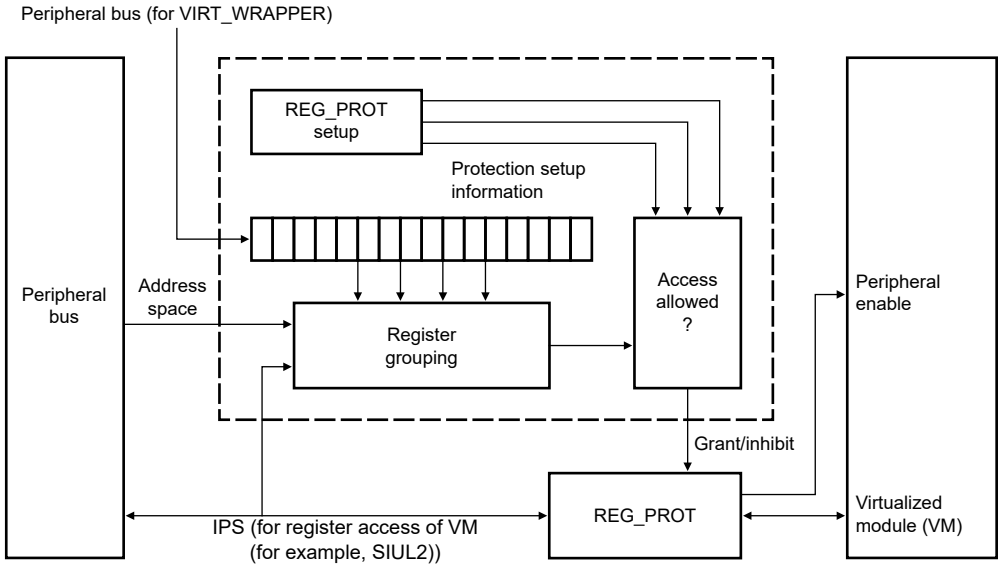



Figure 15. VIRT_WRAPPER block diagram

On a high level, VIRT_WRAPPER is an extension of the REG_PROT wrapper along those lines:

- There is a set of GPRs that receive data over peripheral bus of the VIRT_WRAPPER module. The output of these clients is a quasi-static bus providing the "Protection Control Information" to the remaining logic; this information is kept stable, that is programmed by specific PDAC after boot-up. That should be accessed only by that initiator(s) that is accessing PDAC0. This is taken care by XRDC programming at the SoC.
- The remaining functionality added by VIRT_WRAPPER module consists of two sub-blocks:
  - A first sub-block (the "Grouping PLA") defines the grouping of the registers within the virtualized module. This sub-block is module specific and generated, since this information is specific for the virtualized module, It receives the address information from the peripheral bus and selects the corresponding data from the "Protection Control Information".

— A second sub-block (the "Protection" sub-block) encodes the previously selected "Protection Control Information" and combines it with the information about the received access request. The current protection methodology allows to inhibit only write accesses for a specific initiator or set of initiators. For this purpose, this sub-block evaluates the received access information; especially the byte enables and the PDAC of the accessing initiator. The output of this sub-block is an internal signal "access granted" that is used to flag an access inhibited by the virtualization information.

### 8.2.2 Features

VIRT_WRAPPER includes these features:

- A programmable chip-specific virtualization setup, capable to select a subset of the available initiators in combination with up to four available address spaces
- Virtualizing accesses to the registers within a VM (Virtualized module, such as SIUL2) by inhibiting accesses to a subset of these registers under control of the specified virtualization information. Only write accesses can be inhibited for a specific initiator or set of initiators.
- Inhibiting accesses to registers within REG_PROT associated with VM; in cases, these registers are associated with a register in the VM to which the access is also inhibited. Additionally, accesses to the global control register (GCR) within REG_PROT can be protected separately.

### 8.2.3 Modes of operation

VIRT_WRAPPER is operable when the VM is operable. For details about the availability of the VM, see the chapter of the corresponding module. When there is no virtualization information specified for the VIRT_WRAPPER module, if XRDC is not configured, PDAC slot 0 is open as the default value of configuration register is 11b that is assigned with PDAC slot 0 and any initiator can access through it.

For more details, see "Chip specific VIRT_WRAPPER information".

## 8.3 Functional description

This section describes the following topics:

- PDAC-based protection scheme
- Register group mapping considering SIUL2
- Access errors

### 8.3.1 PDAC-based protection scheme

VIRT_WRAPPER uses a protection scheme of the PDACs sub-blocks of XRDC. Each PDAC slot is assigned to one or more cores (by domain ID assignment in XRDC) as shown in Chip-specific VIRT_WRAPPER information section. Only the write accesses are protected by the virtualization feature. Read accesses are not impacted.

For any protection group defined within the register, two bits specify the protection control information according to the following table.

**Table 30. Protection control information for a single group**

| Value | Mnemonic | Protection | Description |
|---|---|---|---|
| 00 | PDAC1 | Restricted access of registers, MSCR, IMCR, and INTINT, and GCR to PDAC slot 1 | Protection setup information for PDAC slot 1 is used. |
| 11 | PDAC0 | By default, any core can access SIUL2 registers through PDAC slot 0. | Protection setup information for PDAC slot 0 is used. |

### 8.3.2 Clocking

This module has no major clocking considerations.

### 8.3.3 Register group mapping for SIUL2

For the SIUL2 instance, the corresponding virtualization information within the VIRT_WRAPPER module is defined on a per-pad basis. Additionally, the virtualization information protects the control registers (controlling the input multiplexing scheme).

**Table 31. Register group mapping details**

| Register name | Description |
|---|---|
| Reg A | Virtualization of pad output registers |
| Reg B | Virtualization of input multiplexing control registers |
| Reg C | Virtualization of interrupt control registers |
| Reg D | Virtualization of interrupt control registers |

#### 8.3.3.1 Virtualization of pad output registers

Any functional pad (controlled by the SIUL2 instance) is assigned to a separate protection group. This scheme excludes the pads that an MSCR register does not control within an SIUL2 instance (for example, power supply pads). As a result, the protection granularity defined by the virtualization information is a single pad. The control information associated with this protection group affects all the registers related to this pad.

The protection control information for pad *i* is specified within the protection group *i*; thus, it enables a very simple assignment scheme for the related protection control data that is hard-coded. The protection control information for pad *i* affects all the registers related to this pad, in the MSCR*i* and the GPDO*i* registers (in case such registers exist).

SIUL2 can control up to 512 pads that the MSCR, GPDO, and GPDI registers can access individually. The actual number of pads associated with a SIUL2 instance is specific to implementation. GPIO pads are also organized in GPIO ports consisting of a maximum of 16 pads that the parallel port registers (PGPDO, PGPDI, MPGPDO) can access.

Only the pad control and pad output registers are protected. Accesses to the read-only registers GPDI and PGPDI are not affected by virtualization.

As the protection granularity is a single pad, any virtualization information associated with this pad affects the corresponding MCSR and GPDO registers directly. Additionally, any consequence that an illegal access has on the registers implemented within the REG_PROT wrapper is inhibited. For this purpose, any access to the mirrored address space and the corresponding Soft Lock Bit Register (SLBR) is also observed.

The parallel port output registers (PGPDO, MPGPDO) inherit the protection information from the protection information of the included pads according to the following rules:

- When all pads assigned to a GPIO port share the same protection information, the corresponding parallel port output register for this port inherits the same protection information.
- When at least one of the pads assigned to a GPIO port has a different protection information than the other pads assigned to this GPIO port, the write access to the corresponding parallel port output register (PGPDO and MPGPDO) for this port is disabled for all the initiators through all the PDAC slots and a transfer error is generated.

Due to the aforementioned protection group mapping, the data bits of a register encode the protection control information related to one full GPIO port, or a chunk of 16 pads.

#### 8.3.3.2 Virtualization of input multiplexing control registers

Additionally, the SIUL2 instance supports the control of an input multiplexer (INMUX) scheme. This scheme provides the capability to select one of a set of the input pads to be the source of an input function for some of the peripheral modules. Any INMUX

controlled by the SIUL2 instance is assigned to a separate protection group. It therefore provides an equivalent protection granularity within the virtualization information.

The protection group 512+*i* specifies the protection control information for the input multiplexer INMUX *i*. This enables a very simple assignment scheme for the related protection control data that is hard-coded. The protection control information for INMUX *i* affects the input multiplexing control register related to this input—namely, IMCR*i*.

SIUL2 can control up to 512 MSCRs and 512 IMCRs input multiplexers that the IMCRs can access individually. GPIO inputs are not affected by the input multiplexer scheme. Therefore, the associated virtualization information does not affect the corresponding pad input registers.

As the protection granularity is a single INMUX, any virtualization information associated with this INMUX affects the corresponding IMCR directly. Additionally, any consequence that an illegal access has on the registers implemented within the REG_PROT wrapper is inhibited. For this purpose, any access to the mirrored address space and the corresponding SLBR is also observed.

#### 8.3.3.3 Virtualization of interrupt control registers

Additionally, the complete set of interrupt control registers within SIUL2 (address offset range 0010h–00C3h) can be protected as a separate group (protection group #1024).

Another additional protection group (protection group #1055) is defined for the global control register (GCR) of the related REG_PROT wrapper.

The PDAC that accesses the main SIUL2 registers is used to access the SIUL2-mirrored registers and soft-lock-bit registers. This is because these registers are a part of the same 16 KB space.

Access to mirrored SIUL2 registers sets the bit in Soft lock bit as an inherent property of register protection. See the "Register Protection" chapter for details on the soft lock bit behavior, when accessing the mirrored region.

The following table shows an overview of the SIUL2 registers and their protection attributes.

**Table 32. SIUL2 memory map overview and protection**

| Offset range | Register | Size (bits) | Protected | Description |
|---|---|---|---|---|
| 0000–000Fh | MIDR1, MIDR2 | 32 | N | Read-only registers, not protected |
| 0010–00C3h | Interrupt registers | 32 | Y | Protected as a single group—no individual protection of related registers |
| 0240–A3Fh [1] | MSCR | 32 | Y | Amount of registers defines maximum number of PADs to be controlled and protected |
| 0A40–0123Fh | IMCR | 32 | Y | Amount of registers defines maximum number of INMUXes to be controlled and protected |
| 1300–14FFh | GPDO GPDO[0] (8-bit) register controls single PAD[0] pad means that if PAD[0] is assigned to PDAC slot 1 then PDAC slot 1 can write to GPDO[0] (8-bit). | 8 | Y | There are fewer GPDO registers than MSCR registers, as some PADs are not made available as GPIO PADs, but their electrical characteristics can still be programmed |
| 1500–16FFh | GPDI | 8 | N | PAD input register, not protected |
| 1700–173Fh | PGPDO PGPDO[0] (16-bit) register | 16 | Y | Writable with 8-, 16-, and as a pair with 32-bit accesses |

*Table continues on the next page...*

Table 32. SIUL2 memory map overview and protection (continued)

| Offset range | Register | Size (bits) | Protected | Description |
|---|---|---|---|---|
| | controls PAD[0-15] pads meaning if only all the PADs[0-15] are assigned to PDAC slot 1, then PDAC slot 1 can write to PGPDO[0] (16-bit). | | | |
| 1740–177Fh | PGPDI | 16 | N | PAD input register, not protected |
| 1780–17FFh | MPGPDO MPGPDO[0] (32-bit) register controls PAD[0-15] pads meaning if only all the PADs[0-15] are assigned to PDAC slot 1, then PDAC slot 1 can write to MPGPDO[0] (32-bit). | 32 | Y | Only writable with 32-bit accesses |
| 2010–20C3h | Interrupt registers (mirror) The DISR0 DIRER0 DIRSR0 IREER0 IFEER0 IFER0 IFMCR0-31 IFCPR0 can be assigned to individual PDAC control, then all these registers are only accessible through assigned PDAC.[2] | 32 | Y | Mirrored access to interrupt registers in SIUL2 instance |
| 2240–2A3Fh | MSCR (mirror)[3] | 32 | Y | Mirrored access to MSCRs in SIUL2 instance |
| 2A40–323Fh | IMCR (mirror)[3] | 32 | Y | Mirrored access to IMCRs in SIUL2 instance |
| 3300–34FFh | GPDO (mirror)[4] | 8 | Y | Mirrored access to GPDOs in SIUL2 instance |
| 4000–47FFh | SLBR[5] | 8 | Y | Soft lock bit registers within REG_PROT wrapper for this SIUL2 instance |
| 4800–48FFh | GCR | 32 | Y | Global configuration register within REG_PROT wrapper for this SIUL2 instance. There is a single register in configuration space that is area 4 of REG_PROT. The size mentioned is to make the whole area a multiple of 4 kB. See the table "Address accesses |

*Table continues on the next page...*

Table 32. SIUL2 memory map overview and protection (continued)

| Offset range | Register | Size (bits) | Protected | Description |
|---|---|---|---|---|
| | | | | as a function of PROT_MEM" in the REG_PROT chapter to know the exact area for GCRs. |

1. The SIUL2 block guide specifies the support of a maximum of 512 MSCRs.
2. Access to mirrored register address range results in a change of the related SLBR within the REG_PROT wrapper if REG_PROT protects the register. For registers unprotected by REG_PROT, a write to a register mirror has the same impact as a write to the register.
3. Access to mirrored register address range results in a change of the related SLBR within the REG_PROT wrapper if REG_PROT protects the register. For registers unprotected by REG_PROT, write to register mirror has the same impact as write to register.
4. Access to mirrored register address range results in a change of the related SLBR within the REG_PROT wrapper if REG_PROT protects the register. For registers unprotected by REG_PROT, a write to a register mirror has the same impact as a write to the register.
5. Only SLBRs protecting accesses to the interrupt registers, IMCRs, and the MSCR registers are protected. See the register protection sheet attached to the reference manual for a complete list of SIUL2 registers that REG_PROT protects.

## 8.4 External signals

This module has no external signals.

## 8.5 Initialization

This module does not require initialization.

## 8.6 Application Information

This module supports virtualization of accesses to the registers within a SIUL2 by inhibiting or granting accesses to a subset of these registers under control of the specified initiator(core) information.

## 8.7 VIRT_WRAPPER memory map register descriptions

**NOTE**

Access to reserved spaces outside the register bank and holes (unimplemented registers) within register bank generates the transfer error.

### 8.7.1 VIRT_WRAPPER memory map

VIRT_WRAPPER base address: 402A_8000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Parameter_n Register (REG_A15_0) | 32 | RW | FFFF_FFFFh |
| 4h | Parameter_n Register (REG_A31_16) | 32 | RW | FFFF_FFFFh |
| 8h | Parameter_n Register (REG_A47_32) | 32 | RW | FFFF_FFFFh |
| Ch | Parameter_n Register (REG_A63_48) | 32 | RW | FFFF_FFFFh |
| 10h | Parameter_n Register (REG_A79_64) | 32 | RW | FFFF_FFFFh |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 14h | Parameter_n Register (REG_A95_80) | 32 | RW | FFFF_FFFFh |
| 18h | Parameter_n Register (REG_A111_96) | 32 | RW | FFFF_FFFFh |
| 1Ch | Parameter_n Register (REG_A127_112) | 32 | RW | FFFF_FFFFh |
| 20h | Parameter_n Register (REG_A143_128) | 32 | RW | FFFF_FFFFh |
| 24h | Parameter_n Register (REG_A159_144) | 32 | RW | FFFF_FFFFh |
| 28h | Parameter_n Register (REG_A175_160) | 32 | RW | FFFF_FFFFh |
| 2Ch | Parameter_n Register (REG_A191_176) | 32 | RW | FFFF_FFFFh |
| 30h | Parameter_n Register (REG_A207_192) | 32 | RW | FFFF_FFFFh |
| 34h | Parameter_n Register (REG_A223_208) | 32 | RW | FFFF_FFFFh |
| 38h | Parameter_n Register (REG_A239_224) | 32 | RW | FFFF_FFFFh |
| 3Ch | Parameter_n Register (REG_A255_240) | 32 | RW | FFFF_FFFFh |
| 40h | Parameter_n Register (REG_A271_256) | 32 | RW | FFFF_FFFFh |
| 44h | Parameter_n Register (REG_A287_272) | 32 | RW | FFFF_FFFFh |
| 48h | Parameter_n Register (REG_A303_288) | 32 | RW | FFFF_FFFFh |
| 4Ch | Parameter_n Register (REG_A319_304) | 32 | RW | FFFF_FFFFh |
| 50h | Parameter_n Register (REG_A335_320) | 32 | RW | FFFF_FFFFh |
| 54h | Parameter_n Register (REG_A351_336) | 32 | RW | FFFF_FFFFh |
| 58h | Parameter_n Register (REG_A367_352) | 32 | RW | FFFF_FFFFh |
| 5Ch | Parameter_n Register (REG_A383_368) | 32 | RW | FFFF_FFFFh |
| 60h | Parameter_n Register (REG_A399_384) | 32 | RW | FFFF_FFFFh |
| 64h | Parameter_n Register (REG_A415_400) | 32 | RW | FFFF_FFFFh |
| 68h | Parameter_n Register (REG_A431_416) | 32 | RW | FFFF_FFFFh |
| 6Ch | Parameter_n Register (REG_A447_432) | 32 | RW | FFFF_FFFFh |
| 70h | Parameter_n Register (REG_A463_448) | 32 | RW | FFFF_FFFFh |
| 74h | Parameter_n Register (REG_A479_464) | 32 | RW | FFFF_FFFFh |
| 78h | Parameter_n Register (REG_A495_480) | 32 | RW | FFFF_FFFFh |
| 7Ch | Parameter_n Register (REG_A511_496) | 32 | RW | FFFF_FFFFh |
| 80h | Parameter_n Register (REG_B527_512) | 32 | RW | FFFF_FFFFh |
| 84h | Parameter_n Register (REG_B543_528) | 32 | RW | FFFF_FFFFh |
| 88h | Parameter_n Register (REG_B559_544) | 32 | RW | FFFF_FFFFh |
| 8Ch | Parameter_n Register (REG_B575_560) | 32 | RW | FFFF_FFFFh |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 90h | Parameter_n Register (REG_B591_576) | 32 | RW | FFFF_FFFFh |
| 94h | Parameter_n Register (REG_B607_592) | 32 | RW | FFFF_FFFFh |
| 98h | Parameter_n Register (REG_B623_608) | 32 | RW | FFFF_FFFFh |
| 9Ch | Parameter_n Register (REG_B639_624) | 32 | RW | FFFF_FFFFh |
| A0h | Parameter_n Register (REG_B655_640) | 32 | RW | FFFF_FFFFh |
| A4h | Parameter_n Register (REG_B671_656) | 32 | RW | FFFF_FFFFh |
| A8h | Parameter_n Register (REG_B687_672) | 32 | RW | FFFF_FFFFh |
| ACh | Parameter_n Register (REG_B703_688) | 32 | RW | FFFF_FFFFh |
| B0h | Parameter_n Register (REG_B719_704) | 32 | RW | FFFF_FFFFh |
| B4h | Parameter_n Register (REG_B735_720) | 32 | RW | FFFF_FFFFh |
| B8h | Parameter_n Register (REG_B751_736) | 32 | RW | FFFF_FFFFh |
| BCh | Parameter_n Register (REG_B767_752) | 32 | RW | FFFF_FFFFh |
| C0h | Parameter_n Register (REG_B783_768) | 32 | RW | FFFF_FFFFh |
| C4h | Parameter_n Register (REG_B799_784) | 32 | RW | FFFF_FFFFh |
| C8h | Parameter_n Register (REG_B815_800) | 32 | RW | FFFF_FFFFh |
| CCh | Parameter_n Register (REG_B831_816) | 32 | RW | FFFF_FFFFh |
| D0h | Parameter_n Register (REG_B847_832) | 32 | RW | FFFF_FFFFh |
| D4h | Parameter_n Register (REG_B863_848) | 32 | RW | FFFF_FFFFh |
| D8h | Parameter_n Register (REG_B879_864) | 32 | RW | FFFF_FFFFh |
| DCh | Parameter_n Register (REG_B895_880) | 32 | RW | FFFF_FFFFh |
| E0h | Parameter_n Register (REG_B911_896) | 32 | RW | FFFF_FFFFh |
| E4h | Parameter_n Register (REG_B927_912) | 32 | RW | FFFF_FFFFh |
| E8h | Parameter_n Register (REG_B943_928) | 32 | RW | FFFF_FFFFh |
| ECh | Parameter_n Register (REG_B959_944) | 32 | RW | FFFF_FFFFh |
| F0h | Parameter_n Register (REG_B975_960) | 32 | RW | FFFF_FFFFh |
| F4h | Parameter_n Register (REG_B991_976) | 32 | RW | FFFF_FFFFh |
| F8h | Parameter_n Register (REG_B1007_992) | 32 | RW | FFFF_FFFFh |
| FCh | Parameter_n Register (REG_B1023_1008) | 32 | RW | FFFF_FFFFh |
| 100h | Parameter_n Register (REG_C1039_1024) | 32 | RW | FFFF_FFFFh |
| 104h | Parameter_n Register (REG_D1055_1040) | 32 | RW | FFFF_FFFFh |

### 8.7.2 Parameter_n Register (REG_A15_0 - REG_A511_496)

**Offset**

| Register | Offset |
|---|---|
| REG_A15_0 | 0h |
| REG_A31_16 | 4h |
| REG_A47_32 | 8h |
| REG_A63_48 | Ch |
| REG_A79_64 | 10h |
| REG_A95_80 | 14h |
| REG_A111_96 | 18h |
| REG_A127_112 | 1Ch |
| REG_A143_128 | 20h |
| REG_A159_144 | 24h |
| REG_A175_160 | 28h |
| REG_A191_176 | 2Ch |
| REG_A207_192 | 30h |
| REG_A223_208 | 34h |
| REG_A239_224 | 38h |
| REG_A255_240 | 3Ch |
| REG_A271_256 | 40h |
| REG_A287_272 | 44h |
| REG_A303_288 | 48h |
| REG_A319_304 | 4Ch |
| REG_A335_320 | 50h |
| REG_A351_336 | 54h |
| REG_A367_352 | 58h |
| REG_A383_368 | 5Ch |
| REG_A399_384 | 60h |
| REG_A415_400 | 64h |
| REG_A431_416 | 68h |
| REG_A447_432 | 6Ch |
| REG_A463_448 | 70h |
| REG_A479_464 | 74h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| REG_A495_480 | 78h |
| REG_A511_496 | 7Ch |

### Function

This register set is for PAD0-511. They control MSCR, GPDO, PGPDO, and MPGPDO. Two bits assigned per PDAC have attributes of one of the implemented PDACs:

- 00-SIUL2_VIRTWRAPPER_PDAC1
- 11-SIUL2_VIRTWRAPPER_PDAC0

**NOTE**

- Any initiator can access holes (unimplemented registers) of SIUL2 through PDAC0.
- After reset, when XRDC is not configured, any initiator can access SIUL2 registers through PDAC0 without getting transfer error. However, if SIUL2 registers are accessed through other PDAC slots, a transfer error will be generated.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PAD_15 | | PAD_14 | | PAD_13 | | PAD_12 | | PAD_11 | | PAD_10 | | PAD_9 | | PAD_8 | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PAD_7 | | PAD_6 | | PAD_5 | | PAD_4 | | PAD_3 | | PAD_2 | | PAD_1 | | PAD_0 | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### Fields

| Field | Function |
|---|---|
| 31-30<br>PAD_15 | PAD_15<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 29-28<br>PAD_14 | PAD_14<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 27-26<br>PAD_13 | PAD_13 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 25-24<br>PAD_12 | PAD_12<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 23-22<br>PAD_11 | PAD_11<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 21-20<br>PAD_10 | PAD_10<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 19-18<br>PAD_9 | PAD_9<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 17-16<br>PAD_8 | PAD_8<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 15-14<br>PAD_7 | PAD_7<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 13-12<br>PAD_6 | PAD_6<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 11-10<br>PAD_5 | PAD_5<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 9-8<br>PAD_4 | PAD_4<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 7-6<br>PAD_3 | PAD_3<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 5-4<br>PAD_2 | PAD_2<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 3-2<br>PAD_1 | PAD_1<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 1-0<br>PAD_0 | PAD_0<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |

### 8.7.3 Parameter_n Register (REG_B527_512 - REG_B1023_1008)

**Offset**

| Register | Offset |
|---|---|
| REG_B527_512 | 80h |
| REG_B543_528 | 84h |
| REG_B559_544 | 88h |
| REG_B575_560 | 8Ch |
| REG_B591_576 | 90h |
| REG_B607_592 | 94h |
| REG_B623_608 | 98h |
| REG_B639_624 | 9Ch |
| REG_B655_640 | A0h |
| REG_B671_656 | A4h |
| REG_B687_672 | A8h |
| REG_B703_688 | ACh |
| REG_B719_704 | B0h |
| REG_B735_720 | B4h |
| REG_B751_736 | B8h |
| REG_B767_752 | BCh |
| REG_B783_768 | C0h |
| REG_B799_784 | C4h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| REG_B815_800 | C8h |
| REG_B831_816 | CCh |
| REG_B847_832 | D0h |
| REG_B863_848 | D4h |
| REG_B879_864 | D8h |
| REG_B895_880 | DCh |
| REG_B911_896 | E0h |
| REG_B927_912 | E4h |
| REG_B943_928 | E8h |
| REG_B959_944 | ECh |
| REG_B975_960 | F0h |
| REG_B991_976 | F4h |
| REG_B1007_992 | F8h |
| REG_B1023_1008 | FCh |

### Function

Controls access to pads 0 to 255 specific for IMCR registers. Two bits assigned per IMCR register have attributes of one of the implemented PDACs:

- 00-SIUL2_VIRTWRAPPER_PDAC1
- 11-SIUL2_VIRTWRAPPER_PDAC0

**NOTE**

- Any initiator can access holes (unimplemented registers) of SIUL2 through PDAC0.
- After reset, when XRDC is not configured, any initiator can access SIUL2 registers through PDAC0 without getting transfer error. However, if SIUL2 registers are accessed through other PDAC slots, a transfer error will be generated.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | INMUX_15 | | INMUX_14 | | INMUX_13 | | INMUX_12 | | INMUX_11 | | INMUX_10 | | INMUX_9 | | INMUX_8 | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | INMUX_7 | | INMUX_6 | | INMUX_5 | | INMUX_4 | | INMUX_3 | | INMUX_2 | | INMUX_1 | | INMUX_0 | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### Fields

| Field | Function |
|---|---|
| 31-30<br>INMUX_15 | INMUX_15<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 29-28<br>INMUX_14 | INMUX_14<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 27-26<br>INMUX_13 | INMUX_13<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 25-24<br>INMUX_12 | INMUX_12<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 23-22<br>INMUX_11 | INMUX_11<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 21-20<br>INMUX_10 | INMUX_10<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 19-18<br>INMUX_9 | INMUX_9<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 17-16<br>INMUX_8 | INMUX_8<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 15-14<br>INMUX_7 | INMUX_7<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 13-12<br>INMUX_6 | INMUX_6<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 11-10<br>INMUX_5 | INMUX_5 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 9-8<br>INMUX_4 | INMUX_4<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 7-6<br>INMUX_3 | INMUX_3<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 5-4<br>INMUX_2 | INMUX_2<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 3-2<br>INMUX_1 | INMUX_1<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |
| 1-0<br>INMUX_0 | INMUX_0<br>00b - SIUL2_VIRTWRAPPER_PDAC1<br>11b - SIUL2_VIRTWRAPPER_PDAC0 |

### 8.7.4 Parameter_n Register (REG_C1039_1024)

#### Offset

| Register | Offset |
|---|---|
| REG_C1039_1024 | 100h |

#### Function

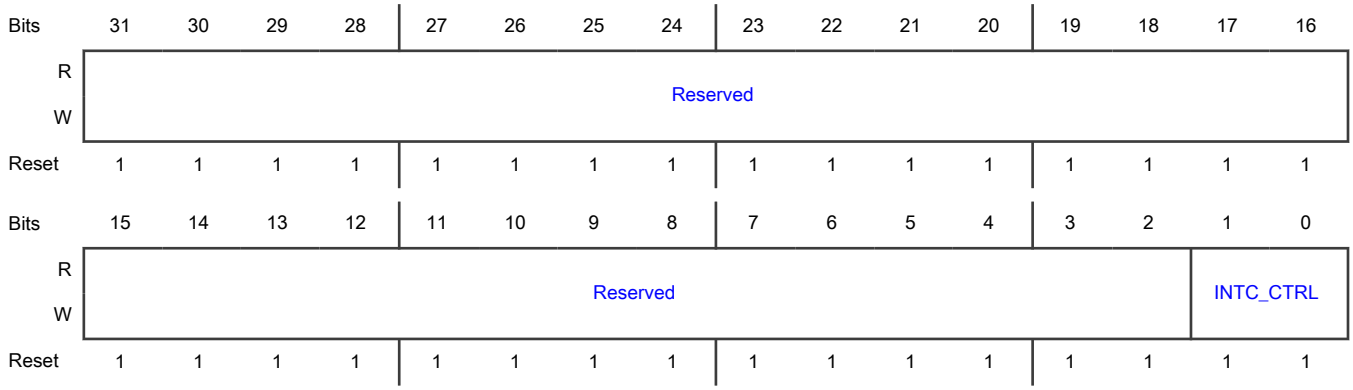
Controls access to DISR0, DIRER0, DIRSR0, IREER0, IFEER0, IFER0, IFMCR, and IFCPR0 interrupt registers. Two bits assigned per PDAC have attributes of one of the implemented PDACs:

- 00-SIUL2_VIRTWRAPPER_PDAC1
- 11-SIUL2_VIRTWRAPPER_PDAC0

**NOTE**

After reset, when XRDC is not configured, any initiator can access SIUL2 registers through PDAC0 without getting transfer error. However, if SIUL2 registers are accessed through other PDAC slots, a transfer error will be generated.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1-0<br>INTC_CTRL | Interrupt register control<br>This bit controls the interrupt register. |

## 8.7.5 Parameter_n Register (REG_D1055_1040)

### Offset

| Register | Offset |
|---|---|
| REG_D1055_1040 | 104h |

### Function

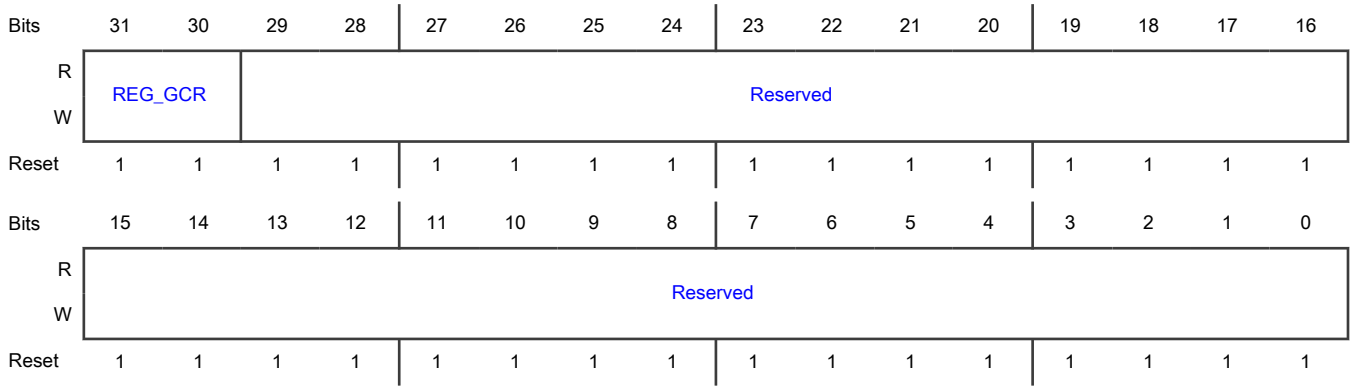
Controls access to GCR protection wrapper registers. Two bits assigned per PDAC have attributes of one of the implemented PDACs:

- 00-SIUL2_VIRTWRAPPER_PDAC1
- 11-SIUL2_VIRTWRAPPER_PDAC0

**NOTE**

After reset, when XRDC is not configured, any initiator can access SIUL2 registers through PDAC0 without getting transfer error. However, if SIUL2 registers are accessed through other PDAC slots, a transfer error will be generated.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>REG_GCR | GCR Register Of REG_PROT<br>Controls access to GCR protection wrapper registers. |
| 29-0<br>— | Reserved |

## 8.8 Glossary

**VM** Virtualized module, such as SIUL2.

**PDAC** Peripheral domain access control. PDAC and PDAC slot are interchangeable terms.

# Chapter 9
# System Integration Unit Lite2 (SIUL2)

## 9.1 Chip-specific SIUL2 information

### 9.1.1 Feature availability

This chip:

- Supports input filter enable (MSCR5[IFE]) only for the reset pad (PTA5). For details, see the "Pad signal description" table in the "Signal Multiplexing" chapter.
- Does not implement the open-drain feature. LPI2C directly configures only the pads related to the I2C and LPUART functions in pseudo open-drain when these functions are muxed to the pads. See the LPI2C and LPUART chapters for more information.
- Reserves GPDO[25:24] and PGPDO1[7:6] because PTA24 and PTA25 are input-only pins.

**NOTE**

EIRQ[0-15] can be used either for interrupt or DMA request. EIRQ[16-31] can only be used for interrupt request.

### 9.1.2 Mapping of MSCR and IMCR instances

- CR numbers 0-511 correspond to the MSCR instances.
- CR numbers 512-1023 correspond to the IMCR instances. IMCRs defined in the attached IOMUX file have an offset of 512 with respect to the IMCR number defined in SIUL2 memory map section.

**NOTE**

IMCR register only supports 32 bit access, any other access might result in unexpected data. See IOMUX file attached to this document for the exact number of IMCR and MSCR register instances.

### 9.1.3 SIUL2 register configuration

There are additional registers as compared to what is mentioned in section 'SIUL2 memory map'. See following table for details.

**Table 33. Registers and applicability**

| Register | Offset | MCXE315/MCXE316 | MCXE317 | MCXE31B |
|---|---|---|---|---|
| MUX0_EMIOS_EN1 | 100h | Yes | No | No |
| MUX0_MISC_EN | 104h | Yes | No | No |
| MUX1_EMIOS_EN | 108h | Yes | No | No |
| MUX1_MISC_EN | 10Ch | Yes | No | No |
| UDR1 | 100h | No | Yes | No |
| UDR0 | 104h | No | Yes | No |
| UDR3 | 108h | No | Yes | No |
| UDR2 | 10Ch | No | Yes | No |

## 9.2 Overview

SIUL2 provides control over all electrical pin controls and ports with 16 bits of bidirectional, general-purpose input and output (I/O) signals. It also performs the following functions:

- Enables you to select the functions and electrical characteristics that appear on external chip pins
- Controls the multiplexing of internal signals from one module to another and controls the chip I/O
- Supports a maximum of 32 external interrupts with trigger event configuration

### 9.2.1 Block diagram

SIUL2 provides a dedicated pad control to general-purpose pads that you can configure as either inputs or outputs. It provides registers for you to read values from GPIO pads configured as inputs and to write values to GPIO pads configured as outputs. Based on the configuration of GPIO, you can:

- Write to an internal register to control the state driven on the associated output pad (when configured as output).
- Detect the state of the associated pad by reading the value from an internal register (when configured as input).
- Read back the pad value to check whether the written value appears on the pad (when configured as input and output).

Access the GPIO data registers in the following ways to allow port access and bit manipulation without read-modify-write operations:

- Access to two 16-bit ports in one access
- Read/write access to a single bit
- 16-bit port write with a bit mask using a single 32-bit access

You can configure:

- External interrupt sources at the chip level to be used with any chip pad.
- Interrupt sources to have a digital filter to reject short glitches on the inputs.

The external interrupt or DMA requests map to the interrupt request pins (REQ) in the chip packages.

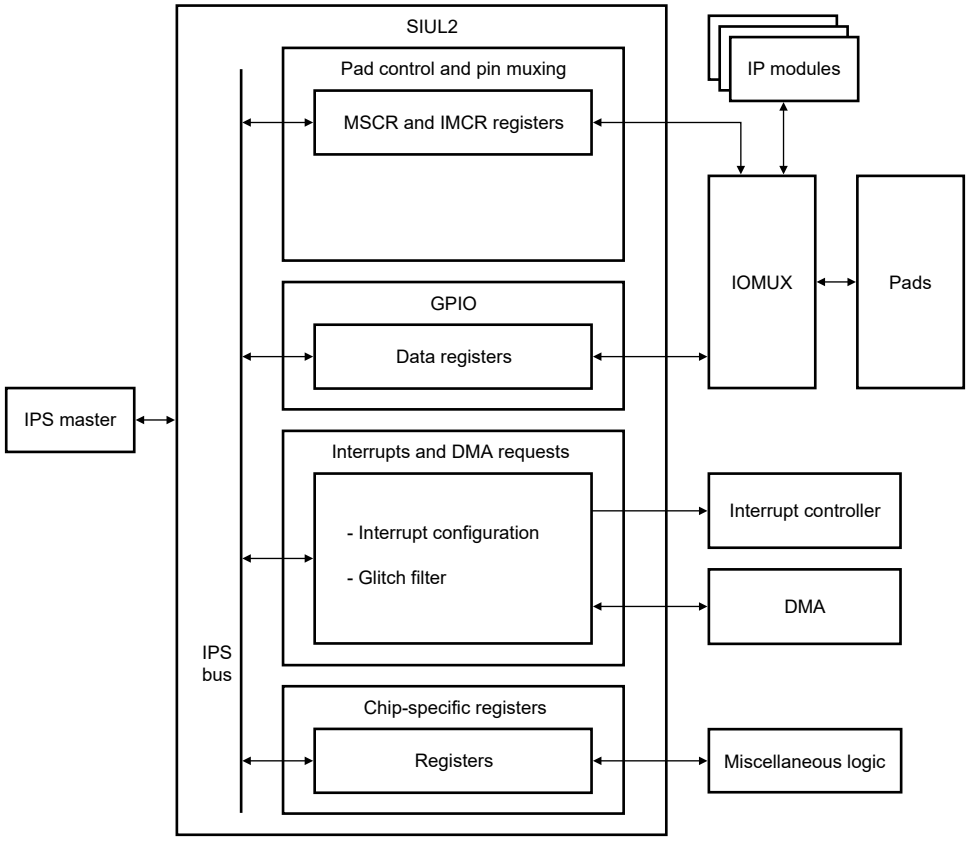



Figure 16. Block diagram

## 9.2.2 Features

SIUL2 supports the following:

- 1 to 32 GPIO ports with data control:
  - Drives data to as many as 16 independent I/O channels
  - Samples data from as many as 16 independent I/O channels
- Read or write of two 16-bit registers with one access for a 32-bit port
- External interrupt and DMA requests:
  - 1 to 32 programmable digital glitch filters, one for each interrupt REQ pin
  - Edge detection
- Multiplexed Signal Configuration Registers (MSCR) to configure the electrical parameters and settings for as many as 512 functional pads.

See the interrupt map file and the DMAMUX map file attached to this document for mapping of interrupt and DMA sources to interrupt vectors and DMA channels.

## 9.3 Functional description

### 9.3.1 Pad control

SIUL2 controls the electrical characteristics of around 512 pads. It provides a consistent interface for all pads, both on a by-port and a by-bit basis.

The setting of each pad out of reset is fixed per chip but you can configure it individually. This way you can select special pull settings or peripheral pad ownership.

You can configure each pad independently of all other pads on the chip or other pads grouped within a single port, therefore allowing grouping of different pad types together in ports and operating the pads individually. Grouping the various functions for each pad into a single register allows configuration of each pad with a single write to a register, which further allows you to duplicate software for similar pads with index changes.

### 9.3.2 General-purpose input and output pads (GPIO)

SIUL2 allows each pad to be configured as either of the following:

- General-purpose input or output pad (GPIO)
- A pad for one or more alternate functions (input or output) determined by the peripheral that uses the pad

You can also implement the GPIO pads without any alternate function.

SIUL2 manages as many as 512 GPIO pads organized as ports that can be accessed for data reads and writes as 8-bit, 16-bit, or 32-bit.

All port accesses are identical, with each read or write performed only at a different location to access a different port width.

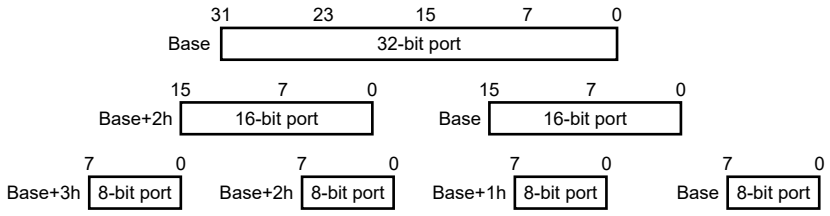



**Figure 17. Data port example arrangement showing configuration for different port width accesses**

SIUL2 has separate data input and data output registers for all pads. You can therefore directly read back an input or output value of a pad to validate what is present on the pad instead of confirming the value that was written to the data input registers. The data output registers support both read and write operations whereas the data input registers support only the read access.

When you configure a pad for using one of its alternate functions, the data input values reflect the respective value of the pad. If you perform a write operation to the data output register for a pad configured as an alternate function (non-GPIO), this write is not reflected by the pad value until reconfigured to GPIO. All general-purpose pads are implemented as bidirectional.

If a bidirectional operation impacts the performance or is not needed for a pad function, you can limit the functionality of the pad to input-only.

### 9.3.3 Clocking

This module has no clocking considerations.

### 9.3.4 External interrupts

SIUL2 supports 1–32 external interrupts allocated to pads by the chip.

See the interrupt map file and the DMAMUX map file attached to this document (device reference manual) for mapping of interrupt and DMA sources to interrupt vectors and DMA channels.

SIUL2 supports 1–4 interrupt vectors to the interrupt controller of the chip. Each interrupt vector can support as many as eight external interrupt sources from the chip pads.

All the external interrupt pads within a single group have equal priority. It is your responsibility to search for the application through a group of sources in an appropriate way.

**NOTE**

Glitch filters applied to external interrupts require a running internal oscillator clock. If such a clock is not available, enabling the glitch filter on an external interrupt disables the interrupt.

The external interrupt signals from a pad have internal synchronizers. Therefore, the width of the interrupt signals should be at least 2.5 or 3 times the internal RC oscillator (IRC) clock cycles to correctly capture the interrupts.

#### 9.3.4.1 External interrupt initialization

Perform the following procedure to enable external interrupts (if you do not perform these steps, you may get a false interrupt flag during interrupt initialization):

1. Write 1 to the appropriate IFER[IFE*n*] fields to enable the glitch filter.
2. Write 0 to DIRER0[EIRE*n*] to mask interrupts.
3. Write 1 to the appropriate IREE*n* fields in Interrupt Rising-Edge Event Enable 0 (IREER0) and IFEE*n* fields in Interrupt Falling-Edge Event Enable 0 (IFEER0) as needed to select the pin polarity.
4. Configure the appropriate fields in MSCR for the external interrupt pins:
    a. Write 0 to the OBE and ODE to disable the output.
    b. Write 1 to IBE field to enable the input buffer of the pin.
    c. If you are using the internal pull-up or pull-down resistors, configure the appropriate PUE and PUS fields.

    **NOTE**

    If you select external interrupt inputs for external interrupt pins, do not configure them as outputs (that is, MSCR[OBE] must not be 0) because it can cause false interrupt detection (such as from a GPIO configuration).
5. Write to the appropriate DIRSR0[DIRS*n*] fields to select a request between DMA or interrupt.
6. Select the desired glitch filter setup for the pins:
    a. Write an appropriate value to IFMCR0[MAXCNT] for the respective external interrupt to the filter counter.
    b.
    c. Write to the appropriate IFER0[IFE*n*] fields to enable the glitch filter for the external interrupt pins.
7. Set the appropriate DISR0[EIF*n*] to clear any flags.
8. Write to the appropriate DIRER0[EIRE*n*] fields to enable the interrupt pins.

**NOTE**

If you do not follow these steps you may get a false interrupt flag during interrupt initialization.

#### 9.3.4.2 External interrupt management

You can enable or disable each interrupt independently using a single rolled up register, DIRER0.

You can configure a pad defined as an external interrupt to recognize interrupts with an active rising edge, an active falling edge, or both, using Interrupt Rising-Edge Event Enable 0 (IREER0) and Interrupt Falling-Edge Event Enable 0 (IFEER0).

**NOTE**

You cannot disable both edge events of a given interrupt.

Each external interrupt has an individual flag, held in DMA or Interrupt Status Flag 0 (DISR0). DISR0 is a clear-by-write-1 register, which prevents inadvertent overwriting of other flags in the same register.

Figure 18 provides an overview of the external interrupt implementation.

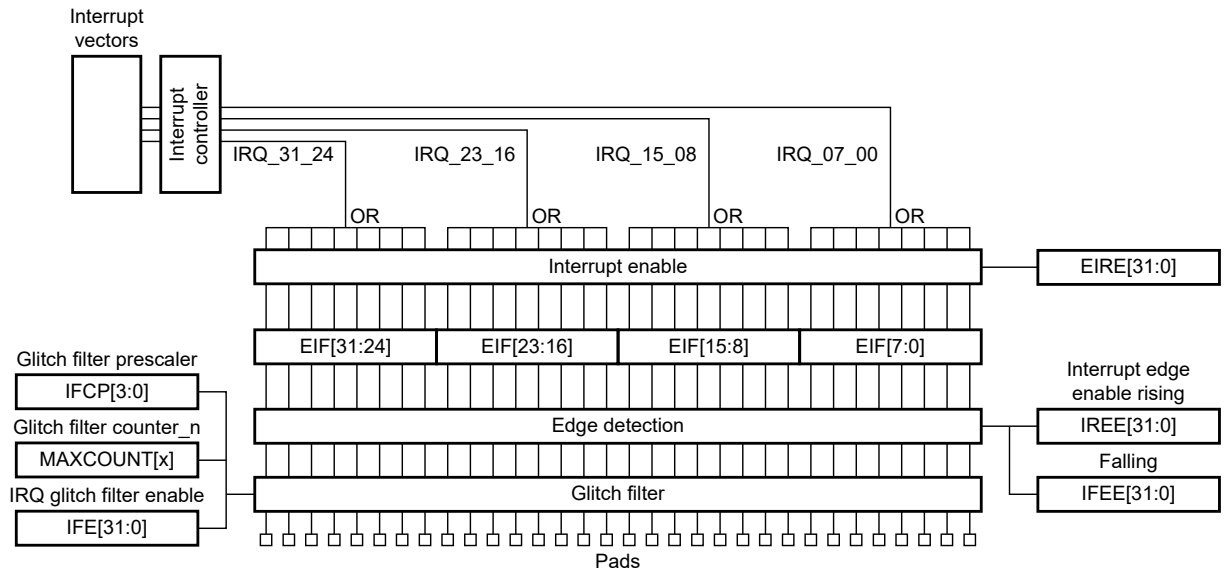



**Figure 18. External interrupt pad diagram**

### 9.3.4.3 External interrupt request

The REQ input pins on the chip are the sources for interrupt or DMA requests. The chip provides one possible interrupt vector for SIUL2. The 32 interrupt request sources map to vectors and channels as shown in Table 34.

**Table 34. Interrupt source mapping to SIUL2 interrupt request output for 32 interrupt sources**

| Vector or channel # | Interrupt vector source |
|---|---|
| 0 | REQ[07] \| REQ[06] \| REQ[05] \| REQ[04] \| REQ[03] \| REQ[02] \| REQ[01] \| REQ[00] |
| 1 | REQ[15] \| REQ[14] \| REQ[13] \| REQ[12] \| REQ[11] \| REQ[10] \| REQ[09] \| REQ[08] |
| 2 | REQ[23] \| REQ[22] \| REQ[21] \| REQ[20] \| REQ[19] \| REQ[18] \| REQ[17] \| REQ[16] |
| 3 | REQ[31] \| REQ[30] \| REQ[29] \| REQ[28] \| REQ[27] \| REQ[26] \| REQ[25] \| REQ[24] |

## 9.3.5 DMA requests

The REQ pins on the chip map to the independent DMA request channels in the DMA controller. See the DMAMUX map file attached to this document for the mapping of DMA sources to DMA channels.

DISR0 and DIRSR0 manage DMA requests in the following way:

- Service a DMA request to clear the DISR0 register flags.
- Write to the appropriate DIRSR0[DIRS*n*] to select a request between DMA or interrupt.
- If you select DMA in DIRSR0 and set the corresponding DISR0 flag for it, SIUL2 sends a DMA request signal as an output.

## 9.4 External signal description

See the IOMUX file attached to this document for more information.

## 9.5 Initialization

This module does not require initialization.

## 9.6 SIUL2 register descriptions

This section describes the SIUL2 registers.

- Undocumented register spaces in the SIUL2 memory map, including addresses shown as blanks, are reserved:
  - Reserved registers or spaces are read as 0.
  - Writes to reserved registers or spaces generate a transfer error.
- Writes to read-only registers generate a transfer error.

**NOTE**

- For the array of 8-bit registers, GPDO*n* and GPDI*n*:
  - An 8-bit access to an unimplemented address (a "hole") within the array region generates a transfer error.
  - However, when you perform a 16-bit or 32-bit access and if any register instance is implemented within the accessed range, a transfer error is not generated even if the range includes a hole.
- For the array of 16-bit registers, PGPDO*n* and PGPDI*n*:
  - A 16-bit access to an unimplemented address (a "hole") within the array region generates a transfer error.
  - However, a 32-bit access does not generate a transfer error for a hole irrespective of whether the other 16-bit range includes a register instance.

### 9.6.1 SIUL2 memory map

SIUL2 base address: 4029_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 4h | SIUL2 MCU ID Register #1 (MIDR1) | 32 | R | See section |
| 8h | SIUL2 MCU ID Register #2 (MIDR2) | 32 | R | See section |
| 10h | DMA or Interrupt Status Flag 0 (DISR0) | 32 | RW | 0000_0000h |
| 18h | DMA or Interrupt Request Enable 0 (DIRER0) | 32 | RW | 0000_0000h |
| 20h | DMA or Interrupt Request Select 0 (DIRSR0) | 32 | RW | 0000_0000h |
| 28h | Interrupt Rising-Edge Event Enable 0 (IREER0) | 32 | RW | 0000_0000h |
| 30h | Interrupt Falling-Edge Event Enable 0 (IFEER0) | 32 | RW | 0000_0000h |
| 38h | Interrupt Filter Enable 0 (IFER0) | 32 | RW | 0000_0000h |
| 40h - BCh | Interrupt Filter Maximum Counter (IFMCR0 - IFMCR31) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 200h | SIUL2 MCU ID Register #3 (MIDR3) | 32 | R | See section |
| 204h | SIUL2 MCU ID Register #4 (MIDR4) | 32 | R | See section |
| 240h - 74Ch | Multiplexed Signal Configuration (MSCR0 - MSCR323) | 32 | RW | See section |
| A40h - 11A4h | Input Multiplexed Signal Configuration (IMCR0 - IMCR473) | 32 | RW | 0000_0000h |
| 1300h | GPIO Pad Data Output (GPDO3) | 8 | RW | 00h |
| 1301h | GPIO Pad Data Output (GPDO2) | 8 | RW | 00h |
| 1302h | GPIO Pad Data Output (GPDO1) | 8 | RW | 00h |
| 1303h | GPIO Pad Data Output (GPDO0) | 8 | RW | 00h |
| 1304h | GPIO Pad Data Output (GPDO7) | 8 | RW | 00h |
| 1305h | GPIO Pad Data Output (GPDO6) | 8 | RW | 00h |
| 1306h | GPIO Pad Data Output (GPDO5) | 8 | RW | 00h |
| 1307h | GPIO Pad Data Output (GPDO4) | 8 | RW | 00h |
| 1308h | GPIO Pad Data Output (GPDO11) | 8 | RW | 00h |
| 1309h | GPIO Pad Data Output (GPDO10) | 8 | RW | 00h |
| 130Ah | GPIO Pad Data Output (GPDO9) | 8 | RW | 00h |
| 130Bh | GPIO Pad Data Output (GPDO8) | 8 | RW | 00h |
| 130Ch | GPIO Pad Data Output (GPDO15) | 8 | RW | 00h |
| 130Dh | GPIO Pad Data Output (GPDO14) | 8 | RW | 00h |
| 130Eh | GPIO Pad Data Output (GPDO13) | 8 | RW | 00h |
| 130Fh | GPIO Pad Data Output (GPDO12) | 8 | RW | 00h |
| 1310h | GPIO Pad Data Output (GPDO19) | 8 | RW | 00h |
| 1311h | GPIO Pad Data Output (GPDO18) | 8 | RW | 00h |
| 1312h | GPIO Pad Data Output (GPDO17) | 8 | RW | 00h |
| 1313h | GPIO Pad Data Output (GPDO16) | 8 | RW | 00h |
| 1314h | GPIO Pad Data Output (GPDO23) | 8 | RW | 00h |
| 1315h | GPIO Pad Data Output (GPDO22) | 8 | RW | 00h |
| 1316h | GPIO Pad Data Output (GPDO21) | 8 | RW | 00h |
| 1317h | GPIO Pad Data Output (GPDO20) | 8 | RW | 00h |
| 1318h | GPIO Pad Data Output (GPDO27) | 8 | RW | 00h |
| 1319h | GPIO Pad Data Output (GPDO26) | 8 | RW | 00h |
| 131Ah | GPIO Pad Data Output (GPDO25) | 8 | RW | 00h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 131Bh | GPIO Pad Data Output (GPDO24) | 8 | RW | 00h |
| 131Ch | GPIO Pad Data Output (GPDO31) | 8 | RW | 00h |
| 131Dh | GPIO Pad Data Output (GPDO30) | 8 | RW | 00h |
| 131Eh | GPIO Pad Data Output (GPDO29) | 8 | RW | 00h |
| 131Fh | GPIO Pad Data Output (GPDO28) | 8 | RW | 00h |
| 1320h | GPIO Pad Data Output (GPDO35) | 8 | RW | 00h |
| 1321h | GPIO Pad Data Output (GPDO34) | 8 | RW | 00h |
| 1322h | GPIO Pad Data Output (GPDO33) | 8 | RW | 00h |
| 1323h | GPIO Pad Data Output (GPDO32) | 8 | RW | 00h |
| 1326h | GPIO Pad Data Output (GPDO37) | 8 | RW | 00h |
| 1327h | GPIO Pad Data Output (GPDO36) | 8 | RW | 00h |
| 1328h | GPIO Pad Data Output (GPDO43) | 8 | RW | 00h |
| 1329h | GPIO Pad Data Output (GPDO42) | 8 | RW | 00h |
| 132Ah | GPIO Pad Data Output (GPDO41) | 8 | RW | 00h |
| 132Bh | GPIO Pad Data Output (GPDO40) | 8 | RW | 00h |
| 132Ch | GPIO Pad Data Output (GPDO47) | 8 | RW | 00h |
| 132Dh | GPIO Pad Data Output (GPDO46) | 8 | RW | 00h |
| 132Eh | GPIO Pad Data Output (GPDO45) | 8 | RW | 00h |
| 132Fh | GPIO Pad Data Output (GPDO44) | 8 | RW | 00h |
| 1330h | GPIO Pad Data Output (GPDO51) | 8 | RW | 00h |
| 1331h | GPIO Pad Data Output (GPDO50) | 8 | RW | 00h |
| 1332h | GPIO Pad Data Output (GPDO49) | 8 | RW | 00h |
| 1333h | GPIO Pad Data Output (GPDO48) | 8 | RW | 00h |
| 1334h | GPIO Pad Data Output (GPDO55) | 8 | RW | 00h |
| 1335h | GPIO Pad Data Output (GPDO54) | 8 | RW | 00h |
| 1336h | GPIO Pad Data Output (GPDO53) | 8 | RW | 00h |
| 1337h | GPIO Pad Data Output (GPDO52) | 8 | RW | 00h |
| 1338h | GPIO Pad Data Output (GPDO59) | 8 | RW | 00h |
| 1339h | GPIO Pad Data Output (GPDO58) | 8 | RW | 00h |
| 133Ah | GPIO Pad Data Output (GPDO57) | 8 | RW | 00h |
| 133Bh | GPIO Pad Data Output (GPDO56) | 8 | RW | 00h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 133Ch | GPIO Pad Data Output (GPDO63) | 8 | RW | 00h |
| 133Dh | GPIO Pad Data Output (GPDO62) | 8 | RW | 00h |
| 133Eh | GPIO Pad Data Output (GPDO61) | 8 | RW | 00h |
| 133Fh | GPIO Pad Data Output (GPDO60) | 8 | RW | 00h |
| 1340h | GPIO Pad Data Output (GPDO67) | 8 | RW | 00h |
| 1341h | GPIO Pad Data Output (GPDO66) | 8 | RW | 00h |
| 1342h | GPIO Pad Data Output (GPDO65) | 8 | RW | 00h |
| 1343h | GPIO Pad Data Output (GPDO64) | 8 | RW | 00h |
| 1344h | GPIO Pad Data Output (GPDO71) | 8 | RW | 00h |
| 1345h | GPIO Pad Data Output (GPDO70) | 8 | RW | 00h |
| 1346h | GPIO Pad Data Output (GPDO69) | 8 | RW | 00h |
| 1347h | GPIO Pad Data Output (GPDO68) | 8 | RW | 00h |
| 1348h | GPIO Pad Data Output (GPDO75) | 8 | RW | 00h |
| 1349h | GPIO Pad Data Output (GPDO74) | 8 | RW | 00h |
| 134Ah | GPIO Pad Data Output (GPDO73) | 8 | RW | 00h |
| 134Bh | GPIO Pad Data Output (GPDO72) | 8 | RW | 00h |
| 134Ch | GPIO Pad Data Output (GPDO79) | 8 | RW | 00h |
| 134Dh | GPIO Pad Data Output (GPDO78) | 8 | RW | 00h |
| 134Eh | GPIO Pad Data Output (GPDO77) | 8 | RW | 00h |
| 134Fh | GPIO Pad Data Output (GPDO76) | 8 | RW | 00h |
| 1350h | GPIO Pad Data Output (GPDO83) | 8 | RW | 00h |
| 1351h | GPIO Pad Data Output (GPDO82) | 8 | RW | 00h |
| 1352h | GPIO Pad Data Output (GPDO81) | 8 | RW | 00h |
| 1353h | GPIO Pad Data Output (GPDO80) | 8 | RW | 00h |
| 1354h | GPIO Pad Data Output (GPDO87) | 8 | RW | 00h |
| 1355h | GPIO Pad Data Output (GPDO86) | 8 | RW | 00h |
| 1356h | GPIO Pad Data Output (GPDO85) | 8 | RW | 00h |
| 1357h | GPIO Pad Data Output (GPDO84) | 8 | RW | 00h |
| 1358h | GPIO Pad Data Output (GPDO91) | 8 | RW | 00h |
| 1359h | GPIO Pad Data Output (GPDO90) | 8 | RW | 00h |
| 135Ah | GPIO Pad Data Output (GPDO89) | 8 | RW | 00h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 135Bh | GPIO Pad Data Output (GPDO88) | 8 | RW | 00h |
| 135Ch | GPIO Pad Data Output (GPDO95) | 8 | RW | 00h |
| 135Dh | GPIO Pad Data Output (GPDO94) | 8 | RW | 00h |
| 135Eh | GPIO Pad Data Output (GPDO93) | 8 | RW | 00h |
| 135Fh | GPIO Pad Data Output (GPDO92) | 8 | RW | 00h |
| 1360h | GPIO Pad Data Output (GPDO99) | 8 | RW | 00h |
| 1361h | GPIO Pad Data Output (GPDO98) | 8 | RW | 00h |
| 1362h | GPIO Pad Data Output (GPDO97) | 8 | RW | 00h |
| 1363h | GPIO Pad Data Output (GPDO96) | 8 | RW | 00h |
| 1364h | GPIO Pad Data Output (GPDO103) | 8 | RW | 00h |
| 1365h | GPIO Pad Data Output (GPDO102) | 8 | RW | 00h |
| 1366h | GPIO Pad Data Output (GPDO101) | 8 | RW | 00h |
| 1367h | GPIO Pad Data Output (GPDO100) | 8 | RW | 00h |
| 1368h | GPIO Pad Data Output (GPDO107) | 8 | RW | 00h |
| 1369h | GPIO Pad Data Output (GPDO106) | 8 | RW | 00h |
| 136Ah | GPIO Pad Data Output (GPDO105) | 8 | RW | 00h |
| 136Bh | GPIO Pad Data Output (GPDO104) | 8 | RW | 00h |
| 136Ch | GPIO Pad Data Output (GPDO111) | 8 | RW | 00h |
| 136Dh | GPIO Pad Data Output (GPDO110) | 8 | RW | 00h |
| 136Eh | GPIO Pad Data Output (GPDO109) | 8 | RW | 00h |
| 136Fh | GPIO Pad Data Output (GPDO108) | 8 | RW | 00h |
| 1370h | GPIO Pad Data Output (GPDO115) | 8 | RW | 00h |
| 1371h | GPIO Pad Data Output (GPDO114) | 8 | RW | 00h |
| 1372h | GPIO Pad Data Output (GPDO113) | 8 | RW | 00h |
| 1373h | GPIO Pad Data Output (GPDO112) | 8 | RW | 00h |
| 1374h | GPIO Pad Data Output (GPDO119) | 8 | RW | 00h |
| 1375h | GPIO Pad Data Output (GPDO118) | 8 | RW | 00h |
| 1376h | GPIO Pad Data Output (GPDO117) | 8 | RW | 00h |
| 1377h | GPIO Pad Data Output (GPDO116) | 8 | RW | 00h |
| 1378h | GPIO Pad Data Output (GPDO123) | 8 | RW | 00h |
| 1379h | GPIO Pad Data Output (GPDO122) | 8 | RW | 00h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 137Ah | GPIO Pad Data Output (GPDO121) | 8 | RW | 00h |
| 137Bh | GPIO Pad Data Output (GPDO120) | 8 | RW | 00h |
| 137Ch | GPIO Pad Data Output (GPDO127) | 8 | RW | 00h |
| 137Dh | GPIO Pad Data Output (GPDO126) | 8 | RW | 00h |
| 137Eh | GPIO Pad Data Output (GPDO125) | 8 | RW | 00h |
| 137Fh | GPIO Pad Data Output (GPDO124) | 8 | RW | 00h |
| 1380h | GPIO Pad Data Output (GPDO131) | 8 | RW | 00h |
| 1381h | GPIO Pad Data Output (GPDO130) | 8 | RW | 00h |
| 1382h | GPIO Pad Data Output (GPDO129) | 8 | RW | 00h |
| 1383h | GPIO Pad Data Output (GPDO128) | 8 | RW | 00h |
| 1384h | GPIO Pad Data Output (GPDO135) | 8 | RW | 00h |
| 1385h | GPIO Pad Data Output (GPDO134) | 8 | RW | 00h |
| 1386h | GPIO Pad Data Output (GPDO133) | 8 | RW | 00h |
| 1387h | GPIO Pad Data Output (GPDO132) | 8 | RW | 00h |
| 1388h | GPIO Pad Data Output (GPDO139) | 8 | RW | 00h |
| 1389h | GPIO Pad Data Output (GPDO138) | 8 | RW | 00h |
| 138Ah | GPIO Pad Data Output (GPDO137) | 8 | RW | 00h |
| 138Bh | GPIO Pad Data Output (GPDO136) | 8 | RW | 00h |
| 138Ch | GPIO Pad Data Output (GPDO143) | 8 | RW | 00h |
| 138Dh | GPIO Pad Data Output (GPDO142) | 8 | RW | 00h |
| 138Eh | GPIO Pad Data Output (GPDO141) | 8 | RW | 00h |
| 138Fh | GPIO Pad Data Output (GPDO140) | 8 | RW | 00h |
| 1390h | GPIO Pad Data Output (GPDO147) | 8 | RW | 00h |
| 1391h | GPIO Pad Data Output (GPDO146) | 8 | RW | 00h |
| 1392h | GPIO Pad Data Output (GPDO145) | 8 | RW | 00h |
| 1393h | GPIO Pad Data Output (GPDO144) | 8 | RW | 00h |
| 1394h | GPIO Pad Data Output (GPDO151) | 8 | RW | 00h |
| 1395h | GPIO Pad Data Output (GPDO150) | 8 | RW | 00h |
| 1396h | GPIO Pad Data Output (GPDO149) | 8 | RW | 00h |
| 1397h | GPIO Pad Data Output (GPDO148) | 8 | RW | 00h |
| 1398h | GPIO Pad Data Output (GPDO155) | 8 | RW | 00h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1399h | GPIO Pad Data Output (GPDO154) | 8 | RW | 00h |
| 139Ah | GPIO Pad Data Output (GPDO153) | 8 | RW | 00h |
| 139Bh | GPIO Pad Data Output (GPDO152) | 8 | RW | 00h |
| 139Ch | GPIO Pad Data Output (GPDO159) | 8 | RW | 00h |
| 139Dh | GPIO Pad Data Output (GPDO158) | 8 | RW | 00h |
| 139Eh | GPIO Pad Data Output (GPDO157) | 8 | RW | 00h |
| 139Fh | GPIO Pad Data Output (GPDO156) | 8 | RW | 00h |
| 13A0h | GPIO Pad Data Output (GPDO163) | 8 | RW | 00h |
| 13A1h | GPIO Pad Data Output (GPDO162) | 8 | RW | 00h |
| 13A2h | GPIO Pad Data Output (GPDO161) | 8 | RW | 00h |
| 13A3h | GPIO Pad Data Output (GPDO160) | 8 | RW | 00h |
| 13A4h | GPIO Pad Data Output (GPDO167) | 8 | RW | 00h |
| 13A5h | GPIO Pad Data Output (GPDO166) | 8 | RW | 00h |
| 13A6h | GPIO Pad Data Output (GPDO165) | 8 | RW | 00h |
| 13A7h | GPIO Pad Data Output (GPDO164) | 8 | RW | 00h |
| 13A8h | GPIO Pad Data Output (GPDO171) | 8 | RW | 00h |
| 13A9h | GPIO Pad Data Output (GPDO170) | 8 | RW | 00h |
| 13AAh | GPIO Pad Data Output (GPDO169) | 8 | RW | 00h |
| 13ABh | GPIO Pad Data Output (GPDO168) | 8 | RW | 00h |
| 13ACh | GPIO Pad Data Output (GPDO175) | 8 | RW | 00h |
| 13ADh | GPIO Pad Data Output (GPDO174) | 8 | RW | 00h |
| 13AEh | GPIO Pad Data Output (GPDO173) | 8 | RW | 00h |
| 13AFh | GPIO Pad Data Output (GPDO172) | 8 | RW | 00h |
| 13B0h | GPIO Pad Data Output (GPDO179) | 8 | RW | 00h |
| 13B1h | GPIO Pad Data Output (GPDO178) | 8 | RW | 00h |
| 13B2h | GPIO Pad Data Output (GPDO177) | 8 | RW | 00h |
| 13B3h | GPIO Pad Data Output (GPDO176) | 8 | RW | 00h |
| 13B4h | GPIO Pad Data Output (GPDO183) | 8 | RW | 00h |
| 13B5h | GPIO Pad Data Output (GPDO182) | 8 | RW | 00h |
| 13B6h | GPIO Pad Data Output (GPDO181) | 8 | RW | 00h |
| 13B7h | GPIO Pad Data Output (GPDO180) | 8 | RW | 00h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 13B8h | GPIO Pad Data Output (GPDO187) | 8 | RW | 00h |
| 13B9h | GPIO Pad Data Output (GPDO186) | 8 | RW | 00h |
| 13BAh | GPIO Pad Data Output (GPDO185) | 8 | RW | 00h |
| 13BBh | GPIO Pad Data Output (GPDO184) | 8 | RW | 00h |
| 13BCh | GPIO Pad Data Output (GPDO191) | 8 | RW | 00h |
| 13BDh | GPIO Pad Data Output (GPDO190) | 8 | RW | 00h |
| 13BEh | GPIO Pad Data Output (GPDO189) | 8 | RW | 00h |
| 13BFh | GPIO Pad Data Output (GPDO188) | 8 | RW | 00h |
| 13C0h | GPIO Pad Data Output (GPDO195) | 8 | RW | 00h |
| 13C1h | GPIO Pad Data Output (GPDO194) | 8 | RW | 00h |
| 13C2h | GPIO Pad Data Output (GPDO193) | 8 | RW | 00h |
| 13C3h | GPIO Pad Data Output (GPDO192) | 8 | RW | 00h |
| 13C4h | GPIO Pad Data Output (GPDO199) | 8 | RW | 00h |
| 13C5h | GPIO Pad Data Output (GPDO198) | 8 | RW | 00h |
| 13C6h | GPIO Pad Data Output (GPDO197) | 8 | RW | 00h |
| 13C7h | GPIO Pad Data Output (GPDO196) | 8 | RW | 00h |
| 13C8h | GPIO Pad Data Output (GPDO203) | 8 | RW | 00h |
| 13C9h | GPIO Pad Data Output (GPDO202) | 8 | RW | 00h |
| 13CAh | GPIO Pad Data Output (GPDO201) | 8 | RW | 00h |
| 13CBh | GPIO Pad Data Output (GPDO200) | 8 | RW | 00h |
| 13CCh | GPIO Pad Data Output (GPDO207) | 8 | RW | 00h |
| 13CDh | GPIO Pad Data Output (GPDO206) | 8 | RW | 00h |
| 13CEh | GPIO Pad Data Output (GPDO205) | 8 | RW | 00h |
| 13CFh | GPIO Pad Data Output (GPDO204) | 8 | RW | 00h |
| 13D0h | GPIO Pad Data Output (GPDO211) | 8 | RW | 00h |
| 13D1h | GPIO Pad Data Output (GPDO210) | 8 | RW | 00h |
| 13D2h | GPIO Pad Data Output (GPDO209) | 8 | RW | 00h |
| 13D3h | GPIO Pad Data Output (GPDO208) | 8 | RW | 00h |
| 13D4h | GPIO Pad Data Output (GPDO215) | 8 | RW | 00h |
| 13D5h | GPIO Pad Data Output (GPDO214) | 8 | RW | 00h |
| 13D6h | GPIO Pad Data Output (GPDO213) | 8 | RW | 00h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 13D7h | GPIO Pad Data Output (GPDO212) | 8 | RW | 00h |
| 13D8h | GPIO Pad Data Output (GPDO219) | 8 | RW | 00h |
| 13D9h | GPIO Pad Data Output (GPDO218) | 8 | RW | 00h |
| 13DAh | GPIO Pad Data Output (GPDO217) | 8 | RW | 00h |
| 13DBh | GPIO Pad Data Output (GPDO216) | 8 | RW | 00h |
| 1500h | GPIO Pad Data Input (GPDI3) | 8 | R | 00h |
| 1501h | GPIO Pad Data Input (GPDI2) | 8 | R | 00h |
| 1502h | GPIO Pad Data Input (GPDI1) | 8 | R | 00h |
| 1503h | GPIO Pad Data Input (GPDI0) | 8 | R | 00h |
| 1504h | GPIO Pad Data Input (GPDI7) | 8 | R | 00h |
| 1505h | GPIO Pad Data Input (GPDI6) | 8 | R | 00h |
| 1506h | GPIO Pad Data Input (GPDI5) | 8 | R | 00h |
| 1507h | GPIO Pad Data Input (GPDI4) | 8 | R | 00h |
| 1508h | GPIO Pad Data Input (GPDI11) | 8 | R | 00h |
| 1509h | GPIO Pad Data Input (GPDI10) | 8 | R | 00h |
| 150Ah | GPIO Pad Data Input (GPDI9) | 8 | R | 00h |
| 150Bh | GPIO Pad Data Input (GPDI8) | 8 | R | 00h |
| 150Ch | GPIO Pad Data Input (GPDI15) | 8 | R | 00h |
| 150Dh | GPIO Pad Data Input (GPDI14) | 8 | R | 00h |
| 150Eh | GPIO Pad Data Input (GPDI13) | 8 | R | 00h |
| 150Fh | GPIO Pad Data Input (GPDI12) | 8 | R | 00h |
| 1510h | GPIO Pad Data Input (GPDI19) | 8 | R | 00h |
| 1511h | GPIO Pad Data Input (GPDI18) | 8 | R | 00h |
| 1512h | GPIO Pad Data Input (GPDI17) | 8 | R | 00h |
| 1513h | GPIO Pad Data Input (GPDI16) | 8 | R | 00h |
| 1514h | GPIO Pad Data Input (GPDI23) | 8 | R | 00h |
| 1515h | GPIO Pad Data Input (GPDI22) | 8 | R | 00h |
| 1516h | GPIO Pad Data Input (GPDI21) | 8 | R | 00h |
| 1517h | GPIO Pad Data Input (GPDI20) | 8 | R | 00h |
| 1518h | GPIO Pad Data Input (GPDI27) | 8 | R | 00h |
| 1519h | GPIO Pad Data Input (GPDI26) | 8 | R | 00h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 151Ah | GPIO Pad Data Input (GPDI25) | 8 | R | 00h |
| 151Bh | GPIO Pad Data Input (GPDI24) | 8 | R | 00h |
| 151Ch | GPIO Pad Data Input (GPDI31) | 8 | R | 00h |
| 151Dh | GPIO Pad Data Input (GPDI30) | 8 | R | 00h |
| 151Eh | GPIO Pad Data Input (GPDI29) | 8 | R | 00h |
| 151Fh | GPIO Pad Data Input (GPDI28) | 8 | R | 00h |
| 1520h | GPIO Pad Data Input (GPDI35) | 8 | R | 00h |
| 1521h | GPIO Pad Data Input (GPDI34) | 8 | R | 00h |
| 1522h | GPIO Pad Data Input (GPDI33) | 8 | R | 00h |
| 1523h | GPIO Pad Data Input (GPDI32) | 8 | R | 00h |
| 1526h | GPIO Pad Data Input (GPDI37) | 8 | R | 00h |
| 1527h | GPIO Pad Data Input (GPDI36) | 8 | R | 00h |
| 1528h | GPIO Pad Data Input (GPDI43) | 8 | R | 00h |
| 1529h | GPIO Pad Data Input (GPDI42) | 8 | R | 00h |
| 152Ah | GPIO Pad Data Input (GPDI41) | 8 | R | 00h |
| 152Bh | GPIO Pad Data Input (GPDI40) | 8 | R | 00h |
| 152Ch | GPIO Pad Data Input (GPDI47) | 8 | R | 00h |
| 152Dh | GPIO Pad Data Input (GPDI46) | 8 | R | 00h |
| 152Eh | GPIO Pad Data Input (GPDI45) | 8 | R | 00h |
| 152Fh | GPIO Pad Data Input (GPDI44) | 8 | R | 00h |
| 1530h | GPIO Pad Data Input (GPDI51) | 8 | R | 00h |
| 1531h | GPIO Pad Data Input (GPDI50) | 8 | R | 00h |
| 1532h | GPIO Pad Data Input (GPDI49) | 8 | R | 00h |
| 1533h | GPIO Pad Data Input (GPDI48) | 8 | R | 00h |
| 1534h | GPIO Pad Data Input (GPDI55) | 8 | R | 00h |
| 1535h | GPIO Pad Data Input (GPDI54) | 8 | R | 00h |
| 1536h | GPIO Pad Data Input (GPDI53) | 8 | R | 00h |
| 1537h | GPIO Pad Data Input (GPDI52) | 8 | R | 00h |
| 1538h | GPIO Pad Data Input (GPDI59) | 8 | R | 00h |
| 1539h | GPIO Pad Data Input (GPDI58) | 8 | R | 00h |
| 153Ah | GPIO Pad Data Input (GPDI57) | 8 | R | 00h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 153Bh | GPIO Pad Data Input (GPDI56) | 8 | R | 00h |
| 153Ch | GPIO Pad Data Input (GPDI63) | 8 | R | 00h |
| 153Dh | GPIO Pad Data Input (GPDI62) | 8 | R | 00h |
| 153Eh | GPIO Pad Data Input (GPDI61) | 8 | R | 00h |
| 153Fh | GPIO Pad Data Input (GPDI60) | 8 | R | 00h |
| 1540h | GPIO Pad Data Input (GPDI67) | 8 | R | 00h |
| 1541h | GPIO Pad Data Input (GPDI66) | 8 | R | 00h |
| 1542h | GPIO Pad Data Input (GPDI65) | 8 | R | 00h |
| 1543h | GPIO Pad Data Input (GPDI64) | 8 | R | 00h |
| 1544h | GPIO Pad Data Input (GPDI71) | 8 | R | 00h |
| 1545h | GPIO Pad Data Input (GPDI70) | 8 | R | 00h |
| 1546h | GPIO Pad Data Input (GPDI69) | 8 | R | 00h |
| 1547h | GPIO Pad Data Input (GPDI68) | 8 | R | 00h |
| 1548h | GPIO Pad Data Input (GPDI75) | 8 | R | 00h |
| 1549h | GPIO Pad Data Input (GPDI74) | 8 | R | 00h |
| 154Ah | GPIO Pad Data Input (GPDI73) | 8 | R | 00h |
| 154Bh | GPIO Pad Data Input (GPDI72) | 8 | R | 00h |
| 154Ch | GPIO Pad Data Input (GPDI79) | 8 | R | 00h |
| 154Dh | GPIO Pad Data Input (GPDI78) | 8 | R | 00h |
| 154Eh | GPIO Pad Data Input (GPDI77) | 8 | R | 00h |
| 154Fh | GPIO Pad Data Input (GPDI76) | 8 | R | 00h |
| 1550h | GPIO Pad Data Input (GPDI83) | 8 | R | 00h |
| 1551h | GPIO Pad Data Input (GPDI82) | 8 | R | 00h |
| 1552h | GPIO Pad Data Input (GPDI81) | 8 | R | 00h |
| 1553h | GPIO Pad Data Input (GPDI80) | 8 | R | 00h |
| 1554h | GPIO Pad Data Input (GPDI87) | 8 | R | 00h |
| 1555h | GPIO Pad Data Input (GPDI86) | 8 | R | 00h |
| 1556h | GPIO Pad Data Input (GPDI85) | 8 | R | 00h |
| 1557h | GPIO Pad Data Input (GPDI84) | 8 | R | 00h |
| 1558h | GPIO Pad Data Input (GPDI91) | 8 | R | 00h |
| 1559h | GPIO Pad Data Input (GPDI90) | 8 | R | 00h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 155Ah | GPIO Pad Data Input (GPDI89) | 8 | R | 00h |
| 155Bh | GPIO Pad Data Input (GPDI88) | 8 | R | 00h |
| 155Ch | GPIO Pad Data Input (GPDI95) | 8 | R | 00h |
| 155Dh | GPIO Pad Data Input (GPDI94) | 8 | R | 00h |
| 155Eh | GPIO Pad Data Input (GPDI93) | 8 | R | 00h |
| 155Fh | GPIO Pad Data Input (GPDI92) | 8 | R | 00h |
| 1560h | GPIO Pad Data Input (GPDI99) | 8 | R | 00h |
| 1561h | GPIO Pad Data Input (GPDI98) | 8 | R | 00h |
| 1562h | GPIO Pad Data Input (GPDI97) | 8 | R | 00h |
| 1563h | GPIO Pad Data Input (GPDI96) | 8 | R | 00h |
| 1564h | GPIO Pad Data Input (GPDI103) | 8 | R | 00h |
| 1565h | GPIO Pad Data Input (GPDI102) | 8 | R | 00h |
| 1566h | GPIO Pad Data Input (GPDI101) | 8 | R | 00h |
| 1567h | GPIO Pad Data Input (GPDI100) | 8 | R | 00h |
| 1568h | GPIO Pad Data Input (GPDI107) | 8 | R | 00h |
| 1569h | GPIO Pad Data Input (GPDI106) | 8 | R | 00h |
| 156Ah | GPIO Pad Data Input (GPDI105) | 8 | R | 00h |
| 156Bh | GPIO Pad Data Input (GPDI104) | 8 | R | 00h |
| 156Ch | GPIO Pad Data Input (GPDI111) | 8 | R | 00h |
| 156Dh | GPIO Pad Data Input (GPDI110) | 8 | R | 00h |
| 156Eh | GPIO Pad Data Input (GPDI109) | 8 | R | 00h |
| 156Fh | GPIO Pad Data Input (GPDI108) | 8 | R | 00h |
| 1570h | GPIO Pad Data Input (GPDI115) | 8 | R | 00h |
| 1571h | GPIO Pad Data Input (GPDI114) | 8 | R | 00h |
| 1572h | GPIO Pad Data Input (GPDI113) | 8 | R | 00h |
| 1573h | GPIO Pad Data Input (GPDI112) | 8 | R | 00h |
| 1574h | GPIO Pad Data Input (GPDI119) | 8 | R | 00h |
| 1575h | GPIO Pad Data Input (GPDI118) | 8 | R | 00h |
| 1576h | GPIO Pad Data Input (GPDI117) | 8 | R | 00h |
| 1577h | GPIO Pad Data Input (GPDI116) | 8 | R | 00h |
| 1578h | GPIO Pad Data Input (GPDI123) | 8 | R | 00h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1579h | GPIO Pad Data Input (GPDI122) | 8 | R | 00h |
| 157Ah | GPIO Pad Data Input (GPDI121) | 8 | R | 00h |
| 157Bh | GPIO Pad Data Input (GPDI120) | 8 | R | 00h |
| 157Ch | GPIO Pad Data Input (GPDI127) | 8 | R | 00h |
| 157Dh | GPIO Pad Data Input (GPDI126) | 8 | R | 00h |
| 157Eh | GPIO Pad Data Input (GPDI125) | 8 | R | 00h |
| 157Fh | GPIO Pad Data Input (GPDI124) | 8 | R | 00h |
| 1580h | GPIO Pad Data Input (GPDI131) | 8 | R | 00h |
| 1581h | GPIO Pad Data Input (GPDI130) | 8 | R | 00h |
| 1582h | GPIO Pad Data Input (GPDI129) | 8 | R | 00h |
| 1583h | GPIO Pad Data Input (GPDI128) | 8 | R | 00h |
| 1584h | GPIO Pad Data Input (GPDI135) | 8 | R | 00h |
| 1585h | GPIO Pad Data Input (GPDI134) | 8 | R | 00h |
| 1586h | GPIO Pad Data Input (GPDI133) | 8 | R | 00h |
| 1587h | GPIO Pad Data Input (GPDI132) | 8 | R | 00h |
| 1588h | GPIO Pad Data Input (GPDI139) | 8 | R | 00h |
| 1589h | GPIO Pad Data Input (GPDI138) | 8 | R | 00h |
| 158Ah | GPIO Pad Data Input (GPDI137) | 8 | R | 00h |
| 158Bh | GPIO Pad Data Input (GPDI136) | 8 | R | 00h |
| 158Ch | GPIO Pad Data Input (GPDI143) | 8 | R | 00h |
| 158Dh | GPIO Pad Data Input (GPDI142) | 8 | R | 00h |
| 158Eh | GPIO Pad Data Input (GPDI141) | 8 | R | 00h |
| 158Fh | GPIO Pad Data Input (GPDI140) | 8 | R | 00h |
| 1590h | GPIO Pad Data Input (GPDI147) | 8 | R | 00h |
| 1591h | GPIO Pad Data Input (GPDI146) | 8 | R | 00h |
| 1592h | GPIO Pad Data Input (GPDI145) | 8 | R | 00h |
| 1593h | GPIO Pad Data Input (GPDI144) | 8 | R | 00h |
| 1594h | GPIO Pad Data Input (GPDI151) | 8 | R | 00h |
| 1595h | GPIO Pad Data Input (GPDI150) | 8 | R | 00h |
| 1596h | GPIO Pad Data Input (GPDI149) | 8 | R | 00h |
| 1597h | GPIO Pad Data Input (GPDI148) | 8 | R | 00h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1598h | GPIO Pad Data Input (GPDI155) | 8 | R | 00h |
| 1599h | GPIO Pad Data Input (GPDI154) | 8 | R | 00h |
| 159Ah | GPIO Pad Data Input (GPDI153) | 8 | R | 00h |
| 159Bh | GPIO Pad Data Input (GPDI152) | 8 | R | 00h |
| 159Ch | GPIO Pad Data Input (GPDI159) | 8 | R | 00h |
| 159Dh | GPIO Pad Data Input (GPDI158) | 8 | R | 00h |
| 159Eh | GPIO Pad Data Input (GPDI157) | 8 | R | 00h |
| 159Fh | GPIO Pad Data Input (GPDI156) | 8 | R | 00h |
| 15A0h | GPIO Pad Data Input (GPDI163) | 8 | R | 00h |
| 15A1h | GPIO Pad Data Input (GPDI162) | 8 | R | 00h |
| 15A2h | GPIO Pad Data Input (GPDI161) | 8 | R | 00h |
| 15A3h | GPIO Pad Data Input (GPDI160) | 8 | R | 00h |
| 15A4h | GPIO Pad Data Input (GPDI167) | 8 | R | 00h |
| 15A5h | GPIO Pad Data Input (GPDI166) | 8 | R | 00h |
| 15A6h | GPIO Pad Data Input (GPDI165) | 8 | R | 00h |
| 15A7h | GPIO Pad Data Input (GPDI164) | 8 | R | 00h |
| 15A8h | GPIO Pad Data Input (GPDI171) | 8 | R | 00h |
| 15A9h | GPIO Pad Data Input (GPDI170) | 8 | R | 00h |
| 15AAh | GPIO Pad Data Input (GPDI169) | 8 | R | 00h |
| 15ABh | GPIO Pad Data Input (GPDI168) | 8 | R | 00h |
| 15ACh | GPIO Pad Data Input (GPDI175) | 8 | R | 00h |
| 15ADh | GPIO Pad Data Input (GPDI174) | 8 | R | 00h |
| 15AEh | GPIO Pad Data Input (GPDI173) | 8 | R | 00h |
| 15AFh | GPIO Pad Data Input (GPDI172) | 8 | R | 00h |
| 15B0h | GPIO Pad Data Input (GPDI179) | 8 | R | 00h |
| 15B1h | GPIO Pad Data Input (GPDI178) | 8 | R | 00h |
| 15B2h | GPIO Pad Data Input (GPDI177) | 8 | R | 00h |
| 15B3h | GPIO Pad Data Input (GPDI176) | 8 | R | 00h |
| 15B4h | GPIO Pad Data Input (GPDI183) | 8 | R | 00h |
| 15B5h | GPIO Pad Data Input (GPDI182) | 8 | R | 00h |
| 15B6h | GPIO Pad Data Input (GPDI181) | 8 | R | 00h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 15B7h | GPIO Pad Data Input (GPDI180) | 8 | R | 00h |
| 15B8h | GPIO Pad Data Input (GPDI187) | 8 | R | 00h |
| 15B9h | GPIO Pad Data Input (GPDI186) | 8 | R | 00h |
| 15BAh | GPIO Pad Data Input (GPDI185) | 8 | R | 00h |
| 15BBh | GPIO Pad Data Input (GPDI184) | 8 | R | 00h |
| 15BCh | GPIO Pad Data Input (GPDI191) | 8 | R | 00h |
| 15BDh | GPIO Pad Data Input (GPDI190) | 8 | R | 00h |
| 15BEh | GPIO Pad Data Input (GPDI189) | 8 | R | 00h |
| 15BFh | GPIO Pad Data Input (GPDI188) | 8 | R | 00h |
| 15C0h | GPIO Pad Data Input (GPDI195) | 8 | R | 00h |
| 15C1h | GPIO Pad Data Input (GPDI194) | 8 | R | 00h |
| 15C2h | GPIO Pad Data Input (GPDI193) | 8 | R | 00h |
| 15C3h | GPIO Pad Data Input (GPDI192) | 8 | R | 00h |
| 15C4h | GPIO Pad Data Input (GPDI199) | 8 | R | 00h |
| 15C5h | GPIO Pad Data Input (GPDI198) | 8 | R | 00h |
| 15C6h | GPIO Pad Data Input (GPDI197) | 8 | R | 00h |
| 15C7h | GPIO Pad Data Input (GPDI196) | 8 | R | 00h |
| 15C8h | GPIO Pad Data Input (GPDI203) | 8 | R | 00h |
| 15C9h | GPIO Pad Data Input (GPDI202) | 8 | R | 00h |
| 15CAh | GPIO Pad Data Input (GPDI201) | 8 | R | 00h |
| 15CBh | GPIO Pad Data Input (GPDI200) | 8 | R | 00h |
| 15CCh | GPIO Pad Data Input (GPDI207) | 8 | R | 00h |
| 15CDh | GPIO Pad Data Input (GPDI206) | 8 | R | 00h |
| 15CEh | GPIO Pad Data Input (GPDI205) | 8 | R | 00h |
| 15CFh | GPIO Pad Data Input (GPDI204) | 8 | R | 00h |
| 15D0h | GPIO Pad Data Input (GPDI211) | 8 | R | 00h |
| 15D1h | GPIO Pad Data Input (GPDI210) | 8 | R | 00h |
| 15D2h | GPIO Pad Data Input (GPDI209) | 8 | R | 00h |
| 15D3h | GPIO Pad Data Input (GPDI208) | 8 | R | 00h |
| 15D4h | GPIO Pad Data Input (GPDI215) | 8 | R | 00h |
| 15D5h | GPIO Pad Data Input (GPDI214) | 8 | R | 00h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 15D6h | GPIO Pad Data Input (GPDI213) | 8 | R | 00h |
| 15D7h | GPIO Pad Data Input (GPDI212) | 8 | R | 00h |
| 15D8h | GPIO Pad Data Input (GPDI219) | 8 | R | 00h |
| 15D9h | GPIO Pad Data Input (GPDI218) | 8 | R | 00h |
| 15DAh | GPIO Pad Data Input (GPDI217) | 8 | R | 00h |
| 15DBh | GPIO Pad Data Input (GPDI216) | 8 | R | 00h |
| 1700h - 1704h | Parallel GPIO Pad Data Output (PGPDO0 - PGPDO3) [1] | 16 | RW | 0000h |
| 1706h | Parallel GPIO Pad Data Output (PGPDO2) | 16 | RW | 0000h |
| 1708h - 1716h | Parallel GPIO Pad Data Output (PGPDO4 - PGPDO11) [1] | 16 | RW | 0000h |
| 1718h | Parallel GPIO Pad Data Output (PGPDO13) | 16 | RW | 0000h |
| 171Ah | Parallel GPIO Pad Data Output (PGPDO12) | 16 | RW | 0000h |
| 1740h - 1744h | Parallel GPIO Pad Data Input (PGPDI0 - PGPDI3) [1] | 16 | R | 0000h |
| 1746h | Parallel GPIO Pad Data Input (PGPDI2) | 16 | R | 0000h |
| 1748h - 1756h | Parallel GPIO Pad Data Input (PGPDI4 - PGPDI11) [1] | 16 | R | 0000h |
| 1758h | Parallel GPIO Pad Data Input (PGPDI13) | 16 | R | 0000h |
| 175Ah | Parallel GPIO Pad Data Input (PGPDI12) | 16 | R | 0000h |
| 1780h - 1784h | Masked Parallel GPIO Pad Data Output (MPGPDO0 - MPGPDO1) | 32 | RW | 0000_0000h |
| 1788h | Masked Parallel GPIO Pad Data Output (MPGPDO2) | 32 | RW | 0000_0000h |
| 178Ch - 17B0h | Masked Parallel GPIO Pad Data Output (MPGPDO3 - MPGPDO12) | 32 | RW | 0000_0000h |
| 17B4h | Masked Parallel GPIO Pad Data Output (MPGPDO13) | 32 | RW | 0000_0000h |

1. In this array, the index and offset values of the registers do not increment in direct alignment. For details, see the register description.

## 9.6.2 SIUL2 MCU ID Register #1 (MIDR1)

### Offset

| Register | Offset |
|---|---|
| MIDR1 | 4h |

### Function

This register holds identification information about the device.

**NOTE**

This register supports only 32-bit accesses. Byte and half-word accesses are not supported.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PRODUCT_LINE_LETTER | | | | | | | | PART_NO | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | MAJOR_MASK | | | | MINOR_MASK | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

### Fields

| Field | Function |
|---|---|
| 31-26<br>PRODUCT_LINE_LETTER | Product Line Letter<br>Identified the ASCII character in MCU Part Number. This field specifies the part number suffix, and needs to be combined with MIDR1[PART_NO] to provide the full chip number.<br>0x0B K<br>This value is set at the factory and cannot be changed. All other values are reserved. |
| 25-16<br>PART_NO | MCU Part Number<br>MCXE315/MCXE316 - 0x137<br>MCXE317 - 0x138<br>MCXE31B - 0x13A |
| 15-8<br>— | Reserved |
| 7-4<br>MAJOR_MASK | Major Mask Revision |
| 3-0<br>MINOR_MASK | Minor Mask Revision |

## 9.6.3 SIUL2 MCU ID Register #2 (MIDR2)

### Offset

| Register | Offset |
|---|---|
| MIDR2 | 8h |

### Function

**NOTE**

This register supports only 32-bit accesses. Byte and half-word accesses are not supported.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TECHNOLOGY | | | | TEMPERATURE | | | | PACKAGE | | | | FREQUENCY | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FLASH_CODE | | FLASH_DATA | | FLASH_SIZE_DATA | | | | FLASH_SIZE_CODE | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

### Fields

| Field | Function |
|---|---|
| 31-29<br>TECHNOLOGY | Technology<br>Identifies the silicon technology.<br>001b - C40EFS3 |
| 28-26<br>TEMPERATURE | Temperature<br>Identifies the ambient temperature range.<br>100b - M = 125C |
| 25-20<br>PACKAGE | Package<br>This field can by read by software to determine the package type that is used for the particular device.<br>00_0011b - 257-MAPBGA<br>10_0011b - 100-MAXQFP<br>10_0110b - 172-MAXQFP |
| 19-16 | Frequency |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| FREQUENCY | Identifies maximum core frequency. Qualified by Product Line Letter to provide wider range of frequencies.<br>0011b - 120 MHz<br>0100b - 160 MHz<br>0101b - 240 MHz |
| 15-14<br>FLASH_CODE | Flash Code<br>Identifies the location of Code Flash, if any, within the package.<br>10b - Monolithic |
| 13-12<br>FLASH_DATA | Flash Data<br>Identifies the location of Data Flash, if any, within the package.<br>10b - Monolithic |
| 11-8<br>FLASH_SIZE_DATA | Flash Size Data<br>Identifies the Flash (EE) memory size.<br>0000b - 64KB<br>0001b - 128KB<br>0010b - 256KB |
| 7-0<br>FLASH_SIZE_CODE | Flash Size Code<br>Identifies the Flash (code) memory size.<br>0000_0010b - 512kB<br>0000_0100b - 1MB<br>0000_1000b - 2.00MB<br>0000_1100b - 3.00MB<br>0001_0000b - 4.00MB<br>0001_1000b - 6.00MB<br>0010_0000b - 8.00MB |

### 9.6.4 DMA or Interrupt Status Flag 0 (DISR0)

#### Offset

| Register | Offset |
|---|---|
| DISR0 | 10h |

#### Function

Contains flags that record an event on the external IRQ pins. This register supports 8-, 16-, and 32-bit accesses.

When an event (as defined in and IFEER0) occurs, the corresponding flag is set. The IRQ flag is set regardless of the state of the corresponding DIRER0[EIRE*n*]. The IRQ flag remains set until you clear it or is cleared by servicing of a DMA request. The IRQ flags are cleared when you write 1 to them. A write of 0 has no effect.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EIF31 | EIF30 | EIF29 | EIF28 | EIF27 | EIF26 | EIF25 | EIF24 | EIF23 | EIF22 | EIF21 | EIF20 | EIF19 | EIF18 | EIF17 | EIF16 |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EIF15 | EIF14 | EIF13 | EIF12 | EIF11 | EIF10 | EIF9 | EIF8 | EIF7 | EIF6 | EIF5 | EIF4 | EIF3 | EIF2 | EIF1 | EIF0 |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>EIF31 | External Interrupt Status Flag 31<br>Indicates whether an interrupt event (as defined by IREER31 and IFEER31) has occurred.<br>If this flag is set (DIRERR31 = 1), it causes an interrupt.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 30<br>EIF30 | External Interrupt Status Flag 30<br>Indicates whether an interrupt event (as defined by IREER30 and IFEER30) has occurred.<br>If this flag is set (DIRERR30 = 1), it causes an interrupt.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No effect<br>1b - Clear the flag |
| 29<br>EIF29 | External Interrupt Status Flag 29<br>Indicates whether an interrupt event (as defined by IREER29 and IFEER29) has occurred.<br>If this flag is set (DIRERR29 = 1), it causes an interrupt.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 28<br>EIF28 | External Interrupt Status Flag 28<br>Indicates whether an interrupt event (as defined by IREER28 and IFEER28) has occurred.<br>If this flag is set (DIRERR28 = 1), it causes an interrupt.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 27<br>EIF27 | External Interrupt Status Flag 27<br>Indicates whether an interrupt event (as defined by IREER27 and IFEER27) has occurred.<br>If this flag is set (DIRERR27 = 1), it causes an interrupt.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 26<br>EIF26 | External Interrupt Status Flag 26<br>Indicates whether an interrupt event (as defined by IREER26 and IFEER26) has occurred.<br>If this flag is set (DIRERR26 = 1), it causes an interrupt.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 25<br>EIF25 | External Interrupt Status Flag 25<br>Indicates whether an interrupt event (as defined by IREER25 and IFEER25) has occurred.<br>If this flag is set (DIRERR25 = 1), it causes an interrupt.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 24<br>EIF24 | External Interrupt Status Flag 24<br>Indicates whether an interrupt event (as defined by IREER24 and IFEER24) has occurred.<br>If this flag is set (DIRERR24 = 1), it causes an interrupt.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 23<br>EIF23 | External Interrupt Status Flag 23<br>Indicates whether an interrupt event (as defined by IREER23 and IFEER23) has occurred.<br>If this flag is set (DIRERR23 = 1), it causes an interrupt.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 22<br>EIF22 | External Interrupt Status Flag 22<br>Indicates whether an interrupt event (as defined by IREER22 and IFEER22) has occurred.<br>If this flag is set (DIRERR22 = 1), it causes an interrupt.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 21<br>EIF21 | External Interrupt Status Flag 21<br>Indicates whether an interrupt event (as defined by IREER21 and IFEER21) has occurred.<br>If this flag is set (DIRERR21 = 1), it causes an interrupt.<br>**NOTE**<br>This field behaves differently for register reads and writes. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 20<br>EIF20 | External Interrupt Status Flag 20<br>Indicates whether an interrupt event (as defined by IREER20 and IFEER20) has occurred.<br>If this flag is set (DIRERR20 = 1), it causes an interrupt.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 19<br>EIF19 | External Interrupt Status Flag 19<br>Indicates whether an interrupt event (as defined by IREER19 and IFEER19) has occurred.<br>If this flag is set (DIRERR19 = 1), it causes an interrupt.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 18<br>EIF18 | External Interrupt Status Flag 18<br>Indicates whether an interrupt event (as defined by IREER18 and IFEER18) has occurred.<br>If this flag is set (DIRERR18 = 1), it causes an interrupt. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 17<br>EIF17 | External Interrupt Status Flag 17<br>Indicates whether an interrupt event (as defined by IREER17 and IFEER17) has occurred.<br>If this flag is set (DIRERR17 = 1), it causes an interrupt.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 16<br>EIF16 | External Interrupt Status Flag 16<br>Indicates whether an interrupt event (as defined by IREER16 and IFEER16) has occurred.<br>If this flag is set (DIRERR16 = 1), it causes an interrupt.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 15<br>EIF15 | External Interrupt Status Flag 15<br>Indicates whether an interrupt event (as defined by IREER15 and IFEER15) has occurred. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | If this flag is set (DIRERR15 = 1), it causes an interrupt or DMA request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 14<br>EIF14 | External Interrupt Status Flag 14<br>Indicates whether an interrupt event (as defined by IREER14 and IFEER14) has occurred.<br>If this flag is set (DIRERR14 = 1), it causes an interrupt or DMA request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 13<br>EIF13 | External Interrupt Status Flag 13<br>Indicates whether an interrupt event (as defined by IREER13 and IFEER13) has occurred.<br>If this flag is set (DIRERR13 = 1), it causes an interrupt or DMA request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 12 | External Interrupt Status Flag 12 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| EIF12 | Indicates whether an interrupt event (as defined by IREER12 and IFEER12) has occurred.<br>If this flag is set (DIRERR12 = 1), it causes an interrupt or DMA request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 11<br>EIF11 | External Interrupt Status Flag 11<br>Indicates whether an interrupt event (as defined by IREER11 and IFEER11) has occurred.<br>If this flag is set (DIRERR11 = 1), it causes an interrupt or DMA request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 10<br>EIF10 | External Interrupt Status Flag 10<br>Indicates whether an interrupt event (as defined by IREER10 and IFEER10) has occurred.<br>If this flag is set (DIRERR10 = 1), it causes an interrupt or DMA request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 9<br>EIF9 | External Interrupt Status Flag 9<br>Indicates whether an interrupt event (as defined by IREER9 and IFEER9) has occurred.<br>If this flag is set (DIRERR9 = 1), it causes an interrupt or DMA request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 8<br>EIF8 | External Interrupt Status Flag 8<br>Indicates whether an interrupt event (as defined by IREER8 and IFEER8) has occurred.<br>If this flag is set (DIRERR8 = 1), it causes an interrupt or DMA request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 7<br>EIF7 | External Interrupt Status Flag 7<br>Indicates whether an interrupt event (as defined by IREER7 and IFEER7) has occurred.<br>If this flag is set (DIRERR7 = 1), it causes an interrupt or DMA request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Clear the flag |
| 6<br>EIF6 | External Interrupt Status Flag 6<br>Indicates whether an interrupt event (as defined by IREER6 and IFEER6) has occurred.<br>If this flag is set (DIRERR6 = 1), it causes an interrupt or DMA request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 5<br>EIF5 | External Interrupt Status Flag 5<br>Indicates whether an interrupt event (as defined by IREER5 and IFEER5) has occurred.<br>If this flag is set (DIRERR5 = 1), it causes an interrupt or DMA request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 4<br>EIF4 | External Interrupt Status Flag 4<br>Indicates whether an interrupt event (as defined by IREER4 and IFEER4) has occurred.<br>If this flag is set (DIRERR4 = 1), it causes an interrupt or DMA request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No effect<br>1b - Clear the flag |
| 3<br>EIF3 | External Interrupt Status Flag 3<br>Indicates whether an interrupt event (as defined by IREER3 and IFEER3) has occurred.<br>If this flag is set (DIRERR3 = 1), it causes an interrupt or DMA request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 2<br>EIF2 | External Interrupt Status Flag 2<br>Indicates whether an interrupt event (as defined by IREER2 and IFEER2) has occurred.<br>If this flag is set (DIRERR2 = 1), it causes an interrupt or DMA request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 1<br>EIF1 | External Interrupt Status Flag 1<br>Indicates whether an interrupt event (as defined by IREER1 and IFEER1) has occurred.<br>If this flag is set (DIRERR1 = 1), it causes an interrupt or DMA request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 0<br>EIF0 | External Interrupt Status Flag 0<br>Indicates whether an interrupt event (as defined by IREER0 and IFEER0) has occurred.<br>If this flag is set (DIRERR0 = 1), it causes an interrupt or DMA request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Interrupt event did not occur on the pad<br>1b - Interrupt event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |

### 9.6.5 DMA or Interrupt Request Enable 0 (DIRER0)

#### Offset

| Register | Offset |
|---|---|
| DIRER0 | 18h |

#### Function

Enables the assertion of a DMA or interrupt request to the interrupt controller if the corresponding DISR0[EIF*n*] flag is 1. The type of request is determined by the corresponding DIRSR0[DIRSR*n*] field.

This register supports 8-, 16-, and 32-bit accesses.

**NOTE**

You cannot enable or disable DIRSR0 after it selects a DMA request.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EIRE31 | EIRE30 | EIRE29 | EIRE28 | EIRE27 | EIRE26 | EIRE25 | EIRE24 | EIRE23 | EIRE22 | EIRE21 | EIRE20 | EIRE19 | EIRE18 | EIRE17 | EIRE16 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EIRE15 | EIRE14 | EIRE13 | EIRE12 | EIRE11 | EIRE10 | EIRE9 | EIRE8 | EIRE7 | EIRE6 | EIRE5 | EIRE4 | EIRE3 | EIRE2 | EIRE1 | EIRE0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>EIRE31 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 30<br>EIRE30 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 29<br>EIRE29 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 28<br>EIRE28 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 27<br>EIRE27 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 26<br>EIRE26 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disable<br>1b - Enable |
| 25<br>EIRE25 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 24<br>EIRE24 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 23<br>EIRE23 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 22<br>EIRE22 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 21<br>EIRE21 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 20<br>EIRE20 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 19<br>EIRE19 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 18 | External Interrupt Request Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| EIRE18 | Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 17<br>EIRE17 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 16<br>EIRE16 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 15<br>EIRE15 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 14<br>EIRE14 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 13<br>EIRE13 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 12<br>EIRE12 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 11<br>EIRE11 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 10<br>EIRE10 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 9<br>EIRE9 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 8<br>EIRE8 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 7<br>EIRE7 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 6<br>EIRE6 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 5<br>EIRE5 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 4<br>EIRE4 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 3<br>EIRE3 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Enable |
| 2<br>EIRE2 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 1<br>EIRE1 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |
| 0<br>EIRE0 | External Interrupt Request Enable<br>Enables the interrupt requests from the corresponding pin.<br>0b - Disable<br>1b - Enable |

### 9.6.6 DMA or Interrupt Request Select 0 (DIRSR0)

#### Offset

| Register | Offset |
|---|---|
| DIRSR0 | 20h |

#### Function

Selects the type of request (DMA or interrupt request). This register supports 8-, 16-, and 32-bit accesses.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DIRSR 31 | DIRSR 30 | DIRSR 29 | DIRSR 28 | DIRSR 27 | DIRSR 26 | DIRSR 25 | DIRSR 24 | DIRSR 23 | DIRSR 22 | DIRSR 21 | DIRSR 20 | DIRSR 19 | DIRSR 18 | DIRSR 17 | DIRSR 16 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DIRSR 15 | DIRSR 14 | DIRSR 13 | DIRSR 12 | DIRSR 11 | DIRSR 10 | DIRSR 9 | DIRSR 8 | DIRSR 7 | DIRSR 6 | DIRSR 5 | DIRSR 4 | DIRSR 3 | DIRSR 2 | DIRSR 1 | DIRSR 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>DIRSR31 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - Reserved |
| 30<br>DIRSR30 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - Reserved |
| 29<br>DIRSR29 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - Reserved |
| 28<br>DIRSR28 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - Reserved |
| 27<br>DIRSR27 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 26<br>DIRSR26 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - Reserved |
| 25<br>DIRSR25 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - Reserved |
| 24<br>DIRSR24 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - Reserved |
| 23<br>DIRSR23 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - Reserved |
| 22<br>DIRSR22 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 21<br>DIRSR21 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - Reserved |
| 20<br>DIRSR20 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - Reserved |
| 19<br>DIRSR19 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - Reserved |
| 18<br>DIRSR18 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - Reserved |
| 17<br>DIRSR17 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 16<br>DIRSR16 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - Reserved |
| 15<br>DIRSR15 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - DMA request |
| 14<br>DIRSR14 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - DMA request |
| 13<br>DIRSR13 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - DMA request |
| 12<br>DIRSR12 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - DMA request |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 11<br>DIRSR11 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - DMA request |
| 10<br>DIRSR10 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - DMA request |
| 9<br>DIRSR9 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - DMA request |
| 8<br>DIRSR8 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - DMA request |
| 7<br>DIRSR7 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - DMA request |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 6<br>DIRSR6 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - DMA request |
| 5<br>DIRSR5 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - DMA request |
| 4<br>DIRSR4 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - DMA request |
| 3<br>DIRSR3 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - DMA request |
| 2<br>DIRSR2 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - DMA request |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 1<br>DIRSR1 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - DMA request |
| 0<br>DIRSR0 | DMA or Interrupt Request Select<br>Selects either a DMA request or an external interrupt request when an edge-triggered event occurs on the corresponding pin.<br>This field determines whether a DMA request or an interrupt request asserts the corresponding DISR0[EIF*n*].<br>0b - Interrupt request<br>1b - DMA request |

## 9.6.7 Interrupt Rising-Edge Event Enable 0 (IREER0)

### Offset

| Register | Offset |
|---|---|
| IREER0 | 28h |

### Function

Enables the rising-edge triggered events on the corresponding interrupt pads. This register supports 8-, 16-, and 32-bit accesses.

**NOTE**

If you write 0 to both the IREE and IFEE fields for the same interrupt source, the interrupt status flag for the corresponding external interrupt never sets.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | IREE31 | IREE30 | IREE29 | IREE28 | IREE27 | IREE26 | IREE25 | IREE24 | IREE23 | IREE22 | IREE21 | IREE20 | IREE19 | IREE18 | IREE17 | IREE16 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | IREE15 | IREE14 | IREE13 | IREE12 | IREE11 | IREE10 | IREE9 | IREE8 | IREE7 | IREE6 | IREE5 | IREE4 | IREE3 | IREE2 | IREE1 | IREE0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>IREE31 | Interrupt Rising-Edge Event Enable 31<br>Enables the rising-edge events to set DISR0[EIF31].<br>0b - Disable<br>1b - Enable |
| 30<br>IREE30 | Interrupt Rising-Edge Event Enable 30<br>Enables the rising-edge events to set DISR0[EIF30].<br>0b - Disable<br>1b - Enable |
| 29<br>IREE29 | Interrupt Rising-Edge Event Enable 29<br>Enables the rising-edge events to set DISR0[EIF29].<br>0b - Disable<br>1b - Enable |
| 28<br>IREE28 | Interrupt Rising-Edge Event Enable 28<br>Enables the rising-edge events to set DISR0[EIF28].<br>0b - Disable<br>1b - Enable |
| 27<br>IREE27 | Interrupt Rising-Edge Event Enable 27<br>Enables the rising-edge events to set DISR0[EIF27].<br>0b - Disable<br>1b - Enable |
| 26<br>IREE26 | Interrupt Rising-Edge Event Enable 26<br>Enables the rising-edge events to set DISR0[EIF26].<br>0b - Disable<br>1b - Enable |
| 25<br>IREE25 | Interrupt Rising-Edge Event Enable 25<br>Enables the rising-edge events to set DISR0[EIF25].<br>0b - Disable<br>1b - Enable |
| 24<br>IREE24 | Interrupt Rising-Edge Event Enable 24<br>Enables the rising-edge events to set DISR0[EIF24].<br>0b - Disable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Enable |
| 23<br>IREE23 | Interrupt Rising-Edge Event Enable 23<br>Enables the rising-edge events to set DISR0[EIF23].<br>0b - Disable<br>1b - Enable |
| 22<br>IREE22 | Interrupt Rising-Edge Event Enable 22<br>Enables the rising-edge events to set DISR0[EIF22].<br>0b - Disable<br>1b - Enable |
| 21<br>IREE21 | Interrupt Rising-Edge Event Enable 21<br>Enables the rising-edge events to set DISR0[EIF21].<br>0b - Disable<br>1b - Enable |
| 20<br>IREE20 | Interrupt Rising-Edge Event Enable 20<br>Enables the rising-edge events to set DISR0[EIF20].<br>0b - Disable<br>1b - Enable |
| 19<br>IREE19 | Interrupt Rising-Edge Event Enable 19<br>Enables the rising-edge events to set DISR0[EIF19].<br>0b - Disable<br>1b - Enable |
| 18<br>IREE18 | Interrupt Rising-Edge Event Enable 18<br>Enables the rising-edge events to set DISR0[EIF18].<br>0b - Disable<br>1b - Enable |
| 17<br>IREE17 | Interrupt Rising-Edge Event Enable 17<br>Enables the rising-edge events to set DISR0[EIF17].<br>0b - Disable<br>1b - Enable |
| 16<br>IREE16 | Interrupt Rising-Edge Event Enable 16<br>Enables the rising-edge events to set DISR0[EIF16]. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disable<br>1b - Enable |
| 15<br>IREE15 | Interrupt Rising-Edge Event Enable 15<br>Enables the rising-edge events to set DISR0[EIF15].<br>0b - Disable<br>1b - Enable |
| 14<br>IREE14 | Interrupt Rising-Edge Event Enable 14<br>Enables the rising-edge events to set DISR0[EIF14].<br>0b - Disable<br>1b - Enable |
| 13<br>IREE13 | Interrupt Rising-Edge Event Enable 13<br>Enables the rising-edge events to set DISR0[EIF13].<br>0b - Disable<br>1b - Enable |
| 12<br>IREE12 | Interrupt Rising-Edge Event Enable 12<br>Enables the rising-edge events to set DISR0[EIF12].<br>0b - Disable<br>1b - Enable |
| 11<br>IREE11 | Interrupt Rising-Edge Event Enable 11<br>Enables the rising-edge events to set DISR0[EIF11].<br>0b - Disable<br>1b - Enable |
| 10<br>IREE10 | Interrupt Rising-Edge Event Enable 10<br>Enables the rising-edge events to set DISR0[EIF10].<br>0b - Disable<br>1b - Enable |
| 9<br>IREE9 | Interrupt Rising-Edge Event Enable 9<br>Enables the rising-edge events to set DISR0[EIF9].<br>0b - Disable<br>1b - Enable |
| 8 | Interrupt Rising-Edge Event Enable 8 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| IREE8 | Enables the rising-edge events to set DISR0[EIF8].<br>0b - Disable<br>1b - Enable |
| 7<br>IREE7 | Interrupt Rising-Edge Event Enable 7<br>Enables the rising-edge events to set DISR0[EIF7].<br>0b - Disable<br>1b - Enable |
| 6<br>IREE6 | Interrupt Rising-Edge Event Enable 6<br>Enables the rising-edge events to set DISR0[EIF6].<br>0b - Disable<br>1b - Enable |
| 5<br>IREE5 | Interrupt Rising-Edge Event Enable 5<br>Enables the rising-edge events to set DISR0[EIF5].<br>0b - Disable<br>1b - Enable |
| 4<br>IREE4 | Interrupt Rising-Edge Event Enable 4<br>Enables the rising-edge events to set DISR0[EIF4].<br>0b - Disable<br>1b - Enable |
| 3<br>IREE3 | Interrupt Rising-Edge Event Enable 3<br>Enables the rising-edge events to set DISR0[EIF3].<br>0b - Disable<br>1b - Enable |
| 2<br>IREE2 | Interrupt Rising-Edge Event Enable 2<br>Enables the rising-edge events to set DISR0[EIF2].<br>0b - Disable<br>1b - Enable |
| 1<br>IREE1 | Interrupt Rising-Edge Event Enable 1<br>Enables the rising-edge events to set DISR0[EIF1].<br>0b - Disable<br>1b - Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 0<br>IREE0 | Interrupt Rising-Edge Event Enable 0<br>Enables the rising-edge events to set DISR0[EIF0].<br>0b - Disable<br>1b - Enable |

## 9.6.8 Interrupt Falling-Edge Event Enable 0 (IFEER0)

### Offset

| Register | Offset |
|---|---|
| IFEER0 | 30h |

### Function

Enables the falling-edge triggered events on the corresponding interrupt pads. This register supports 8-, 16-, and 32-bit accesses.

**NOTE**

If you write 0 to both the IREE and IFEE fields for the same interrupt source, the interrupt status flag for the corresponding external interrupt never sets.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | IFEE31 | IFEE30 | IFEE29 | IFEE28 | IFEE27 | IFEE26 | IFEE25 | IFEE24 | IFEE23 | IFEE22 | IFEE21 | IFEE20 | IFEE19 | IFEE18 | IFEE17 | IFEE16 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | IFEE15 | IFEE14 | IFEE13 | IFEE12 | IFEE11 | IFEE10 | IFEE9 | IFEE8 | IFEE7 | IFEE6 | IFEE5 | IFEE4 | IFEE3 | IFEE2 | IFEE1 | IFEE0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>IFEE31 | Interrupt Falling-Edge Event Enable 31<br>Enables the falling-edge events to set DISR0[EIF31].<br>0b - Disable<br>1b - Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 30<br>IFEE30 | Interrupt Falling-Edge Event Enable 30<br>Enables the falling-edge events to set DISR0[EIF30].<br>0b - Disable<br>1b - Enable |
| 29<br>IFEE29 | Interrupt Falling-Edge Event Enable 29<br>Enables the falling-edge events to set DISR0[EIF29].<br>0b - Disable<br>1b - Enable |
| 28<br>IFEE28 | Interrupt Falling-Edge Event Enable 28<br>Enables the falling-edge events to set DISR0[EIF28].<br>0b - Disable<br>1b - Enable |
| 27<br>IFEE27 | Interrupt Falling-Edge Event Enable 27<br>Enables the falling-edge events to set DISR0[EIF27].<br>0b - Disable<br>1b - Enable |
| 26<br>IFEE26 | Interrupt Falling-Edge Event Enable 26<br>Enables the falling-edge events to set DISR0[EIF26].<br>0b - Disable<br>1b - Enable |
| 25<br>IFEE25 | Interrupt Falling-Edge Event Enable 25<br>Enables the falling-edge events to set DISR0[EIF25].<br>0b - Disable<br>1b - Enable |
| 24<br>IFEE24 | Interrupt Falling-Edge Event Enable 24<br>Enables the falling-edge events to set DISR0[EIF24].<br>0b - Disable<br>1b - Enable |
| 23<br>IFEE23 | Interrupt Falling-Edge Event Enable 23<br>Enables the falling-edge events to set DISR0[EIF23].<br>0b - Disable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Enable |
| 22<br>IFEE22 | Interrupt Falling-Edge Event Enable 22<br>Enables the falling-edge events to set DISR0[EIF22].<br>0b - Disable<br>1b - Enable |
| 21<br>IFEE21 | Interrupt Falling-Edge Event Enable 21<br>Enables the falling-edge events to set DISR0[EIF21].<br>0b - Disable<br>1b - Enable |
| 20<br>IFEE20 | Interrupt Falling-Edge Event Enable 20<br>Enables the falling-edge events to set DISR0[EIF20].<br>0b - Disable<br>1b - Enable |
| 19<br>IFEE19 | Interrupt Falling-Edge Event Enable 19<br>Enables the falling-edge events to set DISR0[EIF19].<br>0b - Disable<br>1b - Enable |
| 18<br>IFEE18 | Interrupt Falling-Edge Event Enable 18<br>Enables the falling-edge events to set DISR0[EIF18].<br>0b - Disable<br>1b - Enable |
| 17<br>IFEE17 | Interrupt Falling-Edge Event Enable 17<br>Enables the falling-edge events to set DISR0[EIF17].<br>0b - Disable<br>1b - Enable |
| 16<br>IFEE16 | Interrupt Falling-Edge Event Enable 16<br>Enables the falling-edge events to set DISR0[EIF16].<br>0b - Disable<br>1b - Enable |
| 15<br>IFEE15 | Interrupt Falling-Edge Event Enable 15<br>Enables the falling-edge events to set DISR0[EIF15]. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disable<br>1b - Enable |
| 14<br>IFEE14 | Interrupt Falling-Edge Event Enable 14<br>Enables the falling-edge events to set DISR0[EIF14].<br>0b - Disable<br>1b - Enable |
| 13<br>IFEE13 | Interrupt Falling-Edge Event Enable 13<br>Enables the falling-edge events to set DISR0[EIF13].<br>0b - Disable<br>1b - Enable |
| 12<br>IFEE12 | Interrupt Falling-Edge Event Enable 12<br>Enables the falling-edge events to set DISR0[EIF12].<br>0b - Disable<br>1b - Enable |
| 11<br>IFEE11 | Interrupt Falling-Edge Event Enable 11<br>Enables the falling-edge events to set DISR0[EIF11].<br>0b - Disable<br>1b - Enable |
| 10<br>IFEE10 | Interrupt Falling-Edge Event Enable 10<br>Enables the falling-edge events to set DISR0[EIF10].<br>0b - Disable<br>1b - Enable |
| 9<br>IFEE9 | Interrupt Falling-Edge Event Enable 9<br>Enables the falling-edge events to set DISR0[EIF9].<br>0b - Disable<br>1b - Enable |
| 8<br>IFEE8 | Interrupt Falling-Edge Event Enable 8<br>Enables the falling-edge events to set DISR0[EIF8].<br>0b - Disable<br>1b - Enable |
| 7 | Interrupt Falling-Edge Event Enable 7 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| IFEE7 | Enables the falling-edge events to set DISR0[EIF7].<br>0b - Disable<br>1b - Enable |
| 6<br>IFEE6 | Interrupt Falling-Edge Event Enable 6<br>Enables the falling-edge events to set DISR0[EIF6].<br>0b - Disable<br>1b - Enable |
| 5<br>IFEE5 | Interrupt Falling-Edge Event Enable 5<br>Enables the falling-edge events to set DISR0[EIF5].<br>0b - Disable<br>1b - Enable |
| 4<br>IFEE4 | Interrupt Falling-Edge Event Enable 4<br>Enables the falling-edge events to set DISR0[EIF4].<br>0b - Disable<br>1b - Enable |
| 3<br>IFEE3 | Interrupt Falling-Edge Event Enable 3<br>Enables the falling-edge events to set DISR0[EIF3].<br>0b - Disable<br>1b - Enable |
| 2<br>IFEE2 | Interrupt Falling-Edge Event Enable 2<br>Enables the falling-edge events to set DISR0[EIF2].<br>0b - Disable<br>1b - Enable |
| 1<br>IFEE1 | Interrupt Falling-Edge Event Enable 1<br>Enables the falling-edge events to set DISR0[EIF1].<br>0b - Disable<br>1b - Enable |
| 0<br>IFEE0 | Interrupt Falling-Edge Event Enable 0<br>Enables the falling-edge events to set DISR0[EIF0].<br>0b - Disable<br>1b - Enable |

## 9.6.9 Interrupt Filter Enable 0 (IFER0)

### Offset

| Register | Offset |
|---|---|
| IFER0 | 38h |

### Function

Enables a digital filter counter on the corresponding interrupt pads to filter out glitches on the inputs. This register supports 8-, 16-, and 32-bit accesses.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | IFE31 | IFE30 | IFE29 | IFE28 | IFE27 | IFE26 | IFE25 | IFE24 | IFE23 | IFE22 | IFE21 | IFE20 | IFE19 | IFE18 | IFE17 | IFE16 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | IFE15 | IFE14 | IFE13 | IFE12 | IFE11 | IFE10 | IFE9 | IFE8 | IFE7 | IFE6 | IFE5 | IFE4 | IFE3 | IFE2 | IFE1 | IFE0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>IFE31 | Interrupt Filter Enable 31<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 30<br>IFE30 | Interrupt Filter Enable 30<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 29<br>IFE29 | Interrupt Filter Enable 29<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 28 | Interrupt Filter Enable 28 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| IFE28 | Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 27<br>IFE27 | Interrupt Filter Enable 27<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 26<br>IFE26 | Interrupt Filter Enable 26<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 25<br>IFE25 | Interrupt Filter Enable 25<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 24<br>IFE24 | Interrupt Filter Enable 24<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 23<br>IFE23 | Interrupt Filter Enable 23<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 22<br>IFE22 | Interrupt Filter Enable 22<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 21<br>IFE21 | Interrupt Filter Enable 21<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 20<br>IFE20 | Interrupt Filter Enable 20<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 19<br>IFE19 | Interrupt Filter Enable 19<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 18<br>IFE18 | Interrupt Filter Enable 18<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 17<br>IFE17 | Interrupt Filter Enable 17<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 16<br>IFE16 | Interrupt Filter Enable 16<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 15<br>IFE15 | Interrupt Filter Enable 15<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 14<br>IFE14 | Interrupt Filter Enable 14<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 13<br>IFE13 | Interrupt Filter Enable 13<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Enable |
| 12<br>IFE12 | Interrupt Filter Enable 12<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 11<br>IFE11 | Interrupt Filter Enable 11<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 10<br>IFE10 | Interrupt Filter Enable 10<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 9<br>IFE9 | Interrupt Filter Enable 9<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 8<br>IFE8 | Interrupt Filter Enable 8<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 7<br>IFE7 | Interrupt Filter Enable 7<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 6<br>IFE6 | Interrupt Filter Enable 6<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 5<br>IFE5 | Interrupt Filter Enable 5<br>Enables the digital glitch filter on the interrupt pad input. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disable<br>1b - Enable |
| 4<br>IFE4 | Interrupt Filter Enable 4<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 3<br>IFE3 | Interrupt Filter Enable 3<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 2<br>IFE2 | Interrupt Filter Enable 2<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 1<br>IFE1 | Interrupt Filter Enable 1<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |
| 0<br>IFE0 | Interrupt Filter Enable 0<br>Enables the digital glitch filter on the interrupt pad input.<br>0b - Disable<br>1b - Enable |

### 9.6.10 Interrupt Filter Maximum Counter (IFMCR0 - IFMCR31)

#### Offset

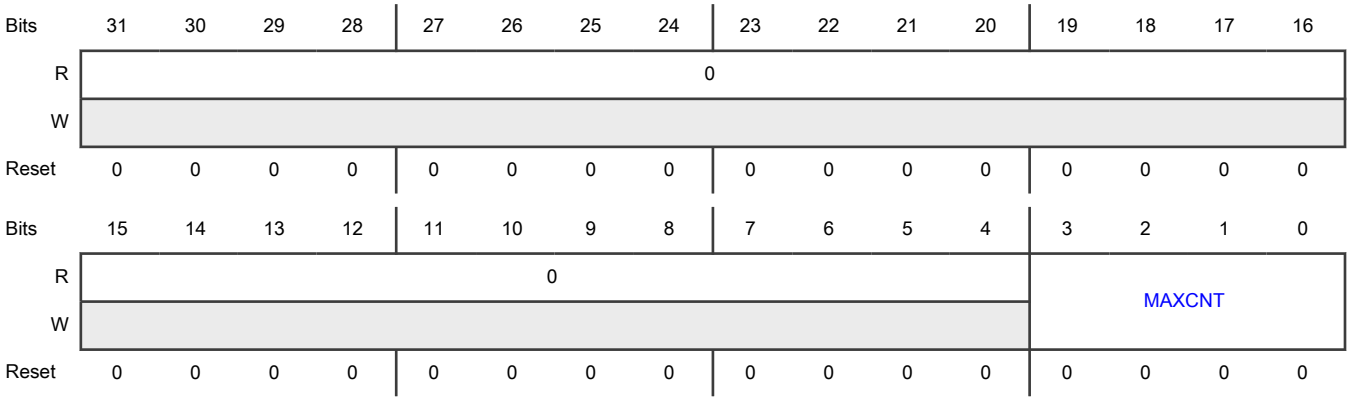
For a = 0 to 31:

| Register | Offset |
|---|---|
| IFMCRa | 40h + (a × 4h) |

#### Function

Configures the filter counter associated with each of the digital glitch filter and supports only 32-bit accesses. It does not support byte and halfword accesses.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>MAXCNT | Maximum Interrupt Filter Counter Setting<br>Specifies the settings of the maximum interrupt filter counter.<br>The value of this field varies between 0d – 15d. Based on its value, it exhibits the following settings:<br>• A value of 0d, 1d, or 2d sets the filter as an all pass filter<br>• A value of 3d to 15d sets the filter period to TCK × MAXCNT + *n* × TCK, where:<br>— *n* is 1, 2, 3, or 4 (*n* accounts for the uncertainty factor in filter period calculation)<br>— TCK is the prescaled filter clock period, which is the IRC clock prescaled to the IFCP value specified in IFCPR |

## 9.6.11 SIUL2 MCU ID Register #3 (MIDR3)

### Offset

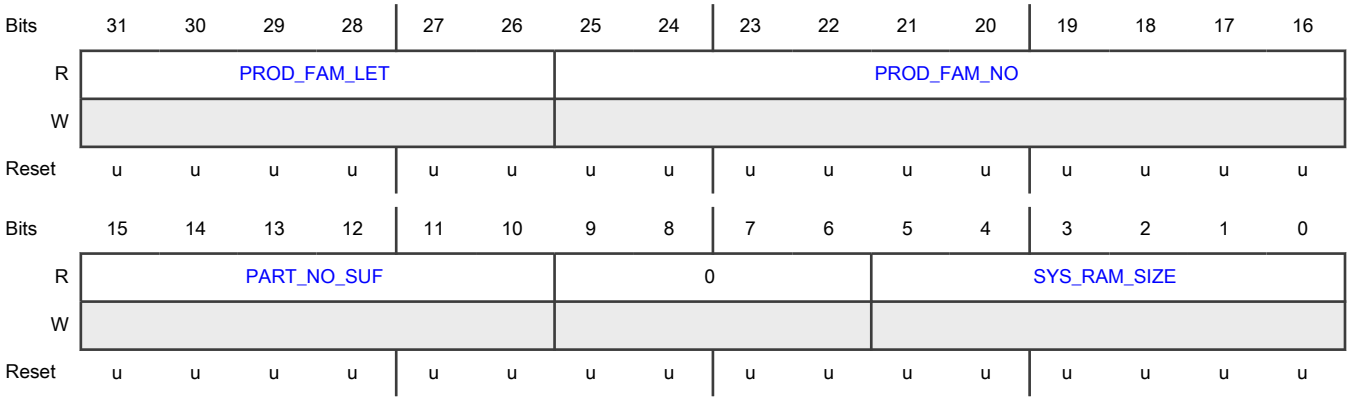
| Register | Offset |
|---|---|
| MIDR3 | 200h |

### Function

**NOTE**

This register supports only 32-bit accesses. Byte and half-word accesses are not supported.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-26<br>PROD_FAM_LET | Product Family Letter<br>Identifies the product family letter.<br>01_0011b - S |
| 25-16<br>PROD_FAM_NO | Product Family Number<br>Identifies the product family number.<br>00_0010_0000b - 32 |
| 15-10<br>PART_NO_SUF | Part Number Suffix<br>Describes the part number suffix.<br>00_0000b - None |
| 9-6<br>— | Reserved |
| 5-0<br>SYS_RAM_SIZE | System RAM Size<br>Total RAM size in SoC, including TCMs.<br>00_0010b - 128kB<br>00_0011b - 192kB<br>00_0100b - 256kB<br>00_0110b - 512kB |

## 9.6.12 SIUL2 MCU ID Register #4 (MIDR4)

### Offset

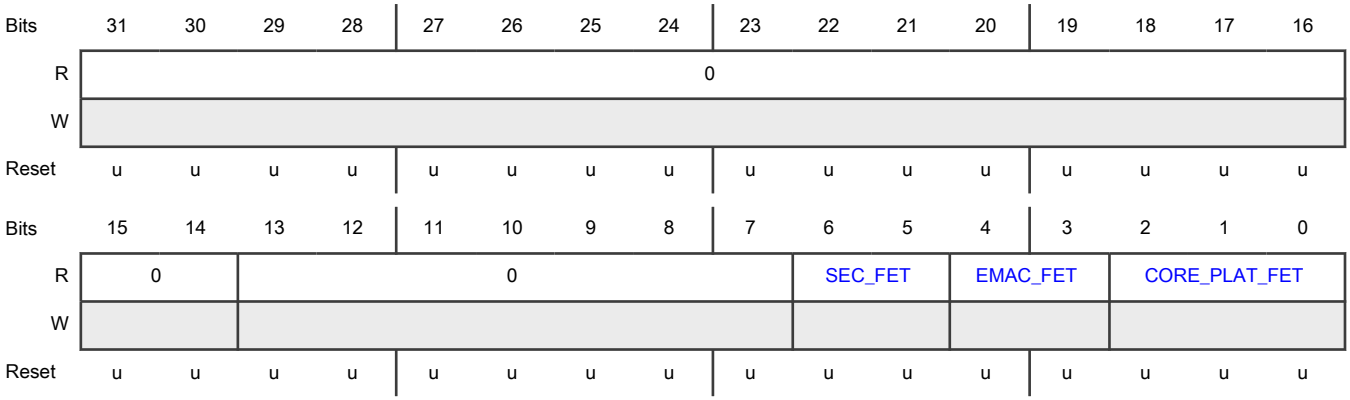
| Register | Offset |
|---|---|
| MIDR4 | 204h |

### Function

**NOTE**

This register supports only 32-bit accesses. Byte and half-word accesses are not supported.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-14<br>— | Reserved |
| 13-7<br>— | Reserved |
| 6-5<br>SEC_FET | Security Feature<br>Security feature.<br>01b - ELE_HSEB |
| 4-3<br>EMAC_FET | Ethernet Feature<br>Ethernet feature. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 00b - No Ethernet<br>01b - Ethernet present |
| 2-0<br>CORE_PLAT_FET | Core Platform Options Feature<br>Core platform options feature.<br>000b - Single core<br>001b - Dual core<br>010b - 1x lock-step core<br>011b - 1x lock-step core + 1 perf. core<br>100b - Triple core |

## 9.6.13 Multiplexed Signal Configuration (MSCR0 - MSCR323)

### Offset

For a = 0 to 323:

| Register | Offset |
|---|---|
| MSCRa | 240h + (a × 4h) |

### Function

Selects the source signal connected to the register's associated destination, which is a chip output pin or a chip pin that can be configured as an output. It also specifies the electrical properties of the associated pin.

This register supports only 32-bit accesses. Byte and half-word write accesses are not supported.

For chip-pin MSCR assignments and pin types, see the IOMUX file attached to this document.

**NOTE**

- Configure these registers only during application initialization; you must not modify them during application runtime.
- Accessing a reserved MSCR*n* register generates a transfer error.
- These registers are a part of the SIUL memory map but the physical implementation of these registers is a part of the IOMUX RTL.
- SIUL2 interprets accesses to MSCR*n* at the module level.

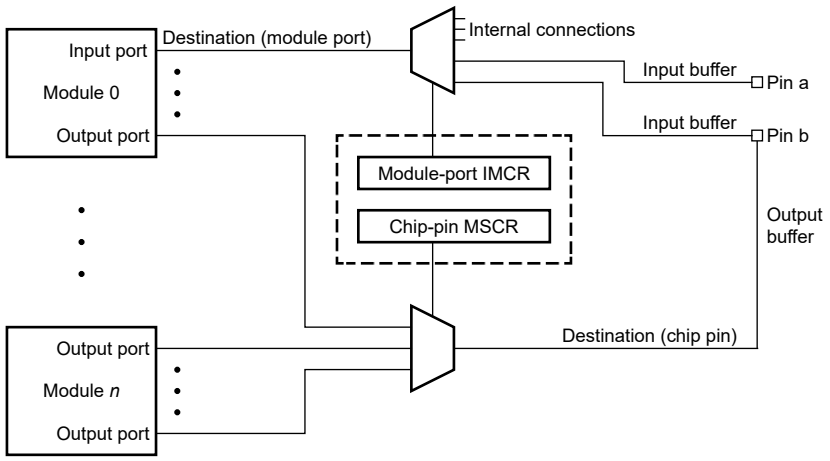



**Figure 19. MSCR and IMCR port and pin connection**

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | OBE | 0 | IBE | 0 | INV | PKE |
| W | | | | | | | | | | | | | | | | |
| Reset | See Register reset values. | | | | | | | | | | | | | | | |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | SRC | PUE | 0 | PUS | 0 | | DSE | 0 | IFE | SMC | 0 | SSS | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | See Register reset values. | | | | | | | | | | | | | | | |

### Register reset values

| Register | Reset value |
|---|---|
| MSCR0–MSCR3 | 0000_0000h |
| MSCR4 | 0008_2827h |
| MSCR5–MSCR9 | 0000_0000h |
| MSCR10 | 0000_0127h |
| MSCR11 | 0000_0000h |
| MSCR12 | 0000_0003h |
| MSCR13–MSCR37 | 0000_0000h |
| MSCR38–MSCR39 | 0001_0000h |
| MSCR40–MSCR65 | 0000_0000h |
| MSCR66–MSCR67 | 0000_4000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Reset value |
|---|---|
| MSCR68 | 0008_2000h |
| MSCR69 | 0008_2800h |
| MSCR70–MSCR75 | 0000_0000h |
| MSCR76 | 0000_4000h |
| MSCR77–MSCR79 | 0000_0000h |
| MSCR80 | 0000_4000h |
| MSCR81–MSCR100 | 0000_0000h |
| MSCR101–MSCR103 | 0000_4000h |
| MSCR104–MSCR105 | 0000_0000h |
| MSCR106–MSCR108 | 0000_4000h |
| MSCR109–MSCR135 | 0000_0000h |
| MSCR136 | 0000_4000h |
| MSCR137–MSCR219 | 0000_0000h |
| MSCR220–MSCR323 | 0001_0000h |

**Fields**

| Field | Function |
|---|---|
| 31-22<br>— | Reserved |
| 21<br>OBE | GPIO Output Buffer Enable<br>Applies only to digital pins. Otherwise this bit is reserved.<br>0b - Output driver disabled<br>1b - Output driver enabled |
| 20<br>— | Reserved |
| 19<br>IBE | Input Buffer Enable<br>Used only when the associated destination is a chip pin. Enables the associated pin's input buffer.<br>0b - Disabled<br>1b - Enabled |
| 18<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 17<br>INV | Invert<br>Inverts the signal selected by SSS before transmitting it to the associated destination (chip pin or module port).<br>0b - Don't invert<br>1b - Invert |
| 16<br>PKE | Pad keeping enable<br>Pad keeping enable<br>0b - Disabled<br>1b - Enabled |
| 15<br>— | Reserved |
| 14<br>SRC | Slew Rate Control<br>0b - Fastest setting<br>1b - Slowest setting |
| 13<br>PUE | Pull Enable<br>Enables the pull function. Used only when the associated destination is a chip pin.<br>0b - Disabled<br>1b - Enabled |
| 12<br>— | Reserved |
| 11<br>PUS | Pull Select<br>Determines whether the pull function is a pullup or pulldown when the pull function is enabled by the PUE field. Used only when the associated destination is a chip pin.<br>0b - Pull down<br>1b - Pull up |
| 10-9<br>— | Reserved |
| 8<br>DSE | DSE<br>Drive strength enable<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 7<br>— | Reserved |
| 6<br>IFE | IFE<br>Input filter enable<br>**NOTE**<br>This field is supported for RESET pad only (PTA5).<br>0b - Disabled<br>1b - Enabled |
| 5<br>SMC | Safe Mode Control<br>Used only when the associated destination is a chip pin. Specifies whether the chip disables the pin's output buffer when the chip enters Safe mode.<br>0b - Disable (The output buffer returns to its previous state when the chip leaves Safe mode.)<br>1b - Don't disable |
| 4<br>— | Reserved |
| 3-0<br>SSS | Source Signal Select<br>Selects a function for the pad. Refer to “SSS” column of the ‘IO Signal Table' tab of the IOMUX spreadsheet attachment. |

## 9.6.14 Input Multiplexed Signal Configuration (IMCR0 - IMCR473)

### Offset

For a = 0 to 473:

| Register | Offset |
|---|---|
| IMCRa | A40h + (a × 4h) |

### Function

Selects the source signal connected to the register's associated destination, which is an internal module port that is an input port or can be configured as an input.

This register supports only 32-bit accesses. Byte and half-word write accesses are not supported.

For IMCR assignments and field values, see the IOMUX file attached to this document.

**NOTE**

- Configure these registers only during application initialization; you must not modify them during application runtime.
- Accessing a reserved IMCR*n* register generates a transfer error.
- These registers are a part of the SIUL memory map but the physical implementation of these registers is a part of the IOMUX RTL.
- SIUL2 interprets accesses to MSCR*n* at the module level.

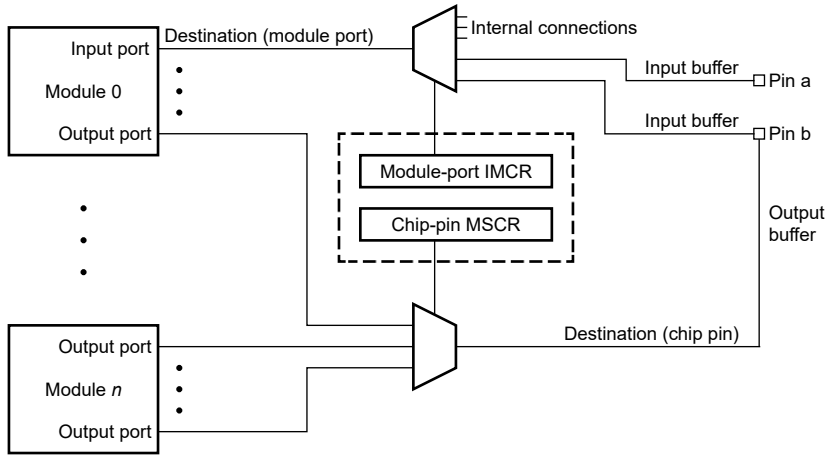



**Figure 20. MSCR and IMCR port and pin connection**

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>SSS | Source Signal Select<br>Selects which source signal is connected to the associated destination (chip pin). |

## 9.6.15 GPIO Pad Data Output (GPDO0 - GPDO219)

### Offset

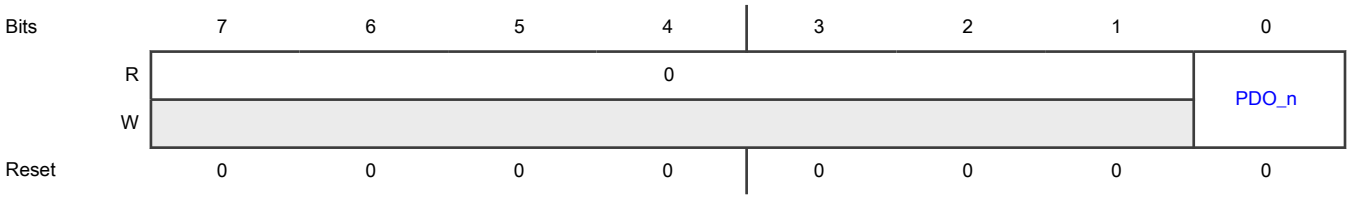
For n = 0 to 37; n = 40 to 219:

| Register | Offset |
|---|---|
| GPDOn | 1300h + (n + 3 - 2 × (n mod 4)) |

### Function

Writes 0 or 1 to a single GPIO pad with a byte access and supports 8-, 16-, and 32-bit accesses.

### Diagram




### Fields

| Field | Function |
|---|---|
| 7-1<br>— | Reserved |
| 0<br>PDO_n | Pad Data Out<br>Stores the data to be driven out on an external GPIO pad (controlled by this register) when you configure the pad as an output.<br>It also represents PDO[*n*], where *n* is the instance of the register.<br>0b - Logic low value<br>1b - Logic high value |

## 9.6.16 GPIO Pad Data Input (GPDI0 - GPDI219)

### Offset

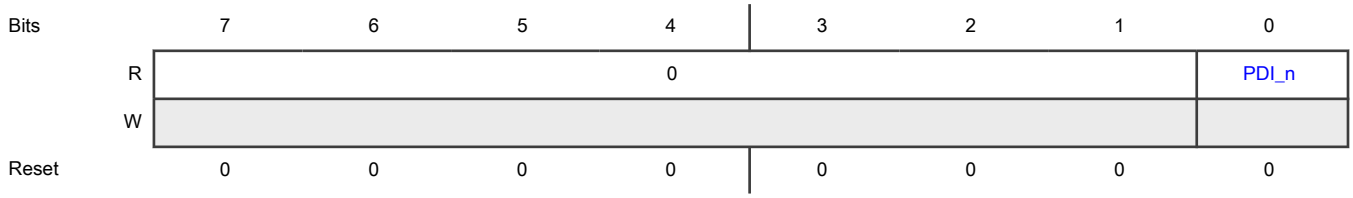
For n = 0 to 37; n = 40 to 219:

| Register | Offset |
|---|---|
| GPDIn | 1500h + (n + 3 - 2 × (n mod 4)) |

### Function

Reads the GPIO pad data with a byte access and supports 8-, 16-, and 32-bit accesses.

### Diagram




### Fields

| Field | Function |
|---|---|
| 7-1<br>— | Reserved |
| 0<br>PDI_n | Pad Data In<br>Stores the value of the external GPIO pad associated with this register.<br>It represents PDI[*n*], where *n* is the instance of the register.<br>0b - Logic low<br>1b - Logic high |

## 9.6.17 Parallel GPIO Pad Data Output (PGPDO0 - PGPDO3)

### Offset

| Register | Offset |
|---|---|
| PGPDO1 | 1700h |
| PGPDO0 | 1702h |
| PGPDO3 | 1704h |

### Function

Sets or clears the respective pads of the chip and supports 8-, 16-, and 32-bit accesses. This register also accesses the same physical resource as the PDO and MPGPDO address locations.

This register sets the values of all the output pins assigned to a chip port with a single 16-bit register write, while the GPDO*n* register sets the value on a specific pin with byte writes.

The access to this register's location is coherent with access to the bitwise GPDO*n*.

For a given PGPDO*x*[PPDO*y*], where *x* is the register instance index and *y* is the field index, the following equation shows the equivalent GPDO*n*[PDO_*n*] field:

PGPDO*x*[PPDO*y*] = GPDO(*x* × 16) + (15 - *y*)[PDO_(*x* × 16) + (15 - *y*)]

Following are some of the examples of mapping:

- PGPDO0[PPDO15] = GPDO0[PDO_0]
- PGPDO2[PPDO15] = GPDO32[PDO_32]
- PGPDO31[PPDO0] = GPDO511[PDO_511]

### Diagram

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PPDO 15 | PPDO 14 | PPDO 13 | PPDO 12 | PPDO 11 | PPDO 10 | PPDO 9 | PPDO 8 | PPDO 7 | PPDO 6 | PPDO 5 | PPDO 4 | PPDO 3 | PPDO 2 | PPDO 1 | PPDO 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 15<br>PPDO15 | Parallel Pad Data Out 15<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 14<br>PPDO14 | Parallel Pad Data Out 14<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 13<br>PPDO13 | Parallel Pad Data Out 13<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 12<br>PPDO12 | Parallel Pad Data Out 12<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 11<br>PPDO11 | Parallel Pad Data Out 11<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 10<br>PPDO10 | Parallel Pad Data Out 10<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 9<br>PPDO9 | Parallel Pad Data Out 9<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Logic low<br>1b - Logic high |
| 8<br>PPDO8 | Parallel Pad Data Out 8<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 7<br>PPDO7 | Parallel Pad Data Out 7<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 6<br>PPDO6 | Parallel Pad Data Out 6<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 5<br>PPDO5 | Parallel Pad Data Out 5<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 4<br>PPDO4 | Parallel Pad Data Out 4<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 3<br>PPDO3 | Parallel Pad Data Out 3<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 2<br>PPDO2 | Parallel Pad Data Out 2<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 1 | Parallel Pad Data Out 1 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| PPDO1 | Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 0<br>PPDO0 | Parallel Pad Data Out 0<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |

### 9.6.18 Parallel GPIO Pad Data Output (PGPDO2)

#### Offset

| Register | Offset |
|---|---|
| PGPDO2 | 1706h |

#### Function

Sets or clears the respective pads of the chip and supports 8-, 16-, and 32-bit accesses. This register also accesses the same physical resource as the PDO and MPGPDO address locations.

This register sets the values of all the output pins assigned to a chip port with a single 16-bit register write, while the GPDO*n* register sets the value on a specific pin with byte writes.

The access to this register's location is coherent with access to the bitwise GPDO*n*.

For a given PGPDO*x*[PPDO*y*], where *x* is the register instance index and *y* is the field index, the following equation shows the equivalent GPDO*n*[PDO_*n*] field:

PGPDO*x*[PPDO*y*] = GPDO(*x* × 16) + (15 - *y*)[PDO_(*x* × 16) + (15 - *y*)]

Following are some of the examples of mapping:

- PGPDO0[PPDO15] = GPDO0[PDO_0]
- PGPDO2[PPDO15] = GPDO32[PDO_32]
- PGPDO31[PPDO0] = GPDO511[PDO_511]

### Diagram

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

### Fields

| Field | Function |
|---|---|
| 15<br>PPDO15 | Parallel Pad Data Out 15<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 14<br>PPDO14 | Parallel Pad Data Out 14<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 13<br>PPDO13 | Parallel Pad Data Out 13<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 12<br>PPDO12 | Parallel Pad Data Out 12<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 11<br>PPDO11 | Parallel Pad Data Out 11<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 10<br>PPDO10 | Parallel Pad Data Out 10<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 9<br>— | Reserved<br>Always write zero to this field. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 8<br>— | Reserved<br>Always write zero to this field. |
| 7<br>PPDO7 | Parallel Pad Data Out 7<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 6<br>PPDO6 | Parallel Pad Data Out 6<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 5<br>PPDO5 | Parallel Pad Data Out 5<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 4<br>PPDO4 | Parallel Pad Data Out 4<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 3<br>PPDO3 | Parallel Pad Data Out 3<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 2<br>PPDO2 | Parallel Pad Data Out 2<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 1<br>PPDO1 | Parallel Pad Data Out 1<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 0 | Parallel Pad Data Out 0 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| PPDO0 | Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |

## 9.6.19 Parallel GPIO Pad Data Output (PGPDO4 - PGPDO11)

### Offset

For n = 4 to 11:

| Register | Offset |
|---|---|
| PGPDOn | 1708h + 2 × (n + 1 – 2 × (n mod 2)) |

### Function

Sets or clears the respective pads of the chip and supports 8-, 16-, and 32-bit accesses. This register also accesses the same physical resource as the PDO and MPGPDO address locations.

This register sets the values of all the output pins assigned to a chip port with a single 16-bit register write, while the GPDO*n* register sets the value on a specific pin with byte writes.

The access to this register's location is coherent with access to the bitwise GPDO*n*.

For a given PGPDO*x*[PPDO*y*], where *x* is the register instance index and *y* is the field index, the following equation shows the equivalent GPDO*n*[PDO_*n*] field:

PGPDO*x*[PPDO*y*] = GPDO(*x* × 16) + (15 - *y*)[PDO_(*x* × 16) + (15 - *y*)]

Following are some of the examples of mapping:

- PGPDO0[PPDO15] = GPDO0[PDO_0]
- PGPDO2[PPDO15] = GPDO32[PDO_32]
- PGPDO31[PPDO0] = GPDO511[PDO_511]

### Diagram

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

### Fields

| Field | Function |
|---|---|
| 15 | Parallel Pad Data Out 15 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| PPDO15 | Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 14<br>PPDO14 | Parallel Pad Data Out 14<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 13<br>PPDO13 | Parallel Pad Data Out 13<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 12<br>PPDO12 | Parallel Pad Data Out 12<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 11<br>PPDO11 | Parallel Pad Data Out 11<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 10<br>PPDO10 | Parallel Pad Data Out 10<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 9<br>PPDO9 | Parallel Pad Data Out 9<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 8<br>PPDO8 | Parallel Pad Data Out 8<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 7<br>PPDO7 | Parallel Pad Data Out 7<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 6<br>PPDO6 | Parallel Pad Data Out 6<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 5<br>PPDO5 | Parallel Pad Data Out 5<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 4<br>PPDO4 | Parallel Pad Data Out 4<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 3<br>PPDO3 | Parallel Pad Data Out 3<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 2<br>PPDO2 | Parallel Pad Data Out 2<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 1<br>PPDO1 | Parallel Pad Data Out 1<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 0<br>PPDO0 | Parallel Pad Data Out 0<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Logic high |

## 9.6.20 Parallel GPIO Pad Data Output (PGPDO13)

### Offset

| Register | Offset |
|---|---|
| PGPDO13 | 1718h |

### Function

Sets or clears the respective pads of the chip and supports 8-, 16-, and 32-bit accesses. This register also accesses the same physical resource as the PDO and MPGPDO address locations.

This register sets the values of all the output pins assigned to a chip port with a single 16-bit register write, while the GPDO*n* register sets the value on a specific pin with byte writes.

The access to this register's location is coherent with access to the bitwise GPDO*n*.

For a given PGPDO*x*[PPDO*y*], where *x* is the register instance index and *y* is the field index, the following equation shows the equivalent GPDO*n*[PDO_*n*] field:

PGPDO*x*[PPDO*y*] = GPDO(*x* × 16) + (15 - *y*)[PDO_(*x* × 16) + (15 - *y*)]

Following are some of the examples of mapping:

- PGPDO0[PPDO15] = GPDO0[PDO_0]
- PGPDO2[PPDO15] = GPDO32[PDO_32]
- PGPDO31[PPDO0] = GPDO511[PDO_511]

### Diagram

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

### Fields

| Field | Function |
|---|---|
| 15<br>PPDO15 | Parallel Pad Data Out 15<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Logic high |
| 14<br>PPDO14 | Parallel Pad Data Out 14<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 13<br>PPDO13 | Parallel Pad Data Out 13<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 12<br>PPDO12 | Parallel Pad Data Out 12<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 11<br>PPDO11 | Parallel Pad Data Out 11<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 10<br>PPDO10 | Parallel Pad Data Out 10<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 9<br>PPDO9 | Parallel Pad Data Out 9<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 8<br>PPDO8 | Parallel Pad Data Out 8<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 7<br>PPDO7 | Parallel Pad Data Out 7<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Logic low<br>1b - Logic high |
| 6<br>PPDO6 | Parallel Pad Data Out 6<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 5<br>PPDO5 | Parallel Pad Data Out 5<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 4<br>PPDO4 | Parallel Pad Data Out 4<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 3<br>— | Reserved<br>Always write zero to this field. |
| 2<br>— | Reserved<br>Always write zero to this field. |
| 1<br>— | Reserved<br>Always write zero to this field. |
| 0<br>— | Reserved<br>Always write zero to this field. |

## 9.6.21 Parallel GPIO Pad Data Output (PGPDO12)

### Offset

| Register | Offset |
|---|---|
| PGPDO12 | 171Ah |

### Function

Sets or clears the respective pads of the chip and supports 8-, 16-, and 32-bit accesses. This register also accesses the same physical resource as the PDO and MPGPDO address locations.

This register sets the values of all the output pins assigned to a chip port with a single 16-bit register write, while the GPDO*n* register sets the value on a specific pin with byte writes.

The access to this register's location is coherent with access to the bitwise GPDO*n*.

For a given PGPDO*x*[PPDO*y*], where *x* is the register instance index and *y* is the field index, the following equation shows the equivalent GPDO*n*[PDO_*n*] field:

PGPDO*x*[PPDO*y*] = GPDO(*x* × 16) + (15 - *y*)[PDO_(*x* × 16) + (15 - *y*)]

Following are some of the examples of mapping:

- PGPDO0[PPDO15] = GPDO0[PDO_0]
- PGPDO2[PPDO15] = GPDO32[PDO_32]
- PGPDO31[PPDO0] = GPDO511[PDO_511]

### Diagram

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

### Fields

| Field | Function |
|---|---|
| 15<br>PPDO15 | Parallel Pad Data Out 15<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 14<br>PPDO14 | Parallel Pad Data Out 14<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 13<br>PPDO13 | Parallel Pad Data Out 13<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 12<br>PPDO12 | Parallel Pad Data Out 12<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 11 | Parallel Pad Data Out 11 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| PPDO11 | Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 10<br>PPDO10 | Parallel Pad Data Out 10<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 9<br>PPDO9 | Parallel Pad Data Out 9<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 8<br>PPDO8 | Parallel Pad Data Out 8<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 7<br>PPDO7 | Parallel Pad Data Out 7<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 6<br>PPDO6 | Parallel Pad Data Out 6<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 5<br>PPDO5 | Parallel Pad Data Out 5<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 4<br>PPDO4 | Parallel Pad Data Out 4<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 3<br>PPDO3 | Parallel Pad Data Out 3<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 2<br>PPDO2 | Parallel Pad Data Out 2<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 1<br>PPDO1 | Parallel Pad Data Out 1<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |
| 0<br>PPDO0 | Parallel Pad Data Out 0<br>Writes to or reads the data register that stores the value to be driven on the pad in Output mode.<br>0b - Logic low<br>1b - Logic high |

## 9.6.22 Parallel GPIO Pad Data Input (PGPDI0 - PGPDI3)

### Offset

| Register | Offset |
|---|---|
| PGPDI1 | 1740h |
| PGPDI0 | 1742h |
| PGPDI3 | 1744h |

### Function

Holds the synchronized input value from the pads and supports 8-, 16-, and 32-bit accesses.

This register reads the values of all input pins assigned to a chip port with a single 16-bit register read, while GPDI*n* registers read the value on a specific pin with a byte read.

The access to this register's location is coherent with the access to the bitwise GPDI*n*.

For a given PGPDI*x*[PPDI*y*], where *x* is the register instance index and *y* is the field index, the following equation shows the equivalent GPDI*n*[PDI_*n*] field:

PGPDI*x*[PPDI*y*] = GPDI(*x* × 16) + (15 - *y*)[PDI_(*x* × 16) + (15 - *y*)]

Following are some of the examples of mapping:

- PGPDI0[PPDI15] = GPDI0[PDI_0]
- PGPDI2[PPDI15] = GPDI32[PDI_32]
- PGPDI31[PPDI0] = GPDI511[PDI_511]

### Diagram

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PPDI15 | PPDI14 | PPDI13 | PPDI12 | PPDI11 | PPDI10 | PPDI9 | PPDI8 | PPDI7 | PPDI6 | PPDI5 | PPDI4 | PPDI3 | PPDI2 | PPDI1 | PPDI0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 15<br>PPDI15 | Parallel Pad Data Input 15<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 14<br>PPDI14 | Parallel Pad Data Input 14<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 13<br>PPDI13 | Parallel Pad Data Input 13<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 12<br>PPDI12 | Parallel Pad Data Input 12<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 11<br>PPDI11 | Parallel Pad Data Input 11<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 10<br>PPDI10 | Parallel Pad Data Input 10<br>Reads the current pad value of the corresponding pad. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Logic low<br>1b - Logic high |
| 9<br>PPDI9 | Parallel Pad Data Input 9<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 8<br>PPDI8 | Parallel Pad Data Input 8<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 7<br>PPDI7 | Parallel Pad Data Input 7<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 6<br>PPDI6 | Parallel Pad Data Input 6<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 5<br>PPDI5 | Parallel Pad Data Input 5<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 4<br>PPDI4 | Parallel Pad Data Input 4<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 3<br>PPDI3 | Parallel Pad Data Input 3<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 2 | Parallel Pad Data Input 2 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| PPDI2 | Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 1<br>PPDI1 | Parallel Pad Data Input 1<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 0<br>PPDI0 | Parallel Pad Data Input 0<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |

## 9.6.23 Parallel GPIO Pad Data Input (PGPDI2)

### Offset

| Register | Offset |
|---|---|
| PGPDI2 | 1746h |

### Function

Holds the synchronized input value from the pads and supports 8-, 16-, and 32-bit accesses.

This register reads the values of all input pins assigned to a chip port with a single 16-bit register read, while GPDI*n* registers read the value on a specific pin with a byte read.

The access to this register's location is coherent with the access to the bitwise GPDI*n*.

For a given PGPDI*x*[PPDI*y*], where *x* is the register instance index and *y* is the field index, the following equation shows the equivalent GPDI*n*[PDI_*n*] field:

PGPDI*x*[PPDI*y*] = GPDI(*x* × 16) + (15 - *y*)[PDI_(*x* × 16) + (15 - *y*)]

Following are some of the examples of mapping:

- PGPDI0[PPDI15] = GPDI0[PDI_0]
- PGPDI2[PPDI15] = GPDI32[PDI_32]
- PGPDI31[PPDI0] = GPDI511[PDI_511]

### Diagram

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PPDI15 | PPDI14 | PPDI13 | PPDI12 | PPDI11 | PPDI10 | Reserved | Reserved | PPDI7 | PPDI6 | PPDI5 | PPDI4 | PPDI3 | PPDI2 | PPDI1 | PPDI0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 15<br>PPDI15 | Parallel Pad Data Input 15<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 14<br>PPDI14 | Parallel Pad Data Input 14<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 13<br>PPDI13 | Parallel Pad Data Input 13<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 12<br>PPDI12 | Parallel Pad Data Input 12<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 11<br>PPDI11 | Parallel Pad Data Input 11<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 10<br>PPDI10 | Parallel Pad Data Input 10<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 9 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 8<br>— | Reserved |
| 7<br>PPDI7 | Parallel Pad Data Input 7<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 6<br>PPDI6 | Parallel Pad Data Input 6<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 5<br>PPDI5 | Parallel Pad Data Input 5<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 4<br>PPDI4 | Parallel Pad Data Input 4<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 3<br>PPDI3 | Parallel Pad Data Input 3<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 2<br>PPDI2 | Parallel Pad Data Input 2<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 1<br>PPDI1 | Parallel Pad Data Input 1<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 0<br>PPDI0 | Parallel Pad Data Input 0<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |

## 9.6.24 Parallel GPIO Pad Data Input (PGPDI4 - PGPDI11)

### Offset

For n = 4 to 11:

| Register | Offset |
|---|---|
| PGPDIn | 1748h + 2 × (n + 1 – 2 × (n mod 2)) |

### Function

Holds the synchronized input value from the pads and supports 8-, 16-, and 32-bit accesses.

This register reads the values of all input pins assigned to a chip port with a single 16-bit register read, while GPDI*n* registers read the value on a specific pin with a byte read.

The access to this register's location is coherent with the access to the bitwise GPDI*n*.

For a given PGPDI*x*[PPDI*y*], where *x* is the register instance index and *y* is the field index, the following equation shows the equivalent GPDI*n*[PDI_*n*] field:

PGPDI*x*[PPDI*y*] = GPDI(*x* × 16) + (15 - *y*)[PDI_(*x* × 16) + (15 - *y*)]

Following are some of the examples of mapping:

- PGPDI0[PPDI15] = GPDI0[PDI_0]
- PGPDI2[PPDI15] = GPDI32[PDI_32]
- PGPDI31[PPDI0] = GPDI511[PDI_511]

### Diagram

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PPDI15 | PPDI14 | PPDI13 | PPDI12 | PPDI11 | PPDI10 | PPDI9 | PPDI8 | PPDI7 | PPDI6 | PPDI5 | PPDI4 | PPDI3 | PPDI2 | PPDI1 | PPDI0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 15<br>PPDI15 | Parallel Pad Data Input 15<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 14<br>PPDI14 | Parallel Pad Data Input 14<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 13<br>PPDI13 | Parallel Pad Data Input 13<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 12<br>PPDI12 | Parallel Pad Data Input 12<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 11<br>PPDI11 | Parallel Pad Data Input 11<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 10<br>PPDI10 | Parallel Pad Data Input 10<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 9 | Parallel Pad Data Input 9 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| PPDI9 | Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 8<br>PPDI8 | Parallel Pad Data Input 8<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 7<br>PPDI7 | Parallel Pad Data Input 7<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 6<br>PPDI6 | Parallel Pad Data Input 6<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 5<br>PPDI5 | Parallel Pad Data Input 5<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 4<br>PPDI4 | Parallel Pad Data Input 4<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 3<br>PPDI3 | Parallel Pad Data Input 3<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 2<br>PPDI2 | Parallel Pad Data Input 2<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 1<br>PPDI1 | Parallel Pad Data Input 1<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 0<br>PPDI0 | Parallel Pad Data Input 0<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |

## 9.6.25 Parallel GPIO Pad Data Input (PGPDI13)

### Offset

| Register | Offset |
|---|---|
| PGPDI13 | 1758h |

### Function

Holds the synchronized input value from the pads and supports 8-, 16-, and 32-bit accesses.

This register reads the values of all input pins assigned to a chip port with a single 16-bit register read, while GPDI*n* registers read the value on a specific pin with a byte read.

The access to this register's location is coherent with the access to the bitwise GPDI*n*.

For a given PGPDI*x*[PPDI*y*], where *x* is the register instance index and *y* is the field index, the following equation shows the equivalent GPDI*n*[PDI_*n*] field:

PGPDI*x*[PPDI*y*] = GPDI(*x* × 16) + (15 - *y*)[PDI_(*x* × 16) + (15 - *y*)]

Following are some of the examples of mapping:

- PGPDI0[PPDI15] = GPDI0[PDI_0]
- PGPDI2[PPDI15] = GPDI32[PDI_32]
- PGPDI31[PPDI0] = GPDI511[PDI_511]

### Diagram

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PPDI15 | PPDI14 | PPDI13 | PPDI12 | PPDI11 | PPDI10 | PPDI9 | PPDI8 | PPDI7 | PPDI6 | PPDI5 | PPDI4 | Reserved | Reserved | Reserved | Reserved |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 15<br>PPDI15 | Parallel Pad Data Input 15<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 14<br>PPDI14 | Parallel Pad Data Input 14<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 13<br>PPDI13 | Parallel Pad Data Input 13<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 12<br>PPDI12 | Parallel Pad Data Input 12<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 11<br>PPDI11 | Parallel Pad Data Input 11<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 10<br>PPDI10 | Parallel Pad Data Input 10<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 9 | Parallel Pad Data Input 9 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| PPDI9 | Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 8<br>PPDI8 | Parallel Pad Data Input 8<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 7<br>PPDI7 | Parallel Pad Data Input 7<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 6<br>PPDI6 | Parallel Pad Data Input 6<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 5<br>PPDI5 | Parallel Pad Data Input 5<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 4<br>PPDI4 | Parallel Pad Data Input 4<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 3<br>— | Reserved |
| 2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>— | Reserved |

## 9.6.26 Parallel GPIO Pad Data Input (PGPDI12)

### Offset

| Register | Offset |
|---|---|
| PGPDI12 | 175Ah |

### Function

Holds the synchronized input value from the pads and supports 8-, 16-, and 32-bit accesses.

This register reads the values of all input pins assigned to a chip port with a single 16-bit register read, while GPDI*n* registers read the value on a specific pin with a byte read.

The access to this register's location is coherent with the access to the bitwise GPDI*n*.

For a given PGPDI*x*[PPDI*y*], where *x* is the register instance index and *y* is the field index, the following equation shows the equivalent GPDI*n*[PDI_*n*] field:

PGPDI*x*[PPDI*y*] = GPDI(*x* × 16) + (15 - *y*)[PDI_(*x* × 16) + (15 - *y*)]

Following are some of the examples of mapping:

- PGPDI0[PPDI15] = GPDI0[PDI_0]
- PGPDI2[PPDI15] = GPDI32[PDI_32]
- PGPDI31[PPDI0] = GPDI511[PDI_511]

### Diagram

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PPDI15 | PPDI14 | PPDI13 | PPDI12 | PPDI11 | PPDI10 | PPDI9 | PPDI8 | PPDI7 | PPDI6 | PPDI5 | PPDI4 | PPDI3 | PPDI2 | PPDI1 | PPDI0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 15<br>PPDI15 | Parallel Pad Data Input 15<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 14<br>PPDI14 | Parallel Pad Data Input 14<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 13<br>PPDI13 | Parallel Pad Data Input 13<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 12<br>PPDI12 | Parallel Pad Data Input 12<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 11<br>PPDI11 | Parallel Pad Data Input 11<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 10<br>PPDI10 | Parallel Pad Data Input 10<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 9<br>PPDI9 | Parallel Pad Data Input 9<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 8<br>PPDI8 | Parallel Pad Data Input 8<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 7<br>PPDI7 | Parallel Pad Data Input 7<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 6<br>PPDI6 | Parallel Pad Data Input 6<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Logic high |
| 5<br>PPDI5 | Parallel Pad Data Input 5<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 4<br>PPDI4 | Parallel Pad Data Input 4<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 3<br>PPDI3 | Parallel Pad Data Input 3<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 2<br>PPDI2 | Parallel Pad Data Input 2<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 1<br>PPDI1 | Parallel Pad Data Input 1<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |
| 0<br>PPDI0 | Parallel Pad Data Input 0<br>Reads the current pad value of the corresponding pad.<br>0b - Logic low<br>1b - Logic high |

## 9.6.27 Masked Parallel GPIO Pad Data Output (MPGPDO0 - MPGPDO1)

**Offset**

| Register | Offset |
|---|---|
| MPGPDO0 | 1780h |
| MPGPDO1 | 1784h |

### Function

Modifies the pad values associated with PGPDO*n* selectively and supports only 32-bit accesses. It does not support byte and half-word accesses.

**NOTE**

Access this register only with 32-bit writes. 8-bit or 16-bit writes do not modify any bits in the register resulting in a transfer error. Read access returns 0.

The accesses to each of this register location is coherent with access to the bitwise GPDO*n*.

For a given MPGPDO*x*[MPPDO*y*], where *x* is the register instance index and *y* is the field index, the following equation shows the equivalent GPDO*n*[PDO_*n*]:

MPGPDO*x*[MPPDO*y*] = GPDO(*x* × 16) + (15 - *y*)[PDO_(*x* × 16) + (15 - *y*)]

Following are some of the examples of mapping:

- MPGPDO0[MPPDO15] = GPDO0[PDO_0]
- MPGPDO2[MPPDO15] = GPDO32[PDO_32]
- MPGPDO31[MPPDO0] = GPDO511[PDO_511]

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | MASK 15 | MASK 14 | MASK 13 | MASK 12 | MASK 11 | MASK 10 | MASK 9 | MASK 8 | MASK 7 | MASK 6 | MASK 5 | MASK 4 | MASK 3 | MASK 2 | MASK 1 | MASK 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | MPPDO15 | MPPDO14 | MPPDO13 | MPPDO12 | MPPDO11 | MPPDO10 | MPPDO9 | MPPDO8 | MPPDO7 | MPPDO6 | MPPDO5 | MPPDO4 | MPPDO3 | MPPDO2 | MPPDO1 | MPPDO0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>MASK15 | Mask Field 15<br>Masks MPPDO15 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO15 is ignored<br>1b - MPPDO15 is written |
| 30<br>MASK14 | Mask Field 14<br>Masks MPPDO14 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO14 is ignored<br>1b - MPPDO14 is written |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 29<br>MASK13 | Mask Field 13<br>Masks MPPDO13 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO13 is ignored<br>1b - MPPDO13 is written |
| 28<br>MASK12 | Mask Field 12<br>Masks MPPDO12 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO12 is ignored<br>1b - MPPDO12 is written |
| 27<br>MASK11 | Mask Field 11<br>Masks MPPDO11 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO11 is ignored<br>1b - MPPDO11 is written |
| 26<br>MASK10 | Mask Field 10<br>Masks MPPDO10 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO10 is ignored<br>1b - MPPDO10 is written |
| 25<br>MASK9 | Mask Field 9<br>Masks MPPDO9 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO9 is ignored<br>1b - MPPDO9 is written |
| 24<br>MASK8 | Mask Field 8<br>Masks MPPDO8 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO8 is ignored<br>1b - MPPDO8 is written |
| 23<br>MASK7 | Mask Field 7<br>Masks MPPDO7 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO7 is ignored<br>1b - MPPDO7 is written |
| 22<br>MASK6 | Mask Field 6<br>Masks MPPDO6 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO6 is ignored |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - MPPDO6 is written |
| 21<br>MASK5 | Mask Field 5<br>Masks MPPDO5 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO5 is ignored<br>1b - MPPDO5 is written |
| 20<br>MASK4 | Mask Field 4<br>Masks MPPDO4 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO4 is ignored<br>1b - MPPDO4 is written |
| 19<br>MASK3 | Mask Field 3<br>Masks MPPDO3 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO3 is ignored<br>1b - MPPDO3 is written |
| 18<br>MASK2 | Mask Field 2<br>Masks MPPDO2 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO2 is ignored<br>1b - MPPDO2 is written |
| 17<br>MASK1 | Mask Field 1<br>Masks MPPDO1 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO1 is ignored<br>1b - MPPDO1 is written |
| 16<br>MASK0 | Mask Field 0<br>Masks MPPDO0 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO0 is ignored<br>1b - MPPDO0 is written |
| 15<br>MPPDO15 | Masked Parallel Pad Data Out 15<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 14<br>MPPDO14 | Masked Parallel Pad Data Out 14<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 13 | Masked Parallel Pad Data Out 13 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| MPPDO13 | Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 12<br>MPPDO12 | Masked Parallel Pad Data Out 12<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 11<br>MPPDO11 | Masked Parallel Pad Data Out 11<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 10<br>MPPDO10 | Masked Parallel Pad Data Out 10<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 9<br>MPPDO9 | Masked Parallel Pad Data Out 9<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 8<br>MPPDO8 | Masked Parallel Pad Data Out 8<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 7<br>MPPDO7 | Masked Parallel Pad Data Out 7<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 6<br>MPPDO6 | Masked Parallel Pad Data Out 6<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 5<br>MPPDO5 | Masked Parallel Pad Data Out 5<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 4<br>MPPDO4 | Masked Parallel Pad Data Out 4<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 3<br>MPPDO3 | Masked Parallel Pad Data Out 3<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 2<br>MPPDO2 | Masked Parallel Pad Data Out 2<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 1<br>MPPDO1 | Masked Parallel Pad Data Out 1<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 0<br>MPPDO0 | Masked Parallel Pad Data Out 0<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |

## 9.6.28 Masked Parallel GPIO Pad Data Output (MPGPDO2)

### Offset

| Register | Offset |
|---|---|
| MPGPDO2 | 1788h |

### Function

Modifies the pad values associated with PGPDO*n* selectively and supports only 32-bit accesses. It does not support byte and half-word accesses.

**NOTE**

Access this register only with 32-bit writes. 8-bit or 16-bit writes do not modify any bits in the register resulting in a transfer error. Read access returns 0.

The accesses to each of this register location is coherent with access to the bitwise GPDO*n*.

For a given MPGPDO*x*[MPPDO*y*], where *x* is the register instance index and *y* is the field index, the following equation shows the equivalent GPDO*n*[PDO_*n*]:

MPGPDO*x*[MPPDO*y*] = GPDO(*x* × 16) + (15 - *y*)[PDO_(*x* × 16) + (15 - *y*)]

Following are some of the examples of mapping:

- MPGPDO0[MPPDO15] = GPDO0[PDO_0]
- MPGPDO2[MPPDO15] = GPDO32[PDO_32]
- MPGPDO31[MPPDO0] = GPDO511[PDO_511]

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | MASK 15 | MASK 14 | MASK 13 | MASK 12 | MASK 11 | MASK 10 | Reserved | Reserved | MASK 7 | MASK 6 | MASK 5 | MASK 4 | MASK 3 | MASK 2 | MASK 1 | MASK 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | MPPDO15 | MPPDO14 | MPPDO13 | MPPDO12 | MPPDO11 | MPPDO10 | Reserved | Reserved | MPPDO7 | MPPDO6 | MPPDO5 | MPPDO4 | MPPDO3 | MPPDO2 | MPPDO1 | MPPDO0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>MASK15 | Mask Field 15<br>Masks MPPDO15 in the corresponding MPGPDO*n* register instance. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - MPPDO15 is ignored<br>1b - MPPDO15 is written |
| 30<br>MASK14 | Mask Field 14<br>Masks MPPDO14 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO14 is ignored<br>1b - MPPDO14 is written |
| 29<br>MASK13 | Mask Field 13<br>Masks MPPDO13 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO13 is ignored<br>1b - MPPDO13 is written |
| 28<br>MASK12 | Mask Field 12<br>Masks MPPDO12 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO12 is ignored<br>1b - MPPDO12 is written |
| 27<br>MASK11 | Mask Field 11<br>Masks MPPDO11 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO11 is ignored<br>1b - MPPDO11 is written |
| 26<br>MASK10 | Mask Field 10<br>Masks MPPDO10 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO10 is ignored<br>1b - MPPDO10 is written |
| 25<br>— | Reserved<br>Always write zero to this field. |
| 24<br>— | Reserved<br>Always write zero to this field. |
| 23<br>MASK7 | Mask Field 7<br>Masks MPPDO7 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO7 is ignored<br>1b - MPPDO7 is written |
| 22 | Mask Field 6 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| MASK6 | Masks MPPDO6 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO6 is ignored<br>1b - MPPDO6 is written |
| 21<br>MASK5 | Mask Field 5<br>Masks MPPDO5 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO5 is ignored<br>1b - MPPDO5 is written |
| 20<br>MASK4 | Mask Field 4<br>Masks MPPDO4 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO4 is ignored<br>1b - MPPDO4 is written |
| 19<br>MASK3 | Mask Field 3<br>Masks MPPDO3 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO3 is ignored<br>1b - MPPDO3 is written |
| 18<br>MASK2 | Mask Field 2<br>Masks MPPDO2 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO2 is ignored<br>1b - MPPDO2 is written |
| 17<br>MASK1 | Mask Field 1<br>Masks MPPDO1 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO1 is ignored<br>1b - MPPDO1 is written |
| 16<br>MASK0 | Mask Field 0<br>Masks MPPDO0 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO0 is ignored<br>1b - MPPDO0 is written |
| 15<br>MPPDO15 | Masked Parallel Pad Data Out 15<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 14<br>MPPDO14 | Masked Parallel Pad Data Out 14<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 13<br>MPPDO13 | Masked Parallel Pad Data Out 13<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 12<br>MPPDO12 | Masked Parallel Pad Data Out 12<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 11<br>MPPDO11 | Masked Parallel Pad Data Out 11<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 10<br>MPPDO10 | Masked Parallel Pad Data Out 10<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 9<br>— | Reserved<br>Always write zero to this field. |
| 8<br>— | Reserved<br>Always write zero to this field. |
| 7<br>MPPDO7 | Masked Parallel Pad Data Out 7<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 6<br>MPPDO6 | Masked Parallel Pad Data Out 6<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 5<br>MPPDO5 | Masked Parallel Pad Data Out 5<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 4<br>MPPDO4 | Masked Parallel Pad Data Out 4<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 3<br>MPPDO3 | Masked Parallel Pad Data Out 3<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 2<br>MPPDO2 | Masked Parallel Pad Data Out 2<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 1<br>MPPDO1 | Masked Parallel Pad Data Out 1<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 0<br>MPPDO0 | Masked Parallel Pad Data Out 0<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |

## 9.6.29 Masked Parallel GPIO Pad Data Output (MPGPDO3 - MPGPDO12)

### Offset

For a = 3 to 12:

| Register | Offset |
|---|---|
| MPGPDOa | 1780h + (a × 4h) |

### Function

Modifies the pad values associated with PGPDO*n* selectively and supports only 32-bit accesses. It does not support byte and half-word accesses.

**NOTE**

Access this register only with 32-bit writes. 8-bit or 16-bit writes do not modify any bits in the register resulting in a transfer error. Read access returns 0.

The accesses to each of this register location is coherent with access to the bitwise GPDO*n*.

For a given MPGPDO*x*[MPPDO*y*], where *x* is the register instance index and *y* is the field index, the following equation shows the equivalent GPDO*n*[PDO_*n*]:

MPGPDO*x*[MPPDO*y*] = GPDO(*x* × 16) + (15 - *y*)[PDO_(*x* × 16) + (15 - *y*)]

Following are some of the examples of mapping:

- MPGPDO0[MPPDO15] = GPDO0[PDO_0]
- MPGPDO2[MPPDO15] = GPDO32[PDO_32]
- MPGPDO31[MPPDO0] = GPDO511[PDO_511]

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | MASK 15 | MASK 14 | MASK 13 | MASK 12 | MASK 11 | MASK 10 | MASK 9 | MASK 8 | MASK 7 | MASK 6 | MASK 5 | MASK 4 | MASK 3 | MASK 2 | MASK 1 | MASK 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | MPPDO15 | MPPDO14 | MPPDO13 | MPPDO12 | MPPDO11 | MPPDO10 | MPPDO9 | MPPDO8 | MPPDO7 | MPPDO6 | MPPDO5 | MPPDO4 | MPPDO3 | MPPDO2 | MPPDO1 | MPPDO0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31 | Mask Field 15 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| MASK15 | Masks MPPDO15 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO15 is ignored<br>1b - MPPDO15 is written |
| 30<br>MASK14 | Mask Field 14<br>Masks MPPDO14 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO14 is ignored<br>1b - MPPDO14 is written |
| 29<br>MASK13 | Mask Field 13<br>Masks MPPDO13 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO13 is ignored<br>1b - MPPDO13 is written |
| 28<br>MASK12 | Mask Field 12<br>Masks MPPDO12 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO12 is ignored<br>1b - MPPDO12 is written |
| 27<br>MASK11 | Mask Field 11<br>Masks MPPDO11 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO11 is ignored<br>1b - MPPDO11 is written |
| 26<br>MASK10 | Mask Field 10<br>Masks MPPDO10 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO10 is ignored<br>1b - MPPDO10 is written |
| 25<br>MASK9 | Mask Field 9<br>Masks MPPDO9 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO9 is ignored<br>1b - MPPDO9 is written |
| 24<br>MASK8 | Mask Field 8<br>Masks MPPDO8 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO8 is ignored<br>1b - MPPDO8 is written |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 23<br>MASK7 | Mask Field 7<br>Masks MPPDO7 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO7 is ignored<br>1b - MPPDO7 is written |
| 22<br>MASK6 | Mask Field 6<br>Masks MPPDO6 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO6 is ignored<br>1b - MPPDO6 is written |
| 21<br>MASK5 | Mask Field 5<br>Masks MPPDO5 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO5 is ignored<br>1b - MPPDO5 is written |
| 20<br>MASK4 | Mask Field 4<br>Masks MPPDO4 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO4 is ignored<br>1b - MPPDO4 is written |
| 19<br>MASK3 | Mask Field 3<br>Masks MPPDO3 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO3 is ignored<br>1b - MPPDO3 is written |
| 18<br>MASK2 | Mask Field 2<br>Masks MPPDO2 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO2 is ignored<br>1b - MPPDO2 is written |
| 17<br>MASK1 | Mask Field 1<br>Masks MPPDO1 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO1 is ignored<br>1b - MPPDO1 is written |
| 16<br>MASK0 | Mask Field 0<br>Masks MPPDO0 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO0 is ignored |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - MPPDO0 is written |
| 15<br>MPPDO15 | Masked Parallel Pad Data Out 15<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 14<br>MPPDO14 | Masked Parallel Pad Data Out 14<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 13<br>MPPDO13 | Masked Parallel Pad Data Out 13<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 12<br>MPPDO12 | Masked Parallel Pad Data Out 12<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 11<br>MPPDO11 | Masked Parallel Pad Data Out 11<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 10<br>MPPDO10 | Masked Parallel Pad Data Out 10<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 9<br>MPPDO9 | Masked Parallel Pad Data Out 9<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 8<br>MPPDO8 | Masked Parallel Pad Data Out 8<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 7<br>MPPDO7 | Masked Parallel Pad Data Out 7<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 6<br>MPPDO6 | Masked Parallel Pad Data Out 6<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 5<br>MPPDO5 | Masked Parallel Pad Data Out 5<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 4<br>MPPDO4 | Masked Parallel Pad Data Out 4<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 3<br>MPPDO3 | Masked Parallel Pad Data Out 3<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 2<br>MPPDO2 | Masked Parallel Pad Data Out 2<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 1<br>MPPDO1 | Masked Parallel Pad Data Out 1<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 0<br>MPPDO0 | Masked Parallel Pad Data Out 0<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |

### 9.6.30 Masked Parallel GPIO Pad Data Output (MPGPDO13)

#### Offset

| Register | Offset |
|---|---|
| MPGPDO13 | 17B4h |

#### Function

Modifies the pad values associated with PGPDO*n* selectively and supports only 32-bit accesses. It does not support byte and half-word accesses.

**NOTE**

Access this register only with 32-bit writes. 8-bit or 16-bit writes do not modify any bits in the register resulting in a transfer error. Read access returns 0.

The accesses to each of this register location is coherent with access to the bitwise GPDO*n*.

For a given MPGPDO*x*[MPPDO*y*], where *x* is the register instance index and *y* is the field index, the following equation shows the equivalent GPDO*n*[PDO_*n*]:

MPGPDO*x*[MPPDO*y*] = GPDO(*x* × 16) + (15 - *y*)[PDO_(*x* × 16) + (15 - *y*)]

Following are some of the examples of mapping:

- MPGPDO0[MPPDO15] = GPDO0[PDO_0]
- MPGPDO2[MPPDO15] = GPDO32[PDO_32]
- MPGPDO31[MPPDO0] = GPDO511[PDO_511]

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | MASK 15 | MASK 14 | MASK 13 | MASK 12 | MASK 11 | MASK 10 | MASK 9 | MASK 8 | MASK 7 | MASK 6 | MASK 5 | MASK 4 | Reserved | Reserved | Reserved | Reserved |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | MPPDO15 | MPPDO14 | MPPDO13 | MPPDO12 | MPPDO11 | MPPDO10 | MPPDO9 | MPPDO8 | MPPDO7 | MPPDO6 | MPPDO5 | MPPDO4 | Reserved | Reserved | Reserved | Reserved |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>MASK15 | Mask Field 15<br>Masks MPPDO15 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO15 is ignored<br>1b - MPPDO15 is written |
| 30<br>MASK14 | Mask Field 14<br>Masks MPPDO14 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO14 is ignored<br>1b - MPPDO14 is written |
| 29<br>MASK13 | Mask Field 13<br>Masks MPPDO13 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO13 is ignored<br>1b - MPPDO13 is written |
| 28<br>MASK12 | Mask Field 12<br>Masks MPPDO12 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO12 is ignored<br>1b - MPPDO12 is written |
| 27<br>MASK11 | Mask Field 11<br>Masks MPPDO11 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO11 is ignored<br>1b - MPPDO11 is written |
| 26 | Mask Field 10 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| MASK10 | Masks MPPDO10 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO10 is ignored<br>1b - MPPDO10 is written |
| 25<br>MASK9 | Mask Field 9<br>Masks MPPDO9 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO9 is ignored<br>1b - MPPDO9 is written |
| 24<br>MASK8 | Mask Field 8<br>Masks MPPDO8 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO8 is ignored<br>1b - MPPDO8 is written |
| 23<br>MASK7 | Mask Field 7<br>Masks MPPDO7 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO7 is ignored<br>1b - MPPDO7 is written |
| 22<br>MASK6 | Mask Field 6<br>Masks MPPDO6 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO6 is ignored<br>1b - MPPDO6 is written |
| 21<br>MASK5 | Mask Field 5<br>Masks MPPDO5 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO5 is ignored<br>1b - MPPDO5 is written |
| 20<br>MASK4 | Mask Field 4<br>Masks MPPDO4 in the corresponding MPGPDO*n* register instance.<br>0b - MPPDO4 is ignored<br>1b - MPPDO4 is written |
| 19<br>— | Reserved<br>Always write zero to this field. |
| 18<br>— | Reserved<br>Always write zero to this field. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 17<br>— | Reserved<br>Always write zero to this field. |
| 16<br>— | Reserved<br>Always write zero to this field. |
| 15<br>MPPDO15 | Masked Parallel Pad Data Out 15<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 14<br>MPPDO14 | Masked Parallel Pad Data Out 14<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 13<br>MPPDO13 | Masked Parallel Pad Data Out 13<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 12<br>MPPDO12 | Masked Parallel Pad Data Out 12<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 11<br>MPPDO11 | Masked Parallel Pad Data Out 11<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 10<br>MPPDO10 | Masked Parallel Pad Data Out 10<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 9<br>MPPDO9 | Masked Parallel Pad Data Out 9<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 8<br>MPPDO8 | Masked Parallel Pad Data Out 8<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 7<br>MPPDO7 | Masked Parallel Pad Data Out 7<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 6<br>MPPDO6 | Masked Parallel Pad Data Out 6<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 5<br>MPPDO5 | Masked Parallel Pad Data Out 5<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 4<br>MPPDO4 | Masked Parallel Pad Data Out 4<br>Writes to the data register that stores the value to be driven on the pad in Output mode. |
| 3 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | Always write zero to this field. |
| 2<br>— | Reserved<br>Always write zero to this field. |
| 1<br>— | Reserved<br>Always write zero to this field. |
| 0<br>— | Reserved<br>Always write zero to this field. |

# Chapter 10
# Touch Sensing Pin Coupling (TSPC)

## 10.1 Chip-specific TSPC information

### 10.1.1 TSPC instances and configuration

This chip has one instance of TSPC module. The GPIO pads being used for TSPC should be configured with Source Signal Select (SSS) = 0. TSPC module works in conjunction with SIUL2.

For the GPIOs and ADC pads supporting TSPC, see the IOMUX file attached to this document. There are two groups of pads which can be configured via TSPC:

1. Group 1 consists of 32 ADC channels and 14 GPIOs
2. Group 2 consists of 32 ADC channels and 6 GPIOs

**NOTE**

Use preferably the same type of I/O pins for driving each electrode. Such I/O pins matching will improve resolution of the capacitance sensing. For more information about types of I/O pins, see the IOMUX file attached to this document.

## 10.2 Overview

TSPC provides simultaneous transitioning of a group of pads into high impedance to support a more robust recognition of touch sense.

### 10.2.1 Block diagram

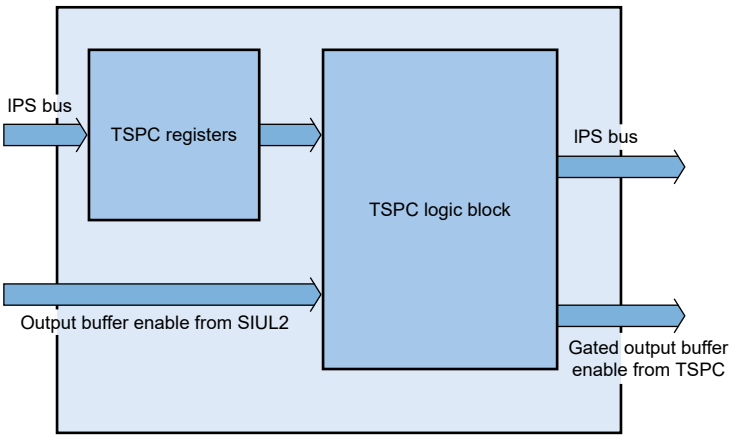



Figure 21. TSPC block diagram

### 10.2.2 Features

TSPC provides a set of control registers for pairing up GPIO pads for the touch sensing pairing operation. These control registers select the set of GPIO pads to pair with the set of ADC channels of an ADC instance. See the chip-specific TSPC configuration for group of pads information.

TSPC supports the following features:

- Up to 512 GPIO pads allow grouping

- OBE of all the grouped pads transitions from high to low simultaneously if SIUL2 transitions a pad's OBE from high to low
- Programming of group registers defines grouping

## 10.3 Functional description

TSPC contains these types of registers:

- Group enable: Each field of this register enables the respective group's OBE register—that is, if any OBE within a group transitions from high to low, it enables that group of OBEs for simultaneous transition from high to low. For example, GRP1_EN is the enable field for group 1 OBE registers. The number of fields in this register is equal to the number of groups.
- Group *n* OBE register *n*: The number of registers for a particular group *n* depends on the number of grouped OBEs in the respective group. For example, if the number of OBEs in group 1 is 36, then two registers of 32 fields each exist. Each field of this register represents an OBE pad. If you write 1 to the field, it indicates the OBE's participation in the grouping. If one of the pad's OBEs transitions from high to low, all the participating OBEs transition from high to low. The remaining OBEs that are not participating in the grouping are not affected.

### 10.3.1 Submodules

#### 10.3.1.1 Logic block description

This block takes the information from register fields to produce gated OBEs.

The simultaneous transition of grouped OBEs occurs when the GRP*n*_EN field in Group Enable (GRP_EN) for the respective group becomes 1.

The logic diagram below shows that if the GRP_EN is high for a particular group, and one of the output buffer enable (ipp_obe_in_b) from SIUL2 belonging to that group transitions from high to low, then all the other output buffer enables belonging to that group also transitions from high to low. If the GRP_EN is set low, the transitions in OBE do not affect the other OBE's belonging to that group.

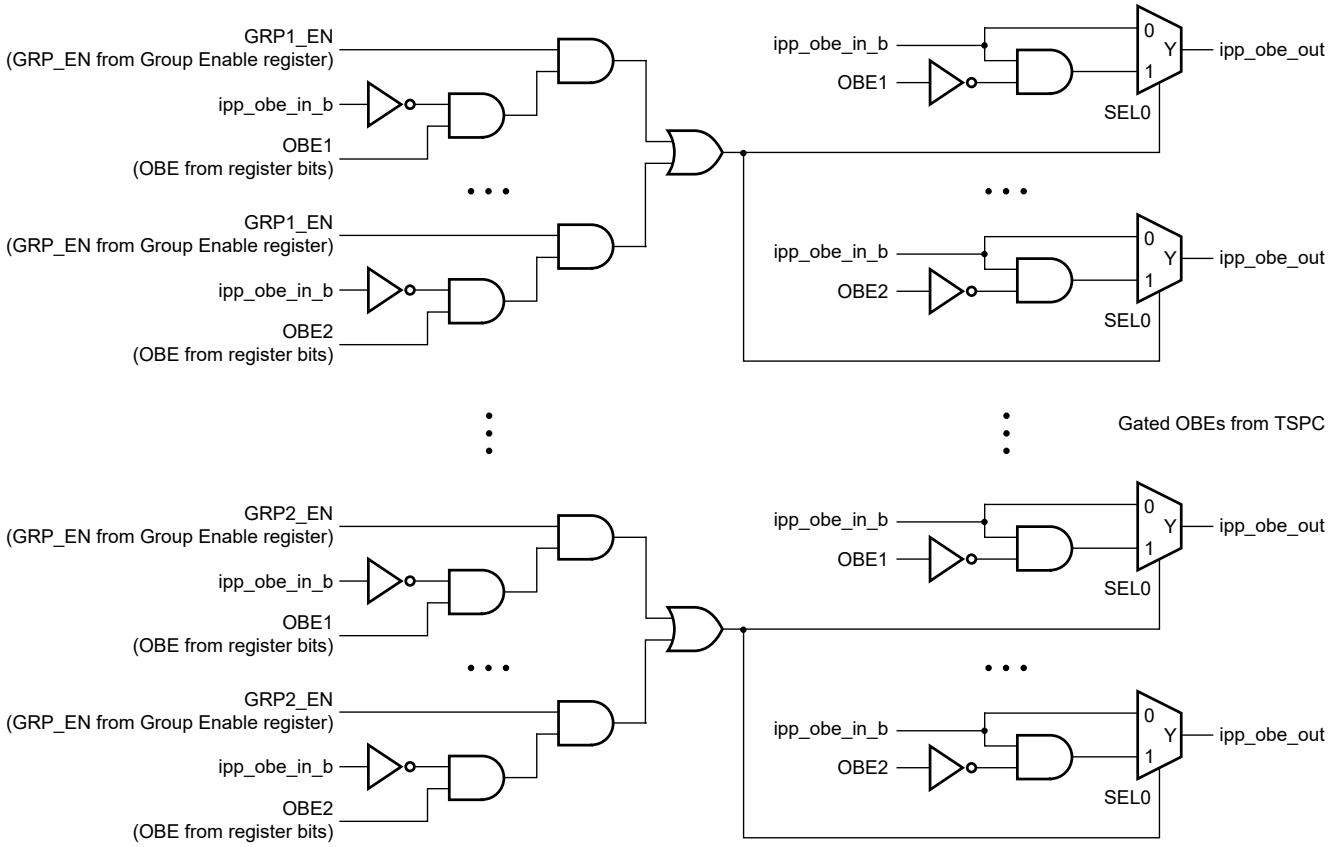



Figure 22. Logic block diagram

## 10.3.2 Operation

This section describes the TSPC_SPEC module's operations.

1. Reset all the registers.
2. Write 1 to the GRP*n*_OBE*x*[OBE*n*] of the OBEs that participate in grouping.
3. Program the corresponding group enable field participating in grouping in Group Enable (GRP_EN).

You must write 0 to the corresponding GRP_EN field when you configure the group *n* OBE fields. You must write 1 to the OBE*n* field in Group OBE (GRP1_OBE1 - GRP2_OBE1) and then write 1 to the corresponding GRP_EN field in Group Enable (GRP_EN).

## 10.3.3 Clocking

The TSPC module has two clocks.

| Clock |
|---|
| ipg_clk |
| ipg_clk_s |

## 10.3.4 Interrupts

This module has no interrupts.

## 10.4 External signals

The TSPC module has two external signals.

| Signal | IO | Description |
|---|---|---|
| ipp_obe_out | Output | Gated output buffer enable from TSPC |
| ipp_obe_in_b | Input | Output buffer enable from SIUL and input for TSPC |

## 10.5 Initialization

This module does not require initialization.

## 10.6 Application information

TSPC IP block provides the functionality of simultaneously transitioning two pads into high impedance, in order to experience a more robust recognition of touch sense.

## 10.7 TSPC register descriptions

### 10.7.1 TSPC memory map

TSPC base address: 402C_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Group Enable (GRP_EN) | 32 | RW | 0000_0000h |
| 50h | Group OBE (GRP1_OBE1) | 32 | RW | 0000_0000h |
| 54h | Group OBE (GRP1_OBE2) | 32 | RW | 0000_0000h |
| A0h | Group OBE (GRP2_OBE1) | 32 | RW | 0000_0000h |
| A4h | Group OBE (GRP2_OBE2) | 32 | RW | 0000_0000h |

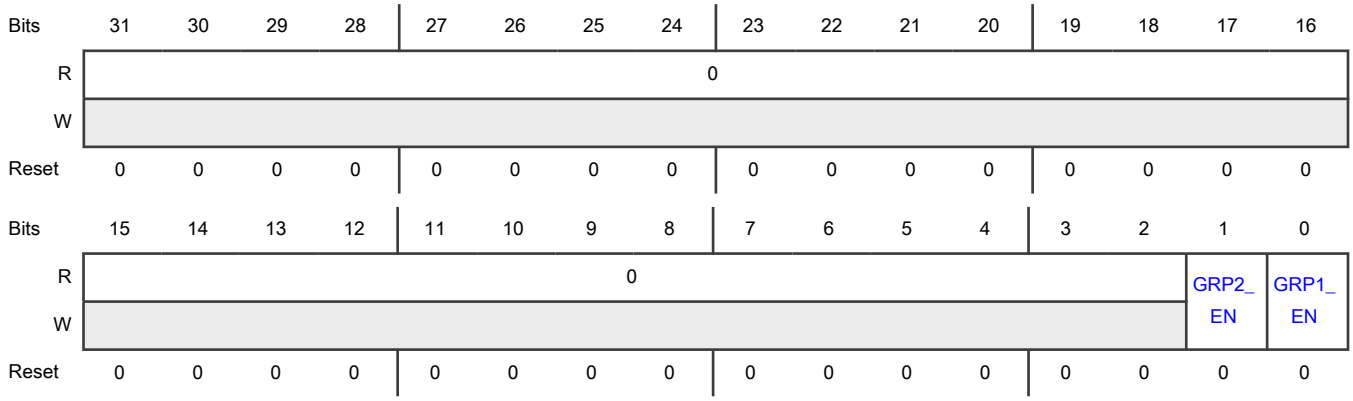
### 10.7.2 Group Enable (GRP_EN)

**Offset**

| Register | Offset |
|---|---|
| GRP_EN | 0h |

**Function**
Allows enabling of the group, the pads of which participate in simultaneous transition.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>GRP2_EN | Enable for GRP2_OBEn Register<br>Enables or disables the GRP2_OBE*n* register that participates in grouping.<br>0b - Disable<br>1b - Enable |
| 0<br>GRP1_EN | Enable for GRP1_OBEn Register<br>Enables or disables the GRP1_OBE*n* register that participates in grouping.<br>0b - Disable<br>1b - Enable |

## 10.7.3 Group OBE (GRP1_OBE1 - GRP2_OBE1)

### Offset

| Register | Offset |
|---|---|
| GRP1_OBE1 | 50h |
| GRP2_OBE1 | A0h |

### Function

Allows grouping of OBEs in the respective group. Each field corresponds to an OBE pad.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | OBE31 | OBE30 | OBE29 | OBE28 | OBE27 | OBE26 | OBE25 | OBE24 | OBE23 | OBE22 | OBE21 | OBE20 | OBE19 | OBE18 | OBE17 | OBE16 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | OBE15 | OBE14 | OBE13 | OBE12 | OBE11 | OBE10 | OBE9 | OBE8 | OBE7 | OBE6 | OBE5 | OBE4 | OBE3 | OBE2 | OBE1 | OBE0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>OBEn | Output Buffer Enable<br>Indicates if a particular OBE participates in grouping.<br>0b - OBE pad does not transition if any OBE in group transitions from high to low. The OBE pad value is unaltered at the output.<br>1b - OBE pad transitions from high to low if any OBE in group transitions from high to low. |

## 10.7.4 Group OBE (GRP1_OBE2)

### Offset

| Register | Offset |
|---|---|
| GRP1_OBE2 | 54h |

### Function

Allows grouping of OBEs in the respective group. Each field corresponds to an OBE pad.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | OBE45 | OBE44 | OBE43 | OBE42 | OBE41 | OBE40 | OBE39 | OBE38 | OBE37 | OBE36 | OBE35 | OBE34 | OBE33 | OBE32 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-14<br>— | Reserved |
| 13-0<br>OBEn | Output Buffer Enable<br>Indicates if a particular OBE participates in grouping.<br>0b - OBE pad does not transition if any OBE in group transitions from high to low. The OBE pad value is unaltered at the output.<br>1b - OBE pad transitions from high to low if any OBE in group transitions from high to low. |

## 10.7.5 Group OBE (GRP2_OBE2)

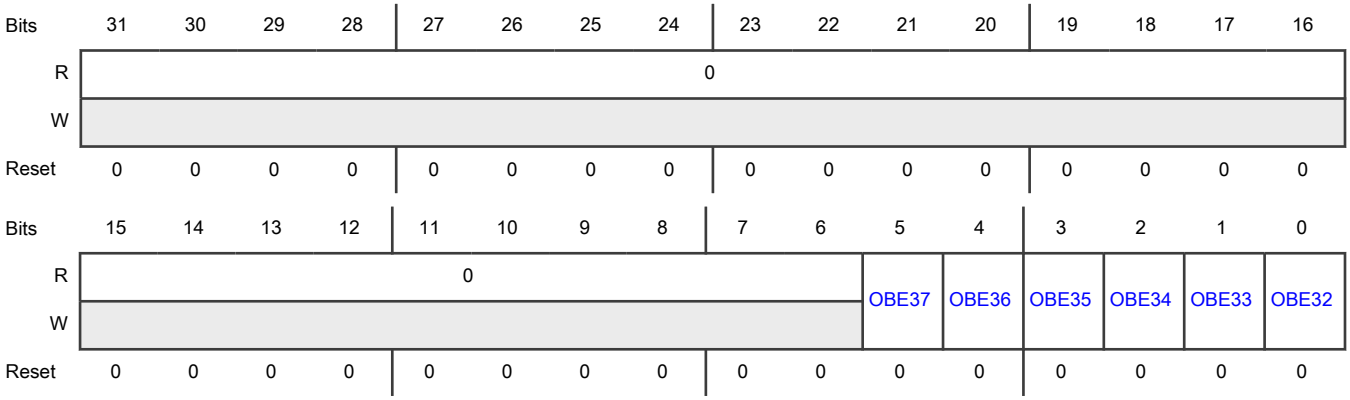
### Offset

| Register | Offset |
|---|---|
| GRP2_OBE2 | A4h |

### Function

Allows grouping of OBEs in the respective group. Each field corresponds to an OBE pad.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved |
| 5-0 | Output Buffer Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| OBEn | Indicates if a particular OBE participates in grouping.<br>0b - OBE pad does not transition if any OBE in group transitions from high to low. The OBE pad value is unaltered at the output.<br>1b - OBE pad transitions from high to low if any OBE in group transitions from high to low. |

## 10.8 Glossary

| | |
|---|---|
| **GPIO** | General purpose input and output |
| **OBE** | Output buffer enable |

# Chapter 11
# Crossbar Switch (AXBS)

## 11.1 Chip-specific AXBS information

### 11.1.1 AXBS instances and connectivity matrix

This chip has up to 5 instances of AXBS.

**Table 35. AXBS instances**

| Instance | MCXE31B | MCXE317 | MCXE315/MCXE316 |
|---|---|---|---|
| AXBS_0 | Yes | Yes | Yes |
| AXBS_1 | Yes | No | No |
| AXBS_2 | Yes | No | No |
| AXBS_3 | Yes | No | No |
| AXBS_4 | Yes | No | No |

**Table 36. AXBS connectivity matrix for MCXE31B**

| Instance | Master and slave assignments |
|---|---|
| AXBS_0 (main) | |

| Master port | Master module | | Slave port | Slave module |
|---|---|---|---|---|
| M0 | Cortex-M7_0 AHBM | | S0 | Flash memory port 0 |
| M1 | AXBS_2 S0 | | S1 | Flash memory port 1 |
| M2 | ELE_HSEB | | S2 | PRAM_0 |
| M3 | EMAC | | S3 | Cortex-M7 TCM |

AXBS_1 (peripheral)

| Master port | Master module | | Slave port | Slave module |
|---|---|---|---|---|
| M0 | Cortex-M7_0 AHBP | | S0 | AIPS_0 |
| M1 | AXBS_2 S1 | | S1 | AIPS_1 |
| M2 | ELE_HSEB | | S2 | AIPS_2 |

AXBS_2 (eDMA)

| Master port | Master module | | Slave port | Slave module |
|---|---|---|---|---|
| M0 | eDMA | | S0 | AXBS_0 M1 |
| | | | S1 | AXBS_1 M1 |

Table 36. AXBS connectivity matrix for MCXE31B (continued)

| Instance | Master and slave assignments | | | | |
|---|---|---|---|---|---|
| AXBS_3 (Cortex-M7 TCM) | | | | | |
| | **Master port** | **Master module** | | **Slave port** | **Slave module** |
| | M0 | AXBS_0 S3 | | S0 | Cortex-M7_0 TCM |
| AXBS_4 (HSE) | | | | | |
| | **Master port** | **Master module** | | **Slave port** | **Slave module** |
| | M0 | ELE_HSEB | | S0 | AXBS_0 M2 |

Table 37. AXBS connectivity matrix for MCXE315/MCXE316 and MCXE317

| Instance | Master and slave assignments | | | |
|---|---|---|---|---|
| AXBS_0 (main) | | | | |
| | **Master port** | **Master module** | **Slave port** | **Slave module** |
| | M0 | Cortex-M7_0 AHBM | S0 | Flash memory port 0 |
| | M1 | AXBS_2 S0 | S1 | Flash memory port 1 |
| | M2 | ELE_HSEB | S2 | PRAM_0 |
| | M3 | EMAC | S3 | Cortex-M7 TCM |
| | M4 | Reserved | S4 | AIPS_0 |
| | | | S5 | AIPS_1 |

## 11.2 Overview

This section provides information on the layout, configuration, and programming of the crossbar switch.

The crossbar switch connects bus initiators and bus targets using a crossbar switch structure. This structure allows all bus initiators to access different bus targets simultaneously, while providing arbitration among the bus initiators when they access the same target. A variety of bus arbitration methods and attributes may be programmed on a target-by-target basis.

### 11.2.1 Features

- Symmetric crossbar bus switch implementation
  - Allows concurrent access from different Initiators to different Targets
  - Target arbitration attributes configured on a Target-by-Target basis
- Single-clock 64-bit transfer
- Support for burst transfers of 64 bits of data
- Support for low-power park mode
- Initiator high-priority elevation
- 64-bit AHB crossbar bus switch compatible with ARM's AMBA Specification v2.0

## 11.3 Functional description

Information about general operation and arbitration are provided in this section.

### 11.3.1 General operation

When a Initiator accesses the crossbar switch, the access is immediately taken. If the targeted target port of the access is available, then the access is immediately presented on the target port. Single-clock or zero-wait-state accesses are possible through the crossbar. If the targeted target port of the access is busy or parked on a different Initiator port, the requesting Initiator sees wait states inserted until the targeted target port can service the Initiator's request. The latency in servicing the request depends on each Initiator's priority level and the responding target's access time.

Because the crossbar switch appears to be just another target to the Initiator device, the Initiator device does not know whether it owns the target port it is targeting. The Initiator waits while it does not have control of the target port it is targeting.

After the Initiator acquires control of the target port, it controls the port until it relinquishes the port by running an IDLE cycle or by targeting a different target port for its next access.

The Initiator can also lose control of the target port if another higher-priority Initiator makes a request to the target port. However, if the Initiator is running a fixed-length burst transfer, it retains control of the target port until that transfer completes.

The crossbar terminates all Initiator IDLE transfers, as opposed to allowing the termination to come from one of the target buses. Additionally, when no Initiator is requesting access to a target port, the crossbar drives IDLE transfers onto the target bus, even though a default Initiator may be granted access to the target port.

When a target bus is being idled by the crossbar, it can park the target port on the Initiator port indicated by CRS*n*[PARK]. This is done to save the initial clock of arbitration delay that otherwise would be seen if the same Initiator had to arbitrate to gain control of the target port. The target port can also be put into low-power park mode to save power, by using CRS*n*[PCTL].

### 11.3.2 Register coherency

The operation of the crossbar is affected as soon as a register is written. The values of the registers do not track with target-port-related Initiator accesses, but instead track only with target accesses.

### 11.3.3 Arbitration

The crossbar switch supports the following arbitration algorithms:

- Fixed priority
- Round-robin

The arbitration scheme is independently programmable for each target port.

#### 11.3.3.1 Fixed-priority operation

When operating in fixed-priority mode, each initiator is assigned a unique priority level in the priority registers (PRS*n*). If two initiators request access to the same target port, the initiator with the highest priority in the selected priority register gains control over the target port.

**NOTE**

In this arbitration mode, a higher-priority initiator can monopolize a target port, preventing access from any lower-priority initiator to the port.

When a initiator makes a request to a target port, the target port checks whether the new requesting initiator's priority level is higher than that of the initiator that currently has control over the target port, unless the target port is in a parked state. The target port performs an arbitration check at every clock edge to ensure that the initiator, if any, has control of the target port.

The following table describes possible scenarios based on the requesting initiator port.

Table 38. Methods of how the crossbar switch grants control of a target port to a initiator

| When | Then the crossbar switch grants control to the requesting initiator |
|---|---|
| Both of the following are true:<br>• The current initiator is not running a transfer.<br>• The new requesting initiator's priority level is higher than that of the current initiator. | At the next clock edge |
| Both of the following are true:<br>• The current initiator is running a fixed-length burst transfer or a locked transfer.<br>• The requesting initiator's priority level is higher than that of the current initiator. | At the end of the burst transfer or locked transfer |
| The requesting initiator's priority level is lower than the current initiator. | At the conclusion of one of the following cycles:<br>• An IDLE cycle<br>• A non-IDLE cycle to a location other than the current target port |

#### 11.3.3.2 Round-robin priority operation

When operating in round-robin mode, each initiator is assigned a relative priority based on the initiator port number. This relative priority is compared to the initiator port number (ID) of the last initiator to perform a transfer on the target bus. The highest priority requests the initiator owns the target bus at the next transfer boundary, accounting for locked and fixed-length burst transfers. Priority is based on how far ahead the ID of the requesting initiator is of the ID of the last initiator.

After a initiator is granted access to a target port, a initiator may perform as many transfers as desired to that port until another initiator requests the same target port. The next initiator in line is granted access to the target port at the next transfer boundary, or possibly on the next clock cycle, if the current initiator has no pending access request.

As an example of arbitration in round-robin mode, assume that the crossbar is implemented with initiator ports 0, 1, 4, and 5. If the last initiator of the target port was initiator 1, and initiators 0, 4, and 5 make simultaneous requests, they are serviced in this order: 4,5, and then 0.

Parking may continue to be used in a round-robin mode, but does not affect the round-robin pointer unless the parked initiator performs a transfer. Handoff occurs to the next initiator in line after one cycle of arbitration. If the target port is put into low-power park mode, the round-robin pointer is reset to point at initiator port 0, giving it the highest priority.

#### 11.3.3.3 Clocking

This module has no clocking considerations.

#### 11.3.3.4 Interrupts

This module has no interrupts.

#### 11.3.3.5 Priority assignment

Each initiator port must be assigned a unique 3-bit priority level. If an attempt is made to program multiple initiator ports with the same priority level within the priority registers (PRS*n*), the crossbar switch responds with a bus error and the registers are not updated.

## 11.4 External signals

This module has no external signals.

## 11.5 Initialization/application information

No initialization is required for the crossbar switch.

Hardware reset ensures that all the register bits used by the crossbar switch are properly initialized to a valid state. However, the following settings and priorities may be programmed to achieve the maximum system performance:

- During the configuration of the crossbar switch, all other initiators must be IDLE.
- To prevent reconfiguration of the crossbar switch, write 1 to CRS*n*[RO].

## 11.6 Memory map and register definition

Each target port of the crossbar switch contains configuration registers. Read- and write transfers require two bus clock cycles. The registers can be read from and written to only in supervisor mode. Additionally, these registers can be read from or written to only by 32-bit accesses.

A bus error response is returned if an unimplemented location is accessed within the crossbar switch.

The CRS*n* and PRS*n* registers can be programmed as read-only to prevent changes to their configuration. After being read-only protected, future writes to them terminate with a data storage error.

**NOTE**

This section shows the registers for all eight initiator and target ports. If a initiator or target is not used on this particular chip, then unexpected results occur when writing to its registers. See the chip configuration details for the exact initiator and target assignments for your chip.

All references to the crossbar switch registers are based on the physical port connections.

### 11.6.1 AXBS register descriptions

#### 11.6.1.1 AXBS memory map

AXBS_LITE base address: 4020_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Priority Target Registers (PRS0) | 32 | RW | 0000_3210h |
| 10h | Control Register (CRS0) | 32 | RW | 0002_0110h |
| 100h | Priority Target Registers (PRS1) | 32 | RW | 0000_3210h |
| 110h | Control Register (CRS1) | 32 | RW | 0002_0110h |
| 200h | Priority Target Registers (PRS2) | 32 | RW | 0000_3210h |
| 210h | Control Register (CRS2) | 32 | RW | 0002_0110h |
| 300h | Priority Target Registers (PRS3) | 32 | RW | 0000_3210h |
| 310h | Control Register (CRS3) | 32 | RW | 0002_0110h |
| 400h | Priority Target Registers (PRS4) | 32 | RW | 0000_3210h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 410h | Control Register (CRS4) | 32 | RW | 0002_0110h |
| 500h | Priority Target Registers (PRS5) | 32 | RW | 0000_3210h |
| 510h | Control Register (CRS5) | 32 | RW | 0002_0110h |
| 600h | Priority Target Registers (PRS6) | 32 | RW | 0000_3210h |
| 610h | Control Register (CRS6) | 32 | RW | 0002_0110h |

### 11.6.1.2 Priority Target Registers (PRS0 - PRS6)

#### Offset

| Register | Offset |
|---|---|
| PRS0 | 0h |
| PRS1 | 100h |
| PRS2 | 200h |
| PRS3 | 300h |
| PRS4 | 400h |
| PRS5 | 500h |
| PRS6 | 600h |

#### Function

The priority target registers(PRS*n*) set the priority of each initiator port on a per target port basis and reside in each target port. The priority register can be accessed only with 32-bit access. After the CRS*n*[RO] bit is set, the PRS*n* register can only be read; attempts to write to it have no effect on PRS*n* and result in a bus-error response to the initiator initiating the write.

Two available initiators must not be programmed with the same priority level. Attempts to program two or more initiators with the same priority level result in a bus-error response and the PRS*n* is not updated.

**NOTE**

Valid values for the M*n* priority fields depend on which initiators are available on the chip. This information can be found in the chip-specific information for the crossbar.

- If the chip contains fewer than three initiators, only one bit is valid.
- If the chip contains fewer than five initiators, only two bits are valid.
- If five or more initiators are present, all three bits of the priority field are used.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | M3 | | | 0 | M2 | | | 0 | M1 | | | 0 | M0 | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-28<br>— | Reserved |
| 27<br>— | Reserved |
| 26-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-20<br>— | Reserved |
| 19<br>— | Reserved |
| 18-16<br>— | Reserved |
| 15<br>— | Reserved |
| 14-12<br>M3 | Initiator 3 Priority<br>This field sets the arbitration priority for this port on the associated target port. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 000b - This initiator has level 1 or highest priority when accessing the target port.<br>001b - This initiator has level 2 priority when accessing the target port.<br>010b - This initiator has level 3 priority when accessing the target port.<br>011b - This initiator has level 4 priority when accessing the target port.<br>100b - This initiator has level 5 priority when accessing the target port.<br>101b - This initiator has level 6 priority when accessing the target port.<br>110b - This initiator has level 7 priority when accessing the target port.<br>111b - This initiator has level 8the or lowest priority when accessing the target port. |
| 11<br>— | Reserved |
| 10-8<br>M2 | Initiator 2 Priority<br>This field sets the arbitration priority for this port on the associated target port.<br>000b - This initiator has level 1 or highest priority when accessing the target port.<br>001b - This initiator has level 2 priority when accessing the target port.<br>010b - This initiator has level 3 priority when accessing the target port.<br>011b - This initiator has level 4 priority when accessing the target port.<br>100b - This initiator has level 5 priority when accessing the target port.<br>101b - This initiator has level 6 priority when accessing the target port.<br>110b - This initiator has level 7 priority when accessing the target port.<br>111b - This initiator has level 8the or lowest priority when accessing the target port. |
| 7<br>— | Reserved |
| 6-4<br>M1 | Initiator 1 Priority<br>This field sets the arbitration priority for this port on the associated target port.<br>000b - This initiator has level 1 or highest priority when accessing the target port.<br>001b - This initiator has level 2 priority when accessing the target port.<br>010b - This initiator has level 3 priority when accessing the target port.<br>011b - This initiator has level 4 priority when accessing the target port.<br>100b - This initiator has level 5 priority when accessing the target port.<br>101b - This initiator has level 6 priority when accessing the target port.<br>110b - This initiator has level 7 priority when accessing the target port.<br>111b - This initiator has level 8 or lowest priority when accessing the target port. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 3<br>— | Reserved |
| 2-0<br>M0 | Initiator 0 Priority<br>This field sets the arbitration priority for this port on the associated target port.<br>000b - This initiator has level 1 or highest priority when accessing the target port.<br>001b - This initiator has level 2 priority when accessing the target port.<br>010b - This initiator has level 3 priority when accessing the target port.<br>011b - This initiator has level 4 priority when accessing the target port.<br>100b - This initiator has level 5 priority when accessing the target port.<br>101b - This initiator has level 6 priority when accessing the target port.<br>110b - This initiator has level 7 priority when accessing the target port.<br>111b - This initiator has level 8 or the lowest priority when accessing the target port. |

### 11.6.1.3 Control Register (CRS0 - CRS6)

#### Offset

| Register | Offset |
|---|---|
| CRS0 | 10h |
| CRS1 | 110h |
| CRS2 | 210h |
| CRS3 | 310h |
| CRS4 | 410h |
| CRS5 | 510h |
| CRS6 | 610h |

#### Function

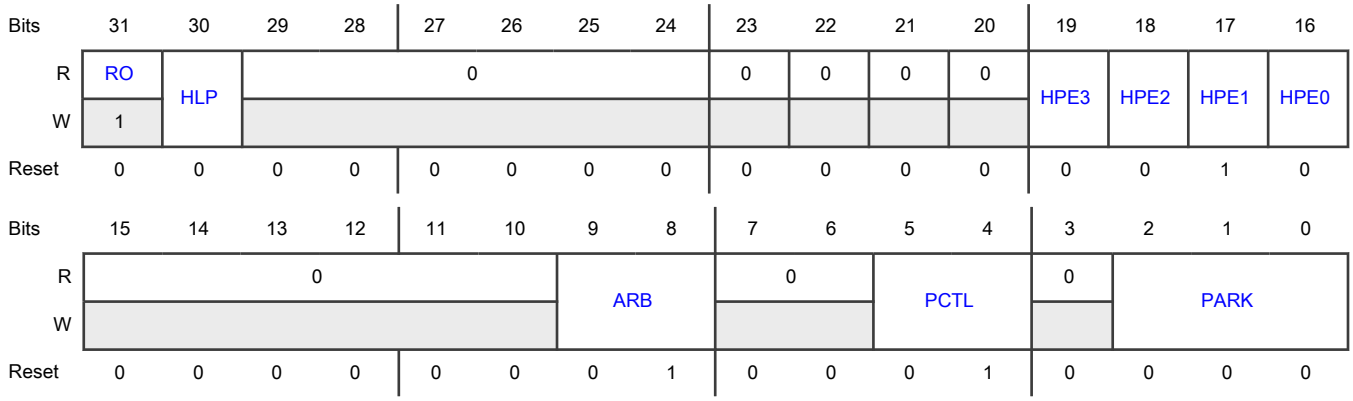
These registers control several features of each target port and must be accessed using 32-bit accesses. After CRS*n*[RO] is set, the PRS*n* can only be read; attempts to write to it have no effect and results in an error response.

**NOTE**

See the chip-specific crossbar information for the reset value of this register.

Not all HPE *n* fields may be active. See the chip-specific crossbar information for which initiators support initiator high-priority elevation. Setting a field corresponding to a initiator that does not support initiator, high-priority elevation has no effect.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>RO | Read Only<br>Forces the PRS*n* and CRS*n* registers to be read-only. After being set, only a hardware reset clears this field.<br>0b - The CRS*n* and PRS*n* registers are writeable<br>1b - The CRS*n* and PRS*n* registers are read-only and cannot be written (attempted writes have no effect on the registers and result in a bus error response). |
| 30<br>HLP | Halt Low Priority<br>This field sets the initial arbitration priority for low power-mode requests . Setting this bit will not affect the ,request for low power-mode from attaining the highest priority after it has control of the target ports.<br>0b - The low-power mode request has the highest priority for arbitration on this target port.<br>1b - The low-power mode request has the lowest initial priority for arbitration on this target port. |
| 29-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22<br>— | Reserved |
| 21<br>— | Reserved |
| 20<br>— | Reserved |
| 19 | High Priority Elevation 3 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| HPE3 | This field enables initiator high-priority elevation for initiator 3 on this target port. If enabled, the initiator can elevate its priority to the highest.<br>0b - Initiator high-priority elevation for initiator 3 is disabled on this target port.<br>1b - Initiator high-priority elevation for initiator 3 is enabled on this target port. |
| 18<br>HPE2 | High Priority Elevation 2<br>This field enables initiator high-priority elevation for initiator 2 on this target port. If enabled, the initiator can elevate its priority to the highest.<br>0b - Initiator high-priority elevation for initiator 2 is disabled on this target port.<br>1b - Initiator high-priority elevation for initiator 2 is enabled on this target port. |
| 17<br>HPE1 | High Priority Elevation 1<br>This field enables initiator high-priority elevation for initiator 1 on this target port. If enabled, the initiator can elevate its priority to the highest.<br>0b - Initiator high-priority elevation for initiator 1 is disabled on this target port.<br>1b - Initiator high-priority elevation for initiator 1 is enabled on this target port. |
| 16<br>HPE0 | High Priority Elevation 0<br>This field enables initiator high-priority elevation for initiator 0 on this target port. If enabled, the initiator can elevate its priority to the highest.<br>0b - Initiator high-priority elevation for initiator 0 is disabled on this target port.<br>1b - Initiator high-priority elevation for initiator 0 is enabled on this target port. |
| 15-10<br>— | Reserved |
| 9-8<br>ARB | Arbitration Mode<br>This field selects the arbitration policy for the target port.<br>00b - Fixed priority<br>01b - Round-robin (rotating) priority<br>10b - Reserved<br>11b - Reserved |
| 7-6<br>— | Reserved |
| 5-4<br>PCTL | Parking Control<br>This field determines the target port's parking control. The low-power park feature results in an overall power savings if the target port is not saturated; however, this forces an extra latency clock when any initiator tries to access the target port when not in use because it is not parked on any initiator. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 00b - When no initiator makes a request, the arbiter parks the target port on the initiator port defined by the PARK bit field.<br>01b - When no initiator makes a request, the arbiter parks the target port on the last initiator to be in control of the target port.<br>10b - Low-power park. When no initiator makes a request, the target port is not parked on a initiator and the arbiter drives all outputs to a constant safe state.<br>11b - Reserved |
| 3<br>— | Reserved |
| 2-0<br>PARK | Park<br>This field determines which initiator port the current target port parks on when no initiators are actively making requests and the PCTL bits are cleared.<br>**NOTE**<br>Select only initiator ports that are present on the chip. Otherwise, undefined behavior might occur.<br>000b - Park on initiator port M0<br>001b - Park on initiator port M1<br>010b - Park on initiator port M2<br>011b - Park on initiator port M3<br>100b - Park on initiator port M4<br>101b - Park on initiator port M5<br>110b - Park on initiator port M6<br>111b - Park on initiator port M7 |

## 11.7 Glossary

**AMBA** Advanced Microcontroller Bus Architecture

**IDLE** A type of transfer that a initiator uses when it does not want to perform a data transfer

**ID** Initiator port number

# Chapter 12
# Peripheral Bridge (AIPS_Lite)

## 12.1 Chip-specific AIPS_Lite information

### 12.1.1 AIPS_Lite instances

The following table summarizes AIPS_Lite instances on MCXE31 product series to support simultaneous data transfers to different peripherals. See the memory map file attached to this document to find information related to these peripherals that are associated with AIPS_Lite instances.

**Table 39. AIPS_Lite instances**

| Instance | MCXE31B | MCXE317 | MCXE315/MCXE316 |
|---|---|---|---|
| AIPS_0 | Yes | Yes | Yes |
| AIPS_1 | Yes | Yes | Yes |
| AIPS_2 | Yes | No | No |

## 12.2 Overview

AIPS_Lite converts the crossbar switch interface to an interface that can access most of the slave peripherals on this chip.

This peripheral bridge occupies 64 MB of the address space, which is divided into peripheral slots of 16 KB each. All the peripherals may not be used. See the memory map chapter for details on slot assignments. The bridge includes separate clock enable inputs for each of the slots to accommodate slower peripherals.

### 12.2.1 Features

Following are the key features of the peripheral bridge:

- Supports peripheral slots with 8-, 16-, and 32-bit datapath width
- Supports a pair of 32-bit transactions for selected 64-bit memory accesses

### 12.2.2 General operation

The slave devices connected to the peripheral bridge are modules that contain a programming model of control and status registers. The system masters read and write these registers through the peripheral bridge.

The register maps of the peripherals are located on 16 KB boundaries. Each peripheral is allocated one or more 16-KB block(s) of the memory map.

## 12.3 Functional description

The peripheral bridge functions as a bus protocol translator between the crossbar switch and the slave peripheral bus. Support is provided for generating a pair of 32-bit slave accesses when performing certain 64-bit peripheral accesses.

The peripheral bridge manages all transactions for the attached slave devices and generates select signals for modules on the peripheral bus by decoding accesses within the attached address space.

### 12.3.1 Access support

All combinations of access size and peripheral data port width are supported. An access that is larger than the target peripheral's data width is decomposed to multiple, smaller accesses. Bus decomposition is terminated by a transfer error caused by an access to an empty register area.

### 12.3.2 Clocking

This module has no clocking considerations.

### 12.3.3 Interrupts

This module has no interrupts.

## 12.4 External signals

This module has no external signals.

## 12.5 Memory map and register definition

The AIPS module(s) on this chip do(es) not contain any user-programmable registers.

# Chapter 13
# Direct Memory Access Multiplexer (DMAMUX)

## 13.1 Chip-specific DMAMUX information

### 13.1.1 DMAMUX instances

This chip has two instances of DMAMUX: DMAMUX_0 and DMAMUX_1. See the DMAMUX map file attached to this document to find information related to the slot number for each DMA trigger source.

For MCXE31B, DMAMUX_0 channels 0-15 and DMAMUX_1 channels 0-15 are mapped to eDMA Transfer Control Descriptor (TCD) 0-15 and eDMA Transfer Control Descriptor (TCD) 16-31, respectively. For the remaining chips, DMAMUX_0 channels 0-5 and DMAMUX_1 channels 0-5 are mapped to eDMA Transfer Control Descriptor (TCD) 0-5 and eDMA Transfer Control Descriptor(TCD) 6-11, respectively. Therefore, it is not recommended to program DMAMUX_0 channels 6-15 and DMAMUX_1 channels 6-15. When programmed, any access terminates with either an error response for channels 8-15 or without an error response for channels 6-7.

### 13.1.2 Periodic DMA triggering

PIT generates periodic trigger events to DMAMUX as shown in this table.

**Table 40. PIT channel assignment for periodic DMA triggering through DMAMUX**

| DMAMUX0/1 ports | PIT channel |
|---|---|
| pit_dma_trigger[0] | PIT0 channel 0 |
| pit_dma_trigger[1] | PIT0 channel 1 |
| pit_dma_trigger[2] | PIT0 channel 2 |
| pit_dma_trigger[3] | PIT0 channel 3 |

## 13.2 Introduction

### 13.2.1 Overview

DMAMUX routes a DMA source, called slot, to any of the 16 DMA channels. See the chip-specific information for details.

### 13.2.2 Block diagram

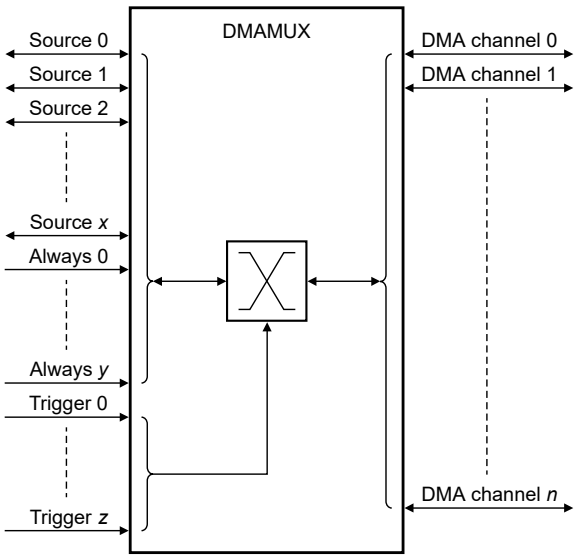



Figure 23. Block diagram

### 13.2.3 Features

- Ability to route up to 61 peripheral slots and up to 2 always-on slots to 16 channels
- Supports 16 independently selectable DMA channel routers (the first 4 channels provide an additional trigger functionality)
- Allows assignment of each channel router to one of the possible peripheral DMA slots or always-on slots

### 13.2.4 Modes of operation

The following table describes the functional operating modes of DMAMUX.

Table 41. Modes of operation

| Mode | Description |
|---|---|
| Disabled | In this mode, the DMA channel is disabled. Because disabling and enabling of DMA channels is done primarily via the DMA configuration registers, this mode is used mainly as the reset state for a DMA channel in the DMA channel MUX. You can also use this mode to temporarily suspend a DMA channel while system reconfiguration takes place, for example, by changing the period of a DMA trigger. |
| Normal | In this mode, a DMA source is routed directly to the specified DMA channel. The operation of DMAMUX in this mode is completely transparent to the system. |
| Periodic trigger | In this mode, a DMA source can only request a DMA transfer, such as, when a transmit buffer becomes empty or a receive buffer becomes full, periodically.<br>You configure the period using an external periodic interrupt timer (PIT). This mode is available only for channels 0–3. See the chip-specific section for more details. |

## 13.3 Functional description

The primary purpose of DMAMUX is to provide flexibility in the system's use of the available DMA channels. Configuration of DMAMUX is intended to be a static procedure performed during the execution of the system boot code. However, if you follow the procedure outlined in Enabling and configuring sources, you can change the configuration of DMAMUX during the normal operation of the system. All DMA channels must be inactive before and during a configuration change.

Functionally, DMAMUX channels can be divided into two classes:

- Channels that implement the normal routing functionality plus the periodic triggering capability
- Channels that implement only the normal routing functionality

### 13.3.1 DMA channels with periodic triggering capability

Besides the normal routing functionality, the first four channels of DMAMUX provide a special periodic triggering capability that you can use to provide an automatic mechanism to transmit bytes, frames, or packets at fixed intervals without the need for processor intervention (see Figure 24).

An external periodic interrupt timer (PIT) generates the trigger; for example, by configuring the external periodic timer, you configure the periodic triggering interval.

**NOTE**

Because of the dynamic nature of the system (owing to DMA channel priorities, bus arbitration, interrupt service routine lengths, and so on), the number of clock cycles between a trigger and the actual DMA transfer cannot be guaranteed.

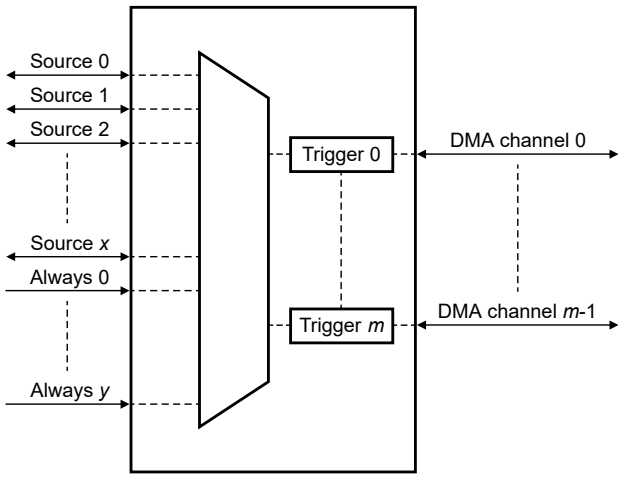



**Figure 24. DMAMUX triggered channels**

The DMA channel triggering capability allows the system to schedule regular DMA transfers, usually on the transmit side of certain peripherals, without the intervention of the processor. This trigger works by gating the request from the peripheral to DMA until a trigger event is observed (see Figure 25).

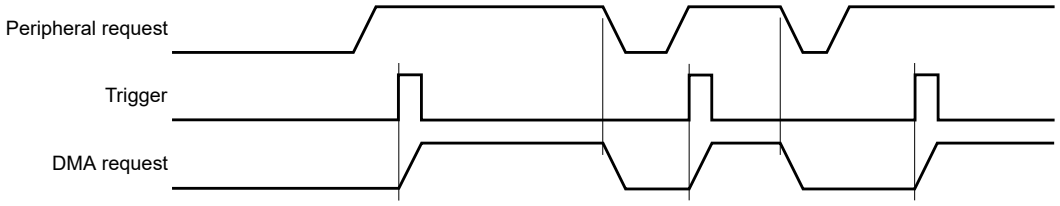



**Figure 25. DMAMUX channel triggering: normal operation**

After the DMA request is serviced, the peripheral negates its request, effectively resetting the gating mechanism until the peripheral reasserts its request and the next trigger event is seen. This means that if a trigger is seen, but the peripheral is not requesting a transfer, then that trigger is ignored (see Figure 26).

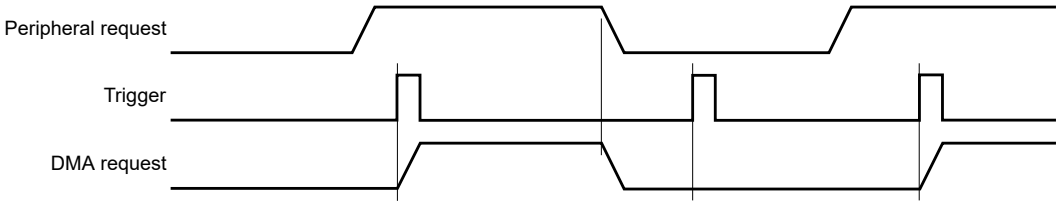



**Figure 26. DMAMUX channel triggering: ignored trigger**

You can use this triggering capability with any peripheral that supports DMA transfers, and is most useful in the following situations:

- Periodic polling of external devices on a particular bus: as an example, the transmit side of an SPI is assigned to a DMA channel with a trigger, as described in the aforementioned text. After periodic polling is set up, SPI requests DMA transfers, presumably from memory, as long as its transmit buffer is empty. By using a trigger on this channel, the SPI transfers can be automatically performed every 5 µs (as an example). On the receive side of SPI, you can configure SPI and DMA to transfer the received data into memory, effectively implementing a method to periodically read data from external devices and transferring the results into memory without processor intervention.
- Using the GPIO ports to drive or sample waveforms: by configuring DMA to transfer data to one or more GPIO ports, it is possible to create complex waveforms using tabular data stored in on-chip memory. Conversely, using DMA to periodically transfer data from one or more GPIO ports, it is possible to sample complex waveforms and store the results in a tabular form in on-chip memory.

### 13.3.2 DMA channels with no triggering capability

The other channels of DMAMUX provide the normal routing functionality as described in Modes of operation.

### 13.3.3 Always-enabled DMA sources

In addition to the peripherals that you can use as DMA sources, 2 additional DMA sources are always enabled. Unlike the peripheral DMA sources, where the peripheral controls the flow of data during DMA transfers, the sources that are always enabled provide no such "throttling" of data transfers. These sources are most useful in the following cases:

- Performing DMA transfers to and from GPIO: moving data to and from one or more GPIO pins, either unthrottled (that is, as fast as possible), or periodically (using the DMA triggering capability)
- Performing DMA transfers from memory to memory: moving data from memory to memory, typically as fast as possible, sometimes with software activation
- Performing DMA transfers from memory to the external bus, or vice-versa: similar to memory-to-memory transfers, this is typically done as quickly as possible
- Any DMA transfer that software must explicitly start or activate

In cases where software must initiate the start of a DMA transfer, an always-enabled DMA source can be used to provide maximum flexibility. When activating a DMA channel via software, subsequent executions of the minor loop require that a new start event be sent. This can either be a new software activation or a transfer request from the DMA channel MUX. The options for doing this are as follows.

- Transfer all data in a single minor loop: by configuring DMA to transfer all of the data in a single minor loop (that is, major loop counter = 1), no reactivation of the channel is necessary. The disadvantage of this option is reduced granularity in determining the load that the DMA transfer imposes on the system. For this option, the DMA channel must be disabled in the DMA channel MUX.
- Use explicit software reactivation: using this option, DMA is configured to transfer the data using both minor and major loops, but the processor is required to reactivate the channel by writing to the DMA registers after every minor loop. For this option, the DMA channel must be disabled in the DMA channel MUX.
- Use an always-enabled DMA source: using this option, DMA is configured to transfer the data using both minor and major loops, and the DMA channel MUX does the channel reactivation. For this option, you must enable the DMA channel and point it to an "always-enabled" source. Channel reactivation can be continuous (DMA triggering is disabled) or can use the DMA triggering capability. In this manner, it is possible to execute periodic transfers of data packets from one source to another, without processor intervention.

## 13.4 Initialization and application information

This section provides instructions for initializing the DMA channel MUX.

### 13.4.1 Reset

The reset state of each field is shown in Memory map and register definition. In summary, after reset, all channels are disabled and you must enable them explicitly before use.

### 13.4.2 Enabling and configuring sources

#### 13.4.2.1 Enable a source with periodic triggering

Perform the following procedure to enable a source with periodic triggering:

1. Determine the DMA channel with which the source is associated. Only the first 4 DMA channels have periodic triggering capability.
2. Write 0 to CHCFG*n*[ENBL] and CHCFG*n*[TRIG] of the DMA channel.
3. Ensure that the DMA channel is properly configured in DMA. You can enable the DMA channel at this point.
4. Configure the corresponding timer.
5. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that CHCFG*n*[ENBL] and CHCFG*n*[TRIG] are 1.

The following is an example. See the chip-specific information for the number of DMA channels on this chip that have triggering capability.

Example: To configure source 5 transmit for use with DMA channel 1, with periodic triggering capability:

1. Write 0h to CHCFG1.
2. Configure channel 1 in DMA, including enabling the channel.
3. Configure a timer for the desired trigger interval.
4. Write C5h to CHCFG1.

The following code example illustrates steps 1 and 4 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);

In File main.c:
#include "registers.h"
:
:
*CHCFG1 = 0x00;                              /* Clear all the fields of CHCFG1 register */
*CHCFG1 = 0xC5;                              /* ENBL = 1  DMA Channel is enabled */
                                             /* TRIG = 1  Triggering is enabled */
                                             /* SOURCE = 5  DMA Source 5 is selected */
```

**Code Listing 1. Configuring source 5 with DMA channel 1 (with periodic triggering)**

#### 13.4.2.2 Enable a source without periodic triggering

Perform the following procedure to enable a source without periodic triggering:

1. Determine the DMA channel with which the source is associated. Only the first 4 DMA channels have periodic triggering capability.
2. Write 0 to CHCFG*n*[ENBL] and CHCFG*n*[TRIG] of the DMA channel.
3. Ensure that the DMA channel is properly configured in DMA. You can enable the DMA channel at this point.
4. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that CHCFG*n*[ENBL] is 1 while CHCFG*n*[TRIG] is 0.

The following is an example. See the chip configuration details for the number of DMA channels on this chip that have triggering capability.

Example: To configure source 5 transmit for use with DMA channel 1, with no periodic triggering capability:

1. Write 0h to CHCFG1.
2. Configure channel 1 in DMA, including enabling the channel.
3. Write 85h to CHCFG1.

The following code example illustrates steps 1 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);

In File main.c:
#include "registers.h"
:
:
*CHCFG1 = 0x00;                              /* Clear all the fields of CHCFG1 register */
*CHCFG1 = 0x85;                              /* ENBL = 1  DMA Channel is enabled */
                                             /* TRIG = 0  Triggering is disabled */
                                             /* SOURCE = 5  DMA Source 5 is selected */
```

**Code Listing 2. Configuring source 5 with DMA channel 1 (without periodic triggering)**

#### 13.4.2.3 Disable a source

You can disable a particular DMA source by not writing the corresponding source value into any of the CHCFG registers. Additionally, some module-specific configurations may be necessary. See the "Enhanced Direct Memory Access (eDMA)" chapter for details.

To switch the source of a DMA channel:

1. Disable the DMA channel in the DMA and reconfigure the channel for the new source.
2. Write 0 to CHCFG*n*[ENBL] and CHCFG*n*[TRIG] of the DMA channel.
3. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that CHCFG*n*[ENBL] and CHCFG*n*[TRIG] are 1.

Example: To switch DMA channel 8 from source 5 transmit to source 7 transmit:

1. In the DMA configuration registers, disable DMA channel 8 and reconfigure it to handle the transfers to peripheral slot 7. This example assumes channel 8 does not have the triggering capability.
2. Write 0h to CHCFG8.
3. Write 87h to CHCFG8. (In this example, writing 1 to CHCFG[TRIG] has no effect because of the assumption that channel 8 does not support the periodic triggering functionality.)

The following code example illustrates steps 2 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes char is 8-bits */
volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);
volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);
volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);
volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);
volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);
volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);

In File main.c:
#include "registers.h"
:
:
*CHCFG8 = 0x00;                              /* Clear all the fields of CHCFG1 register */
*CHCFG8 = 0x87;                              /* ENBL = 1  DMA Channel is enabled */
                                             /* TRIG = 0  Triggering is disabled */
                                             /* SOURCE = 7  DMA Source 7 is selected */
```

**Code Listing 3. Switching DMA channel 8 from source 5 transmit to source 7 transmit**

## 13.5 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in DMAMUX.

### 13.5.1 DMAMUX register descriptions

#### 13.5.1.1 DMAMUX memory map

DMAMUX_0 base address: 4028_0000h

DMAMUX_1 base address: 4028_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Channel Configuration (CHCFG3) | 8 | RW | 00h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1h | Channel Configuration (CHCFG2) | 8 | RW | 00h |
| 2h | Channel Configuration (CHCFG1) | 8 | RW | 00h |
| 3h | Channel Configuration (CHCFG0) | 8 | RW | 00h |
| 4h | Channel Configuration (CHCFG7) | 8 | RW | 00h |
| 5h | Channel Configuration (CHCFG6) | 8 | RW | 00h |
| 6h | Channel Configuration (CHCFG5) | 8 | RW | 00h |
| 7h | Channel Configuration (CHCFG4) | 8 | RW | 00h |
| 8h | Channel Configuration (CHCFG11) | 8 | RW | 00h |
| 9h | Channel Configuration (CHCFG10) | 8 | RW | 00h |
| Ah | Channel Configuration (CHCFG9) | 8 | RW | 00h |
| Bh | Channel Configuration (CHCFG8) | 8 | RW | 00h |
| Ch | Channel Configuration (CHCFG15) | 8 | RW | 00h |
| Dh | Channel Configuration (CHCFG14) | 8 | RW | 00h |
| Eh | Channel Configuration (CHCFG13) | 8 | RW | 00h |
| Fh | Channel Configuration (CHCFG12) | 8 | RW | 00h |

### 13.5.1.2 Channel Configuration (CHCFG0 - CHCFG15)

#### Offset

For n = 0 to 15:

| Register | Offset |
|---|---|
| CHCFGn | 0h + (n + 3 - 2 × (n mod 4)) |

#### Function

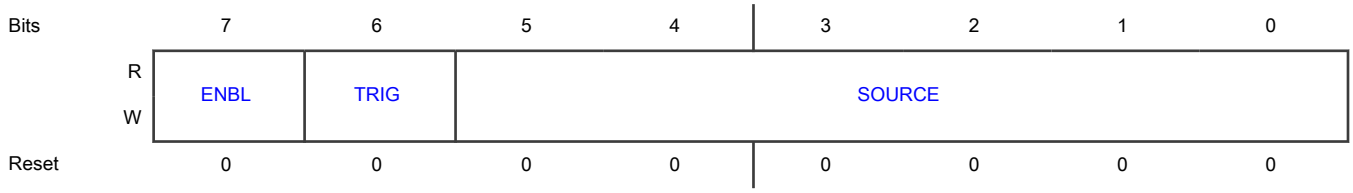
Provides you with the configuration to enable or disable, and to associate each of the DMA channels with one of the DMA slots (peripheral slots or always-on slots) in the system.

**NOTE**

Writing to multiple CHCFG registers with the same source value results in unpredictable behavior. This is true, even if a channel is disabled (CHCFG*n*[ENBL] == 0).

Before changing the trigger or source settings, you must disable a DMA channel using CHCFG*n*[ENBL].

### Diagram




### Fields

| Field | Function |
|---|---|
| 7<br>ENBL | DMA Channel Enable<br>Enables the DMA channel.<br>The condition when the DMA channel is disabled (ENBL = 0) is called Disabled mode. This mode is primarily used during the configuration of DMAMUX. DMA has separate channel enables or disables, which you must use to disable or reconfigure a DMA channel.<br>0b - Disable<br>1b - Enable |
| 6<br>TRIG | DMA Channel Trigger Enable<br>Enables the periodic triggering capability for the triggered DMA channel.<br>If triggering is disabled and ENBL = 1, the DMA channel routes the specified source to the DMA channel (Normal mode).<br>If triggering is enabled and ENBL = 1, then DMAMUX is in Periodic Trigger mode.<br>0b - Disable<br>1b - Enable |
| 5-0<br>SOURCE | DMA Channel Source (Slot)<br>Specifies which DMA source, if any, is routed to a particular DMA channel. See the chip-specific DMAMUX information for details about peripherals and their slot numbers. |

## 13.6 Glossary

| | |
|---|---|
| **Slot** | DMA source that either reflects a peripheral source or an always-enabled source |
| **Periodic trigger** | Trigger that an external source generates at periodic intervals for DMA transfer |

# Chapter 14
# Enhanced Direct Memory Access (eDMA)

## 14.1 Chip-specific eDMA information

### 14.1.1 DMA channels

See the following table for number of DMA channels across MCXE31 product series.

**Table 42. DMA channels**

| Chip | No. of DMA channels |
|---|---|
| MCXE31B | 32 |
| MCXE315/MCXE316/MCXE317 | 12 |

**NOTE**

TCD_CH[12-31] are not available in MCXE317, so the registers corresponding to these channels are not available in MCXE317 variant. See the memory map file attached to this document for details.

## 14.2 Introduction

The enhanced direct memory access (eDMA) controller is capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes:

- A DMA engine that performs:
  - Source address and destination address calculations
  - Data-movement operations
- Local memory containing transfer control descriptors for each of the 32 channels

### 14.2.1 eDMA system block diagram

Figure 27 illustrates the components of the eDMA system, including the eDMA module (engine).

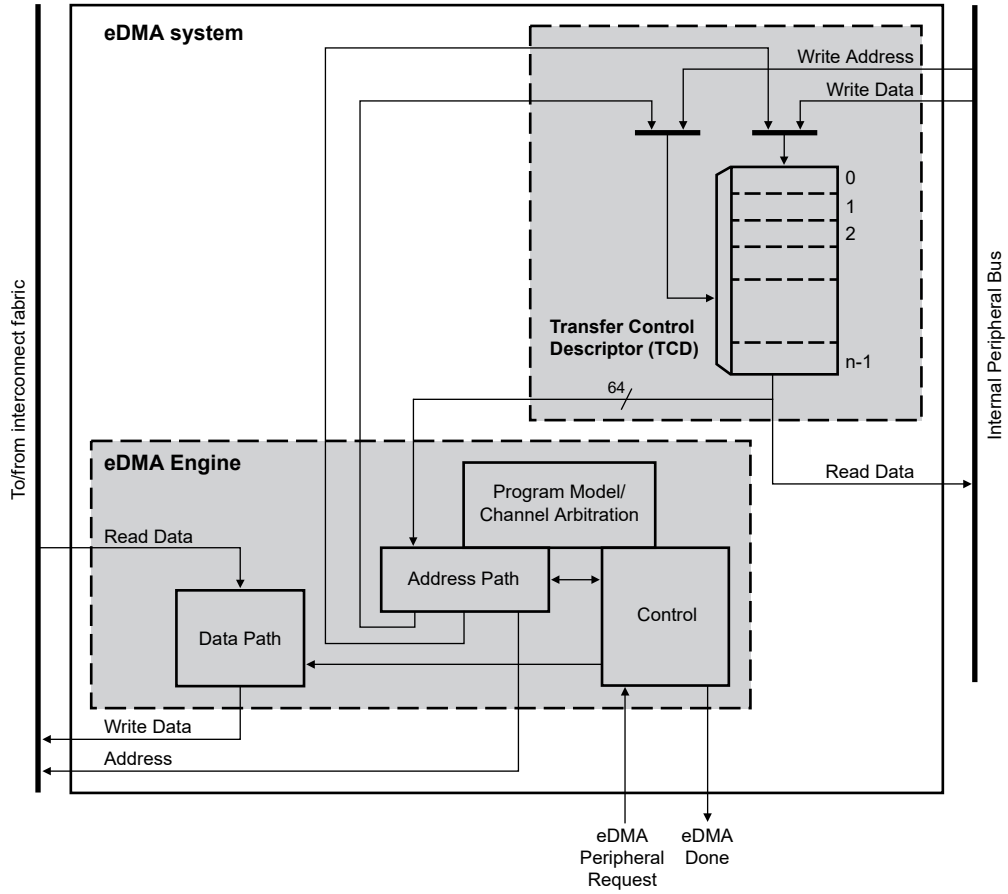



Figure 27. eDMA system block diagram

## 14.2.2 Block parts

The eDMA module is partitioned into two major modules: the eDMA engine and the transfer control descriptor local memory.

The eDMA engine is further partitioned into four submodules:

Table 43. eDMA engine submodules

| Submodule | Function |
|---|---|
| Address path | This block:<br>• Implements a primary channel and secondary (preempt) channel<br>• Manages all master bus-address calculations<br>All the channels provide the same functionality. This structure allows data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel activation is asserted while the primary channel is active.<br>After a channel is activated, it runs until the minor loop is completed, unless preempted by a higher priority channel. This provides a mechanism (enabled by CH*n*_PRI[ECP]) where a large data transfer can be preempted to minimize the time another channel is blocked from execution. |

*Table continues on the next page...*

Table 43. eDMA engine submodules (continued)

| Submodule | Function |
|---|---|
| | When any channel is selected to execute, the contents of its TCD are read from local memory and loaded into the address path channel x registers for a normal start and into channel y registers for a preemption start. After the minor loop completes execution, the address path hardware writes the new values for the TCD*n*_{SADDR, DADDR, CITER} back to local memory. If the major iteration count is exhausted, additional processing is performed, including the final address pointer updates, reloading the TCD*n*_CITER field, and a possible fetch of a new TCD*n* from memory as part of a scatter/gather operation. See Dynamic scatter/gather for more details. |
| Data path | This block implements the bus master read/write data path. It includes a data buffer and the necessary multiplex logic to support any required data alignment. The internal read data bus is the primary input, and the internal write data bus is the primary output.<br>The address and data path modules directly support the 2-stage pipelined internal bus. The address path module represents the first stage of the bus pipeline (address phase), and the data path module implements the second stage of the pipeline (data phase). |
| Program model/channel arbitration | This block implements the first section of the eDMA programming model as well as the channel arbitration logic. The programming model registers are connected to the internal peripheral bus. The eDMA peripheral request inputs and interrupt request outputs are also connected to this block (via control logic). |
| Control | This block provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read/destination write operations until the number of bytes specified in the minor loop byte count has been moved from the source to the destination.<br>For descriptors where the sizes are not equal, multiple accesses of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-bit data and the destination is 32-bit data, the eDMA performs two reads, then one 32-bit write. |

The transfer control descriptor local memory is further partitioned into:

Table 44. Transfer control descriptor memory

| Submodule | Description |
|---|---|
| Memory controller | This logic implements the required dual-ported controller, and manages accesses from the eDMA engine as well as references from the internal peripheral bus. As noted earlier, in simultaneous accesses, the eDMA engine is given priority and the peripheral transaction is stalled. |
| Memory array | TCD storage for each channel's transfer profile. |

### 14.2.3 Features

The eDMA is a highly programmable data-transfer engine optimized to minimize any required intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known and not defined within the transferred data itself. The eDMA module features:

- All data movement via dual-address transfers: read from source, write to destination
  - Programmable source and destination addresses and transfer size
  - Support for complex address calculations
- 32-channel implementation that performs complex data transfers with minimal intervention from a host processor
  - Internal data buffer, used as temporary storage for all transfers

  - Connections to the crossbar switch for bus mastering the data movement
- TCD organized to support two-deep, nested transfer operations
  - 32-byte TCD stored in local memory for each channel
  - An inner data transfer loop defined by a minor byte transfer count
  - An outer data transfer loop defined by a major iteration count
- Channel activation via one of three methods:
  - Explicit software initiation
  - Initiation via a channel-to-channel linking mechanism for continuous transfers
  - Peripheral-paced hardware requests, one per channel
- Fixed-priority and round-robin channel arbitration
- Channel completion reported via programmable interrupt requests
  - One interrupt per channel, which can be asserted at completion of major iteration count
  - Programmable error terminations per channel that are logically summed together to form one error interrupt to the interrupt controller
- Programmable support for scatter/gather DMA processing
- Support for complex data structures

In the discussion of this module, *n* is used to reference the channel number.

## 14.3 Modes of operation

The eDMA operates in the following modes:

**Table 45. Modes of operation**

| Mode | Description |
|---|---|
| Normal | In Normal mode, eDMA transfers data between a source and a destination. The source and destination can be a memory block or an I/O block capable of operation with eDMA.<br>A service request initiates a transfer of a specific number of bytes (NBYTES) as specified in the TCD. The minor loop is the sequence of read-write operations that transfers these NBYTES per service request. Each service request executes one iteration of the major loop, which transfers NBYTES of data. |
| Debug | eDMA operation is configurable in Debug mode via the control register:<br>• If CSR[EDBG] is cleared to 0, eDMA continues to operate.<br>• If CSR[EDBG] is set to 1, eDMA stops transferring data. If Debug mode is entered when a channel is active, eDMA continues operation until the channel retires. |

## 14.4 Functional description

The operation of eDMA is described in the following subsections.

### 14.4.1 eDMA basic data flow

The basic flow of a data transfer can be partitioned into three segments.

As shown in the following diagram, the first segment involves the channel activation:

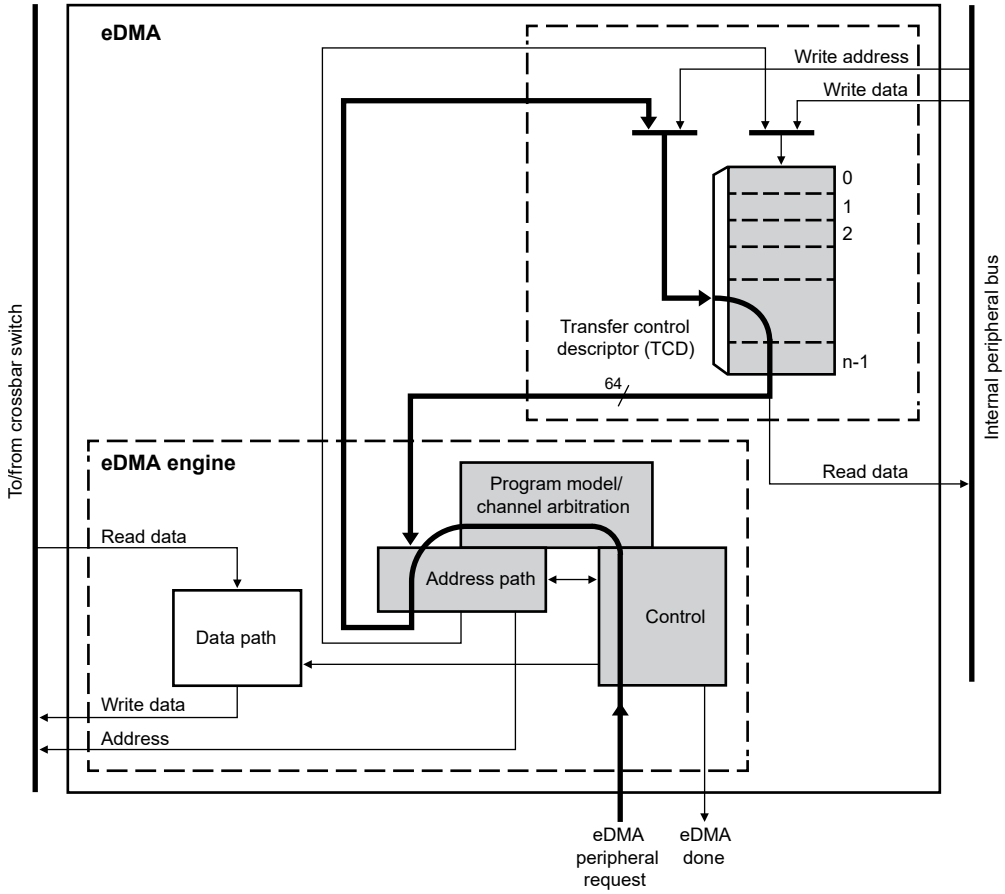



Figure 28. eDMA operation, part 1

This example uses the assertion of the eDMA peripheral request signal to request service for channel *n*. Channel activation via software and the TCD*n*_CSR[START] field follows the same basic flow as peripheral requests. The eDMA request input signal is registered internally and then routed through the eDMA engine: first through the control module, then into the program model and channel arbitration.

In the next cycle, the channel arbitration begins using fixed-priority plus the optional round-robin algorithm. After arbitration is complete, the activated channel number is sent through the address path and converted into the required address to access the local memory for TCD*n*. Next, the TCD memory is accessed and the required descriptor is read from the local memory and then loaded into the eDMA engine address path's primary or secondary channel execution registers. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the address path registers.

The following diagram illustrates the second part of the basic data flow:

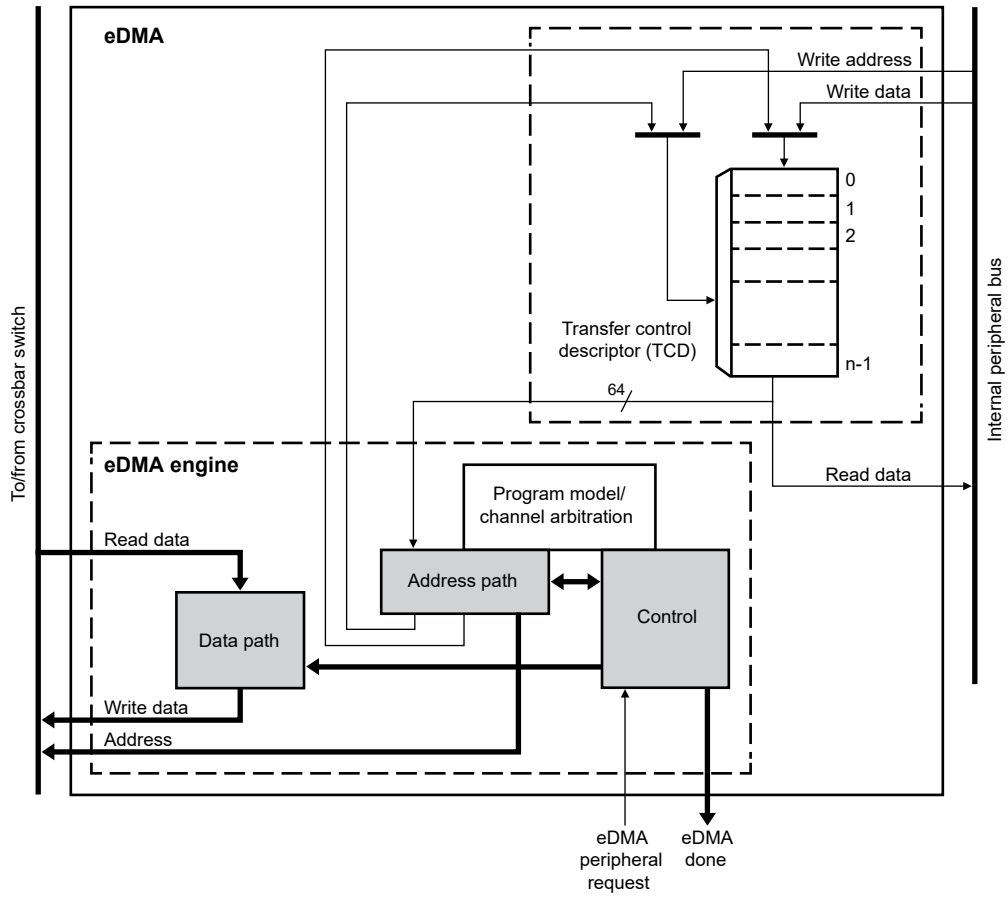



Figure 29. eDMA operation, part 2

The modules associated with the data transfer (address path, data path, and control) go through the required sequence of source reads and destination writes to perform the actual data movement. The source reads are initiated, and the fetched data is temporarily stored in the data path block until it is gated onto the internal bus during the destination write. This source read/destination write processing continues until the byte count, NBYTES, has been transferred.

After NBYTES of data has been moved, the final phase of the basic data flow is performed. In this segment, the address path logic performs the required updates to certain fields in the appropriate TCD (for example, SADDR, DADDR, CITER). If the major iteration count is exhausted, additional operations are performed. These include the final address adjustments and reloading of the BITER field into the CITER field. Assertion of an optional interrupt request also occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor (if scatter/gather is enabled). The updates to the TCD memory and the assertion of an interrupt request are shown in the following diagram.

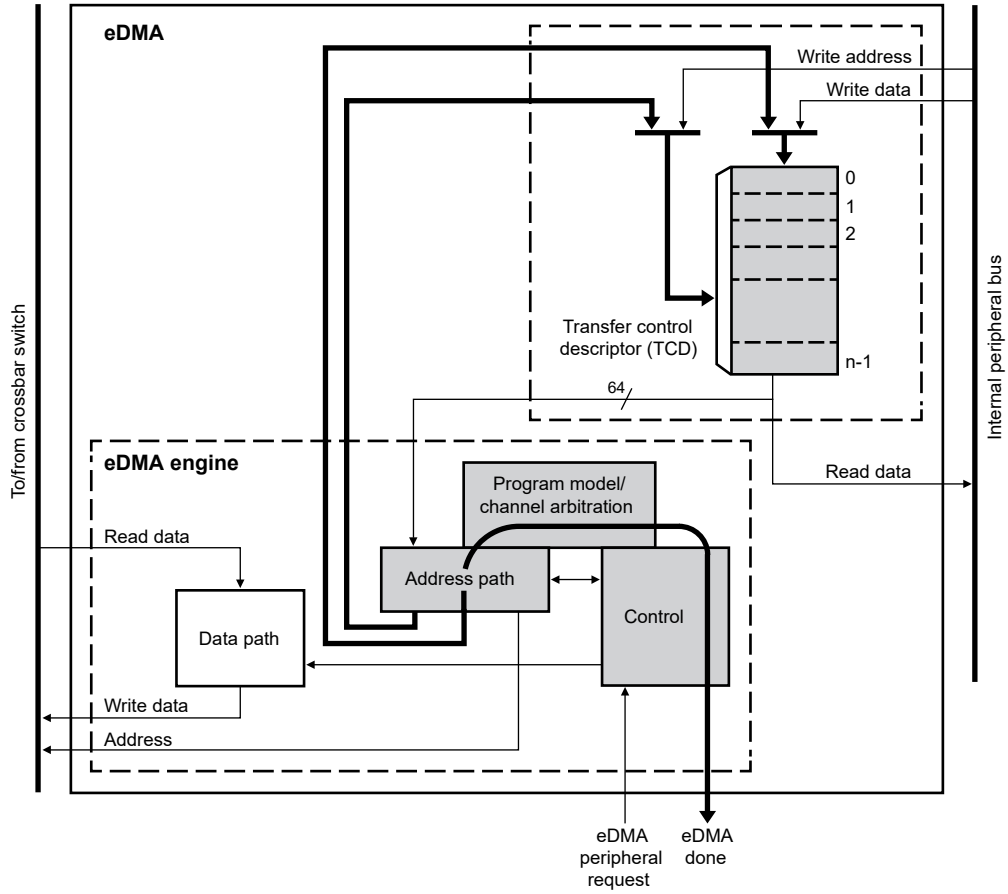



Figure 30. eDMA operation, part 3

## 14.4.2 Fault reporting and handling

Channel errors are reported in the Error Status register (CHn_CSR and TCDn_CSR) and can be caused by any of the following:

- A configuration error, which is an illegal setting in the transfer control descriptor
- An active channel canceled via a "cancel transfer with error" hardware or software request
- A TCD memory error
- An error termination to a bus master read or write cycle

A configuration error is reported when an inconsistent state is represented by one of these factors:

- Starting source or destination address
- Source or destination offsets
- Minor loop byte count
- Transfer size

Each of these possible causes is detailed below:

- The addresses and offsets must be aligned on zero-modulo-transfer-sized boundaries.
- The minor loop byte count must be a multiple of the source and destination transfer sizes.
- All source reads and destination writes must be configured to the natural boundary of the programmed transfer size.

**NOTE**

To aid in debugging, set the Halt After Error field in the DMA's Control Status register, CSR[HAE]. Upon any error condition, the DMA is halted after the error is recorded. The DMA remains halted and does not process any channel service requests. After the error is fixed, the DMA may be enabled again by clearing the Halt field, CSR[HALT].

- If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32-byte boundary.
- If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCDn_CITER[ELINK] field does not equal the TCDn_BITER[ELINK] field.

If enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, are reported as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion if properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

The occurrence of any error causes the eDMA engine to stop normal processing of the active channel immediately (it goes to its error processing states and the transaction to the system bus still has pipeline effect), and the appropriate channel field in the eDMA error register is set to 1. At the same time, the details of the error condition are loaded into the Error Status register (CHn_CSR and TCDn_CSR). The major loop complete indicators, setting the transfer control descriptor DONE flag, and the possible assertion of an interrupt request are not affected when an error is detected.

After the error status has been updated, the eDMA engine continues operating by servicing the next appropriate channel. A channel that experiences an error condition is not automatically disabled. If a channel is terminated by an error and then issues another service request before the error is fixed, that channel executes and terminates with the same error condition.

The error status fields are read-only. These error indicators are sticky and cannot be cleared. They show the last recorded error until the DMA is reset. The valid field (VLD) is used to determine if a new error condition exists. This field is the logical OR of each channel's error interrupt field (ERR).

After the software has resolved any errors and cleared all of the error interrupt fields, the valid field is cleared to 0 but the cause of the last error is still indicated.

### 14.4.3 Channel preemption

Channel preemption is enabled on a per-channel basis by setting the CHn_PRI[ECP] field. Channel preemption allows the executing channel's data transfers to temporarily suspend in favor of starting a higher-priority channel. After the preempting channel has completed all of its minor loop data transfers, the preempted channel is restored and resumes execution.

After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel is suspended, and the higher-priority channel is serviced. Nested preemption, that is, attempting to preempt a preempting channel, is not supported. After a preempting channel begins execution, it cannot be preempted.

A channel's ability to preempt another channel can be disabled by setting CHn_PRI[DPA] to 1. When a channel's preempt ability is disabled, that channel cannot suspend a lower-priority channel's data transfer, regardless of the lower-priority channel's ECP setting. This allows for a pool of low-priority, large-data-moving channels to be defined.

You can configure these low-priority channels to not preempt each other, thus preventing a low-priority channel from consuming the preempt slot normally available to a true high-priority channel. When you enable round-robin channel arbitration mode (CSR[ERCA] is set to 1), any channel with a priority level equal to 0 (CHn_PRI[APL] = 0) has preemption disabled and cannot preempt another channel.

## 14.5 Initialization/application information

The following sections discuss initialization of the eDMA and programming considerations.

### 14.5.1 eDMA initialization

To initialize the eDMA:

1. Write to the CSR if a configuration other than the default is wanted.
2. Write the channel priority levels to the CHn_PRI registers and group priority levels to the CHn_GRPRI registers if a configuration other than the default is wanted.
3. Enable error interrupts in the CHn_CSR[EEI] registers if they are wanted.
4. Write the 32-byte TCD for each channel that may request service.
5. Enable any hardware service requests via the CHn_CSR[ERQ] registers.
6. Request channel service via either:
   - Software: setting TCDn_CSR[START]
   - Hardware: slave device asserting its eDMA peripheral request signal

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The eDMA engine reads the entire TCD, including the TCD control and status fields, as shown in Table 46, for the selected channel into its internal address path module.

As the TCD is read, the first transfer is initiated on the internal bus, unless a configuration error is detected. Transfers from the source, defined by TCDn_SADDR, to the destination, defined by TCD_DADDR, continue until the number of bytes specified by TCDn_NBYTES are transferred.

When the transfer is complete, the eDMA engine's local TCDn_SADDR, TCDn_DADDR, and TCDn_CITER are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, then eDMA executes further post-processing, such as interrupts, major loop channel linking, and scatter/gather operations, if enabled.

**Table 46. TCD control and status (TCDn_CSR) fields**

| TCDn_CSR field name | Description |
|---|---|
| START | Control field to start the channel explicitly when using a software-initiated DMA service (automatically cleared by hardware) |
| EEOP | Control field to enable end-of-packet processing |
| ESDA | Control field to enable storing of the destination address to system memory after the major loop completes |
| DREQ | Control field to disable hardware-initiated DMA service requests after major loop completion |
| BWC | Control field for throttling the bandwidth control of a channel |
| ESG | Control field to enable the scatter-gather feature |
| INTHALF | Control field to enable interrupt when major loop is half-complete |
| INTMAJOR | Control field to enable interrupt when major loop completes |

**Table 47. Channel control and status (CHn_CSR) fields**

| CHn_CSR field name | Description |
|---|---|
| ACTIVE | Status field indicating the channel is currently in execution |
| DONE | Status field indicating major loop completion (cleared by software when a channel begins execution) |
| EEI | Control field to enable error interrupts |

*Table continues on the next page...*

Table 47. Channel control and status (CHn_CSR) fields (continued)

| CHn_CSR field name | Description |
|---|---|
| EARQ | Control field to enable external, asynchronous wakeup event in conjunction with the ERQ field |
| ERQ | Control field to enable hardware service requests |

The following figure shows how each DMA request initiates one minor-loop transfer, or iteration, without CPU intervention. DMA arbitration can occur after each minor loop, and one level of minor loop DMA preemption is allowed. The number of minor loops in a major loop is specified by the beginning iteration count (BITER).

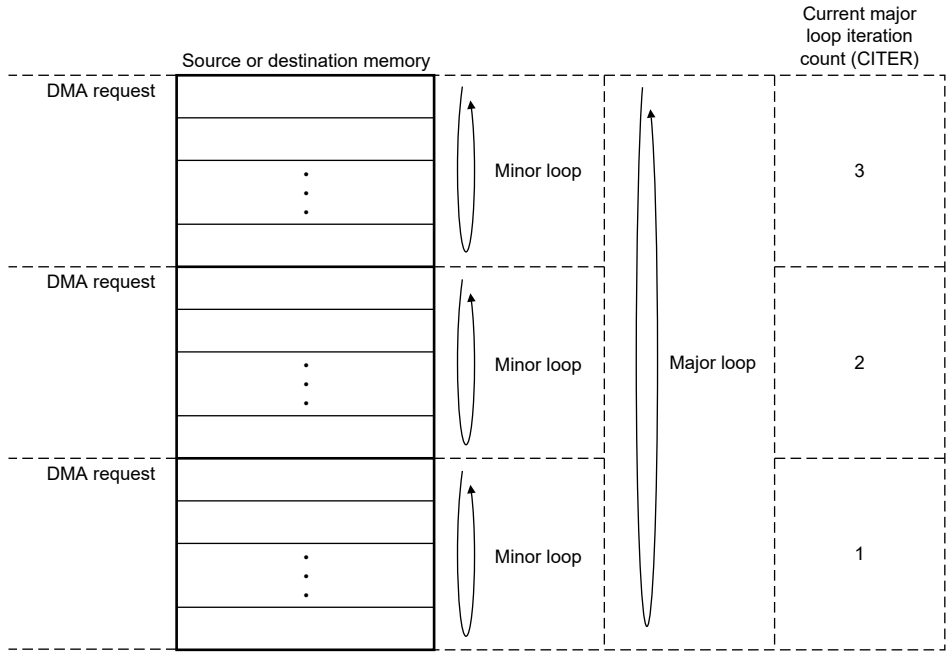



Figure 31. Example of multiple loop iterations

The following figure lists the memory array terms and how the TCD settings are related.

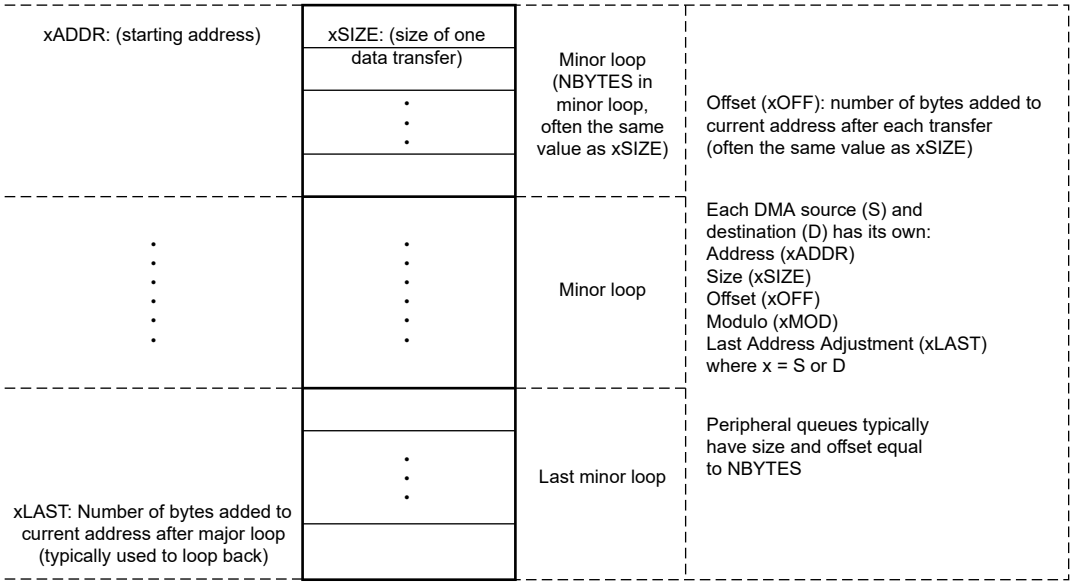



**Figure 32. Memory array terms**

## 14.5.2 eDMA arbitration

The eDMA uses a layered arbitration scheme composed of multiple priority levels. The eDMA uses a fixed-priority arbitration scheme with optional round-robin arbitration under specific conditions. The priorities are evaluated in the following order:

**Table 48. eDMA arbitration priorities**

| Priority | Scheme | Description |
|---|---|---|
| 1 (Highest) | Arbitration group priority | Each channel is assigned an arbitration group via the CHn_GRPRI registers. Priority is given to the highest value (31 being the highest possible value) down to the lowest value (zero, the default). |
| 2 | Channel priority | Each channel is assigned a channel priority level via the CHn_PRI registers. The channel priority is a relative priority level within an arbitration group. Priority is given to the highest value (seven being the highest possible value) down to the lowest value (zero, the default). Channel priorities within each arbitration group need not be unique. If multiple channels have the same channel priority level, the channel number will be used to determine priority as defined in row three. |
| 3 | Channel number | When two or more channels have the same arbitration group priority and channel priority, the channel number (CHn_NUM) is used to determine the highest priority. Priority is giver to the |

*Table continues on the next page...*

Table 48. eDMA arbitration priorities (continued)

| Priority | Scheme | Description |
|---|---|---|
| | | highest channel number. Lowest priority is channel 0. The channel numbers are static and cannot be changed in the programmer's model. |
| 4 (Lowest) | Round-robin | When round-robin is enabled, any channel configured for round-robin operation has lowest priority within an arbitration group. Round-robin is enabled by setting the CSR[ERCA] field to 1. After being enabled, channels with a channel priority of zero (CHn_PRI=0) will use round-robin arbitration. Round-robin arbitration will rotate the channel selection among the channels requesting service with CHn_PRI=0 within the arbitration group. Any non-zero channel within the arbitration group will continue to use fixed-priority arbitration, and if requesting service will be selected over any round-robin channels. |

For fixed arbitration, the overall priority can be considered a number composed of three concatenated priority levels: CHn_GRPRI:CHn_PRI:CH_NUM. The largest number has the highest priority and the lowest number has the lowest priority.

For round-robin arbitration, the priority number is CHn_GRPRI:0:X. The module rotates through the CHn_PRI=0 channels requesting service without regard to priority among these channels. Any channel within the arbitration group for which CHn_PRI is greater than 0 will be serviced before the round-robin channels.

### 14.5.3 Programming errors

The eDMA performs various tests on the transfer control descriptor to verify consistency in the descriptor data.

The channel number causing the error is recorded in the Error Status register (CHn_CSR and TCDn_CSR). If the error source is not removed before the next activation of the problematic channel, the error is detected and recorded again. Setting the halt after error field, CSR[HAE], will halt the DMA and prevent reoccurrence of the error.

### 14.5.4 Arbitration mode considerations

This section discusses arbitration considerations for eDMA.

#### 14.5.4.1 Fixed group arbitration, fixed channel arbitration

In this mode, eDMA selects for execution the channel service request from the highest-priority channel in the highest-priority group. If eDMA is programmed so that the channels within a high-priority group have a high number of requests or large data transfers, that group may consume all the bandwidth of the eDMA controller. That is, no lower-priority groups are serviced if there is always at least one DMA request pending on a channel in the highest-priority group when the controller arbitrates the next DMA request. The advantage of this scenario is that latency can be small for channels that need to be serviced quickly.

#### 14.5.4.2 Fixed group arbitration, round-robin channel arbitration

The highest-priority group with a request is serviced. Lower-priority groups are serviced if no pending requests exist in the higher-priority groups.

Within each group, channels are serviced starting with the highest non-zero channel priority. For all channels with a channel priority programmed to 0, selection begins with the highest channel number requesting service and then rotates through to the lowest channel number requesting service. The round-robin channel arbitration can provide a fairness mechanism to lower-priority channels.

This scenario could cause the same bandwidth consumption problem as indicated in Fixed group arbitration, fixed channel arbitration, but all the channels in the highest-priority group will be serviced. Service latency is short on the highest-priority group, but could potentially be very much longer as the group priority decreases.

## 14.5.5 Performing DMA transfers

This section presents examples on how to perform DMA transfers with the eDMA.

### 14.5.5.1 Single request

To perform a simple transfer of n bytes of data with one activation, set the major loop to one (TCD*n*_CITER = TCD*n*_BITER = 1). The data transfer begins after the channel service request is acknowledged and the channel is selected to execute. After the transfer is complete, the CH*n*_CSR[DONE] field is set to 1 and an interrupt is generated if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has a byte-wide memory port located at 0x1000. The destination memory has a 32-bit port located at 0x2000. The address offsets are programmed in increments to match the transfer size: one byte for the source, and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

```
TCDn_CITER = TCDn_BITER = 1
TCDn_NBYTES = 16
TCDn_SADDR = 0x1000
TCDn_SOFF = 1
TCDn_ATTR[SSIZE] = 0
TCDn_SLAST = -16
TCDn_DADDR = 0x2000
TCDn_DOFF = 4
TCDn_ATTR[DSIZE] = 2
TCDn_DLAST_SGA= -16
TCDn_CSR[INTMAJ] = 1
TCDn_CSR[START] = 1 (should be written last after all other fields have been initialized)
All other TCDn fields = 0
```

This generates the following event sequence:

1. User write to the TCD*n*_CSR[START] field requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes:
   - CH*n*_CSR[DONE] = 0
   - TCD*n*_CSR[START] = 0
   - CH*n*_CSR[ACTIVE] = 1
4. eDMA engine reads: channel TCD data from local memory to internal register file.
5. The source-to-destination transfers are executed as follows:
   a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b. Write 32 bits to location 0x2000 → first iteration of the minor loop.
   c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d. Write 32 bits to location 0x2004 → second iteration of the minor loop.

e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.

f. Write 32 bits to location 0x2008 → third iteration of the minor loop.

g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.

h. Write 32 bits to location 0x200C → last iteration of the minor loop → major loop complete.

6. The eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 1 (TCD*n*_BITER).
7. The eDMA engine writes: CH*n*_CSR[ACTIVE] = 0, CH*n*_CSR[DONE] = 1, CH*n*_INT[INT] = 1.
8. The channel retires and the eDMA goes idle or services the next channel.

#### 14.5.5.2 Multiple requests

The following example transfers 32 bytes via two hardware requests, but is otherwise the same as the previous example. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations of the major loop, transferring 16 bytes per iteration. After the channel's hardware requests are enabled via the CHn_CSR[ERQ] register field, the slave device initiates channel service requests.

```
TCDn_CITER = TCDn_BITER = 2
TCDn_SLAST = -32
TCDn_DLAST_SGA = -32
```

This would generate the following sequence of events:

1. First hardware (eDMA peripheral) requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: CH*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, CH*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD*n* data from local memory to internal register file.
5. The source-to-destination transfers are executed as follows:
   a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b. Write 32 bits to location 0x2000 → first iteration of the minor loop.
   c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d. Write 32 bits to location 0x2004 → second iteration of the minor loop.
   e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
   f. Write 32 bits to location 0x2008 → third iteration of the minor loop.
   g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
   h. Write 32 bits to location 0x200C → last iteration of the minor loop.
6. eDMA engine writes: TCD*n*_SADDR = 0x1010, TCD*n*_DADDR = 0x2010, TCD*n*_CITER = 1.
7. eDMA engine writes: CH*n*_CSR[ACTIVE] = 0.
8. The channel retires, which concludes one iteration of the major loop. The eDMA goes idle or services the next channel.
9. Second hardware (eDMA peripheral) requests channel service.
10. The channel is selected by arbitration for servicing.
11. eDMA engine writes: CH*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, CH*n*_CSR[ACTIVE] = 1.
12. eDMA engine reads: Channel TCD data from local memory to internal register file.
13. The source-to-destination transfers are executed as follows:
    a. Read byte from location 0x1010, read byte from location 0x1011, read byte from 0x1012, read byte from 0x1013.

b. Write 32 bits to location 0x2010 → first iteration of the minor loop.

c. Read byte from location 0x1014, read byte from location 0x1015, read byte from 0x1016, read byte from 0x1017.

d. Write 32 bits to location 0x2014 → second iteration of the minor loop.

e. Read byte from location 0x1018, read byte from location 0x1019, read byte from 0x101A, read byte from 0x101B.

f. Write 32 bits to location 0x2018 → third iteration of the minor loop.

g. Read byte from location 0x101C, read byte from location 0x101D, read byte from 0x101E, read byte from 0x101F.

h. Write 32 bits to location 0x201C → last iteration of the minor loop → major loop complete.

14. eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 2 (TCD*n*_BITER).
15. eDMA engine writes: CH*n*_CSR[ACTIVE] = 0, CH*n*_CSR[DONE] = 1, CH*n*_INT[INT] = 1.
16. The channel retires, which concludes with the major loop complete. The eDMA goes idle or services the next channel.

#### 14.5.5.3 Using the modulo feature

The modulo feature of the eDMA allows implementation of a circular data queue in which the size of the queue is a power of 2. xMOD is a 5-bit field for the source and destination in the TCD, and it specifies which lower address bits increment from their original value after the address+offset calculation. All upper address bits remain the same as in the original value. A setting of 0 for this field disables the modulo feature. Modulo addressing applies to cases where the minor loop offset is enabled; that is, the upper address bits remain the same after the minor loop offset is added to the source or destination address.

The following table shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value but the 28 upper address bits (0x1234567*x*) retain their original value. In this example, the source address is set to 0x12345670, the offset is set to four bytes, and the MOD field is set to four, which allows for a $2^4$ byte (16 byte) queue size.

Table 49. Modulo example

| Transfer number | Address |
|---|---|
| 1 | 0x12345670 |
| 2 | 0x12345674 |
| 3 | 0x12345678 |
| 4 | 0x1234567C |
| 5 | 0x12345670 |
| 6 | 0x12345674 |

### 14.5.6 Monitoring transfer descriptor status

This section discusses how to monitor eDMA status.

#### 14.5.6.1 Testing for minor loop completion

There are two methods to test for minor loop completion when using software-initiated service requests.

1. The first method is to read the TCDn_CITER field and test for a change.
2. The second method, extracted from the sequence shown below, is to test the TCDn_CSR[START] field and the CHn_CSR[ACTIVE] field. The minor-loop-complete condition is indicated by both fields reading 0 after TCDn_CSR[START] is set to 1. Polling the CHn_CSR[ACTIVE] field only may be inconclusive because the active status may be missed if the channel execution is short in duration.

The CHn_CSR and TCDn_CSR status fields execute the following sequence for a software-activated channel:

| Stage | TCDn_CSR field | CHn_CSR fields | | State |
|---|---|---|---|---|
| | START | ACTIVE | DONE | |
| 1 | 1 | 0 | 0 | Initiate channel service request via software. |
| 2 | 0 | 1 | 0 | Channel is executing. |
| 3a | 0 | 0 | 0 | Channel has completed the minor loop and is idle. |
| 3b | 0 | 0 | 1 | Channel has completed the major loop and is idle. |

The best method to test for minor-loop completion when using hardware-initiated (that is, peripheral-initiated) service requests is to read the TCDn_CITER field and test for a change. The hardware request and acknowledge handshake signals are not visible in the programmer's model.

The TCD status fields execute the following sequence for a hardware-activated channel:

| Stage | TCDn_CSR field | CHn_CSR fields | | State |
|---|---|---|---|---|
| | START | ACTIVE | DONE | |
| 1 | 0 | 0 | 0 | Initiate channel service request via hardware (peripheral request asserted). |
| 2 | 0 | 1 | 0 | Channel is executing. |
| 3a | 0 | 0 | 0 | Channel has completed the minor loop and is idle. |
| 3b | 0 | 0 | 1 | Channel has completed the major loop and is idle. |

For both activation types, the major-loop-complete status is explicitly indicated via the CHn_CSR[DONE] field.

The TCDn_CSR[START] field is cleared to 0 automatically when the channel begins execution, regardless of how the channel activates.

#### 14.5.6.2 Reading the transfer descriptors of active channels

The eDMA reads back the true TCDn_SADDR, TCDn_DADDR, and TCDn_NBYTES values if they are read when a channel executes. The true values of SADDR, DADDR, and NBYTES are the values the eDMA engine currently uses in its internal register file, and not the values in the TCD local memory for that channel. The addresses, SADDR and DADDR, and NBYTES (which decrements to zero as the transfer progresses), can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

#### 14.5.6.3 Checking channel preemption status

A preemptive situation is one in which a preempt-enabled channel is executing and a higher-priority request becomes active. When round-robin channel arbitration mode is enabled, all channels with their channel priority set to 0 lose their preempt ability. Channel priorities of 0 are treated as equal, that is, they are constantly rotating, when round-robin arbitration mode is enabled.

The CHn_CSR[ACTIVE] field for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended when the preempting channel executes one major loop iteration. If two CHn_CSR[ACTIVE] fields are set simultaneously in the global TCD map, a higher-priority channel is actively preempting a lower-priority channel.

### 14.5.7 Channel linking

Channel linking (or chaining) is a mechanism in which one channel sets the TCDn_CSR[START] field of another channel (or itself), thus initiating a service request for that channel. When properly enabled, the eDMA engine automatically performs this operation at the major or minor loop completion.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCDn_CITER[ELINK] field determines whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, using an initial field setting of:

```
TCDn_CITER[ELINK] = 1
TCDn_CITER[LINKCH] = 0xC
TCDn_CITER[CITER] value = 0x4
TCDn_CSR[MAJORELINK] = 1
TCDn_CSR[MAJORLINKCH] = 0x7
```

executes as:

1. Minor loop done → set TCD12_CSR[START] field
2. Minor loop done → set TCD12_CSR[START] field
3. Minor loop done → set TCD12_CSR[START] field
4. Minor loop done, major loop done→ set TCD7_CSR[START] field

When minor loop linking is enabled (TCDn_CITER[ELINK] = 1), the TCDn_CITER[CITER] field uses a nine-bit vector to form the current iteration count. When minor loop linking is disabled (TCDn_CITER[ELINK] = 0), the TCDn_CITER[CITER] field uses a 15-bit vector to form the current iteration count. The bits associated with the TCDn_CITER[LINKCH] field are concatenated onto the CITER value to increase the range of the CITER.

**NOTE**

The TCDn_CITER[ELINK] field and the TCDn_BITER[ELINK] field must be equal — if they are not, a configuration error is reported. The CITER and BITER vector widths must be equal to calculate the major loop halfway done interrupt point.

The following table summarizes how a DMA channel can link to another DMA channel, that is, use another channel's TCD, at the end of a loop.

**Table 50. Channel linking parameters**

| Wanted link behavior | TCD control field name | Description |
|---|---|---|
| Link at end of minor loop | TCDn_CITER[ELINK] | Enable channel-to-channel linking on minor loop completion (current iteration) |
| | TCDn_CITER[LINKCH] | Link channel number when linking at end of minor loop (current iteration) |
| Link at end of major loop | TCDn_CSR[MAJORELINK] | Enable channel-to-channel linking on major loop completion |
| | TCDn_CSR[MAJORLINKCH] | Link channel number when linking at end of major loop |

### 14.5.8 Dynamic programming

This section provides recommended methods to change the programming model during channel execution.

#### 14.5.8.1 Dynamically changing the channel priority

To change group or channel priority levels:

1. Halt the DMA by writing 1 to the CSR[HALT] field.
2. Change the group or channel priorities as wanted.
3. Enable normal DMA operations by writing 0 to the CSR[HALT] field.

#### 14.5.8.2 Dynamic channel linking

Dynamic channel linking is the process of setting the TCDn_CSR[MAJORELINK] field during channel execution (see the diagram in TCD structure). This field is read from the TCD local memory at the end of channel execution, thus allowing you to enable the feature during channel execution.

Because you are allowed to change the configuration during execution, you need a coherency model. Consider the scenario where you attempt to execute a dynamic channel link by enabling the TCDn_CSR[MAJORELINK] field at the same time the eDMA engine is retiring the channel. TCDn_CSR[MAJORELINK] would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

We recommend that you use the following coherency model when executing a dynamic channel link request.

1. Write 1 to the TCDn_CSR[MAJORELINK] field.
2. Read back the TCDn_CSR[MAJORELINK] field.
3. Test the TCDn_CSR[MAJORELINK] request status:
   - If TCDn_CSR[MAJORELINK] = 1, the dynamic link attempt was successful.
   - If TCDn_CSR[MAJORELINK] = 0, the attempted dynamic link did not succeed (the channel was already retiring).

For this request, the TCD local memory controller forces the TCDn_CSR[MAJORELINK] field to 0 on any writes to a channel's TCDn_CSR[7:0] after that channel's CHn_CSR[DONE] field is set to 1, indicating the major loop is complete.

**NOTE**

You must clear the CHn_CSR[DONE] field to 0 before writing to the TCDn_CSR[MAJORELINK] field. The CHn_CSR[DONE] field is cleared to 0 automatically by the eDMA engine after a channel begins execution.

#### 14.5.8.3 Dynamic scatter/gather

Scatter/gather is the process of automatically loading a new TCD into a channel. It allows a DMA channel to use multiple TCDs; this enables a DMA channel to scatter the DMA data to multiple destinations or gather it from multiple sources. When scatter/gather is enabled and the channel has finished its major loop, a new TCD is fetched from system memory and loaded into that channel's descriptor location in the eDMA programmer's model, thus replacing the current descriptor.

Because you are allowed to change the configuration during execution, you need a coherency model. Consider the scenario where you attempt to execute a dynamic scatter/gather operation by enabling the TCDn_CSR[ESG] field at the same time the eDMA engine is retiring the channel. The TCDn_CSR[ESG] field would be set in the programmer's model, but it would be unclear whether the actual scatter/gather request was honored before the channel retired.

Two methods are recommended for executing a dynamic scatter/gather request. Whenever the TCDn_CSR is written, the TCD local memory controller forces the TCDn_CSR[ESG] field to 0 on any writes to a channel's TCDn_CSR[7:0] after that channel's CHn_CSR[DONE] field has been set to 1, indicating the major loop is complete. If attempting to set the ESG, ensure the DONE field is cleared to 0.

**NOTE**

You must clear the CHn_CSR[DONE] field to 0 before writing the TCDn_CSR[MAJORELINK] or TCDn_CSR[ESG] fields. The CHn_CSR[DONE] field is cleared to 0 automatically by the eDMA engine after a channel begins execution and is set to 1 upon major loop completion.

#### 14.5.8.3.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request.

When the TCDn_CSR[MAJORELINK] field is 0, the TCDn_CSR[MAJORLINKCH] field is not used by the eDMA. In this case, the TCDn_CSR[MAJORLINKCH] bits may be used for other purposes. This method uses the TCDn_CSR[MAJORLINKCH] field as a TCDn_CSR identification (ID).

When the descriptors are built, write a unique TCDn_CSR ID in the TCDn_CSR[MAJORLINKCH] field for each TCDn_CSR associated with a channel using dynamic scatter/gather.

1. Write a 1 to the TCDn_CSR[DREQ] field. Should a dynamic scatter/gather attempt fail, setting the TCDn_CSR[DREQ] field to 1 will prevent future hardware activation of this channel. This stops the channel from executing with a destination address (daddr) that was calculated using a scatter/gather address (written in the next step) instead of a DLAST final offset value.
2. Write the TCDn_DLAST_SGA field with the scatter/gather address.
3. Write a 1 to the TCDn_CSR[ESG] field.
4. Read back the 16-bit TCDn_CSR control/status field.
5. Test the TCDn_CSR[ESG] request status and TCDn_CSR[MAJORLINKCH] value:
   - If ESG = 1, the dynamic scatter/gather attempt was successful.
   - If ESG = 0 and the MAJORLINKCH (ID) did not change, the dynamic scatter/gather attempt was not successful (the channel was already retiring).
   - If ESG = 0 and the MAJORLINKCH (ID) changed, the dynamic scatter/gather attempt was successful (the new TCDn_CSR's ESG value cleared the ESG field to 0).

#### 14.5.8.3.2 Method 2 (channel using major loop channel linking)

For a channel using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request. This method uses the TCDn_DLAST_SGA field as a TCD identification (ID).

1. Write a 1 to the TCDn_CSR[DREQ] field. Should a dynamic scatter/gather attempt fail, setting the DREQ field to 1 will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (DADDR) that was calculated using a scatter/gather address (written in the next step) instead of a DLAST final offset value.
2. Write the TCDn_DLAST_SGA field with the scatter/gather address.
3. Write a 1 to the TCDn_CSR[ESG] field.
4. Read back the TCDn_CSR[ESG] field.
5. Test the TCDn_CSR[ESG] request status:
   - If ESG = 1, the dynamic scatter/gather attempt was successful.
   - If ESG = 0, read the 32-bit TCDn_DLAST_SGA field.
   - If ESG = 0 and the TCDn_DLAST_SGA did not change, the dynamic scatter/gather attempt was not successful (the channel was already retiring).
   - If ESG = 0 and the TCDn_DLAST_SGA changed, the dynamic scatter/gather attempt was successful (the new TCDn_CSR's ESG value cleared the ESG field to 0).

### 14.5.9 Suspend/resume a DMA channel with active hardware service requests

The DMA allows you to move data from memory or peripheral registers to another location in memory or to peripheral registers without CPU interaction. After the DMA and peripherals are configured and active, it is rare but supported to suspend a peripheral's service request dynamically. In this scenario, there are certain restrictions to disabling a DMA hardware service request. For

coherency, you must follow a specific procedure. This section provides guidance on how to coherently suspend and resume a Direct Memory Access (DMA) channel when the DMA is triggered by a slave module such as the Serial Peripheral Interface (DSPI), Sigma Delta Analog to Digital Convertor (SDADC), or other module.

#### 14.5.9.1 Suspend an active DMA channel

To suspend an active DMA channel:

1. Stop the DMA service request at the peripheral first. Confirm it has been disabled by reading back the appropriate register in the peripheral.
2. Check the DMA's Hardware Request Status (HRS) to ensure there is no service request to the DMA channel being suspended. Then disable the hardware service request by clearing the ERQ field to 0 on the appropriate DMA channel.

For example, assume the DSPI is set as a master for transmitting data via a DMA service request when the TXFIFO has an empty slot. The DMA will transfer the next command and data to the TXFIFO upon the request. If you need to suspend the DMA/DSPI transfer loop, perform the following steps:

1. Disable the DMA service request at the source by writing 0 to DSPI_RSER[TFFF_RE]. Confirm that DSPI_RSER[TFFF_RE] is 0.
2. Ensure there is no DMA service request from the DSPI by verifying that HRS[HRS] is 0 for the appropriate channel. If no service request is present, disable the DMA channel by clearing the channel's ERQ field to 0. If a service request is present, wait until the request has been processed and the HRS field reads 0.

#### 14.5.9.2 Resume a DMA channel

To resume a DMA channel:

1. Enable the DMA service request on the appropriate channel by setting its ERQ field to 1.
2. Enable the DMA service request at the peripheral.

## 14.6 Memory map/register definition

The eDMA programming model is partitioned into three parts:

1. The first part defines a number of registers providing overall control functions and is known as the management page.
2. The second part corresponds to the channel (CH) control, status, and configuration.
3. The third part corresponds to the local TCD memory.

#### TCD memory

Each channel requires a 32-byte transfer control descriptor for defining the data movement operation. Each TCDn definition is presented as 11 registers of 16 or 32 bits.

#### TCD initialization

The TCD memory is in an unknown state after reset. Only the TCD START bit is initialized to 0. Prior to activating a channel, you must initialize its TCD with the appropriate transfer profile.

TCD structure

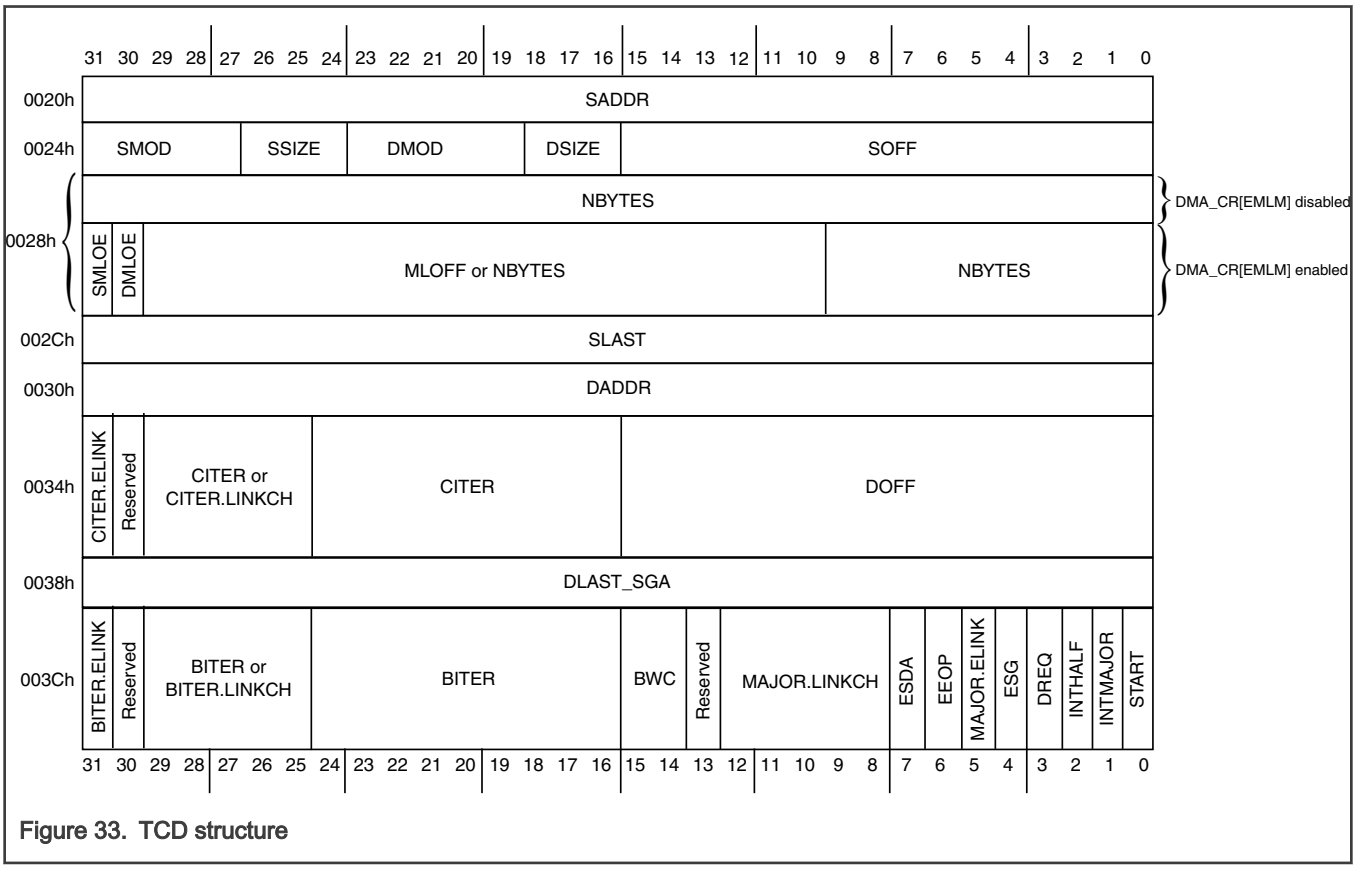



Figure 33. TCD structure

Accesses to reserved memory and fields

- Reading reserved fields in a register returns the value of zero.
- Writes to reserved fields in a register are ignored.
- Reading or writing a reserved memory location generates a bus error.

## 14.6.1 eDMA register descriptions

### 14.6.1.1 eDMA memory map

eDMA base address: 4020_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Management Page Control (CSR) | 32 | RW | 0030_0000h |
| 4h | Management Page Error Status (ES) | 32 | R | 0000_0000h |
| 8h | Management Page Interrupt Request Status (INT) | 32 | R | 0000_0000h |
| Ch | Management Page Hardware Request Status (HRS) | 32 | R | 0000_0000h |
| 100h - 17Ch | Channel Arbitration Group (CH0_GRPRI - CH31_GRPRI) | 32 | RW | 0000_0000h |

### 14.6.1.2 Management Page Control (CSR)

#### Offset

| Register | Offset |
|---|---|
| CSR | 0h |

#### Function

The Management Page Control register defines the basic operating configuration of the DMA.

Arbitration uses a two-tier priority system; group and channel priority. The eDMA assigns each channel to a priority group. Group arbitration is fixed-priority and cannot be changed. Channel arbitration uses fixed priority and may be configured to use a selective round-robin scheme for specified channels within each priority group. For fixed-priority arbitration, eDMA selects for execution the highest priority channel requesting service in the highest priority arbitration group.

The channel priority registers assign the relative priorities within each arbitration group; see CHn_PRI. All channels with a non-zero CHn_PRI value use fixed-priority arbitration.

When you enable round-robin arbitration, all channels with channel priority set to zero do not have a priority and, of those channels requesting service, are cycled through (from high to low channel number) without regard to priority relative to each other within the same priority group. Any channel with a non-zero CHn_PRI value automatically has a higher priority over the round-robin channels. A channel's priority group is assigned in Channel Arbitration Group (CH0_GRPRI - CH31_GRPRI).

**NOTE**

For correct operation, changes to the CSR[ERCA, GCLC, GMRC] fields must be performed when the DMA channels are inactive; that is, when the CSR[ACTIVE] field is 0.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ACTIVE | Reserved | | | ACTIVE_ID | | | | Reserved | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | CX | ECX | GMRC | GCLC | HALT | HAE | Reserved | ERCA | EDBG | Reserved |
| W | 0 | | | | | | | | | | | | 0 | | | 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31<br>ACTIVE | DMA Active Status<br>0b - eDMA is idle<br>1b - eDMA is executing a channel |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 30-29<br>— | Reserved |
| 28-24<br>ACTIVE_ID | Active Channel ID<br>This field identifies the channel number that is executing when the ACTIVE bit is 1. |
| 23-16<br>— | Reserved |
| 15-10<br>— | Reserved |
| 9<br>CX | Cancel Transfer<br>When set to 1, this field cancels the remaining data transfer, stops the executing channel, and forces the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. CX clears itself to 0 after the cancel has been honored. This cancel retires the channel normally as if the minor loop had been completed.<br>0b - Normal operation<br>1b - Cancel the remaining data transfer |
| 8<br>ECX | Cancel Transfer With Error<br>Cancellation of the remaining data transfer is similar to that of the CX field. Execution of the the channel is stopped and the minor loop is forced to finish. The cancellation takes effect after the last write of the current read/write sequence. The ECX field clears itself to 0 after the cancel is honored. In addition to cancelling the transfer, ECX treats the cancel as an error condition, thus updating Management Page Error Status (ES) and generating an optional error interrupt.<br>0b - Normal operation<br>1b - Cancel the remaining data transfer |
| 7<br>GMRC | Global Master ID Replication Control<br>**NOTE**<br>If master ID replication is disabled, the privileged protection level (Supervisor mode) for DMA transfers is used.<br>0b - Master ID replication disabled for all channels<br>1b - Master ID replication available and controlled by each channel's CHn_SBR[EMI] setting |
| 6<br>GCLC | Global Channel Linking Control<br>0b - Channel linking disabled for all channels<br>1b - Channel linking available and controlled by each channel's link settings |
| 5 | Halt DMA Operations |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| HALT | This field stalls the start of any new channels. Executing channels are allowed to complete. Channel execution resumes when this field is cleared to 0.<br>0b - Normal operation<br>1b - Stall the start of any new channels |
| 4<br>HAE | Halt After Error<br>When this field is set to 1, any error causes the HALT field to be set to 1. Then all service requests are ignored until the HALT field is cleared to 0.<br>0b - Normal operation<br>1b - Any error causes the HALT field to be set to 1 |
| 3<br>— | Reserved |
| 2<br>ERCA | Enable Round Robin Channel Arbitration<br>0b - Round-robin channel arbitration disabled. Fixed priority arbitration used for channel selection within each group<br>1b - Round-robin channel arbitration enabled. Round-robin arbitration used for channel selection within each group |
| 1<br>EDBG | Enable Debug<br>When in debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete. DMA resumes channel execution when the system exits debug mode or clears the EDBG field to 0.<br>0b - Debug mode disabled. When in debug mode, the DMA continues to operate<br>1b - Debug mode is enabled. When in debug mode, the DMA stalls the start of a new channel |
| 0<br>— | Reserved |

#### 14.6.1.3 Management Page Error Status (ES)

**Offset**

| Register | Offset |
|---|---|
| ES | 4h |

**Function**

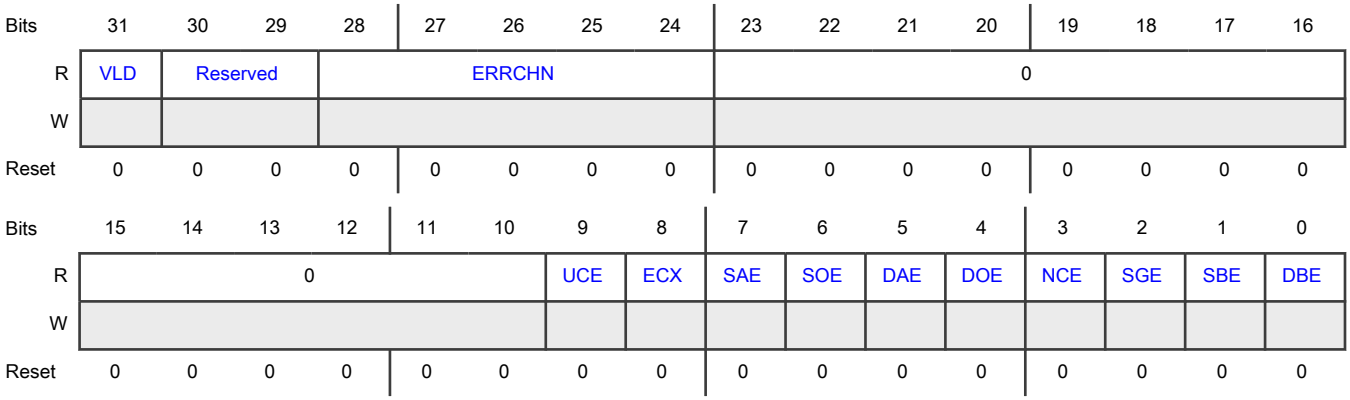
The ES provides information concerning the last recorded channel error. Channel errors can be caused by:

- An illegal setting in the transfer control descriptor
- An error termination to a bus master read or write cycle

- An uncorrectable error that occurred when the device was accessing the TCD SRAM
- A "cancel transfer with error" request was made via the corresponding cancel transfer field or input signal

Upon any error condition, the software must initialize the TCD of the channel that contains the error, as it is in an incomplete state after an error. See Fault reporting and handling for more details.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>VLD | Valid<br>Logical OR of all ERR status fields.<br>0b - No ERR fields are set to 1<br>1b - At least one ERR field is set to 1, indicating a valid error exists that software has not cleared |
| 30-29<br>— | Reserved |
| 28-24<br>ERRCHN | Error Channel Number or Canceled Channel Number<br>The channel number of the last recorded error or last recorded error-canceled transfer. |
| 23-10<br>— | Reserved |
| 9<br>UCE | Uncorrectable TCD Error During Channel Execution<br>UCE is set to 1 only when an uncorrectable ECC error occurs on an access generated by the DMA. If a CPU access to the TCD causes an uncorrectable ECC error, then that access receives a bus error response.<br>**NOTE**<br>When the eDMA sees a RAM error on an IPS access (when you are accessing a TCD), it reports the error as a bus abort. When the DMA engine receives a RAM error (the execution engine is accessing a TCD) it is recorded in the Error Status register, ES[UCE], along with the channel number. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No uncorrectable ECC error<br>1b - Last recorded error was an uncorrectable TCD RAM error |
| 8<br>ECX | Transfer Canceled<br>The ECX operation is a management page function. When employed, the targeted channel's CHn_ES register reports an unspecified error; that is, only the ERR field is set to 1. The management page has full view of the error condition.<br>0b - No canceled transfers<br>1b - Last recorded entry was a canceled transfer by the error cancel transfer input |
| 7<br>SAE | Source Address Error<br>When this field is 1, it indicates that TCDn_SADDR is inconsistent with TCDn_ATTR[SSIZE].<br>0b - No source address configuration error<br>1b - Last recorded error was a configuration error detected in the TCDn_SADDR field |
| 6<br>SOE | Source Offset Error<br>When this field is 1, it indicates that TCDn_SOFF is inconsistent with TCDn_ATTR[SSIZE].<br>0b - No source offset configuration error<br>1b - Last recorded error was a configuration error detected in the TCDn_SOFF field |
| 5<br>DAE | Destination Address Error<br>When this field is 1, it indicates that TCDn_DADDR is inconsistent with TCDn_ATTR[DSIZE].<br>0b - No destination address configuration error<br>1b - Last recorded error was a configuration error detected in the TCDn_DADDR field |
| 4<br>DOE | Destination Offset Error<br>When this field is 1, it indicates that TCDn_DOFF is inconsistent with TCDn_ATTR[DSIZE].<br>0b - No destination offset configuration error<br>1b - Last recorded error was a configuration error detected in the TCDn_DOFF field |
| 3<br>NCE | NBYTES/CITER Configuration Error<br>This error indicates that one of the following has occurred:<br>• TCDn_NBYTES is not a multiple of TCDn_ATTR[SSIZE] and TCDn_ATTR[DSIZE]<br>• TCDn_CITER[CITER] is equal to zero<br>• TCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK]<br>0b - No NBYTES/CITER configuration error<br>1b - The last recorded error was NBYTES equal to zero or a CITER not equal to BITER error. Last recorded error was a configuration error detected in the TCDn_NBYTES or TCDn_CITER fields |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 2<br>SGE | Scatter/Gather Configuration Error<br>When this field is 1, it indicates that TCDn_DLAST_SGA is not on a 32-byte boundary. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCDn_CSR[ESG] is enabled.<br>0b - No scatter/gather configuration error<br>1b - Last recorded error was a configuration error detected in the TCDn_DLAST_SGA field |
| 1<br>SBE | Source Bus Error<br>0b - No source bus error<br>1b - Last recorded error was a bus error on a source read |
| 0<br>DBE | Destination Bus Error<br>0b - No destination bus error<br>1b - Last recorded error was a bus error on a destination write |

#### 14.6.1.4 Management Page Interrupt Request Status (INT)

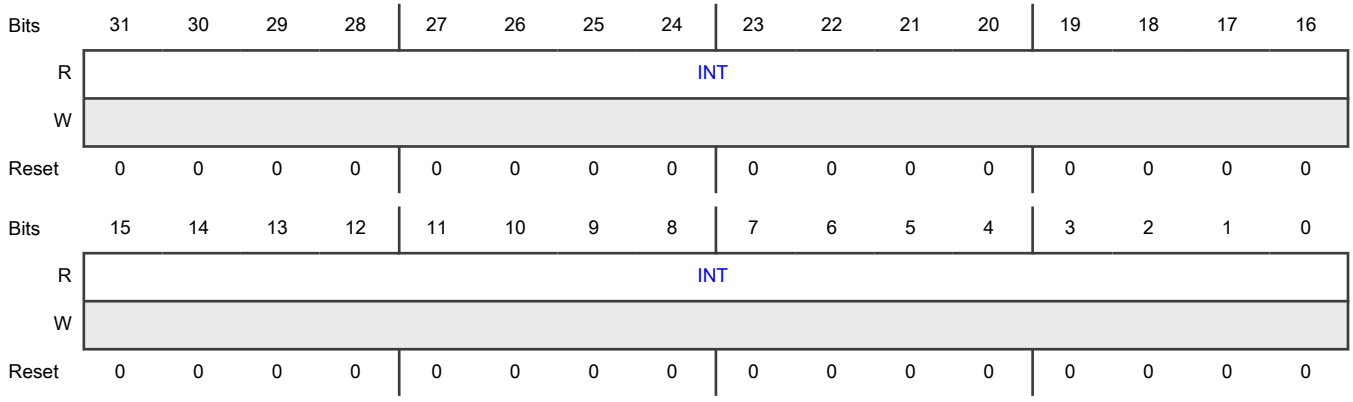
##### Offset

| Register | Offset |
|---|---|
| INT | 8h |

##### Function

This register shows the current state of the interrupt service requests for all eDMA channels.

##### Diagram



### Fields

| Field | Function |
|---|---|
| 31-0<br>INT | Interrupt Request Status<br>The INT register presents the interrupt request status for each eDMA channel. Depending on the appropriate field setting in the transfer control descriptors, the eDMA engine generates an interrupt on data transfer completion or an error condition. The eDMA routes channel interrupt requests to the interrupt controller. During the interrupt service routine associated with any given channel, it is the software's responsibility to clear the appropriate field in the channel's interrupt request register, CHn_INT, thus negating the interrupt request.<br>0b - Interrupt request for corresponding channel not present<br>1b - Interrupt request for corresponding channel present |

## 14.6.1.5 Management Page Hardware Request Status (HRS)

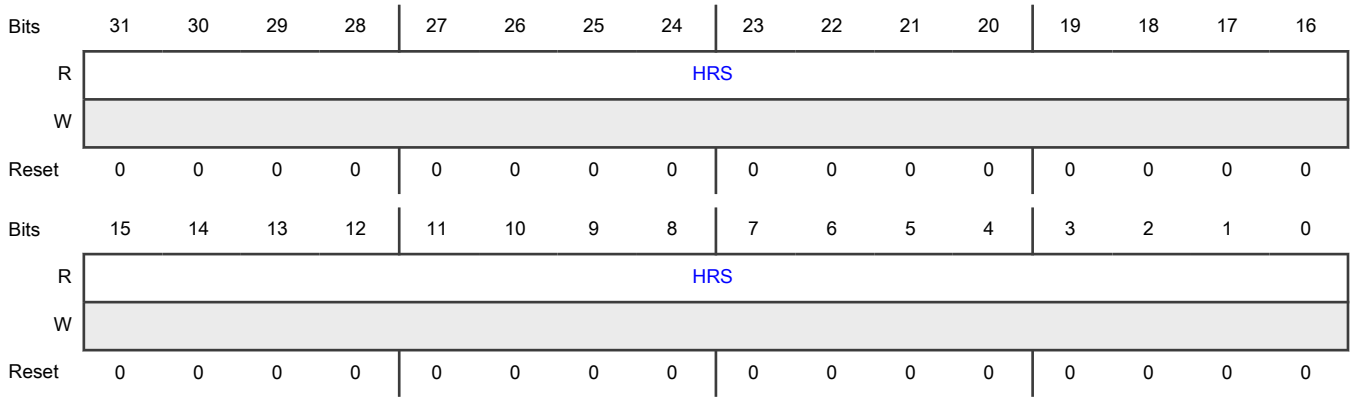
### Offset

| Register | Offset |
|---|---|
| HRS | Ch |

### Function

The hardware request status register (HRS) shows the current state of the hardware service request signaling as seen by eDMA's arbitration logic.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>HRS | Hardware Request Status |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | The HRS bit for its respective channel remains asserted for the period when a hardware request is present on the channel. After the request is completed and the channel is free, the hardware automatically clears the corresponding HRS bit to 0.<br>0b - Hardware service request for corresponding channel is not present<br>1b - Hardware service request for corresponding channel is present |

### 14.6.1.6 Channel Arbitration Group (CH0_GRPRI - CH31_GRPRI)

#### Offset

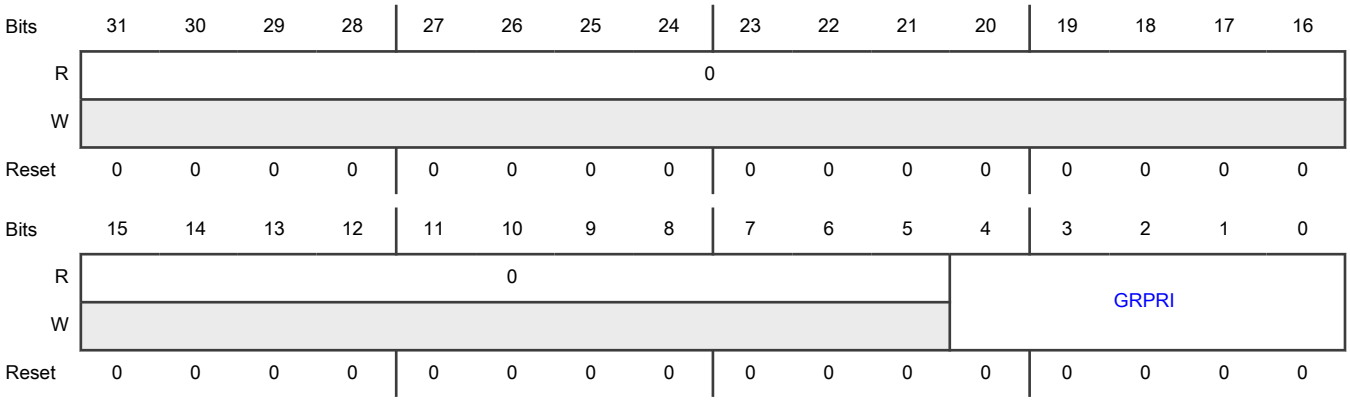
For n = 0 to 31:

| Register | Offset |
|---|---|
| CHn_GRPRI | 100h + (n × 4h) |

#### Function

The contents of this register define the arbitration group associated with each channel. Using a fixed-priority group arbitration scheme, eDMA evaluates the arbitration group priorities by numeric value from highest group number to lowest; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, and so on. The range of the group priority values is limited to the values of 0 through 31. Within each arbitration group, the channel priority assignment CH*n*_PRI determines the highest-priority channel.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-5<br>— | Reserved |
| 4-0<br>GRPRI | Arbitration Group For Channel n<br>Fixed-priority arbitration group number. |

## 14.6.2 TCD register descriptions

### 14.6.2.1 TCD memory map

TCD base address: 4021_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Channel Control and Status (CH0_CSR) | 32 | RW | 0000_0000h |
| 4h | Channel Error Status (CH0_ES) | 32 | RW | 0000_0000h |
| 8h | Channel Interrupt Status (CH0_INT) | 32 | RW | 0000_0000h |
| Ch | Channel System Bus (CH0_SBR) | 32 | RW | 0000_8002h |
| 10h | Channel Priority (CH0_PRI) | 32 | RW | 0000_0000h |
| 20h | TCD Source Address (TCD0_SADDR) | 32 | RW | See section |
| 24h | TCD Signed Source Address Offset (TCD0_SOFF) | 16 | RW | See section |
| 26h | TCD Transfer Attributes (TCD0_ATTR) | 16 | RW | See section |
| 28h | TCD Transfer Size Without Minor Loop Offsets (TCD0_NBYTES_MLOFFNO) | 32 | RW | See section |
| 28h | TCD Transfer Size with Minor Loop Offsets (TCD0_NBYTES_MLOFFYES) | 32 | RW | See section |
| 2Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD0_SLAST_SDA) | 32 | RW | See section |
| 30h | TCD Destination Address (TCD0_DADDR) | 32 | RW | See section |
| 34h | TCD Signed Destination Address Offset (TCD0_DOFF) | 16 | RW | See section |
| 36h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD0_CITER_ELINKNO) | 16 | RW | See section |
| 36h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD0_CITER_ELINKYES) | 16 | RW | See section |
| 38h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD0_DLAST_SGA) | 32 | RW | See section |
| 3Ch | TCD Control and Status (TCD0_CSR) | 16 | RW | See section |
| 3Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD0_BITER_ELINKNO) | 16 | RW | See section |
| 3Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD0_BITER_ELINKYES) | 16 | RW | See section |
| 4000h | Channel Control and Status (CH1_CSR) | 32 | RW | 0000_0000h |
| 4004h | Channel Error Status (CH1_ES) | 32 | RW | 0000_0000h |
| 4008h | Channel Interrupt Status (CH1_INT) | 32 | RW | 0000_0000h |
| 400Ch | Channel System Bus (CH1_SBR) | 32 | RW | 0000_8002h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 4010h | Channel Priority (CH1_PRI) | 32 | RW | 0000_0000h |
| 4020h | TCD Source Address (TCD1_SADDR) | 32 | RW | See section |
| 4024h | TCD Signed Source Address Offset (TCD1_SOFF) | 16 | RW | See section |
| 4026h | TCD Transfer Attributes (TCD1_ATTR) | 16 | RW | See section |
| 4028h | TCD Transfer Size Without Minor Loop Offsets (TCD1_NBYTES_MLOFFNO) | 32 | RW | See section |
| 4028h | TCD Transfer Size with Minor Loop Offsets (TCD1_NBYTES_MLOFFYES) | 32 | RW | See section |
| 402Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD1_SLAST_SDA) | 32 | RW | See section |
| 4030h | TCD Destination Address (TCD1_DADDR) | 32 | RW | See section |
| 4034h | TCD Signed Destination Address Offset (TCD1_DOFF) | 16 | RW | See section |
| 4036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD1_CITER_ELINKNO) | 16 | RW | See section |
| 4036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD1_CITER_ELINKYES) | 16 | RW | See section |
| 4038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD1_DLAST_SGA) | 32 | RW | See section |
| 403Ch | TCD Control and Status (TCD1_CSR) | 16 | RW | See section |
| 403Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD1_BITER_ELINKNO) | 16 | RW | See section |
| 403Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD1_BITER_ELINKYES) | 16 | RW | See section |
| 8000h | Channel Control and Status (CH2_CSR) | 32 | RW | 0000_0000h |
| 8004h | Channel Error Status (CH2_ES) | 32 | RW | 0000_0000h |
| 8008h | Channel Interrupt Status (CH2_INT) | 32 | RW | 0000_0000h |
| 800Ch | Channel System Bus (CH2_SBR) | 32 | RW | 0000_8002h |
| 8010h | Channel Priority (CH2_PRI) | 32 | RW | 0000_0000h |
| 8020h | TCD Source Address (TCD2_SADDR) | 32 | RW | See section |
| 8024h | TCD Signed Source Address Offset (TCD2_SOFF) | 16 | RW | See section |
| 8026h | TCD Transfer Attributes (TCD2_ATTR) | 16 | RW | See section |
| 8028h | TCD Transfer Size Without Minor Loop Offsets (TCD2_NBYTES_MLOFFNO) | 32 | RW | See section |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 8028h | TCD Transfer Size with Minor Loop Offsets (TCD2_NBYTES_MLOFFYES) | 32 | RW | See section |
| 802Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD2_SLAST_SDA) | 32 | RW | See section |
| 8030h | TCD Destination Address (TCD2_DADDR) | 32 | RW | See section |
| 8034h | TCD Signed Destination Address Offset (TCD2_DOFF) | 16 | RW | See section |
| 8036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD2_CITER_ELINKNO) | 16 | RW | See section |
| 8036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD2_CITER_ELINKYES) | 16 | RW | See section |
| 8038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD2_DLAST_SGA) | 32 | RW | See section |
| 803Ch | TCD Control and Status (TCD2_CSR) | 16 | RW | See section |
| 803Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD2_BITER_ELINKNO) | 16 | RW | See section |
| 803Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD2_BITER_ELINKYES) | 16 | RW | See section |
| C000h | Channel Control and Status (CH3_CSR) | 32 | RW | 0000_0000h |
| C004h | Channel Error Status (CH3_ES) | 32 | RW | 0000_0000h |
| C008h | Channel Interrupt Status (CH3_INT) | 32 | RW | 0000_0000h |
| C00Ch | Channel System Bus (CH3_SBR) | 32 | RW | 0000_8002h |
| C010h | Channel Priority (CH3_PRI) | 32 | RW | 0000_0000h |
| C020h | TCD Source Address (TCD3_SADDR) | 32 | RW | See section |
| C024h | TCD Signed Source Address Offset (TCD3_SOFF) | 16 | RW | See section |
| C026h | TCD Transfer Attributes (TCD3_ATTR) | 16 | RW | See section |
| C028h | TCD Transfer Size Without Minor Loop Offsets (TCD3_NBYTES_MLOFFNO) | 32 | RW | See section |
| C028h | TCD Transfer Size with Minor Loop Offsets (TCD3_NBYTES_MLOFFYES) | 32 | RW | See section |
| C02Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD3_SLAST_SDA) | 32 | RW | See section |
| C030h | TCD Destination Address (TCD3_DADDR) | 32 | RW | See section |
| C034h | TCD Signed Destination Address Offset (TCD3_DOFF) | 16 | RW | See section |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| C036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD3_CITER_ELINKNO) | 16 | RW | See section |
| C036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD3_CITER_ELINKYES) | 16 | RW | See section |
| C038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD3_DLAST_SGA) | 32 | RW | See section |
| C03Ch | TCD Control and Status (TCD3_CSR) | 16 | RW | See section |
| C03Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD3_BITER_ELINKNO) | 16 | RW | See section |
| C03Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD3_BITER_ELINKYES) | 16 | RW | See section |
| 1_0000h | Channel Control and Status (CH4_CSR) | 32 | RW | 0000_0000h |
| 1_0004h | Channel Error Status (CH4_ES) | 32 | RW | 0000_0000h |
| 1_0008h | Channel Interrupt Status (CH4_INT) | 32 | RW | 0000_0000h |
| 1_000Ch | Channel System Bus (CH4_SBR) | 32 | RW | 0000_8002h |
| 1_0010h | Channel Priority (CH4_PRI) | 32 | RW | 0000_0000h |
| 1_0020h | TCD Source Address (TCD4_SADDR) | 32 | RW | See section |
| 1_0024h | TCD Signed Source Address Offset (TCD4_SOFF) | 16 | RW | See section |
| 1_0026h | TCD Transfer Attributes (TCD4_ATTR) | 16 | RW | See section |
| 1_0028h | TCD Transfer Size Without Minor Loop Offsets (TCD4_NBYTES_MLOFFNO) | 32 | RW | See section |
| 1_0028h | TCD Transfer Size with Minor Loop Offsets (TCD4_NBYTES_MLOFFYES) | 32 | RW | See section |
| 1_002Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD4_SLAST_SDA) | 32 | RW | See section |
| 1_0030h | TCD Destination Address (TCD4_DADDR) | 32 | RW | See section |
| 1_0034h | TCD Signed Destination Address Offset (TCD4_DOFF) | 16 | RW | See section |
| 1_0036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD4_CITER_ELINKNO) | 16 | RW | See section |
| 1_0036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD4_CITER_ELINKYES) | 16 | RW | See section |
| 1_0038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD4_DLAST_SGA) | 32 | RW | See section |
| 1_003Ch | TCD Control and Status (TCD4_CSR) | 16 | RW | See section |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1_003Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD4_BITER_ELINKNO) | 16 | RW | See section |
| 1_003Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD4_BITER_ELINKYES) | 16 | RW | See section |
| 1_4000h | Channel Control and Status (CH5_CSR) | 32 | RW | 0000_0000h |
| 1_4004h | Channel Error Status (CH5_ES) | 32 | RW | 0000_0000h |
| 1_4008h | Channel Interrupt Status (CH5_INT) | 32 | RW | 0000_0000h |
| 1_400Ch | Channel System Bus (CH5_SBR) | 32 | RW | 0000_8002h |
| 1_4010h | Channel Priority (CH5_PRI) | 32 | RW | 0000_0000h |
| 1_4020h | TCD Source Address (TCD5_SADDR) | 32 | RW | See section |
| 1_4024h | TCD Signed Source Address Offset (TCD5_SOFF) | 16 | RW | See section |
| 1_4026h | TCD Transfer Attributes (TCD5_ATTR) | 16 | RW | See section |
| 1_4028h | TCD Transfer Size Without Minor Loop Offsets (TCD5_NBYTES_MLOFFNO) | 32 | RW | See section |
| 1_4028h | TCD Transfer Size with Minor Loop Offsets (TCD5_NBYTES_MLOFFYES) | 32 | RW | See section |
| 1_402Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD5_SLAST_SDA) | 32 | RW | See section |
| 1_4030h | TCD Destination Address (TCD5_DADDR) | 32 | RW | See section |
| 1_4034h | TCD Signed Destination Address Offset (TCD5_DOFF) | 16 | RW | See section |
| 1_4036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD5_CITER_ELINKNO) | 16 | RW | See section |
| 1_4036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD5_CITER_ELINKYES) | 16 | RW | See section |
| 1_4038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD5_DLAST_SGA) | 32 | RW | See section |
| 1_403Ch | TCD Control and Status (TCD5_CSR) | 16 | RW | See section |
| 1_403Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD5_BITER_ELINKNO) | 16 | RW | See section |
| 1_403Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD5_BITER_ELINKYES) | 16 | RW | See section |
| 1_8000h | Channel Control and Status (CH6_CSR) | 32 | RW | 0000_0000h |
| 1_8004h | Channel Error Status (CH6_ES) | 32 | RW | 0000_0000h |
| 1_8008h | Channel Interrupt Status (CH6_INT) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1_800Ch | Channel System Bus (CH6_SBR) | 32 | RW | 0000_8002h |
| 1_8010h | Channel Priority (CH6_PRI) | 32 | RW | 0000_0000h |
| 1_8020h | TCD Source Address (TCD6_SADDR) | 32 | RW | See section |
| 1_8024h | TCD Signed Source Address Offset (TCD6_SOFF) | 16 | RW | See section |
| 1_8026h | TCD Transfer Attributes (TCD6_ATTR) | 16 | RW | See section |
| 1_8028h | TCD Transfer Size Without Minor Loop Offsets (TCD6_NBYTES_MLOFFNO) | 32 | RW | See section |
| 1_8028h | TCD Transfer Size with Minor Loop Offsets (TCD6_NBYTES_MLOFFYES) | 32 | RW | See section |
| 1_802Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD6_SLAST_SDA) | 32 | RW | See section |
| 1_8030h | TCD Destination Address (TCD6_DADDR) | 32 | RW | See section |
| 1_8034h | TCD Signed Destination Address Offset (TCD6_DOFF) | 16 | RW | See section |
| 1_8036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD6_CITER_ELINKNO) | 16 | RW | See section |
| 1_8036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD6_CITER_ELINKYES) | 16 | RW | See section |
| 1_8038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD6_DLAST_SGA) | 32 | RW | See section |
| 1_803Ch | TCD Control and Status (TCD6_CSR) | 16 | RW | See section |
| 1_803Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD6_BITER_ELINKNO) | 16 | RW | See section |
| 1_803Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD6_BITER_ELINKYES) | 16 | RW | See section |
| 1_C000h | Channel Control and Status (CH7_CSR) | 32 | RW | 0000_0000h |
| 1_C004h | Channel Error Status (CH7_ES) | 32 | RW | 0000_0000h |
| 1_C008h | Channel Interrupt Status (CH7_INT) | 32 | RW | 0000_0000h |
| 1_C00Ch | Channel System Bus (CH7_SBR) | 32 | RW | 0000_8002h |
| 1_C010h | Channel Priority (CH7_PRI) | 32 | RW | 0000_0000h |
| 1_C020h | TCD Source Address (TCD7_SADDR) | 32 | RW | See section |
| 1_C024h | TCD Signed Source Address Offset (TCD7_SOFF) | 16 | RW | See section |
| 1_C026h | TCD Transfer Attributes (TCD7_ATTR) | 16 | RW | See section |
| 1_C028h | TCD Transfer Size Without Minor Loop Offsets (TCD7_NBYTES_MLOFFNO) | 32 | RW | See section |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1_C028h | TCD Transfer Size with Minor Loop Offsets (TCD7_NBYTES_MLOFFYES) | 32 | RW | See section |
| 1_C02Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD7_SLAST_SDA) | 32 | RW | See section |
| 1_C030h | TCD Destination Address (TCD7_DADDR) | 32 | RW | See section |
| 1_C034h | TCD Signed Destination Address Offset (TCD7_DOFF) | 16 | RW | See section |
| 1_C036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD7_CITER_ELINKNO) | 16 | RW | See section |
| 1_C036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD7_CITER_ELINKYES) | 16 | RW | See section |
| 1_C038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD7_DLAST_SGA) | 32 | RW | See section |
| 1_C03Ch | TCD Control and Status (TCD7_CSR) | 16 | RW | See section |
| 1_C03Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD7_BITER_ELINKNO) | 16 | RW | See section |
| 1_C03Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD7_BITER_ELINKYES) | 16 | RW | See section |
| 2_0000h | Channel Control and Status (CH8_CSR) | 32 | RW | 0000_0000h |
| 2_0004h | Channel Error Status (CH8_ES) | 32 | RW | 0000_0000h |
| 2_0008h | Channel Interrupt Status (CH8_INT) | 32 | RW | 0000_0000h |
| 2_000Ch | Channel System Bus (CH8_SBR) | 32 | RW | 0000_8002h |
| 2_0010h | Channel Priority (CH8_PRI) | 32 | RW | 0000_0000h |
| 2_0020h | TCD Source Address (TCD8_SADDR) | 32 | RW | See section |
| 2_0024h | TCD Signed Source Address Offset (TCD8_SOFF) | 16 | RW | See section |
| 2_0026h | TCD Transfer Attributes (TCD8_ATTR) | 16 | RW | See section |
| 2_0028h | TCD Transfer Size Without Minor Loop Offsets (TCD8_NBYTES_MLOFFNO) | 32 | RW | See section |
| 2_0028h | TCD Transfer Size with Minor Loop Offsets (TCD8_NBYTES_MLOFFYES) | 32 | RW | See section |
| 2_002Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD8_SLAST_SDA) | 32 | RW | See section |
| 2_0030h | TCD Destination Address (TCD8_DADDR) | 32 | RW | See section |
| 2_0034h | TCD Signed Destination Address Offset (TCD8_DOFF) | 16 | RW | See section |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 2_0036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD8_CITER_ELINKNO) | 16 | RW | See section |
| 2_0036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD8_CITER_ELINKYES) | 16 | RW | See section |
| 2_0038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD8_DLAST_SGA) | 32 | RW | See section |
| 2_003Ch | TCD Control and Status (TCD8_CSR) | 16 | RW | See section |
| 2_003Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD8_BITER_ELINKNO) | 16 | RW | See section |
| 2_003Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD8_BITER_ELINKYES) | 16 | RW | See section |
| 2_4000h | Channel Control and Status (CH9_CSR) | 32 | RW | 0000_0000h |
| 2_4004h | Channel Error Status (CH9_ES) | 32 | RW | 0000_0000h |
| 2_4008h | Channel Interrupt Status (CH9_INT) | 32 | RW | 0000_0000h |
| 2_400Ch | Channel System Bus (CH9_SBR) | 32 | RW | 0000_8002h |
| 2_4010h | Channel Priority (CH9_PRI) | 32 | RW | 0000_0000h |
| 2_4020h | TCD Source Address (TCD9_SADDR) | 32 | RW | See section |
| 2_4024h | TCD Signed Source Address Offset (TCD9_SOFF) | 16 | RW | See section |
| 2_4026h | TCD Transfer Attributes (TCD9_ATTR) | 16 | RW | See section |
| 2_4028h | TCD Transfer Size Without Minor Loop Offsets (TCD9_NBYTES_MLOFFNO) | 32 | RW | See section |
| 2_4028h | TCD Transfer Size with Minor Loop Offsets (TCD9_NBYTES_MLOFFYES) | 32 | RW | See section |
| 2_402Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD9_SLAST_SDA) | 32 | RW | See section |
| 2_4030h | TCD Destination Address (TCD9_DADDR) | 32 | RW | See section |
| 2_4034h | TCD Signed Destination Address Offset (TCD9_DOFF) | 16 | RW | See section |
| 2_4036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD9_CITER_ELINKNO) | 16 | RW | See section |
| 2_4036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD9_CITER_ELINKYES) | 16 | RW | See section |
| 2_4038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD9_DLAST_SGA) | 32 | RW | See section |
| 2_403Ch | TCD Control and Status (TCD9_CSR) | 16 | RW | See section |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 2_403Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD9_BITER_ELINKNO) | 16 | RW | See section |
| 2_403Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD9_BITER_ELINKYES) | 16 | RW | See section |
| 2_8000h | Channel Control and Status (CH10_CSR) | 32 | RW | 0000_0000h |
| 2_8004h | Channel Error Status (CH10_ES) | 32 | RW | 0000_0000h |
| 2_8008h | Channel Interrupt Status (CH10_INT) | 32 | RW | 0000_0000h |
| 2_800Ch | Channel System Bus (CH10_SBR) | 32 | RW | 0000_8002h |
| 2_8010h | Channel Priority (CH10_PRI) | 32 | RW | 0000_0000h |
| 2_8020h | TCD Source Address (TCD10_SADDR) | 32 | RW | See section |
| 2_8024h | TCD Signed Source Address Offset (TCD10_SOFF) | 16 | RW | See section |
| 2_8026h | TCD Transfer Attributes (TCD10_ATTR) | 16 | RW | See section |
| 2_8028h | TCD Transfer Size Without Minor Loop Offsets (TCD10_NBYTES_MLOFFNO) | 32 | RW | See section |
| 2_8028h | TCD Transfer Size with Minor Loop Offsets (TCD10_NBYTES_MLOFFYES) | 32 | RW | See section |
| 2_802Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD10_SLAST_SDA) | 32 | RW | See section |
| 2_8030h | TCD Destination Address (TCD10_DADDR) | 32 | RW | See section |
| 2_8034h | TCD Signed Destination Address Offset (TCD10_DOFF) | 16 | RW | See section |
| 2_8036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD10_CITER_ELINKNO) | 16 | RW | See section |
| 2_8036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD10_CITER_ELINKYES) | 16 | RW | See section |
| 2_8038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD10_DLAST_SGA) | 32 | RW | See section |
| 2_803Ch | TCD Control and Status (TCD10_CSR) | 16 | RW | See section |
| 2_803Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD10_BITER_ELINKNO) | 16 | RW | See section |
| 2_803Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD10_BITER_ELINKYES) | 16 | RW | See section |
| 2_C000h | Channel Control and Status (CH11_CSR) | 32 | RW | 0000_0000h |
| 2_C004h | Channel Error Status (CH11_ES) | 32 | RW | 0000_0000h |
| 2_C008h | Channel Interrupt Status (CH11_INT) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 2_C00Ch | Channel System Bus (CH11_SBR) | 32 | RW | 0000_8002h |
| 2_C010h | Channel Priority (CH11_PRI) | 32 | RW | 0000_0000h |
| 2_C020h | TCD Source Address (TCD11_SADDR) | 32 | RW | See section |
| 2_C024h | TCD Signed Source Address Offset (TCD11_SOFF) | 16 | RW | See section |
| 2_C026h | TCD Transfer Attributes (TCD11_ATTR) | 16 | RW | See section |
| 2_C028h | TCD Transfer Size Without Minor Loop Offsets (TCD11_NBYTES_MLOFFNO) | 32 | RW | See section |
| 2_C028h | TCD Transfer Size with Minor Loop Offsets (TCD11_NBYTES_MLOFFYES) | 32 | RW | See section |
| 2_C02Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD11_SLAST_SDA) | 32 | RW | See section |
| 2_C030h | TCD Destination Address (TCD11_DADDR) | 32 | RW | See section |
| 2_C034h | TCD Signed Destination Address Offset (TCD11_DOFF) | 16 | RW | See section |
| 2_C036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD11_CITER_ELINKNO) | 16 | RW | See section |
| 2_C036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD11_CITER_ELINKYES) | 16 | RW | See section |
| 2_C038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD11_DLAST_SGA) | 32 | RW | See section |
| 2_C03Ch | TCD Control and Status (TCD11_CSR) | 16 | RW | See section |
| 2_C03Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD11_BITER_ELINKNO) | 16 | RW | See section |
| 2_C03Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD11_BITER_ELINKYES) | 16 | RW | See section |
| 20_0000h | Channel Control and Status (CH12_CSR) | 32 | RW | 0000_0000h |
| 20_0004h | Channel Error Status (CH12_ES) | 32 | RW | 0000_0000h |
| 20_0008h | Channel Interrupt Status (CH12_INT) | 32 | RW | 0000_0000h |
| 20_000Ch | Channel System Bus (CH12_SBR) | 32 | RW | 0000_8002h |
| 20_0010h | Channel Priority (CH12_PRI) | 32 | RW | 0000_0000h |
| 20_0020h | TCD Source Address (TCD12_SADDR) | 32 | RW | See section |
| 20_0024h | TCD Signed Source Address Offset (TCD12_SOFF) | 16 | RW | See section |
| 20_0026h | TCD Transfer Attributes (TCD12_ATTR) | 16 | RW | See section |
| 20_0028h | TCD Transfer Size Without Minor Loop Offsets (TCD12_NBYTES_MLOFFNO) | 32 | RW | See section |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 20_0028h | TCD Transfer Size with Minor Loop Offsets (TCD12_NBYTES_MLOFFYES) | 32 | RW | See section |
| 20_002Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD12_SLAST_SDA) | 32 | RW | See section |
| 20_0030h | TCD Destination Address (TCD12_DADDR) | 32 | RW | See section |
| 20_0034h | TCD Signed Destination Address Offset (TCD12_DOFF) | 16 | RW | See section |
| 20_0036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD12_CITER_ELINKNO) | 16 | RW | See section |
| 20_0036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD12_CITER_ELINKYES) | 16 | RW | See section |
| 20_0038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD12_DLAST_SGA) | 32 | RW | See section |
| 20_003Ch | TCD Control and Status (TCD12_CSR) | 16 | RW | See section |
| 20_003Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD12_BITER_ELINKNO) | 16 | RW | See section |
| 20_003Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD12_BITER_ELINKYES) | 16 | RW | See section |
| 20_4000h | Channel Control and Status (CH13_CSR) | 32 | RW | 0000_0000h |
| 20_4004h | Channel Error Status (CH13_ES) | 32 | RW | 0000_0000h |
| 20_4008h | Channel Interrupt Status (CH13_INT) | 32 | RW | 0000_0000h |
| 20_400Ch | Channel System Bus (CH13_SBR) | 32 | RW | 0000_8002h |
| 20_4010h | Channel Priority (CH13_PRI) | 32 | RW | 0000_0000h |
| 20_4020h | TCD Source Address (TCD13_SADDR) | 32 | RW | See section |
| 20_4024h | TCD Signed Source Address Offset (TCD13_SOFF) | 16 | RW | See section |
| 20_4026h | TCD Transfer Attributes (TCD13_ATTR) | 16 | RW | See section |
| 20_4028h | TCD Transfer Size Without Minor Loop Offsets (TCD13_NBYTES_MLOFFNO) | 32 | RW | See section |
| 20_4028h | TCD Transfer Size with Minor Loop Offsets (TCD13_NBYTES_MLOFFYES) | 32 | RW | See section |
| 20_402Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD13_SLAST_SDA) | 32 | RW | See section |
| 20_4030h | TCD Destination Address (TCD13_DADDR) | 32 | RW | See section |
| 20_4034h | TCD Signed Destination Address Offset (TCD13_DOFF) | 16 | RW | See section |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 20_4036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD13_CITER_ELINKNO) | 16 | RW | See section |
| 20_4036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD13_CITER_ELINKYES) | 16 | RW | See section |
| 20_4038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD13_DLAST_SGA) | 32 | RW | See section |
| 20_403Ch | TCD Control and Status (TCD13_CSR) | 16 | RW | See section |
| 20_403Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD13_BITER_ELINKNO) | 16 | RW | See section |
| 20_403Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD13_BITER_ELINKYES) | 16 | RW | See section |
| 20_8000h | Channel Control and Status (CH14_CSR) | 32 | RW | 0000_0000h |
| 20_8004h | Channel Error Status (CH14_ES) | 32 | RW | 0000_0000h |
| 20_8008h | Channel Interrupt Status (CH14_INT) | 32 | RW | 0000_0000h |
| 20_800Ch | Channel System Bus (CH14_SBR) | 32 | RW | 0000_8002h |
| 20_8010h | Channel Priority (CH14_PRI) | 32 | RW | 0000_0000h |
| 20_8020h | TCD Source Address (TCD14_SADDR) | 32 | RW | See section |
| 20_8024h | TCD Signed Source Address Offset (TCD14_SOFF) | 16 | RW | See section |
| 20_8026h | TCD Transfer Attributes (TCD14_ATTR) | 16 | RW | See section |
| 20_8028h | TCD Transfer Size Without Minor Loop Offsets (TCD14_NBYTES_MLOFFNO) | 32 | RW | See section |
| 20_8028h | TCD Transfer Size with Minor Loop Offsets (TCD14_NBYTES_MLOFFYES) | 32 | RW | See section |
| 20_802Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD14_SLAST_SDA) | 32 | RW | See section |
| 20_8030h | TCD Destination Address (TCD14_DADDR) | 32 | RW | See section |
| 20_8034h | TCD Signed Destination Address Offset (TCD14_DOFF) | 16 | RW | See section |
| 20_8036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD14_CITER_ELINKNO) | 16 | RW | See section |
| 20_8036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD14_CITER_ELINKYES) | 16 | RW | See section |
| 20_8038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD14_DLAST_SGA) | 32 | RW | See section |
| 20_803Ch | TCD Control and Status (TCD14_CSR) | 16 | RW | See section |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 20_803Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD14_BITER_ELINKNO) | 16 | RW | See section |
| 20_803Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD14_BITER_ELINKYES) | 16 | RW | See section |
| 20_C000h | Channel Control and Status (CH15_CSR) | 32 | RW | 0000_0000h |
| 20_C004h | Channel Error Status (CH15_ES) | 32 | RW | 0000_0000h |
| 20_C008h | Channel Interrupt Status (CH15_INT) | 32 | RW | 0000_0000h |
| 20_C00Ch | Channel System Bus (CH15_SBR) | 32 | RW | 0000_8002h |
| 20_C010h | Channel Priority (CH15_PRI) | 32 | RW | 0000_0000h |
| 20_C020h | TCD Source Address (TCD15_SADDR) | 32 | RW | See section |
| 20_C024h | TCD Signed Source Address Offset (TCD15_SOFF) | 16 | RW | See section |
| 20_C026h | TCD Transfer Attributes (TCD15_ATTR) | 16 | RW | See section |
| 20_C028h | TCD Transfer Size Without Minor Loop Offsets (TCD15_NBYTES_MLOFFNO) | 32 | RW | See section |
| 20_C028h | TCD Transfer Size with Minor Loop Offsets (TCD15_NBYTES_MLOFFYES) | 32 | RW | See section |
| 20_C02Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD15_SLAST_SDA) | 32 | RW | See section |
| 20_C030h | TCD Destination Address (TCD15_DADDR) | 32 | RW | See section |
| 20_C034h | TCD Signed Destination Address Offset (TCD15_DOFF) | 16 | RW | See section |
| 20_C036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD15_CITER_ELINKNO) | 16 | RW | See section |
| 20_C036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD15_CITER_ELINKYES) | 16 | RW | See section |
| 20_C038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD15_DLAST_SGA) | 32 | RW | See section |
| 20_C03Ch | TCD Control and Status (TCD15_CSR) | 16 | RW | See section |
| 20_C03Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD15_BITER_ELINKNO) | 16 | RW | See section |
| 20_C03Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD15_BITER_ELINKYES) | 16 | RW | See section |
| 21_0000h | Channel Control and Status (CH16_CSR) | 32 | RW | 0000_0000h |
| 21_0004h | Channel Error Status (CH16_ES) | 32 | RW | 0000_0000h |
| 21_0008h | Channel Interrupt Status (CH16_INT) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 21_000Ch | Channel System Bus (CH16_SBR) | 32 | RW | 0000_8002h |
| 21_0010h | Channel Priority (CH16_PRI) | 32 | RW | 0000_0000h |
| 21_0020h | TCD Source Address (TCD16_SADDR) | 32 | RW | See section |
| 21_0024h | TCD Signed Source Address Offset (TCD16_SOFF) | 16 | RW | See section |
| 21_0026h | TCD Transfer Attributes (TCD16_ATTR) | 16 | RW | See section |
| 21_0028h | TCD Transfer Size Without Minor Loop Offsets (TCD16_NBYTES_MLOFFNO) | 32 | RW | See section |
| 21_0028h | TCD Transfer Size with Minor Loop Offsets (TCD16_NBYTES_MLOFFYES) | 32 | RW | See section |
| 21_002Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD16_SLAST_SDA) | 32 | RW | See section |
| 21_0030h | TCD Destination Address (TCD16_DADDR) | 32 | RW | See section |
| 21_0034h | TCD Signed Destination Address Offset (TCD16_DOFF) | 16 | RW | See section |
| 21_0036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD16_CITER_ELINKNO) | 16 | RW | See section |
| 21_0036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD16_CITER_ELINKYES) | 16 | RW | See section |
| 21_0038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD16_DLAST_SGA) | 32 | RW | See section |
| 21_003Ch | TCD Control and Status (TCD16_CSR) | 16 | RW | See section |
| 21_003Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD16_BITER_ELINKNO) | 16 | RW | See section |
| 21_003Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD16_BITER_ELINKYES) | 16 | RW | See section |
| 21_4000h | Channel Control and Status (CH17_CSR) | 32 | RW | 0000_0000h |
| 21_4004h | Channel Error Status (CH17_ES) | 32 | RW | 0000_0000h |
| 21_4008h | Channel Interrupt Status (CH17_INT) | 32 | RW | 0000_0000h |
| 21_400Ch | Channel System Bus (CH17_SBR) | 32 | RW | 0000_8002h |
| 21_4010h | Channel Priority (CH17_PRI) | 32 | RW | 0000_0000h |
| 21_4020h | TCD Source Address (TCD17_SADDR) | 32 | RW | See section |
| 21_4024h | TCD Signed Source Address Offset (TCD17_SOFF) | 16 | RW | See section |
| 21_4026h | TCD Transfer Attributes (TCD17_ATTR) | 16 | RW | See section |
| 21_4028h | TCD Transfer Size Without Minor Loop Offsets (TCD17_NBYTES_MLOFFNO) | 32 | RW | See section |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 21_4028h | TCD Transfer Size with Minor Loop Offsets (TCD17_NBYTES_MLOFFYES) | 32 | RW | See section |
| 21_402Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD17_SLAST_SDA) | 32 | RW | See section |
| 21_4030h | TCD Destination Address (TCD17_DADDR) | 32 | RW | See section |
| 21_4034h | TCD Signed Destination Address Offset (TCD17_DOFF) | 16 | RW | See section |
| 21_4036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD17_CITER_ELINKNO) | 16 | RW | See section |
| 21_4036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD17_CITER_ELINKYES) | 16 | RW | See section |
| 21_4038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD17_DLAST_SGA) | 32 | RW | See section |
| 21_403Ch | TCD Control and Status (TCD17_CSR) | 16 | RW | See section |
| 21_403Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD17_BITER_ELINKNO) | 16 | RW | See section |
| 21_403Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD17_BITER_ELINKYES) | 16 | RW | See section |
| 21_8000h | Channel Control and Status (CH18_CSR) | 32 | RW | 0000_0000h |
| 21_8004h | Channel Error Status (CH18_ES) | 32 | RW | 0000_0000h |
| 21_8008h | Channel Interrupt Status (CH18_INT) | 32 | RW | 0000_0000h |
| 21_800Ch | Channel System Bus (CH18_SBR) | 32 | RW | 0000_8002h |
| 21_8010h | Channel Priority (CH18_PRI) | 32 | RW | 0000_0000h |
| 21_8020h | TCD Source Address (TCD18_SADDR) | 32 | RW | See section |
| 21_8024h | TCD Signed Source Address Offset (TCD18_SOFF) | 16 | RW | See section |
| 21_8026h | TCD Transfer Attributes (TCD18_ATTR) | 16 | RW | See section |
| 21_8028h | TCD Transfer Size Without Minor Loop Offsets (TCD18_NBYTES_MLOFFNO) | 32 | RW | See section |
| 21_8028h | TCD Transfer Size with Minor Loop Offsets (TCD18_NBYTES_MLOFFYES) | 32 | RW | See section |
| 21_802Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD18_SLAST_SDA) | 32 | RW | See section |
| 21_8030h | TCD Destination Address (TCD18_DADDR) | 32 | RW | See section |
| 21_8034h | TCD Signed Destination Address Offset (TCD18_DOFF) | 16 | RW | See section |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 21_8036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD18_CITER_ELINKNO) | 16 | RW | See section |
| 21_8036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD18_CITER_ELINKYES) | 16 | RW | See section |
| 21_8038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD18_DLAST_SGA) | 32 | RW | See section |
| 21_803Ch | TCD Control and Status (TCD18_CSR) | 16 | RW | See section |
| 21_803Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD18_BITER_ELINKNO) | 16 | RW | See section |
| 21_803Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD18_BITER_ELINKYES) | 16 | RW | See section |
| 21_C000h | Channel Control and Status (CH19_CSR) | 32 | RW | 0000_0000h |
| 21_C004h | Channel Error Status (CH19_ES) | 32 | RW | 0000_0000h |
| 21_C008h | Channel Interrupt Status (CH19_INT) | 32 | RW | 0000_0000h |
| 21_C00Ch | Channel System Bus (CH19_SBR) | 32 | RW | 0000_8002h |
| 21_C010h | Channel Priority (CH19_PRI) | 32 | RW | 0000_0000h |
| 21_C020h | TCD Source Address (TCD19_SADDR) | 32 | RW | See section |
| 21_C024h | TCD Signed Source Address Offset (TCD19_SOFF) | 16 | RW | See section |
| 21_C026h | TCD Transfer Attributes (TCD19_ATTR) | 16 | RW | See section |
| 21_C028h | TCD Transfer Size Without Minor Loop Offsets (TCD19_NBYTES_MLOFFNO) | 32 | RW | See section |
| 21_C028h | TCD Transfer Size with Minor Loop Offsets (TCD19_NBYTES_MLOFFYES) | 32 | RW | See section |
| 21_C02Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD19_SLAST_SDA) | 32 | RW | See section |
| 21_C030h | TCD Destination Address (TCD19_DADDR) | 32 | RW | See section |
| 21_C034h | TCD Signed Destination Address Offset (TCD19_DOFF) | 16 | RW | See section |
| 21_C036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD19_CITER_ELINKNO) | 16 | RW | See section |
| 21_C036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD19_CITER_ELINKYES) | 16 | RW | See section |
| 21_C038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD19_DLAST_SGA) | 32 | RW | See section |
| 21_C03Ch | TCD Control and Status (TCD19_CSR) | 16 | RW | See section |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 21_C03Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD19_BITER_ELINKNO) | 16 | RW | See section |
| 21_C03Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD19_BITER_ELINKYES) | 16 | RW | See section |
| 22_0000h | Channel Control and Status (CH20_CSR) | 32 | RW | 0000_0000h |
| 22_0004h | Channel Error Status (CH20_ES) | 32 | RW | 0000_0000h |
| 22_0008h | Channel Interrupt Status (CH20_INT) | 32 | RW | 0000_0000h |
| 22_000Ch | Channel System Bus (CH20_SBR) | 32 | RW | 0000_8002h |
| 22_0010h | Channel Priority (CH20_PRI) | 32 | RW | 0000_0000h |
| 22_0020h | TCD Source Address (TCD20_SADDR) | 32 | RW | See section |
| 22_0024h | TCD Signed Source Address Offset (TCD20_SOFF) | 16 | RW | See section |
| 22_0026h | TCD Transfer Attributes (TCD20_ATTR) | 16 | RW | See section |
| 22_0028h | TCD Transfer Size Without Minor Loop Offsets (TCD20_NBYTES_MLOFFNO) | 32 | RW | See section |
| 22_0028h | TCD Transfer Size with Minor Loop Offsets (TCD20_NBYTES_MLOFFYES) | 32 | RW | See section |
| 22_002Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD20_SLAST_SDA) | 32 | RW | See section |
| 22_0030h | TCD Destination Address (TCD20_DADDR) | 32 | RW | See section |
| 22_0034h | TCD Signed Destination Address Offset (TCD20_DOFF) | 16 | RW | See section |
| 22_0036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD20_CITER_ELINKNO) | 16 | RW | See section |
| 22_0036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD20_CITER_ELINKYES) | 16 | RW | See section |
| 22_0038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD20_DLAST_SGA) | 32 | RW | See section |
| 22_003Ch | TCD Control and Status (TCD20_CSR) | 16 | RW | See section |
| 22_003Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD20_BITER_ELINKNO) | 16 | RW | See section |
| 22_003Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD20_BITER_ELINKYES) | 16 | RW | See section |
| 22_4000h | Channel Control and Status (CH21_CSR) | 32 | RW | 0000_0000h |
| 22_4004h | Channel Error Status (CH21_ES) | 32 | RW | 0000_0000h |
| 22_4008h | Channel Interrupt Status (CH21_INT) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 22_400Ch | Channel System Bus (CH21_SBR) | 32 | RW | 0000_8002h |
| 22_4010h | Channel Priority (CH21_PRI) | 32 | RW | 0000_0000h |
| 22_4020h | TCD Source Address (TCD21_SADDR) | 32 | RW | See section |
| 22_4024h | TCD Signed Source Address Offset (TCD21_SOFF) | 16 | RW | See section |
| 22_4026h | TCD Transfer Attributes (TCD21_ATTR) | 16 | RW | See section |
| 22_4028h | TCD Transfer Size Without Minor Loop Offsets (TCD21_NBYTES_MLOFFNO) | 32 | RW | See section |
| 22_4028h | TCD Transfer Size with Minor Loop Offsets (TCD21_NBYTES_MLOFFYES) | 32 | RW | See section |
| 22_402Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD21_SLAST_SDA) | 32 | RW | See section |
| 22_4030h | TCD Destination Address (TCD21_DADDR) | 32 | RW | See section |
| 22_4034h | TCD Signed Destination Address Offset (TCD21_DOFF) | 16 | RW | See section |
| 22_4036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD21_CITER_ELINKNO) | 16 | RW | See section |
| 22_4036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD21_CITER_ELINKYES) | 16 | RW | See section |
| 22_4038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD21_DLAST_SGA) | 32 | RW | See section |
| 22_403Ch | TCD Control and Status (TCD21_CSR) | 16 | RW | See section |
| 22_403Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD21_BITER_ELINKNO) | 16 | RW | See section |
| 22_403Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD21_BITER_ELINKYES) | 16 | RW | See section |
| 22_8000h | Channel Control and Status (CH22_CSR) | 32 | RW | 0000_0000h |
| 22_8004h | Channel Error Status (CH22_ES) | 32 | RW | 0000_0000h |
| 22_8008h | Channel Interrupt Status (CH22_INT) | 32 | RW | 0000_0000h |
| 22_800Ch | Channel System Bus (CH22_SBR) | 32 | RW | 0000_8002h |
| 22_8010h | Channel Priority (CH22_PRI) | 32 | RW | 0000_0000h |
| 22_8020h | TCD Source Address (TCD22_SADDR) | 32 | RW | See section |
| 22_8024h | TCD Signed Source Address Offset (TCD22_SOFF) | 16 | RW | See section |
| 22_8026h | TCD Transfer Attributes (TCD22_ATTR) | 16 | RW | See section |
| 22_8028h | TCD Transfer Size Without Minor Loop Offsets (TCD22_NBYTES_MLOFFNO) | 32 | RW | See section |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 22_8028h | TCD Transfer Size with Minor Loop Offsets (TCD22_NBYTES_MLOFFYES) | 32 | RW | See section |
| 22_802Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD22_SLAST_SDA) | 32 | RW | See section |
| 22_8030h | TCD Destination Address (TCD22_DADDR) | 32 | RW | See section |
| 22_8034h | TCD Signed Destination Address Offset (TCD22_DOFF) | 16 | RW | See section |
| 22_8036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD22_CITER_ELINKNO) | 16 | RW | See section |
| 22_8036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD22_CITER_ELINKYES) | 16 | RW | See section |
| 22_8038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD22_DLAST_SGA) | 32 | RW | See section |
| 22_803Ch | TCD Control and Status (TCD22_CSR) | 16 | RW | See section |
| 22_803Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD22_BITER_ELINKNO) | 16 | RW | See section |
| 22_803Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD22_BITER_ELINKYES) | 16 | RW | See section |
| 22_C000h | Channel Control and Status (CH23_CSR) | 32 | RW | 0000_0000h |
| 22_C004h | Channel Error Status (CH23_ES) | 32 | RW | 0000_0000h |
| 22_C008h | Channel Interrupt Status (CH23_INT) | 32 | RW | 0000_0000h |
| 22_C00Ch | Channel System Bus (CH23_SBR) | 32 | RW | 0000_8002h |
| 22_C010h | Channel Priority (CH23_PRI) | 32 | RW | 0000_0000h |
| 22_C020h | TCD Source Address (TCD23_SADDR) | 32 | RW | See section |
| 22_C024h | TCD Signed Source Address Offset (TCD23_SOFF) | 16 | RW | See section |
| 22_C026h | TCD Transfer Attributes (TCD23_ATTR) | 16 | RW | See section |
| 22_C028h | TCD Transfer Size Without Minor Loop Offsets (TCD23_NBYTES_MLOFFNO) | 32 | RW | See section |
| 22_C028h | TCD Transfer Size with Minor Loop Offsets (TCD23_NBYTES_MLOFFYES) | 32 | RW | See section |
| 22_C02Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD23_SLAST_SDA) | 32 | RW | See section |
| 22_C030h | TCD Destination Address (TCD23_DADDR) | 32 | RW | See section |
| 22_C034h | TCD Signed Destination Address Offset (TCD23_DOFF) | 16 | RW | See section |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 22_C036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD23_CITER_ELINKNO) | 16 | RW | See section |
| 22_C036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD23_CITER_ELINKYES) | 16 | RW | See section |
| 22_C038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD23_DLAST_SGA) | 32 | RW | See section |
| 22_C03Ch | TCD Control and Status (TCD23_CSR) | 16 | RW | See section |
| 22_C03Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD23_BITER_ELINKNO) | 16 | RW | See section |
| 22_C03Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD23_BITER_ELINKYES) | 16 | RW | See section |
| 23_0000h | Channel Control and Status (CH24_CSR) | 32 | RW | 0000_0000h |
| 23_0004h | Channel Error Status (CH24_ES) | 32 | RW | 0000_0000h |
| 23_0008h | Channel Interrupt Status (CH24_INT) | 32 | RW | 0000_0000h |
| 23_000Ch | Channel System Bus (CH24_SBR) | 32 | RW | 0000_8002h |
| 23_0010h | Channel Priority (CH24_PRI) | 32 | RW | 0000_0000h |
| 23_0020h | TCD Source Address (TCD24_SADDR) | 32 | RW | See section |
| 23_0024h | TCD Signed Source Address Offset (TCD24_SOFF) | 16 | RW | See section |
| 23_0026h | TCD Transfer Attributes (TCD24_ATTR) | 16 | RW | See section |
| 23_0028h | TCD Transfer Size Without Minor Loop Offsets (TCD24_NBYTES_MLOFFNO) | 32 | RW | See section |
| 23_0028h | TCD Transfer Size with Minor Loop Offsets (TCD24_NBYTES_MLOFFYES) | 32 | RW | See section |
| 23_002Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD24_SLAST_SDA) | 32 | RW | See section |
| 23_0030h | TCD Destination Address (TCD24_DADDR) | 32 | RW | See section |
| 23_0034h | TCD Signed Destination Address Offset (TCD24_DOFF) | 16 | RW | See section |
| 23_0036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD24_CITER_ELINKNO) | 16 | RW | See section |
| 23_0036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD24_CITER_ELINKYES) | 16 | RW | See section |
| 23_0038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD24_DLAST_SGA) | 32 | RW | See section |
| 23_003Ch | TCD Control and Status (TCD24_CSR) | 16 | RW | See section |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 23_003Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD24_BITER_ELINKNO) | 16 | RW | See section |
| 23_003Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD24_BITER_ELINKYES) | 16 | RW | See section |
| 23_4000h | Channel Control and Status (CH25_CSR) | 32 | RW | 0000_0000h |
| 23_4004h | Channel Error Status (CH25_ES) | 32 | RW | 0000_0000h |
| 23_4008h | Channel Interrupt Status (CH25_INT) | 32 | RW | 0000_0000h |
| 23_400Ch | Channel System Bus (CH25_SBR) | 32 | RW | 0000_8002h |
| 23_4010h | Channel Priority (CH25_PRI) | 32 | RW | 0000_0000h |
| 23_4020h | TCD Source Address (TCD25_SADDR) | 32 | RW | See section |
| 23_4024h | TCD Signed Source Address Offset (TCD25_SOFF) | 16 | RW | See section |
| 23_4026h | TCD Transfer Attributes (TCD25_ATTR) | 16 | RW | See section |
| 23_4028h | TCD Transfer Size Without Minor Loop Offsets (TCD25_NBYTES_MLOFFNO) | 32 | RW | See section |
| 23_4028h | TCD Transfer Size with Minor Loop Offsets (TCD25_NBYTES_MLOFFYES) | 32 | RW | See section |
| 23_402Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD25_SLAST_SDA) | 32 | RW | See section |
| 23_4030h | TCD Destination Address (TCD25_DADDR) | 32 | RW | See section |
| 23_4034h | TCD Signed Destination Address Offset (TCD25_DOFF) | 16 | RW | See section |
| 23_4036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD25_CITER_ELINKNO) | 16 | RW | See section |
| 23_4036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD25_CITER_ELINKYES) | 16 | RW | See section |
| 23_4038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD25_DLAST_SGA) | 32 | RW | See section |
| 23_403Ch | TCD Control and Status (TCD25_CSR) | 16 | RW | See section |
| 23_403Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD25_BITER_ELINKNO) | 16 | RW | See section |
| 23_403Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD25_BITER_ELINKYES) | 16 | RW | See section |
| 23_8000h | Channel Control and Status (CH26_CSR) | 32 | RW | 0000_0000h |
| 23_8004h | Channel Error Status (CH26_ES) | 32 | RW | 0000_0000h |
| 23_8008h | Channel Interrupt Status (CH26_INT) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 23_800Ch | Channel System Bus (CH26_SBR) | 32 | RW | 0000_8002h |
| 23_8010h | Channel Priority (CH26_PRI) | 32 | RW | 0000_0000h |
| 23_8020h | TCD Source Address (TCD26_SADDR) | 32 | RW | See section |
| 23_8024h | TCD Signed Source Address Offset (TCD26_SOFF) | 16 | RW | See section |
| 23_8026h | TCD Transfer Attributes (TCD26_ATTR) | 16 | RW | See section |
| 23_8028h | TCD Transfer Size Without Minor Loop Offsets (TCD26_NBYTES_MLOFFNO) | 32 | RW | See section |
| 23_8028h | TCD Transfer Size with Minor Loop Offsets (TCD26_NBYTES_MLOFFYES) | 32 | RW | See section |
| 23_802Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD26_SLAST_SDA) | 32 | RW | See section |
| 23_8030h | TCD Destination Address (TCD26_DADDR) | 32 | RW | See section |
| 23_8034h | TCD Signed Destination Address Offset (TCD26_DOFF) | 16 | RW | See section |
| 23_8036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD26_CITER_ELINKNO) | 16 | RW | See section |
| 23_8036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD26_CITER_ELINKYES) | 16 | RW | See section |
| 23_8038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD26_DLAST_SGA) | 32 | RW | See section |
| 23_803Ch | TCD Control and Status (TCD26_CSR) | 16 | RW | See section |
| 23_803Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD26_BITER_ELINKNO) | 16 | RW | See section |
| 23_803Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD26_BITER_ELINKYES) | 16 | RW | See section |
| 23_C000h | Channel Control and Status (CH27_CSR) | 32 | RW | 0000_0000h |
| 23_C004h | Channel Error Status (CH27_ES) | 32 | RW | 0000_0000h |
| 23_C008h | Channel Interrupt Status (CH27_INT) | 32 | RW | 0000_0000h |
| 23_C00Ch | Channel System Bus (CH27_SBR) | 32 | RW | 0000_8002h |
| 23_C010h | Channel Priority (CH27_PRI) | 32 | RW | 0000_0000h |
| 23_C020h | TCD Source Address (TCD27_SADDR) | 32 | RW | See section |
| 23_C024h | TCD Signed Source Address Offset (TCD27_SOFF) | 16 | RW | See section |
| 23_C026h | TCD Transfer Attributes (TCD27_ATTR) | 16 | RW | See section |
| 23_C028h | TCD Transfer Size Without Minor Loop Offsets (TCD27_NBYTES_MLOFFNO) | 32 | RW | See section |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 23_C028h | TCD Transfer Size with Minor Loop Offsets (TCD27_NBYTES_MLOFFYES) | 32 | RW | See section |
| 23_C02Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD27_SLAST_SDA) | 32 | RW | See section |
| 23_C030h | TCD Destination Address (TCD27_DADDR) | 32 | RW | See section |
| 23_C034h | TCD Signed Destination Address Offset (TCD27_DOFF) | 16 | RW | See section |
| 23_C036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD27_CITER_ELINKNO) | 16 | RW | See section |
| 23_C036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD27_CITER_ELINKYES) | 16 | RW | See section |
| 23_C038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD27_DLAST_SGA) | 32 | RW | See section |
| 23_C03Ch | TCD Control and Status (TCD27_CSR) | 16 | RW | See section |
| 23_C03Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD27_BITER_ELINKNO) | 16 | RW | See section |
| 23_C03Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD27_BITER_ELINKYES) | 16 | RW | See section |
| 24_0000h | Channel Control and Status (CH28_CSR) | 32 | RW | 0000_0000h |
| 24_0004h | Channel Error Status (CH28_ES) | 32 | RW | 0000_0000h |
| 24_0008h | Channel Interrupt Status (CH28_INT) | 32 | RW | 0000_0000h |
| 24_000Ch | Channel System Bus (CH28_SBR) | 32 | RW | 0000_8002h |
| 24_0010h | Channel Priority (CH28_PRI) | 32 | RW | 0000_0000h |
| 24_0020h | TCD Source Address (TCD28_SADDR) | 32 | RW | See section |
| 24_0024h | TCD Signed Source Address Offset (TCD28_SOFF) | 16 | RW | See section |
| 24_0026h | TCD Transfer Attributes (TCD28_ATTR) | 16 | RW | See section |
| 24_0028h | TCD Transfer Size Without Minor Loop Offsets (TCD28_NBYTES_MLOFFNO) | 32 | RW | See section |
| 24_0028h | TCD Transfer Size with Minor Loop Offsets (TCD28_NBYTES_MLOFFYES) | 32 | RW | See section |
| 24_002Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD28_SLAST_SDA) | 32 | RW | See section |
| 24_0030h | TCD Destination Address (TCD28_DADDR) | 32 | RW | See section |
| 24_0034h | TCD Signed Destination Address Offset (TCD28_DOFF) | 16 | RW | See section |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 24_0036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD28_CITER_ELINKNO) | 16 | RW | See section |
| 24_0036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD28_CITER_ELINKYES) | 16 | RW | See section |
| 24_0038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD28_DLAST_SGA) | 32 | RW | See section |
| 24_003Ch | TCD Control and Status (TCD28_CSR) | 16 | RW | See section |
| 24_003Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD28_BITER_ELINKNO) | 16 | RW | See section |
| 24_003Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD28_BITER_ELINKYES) | 16 | RW | See section |
| 24_4000h | Channel Control and Status (CH29_CSR) | 32 | RW | 0000_0000h |
| 24_4004h | Channel Error Status (CH29_ES) | 32 | RW | 0000_0000h |
| 24_4008h | Channel Interrupt Status (CH29_INT) | 32 | RW | 0000_0000h |
| 24_400Ch | Channel System Bus (CH29_SBR) | 32 | RW | 0000_8002h |
| 24_4010h | Channel Priority (CH29_PRI) | 32 | RW | 0000_0000h |
| 24_4020h | TCD Source Address (TCD29_SADDR) | 32 | RW | See section |
| 24_4024h | TCD Signed Source Address Offset (TCD29_SOFF) | 16 | RW | See section |
| 24_4026h | TCD Transfer Attributes (TCD29_ATTR) | 16 | RW | See section |
| 24_4028h | TCD Transfer Size Without Minor Loop Offsets (TCD29_NBYTES_MLOFFNO) | 32 | RW | See section |
| 24_4028h | TCD Transfer Size with Minor Loop Offsets (TCD29_NBYTES_MLOFFYES) | 32 | RW | See section |
| 24_402Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD29_SLAST_SDA) | 32 | RW | See section |
| 24_4030h | TCD Destination Address (TCD29_DADDR) | 32 | RW | See section |
| 24_4034h | TCD Signed Destination Address Offset (TCD29_DOFF) | 16 | RW | See section |
| 24_4036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD29_CITER_ELINKNO) | 16 | RW | See section |
| 24_4036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD29_CITER_ELINKYES) | 16 | RW | See section |
| 24_4038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD29_DLAST_SGA) | 32 | RW | See section |
| 24_403Ch | TCD Control and Status (TCD29_CSR) | 16 | RW | See section |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 24_403Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD29_BITER_ELINKNO) | 16 | RW | See section |
| 24_403Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD29_BITER_ELINKYES) | 16 | RW | See section |
| 24_8000h | Channel Control and Status (CH30_CSR) | 32 | RW | 0000_0000h |
| 24_8004h | Channel Error Status (CH30_ES) | 32 | RW | 0000_0000h |
| 24_8008h | Channel Interrupt Status (CH30_INT) | 32 | RW | 0000_0000h |
| 24_800Ch | Channel System Bus (CH30_SBR) | 32 | RW | 0000_8002h |
| 24_8010h | Channel Priority (CH30_PRI) | 32 | RW | 0000_0000h |
| 24_8020h | TCD Source Address (TCD30_SADDR) | 32 | RW | See section |
| 24_8024h | TCD Signed Source Address Offset (TCD30_SOFF) | 16 | RW | See section |
| 24_8026h | TCD Transfer Attributes (TCD30_ATTR) | 16 | RW | See section |
| 24_8028h | TCD Transfer Size Without Minor Loop Offsets (TCD30_NBYTES_MLOFFNO) | 32 | RW | See section |
| 24_8028h | TCD Transfer Size with Minor Loop Offsets (TCD30_NBYTES_MLOFFYES) | 32 | RW | See section |
| 24_802Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD30_SLAST_SDA) | 32 | RW | See section |
| 24_8030h | TCD Destination Address (TCD30_DADDR) | 32 | RW | See section |
| 24_8034h | TCD Signed Destination Address Offset (TCD30_DOFF) | 16 | RW | See section |
| 24_8036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD30_CITER_ELINKNO) | 16 | RW | See section |
| 24_8036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD30_CITER_ELINKYES) | 16 | RW | See section |
| 24_8038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD30_DLAST_SGA) | 32 | RW | See section |
| 24_803Ch | TCD Control and Status (TCD30_CSR) | 16 | RW | See section |
| 24_803Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD30_BITER_ELINKNO) | 16 | RW | See section |
| 24_803Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD30_BITER_ELINKYES) | 16 | RW | See section |
| 24_C000h | Channel Control and Status (CH31_CSR) | 32 | RW | 0000_0000h |
| 24_C004h | Channel Error Status (CH31_ES) | 32 | RW | 0000_0000h |
| 24_C008h | Channel Interrupt Status (CH31_INT) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 24_C00Ch | Channel System Bus (CH31_SBR) | 32 | RW | 0000_8002h |
| 24_C010h | Channel Priority (CH31_PRI) | 32 | RW | 0000_0000h |
| 24_C020h | TCD Source Address (TCD31_SADDR) | 32 | RW | See section |
| 24_C024h | TCD Signed Source Address Offset (TCD31_SOFF) | 16 | RW | See section |
| 24_C026h | TCD Transfer Attributes (TCD31_ATTR) | 16 | RW | See section |
| 24_C028h | TCD Transfer Size Without Minor Loop Offsets (TCD31_NBYTES_MLOFFNO) | 32 | RW | See section |
| 24_C028h | TCD Transfer Size with Minor Loop Offsets (TCD31_NBYTES_MLOFFYES) | 32 | RW | See section |
| 24_C02Ch | TCD Last Source Address Adjustment / Store DADDR Address (TCD31_SLAST_SDA) | 32 | RW | See section |
| 24_C030h | TCD Destination Address (TCD31_DADDR) | 32 | RW | See section |
| 24_C034h | TCD Signed Destination Address Offset (TCD31_DOFF) | 16 | RW | See section |
| 24_C036h | TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD31_CITER_ELINKNO) | 16 | RW | See section |
| 24_C036h | TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD31_CITER_ELINKYES) | 16 | RW | See section |
| 24_C038h | TCD Last Destination Address Adjustment / Scatter Gather Address (TCD31_DLAST_SGA) | 32 | RW | See section |
| 24_C03Ch | TCD Control and Status (TCD31_CSR) | 16 | RW | See section |
| 24_C03Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD31_BITER_ELINKNO) | 16 | RW | See section |
| 24_C03Eh | TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD31_BITER_ELINKYES) | 16 | RW | See section |

### 14.6.2.2 Channel Control and Status (CH0_CSR - CH31_CSR)

**Offset**

| Register | Offset |
|---|---|
| CH0_CSR | 0h |
| CH1_CSR | 4000h |
| CH2_CSR | 8000h |
| CH3_CSR | C000h |

*Table continues on the next page...*

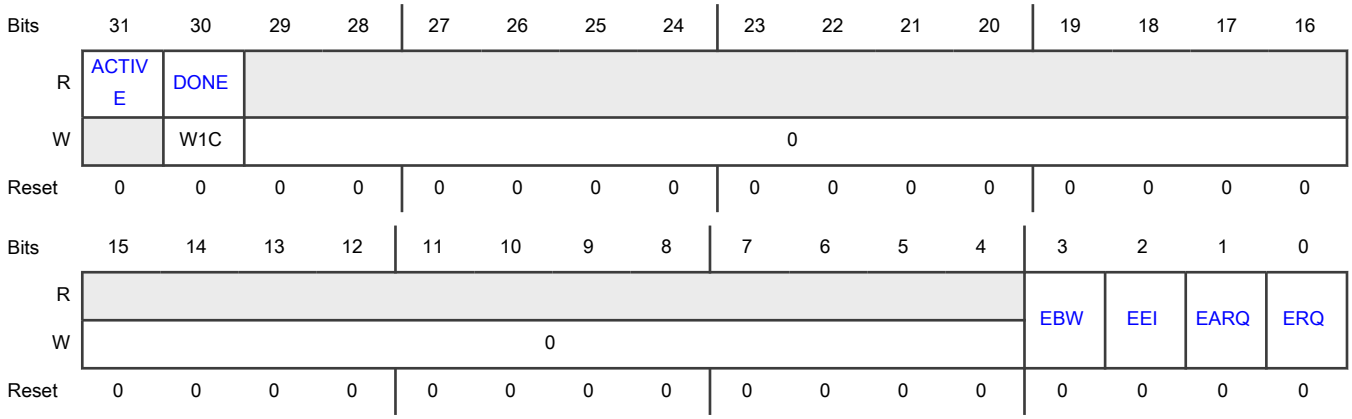
*Table continued from the previous page...*

| Register | Offset |
|---|---|
| CH4_CSR | 1_0000h |
| CH5_CSR | 1_4000h |
| CH6_CSR | 1_8000h |
| CH7_CSR | 1_C000h |
| CH8_CSR | 2_0000h |
| CH9_CSR | 2_4000h |
| CH10_CSR | 2_8000h |
| CH11_CSR | 2_C000h |
| CH12_CSR | 20_0000h |
| CH13_CSR | 20_4000h |
| CH14_CSR | 20_8000h |
| CH15_CSR | 20_C000h |
| CH16_CSR | 21_0000h |
| CH17_CSR | 21_4000h |
| CH18_CSR | 21_8000h |
| CH19_CSR | 21_C000h |
| CH20_CSR | 22_0000h |
| CH21_CSR | 22_4000h |
| CH22_CSR | 22_8000h |
| CH23_CSR | 22_C000h |
| CH24_CSR | 23_0000h |
| CH25_CSR | 23_4000h |
| CH26_CSR | 23_8000h |
| CH27_CSR | 23_C000h |
| CH28_CSR | 24_0000h |
| CH29_CSR | 24_4000h |
| CH30_CSR | 24_8000h |
| CH31_CSR | 24_C000h |

### Function

This register contains several fields related to hardware and interrupt requests, configuration, and status for the given channel.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>ACTIVE | Channel Active<br>The ACTIVE field indicates the channel was selected by arbitration and is executing the prescribed transfers. The eDMA sets it to 1 when channel service begins, and clears it to 0 as the minor loop completes or when any error condition is detected. Except for dynamic scatter/gather or dynamic channel linking, you must not modify the transfer control descriptor when a channel is active. |
| 30<br>DONE | Channel Done<br>The DONE field indicates the eDMA has completed the major loop. The eDMA engine sets this field as the CITER count reaches zero. If enabled, the eDMA generates an interrupt request corresponding to this completed channel. The software clears it, or the hardware clears it when the channel is activated.<br>**NOTE**<br>This field must be cleared to 0 before writing the MAJORELINK or ESG fields. |
| 29-4<br>— | Reserved |
| 3<br>EBW | Enable Buffered Writes<br>When buffered writes are enabled, all writes except for the last write sequence of the minor loop are signaled by the eDMA as bufferable.<br>0b - Buffered writes on system bus disabled. Buffered writes on system bus disabled<br>1b - Buffered writes on system bus enabled. Bufferable write signal asserted on all system bus writes except during last write sequence |
| 2<br>EEI | Enable Error Interrupt<br>The EEI field enables the error interrupt signal for the channel. The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller.<br>0b - Error signal for corresponding channel does not generate error interrupt |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Assertion of error signal for corresponding channel generates error interrupt request |
| 1<br>EARQ | Enable Asynchronous DMA Request In Stop Mode For Channel<br>The enable asynchronous DMA request field (EARQ) does not affect DMA operations. When set to 1, this field allows the hardware service request enable field (ERQ) to propagate out of the DMA to the power controller. When cleared to 0, this field masks the hardware service request enable field to the power controller.<br>0b - Disable asynchronous DMA request for the channel<br>1b - Enable asynchronous DMA request for the channel |
| 0<br>ERQ | Enable DMA Request<br>Disable a channel's hardware service request at the source before clearing the channel's ERQ field. The DMA hardware request input signal and the enable request field (ERQ) must be asserted before a channel's hardware service request is accepted. The state of the eDMA enable request field does not affect a channel service request made explicitly through software or channel linking. The state of the ERQ field does not affect the channel's START field.<br>0b - DMA hardware request signal for corresponding channel disabled<br>1b - DMA hardware request signal for corresponding channel enabled |

#### 14.6.2.3 Channel Error Status (CH0_ES - CH31_ES)

**Offset**

| Register | Offset |
|---|---|
| CH0_ES | 4h |
| CH1_ES | 4004h |
| CH2_ES | 8004h |
| CH3_ES | C004h |
| CH4_ES | 1_0004h |
| CH5_ES | 1_4004h |
| CH6_ES | 1_8004h |
| CH7_ES | 1_C004h |
| CH8_ES | 2_0004h |
| CH9_ES | 2_4004h |
| CH10_ES | 2_8004h |
| CH11_ES | 2_C004h |
| CH12_ES | 20_0004h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| CH13_ES | 20_4004h |
| CH14_ES | 20_8004h |
| CH15_ES | 20_C004h |
| CH16_ES | 21_0004h |
| CH17_ES | 21_4004h |
| CH18_ES | 21_8004h |
| CH19_ES | 21_C004h |
| CH20_ES | 22_0004h |
| CH21_ES | 22_4004h |
| CH22_ES | 22_8004h |
| CH23_ES | 22_C004h |
| CH24_ES | 23_0004h |
| CH25_ES | 23_4004h |
| CH26_ES | 23_8004h |
| CH27_ES | 23_C004h |
| CH28_ES | 24_0004h |
| CH29_ES | 24_4004h |
| CH30_ES | 24_8004h |
| CH31_ES | 24_C004h |

### Function

The ES provides information concerning the last recorded channel error. Channel errors can be caused by:

- An illegal setting in the transfer control descriptor
- An error termination to a bus master read or write cycle

The ERR field signals the presence of an error for the channel. The eDMA engine signals the occurrence of an error condition by setting the appropriate field in this register. The outputs of this register are enabled by the contents of the CHn_CSR[EEI] field, then logically summed across all channels to form an error interrupt request, which may be routed to the interrupt controller. In addition, this enabled error status is logically OR'd onto the channel done interrupt, CHn_INT[INT], thus forming a done or error interrupt on a per channel basis.

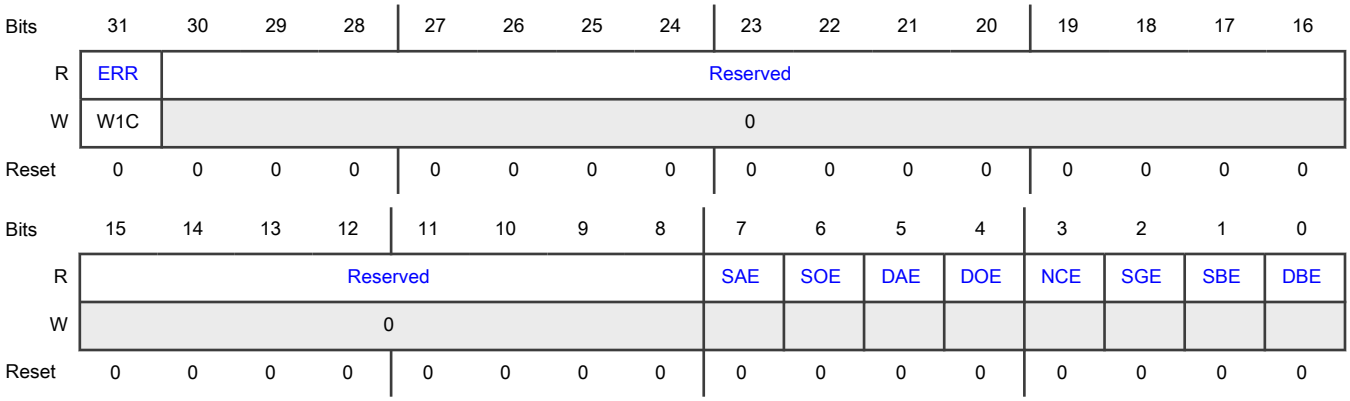
During the execution of the interrupt service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error-interrupt request. The normal DMA channel completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when eDMA detects an error. The contents of this ERR register field can also be polled because a non-zero value indicates the presence of a channel error, regardless of the state of the EEI mask.

The state of any given channel's error indicators is affected by writes to this register. Writing a 1 to the ERR field clears the channel's error status, and writing a 0 has no effect.

An unspecified error, where only the ERR field is set to 1, indicates that either a transfer was cancelled with an error or else an uncorrectable TCD error occurred. The Management Page Error Status register has full view of the error condition.

See Fault reporting and handling for more details.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>ERR | Error In Channel<br>0b - An error in this channel has not occurred<br>1b - An error in this channel has occurred |
| 30-8<br>— | Reserved |
| 7<br>SAE | Source Address Error<br>TCDn_SADDR is inconsistent with TCDn_ATTR[SSIZE].<br>0b - No source address configuration error<br>1b - Last recorded error was a configuration error detected in the TCDn_SADDR field |
| 6<br>SOE | Source Offset Error<br>TCDn_SOFF is inconsistent with TCDn_ATTR[SSIZE].<br>0b - No source offset configuration error<br>1b - Last recorded error was a configuration error detected in the TCDn_SOFF field |
| 5<br>DAE | Destination Address Error<br>TCDn_DADDR is inconsistent with TCDn_ATTR[DSIZE].<br>0b - No destination address configuration error<br>1b - Last recorded error was a configuration error detected in the TCDn_DADDR field |
| 4 | Destination Offset Error<br>TCDn_DOFF is inconsistent with TCDn_ATTR[DSIZE]. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| DOE | 0b - No destination offset configuration error<br>1b - Last recorded error was a configuration error detected in the TCDn_DOFF field |
| 3<br>NCE | NBYTES/CITER Configuration Error<br>This error indicates that one of the following has occurred:<br>• TCDn_NBYTES is not a multiple of TCDn_ATTR[SSIZE] and TCDn_ATTR[DSIZE]<br>• TCDn_CITER[CITER] is equal to zero<br>• TCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK]<br>0b - No NBYTES/CITER configuration error<br>1b - Last recorded error was a configuration error detected in the TCDn_NBYTES or TCDn_CITER fields |
| 2<br>SGE | Scatter/Gather Configuration Error<br>When this field is 1, it indicates that TCDn_DLAST_SGA is not on a 32-byte boundary. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCDn_CSR[ESG]is enabled.<br>0b - No scatter/gather configuration error<br>1b - Last recorded error was a configuration error detected in the TCDn_DLAST_SGA field |
| 1<br>SBE | Source Bus Error<br>0b - No source bus error<br>1b - Last recorded error was bus error on source read |
| 0<br>DBE | Destination Bus Error<br>0b - No destination bus error<br>1b - Last recorded error was bus error on destination write |

#### 14.6.2.4 Channel Interrupt Status (CH0_INT - CH31_INT)

**Offset**

| Register | Offset |
|---|---|
| CH0_INT | 8h |
| CH1_INT | 4008h |
| CH2_INT | 8008h |
| CH3_INT | C008h |
| CH4_INT | 1_0008h |
| CH5_INT | 1_4008h |

*Table continues on the next page...*

*Table continued from the previous page...*

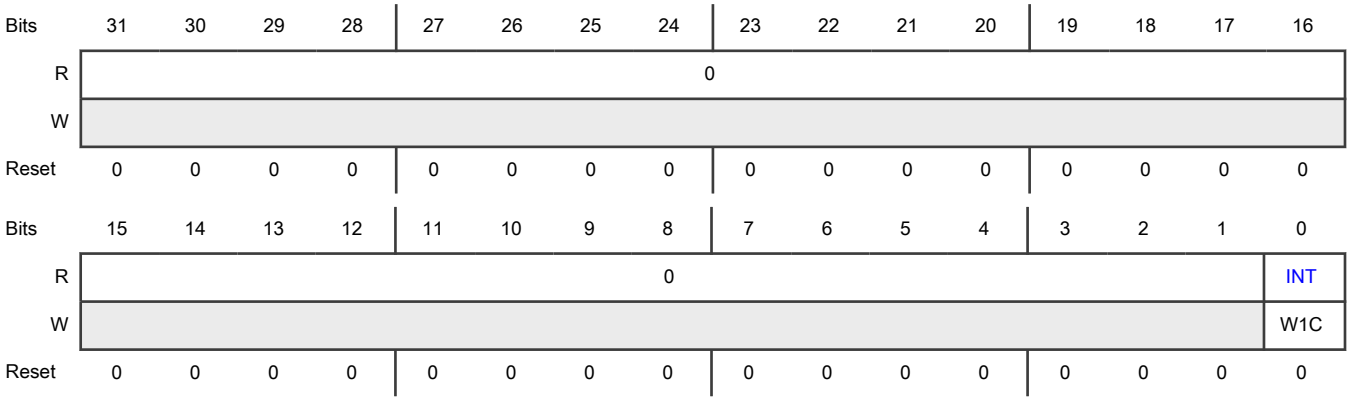
| Register | Offset |
|---|---|
| CH6_INT | 1_8008h |
| CH7_INT | 1_C008h |
| CH8_INT | 2_0008h |
| CH9_INT | 2_4008h |
| CH10_INT | 2_8008h |
| CH11_INT | 2_C008h |
| CH12_INT | 20_0008h |
| CH13_INT | 20_4008h |
| CH14_INT | 20_8008h |
| CH15_INT | 20_C008h |
| CH16_INT | 21_0008h |
| CH17_INT | 21_4008h |
| CH18_INT | 21_8008h |
| CH19_INT | 21_C008h |
| CH20_INT | 22_0008h |
| CH21_INT | 22_4008h |
| CH22_INT | 22_8008h |
| CH23_INT | 22_C008h |
| CH24_INT | 23_0008h |
| CH25_INT | 23_4008h |
| CH26_INT | 23_8008h |
| CH27_INT | 23_C008h |
| CH28_INT | 24_0008h |
| CH29_INT | 24_4008h |
| CH30_INT | 24_8008h |
| CH31_INT | 24_C008h |

### Function

The INT field signals the presence of an interrupt request for the channel. Depending on the appropriate bit setting in the transfe control descriptors, the eDMA engine generates an interrupt on data transfer completion or an error condition.

The outputs of this register are directly routed to the interrupt controller. During the interrupt service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. On writes to INT, a 1 clears the channel's interrupt request. A zero has no effect on the channel's current interrupt status.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>INT | Interrupt Request<br>0b - Interrupt request for corresponding channel cleared<br>1b - Interrupt request for corresponding channel active |

## 14.6.2.5 Channel System Bus (CH0_SBR - CH31_SBR)

### Offset

| Register | Offset |
|---|---|
| CH0_SBR | Ch |
| CH1_SBR | 400Ch |
| CH2_SBR | 800Ch |
| CH3_SBR | C00Ch |
| CH4_SBR | 1_000Ch |
| CH5_SBR | 1_400Ch |
| CH6_SBR | 1_800Ch |
| CH7_SBR | 1_C00Ch |
| CH8_SBR | 2_000Ch |
| CH9_SBR | 2_400Ch |
| CH10_SBR | 2_800Ch |
| CH11_SBR | 2_C00Ch |

*Table continues on the next page...*

*Table continued from the previous page...*

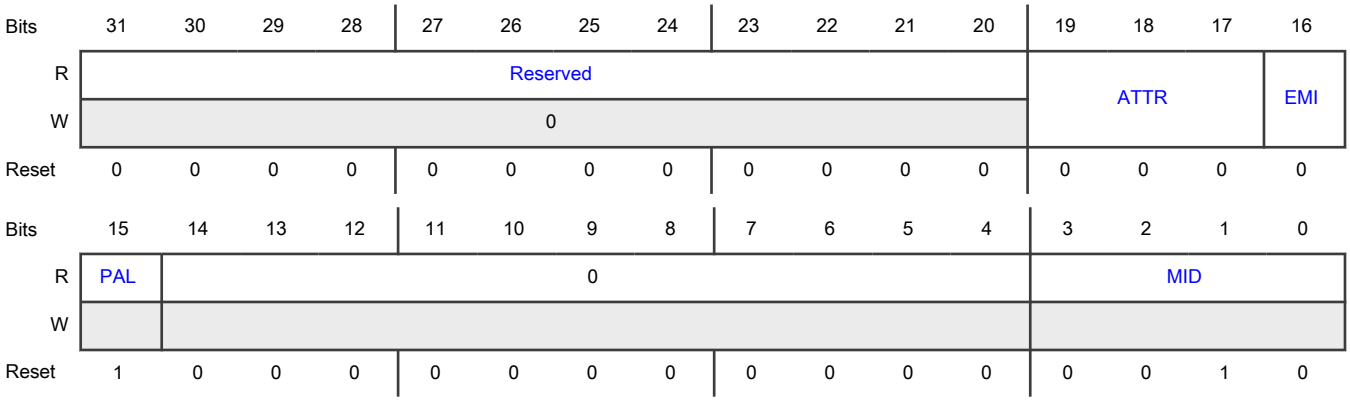
| Register | Offset |
|---|---|
| CH12_SBR | 20_000Ch |
| CH13_SBR | 20_400Ch |
| CH14_SBR | 20_800Ch |
| CH15_SBR | 20_C00Ch |
| CH16_SBR | 21_000Ch |
| CH17_SBR | 21_400Ch |
| CH18_SBR | 21_800Ch |
| CH19_SBR | 21_C00Ch |
| CH20_SBR | 22_000Ch |
| CH21_SBR | 22_400Ch |
| CH22_SBR | 22_800Ch |
| CH23_SBR | 22_C00Ch |
| CH24_SBR | 23_000Ch |
| CH25_SBR | 23_400Ch |
| CH26_SBR | 23_800Ch |
| CH27_SBR | 23_C00Ch |
| CH28_SBR | 24_000Ch |
| CH29_SBR | 24_400Ch |
| CH30_SBR | 24_800Ch |
| CH31_SBR | 24_C00Ch |

### Function

The Channel System Bus register places identification and attribute information on the system bus interface for the eDMA.

The ATTR register outputs the register values onto the system bus interface for further decoding by the security system.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19-17<br>ATTR | Attribute Output<br>DMA's system bus attribute output value. |
| 16<br>EMI | Enable Master ID Replication<br>The eDMA master ID replication field allows the eDMA to use the same protection level and system bus ID of the master programming the eDMA's TCD. When enabled, the eDMA uses the master ID and protection level stored in the CHn_SBR registers, instead of the eDMA's default values. When a master (for example a core) programs a TCD, its master ID and protection level are captured when the TCD*n*_CSR control attributes are written. A scatter/gather operation does not affect the CHn_SBR registers. You can write the EMI only if CSR[GMRC] = 1, which means Global Master ID Replication Control is enabled; otherwise, the EMI is forced to zero.<br>**NOTE**<br>If master ID replication is disabled, the privileged protection level (Supervisor mode) for DMA transfers is used.<br>0b - Master ID replication is disabled<br>1b - Master ID replication is enabled |
| 15<br>PAL | Privileged Access Level<br>This field controls DMA's protection level on the system bus when the channel is active.<br>**NOTE**<br>When you enable master ID replication, the value captured in this register is the privilege level of the core or other master writing the channel's transfer control descriptor, which is the lower byte of TCD*n*_CSR.<br>0b - User protection level for DMA transfers<br>1b - Privileged protection level for DMA transfers |
| 14-4<br>— | Reserved |
| 3-0<br>MID | Master ID<br>This field controls the DMA's master ID on the system bus when the channel is active.<br>**NOTE**<br>The ID captured in this register reflects the master ID of the core or other master writing the channel's control attributes, which are in the lower byte of TCD*n*_CSR. |

#### 14.6.2.6 Channel Priority (CH0_PRI - CH31_PRI)

**Offset**

| Register | Offset |
|---|---|
| CH0_PRI | 10h |
| CH1_PRI | 4010h |
| CH2_PRI | 8010h |
| CH3_PRI | C010h |
| CH4_PRI | 1_0010h |
| CH5_PRI | 1_4010h |
| CH6_PRI | 1_8010h |
| CH7_PRI | 1_C010h |
| CH8_PRI | 2_0010h |
| CH9_PRI | 2_4010h |
| CH10_PRI | 2_8010h |
| CH11_PRI | 2_C010h |
| CH12_PRI | 20_0010h |
| CH13_PRI | 20_4010h |
| CH14_PRI | 20_8010h |
| CH15_PRI | 20_C010h |
| CH16_PRI | 21_0010h |
| CH17_PRI | 21_4010h |
| CH18_PRI | 21_8010h |
| CH19_PRI | 21_C010h |
| CH20_PRI | 22_0010h |
| CH21_PRI | 22_4010h |
| CH22_PRI | 22_8010h |
| CH23_PRI | 22_C010h |
| CH24_PRI | 23_0010h |
| CH25_PRI | 23_4010h |
| CH26_PRI | 23_8010h |
| CH27_PRI | 23_C010h |
| CH28_PRI | 24_0010h |
| CH29_PRI | 24_4010h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| CH30_PRI | 24_8010h |
| CH31_PRI | 24_C010h |

### Function

The contents of these registers define unique priorities associated with each channel within the same channel group. Channel grouping is programmed via Channel Arbitration Group (CH0_GRPRI - CH31_GRPRI).

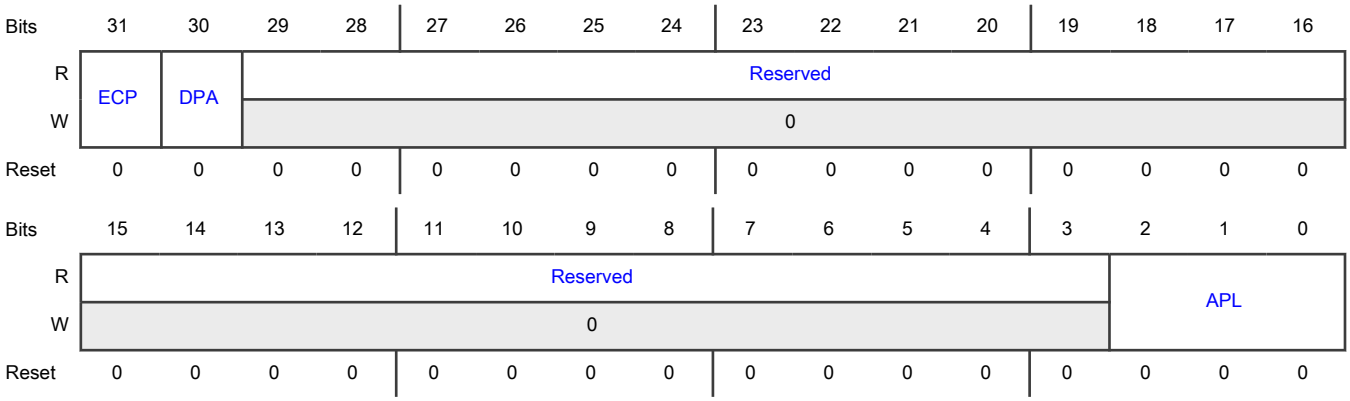
The channel priorities within a group are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, and so on. Software must program the channel priorities with unique values; otherwise, channel numbers with the same, non-zero value, will be selected based on channel number with the higher channel number having higher priority.

If more than one channel in a group has an arbitration priority level value of zero, then the arbitration mode field CSR[ERCA] is used to determine the arbitration scheme for all channels with APL=0 within a group.

When you enable round-robin channel arbitration (CSR[ERCA] = 1), all channels with APL=0 within a group will use a round-robin arbitration scheme, which rotates among these channels requesting service without regard to priority. Round-robin provides a fairness mechanism within an arbitration group.

When you enable fixed-priority channel arbitration (CSR[ERCA] = 0), eDMA selects channels with APL=0 based on channel number, with the higher channel number having higher priority.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>ECP | Enable Channel Preemption<br>0b - Channel cannot be suspended by a higher-priority channel's service request<br>1b - Channel can be temporarily suspended by a higher-priority channel's service request |
| 30<br>DPA | Disable Preempt Ability<br>0b - Channel can suspend a lower-priority channel<br>1b - Channel cannot suspend any other channel, regardless of channel priority |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 29-3<br>— | Reserved |
| 2-0<br>APL | Arbitration Priority Level<br>Channel priority level for arbitration within the assigned arbitration group. |

#### 14.6.2.7 TCD Source Address (TCD0_SADDR - TCD31_SADDR)

**Offset**

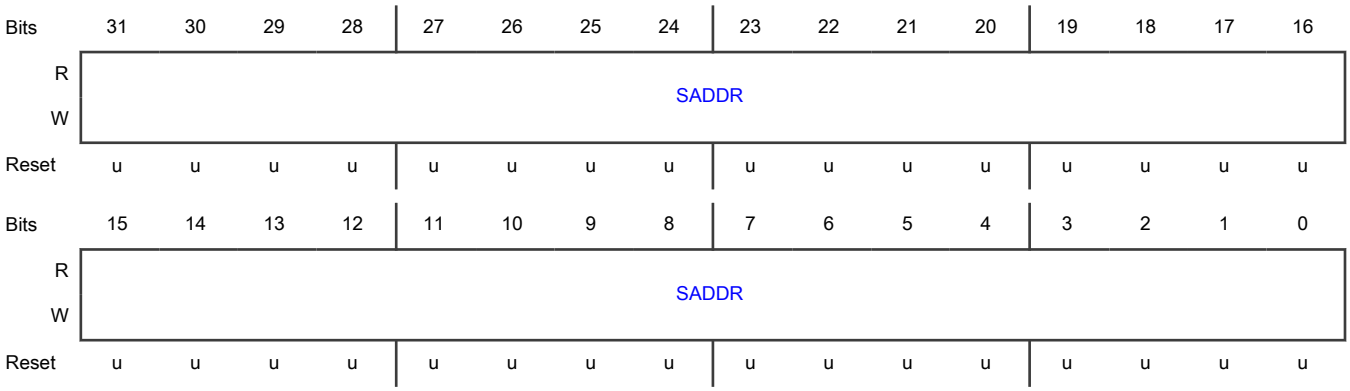
| Register | Offset |
|---|---|
| TCD0_SADDR | 20h |
| TCD1_SADDR | 4020h |
| TCD2_SADDR | 8020h |
| TCD3_SADDR | C020h |
| TCD4_SADDR | 1_0020h |
| TCD5_SADDR | 1_4020h |
| TCD6_SADDR | 1_8020h |
| TCD7_SADDR | 1_C020h |
| TCD8_SADDR | 2_0020h |
| TCD9_SADDR | 2_4020h |
| TCD10_SADDR | 2_8020h |
| TCD11_SADDR | 2_C020h |
| TCD12_SADDR | 20_0020h |
| TCD13_SADDR | 20_4020h |
| TCD14_SADDR | 20_8020h |
| TCD15_SADDR | 20_C020h |
| TCD16_SADDR | 21_0020h |
| TCD17_SADDR | 21_4020h |
| TCD18_SADDR | 21_8020h |
| TCD19_SADDR | 21_C020h |
| TCD20_SADDR | 22_0020h |
| TCD21_SADDR | 22_4020h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| TCD22_SADDR | 22_8020h |
| TCD23_SADDR | 22_C020h |
| TCD24_SADDR | 23_0020h |
| TCD25_SADDR | 23_4020h |
| TCD26_SADDR | 23_8020h |
| TCD27_SADDR | 23_C020h |
| TCD28_SADDR | 24_0020h |
| TCD29_SADDR | 24_4020h |
| TCD30_SADDR | 24_8020h |
| TCD31_SADDR | 24_C020h |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>SADDR | Source Address<br>Memory address pointing to the source data. |

## 14.6.2.8 TCD Signed Source Address Offset (TCD0_SOFF - TCD31_SOFF)

### Offset

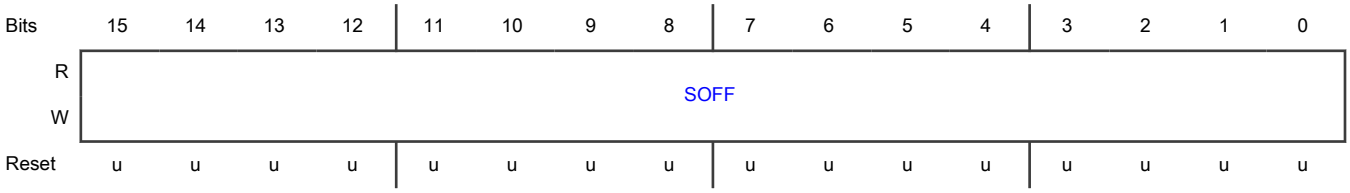
| Register | Offset |
|---|---|
| TCD0_SOFF | 24h |
| TCD1_SOFF | 4024h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| TCD2_SOFF | 8024h |
| TCD3_SOFF | C024h |
| TCD4_SOFF | 1_0024h |
| TCD5_SOFF | 1_4024h |
| TCD6_SOFF | 1_8024h |
| TCD7_SOFF | 1_C024h |
| TCD8_SOFF | 2_0024h |
| TCD9_SOFF | 2_4024h |
| TCD10_SOFF | 2_8024h |
| TCD11_SOFF | 2_C024h |
| TCD12_SOFF | 20_0024h |
| TCD13_SOFF | 20_4024h |
| TCD14_SOFF | 20_8024h |
| TCD15_SOFF | 20_C024h |
| TCD16_SOFF | 21_0024h |
| TCD17_SOFF | 21_4024h |
| TCD18_SOFF | 21_8024h |
| TCD19_SOFF | 21_C024h |
| TCD20_SOFF | 22_0024h |
| TCD21_SOFF | 22_4024h |
| TCD22_SOFF | 22_8024h |
| TCD23_SOFF | 22_C024h |
| TCD24_SOFF | 23_0024h |
| TCD25_SOFF | 23_4024h |
| TCD26_SOFF | 23_8024h |
| TCD27_SOFF | 23_C024h |
| TCD28_SOFF | 24_0024h |
| TCD29_SOFF | 24_4024h |
| TCD30_SOFF | 24_8024h |
| TCD31_SOFF | 24_C024h |

### Diagram




### Fields

| Field | Function |
|---|---|
| 15-0<br>SOFF | Source Address Signed Offset<br>Sign-extended offset applied to the current source address to form the next-state value as each source read is completed. |

## 14.6.2.9 TCD Transfer Attributes (TCD0_ATTR - TCD31_ATTR)

### Offset

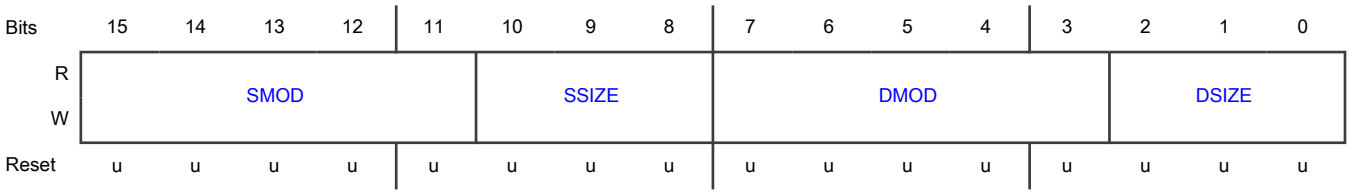
| Register | Offset |
|---|---|
| TCD0_ATTR | 26h |
| TCD1_ATTR | 4026h |
| TCD2_ATTR | 8026h |
| TCD3_ATTR | C026h |
| TCD4_ATTR | 1_0026h |
| TCD5_ATTR | 1_4026h |
| TCD6_ATTR | 1_8026h |
| TCD7_ATTR | 1_C026h |
| TCD8_ATTR | 2_0026h |
| TCD9_ATTR | 2_4026h |
| TCD10_ATTR | 2_8026h |
| TCD11_ATTR | 2_C026h |
| TCD12_ATTR | 20_0026h |
| TCD13_ATTR | 20_4026h |
| TCD14_ATTR | 20_8026h |
| TCD15_ATTR | 20_C026h |
| TCD16_ATTR | 21_0026h |
| TCD17_ATTR | 21_4026h |
| TCD18_ATTR | 21_8026h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| TCD19_ATTR | 21_C026h |
| TCD20_ATTR | 22_0026h |
| TCD21_ATTR | 22_4026h |
| TCD22_ATTR | 22_8026h |
| TCD23_ATTR | 22_C026h |
| TCD24_ATTR | 23_0026h |
| TCD25_ATTR | 23_4026h |
| TCD26_ATTR | 23_8026h |
| TCD27_ATTR | 23_C026h |
| TCD28_ATTR | 24_0026h |
| TCD29_ATTR | 24_4026h |
| TCD30_ATTR | 24_8026h |
| TCD31_ATTR | 24_C026h |

### Diagram




### Fields

| Field | Function |
|---|---|
| 15-11<br>SMOD | Source Address Modulo<br>This field defines a specific address range, which is the value after the SADDR + SOFF calculation is performed on the original register value. Setting this field makes it easy to implement a circular data queue.<br>For data queues requiring power-of-2-sized bytes, the queue must start at a 0-modulo-size address and the SMOD field must be set to the appropriate value for the queue, freezing the required number of upper address bits.<br>The value programmed into this field specifies the number of lower address bits that are allowed to change. For a circular queue application, you typically set TCDn_SOFF[SOFF] to the transfer size to implement post-increment addressing, with the SMOD function constraining the addresses to a 0-modulo-size range.<br>0_0000b - Source address modulo feature disabled<br>0_0001b - Source address modulo feature enabled for any non-zero value [1-31] |
| 10-8 | Source Data Transfer Size |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| SSIZE | 000b - 8-bit<br>001b - 16-bit<br>010b - 32-bit<br>011b - 64-bit<br>100b - 16-byte<br>101b - 32-byte<br>110b - 64-byte<br>111b - Reserved |
| 7-3<br>DMOD | Destination Address Modulo<br>See the SMOD definition. |
| 2-0<br>DSIZE | Destination Data Transfer Size<br>See the SSIZE definition. |

#### 14.6.2.10 TCD Transfer Size Without Minor Loop Offsets (TCD0_NBYTES_MLOFFNO - TCD31_NBYTES_MLOFFNO)

**Offset**

| Register | Offset |
|---|---|
| TCD0_NBYTES_MLOFFNO | 28h |
| TCD1_NBYTES_MLOFFNO | 4028h |
| TCD2_NBYTES_MLOFFNO | 8028h |
| TCD3_NBYTES_MLOFFNO | C028h |
| TCD4_NBYTES_MLOFFNO | 1_0028h |
| TCD5_NBYTES_MLOFFNO | 1_4028h |
| TCD6_NBYTES_MLOFFNO | 1_8028h |
| TCD7_NBYTES_MLOFFNO | 1_C028h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| TCD8_NBYTES_MLOFFNO | 2_0028h |
| TCD9_NBYTES_MLOFFNO | 2_4028h |
| TCD10_NBYTES_MLOFFNO | 2_8028h |
| TCD11_NBYTES_MLOFFNO | 2_C028h |
| TCD12_NBYTES_MLOFFNO | 20_0028h |
| TCD13_NBYTES_MLOFFNO | 20_4028h |
| TCD14_NBYTES_MLOFFNO | 20_8028h |
| TCD15_NBYTES_MLOFFNO | 20_C028h |
| TCD16_NBYTES_MLOFFNO | 21_0028h |
| TCD17_NBYTES_MLOFFNO | 21_4028h |
| TCD18_NBYTES_MLOFFNO | 21_8028h |
| TCD19_NBYTES_MLOFFNO | 21_C028h |
| TCD20_NBYTES_MLOFFNO | 22_0028h |
| TCD21_NBYTES_MLOFFNO | 22_4028h |
| TCD22_NBYTES_MLOFFNO | 22_8028h |
| TCD23_NBYTES_MLOFFNO | 22_C028h |
| TCD24_NBYTES_MLOFFNO | 23_0028h |
| TCD25_NBYTES_MLOFFNO | 23_4028h |
| TCD26_NBYTES_MLOFFNO | 23_8028h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| TCD27_NBYTES_MLOFFNO | 23_C028h |
| TCD28_NBYTES_MLOFFNO | 24_0028h |
| TCD29_NBYTES_MLOFFNO | 24_4028h |
| TCD30_NBYTES_MLOFFNO | 24_8028h |
| TCD31_NBYTES_MLOFFNO | 24_C028h |

### Function

The TCDn_NBYTES field defines the number of bytes to transfer per service request.

Minor loop offsets are address offset values added to the final source address (TCDn_SADDR), or destination address (TCDn_DADDR), upon minor loop completion. Minor loop completion is when the channel has finished the service request and has transferred NBYTES. When minor loop offsets are enabled, the minor loop offset value (TCDn_NBYTES_MLOFFYES[MLOFF]) is added to the final source address (TCDn_SADDR), to the final destination address (TCDn_DADDR), or to both, prior to the addresses being written back to the TCD. If the major loop is complete, the minor loop offset is ignored and the major loop address offsets (TCDn_SLAST_SDA and TCDn_DLAST_SGA) are used to compute the next TCDn_SADDR and TCDn_DADDR values.

When minor loop mapping is enabled (SMLOE or DMLOE is 1), TCDn_NBYTES_MLOFFNO/TCDn_NBYTES_MLOFFYES is redefined. A portion of TCDn_NBYTES_MLOFFNO/TCDn_NBYTES_MLOFFYES is used to specify multiple fields:

- A source enable bit (SMLOE) to specify the minor loop offset must be applied to the source address (TCDn_SADDR) upon minor loop completion
- A destination enable bit (DMLOE) to specify the minor loop offset must be applied to the destination address (TCDn_DADDR) upon minor loop completion
- The sign extended minor loop offset value (MLOFF)

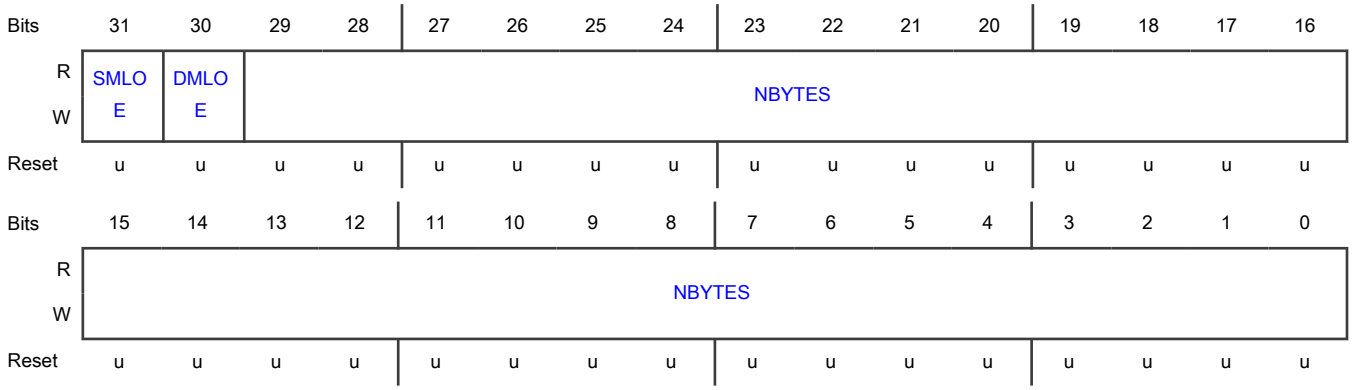
The same offset value (MLOFF) is used for both source and destination minor loop offsets. When either minor loop offset is enabled (SMLOE set or DMLOE set), the NBYTES field is reduced to 10 bits. If both minor loop offsets are disabled (SMLOE cleared and DMLOE cleared), the NBYTES field is a 30-bit vector.

One of two register profiles (this register or TCDn_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether source or destination minor loop mapping is enabled.

TCDn_NBYTES_MLOFFNO/TCDn_NBYTES_MLOFFYES is defined as follows:

- If SMLOE = 0 and DMLOE = 0, then see the TCDn_NBYTES_MLOFFNO register description.
- If either SMLOE or DMLOE is 1, then see the TCDn_NBYTES_MLOFFYES register description.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>SMLOE | Source Minor Loop Offset Enable<br>Selects whether the minor loop offset is applied to the source address upon minor loop completion.<br>0b - Minor loop offset not applied to SADDR<br>1b - Minor loop offset applied to SADDR |
| 30<br>DMLOE | Destination Minor Loop Offset Enable<br>Selects whether the minor loop offset is applied to the destination address upon minor loop completion.<br>0b - Minor loop offset not applied to DADDR<br>1b - Minor loop offset applied to DADDR |
| 29-0<br>NBYTES | Number of Bytes To Transfer Per Service Request<br>Number of bytes to be transferred for each service request of the channel.<br>When a channel activates, the module loads the appropriate TCD contents into the eDMA engine and performs the appropriate reads and writes until the byte transfer count has been reached. This process is normally an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption.<br>After the byte count is exhausted, the SADDR and DADDR values are written back into the TCD memory, and the major loop iteration count (CITER) is decremented by one and written back to the TCD memory. If the major iteration count is complete, additional processing is performed. |

## 14.6.2.11 TCD Transfer Size with Minor Loop Offsets (TCD0_NBYTES_MLOFFYES - TCD31_NBYTES_MLOFFYES)

### Offset

| Register | Offset |
|---|---|
| TCD0_NBYTES_MLOFFYES | 28h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| TCD1_NBYTES_MLOFFYES | 4028h |
| TCD2_NBYTES_MLOFFYES | 8028h |
| TCD3_NBYTES_MLOFFYES | C028h |
| TCD4_NBYTES_MLOFFYES | 1_0028h |
| TCD5_NBYTES_MLOFFYES | 1_4028h |
| TCD6_NBYTES_MLOFFYES | 1_8028h |
| TCD7_NBYTES_MLOFFYES | 1_C028h |
| TCD8_NBYTES_MLOFFYES | 2_0028h |
| TCD9_NBYTES_MLOFFYES | 2_4028h |
| TCD10_NBYTES_MLOFFYES | 2_8028h |
| TCD11_NBYTES_MLOFFYES | 2_C028h |
| TCD12_NBYTES_MLOFFYES | 20_0028h |
| TCD13_NBYTES_MLOFFYES | 20_4028h |
| TCD14_NBYTES_MLOFFYES | 20_8028h |
| TCD15_NBYTES_MLOFFYES | 20_C028h |
| TCD16_NBYTES_MLOFFYES | 21_0028h |
| TCD17_NBYTES_MLOFFYES | 21_4028h |
| TCD18_NBYTES_MLOFFYES | 21_8028h |
| TCD19_NBYTES_MLOFFYES | 21_C028h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| TCD20_NBYTES_MLOFFYES | 22_0028h |
| TCD21_NBYTES_MLOFFYES | 22_4028h |
| TCD22_NBYTES_MLOFFYES | 22_8028h |
| TCD23_NBYTES_MLOFFYES | 22_C028h |
| TCD24_NBYTES_MLOFFYES | 23_0028h |
| TCD25_NBYTES_MLOFFYES | 23_4028h |
| TCD26_NBYTES_MLOFFYES | 23_8028h |
| TCD27_NBYTES_MLOFFYES | 23_C028h |
| TCD28_NBYTES_MLOFFYES | 24_0028h |
| TCD29_NBYTES_MLOFFYES | 24_4028h |
| TCD30_NBYTES_MLOFFYES | 24_8028h |
| TCD31_NBYTES_MLOFFYES | 24_C028h |

### Function

The TCDn_NBYTES field defines the number of bytes to transfer per service request.

Minor loop offset is an address offset value added to the final source address (TCDn_SADDR) or destination address (TCDn_DADDR) upon minor loop completion. Minor loop completion occurs when the channel has finished the service request and has transferred NBYTES. Minor loop offsets are enabled by setting either the source enable bit (SMLOE) or the destination enable bit (DMLOE).

The source enable bit (SMLOE) specifies the minor loop offset value (MLOFF) that is to be applied to the source address (TCDn_SADDR) upon minor loop completion. The destination enable bit (DMLOE) specifies the minor loop offset (MLOFF) that is to be applied to the destination address (TCDn_DADDR) upon minor loop completion.

If the major loop is complete, the minor loop offsets are ignored and the major loop address offsets (TCDn_SLAST_SDA and TCDn_DLAST_SGA) are used to compute the next TCDn_SADDR and TCDn_DADDR values.

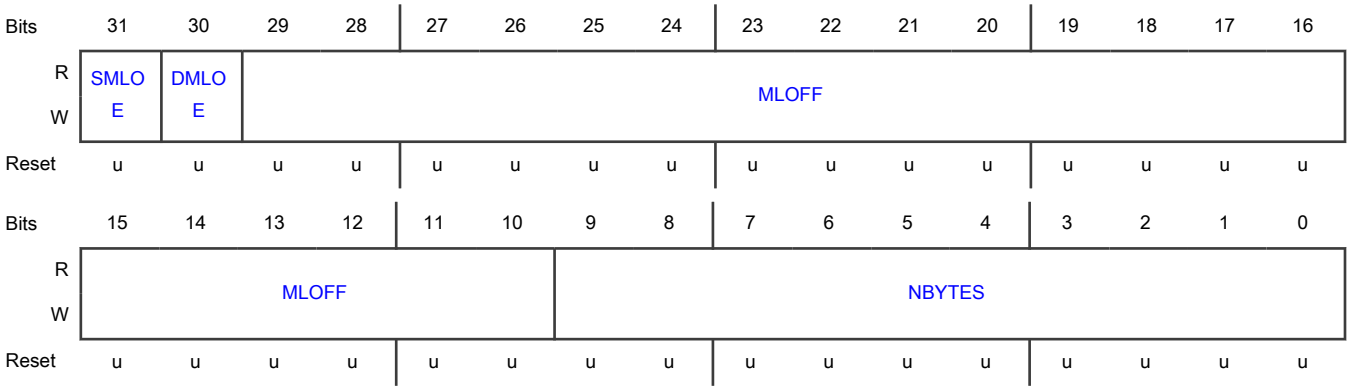
When you enable the minor loop offset overlay (either SMLOE or DMLOE is 1), eDMA redefines TCDn_NBYTES_MLOFFNO/TCDn_NBYTES_MLOFFYES. A portion of TCDn_NBYTES_MLOFFNO/TCDn_NBYTES_MLOFFYES specifies the sign-extended minor loop offset value (MLOFF). The same offset value (MLOFF) applies to both source and destination minor loop offsets. When the minor loop offset is enabled, you must align it to the transfer size of the source or destination it is associated with. When either minor loop offset is enabled (SMLOE set or DMLOE set), the NBYTES field is reduced to 10 bits. If both minor loop offsets are disabled (SMLOE cleared and DMLOE cleared), the NBYTES field is a 30-bit vector.

One of two register profiles (this register or TCDn_NBYTES_MLOFFNO) defines the number of bytes to transfer per request. Which register to use depends on whether source or destination minor loop mapping is enabled.

TCDn_NBYTES_MLOFFYES is defined as follows:

- If either minor loop offset is enabled (SMLOE or DMLOE = 1), then see the TCDn_NBYTES_MLOFFYES register description.
- If SMLOE and DMLOE are both 0, then see the TCDn_NBYTES_MLOFFNO register description.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>SMLOE | Source Minor Loop Offset Enable<br>Selects whether the minor loop offset is applied to the source address upon minor loop completion.<br>0b - Minor loop offset not applied to SADDR<br>1b - Minor loop offset applied to SADDR |
| 30<br>DMLOE | Destination Minor Loop Offset Enable<br>Selects whether the minor loop offset is applied to the destination address upon minor loop completion.<br>0b - Minor loop offset not applied to DADDR<br>1b - Minor loop offset applied to DADDR |
| 29-10<br>MLOFF | Minor Loop Offset<br>If SMLOE or DMLOE is 1, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop completes. |
| 9-0<br>NBYTES | Number of Bytes To Transfer Per Service Request<br>The number of bytes to be transferred in each service request of the channel.<br>As a channel activates, the module loads the appropriate TCD contents into the eDMA engine and performs the appropriate reads and writes until the minor byte transfer count has been reached. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption.<br>After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, and the major iteration count is decremented and restored to the TCD memory. If the major iteration count is complete, additional processing is performed. |

#### 14.6.2.12 TCD Last Source Address Adjustment / Store DADDR Address (TCD0_SLAST_SDA - TCD31_SLAST_SDA)

**Offset**

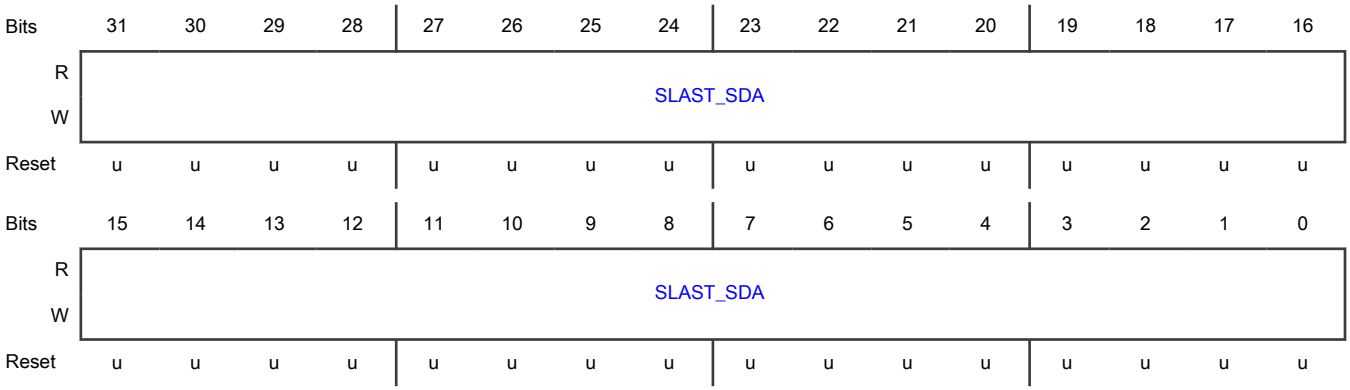
| Register | Offset |
|---|---|
| TCD0_SLAST_SDA | 2Ch |
| TCD1_SLAST_SDA | 402Ch |
| TCD2_SLAST_SDA | 802Ch |
| TCD3_SLAST_SDA | C02Ch |
| TCD4_SLAST_SDA | 1_002Ch |
| TCD5_SLAST_SDA | 1_402Ch |
| TCD6_SLAST_SDA | 1_802Ch |
| TCD7_SLAST_SDA | 1_C02Ch |
| TCD8_SLAST_SDA | 2_002Ch |
| TCD9_SLAST_SDA | 2_402Ch |
| TCD10_SLAST_SDA | 2_802Ch |
| TCD11_SLAST_SDA | 2_C02Ch |
| TCD12_SLAST_SDA | 20_002Ch |
| TCD13_SLAST_SDA | 20_402Ch |
| TCD14_SLAST_SDA | 20_802Ch |
| TCD15_SLAST_SDA | 20_C02Ch |
| TCD16_SLAST_SDA | 21_002Ch |
| TCD17_SLAST_SDA | 21_402Ch |
| TCD18_SLAST_SDA | 21_802Ch |
| TCD19_SLAST_SDA | 21_C02Ch |
| TCD20_SLAST_SDA | 22_002Ch |
| TCD21_SLAST_SDA | 22_402Ch |
| TCD22_SLAST_SDA | 22_802Ch |
| TCD23_SLAST_SDA | 22_C02Ch |
| TCD24_SLAST_SDA | 23_002Ch |
| TCD25_SLAST_SDA | 23_402Ch |
| TCD26_SLAST_SDA | 23_802Ch |
| TCD27_SLAST_SDA | 23_C02Ch |
| TCD28_SLAST_SDA | 24_002Ch |
| TCD29_SLAST_SDA | 24_402Ch |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| TCD30_SLAST_SDA | 24_802Ch |
| TCD31_SLAST_SDA | 24_C02Ch |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>SLAST_SDA | Last Source Address Adjustment / Store DADDR Address<br>Source last address adjustment or the system memory address for destination address (DADDR) storage.<br>If (TCDn_CSR[ESDA] = 0), then:<br>• Adjustment value is added to the source address at the completion of the major iteration count. This value can be used to restore the source address to the initial value or adjust the address to reference the next data structure.<br>• This field uses two's complement notation for the final source address adjustment.<br>Otherwise:<br>• This address points to the 32-bit-aligned memory location where the destination address (DADDR) is to be stored in system memory. By saving the final destination address in system memory via the ESDA feature, you are able to compute the size of a variable destination data buffer by simply subtracting the beginning DADDR from the final, saved DADDR. This feature is used together with the scatter/gather operation to prevent the loss of the final DADDR, which is overwritten during the scatter/gather operation.<br>The "Store Destination Address" (SDA) value must be a 32-bit-aligned location because the eDMA forces the lower two address bits of the SLAST_SDA field to zero when ESDA is enabled. The module performs this write operation when the major loop is done; that is, when the major iteration count (CITER) decrements to zero. |

### 14.6.2.13 TCD Destination Address (TCD0_DADDR - TCD31_DADDR)

**Offset**

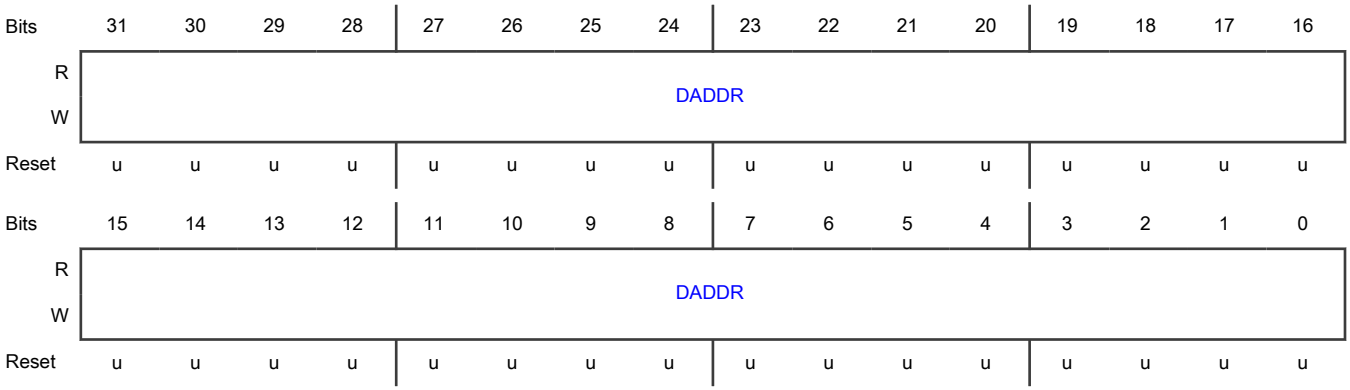
| Register | Offset |
|---|---|
| TCD0_DADDR | 30h |
| TCD1_DADDR | 4030h |
| TCD2_DADDR | 8030h |
| TCD3_DADDR | C030h |
| TCD4_DADDR | 1_0030h |
| TCD5_DADDR | 1_4030h |
| TCD6_DADDR | 1_8030h |
| TCD7_DADDR | 1_C030h |
| TCD8_DADDR | 2_0030h |
| TCD9_DADDR | 2_4030h |
| TCD10_DADDR | 2_8030h |
| TCD11_DADDR | 2_C030h |
| TCD12_DADDR | 20_0030h |
| TCD13_DADDR | 20_4030h |
| TCD14_DADDR | 20_8030h |
| TCD15_DADDR | 20_C030h |
| TCD16_DADDR | 21_0030h |
| TCD17_DADDR | 21_4030h |
| TCD18_DADDR | 21_8030h |
| TCD19_DADDR | 21_C030h |
| TCD20_DADDR | 22_0030h |
| TCD21_DADDR | 22_4030h |
| TCD22_DADDR | 22_8030h |
| TCD23_DADDR | 22_C030h |
| TCD24_DADDR | 23_0030h |
| TCD25_DADDR | 23_4030h |
| TCD26_DADDR | 23_8030h |
| TCD27_DADDR | 23_C030h |
| TCD28_DADDR | 24_0030h |
| TCD29_DADDR | 24_4030h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| TCD30_DADDR | 24_8030h |
| TCD31_DADDR | 24_C030h |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>DADDR | Destination Address<br>Memory address pointing to the destination data. |

## 14.6.2.14 TCD Signed Destination Address Offset (TCD0_DOFF - TCD31_DOFF)

### Offset

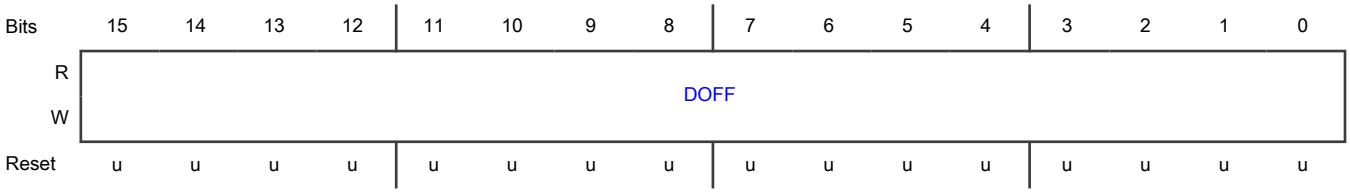
| Register | Offset |
|---|---|
| TCD0_DOFF | 34h |
| TCD1_DOFF | 4034h |
| TCD2_DOFF | 8034h |
| TCD3_DOFF | C034h |
| TCD4_DOFF | 1_0034h |
| TCD5_DOFF | 1_4034h |
| TCD6_DOFF | 1_8034h |
| TCD7_DOFF | 1_C034h |
| TCD8_DOFF | 2_0034h |
| TCD9_DOFF | 2_4034h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| TCD10_DOFF | 2_8034h |
| TCD11_DOFF | 2_C034h |
| TCD12_DOFF | 20_0034h |
| TCD13_DOFF | 20_4034h |
| TCD14_DOFF | 20_8034h |
| TCD15_DOFF | 20_C034h |
| TCD16_DOFF | 21_0034h |
| TCD17_DOFF | 21_4034h |
| TCD18_DOFF | 21_8034h |
| TCD19_DOFF | 21_C034h |
| TCD20_DOFF | 22_0034h |
| TCD21_DOFF | 22_4034h |
| TCD22_DOFF | 22_8034h |
| TCD23_DOFF | 22_C034h |
| TCD24_DOFF | 23_0034h |
| TCD25_DOFF | 23_4034h |
| TCD26_DOFF | 23_8034h |
| TCD27_DOFF | 23_C034h |
| TCD28_DOFF | 24_0034h |
| TCD29_DOFF | 24_4034h |
| TCD30_DOFF | 24_8034h |
| TCD31_DOFF | 24_C034h |

### Diagram




### Fields

| Field | Function |
|---|---|
| 15-0 | Destination Address Signed Offset |

*Table continues on the next page...*

| Field | Function |
|---|---|
| DOFF | Sign-extended offset that is applied to the current destination address to form the next-state value as each destination write is completed. |

#### 14.6.2.15 TCD Current Major Loop Count (Minor Loop Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD31_CITER_ELINKNO)

**Offset**

| Register | Offset |
|---|---|
| TCD0_CITER_ELINKNO | 36h |
| TCD1_CITER_ELINKNO | 4036h |
| TCD2_CITER_ELINKNO | 8036h |
| TCD3_CITER_ELINKNO | C036h |
| TCD4_CITER_ELINKNO | 1_0036h |
| TCD5_CITER_ELINKNO | 1_4036h |
| TCD6_CITER_ELINKNO | 1_8036h |
| TCD7_CITER_ELINKNO | 1_C036h |
| TCD8_CITER_ELINKNO | 2_0036h |
| TCD9_CITER_ELINKNO | 2_4036h |
| TCD10_CITER_ELINKNO | 2_8036h |
| TCD11_CITER_ELINKNO | 2_C036h |
| TCD12_CITER_ELINKNO | 20_0036h |
| TCD13_CITER_ELINKNO | 20_4036h |
| TCD14_CITER_ELINKNO | 20_8036h |
| TCD15_CITER_ELINKNO | 20_C036h |
| TCD16_CITER_ELINKNO | 21_0036h |
| TCD17_CITER_ELINKNO | 21_4036h |
| TCD18_CITER_ELINKNO | 21_8036h |

*Table continues on the next page...*

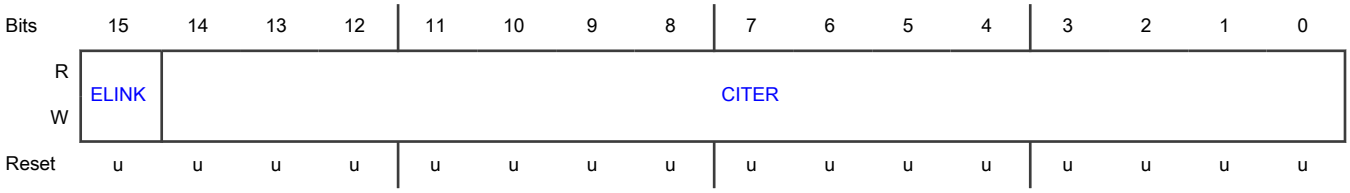
*Table continued from the previous page...*

| Register | Offset |
|---|---|
| TCD19_CITER_ELINKNO | 21_C036h |
| TCD20_CITER_ELINKNO | 22_0036h |
| TCD21_CITER_ELINKNO | 22_4036h |
| TCD22_CITER_ELINKNO | 22_8036h |
| TCD23_CITER_ELINKNO | 22_C036h |
| TCD24_CITER_ELINKNO | 23_0036h |
| TCD25_CITER_ELINKNO | 23_4036h |
| TCD26_CITER_ELINKNO | 23_8036h |
| TCD27_CITER_ELINKNO | 23_C036h |
| TCD28_CITER_ELINKNO | 24_0036h |
| TCD29_CITER_ELINKNO | 24_4036h |
| TCD30_CITER_ELINKNO | 24_8036h |
| TCD31_CITER_ELINKNO | 24_C036h |

### Function

If TCDn_CITER[ELINK] is 0, the TCDn_CITER register is defined as follows.

### Diagram




### Fields

| Field | Function |
|---|---|
| 15<br>ELINK | Enable Link<br>As the channel completes the minor loop, this flag enables linking to another channel as defined by the relevant LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel to 1.<br>If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of MAJORELINK channel linking.<br>**NOTE**<br>This field must be equal to the BITER[ELINK] field; otherwise, a configuration error is reported.<br>0b - Channel-to-channel linking disabled<br>1b - Channel-to-channel linking enabled |
| 14-0<br>CITER | Current Major Iteration Count<br>This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the channel finishes a service request and is written back to TCD memory. After the major iteration count is exhausted, the channel performs a number of operations — for example, final source and destination address calculations — and optionally generates an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field.<br>**NOTE**<br>When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field.<br>**NOTE**<br>If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001. |

### 14.6.2.16 TCD Current Major Loop Count (Minor Loop Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD31_CITER_ELINKYES)

**Offset**

| Register | Offset |
|---|---|
| TCD0_CITER_ELINKYES | 36h |
| TCD1_CITER_ELINKYES | 4036h |
| TCD2_CITER_ELINKYES | 8036h |
| TCD3_CITER_ELINKYES | C036h |
| TCD4_CITER_ELINKYES | 1_0036h |
| TCD5_CITER_ELINKYES | 1_4036h |
| TCD6_CITER_ELINKYES | 1_8036h |
| TCD7_CITER_ELINKYES | 1_C036h |
| TCD8_CITER_ELINKYES | 2_0036h |
| TCD9_CITER_ELINKYES | 2_4036h |
| TCD10_CITER_ELINKYES | 2_8036h |
| TCD11_CITER_ELINKYES | 2_C036h |
| TCD12_CITER_ELINKYES | 20_0036h |
| TCD13_CITER_ELINKYES | 20_4036h |
| TCD14_CITER_ELINKYES | 20_8036h |
| TCD15_CITER_ELINKYES | 20_C036h |
| TCD16_CITER_ELINKYES | 21_0036h |
| TCD17_CITER_ELINKYES | 21_4036h |

*Table continues on the next page...*

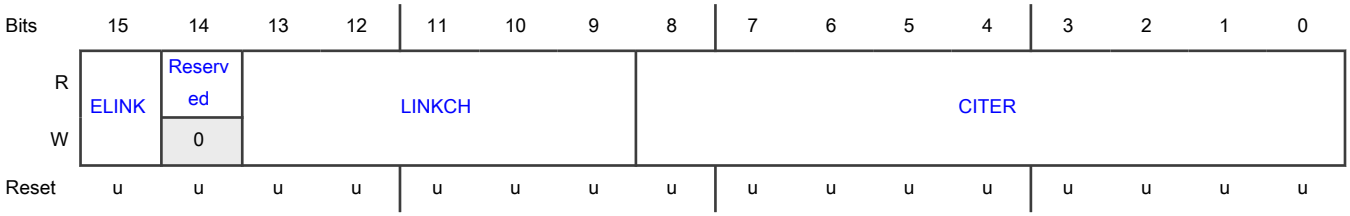
*Table continued from the previous page...*

| Register | Offset |
|---|---|
| TCD18_CITER_ELINKYES | 21_8036h |
| TCD19_CITER_ELINKYES | 21_C036h |
| TCD20_CITER_ELINKYES | 22_0036h |
| TCD21_CITER_ELINKYES | 22_4036h |
| TCD22_CITER_ELINKYES | 22_8036h |
| TCD23_CITER_ELINKYES | 22_C036h |
| TCD24_CITER_ELINKYES | 23_0036h |
| TCD25_CITER_ELINKYES | 23_4036h |
| TCD26_CITER_ELINKYES | 23_8036h |
| TCD27_CITER_ELINKYES | 23_C036h |
| TCD28_CITER_ELINKYES | 24_0036h |
| TCD29_CITER_ELINKYES | 24_4036h |
| TCD30_CITER_ELINKYES | 24_8036h |
| TCD31_CITER_ELINKYES | 24_C036h |

### Function

If TCDn_CITER[ELINK] is 1, the TCDn_CITER register is defined as follows.

### Diagram




### Fields

| Field | Function |
|---|---|
| 15<br>ELINK | Enable Link<br>As the channel completes the minor loop, this flag enables linking to another channel as defined by the relevant LINKCH field. When enabled, an internal mechanism sets the TCDn_CSR[START] field of the specified channel (LINKCH) upon minor loop completion.<br>If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of MAJORELINK channel linking.<br>**NOTE**<br>This field must be equal to the BITER[ELINK] field; otherwise, a configuration error is reported.<br>0b - Channel-to-channel linking disabled<br>1b - Channel-to-channel linking enabled |
| 14<br>— | Reserved |
| 13-9<br>LINKCH | Minor Loop Link Channel Number<br>If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted the eDMA engine initiates a channel service request to the channel defined by this field by writing that channel's TCDn_CSR[START] field to 1. |
| 8-0<br>CITER | Current Major Iteration Count<br>This 9-bit (ELINK = 1) or 15-bit (ELINK = 0) count represents the current major loop count for the channel. It is decremented each time the channel finishes a service request and is written back to the TCD memory. After the major iteration count is exhausted, the channel performs a number of operations — for example, final source and destination address calculations — and optionally generates an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field.<br>**NOTE**<br>When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field.<br>**NOTE**<br>If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001. |

#### 14.6.2.17 TCD Last Destination Address Adjustment / Scatter Gather Address (TCD0_DLAST_SGA - TCD31_DLAST_SGA)

**Offset**

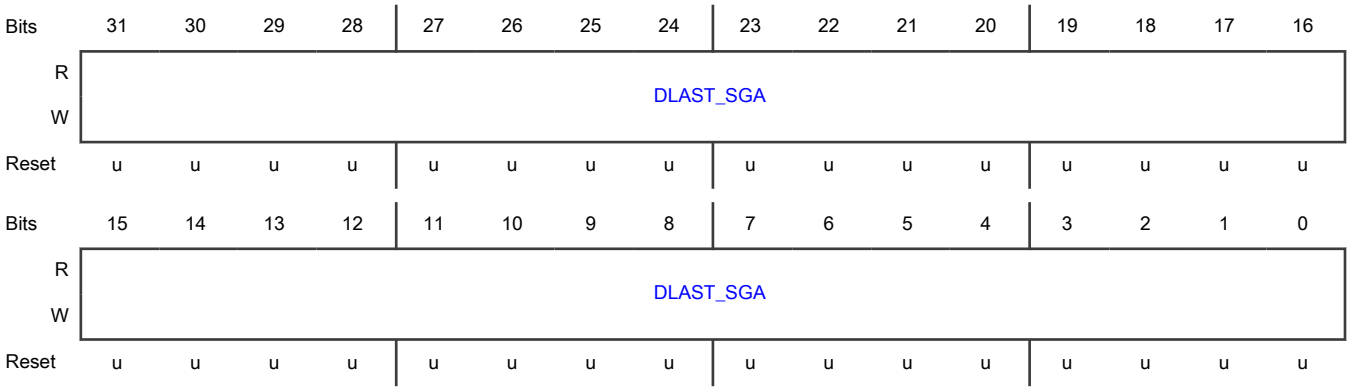
| Register | Offset |
|---|---|
| TCD0_DLAST_SGA | 38h |
| TCD1_DLAST_SGA | 4038h |
| TCD2_DLAST_SGA | 8038h |
| TCD3_DLAST_SGA | C038h |
| TCD4_DLAST_SGA | 1_0038h |
| TCD5_DLAST_SGA | 1_4038h |
| TCD6_DLAST_SGA | 1_8038h |
| TCD7_DLAST_SGA | 1_C038h |
| TCD8_DLAST_SGA | 2_0038h |
| TCD9_DLAST_SGA | 2_4038h |
| TCD10_DLAST_SGA | 2_8038h |
| TCD11_DLAST_SGA | 2_C038h |
| TCD12_DLAST_SGA | 20_0038h |
| TCD13_DLAST_SGA | 20_4038h |
| TCD14_DLAST_SGA | 20_8038h |
| TCD15_DLAST_SGA | 20_C038h |
| TCD16_DLAST_SGA | 21_0038h |
| TCD17_DLAST_SGA | 21_4038h |
| TCD18_DLAST_SGA | 21_8038h |
| TCD19_DLAST_SGA | 21_C038h |
| TCD20_DLAST_SGA | 22_0038h |
| TCD21_DLAST_SGA | 22_4038h |
| TCD22_DLAST_SGA | 22_8038h |
| TCD23_DLAST_SGA | 22_C038h |
| TCD24_DLAST_SGA | 23_0038h |
| TCD25_DLAST_SGA | 23_4038h |
| TCD26_DLAST_SGA | 23_8038h |
| TCD27_DLAST_SGA | 23_C038h |
| TCD28_DLAST_SGA | 24_0038h |
| TCD29_DLAST_SGA | 24_4038h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| TCD30_DLAST_SGA | 24_8038h |
| TCD31_DLAST_SGA | 24_C038h |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>DLAST_SGA | Last Destination Address Adjustment / Scatter Gather Address<br>Adjustment of the last destination address or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather).<br>If (TCDn_CSR[ESG] = 0) then:<br>• Adjustment value is added to the destination address at the completion of the major iteration count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure.<br>• This field uses two's complement notation for the final destination address adjustment.<br>Otherwise:<br>• This address points to the beginning of a 0-modulo 32-byte region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo 32-byte, or else a configuration error is reported. |

## 14.6.2.18 TCD Control and Status (TCD0_CSR - TCD31_CSR)

### Offset

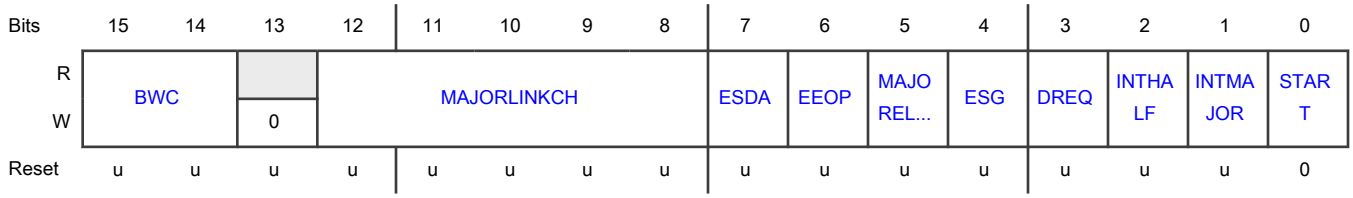
| Register | Offset |
|---|---|
| TCD0_CSR | 3Ch |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| TCD1_CSR | 403Ch |
| TCD2_CSR | 803Ch |
| TCD3_CSR | C03Ch |
| TCD4_CSR | 1_003Ch |
| TCD5_CSR | 1_403Ch |
| TCD6_CSR | 1_803Ch |
| TCD7_CSR | 1_C03Ch |
| TCD8_CSR | 2_003Ch |
| TCD9_CSR | 2_403Ch |
| TCD10_CSR | 2_803Ch |
| TCD11_CSR | 2_C03Ch |
| TCD12_CSR | 20_003Ch |
| TCD13_CSR | 20_403Ch |
| TCD14_CSR | 20_803Ch |
| TCD15_CSR | 20_C03Ch |
| TCD16_CSR | 21_003Ch |
| TCD17_CSR | 21_403Ch |
| TCD18_CSR | 21_803Ch |
| TCD19_CSR | 21_C03Ch |
| TCD20_CSR | 22_003Ch |
| TCD21_CSR | 22_403Ch |
| TCD22_CSR | 22_803Ch |
| TCD23_CSR | 22_C03Ch |
| TCD24_CSR | 23_003Ch |
| TCD25_CSR | 23_403Ch |
| TCD26_CSR | 23_803Ch |
| TCD27_CSR | 23_C03Ch |
| TCD28_CSR | 24_003Ch |
| TCD29_CSR | 24_403Ch |
| TCD30_CSR | 24_803Ch |
| TCD31_CSR | 24_C03Ch |

### Diagram




### Fields

| Field | Function |
|---|---|
| 15-14<br>BWC | Bandwidth Control<br>Throttles the amount of bus bandwidth consumed by the eDMA. Generally, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces eDMA to stall after the completion of each read/write access, to control the bus request bandwidth seen by the system bus interconnect.<br>**NOTE**<br>If the source and destination sizes are equal, this field is ignored between the first and second transfers and after the last write of each minor loop. This behavior is a side effect of reducing start-up latency.<br>00b - No eDMA engine stalls<br>01b - Enable eDMA master high-priority elevation (HPE) mode. No eDMA engine stalls.<br>10b - eDMA engine stalls for 4 cycles after each R/W<br>11b - eDMA engine stalls for 8 cycles after each R/W |
| 13<br>— | Reserved |
| 12-8<br>MAJORLINKCH | Major Loop Link Channel Number<br>If (MAJORELINK = 0) then:<br>• No channel-to-channel linking, or chaining, is performed after the major loop counter is exhausted.<br>Otherwise:<br>• After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] field to 1. |
| 7<br>ESDA | Enable Store Destination Address<br>As the channel completes the major loop by either the current iteration counter (CITER) decrementing to 0, or by receiving an enabled end-of-packet signal, this field enables writing the destination address (DADDR) to the address stored in the SLAST_SDA field. The value written to system memory is the last DADDR value prior to the DLAST_SGA offset being applied, or overwritten by an enabled scatter/gather operation. When the SDA bit is 1, SLAST_SDA contains the write pointer instead of the final source address offset. Because this is a pointer and not a final offset, a last source address offset of zero is applied to SADDR instead of the SLAST_SGA value.<br>0b - Ability to store destination address to system memory disabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Ability to store destination address to system memory enabled |
| 6<br>EEOP | Enable End-Of-Packet Processing<br>When enabled by the EEOP field, an end-of-packet hardware input signal directs eDMA to discontinue executing the active channel, and to treat the shutdown as the major-loop-completed event. If the EEOP field is 1, the end-of-packet signal from supported peripherals is accepted. If the EEOP field is 0, the end-of-packet input is ignored. With an end-of-packet retirement, the current TCD destination address (or ESDA-saved destination address), minus the software-saved initial address (DADDR), reflects the total amount of data transferred.<br>0b - End-of-packet operation disabled<br>1b - End-of-packet hardware input signal enabled |
| 5<br>MAJORELINK | Enable Link When Major Loop Complete<br>As the channel completes the major loop, this flag enables linking to another channel defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] field of the specified channel.<br>**NOTE**<br>To support the dynamic linking coherency model, this field is forced to 0 if written when TCDn_CSR[DONE] is 1.<br>0b - Channel-to-channel linking disabled<br>1b - Channel-to-channel linking enabled |
| 4<br>ESG | Enable Scatter/Gather Processing<br>As the channel completes the major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses TCDn_DLAST_SGA as a memory pointer to a 0-modulo 32-bit address containing a 32-byte data structure, which is loaded as the transfer control descriptor into local memory.<br>**NOTE**<br>To support the dynamic scatter/gather coherency model, this field is forced to 0 if written when TCDn_CSR[DONE] is 1.<br>0b - Current channel's TCD is normal format<br>1b - Current channel's TCD specifies scatter/gather format. |
| 3<br>DREQ | Disable Request<br>If this flag is 1, the eDMA hardware automatically clears the corresponding ERQ bit when the current major iteration count reaches 0.<br>0b - No operation. Channel's ERQ field not affected<br>1b - Clear the ERQ field to 0 upon major loop completion, thus disabling hardware service requests. Channel's ERQ field cleared to 0 when major loop complete |
| 2 | Enable Interrupt If Major Counter Half-complete |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| INTHALF | If this flag is 1, the channel generates an interrupt request by setting the appropriate field in the INT register to 1 when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER = (BITER/2)). This halfway point interrupt request is provided to support double-buffered, also known as ping-pong, schemes, or other types of data movement where the processor needs an early indication of the transfer's progress.<br>**NOTE**<br>If BITER = 1, do not use INTHALF; use INTMAJOR instead.<br>0b - Halfway point interrupt disabled<br>1b - Halfway point interrupt enabled |
| 1<br>INTMAJOR | Enable Interrupt If Major count complete<br>If this flag is 1, the channel generates an interrupt request by setting the appropriate field in the INT register to 1 when the current major iteration count (CITER) reaches 0.<br>0b - End-of-major loop interrupt disabled<br>1b - End-of-major loop interrupt enabled |
| 0<br>START | Channel Start<br>If this flag is 1, the channel is requesting service. The eDMA hardware automatically clears this flag to 0 after the channel begins execution.<br>0b - Channel not explicitly started<br>1b - Channel explicitly started via a software-initiated service request |

#### 14.6.2.19 TCD Beginning Major Loop Count (Minor Loop Channel Linking Disabled) (TCD0_BITER_ELINKNO - TCD31_BITER_ELINKNO)

**Offset**

| Register | Offset |
|---|---|
| TCD0_BITER_ELINKNO | 3Eh |
| TCD1_BITER_ELINKNO | 403Eh |
| TCD2_BITER_ELINKNO | 803Eh |
| TCD3_BITER_ELINKNO | C03Eh |
| TCD4_BITER_ELINKNO | 1_003Eh |
| TCD5_BITER_ELINKNO | 1_403Eh |
| TCD6_BITER_ELINKNO | 1_803Eh |
| TCD7_BITER_ELINKNO | 1_C03Eh |
| TCD8_BITER_ELINKNO | 2_003Eh |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| TCD9_BITER_ELINKNO | 2_403Eh |
| TCD10_BITER_ELINKNO | 2_803Eh |
| TCD11_BITER_ELINKNO | 2_C03Eh |
| TCD12_BITER_ELINKNO | 20_003Eh |
| TCD13_BITER_ELINKNO | 20_403Eh |
| TCD14_BITER_ELINKNO | 20_803Eh |
| TCD15_BITER_ELINKNO | 20_C03Eh |
| TCD16_BITER_ELINKNO | 21_003Eh |
| TCD17_BITER_ELINKNO | 21_403Eh |
| TCD18_BITER_ELINKNO | 21_803Eh |
| TCD19_BITER_ELINKNO | 21_C03Eh |
| TCD20_BITER_ELINKNO | 22_003Eh |
| TCD21_BITER_ELINKNO | 22_403Eh |
| TCD22_BITER_ELINKNO | 22_803Eh |
| TCD23_BITER_ELINKNO | 22_C03Eh |
| TCD24_BITER_ELINKNO | 23_003Eh |
| TCD25_BITER_ELINKNO | 23_403Eh |
| TCD26_BITER_ELINKNO | 23_803Eh |
| TCD27_BITER_ELINKNO | 23_C03Eh |

*Table continues on the next page...*

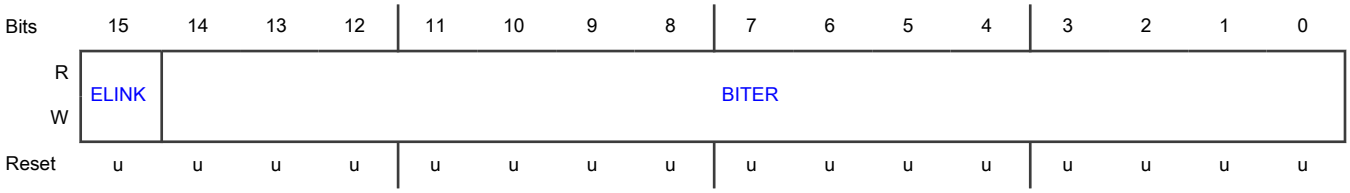
*Table continued from the previous page...*

| Register | Offset |
|---|---|
| TCD28_BITER_ELINKNO | 24_003Eh |
| TCD29_BITER_ELINKNO | 24_403Eh |
| TCD30_BITER_ELINKNO | 24_803Eh |
| TCD31_BITER_ELINKNO | 24_C03Eh |

### Function

If the TCDn_BITER[ELINK] field is 0, the TCDn_BITER register is defined as follows.

### Diagram




### Fields

| Field | Function |
|---|---|
| 15<br>ELINK | Enables Link<br>As the channel completes the minor loop, this flag enables linking to another channel as defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] field of the specified channel. If channel linking is disabled, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking.<br>**NOTE**<br>When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, eDMA reloads the contents of this field into the CITER field.<br>0b - Channel-to-channel linking disabled<br>1b - Channel-to-channel linking enabled |
| 14-0<br>BITER | Starting Major Iteration Count<br>As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be set equal to the value in the CITER field. As the major iteration count is exhausted, eDMA reloads the contents of this field into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER must be 0x0001. |

### 14.6.2.20 TCD Beginning Major Loop Count (Minor Loop Channel Linking Enabled) (TCD0_BITER_ELINKYES - TCD31_BITER_ELINKYES)

**Offset**

| Register | Offset |
|---|---|
| TCD0_BITER_ELINKYES | 3Eh |
| TCD1_BITER_ELINKYES | 403Eh |
| TCD2_BITER_ELINKYES | 803Eh |
| TCD3_BITER_ELINKYES | C03Eh |
| TCD4_BITER_ELINKYES | 1_003Eh |
| TCD5_BITER_ELINKYES | 1_403Eh |
| TCD6_BITER_ELINKYES | 1_803Eh |
| TCD7_BITER_ELINKYES | 1_C03Eh |
| TCD8_BITER_ELINKYES | 2_003Eh |
| TCD9_BITER_ELINKYES | 2_403Eh |
| TCD10_BITER_ELINKYES | 2_803Eh |
| TCD11_BITER_ELINKYES | 2_C03Eh |
| TCD12_BITER_ELINKYES | 20_003Eh |
| TCD13_BITER_ELINKYES | 20_403Eh |
| TCD14_BITER_ELINKYES | 20_803Eh |
| TCD15_BITER_ELINKYES | 20_C03Eh |
| TCD16_BITER_ELINKYES | 21_003Eh |
| TCD17_BITER_ELINKYES | 21_403Eh |

*Table continues on the next page...*

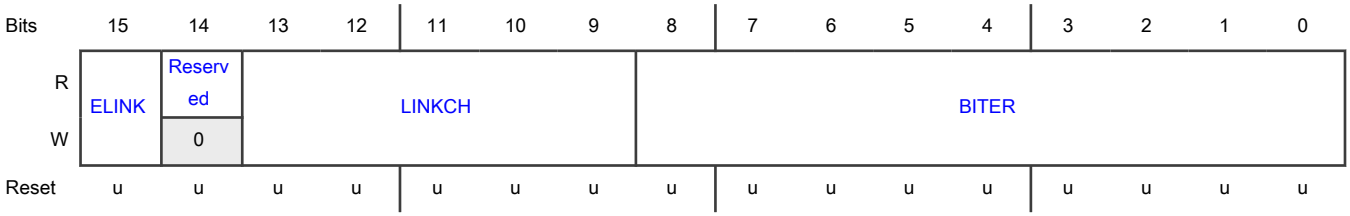
*Table continued from the previous page...*

| Register | Offset |
|---|---|
| TCD18_BITER_ELINKYES | 21_803Eh |
| TCD19_BITER_ELINKYES | 21_C03Eh |
| TCD20_BITER_ELINKYES | 22_003Eh |
| TCD21_BITER_ELINKYES | 22_403Eh |
| TCD22_BITER_ELINKYES | 22_803Eh |
| TCD23_BITER_ELINKYES | 22_C03Eh |
| TCD24_BITER_ELINKYES | 23_003Eh |
| TCD25_BITER_ELINKYES | 23_403Eh |
| TCD26_BITER_ELINKYES | 23_803Eh |
| TCD27_BITER_ELINKYES | 23_C03Eh |
| TCD28_BITER_ELINKYES | 24_003Eh |
| TCD29_BITER_ELINKYES | 24_403Eh |
| TCD30_BITER_ELINKYES | 24_803Eh |
| TCD31_BITER_ELINKYES | 24_C03Eh |

### Function

If the TCDn_BITER[ELINK] field is set, the TCDn_BITER register is defined as follows.

### Diagram




### Fields

| Field | Function |
|---|---|
| 15<br>ELINK | Enable Link<br>As the channel completes the minor loop, this flag enables linking to another channel as defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] field of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking.<br>**NOTE**<br>When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, eDMA reloads the contents of this field into the CITER field.<br>0b - Channel-to-channel linking disabled<br>1b - Channel-to-channel linking enabled |
| 14<br>— | Reserved |
| 13-9<br>LINKCH | Link Channel Number<br>If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] field.<br>**NOTE**<br>When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, eDMA reloads the contents of this field into the CITER field. |
| 8-0<br>BITER | Starting Major Iteration Count<br>As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be set equal to the value in the CITER field. As the major iteration count is exhausted, eDMA reloads the contents of this field into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER must be 0x0001. |

## 14.7 Glossary

**TCD** Transfer control descriptor

# Chapter 15
# Interrupt Monitor (INTM)

## 15.1 Overview

INTM provides a mechanism to monitor the latency of the responses on interrupt requests to ensure that the processing of these critical interrupts executes within the expected time frame, increasing the reliability of the device.

### 15.1.1 Block diagram

The following block diagram shows the major registers involved in monitoring the interrupt sources.

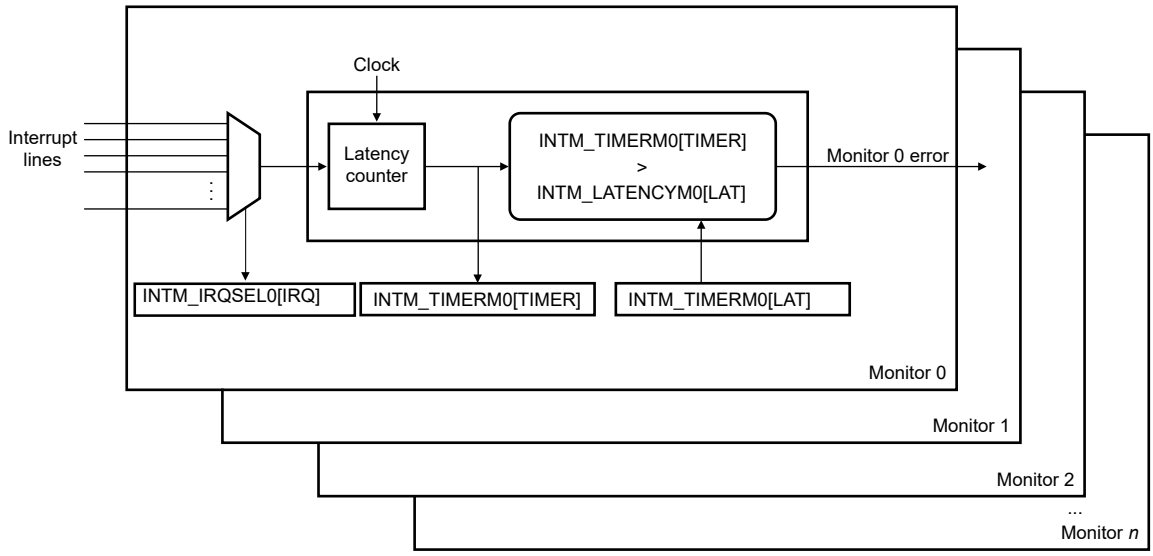



Figure 34. INTM block diagram

### 15.1.2 Features

INTM supports the following features:

- 4 programmable interrupt monitors
- Programmable interrupt source per monitor
- Programmable 24-bit maximum latency counter per monitor
- Timer expired status bit per monitor
- One interrupt acknowledge for all monitors

## 15.2 Functional description

Interrupt Request Select for Monitor a (INTM_IRQSEL0 - INTM_IRQSEL3) provides selection of the source, Monitor Mode (INTM_MM) enables the monitor, Interrupt Acknowledge (INTM_IACK) captures the event when the interrupt is acknowledged, Interrupt Latency for Monitor a (INTM_LATENCY0 - INTM_LATENCY3) can be programmed to trigger a monitor error when a threshold value is exceeded, Status for Monitor a (INTM_STATUS0 - INTM_STATUS3) is available to read the monitor error over peripheral bus.

**NOTE**

INTM supports only level-type interrupts. Hence, the pulse-type interrupts are converted into level-type interrupts for INTM.

### 15.2.1 Clocking

This module has no clocking considerations.

### 15.2.2 Interrupts

This module has no interrupts.

## 15.3 External signals

This module has no external signals.

## 15.4 Initialization

To monitor a subset of interrupt sources:

1. Program Interrupt Request Select for Monitor a (INTM_IRQSEL0 - INTM_IRQSEL3) with a value corresponding to the interrupt source number to be monitored.
   - You can monitor only defined interrupt sources.
2. Program Interrupt Latency for Monitor a (INTM_LATENCY0 - INTM_LATENCY3) to the maximum expected latency time for each monitored interrupt source.
3. Write 1 to INTM_MM[MM] to monitor the registers.
4. During the interrupt service routine, write the IRQ of the ISR to Interrupt Acknowledge (INTM_IACK).
5. If the maximum expected latency time exceeds the limit defined in Interrupt Latency for Monitor a (INTM_LATENCY0 - INTM_LATENCY3), then an error indication is sent to Fault Collection Control Unit(FCCU). You can also read Status for Monitor a (INTM_STATUS0 - INTM_STATUS3) to see this information.

To clear an error condition, write 0 to the corresponding INTM_TIMERa[TIMER] field for which the interrupt source exceeded the programmed latency value. In case the source is unknown, NXP recommends you to write 0 to all INTM_TIMERa[TIMER] fields.

## 15.5 INTM register descriptions

INTM allows you to set a maximum timer count for the interrupt latency from interrupt request to interrupt acknowledge. This mechanism monitors the latency of interrupt sources to ensure that these critical interrupts execute within the expected time frame, increasing the reliability of the chip. The hardware for this function is isolated in its own hierarchy to decrease the risk of fault interference.

An error indication is sent to FCCU if the interrupt processing latency exceeds the programmed threshold of the expected latency. The error indication can be read from INTM_STATUSa[STATUS].

### 15.5.1 INTM memory map

INTM base address: 4027_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Monitor Mode (INTM_MM) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 4h | Interrupt Acknowledge (INTM_IACK) | 32 | W | 0000_0000h |
| 8h | Interrupt Request Select for Monitor 0 (INTM_IRQSEL0) | 32 | RW | 0000_0000h |
| Ch | Interrupt Latency for Monitor 0 (INTM_LATENCY0) | 32 | RW | 0000_0000h |
| 10h | Timer for Monitor 0 (INTM_TIMER0) | 32 | RW | 0000_0000h |
| 14h | Status for Monitor 0 (INTM_STATUS0) | 32 | R | 0000_0000h |
| 18h | Interrupt Request Select for Monitor 1 (INTM_IRQSEL1) | 32 | RW | 0000_0000h |
| 1Ch | Interrupt Latency for Monitor 1 (INTM_LATENCY1) | 32 | RW | 0000_0000h |
| 20h | Timer for Monitor 1 (INTM_TIMER1) | 32 | RW | 0000_0000h |
| 24h | Status for Monitor 1 (INTM_STATUS1) | 32 | R | 0000_0000h |
| 28h | Interrupt Request Select for Monitor 2 (INTM_IRQSEL2) | 32 | RW | 0000_0000h |
| 2Ch | Interrupt Latency for Monitor 2 (INTM_LATENCY2) | 32 | RW | 0000_0000h |
| 30h | Timer for Monitor 2 (INTM_TIMER2) | 32 | RW | 0000_0000h |
| 34h | Status for Monitor 2 (INTM_STATUS2) | 32 | R | 0000_0000h |
| 38h | Interrupt Request Select for Monitor 3 (INTM_IRQSEL3) | 32 | RW | 0000_0000h |
| 3Ch | Interrupt Latency for Monitor 3 (INTM_LATENCY3) | 32 | RW | 0000_0000h |
| 40h | Timer for Monitor 3 (INTM_TIMER3) | 32 | RW | 0000_0000h |
| 44h | Status for Monitor 3 (INTM_STATUS3) | 32 | R | 0000_0000h |

### 15.5.2 Monitor Mode (INTM_MM)

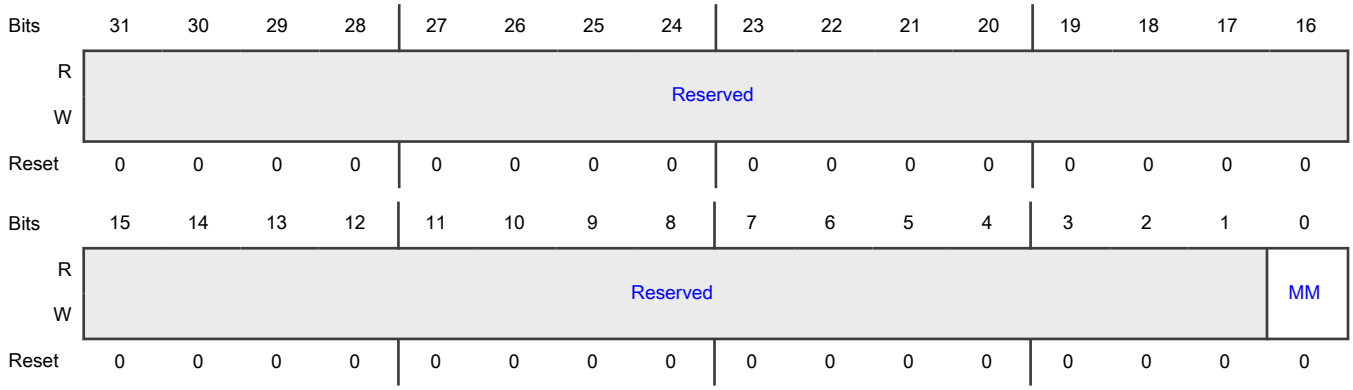
#### Offset

| Register | Offset |
|---|---|
| INTM_MM | 0h |

#### Function

Enables the cycle count timer on a monitored interrupt request for comparison to the latency register.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>MM | Monitor Mode<br>Controls whether the INTM monitors the latency of an interrupt response.<br>0b - Disable<br>1b - Enable |

## 15.5.3 Interrupt Acknowledge (INTM_IACK)

### Offset

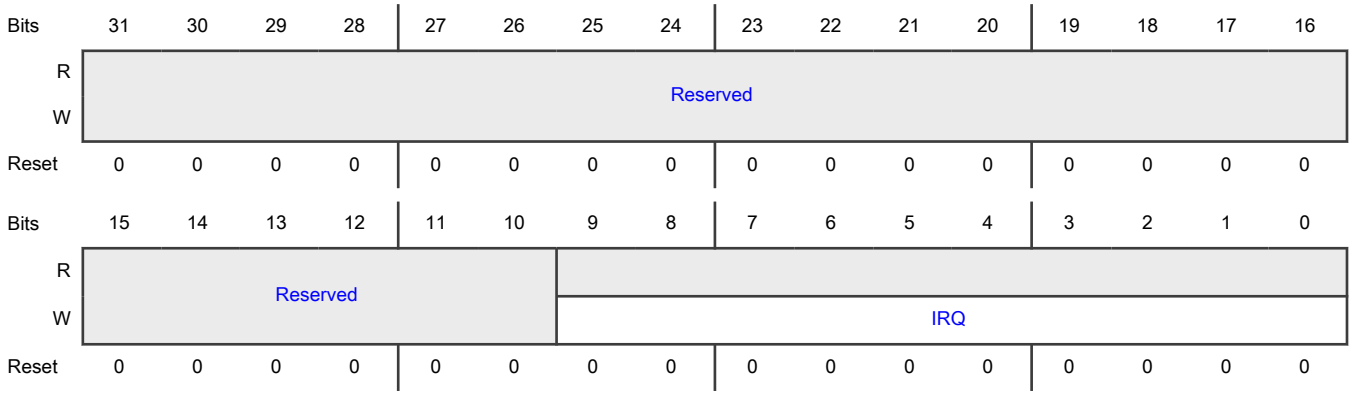
| Register | Offset |
|---|---|
| INTM_IACK | 4h |

### Function

Acknowledges the interrupt processing.

The Interrupt service routine writes to this register after the register acknowledges interrupt processing. This write operation must provide the number of the processed interrupt, which is compared with the interrupt request number in each Interrupt Request Select for Monitor a (INTM_IRQSEL0 - INTM_IRQSEL3) and stops the corresponding Timer for Monitor a (INTM_TIMER0 - INTM_TIMER3) when it found a match.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9-0<br>IRQ | Interrupt Request<br>Specifies the interrupt request number to stop Timer for Monitor a (INTM_TIMER0 - INTM_TIMER3). |

## 15.5.4 Interrupt Request Select for Monitor a (INTM_IRQSEL0 - INTM_IRQSEL3)

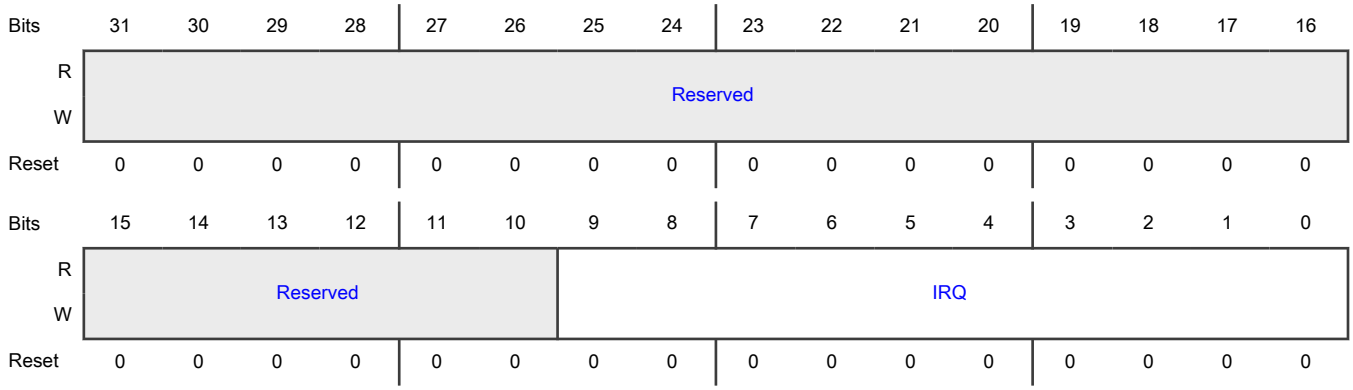
### Offset

| Register | Offset |
|---|---|
| INTM_IRQSEL0 | 8h |
| INTM_IRQSEL1 | 18h |
| INTM_IRQSEL2 | 28h |
| INTM_IRQSEL3 | 38h |

### Function

Indicates which interrupt request must be monitored or checked.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9-0<br>IRQ | Interrupt Request<br>Selects the interrupt request number to monitor. |

## 15.5.5 Interrupt Latency for Monitor a (INTM_LATENCY0 - INTM_LATENCY3)

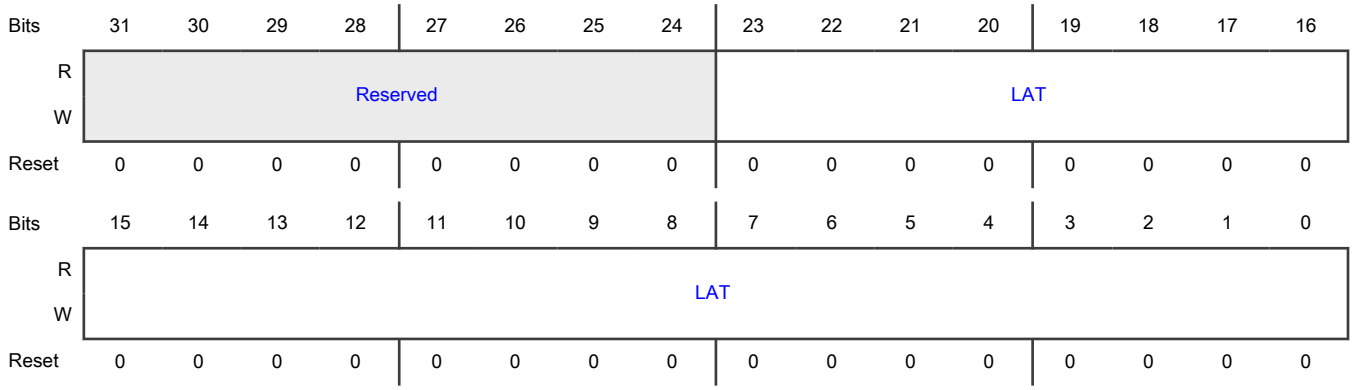
### Offset

| Register | Offset |
|---|---|
| INTM_LATENCY0 | Ch |
| INTM_LATENCY1 | 1Ch |
| INTM_LATENCY2 | 2Ch |
| INTM_LATENCY3 | 3Ch |

### Function

Indicates the maximum time before an error signal is asserted for a detected interrupt request to interrupt acknowledge.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>LAT | Latency<br>Specifies the maximum number of INTM clock cycles allowed for the monitored interrupt request.<br>Maximum programmed value must not exceed FF_FFFDh to allow for proper timer error capture. |

## 15.5.6 Timer for Monitor a (INTM_TIMER0 - INTM_TIMER3)

### Offset

| Register | Offset |
|---|---|
| INTM_TIMER0 | 10h |
| INTM_TIMER1 | 20h |
| INTM_TIMER2 | 30h |
| INTM_TIMER3 | 40h |

### Function

Counts the number of INTM clock cycles for a detected interrupt request to an acknowledged interrupt processing.

Initializes to 1 in following conditions:

- Interrupt Latency for Monitor a (INTM_LATENCY0 - INTM_LATENCY3) is nonzero
- Monitor error is not asserted
- Zero to one transition on the monitored interrupt request

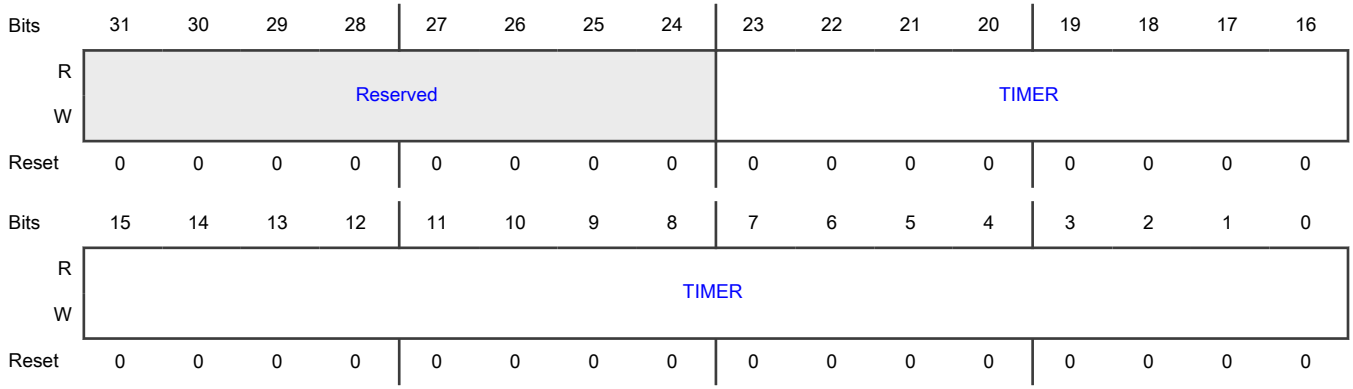
Following are the conditions to enable Timer for Monitor a (INTM_TIMER0 - INTM_TIMER3):

- INTM_MM = 1 and zero-to-one transition on the monitored interrupt source

Following are the conditions to disable Timer for Monitor a (INTM_TIMER0 - INTM_TIMER3):

- Reset
- INTM_MM = 1 and monitor error
- INTM_MM = 1 and monitored interrupt request is not asserted
- Interrupt acknowledge for corresponding interrupt request
- After the timer is enabled, writing 0 to INTM_MM[MM] has no effect.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>TIMER | Timer<br>Counts the number of INTM clock cycles up to 24 bits of resolution. |

## 15.5.7 Status for Monitor a (INTM_STATUS0 - INTM_STATUS3)

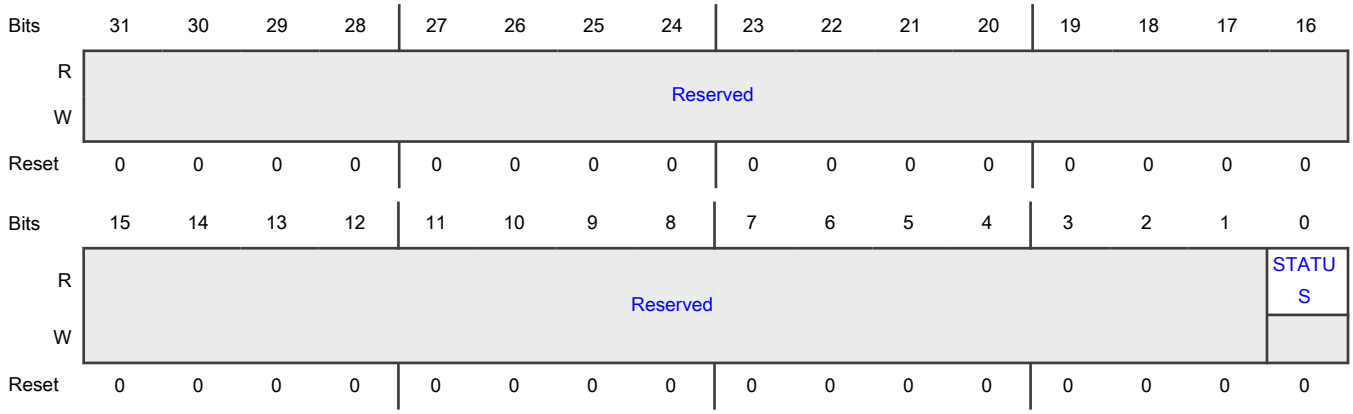
### Offset

| Register | Offset |
|---|---|
| INTM_STATUS0 | 14h |
| INTM_STATUS1 | 24h |
| INTM_STATUS2 | 34h |
| INTM_STATUS3 | 44h |

### Function

Defines the monitor status. Indicates whether Timer for Monitor a (INTM_TIMER0 - INTM_TIMER3) value has exceeded Interrupt Latency for Monitor a (INTM_LATENCY0 - INTM_LATENCY3) value. You can clear the monitor status by writing 0 to the corresponding INTM_TIMERa[TIMER].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>STATUS | Monitor status<br>Indicates whether Timer for Monitor a (INTM_TIMER0 - INTM_TIMER3) value has exceeded the Interrupt Latency for Monitor a (INTM_LATENCY0 - INTM_LATENCY3) value.<br>0b - Did not exceed<br>1b - Exceeded |

# Chapter 16
# Semaphores2 (SEMA42)

## 16.1 Chip-specific SEMA42 information

### 16.1.1 SEMA42 instance and configuration

This chip has up to one instance of SEMA42. See the following table for availability of SEMA42 instance across MCXE31 product series.

**Table 51. SEMA42 instance**

| Instance | MCXE31B |
|---|---|
| SEMA42 | Yes |

See "Chip-specific XRDC information" for the domain ID of each master on the MCXE31 product series. The following table shows the number of domains per chip:

## 16.2 Overview

SEMA42 is a memory-mapped module that provides robust hardware support needed in multi-core systems for implementing semaphores and provides a simple mechanism to achieve "lock and unlock" operations via a single - write access. The hardware semaphore module provides hardware-enforced gates as well as other useful system functions related to the gating mechanisms.

**NOTE**

Terminology in this chapter has been updated as follows:

**Table 52. Updated terms**

| Updated term | Deprecated term |
|---|---|
| Initiator | Master |
| Target | Slave |
| Stop | Abort |

### 16.2.1 Block diagram

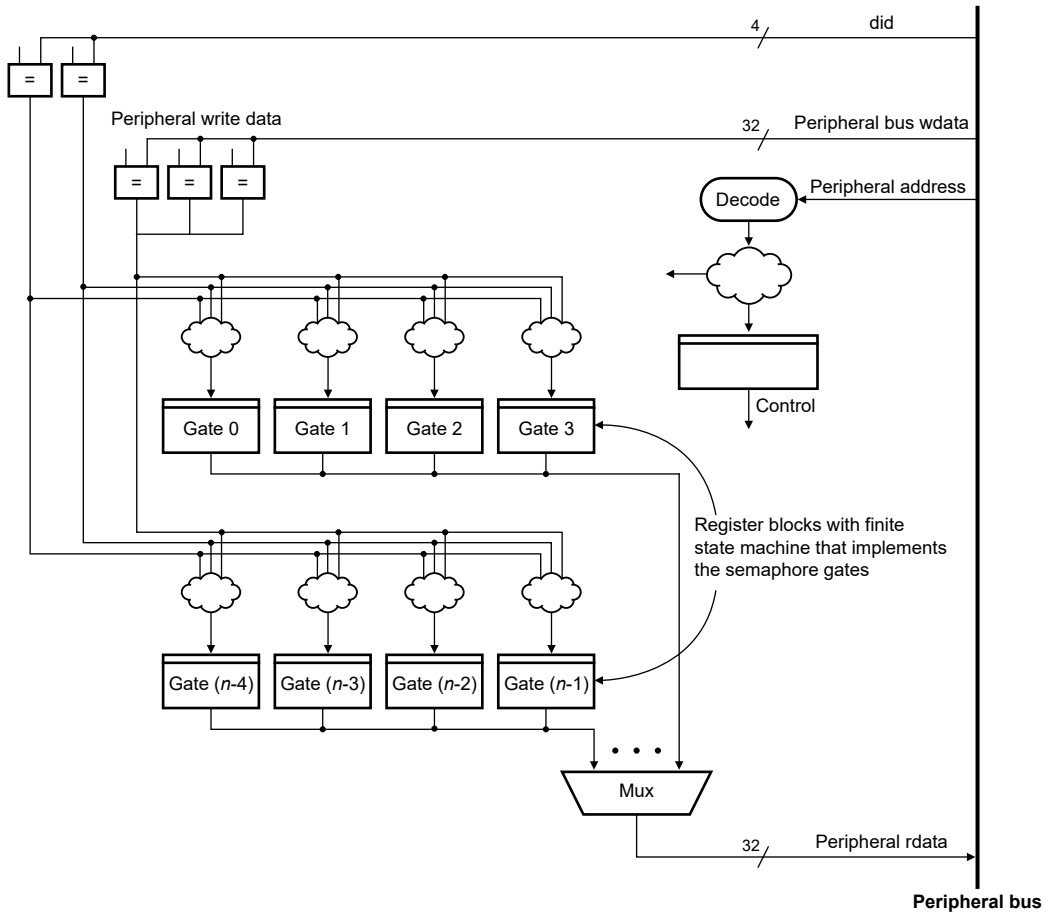



Figure 35. SEMA42 block diagram

### 16.2.2 Features

SEMA42 implements hardware-enforced semaphores as an IPS-mapped target peripheral device. The feature set includes:

- Support for 16 hardware-enforced gates in a multi-domain configuration that supports up to 15 domains.
- Each hardware gate appears as a 16-state, 4-bit state machine.
- Support for secure reset mechanisms to clear the contents of individual gates, as well as a clear-all capability.
- Memory-mapped target peripheral that offers programming-model accesses.

## 16.3 Functional description

The intent of the hardware gate implementation is to specify a protocol where the locking domain must unlock the gate. However, some systems may require a reset function to re-initialize the state of any gate(s) without requiring a system-level reset. To support this special gate reset requirement, SEMA42 implements a secure reset mechanism that allows you to initialize a hardware gate (or all the gates) by following a specific dual-write access pattern. The secure-gate reset:

- Uses a technique similar to that required for the servicing of a software watchdog timer
- Requires two consecutive writes with pre-defined data patterns from the same domain

You must do this to force the clearing of the specified gate(s). The required access pattern as follows:

1. A domain performs a 16-bit write to the RSTGT memory location. The most significant byte ( RSTGT_W[RSTGDP] ) must be E2h. The value of least significant byte is irrelevant for this reference and can be anything.
2. The same domain then performs a second 16-bit write to the RSTGT location. For this write, the upper byte ( RSTGT_W[RSTGDP] ) is the logical complement of the first data pattern (1Dh) and the lower byte (RSTGT_W[RSTGTN] ) specifies the gate(s) to be reset. This gate field can specify a single gate or all gates to be cleared. If the same domain writes incorrect data on the second access or another domain performs the second write access, SEMA42 stops the special gate reset sequence and does not assert an error signal.
3. Reads of the RSTGT location return information on the 2-bit reset state machine (RSTGT_R[RSTGSM] ) that implements these functions:
   - The domain performing the reset ( RSTGT_R[RSTGMS] )
   - The last-cleared gate number(s) (RSTGT_R[RSTGTN] )

   Reads of the RSTGT register do not affect the secure-reset finite state machine in any manner.

### 16.3.1 Multi-core programming: software gates

Multi-processor systems require a function that can be used to safely and easily provide a locking mechanism for system software to control access to shared data structures, shared hardware resources, and so on. The software uses the gating mechanisms to serialize (and synchronize) accesses to shared data and/or resources to prevent race conditions and preserve memory coherency between different processes and domains.

Consider the following description of a typical use-case: dm*X* enters a section of code, where shared data values are to be updated. The domain must first acquire a semaphore. Think of this as the locking (or closing) of a software gate. After the gate locks, a properly-architected software system does not allow other processes (or domains) to execute the same code segment or modify the shared data structure protected by the gate. In other words, the system locks out other processes/domains. Many software implementations include a spin-wait loop within the lock function until the gate locks. After domain X obtains the lock, domain X continues execution and updates the data values protected by the particular lock. After domain X completes the updates, it unlocks (or opens) the software gate, allowing other processes/domains access to the updated data values.

A correctly-implemented system solution must follow these important rules:

- A gate variable must protect all writes to shared data values or shared hardware resources.
- After a domain locks a gate, the system must block other processes/domains from accessing the shared data or resources. Software conventions enforce this.
- The domain that locks a particular gate is the only domain that can open (unlock) that gate.

Information in the hardware gate identifying the locking domain can be extremely useful for system-level debugging.

### 16.3.2 16 Hardware-enforced gates

Gates appear as a 16-entry byte-size array with read and write accesses.

Domains lock gates by writing "domainID_number+1" to the appropriate gate and must read back the gate value to verify that the lock operation succeeded.

After the gate locks, the locking domain unlocks the gate by writing zeroes.

- 16-state implementation

  If gate = 0h, then state = unlocked

  if gate = 1h, then state = locked by domain (initiator) 0

  if gate = 2h, then state = locked by domain (initiator) 1

  …

  if gate = Fh, then state = locked by domain (initiator) 14

SEMA42 uses the logical domain number and the specified data patterns as reference attributes to validate all write operation.

After a gate locks, the locking domain must unlock the gate by writing zeroes.

### 16.3.3 State machine of the GATE*n* registers

This section describes more about the SEMA42 functional operation and specific details of the state machines of the GATE*n* registers.

As described previously, each of the GATE*n* registers implements a 4-bit, 16-state machine. The following figure shows a simplified diagram of the state transitions for each gate.

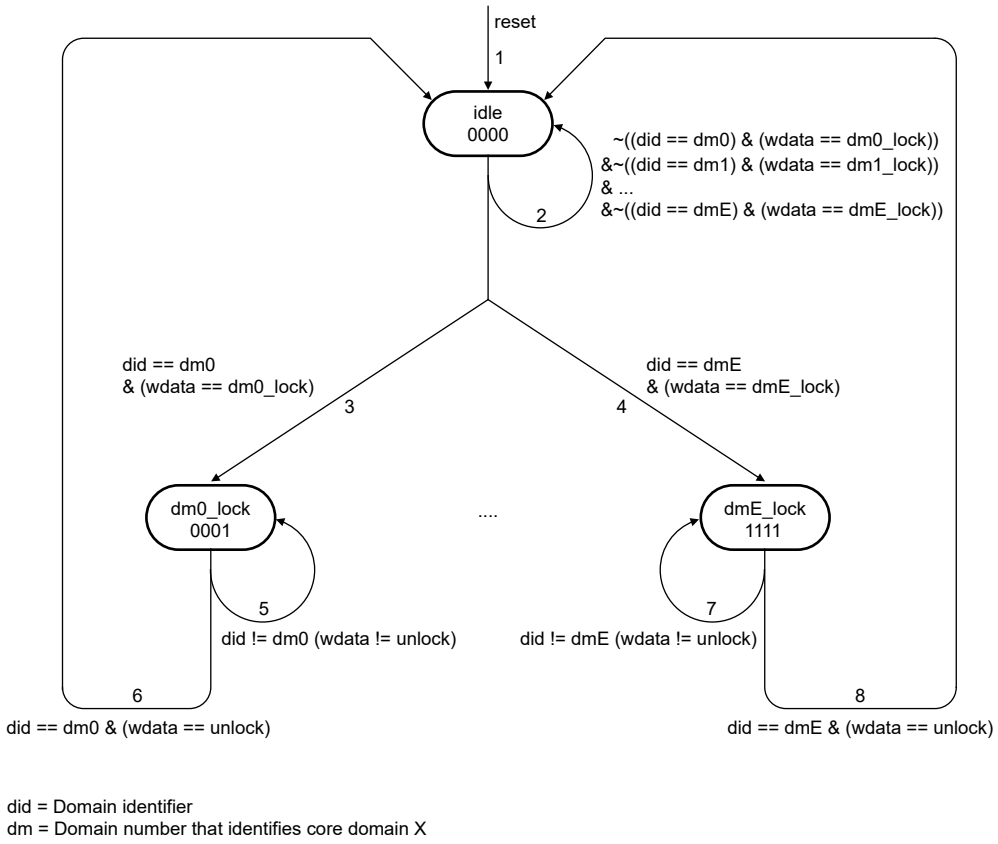



**Figure 36. GATE*n* state machine**

In the figure above, "dmE" represents domain 14 (E in hexadecimal). The platform passes the domain number to SEMA42.

The following table defines the GATE*n* state transitions.

**Table 53. GATE*n* state transitions**

| Current state | Next state | Transition | Description |
|---|---|---|---|
| – | idle | 1 | Any reset, whether a system reset or a software-initiated gate reset, unconditionally forces the gate into the idle state. |
| idle | idle | 2 | Unless a write of the appropriate lock value from the corresponding domain occurs, the gate remains in the idle state. |
| idle | dm0_lock | 3 | When domain 0h initiates a write of the dm0_lock data value, the gate transitions into the dm0_lock state. |

*Table continues on the next page...*

Table 53. GATE*n* state transitions (continued)

| Current state | Next state | Transition | Description |
|---|---|---|---|
| idle | dmE_lock | 4 | When domain Eh initiates a write of the dmE_lock value, the gate transitions into the dmE_lock state. |
| dm0_lock | dm0_lock | 5 | When in this state, the gate remains here if any attempted write is not from domain 0h with the unlock data value. |
| dm0_lock | idle | 6 | The gate returns to the idle (unlocked) state after a write from domain 0h with the unlock data value occurs. |
| dmE_lock | dmE_lock | 7 | When in this state, the gate remains here if any attempted write is not from domain Eh with the unlock data value. |
| dmE_lock | idle | 8 | The gate returns to the idle (unlocked) state after a write from domain Eh with the unlock data value occurs. |

SEMA42 uses these gate data values:

- The lock data value is (domain number) + 1.
- The unlock data value is 00h.

### 16.3.4 Clocking

| Type of clock | Description |
|---|---|
| Module clock (clk) | Functions the module and controls the access to the registers. |

### 16.3.5 Interrupts

This module has no interrupts.

## 16.4 External signals

This module has no external signals.

## 16.5 Initialization

This module does not require initialization.

## 16.6 Memory map/register definition

You can access these registers only in Supervisor mode. User accesses terminate with an error.

### 16.6.1 SEMA42 register descriptions

#### 16.6.1.1 SEMA42 memory map

SEMA42 base address: 4046_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h - Fh | Gate (GATE0 - GATE15) [1] | 8 | RW | 00h |
| 42h | Reset Gate Read (RSTGT_R) | 16 | R | 0000h |
| 42h | Reset Gate Write (RSTGT_W) | 16 | W | See section |

1. In this array, the index and offset values of the registers do not increment in direct alignment. For details, see the register description.

### 16.6.1.2 Gate (GATE0 - GATE15)

#### Offset

For n = 0 to 15:

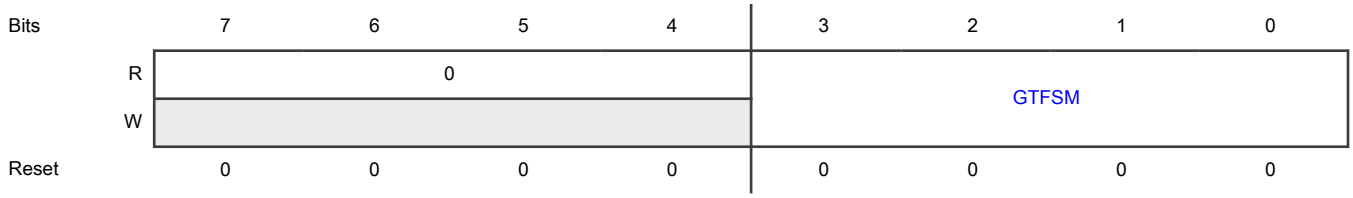
| Register | Offset |
|---|---|
| GATEn | 0h + (n + 3 - 2 × (n mod 4)) |

#### Function

Implements each semaphore gate in a 4-bit finite state machine, right-justified in a byte data structure. The hardware uses the logical domain-identifier number in conjunction with the data patterns to validate all attempted write operations. Only domain initiators can modify the gate registers. After a gate locks, only the locking domain must open (unlock) the gate.

You can read multiple gate values in a single access. However, you can update only a single gate at a time, via a write operation. If you attempt to write a byte-wide value that is neither the unlock value (00h) nor the appropriate lock value (domainID_number + 1), SEMA42 considers this as "no operation" and does not change any gate state. Attempts to write multiple gates in a single-aligned access with a size larger than 8 bits (byte) generate an error termination and do not allow any gate state changes.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 7-4<br>— | Reserved |
| 3-0<br>GTFSM | Gate Finite State Machine<br>Indicates the state of the gate for the last domain that locked the gate. This can be useful during system debug. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | The hardware gate has a 16-state implementation, defined as:<br>0000b - The gate is unlocked (free).<br>0001b - Domain 0 locked the gate.<br>0010b - Domain 1 locked the gate.<br>0011b - Domain 2 locked the gate.<br>0100b - Domain 3 locked the gate.<br>0101b - Domain 4 locked the gate.<br>0110b - Domain 5 locked the gate.<br>0111b - Domain 6 locked the gate.<br>1000b - Domain 7 locked the gate.<br>1001b - Domain 8 locked the gate.<br>1010b - Domain 9 locked the gate.<br>1011b - Domain 10 locked the gate.<br>1100b - Domain 11 locked the gate.<br>1101b - Domain 12 locked the gate.<br>1110b - Domain 13 locked the gate.<br>1111b - Domain 14 locked the gate. |

#### 16.6.1.3 Reset Gate Read (RSTGT_R)

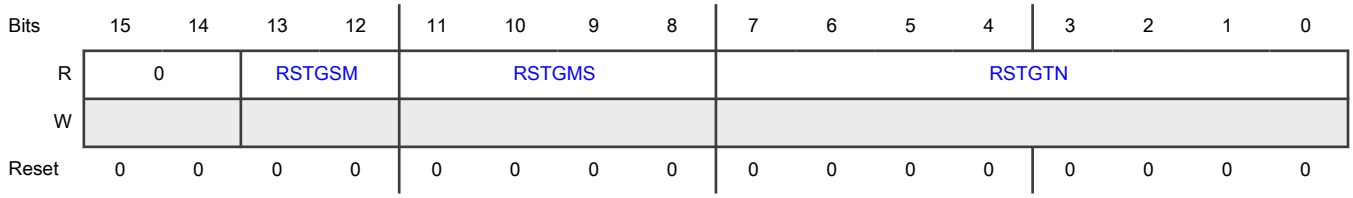
**Offset**

| Register | Offset |
|---|---|
| RSTGT_R | 42h |

**Function**

Describes the specific hardware gate to be reset and records the logic number of domain. Reset Gate Write (RSTGT_W) also describe the same register showing the fields when you write it.

### Diagram




### Fields

| Field | Function |
|---|---|
| 15-14<br>— | Reserved |
| 13-12<br>RSTGSM | Reset Gate Finite State Machine<br>Indicates the encoded state machine value when you read the register. RSTGSM = 10b is valid for only a single machine cycle, so a read can never return this value. SEMA42 maintains the reset state machine in a 2-bit, 3-state implementation, defined as follows:<br>00b - Idle, waiting for the first data pattern write.<br>01b - Waiting for the second data pattern write<br>10b - The 2-write sequence has completed. Generate the specified gate reset(s). After the reset is performed, this machine returns to the idle (waiting for first data pattern write) state.<br>11b - This state encoding is never used and therefore reserved. |
| 11-8<br>RSTGMS | Reset Gate Domain<br>Records the logical number of the domain performing the gate reset function. The logical number is the domain number. This domain number is determined by the XRDC's initiator Domain Assignment Controller (XRDC_MDAC). To succeed, this function requires that the same domain initiate the two consecutive writes to this register. SEMA42 updates the field each time a write to this register occurs. |
| 7-0<br>RSTGTN | Reset Gate Number<br>Specifies the specific hardware gate to be reset. The second write updates this field.<br>If RSTGTN < 64, SEMA42 resets the single gate defined by RSTGTN. Otherwise, SEMA42 resets all the gates. |

## 16.6.1.4 Reset Gate Write (RSTGT_W)

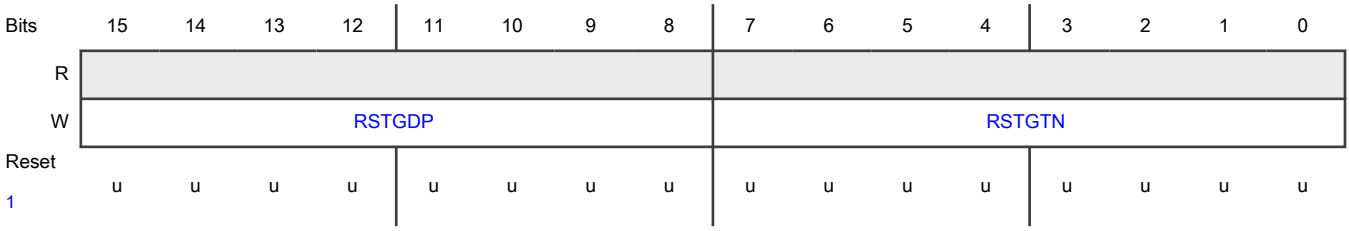
### Offset

| Register | Offset |
|---|---|
| RSTGT_W | 42h |

### Function

Specifies the hardware gate to reset when data patterns are specified.

### Diagram




1. Reset value is not applicable for writes.

### Fields

| Field | Function |
|---|---|
| 15-8<br>RSTGDP | Reset Gate Data Pattern<br>You access this field with the specified data patterns on the two consecutive writes to enable the gate-reset mechanism. For the first write, RSTGDP must be E2h. For the second write, RSTGDP must be 1Dh. |
| 7-0<br>RSTGTN | Reset Gate Number<br>Specifies the specific hardware gate to be reset. The second write updates this field.<br>If RSTGTN < 64, SEMA42 resets the single gate defined by RSTGTN. Otherwise, SEMA42 resets all the gates. |

## 16.7 Glossary

**IPS** Protocol used for peripheral accesses to the programming model

**dm*X*** Domain X

# Chapter 17
# Crossbar Integrity Checker (XBIC)

## 17.1 Chip-specific XBIC information

### 17.1.1 XBIC instances and configuration

This chip has up to five instances of XBIC. The following tables describes the instances and their configuration.

**Table 54. XBIC instances**

| Instance | MCXE31B | MCXE315/ MCXE316/MCXE317 |
|---|---|---|
| XBIC_0 | Yes | Yes |
| XBIC_1 | Yes | No |
| XBIC_2 | Yes | No |
| XBIC_3 | Yes | No |
| XBIC_4 | No | No |

**Table 55. XBIC configuration for MCXE31B**

| Instance | Available on crossbar | Master and slave assignments |
|---|---|---|

XBIC_0 / AXBS_0 (main)

| Master port | Master module | Slave port | Slave module |
|---|---|---|---|
| M0 | Cortex-M7_0 AHBM | S0 | Flash memory port 0 |
| M1 | AXBS_2 S0 | S1 | Flash memory port 1 |
| M2 | ELE_HSEB | S2 | PRAM_0 |
| M3 | EMAC | S3 | Cortex-M7 TCM |
| -[1] | - | S4 | Flash memory Port 2 |
| | | S5 | QuadSPI |
| | | S6 | PRAM_1 |

XBIC_1 / AXBS_1 (peripheral)

| Master port | Master module | Slave port | Slave module |
|---|---|---|---|
| M0 | Cortex-M7_0 AHBP | S0 | AIPS_0 |
| M1 | AXBS_2 S1 | S1 | AIPS_1 |
| M2 | ELE_HSEB | S2 | AIPS_2 |
| M3[1] | Cortex-M7_1 AHBP | | |

1. These ports are reserved for MCXE31B.

Table 55. XBIC configuration for MCXE31B (continued)

| Instance | Available on crossbar | Master and slave assignments | | | | |
|---|---|---|---|---|---|---|
| XBIC_2 | AXBS_2 (eDMA) | | | | | |
| | | **Master port** | **Master module** | | **Slave port** | **Slave module** |
| | | M0 | eDMA | | S0 | AXBS_0 M1 |
| | | | | | S1 | AXBS_1 M1 |
| XBIC_3 | AXBS_3 (Cortex-M7 TCM) | | | | | |
| | | **Master port** | **Master module** | | **Slave port** | **Slave module** |
| | | M0 | AXBS_0 S3 | | S0 | Cortex-M7_0 TCM |
| | | | | | S1[1] | Cortex-M7_1 TCM |

Table 56. XBIC configuration for MCXE317

| Instance | Available on crossbar | Master and slave assignments | | | | |
|---|---|---|---|---|---|---|
| XBIC_0 | AXBS_0 (main) | | | | | |
| | | **Master port** | **Master module** | | **Slave port** | **Slave module** |
| | | M0 | Cortex-M7_0 AHBM | | S0 | Flash memory port 0 |
| | | M1 | eDMA | | S1 | Flash memory port 1 |
| | | M2 | ELE_HSEB | | S2 | PRAM_0 |
| | | M3 | Cortex-M7_0 AHBP | | S3 | Cortex-M7 TCM |
| | | M4 | Reserved | | S4 | AIPS_0 |
| | | | | | S5 | AIPS_1 |

The errors detected are connected to FCCU. See the memory map file attached to this document for details.

### 17.1.2 Target master IDs

See "Chip-specific XRDC information" for master IDs.

## 17.2 Overview

XBIC is a safety module that verifies the integrity of crossbar transfers.

### 17.2.1 Block diagram

The chip routes crossbar transfer attribute information for all mapped initiator and target ports to XBIC, which calculates and checks the EDC value of the attribute information, as shown in the following diagram.

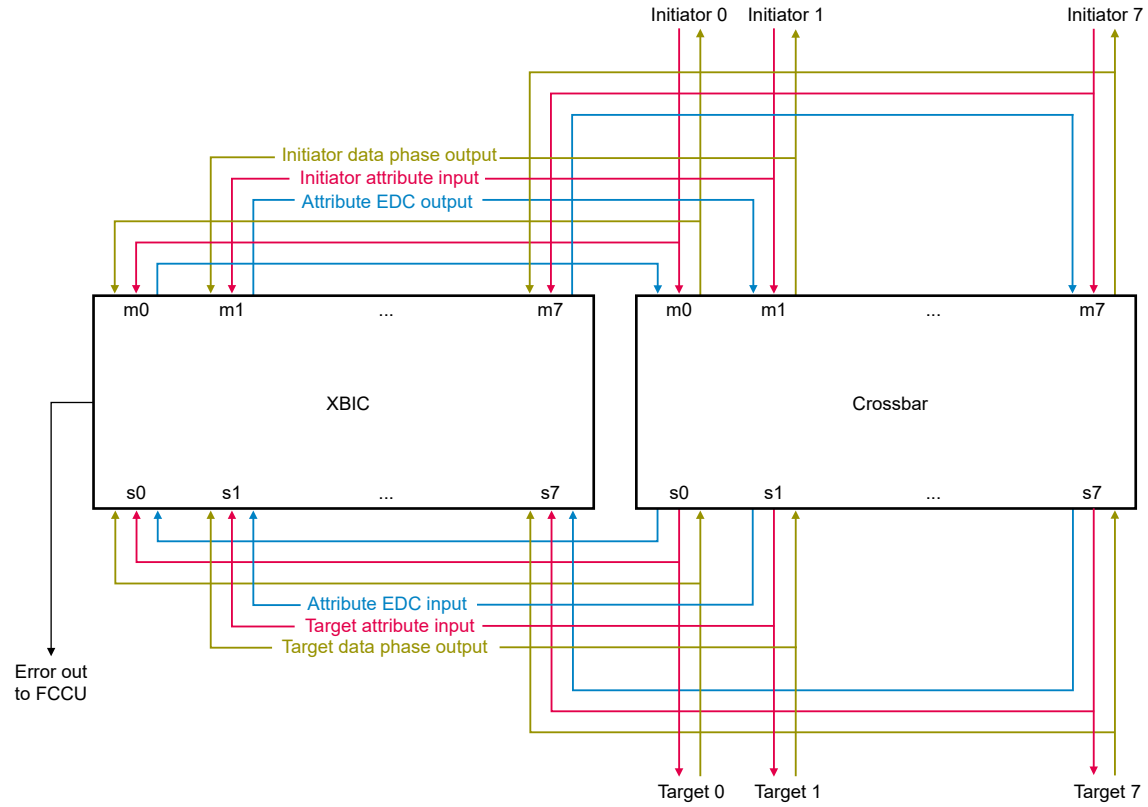



Figure 37. XBIC system block diagram

The above figure illustrates one of many possible XBIC and crossbar configurations. See the Chip-specific XBIC information section for actual port mappings.

### 17.2.2 Features

XBIC includes the following features:

- Verification of attribute information for crossbar transfers [1],[2]
  - EDC detects single-bit and double-bit errors
- Verification of feedback information for each data phase during crossbar transfers
- Error injection for testing
  - Programmable initiator and target port specifiers
  - Programmable 8-bit toggle vector to insert error in initiator EDC value
  - Address, EDC syndrome, initiator, and target port information are captured when error is detected
- Crossbar transfer attribute integrity check programmable on a per-target-port basis
- Feedback integrity check programmable on a per-initiator-port basis

## 17.3 Functional description

XBIC verifies the integrity of the crossbar interface on an individual port basis according to the configuration specified via the MCR register. When XBIC detects an error, it reports relevant information and sends an error signal to the FCCU module, but does

[1] See Table 57 for a list of crossbar attribute signals verified.
[2] The chip verifies read and write data separately, via the end-to-end ECC architecture.

not generate a bus error. XBIC integrity checking is independent of end-to-end ECC, which monitors the integrity of the transfer address and data.

During the address phase of a transfer, XBIC verifies the crossbar attribute information using an 8-bit EDC, which detects any single-bit or double-bit errors. When XBIC detects an error (an attribute integrity error), due to either hardware fault or error injection, it reports information related to the error in ESR and EAR.

During the data phase of a data transfer, XBIC verifies the integrity of response signals from target to intiator as they pass through the crossbar. When XBIC detects an error in the response signals (a feedback integrity error), it reports the XBIC target and intiator ports in ESR[DPSE0] - ESR[DPSE7] and ESR[DPME0] - ESR[DPME7], respectively.

During the data phase, XBIC sends an alarm to the FCCU module if a intiator port attempts back-to-back accesses in which:

1. The first access terminates normally.
2. The second attempts access to an address space not mapped to any target on the crossbar.

The resultant 'absent target error' causes the crossbar to generate a bus error response to the requesting intiator. XBIC detects the bus error response as a difference because the bus error did not originate from the target port.

You can program XBIC to inject EDC errors for testing purposes. Error injection targets a single target port and a single intiator port, as specified by the configuration settings in the EIR register. When XBIC inserts an error, it changes the EDC syndrome, causing the XBIC to assert an error indication to the FCCU module. Otherwise, transfers are unaffected by error injection. This enables verification of the check logic without compromising the integrity of the data transfer. After you enable error injection function by writing 1 to EIR[EIE], XBIC induces errors on all subsequent targeted transactions until you write a 0 to the field. After the FCCU error indication asserts, it remains asserted even after you write 0 to EIR[EIE]. The error indication deasserts after FCCU specifically clears the existing error. After FCCU clears the XBIC error, additional error injection testing can continue.

To trace a fault reported by the XBIC to the FCCU:

1. Note the error reported by the FCCU. For example, "NCF[46]".
2. Locate the source module and error description in the attached fault mapping spreadsheet. For example:
   - Channel number: NCF[46]
   - Source module: AXBS_1 integrity checker
   - Description: Instruction crossbar error indication to FCCU if syndrome calculated using EDC on the data is not zero
3. Refer to the Chip-specific XBIC information section for the source XBIC module to determine the specific XBIC module instanceâ€""XBIC_1", for example.
4. Determine the type of error (attribute integrity error or feedback integrity error) and the XBIC port(s) involved by reading the information reported in the ESR register of the reported XBIC instance.
5. For attribute integrity check errors, read the XBIC EAR register for the target address of the requested transfer.
6. Refer to the Chip-specific XBIC information section and possibly the Chip-specific AXBS information section for port mapping of XBIC ports to AXBS ports.

Decode a single-bit error syndrome value reported in ESR[SYN] by finding the value in the following table. Any syndrome value not included in the table indicates a multi-bit error.

**Table 57. Hexadecimal attribute single-bit error syndromes**

| Signal | SYN | Signal | SYN | Signal | SYN | Signal | SYN |
|---|---|---|---|---|---|---|---|
| hwrite | 07 | hbstrb[7] | 70 | hdecor[31] | 25 | hdecor[15] | 23 |
| htrans[0] | 0b | hbstrb[6] | 32 | hdecor[30] | 68 | hdecor[14] | 51 |
| hsize[2] | 0d | hbstrb[5] | 52 | hdecor[29] | c7 | hdecor[13] | 54 |

*Table continues on the next page...*

Table 57. Hexadecimal attribute single-bit error syndromes (continued)

| Signal | SYN | Signal | SYN | Signal | SYN | Signal | SYN |
|---|---|---|---|---|---|---|---|
| hsize[1] | 0e | hbstrb[4] | a8 | hdecor[28] | 83 | hdecor[12] | 61 |
| hsize[0] | 13 | hbstrb[3] | 43 | hdecor[27] | 85 | hdecor[11] | e3 |
| hprot[5] | 15 | hbstrb[2] | 45 | hdecor[26] | 86 | hdecor[10] | e6 |
| hprot[4] | 16 | hbstrb[1] | 4c | hdecor[25] | 89 | hdecor[9] | f8 |
| hprot[3] | 19 | hbstrb[0] | a4 | hdecor[24] | 8a | hdecor[8] | 38 |
| hprot[2] | 1a | hmaster[3] | a2 | hdecor[23] | 8c | hdecor[7] | 58 |
| hprot[1] | 1c | hmaster[2] | b0 | hdecor[22] | 49 | hdecor[6] | 37 |
| hprot[0] | 91 | hmaster[1] | c1 | hdecor[21] | 92 | hdecor[5] | f1 |
| hburst[2] | a1 | hmaster[0] | c2 | hdecor[20] | 94 | hdecor[4] | 3b |
| hburst[1] | 64 | hslave[2] | c4 | hdecor[19] | 98 | hdecor[3] | 3d |
| hburst[0] | 29 | hslave[1] | c8 | hdecor[18] | 46 | hdecor[2] | 3e |
| hmastlock | 2a | hslave[0] | d0 | hdecor[17] | 34 | hdecor[1] | 4f |
| hunalign | 2c | hdecorated | e0 | hdecor[16] | 4a | hdecor[0] | 6e |
| edc[7] | 80 | edc[6] | 40 | edc[5] | 20 | edc[4] | 10 |
| edc[3] | 08 | edc[2] | 04 | edc[1] | 02 | edc[0] | 01 |

### 17.3.1 Interrupts

This module has no interrupts.

## 17.4 External signals

XBIC has no external interface signals.

## 17.5 Initialization

This module does not require initialization.

## 17.6 XBIC register descriptions

The XBIC programming model consists of four 32-bit registers. Software can access this model only in supervisor mode using 32-bit (word) accesses. Each of the following generates a transfer error back to the requesting intiator. Such errors could cause core exceptions apart from other problems.

- Access size other than 32-bit
- Access to an undefined (reserved) address
- Access in user mode

### 17.6.1 XBIC memory map

XBIC_AXBS base address: 4020_4000h

XBIC_AXBS_PERI base address: 4020_8000h

XBIC_AXBS_TCM base address: 4040_0000h

XBIC_AXBS_eDMA base address: 4040_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | XBIC Module Control (MCR) | 32 | RW | FFFF_0000h |
| 4h | XBIC Error Injection Attributes (EIR) | 32 | RW | 0000_0000h |
| 8h | XBIC Error Status Attributes (ESR) | 32 | RW | 0000_0000h |
| Ch | XBIC Error Address (EAR) | 32 | R | 0000_0000h |

### 17.6.2 XBIC Module Control (MCR)

**Offset**

| Register | Offset |
|---|---|
| MCR | 0h |

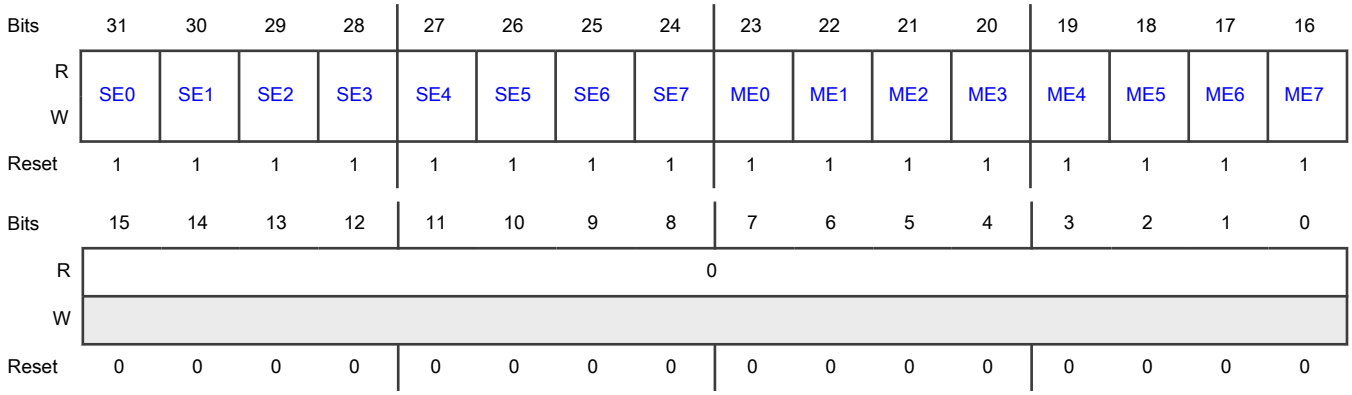
**Function**

Use this register to turn attribute integrity checking and feedback integrity checking on or off on a per-port basis.

- Turn on attribute integrity checking on one or more XBIC target ports by ensuring that the associated SE*n* field(s) have a value of 1. For example, setting field SE*0* enables attribute integrity checking for target port 0. The default (reset) behavior is for attribute integrity checking to be performed for all target ports. XBIC performs EDC-based checks on all transfer requests targeting the selected target port(s). The signals verified are transfer attribute signals going from initiator to target. When XBIC detects an attribute integrity error, it reports relevant information in the ESR and EAR registers.
- Turn on feedback integrity checking on one or more XBIC initiator ports by ensuring that the associated ME*n* field(s) have a value of 1. For example, setting field ME*0* to 1 enables feedback integrity checking for initiator port 0. The default (reset) behavior is for feedback integrity checking to be performed for all initiator ports. XBIC checks target-to-initiator feedback signals for transfer requests originating from the selected XBIC initiator port(s). If any feedback signal value is different at the initiator and target ports during the data phase, XBIC reports the relevant initiator and target ports in the ESR register.

Each field in this register references a specific initiator or target port using XBIC port numbering. Referring to Figure 37, "target port 0" refers to XBIC port "s0" in the figure, "target port 1" refers to port "s1", and so on. Similarly, "initiator port 0" refers to XBIC port "m0", "initiator port 1" refers to port "m1", and so on. See the "Chip-specific XBIC information" section in this document for the mapping of XBIC instances to AXBS instances and XBIC ports to AXBS ports. See the "Chip-specific AXBS information" section in this document for the device component(s) mapped to each port of an AXBS instance.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>SE0 | target port EDC Error Detection Enable<br>0b - Attribute integrity checking disabled for target port 0<br>1b - Attribute integrity checking enabled for target port 0 |
| 30<br>SE1 | target port EDC Error Detection Enable<br>0b - Attribute integrity checking disabled for target port 1<br>1b - Attribute integrity checking enabled for target port 1 |
| 29<br>SE2 | target port EDC Error Detection Enable<br>0b - Attribute integrity checking disabled for target port 2<br>1b - Attribute integrity checking enabled for target port 2 |
| 28<br>SE3 | target port EDC Error Detection Enable<br>0b - Attribute integrity checking disabled for target port 3<br>1b - Attribute integrity checking enabled for target port 3 |
| 27<br>SE4 | target port EDC Error Detection Enable<br>0b - Attribute integrity checking disabled for target port 4<br>1b - Attribute integrity checking enabled for target port 4 |
| 26<br>SE5 | target port EDC Error Detection Enable<br>0b - Attribute integrity checking disabled for target port 5<br>1b - Attribute integrity checking enabled for target port 5 |
| 25<br>SE6 | target port EDC Error Detection Enable<br>0b - Attribute integrity checking disabled for target port 6<br>1b - Attribute integrity checking enabled for target port 6 |
| 24 | target Port EDC Error Detection Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| SE7 | 0b - Attribute integrity checking disabled for target port 7<br>1b - Attribute integrity checking enabled for target port 7 |
| 23<br>ME0 | initiator Port Enable For Feedback Integrity Check<br>0b - Feedback integrity checking disabled for initiator port 0<br>1b - Feedback integrity checking enabled for initiator port 0 |
| 22<br>ME1 | initiator Port Enable For Feedback Integrity Check<br>0b - Feedback integrity checking disabled for initiator port 1<br>1b - Feedback integrity checking enabled for initiator port 1 |
| 21<br>ME2 | initiator Port Enable For Feedback Integrity Check<br>0b - Feedback integrity checking disabled for initiator port 2<br>1b - Feedback integrity checking enabled for initiator port 2 |
| 20<br>ME3 | initiator Port Enable For Feedback Integrity Check<br>0b - Feedback integrity checking disabled for initiator port 3<br>1b - Feedback integrity checking enabled for initiator port 3 |
| 19<br>ME4 | initiator Port Enable For Feedback Integrity Check<br>0b - Feedback integrity checking disabled for initiator port 4<br>1b - Feedback integrity checking enabled for initiator port 4 |
| 18<br>ME5 | initiator Port Enable For Feedback Integrity Check<br>0b - Feedback integrity checking disabled for initiator port 5<br>1b - Feedback integrity checking enabled for initiator port 5 |
| 17<br>ME6 | initiator Port Enable For Feedback Integrity Check<br>0b - Feedback integrity checking disabled for initiator port 6<br>1b - Feedback integrity checking enabled for initiator port 6 |
| 16<br>ME7 | initiator Port Enable For Feedback Integrity Check<br>0b - Feedback integrity checking disabled for initiator port 7<br>1b - Feedback integrity checking enabled for initiator port 7 |
| 15-0<br>— | Reserved |

## 17.6.3 XBIC Error Injection Attributes (EIR)

### Offset

| Register | Offset |
|---|---|
| EIR | 4h |

### Function

Use this register to configure the XBIC error injection function and turn it on or off. When enabled, the XBIC error injection function inserts an attribute integrity error when the targeted initiator requests a transaction of the targeted target port. The inserted error changes the calculated EDC syndrome value, causing XBIC to:

- Capture transfer information in the ESR and EAR registers
- Assert an error signal to the FCCU module

Otherwise, XBIC error injection does not affect transfers.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | SLV | | | MST | | | | SYN | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>EIE | Error Injection Enable<br>0b - Error injection disabled<br>1b - Error injection enabled |
| 30-15<br>— | Reserved |
| 14-12<br>SLV | Target Port<br>Specifies the target port for error injection—other target ports are unaffected.<br>Specify the target port by its XBIC target port number (0–7). See the "Chip-specific XBIC information" section in this document for the mapping of XBIC instances to AXBS instances and XBIC ports to AXBS ports. See the "Chip-specific AXBS information" section in this document for the device component(s) mapped to each port of an AXBS instance. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 11-8<br>MST | Target initiator ID<br>Specifies the target initiator port for error injection—transfers with other intiators are unaffected.<br>Specify the target initiator port using the logical initiator ID number of the target bus initiator. See the "Chip-specific XBIC information" section in this document for the initiator IDs and their corresponding components. |
| 7-0<br>SYN | Syndrome<br>XBIC performs an exclusive OR operation on the value in this field and the calculated syndrome, to generate an error with the specified syndrome. A value of zero does not generate an error. See Table 57 for a list of transfer attribute single-bit error syndromes, noting that the values given in the table are hexadecimal. |

### 17.6.4 XBIC Error Status Attributes (ESR)

#### Offset

| Register | Offset |
|---|---|
| ESR | 8h |

#### Function

In this register, XBIC reports information about the most recent transfer with an error detected. If XBIC detects an attribute integrity check error, it reports:

- The target port identifier (SLV)
- The initiator port identifier (MST) and the error syndrome (SYN)

If XBIC detects a mismatch among feedback signals during the data phase:

- The DPSE0 - DPSE7 field with a value of 1 indicates the XBIC target port. In the DPSE0-DPSE7 field descriptions, the target port number refers to the XBIC target port number. Referring to Figure 37, "target port 0" refers to XBIC port "s0" in the figure, "target port 1" refers to port "s1", and so on. See the "Chip-specific XBIC information" section in this document for the mapping of XBIC instances to AXBS instances and XBIC ports to AXBS ports. See the "Chip-specific AXBS information" section in this document for the device component(s) mapped to each port of an AXBS instance.
- The DPME0 - DPME7 field with a value of 1 indicates the XBIC initiator port. In the DPME0-DPME7 field descriptions, the initiator port number refers to the XBIC initiator port number. Referring to Figure 37, "initiator port 0" refers to XBIC port "m0" in the figure, "initiator port 1" refers to port "m1", and so on. See the "Chip-specific XBIC information" section in this document for the mapping of XBIC instances to AXBS instances and XBIC ports to AXBS ports. See the "Chip-specific AXBS information" section in this document for the device component(s) mapped to each port of an AXBS instance.

XBIC sets this register to all 0s only on reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VLD | DPSE 0 | DPSE 1 | DPSE 2 | DPSE 3 | DPSE 4 | DPSE 5 | DPSE 6 | DPSE 7 | DPME 0 | DPME 1 | DPME 2 | DPME 3 | DPME 4 | DPME 5 | DPME 6 |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DPME 7 | SLV | | | MST | | | | SYN | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>VLD | Error Status Valid<br>0b - No error detected—other fields of the ESR and EAR registers are invalid<br>1b - Error detected—all fields of the ESR and EAR registers are valid |
| 30<br>DPSE0 | Data Phase target Port Error<br>0b - No feedback integrity error detected on target port 0<br>1b - Feedback integrity error detected on target port 0 |
| 29<br>DPSE1 | Data Phase target Port Error<br>0b - No feedback integrity error detected on target port 1<br>1b - Feedback integrity error detected on target port 1 |
| 28<br>DPSE2 | Data Phase target Port Error<br>0b - No feedback integrity error detected on target port 2<br>1b - Feedback integrity error detected on target port 2 |
| 27<br>DPSE3 | Data Phase target Port Error<br>0b - No feedback integrity error detected on target port 3<br>1b - Feedback integrity error detected on target port 3 |
| 26<br>DPSE4 | Data Phase target Port Error<br>0b - No feedback integrity error detected on target port 4<br>1b - Feedback integrity error detected on target port 4 |
| 25<br>DPSE5 | Data Phase target Port Error<br>0b - No feedback integrity error detected on target port 5<br>1b - Feedback integrity error detected on target port 5 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 24<br>DPSE6 | Data Phase target Port Error<br>0b - No feedback integrity error detected on target port 6<br>1b - Feedback integrity error detected on target port 6 |
| 23<br>DPSE7 | Data Phase target Port Error<br>0b - No feedback integrity error detected on target port 7<br>1b - Feedback integrity error detected on target port 7 |
| 22<br>DPME0 | Data Phase initiator Port Error<br>0b - No feedback integrity error detected on initiator port 0<br>1b - Feedback integrity error detected on initiator port 0 |
| 21<br>DPME1 | Data Phase initiator Port Error<br>0b - No feedback integrity error detected on initiator port 1<br>1b - Feedback integrity error detected on initiator port 1 |
| 20<br>DPME2 | Data Phase initiator Port Error<br>0b - No feedback integrity error detected on initiator port 2<br>1b - Feedback integrity error detected on initiator port 2 |
| 19<br>DPME3 | Data Phase initiator Port Error<br>0b - No feedback integrity error detected on initiator port 3<br>1b - Feedback integrity error detected on initiator port 3 |
| 18<br>DPME4 | Data Phase initiator Port Error<br>0b - No feedback integrity error detected on initiator port 4<br>1b - Feedback integrity error detected on initiator port 4 |
| 17<br>DPME5 | Data Phase initiator Port Error<br>0b - No feedback integrity error detected on initiator port 5<br>1b - Feedback integrity error detected on initiator port 5 |
| 16<br>DPME6 | Data Phase initiator Port Error<br>0b - No feedback integrity error detected on initiator port 6<br>1b - Feedback integrity error detected on initiator port 6 |
| 15<br>DPME7 | Data Phase initiator Port Error<br>0b - No feedback integrity error detected on initiator port 7<br>1b - Feedback integrity error detected on initiator port 7 |
| 14-12<br>SLV | target Port<br>target port targeted by the most recent transfer with an attribute integrity check error detected. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | The value in this field is the XBIC target port number (0–7). See the "Chip-specific XBIC information" section in this document for the mapping of XBIC instances to AXBS instances and XBIC ports to AXBS ports. See the "Chip-specific AXBS information" section in this document for the device component(s) mapped to each port of an AXBS instance. |
| 11-8<br>MST | initiator ID<br>initiator port that requested the most recent transfer with an attribute integrity check error detected.<br>The value in this field is the logical initiator ID number of the bus initiator. See the "Chip-specific XBIC information" section in this document for the initiator IDs and their corresponding components. |
| 7-0<br>SYN | Syndrome<br>Syndrome calculated for the most recent transfer with an attribute integrity check error detected.<br>For single-bit errors, identify the signal in error by matching the SYN value in Table 57, noting that the syndrome (SYN) values in the table are hexadecimal. |

## 17.6.5 XBIC Error Address (EAR)

### Offset

| Register | Offset |
|---|---|
| EAR | Ch |

### Function

In this register, XBIC reports the address of the most recent transfer with an attribute integrity check error detected—either because of a hardware fault or error injection. XBIC sets this register to all 0s only on reset.

**NOTE**

An attempted write to this read-only register results in a transfer error.

### Diagram

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

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

**Fields**

| Field | Function |
|---|---|
| 31-0<br>ADDR | Error Address<br>Address of the most recent transfer with an attribute integrity check error detected. |

## 17.7 Glossary

**EDC** Error Detection Code

**hdecor[31:0]** Non-standard AHB address phase signal for transporting optional decorated storage instruction information

**hdecorated** Non-standard AHB address phase signal for transporting optional decorated storage instruction information

# Chapter 18
# Extended Resource Domain Controller (XRDC)

## 18.1 Chip-specific XRDC information

### 18.1.1 MDAC configuration

All MDACs with DID = 0 use the default DID parameter.

Table 58. MDAC configuration

| Submodule instance | Configuration | XRDC MDACFG# | Bus master | Master ID value | Default DID | PID | Nonsecure input | Applicability |
|---|---|---|---|---|---|---|---|---|
| XRDC_MDAC0 | Processor | 1 | Cortex-M7_0, AXI, AHBP<br>Cortex-M7_0 debug | 0h | 0h | PID0 | 0b | MCXE315/ MCXE316/ MCXE317/ MCXE31B |
| XRDC_MDAC1 | Nonprocessor | 1 | eDMA AHB | 2h | 0h | — | 1b | MCXE315/ MCXE316/ MCXE317/ MCXE31B |
| XRDC_MDAC2 | Nonprocessor | 1 | Test port AHB | Ah | 0h | — | 1b | MCXE315/ MCXE316/ MCXE317/ MCXE31B |
| XRDC_MDAC3 | Processor | 1 | ELE_HSEB AHB<br>ELE_HSEB debug | 3h | 0h | PID3 | 0b | MCXE315/ MCXE316/ MCXE317/ MCXE31B |
| XRDC_MDAC5 | Nonprocessor | 1 | EMAC AHB | 4h | 0h | — | 1b | MCXE31B |

### 18.1.2 MRC configuration

Table 59. MRC configuration

| Submodule instance | Region format | Number of region descriptors | Slaves protected (port number) | Applicability |
|---|---|---|---|---|
| XRDC_MRC0 | Auto | 16[1] or 8 | PFLASH_0 (0)<br>PFLASH_1 (1)<br>PFLASH_2 (2)[1]<br>PFLASH_WR (3) | MCXE315/MCXE316/ MCXE317/MCXE31B |
| XRDC_MRC1 | Auto | 16[1] or 8 | PRAM0_0 (0)<br>PRAM1_0[1] (2) | MCXE315/MCXE316/ MCXE317/MCXE31B |

*Table continues on the next page...*

Table 59. MRC configuration (continued)

| Submodule instance | Region format | Number of region descriptors | Slaves protected (port number) | Applicability |
|---|---|---|---|---|
| | | | TCM backdoor (1) | |
| XRDC_MRC2 | Auto | 4 | QuadSPI (0) | MCXE31B |

1. Applicable to MCXE31B

### 18.1.3 PAC configuration

Table 60. PAC configuration

| Module name | Slaves protected | Applicability |
|---|---|---|
| XRDC_PAC0 | AIPS_0 | MCXE315/MCXE316/MCXE317/MCXE31B |
| XRDC_PAC1 | AIPS_1 | MCXE315/MCXE316/MCXE317/MCXE31B |
| XRDC_PAC2 | AIPS_2 | MCXE31B |

**NOTE**

For PDAC registers assignment to peripherals, see the memory map file attached to this document.

### 18.1.4 Domain Error Word registers (DERR_W*x*_0-18) mapping

The mapping of the domain error capture registers is as follows:

Table 61. Domain Error Word registers mapping

| Register | Corresponding MRC/PAC | Available in chips |
|---|---|---|
| DERR_W*x*_0 | MRC0 | MCXE315/MCXE316/MCXE317/MCXE31B |
| DERR_W*x*_1 | MRC1 | MCXE315/MCXE316/MCXE317/MCXE31B |
| DERR_W*x*_2 | MRC2 | MCXE31B |
| DERR_W*x*_16 | PAC0 | MCXE315/MCXE316/MCXE317/MCXE31B |
| DERR_W*x*_17 | PAC1 | MCXE315/MCXE316/MCXE317/MCXE31B |
| DERR_W*x*_18 | PAC2 | MCXE31B |

Where *x*: 0, 1, 2, 3.

If the above registers are accessed in chips wherein they are not present, a bus error gets reported.

### 18.1.5 Exceptions and violations

A write attempt by a noncore bus master outside the defined ranges leads to an exception in case the XRDC region is defined to prevent noncore master access. The chip generates a bus error if you violate XRDC policies.

### 18.1.6 Configuration using SBAF

SBAF must protect access to its own resources. Hence, XRDC is configured during initialization. SBAF provides the mechanism for you to configure XRDC during boot. You must program the configuration data in the application flash memory region.

#### 18.1.6.1 Default configuration of MDAC when ELE_HSEB firmware feature is disabled

Table 62. MDAC registers default value

| MDAC register | Value | Domain number assigned |
|---|---|---|
| ELE_HSEB CPU (MDAC3) | C000_0002h | 0 |

All the other MDAC domain numbers are 0 and SBAF does not lock them.

#### 18.1.6.2 Default configuration of MRC registers when ELE_HSEB firmware usage feature flag is disabled

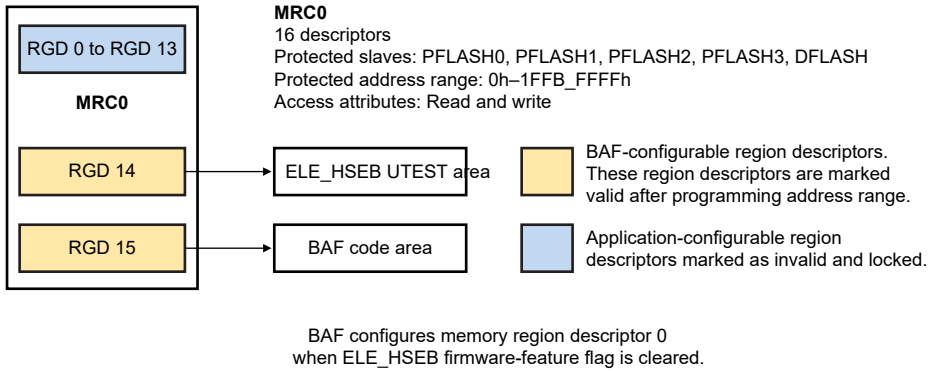



Figure 38. Default configuration of MRC registers when ELE_HSEB firmware usage feature flag is disabled

Table 63. Default configuration of MRC registers when ELE_HSEB firmware usage feature flag is disabled

| MRC number | Region descriptor number | Region address range (in hex) | Name | Remarks | SBAF's configuration |
|---|---|---|---|---|---|
| 0 | 14 | 1B00_0200 - 1B00_06FF | ELE_HSEB UTEST area | UTEST area that is accessible by ELE_HSEB core only. Start address is 1B00_0200h and 1B00_06FFh for others LC | W0 = start address = 1B00_0200h<br>W1 = end address = 1B00_06FFh<br>W2 = {SE = 0, SNUM = 0, D3ACP = 7h [all permission]} |
| 1 | 15 | 007F_4000 - 007F_FFFF | SBAF code area | Code area for SBAF and ELE_HSEB firmware | W0 = start address= 007F_4000h<br>W1 = end address= 007F_FFFFh<br>W2 = {SE = 0, SNUM = 0, D2ACP =7h [all permission], D1ACP = 0h [no permission], D0ACP = 0h [no permission]} |

#### 18.1.6.3 PDAC default configuration

SBAF configures and lock the following peripherals for its exclusive use. Read and write access to these peripheral are not provided to application domains.

Table 64. PDAC default configuration

| Peripheral | Peripheral PDAC number | Remarks |
|---|---|---|
| Flash memory controller alternate | 155 | ELE_HSEB uses the alternate interface exclusively. |
| Flash memory alternate | 188 | ELE_HSEB uses the alternate interface exclusively. |
| HSE_GPR | 231 | All cores have all permissions. |

### 18.1.7 Configuration when ELE_HSEB firmware-feature flag is cleared

When the ELE_HSEB firmware-feature flag is cleared, SBAF:

- Does not permit XRDC configuration.
- Locks the aforementioned configurations.

## 18.2 Overview

XRDC manages access control between initiators (cores and noncore initiators) and targets (memories and peripherals) by placing them in virtual groups called domains.

Conceptually, a domain is one or more initiators and memories and peripherals, that are isolated from others. It may help to look at a domain as a permissions group within a computing environment. All initiators in a domain have the same access to chip resources such as memory and peripherals. See Introduction to domains for more information on domains.

The protection provided by XRDC access control is in addition to the local memory protection unit contained within each core.

**NOTE**

Terminology in this chapter has been updated as follows:

Table 65. Updated terms

| Updated term | Deprecated term |
|---|---|
| Initiator | Master |
| Target | Slave |

### 18.2.1 Features

- Enables you to partition chip resources (initiator and target) into access-controlled domain.
  - Each domain has a unique DID .
  - The DID is an attribute of every system bus transaction.
- Provides a four-level hierarchical access control scheme for defining an ACP for each target in a domain. See Access control model for more information.
  - Memory region descriptors define access policies for address ranges within memories.
  - Peripheral access control registers define access policies for individual peripherals.
- Supports optional hardware semaphores to dynamically modify access rights for target resources.

### 18.2.2 Block diagram

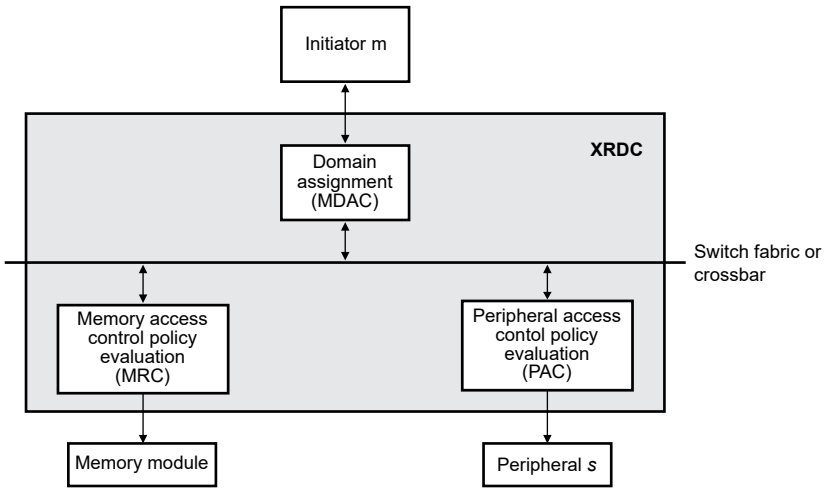



Figure 39. Block diagram

### 18.2.3 Block descriptions

| Block | Description |
|---|---|
| Domain assignment | A process that adds information to transactions, including:<br>• DID<br>• Privileged attribute<br>• Secure attribute<br>Domain assignment is performed by the MDAC submodule.<br>See:<br>• Domain assignment<br>• Master domain assignment controller (MDAC) |
| Initiator | A core or noncore (for example, DMA) module that can initiate transactions with memory or peripheral resources. |
| Memory | Non-volatile memory, RAM, or other memory. |
| Memory access control policy evaluation | A process that determines whether the domain associated with a transaction has access rights to a memory location. The process is performed by the MRC submodule.<br>See:<br>• Memory region ACP evaluation<br>• Memory region controller (MRC) |
| Peripheral | A nonmemory resource module within the chip—an ADC, timer module, or communications module, for example. |
| Peripheral access control policy evaluation | A process that determines whether the domain associated with a transaction has access rights to a peripheral. The process is performed by the PAC submodule. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Block | Description |
|---|---|
| | See Peripheral access controller (PAC). |
| Crossbar | The chip's module and I/O interconnect infrastructure. |

### 18.2.4 Indexes used in this chapter

**Table 66. Indexes used in this chapter**

| Index | Description |
|---|---|
| *c* | Memory controller number. For example, MRC*c*. |
| *d* | Domain number. For example, PDAC_W0_6[D*d*ACP]. |
| *m* | Initiator number. For example, PID*m*. |
| *r* | Memory region number. For example, RGD_W0_*r*. |
| *s* | PDAC slot number. For example, PDAC_W0_*s*[D0ACP]. |
| *w* | Word number. In a group of registers consisting of consecutive 32-bit registers, *w* is the 0-indexed register number. For example, MRGD_W*w*_0. |

### 18.2.5 Exceptions and violations

A write attempt by a noncore bus initiator outside the defined ranges leads to an exception in case the XRDC region is defined to prevent noncore initiator access. The chip generates a bus error if you violate XRDC policies.

## 18.3 Modes of operation

XRDC does not support any special modes of operation.

## 18.4 External signal description

XRDC does not have any external signals.

## 18.5 Functional description

### 18.5.1 Introduction to domains

A domain typically consists of one or more initiators, along with the memories and peripherals those initiators are allowed to access. Because it is access-controlled, a domain acts as an independent computing environment.

Domains enable applications to coexist on the same silicon with a firewall between them, enforcing absolute interference protection.

Generally, you create each domain to meet a specific need. Examples of XRDC domain usage include:

- Isolation of real-time from non-real-time applications to ensure resource availability
- Isolation of safety-critical code from non-safety-critical code
- Isolation of third-party untrusted applications from trusted software
- Isolation of memory regions to ensure data security or to prevent accidental overwrites
- Limiting write access for a specific area of system memory to a single DMA module instance

- Limiting read access for a specific area of system memory to a specific processor core

You can assign a core to multiple domains but only one of those domains can be active at a given time.

Within each XRDC instance, each domain has a unique identifier, known as its DID. If an XRDC instance has 16 DIDs, that means the instance has 16 available domains.

You control which initiators can access a peripheral by configuring the domain ACP for each peripheral. Similarly, you control which initiators can access a memory region by configuring the ACP for each memory region.

## 18.5.2 Submodules

XRDC implements its functionality through its hardware submodules:

- Master domain assignment controller (MDAC)
- Memory region controller (MRC)
- Peripheral access controller (PAC)

### 18.5.2.1 Master domain assignment controller (MDAC)

The MDAC submodule performs domain assignment logic. XRDC contains an MDAC submodule for each XRDC-protected initiator. Each MDAC assigns a DID to every transaction from its associated initiator. You configure the domain assignment activity for each MDAC through a set of registers, in one of two formats:

- DFMT0 core domain assignment registers
- DFMT1 noncore domain assignment registers

To understand the role of domains in XRDC protection, see Introduction to domains.

For a full explanation of the domain assignment process, see Domain assignment.

### 18.5.2.2 Memory region controller (MRC)

The MRC submodule performs memory region access control. Each XRDC instance contains the number of MRCs indicated in HWCFG0[NMRC]. Each MRC is associated with one or more memories (see the chip-specific XRDC information for details). The MRC controls memory access using memory region descriptors. MRCFG*d*[NMRGD] indicates the number of region descriptors available for MRC *c*.

Each memory region descriptor defines a memory address range and a configurable access control policy for each domain using a set of four or five 32-bit registers (see Memory ACP evaluation registers).

Memory region descriptors also support including an optional hardware semaphore in the ACP evaluation for memory regions shared by multiple domains (see Hardware semaphores and dynamic access rights).

For a full explanation of the memory region ACP evaluation process, see Memory region ACP evaluation.

### 18.5.2.3 Peripheral access controller (PAC)

The PAC submodule provides domain access control for all peripherals connected to a single peripheral bus. Each XRDC contains the number of PACs indicated in HWCFG0[NPAC]. Each PAC supports up to 128 peripheral slots (see Finding the PDAC slot number for a peripheral). You configure the ACP for each peripheral using a set of PDAC_W*w*_*s* registers (see Peripheral ACP evaluation registers).

Peripheral access control also enable a hardware semaphore to be included in the access control policy evaluation for peripherals shared by multiple domains. See Hardware semaphores and dynamic access rights for more details.

For a full explanation of the peripheral ACP evaluation process, see Peripheral ACP evaluation process.

### 18.5.3 Transaction protection

During application execution, high-level chip modules such as cores or DMA, Ethernet, known as initiators, initiate transaction requests to memory and peripheral resources. XRDC adds protection capabilities to ensure the requesting initiator accesses only the resources that it is authorized to access. These capabilities support security and safety requirements.

XRDC provides this protection by adding two steps to the unprotected transaction flow, as shown in XRDC transaction flow.

XRDC transaction processing differs depending on:

- The type of initiator making the request (see Transaction request sources).
- The type of target receiving the request (see Transaction targets).

**NOTE**

See chip-specific XRDC information for the domain ID of each initiator on chip.

### 18.5.4 XRDC transaction flow

**Table 67. XRDC transaction flow**

| Step | Operation | Performed by | Description |
|---|---|---|---|
| 1 | Transaction request | Initiator | An initiator requests a read or write transaction targeting memory or a peripheral. |
| 2 | Domain assignment | XRDC | XRDC MDAC submodule intercepts the request and performs domain assignment, which adds this metadata to the transaction:<br>• DID<br>• Privileged attribute<br>• Secure attribute |
| 3 | Interconnect | Chip | The chip transmits the domain-assigned transaction request across the interconnect (crossbar). |
| 4 | ACP evaluation | XRDC | XRDC MRC or PAC submodule intercepts the transaction request and evaluates it against the target's ACP to determine whether the requesting initiator has sufficient access rights to the target. If it does, the transaction continues. Otherwise, XRDC generates an access violation error and the transaction terminates. |
| 5 | Transaction response | Target resource | If the previous step does not generate an access violation error, the target resource processes the transaction request and transmits data (for read transactions) and transaction status information (for all transactions) back across the interconnect to the requesting initiator.<br>XRDC is not involved in the flow of information from the target resource back to the requesting initiator. |

### 18.5.5 Domain assignment

#### 18.5.5.1 Overview

Domain assignment associates a DID with each transaction request from an initiator. To determine the DID for a transaction, XRDC evaluates the domain-specific configuration data in the set of MDAC registers (MDA_W*n*_*m*_DFMT0 or MDA_W*n*_*m*_DFMT1) associated with the requesting initiator. The exact process depends on the source of the request (see Transaction request sources).

If the value of MDACFG*m*[NMDAR] is 1, which means a single W*w* register for a given initiator, the specified domain identifier is used directly. In case this value is more than 1, which means there are multiple W*w* registers for a given initiator, MDAC evaluates the conditional terms to determine a "hit". For all W*w* hits, their corresponding domain identifiers are logically summed together (boolean OR). Use cases are typically expected to hit in a single W*w*. Special care is needed if none of the conditional terms hit in any Wn evaluation; in this case, the generated DID = 0 and you must be aware of any potential access rights granted for this DID.

The number of MDAC registers can vary for each initiator. See the chip-specific information. MDACFG*m*[NMDAR] indicates the number of MDA registers, where *m* is the initiator number. You need *m* to locate the registers relevant for domain assignment. See the chip-specific XRDC information for more on initiator numbers. See:

- Register settings for DFMT0 direct domain assignment transactions
- Register settings for DFMT0 PID-based transactions
- Register settings for DFMT1 direct domain assignment transactions

Domain assignment also assigns the secure and privileged attributes to the transaction.

#### 18.5.5.2 PID-based domain assignment

To provide more flexibility in routing core tasks to chip resources in different access-controlled domains, XRDC supports the use of a PID. If the core initiator contains a built-in PID register, indicated by HWCFG2[PIDP*m*] = 1 , XRDC reflects the core PID value in PID*m*[PID], and bit 5 of that field indicates the secure attribute for the transaction request. Otherwise, an application can mimic PID-based domain assignment by writing a value to that field.

#### 18.5.5.3 Transaction request sources

The domain assignment process for an XRDC-protected transaction request depends on the source of the request.

**Table 68. Transaction request sources**

| Request source | Topic | Brief description |
|---|---|---|
| Core initiator | DFMT0 direct domain assignment example | Direct domain assignment using a DFMT0 master domain assignment register. |
| Core initiator | DFMT0 PID-based domain assignment example | PID-based domain assignment using a DFMT0 master domain assignment register. |
| Core initiator | DFMT1 direct domain assignment example-single MDA | Direct domain assignment using a DFMT1 master domain assignment register with single MDA. |

#### 18.5.5.4 DFMT0 core domain assignment registers

**Table 69. DFMT0 core domain assignment registers**

| Register | Index |
|---|---|
| MDACFG*m* | *m* = initiator number. See the chip-specific XRDC information for the available list of initiators with their IDs. |
| MDA_W*w*_*m*_DFMT0[1] | *m* = initiator number.<br>*w* = word (see MDA register structure). MDACFG*m*[NMDAR] indicates the number of MDA registers (words) per initiator. |
| PID*m* | *m* = initiator number. |

1. See MDA register structure.

#### 18.5.5.5 DFMT1 noncore domain assignment registers

**Table 70. DFMT1 noncore domain assignment registers**

| Register | Index |
|---|---|
| MDACFG*m* | *m* = initiator number. See the chip-specific XRDC information for the available list of initiators and their IDs. |
| MDA_W*w*_*m*_DFMT1[1] | *m* = initiator number.<br>*w* = word. MDACFG*m*[NMDAR] indicates the number of MDA registers (words) per initiator. |

1. See MDA register structure.

#### 18.5.5.6 MDA register structure

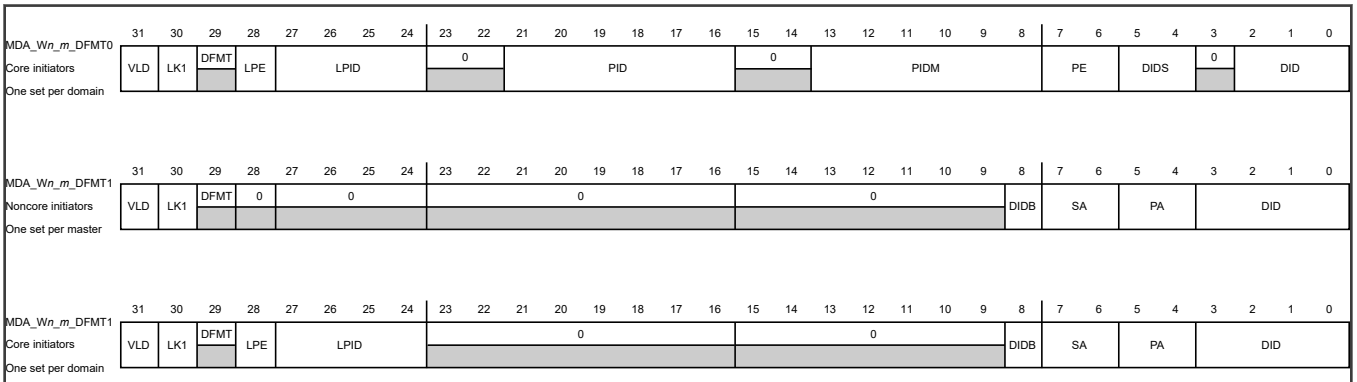



**Figure 40. MDA register structure**

**NOTE**

This XRDC configuration does not have LPE and LPID fields and should be considered as reserved fields.

### 18.5.6 ACP evaluation

#### 18.5.6.1 Overview

When XRDC is not enabled, all peripherals and memories allow unrestricted access. XRDC allows you to limit that access to requests from a particular domain or domains with an application-specified ACP. XRDC intercepts the transaction request and evaluates it against the target's ACP to determine whether the requesting initiator has sufficient access rights to the target, based on the:

- Domain ID assignment (DID)
- Privileged attribute
- Secured attribute

XRDC obtains the target resource's domain ACP from the associated Domain Access Control Policy (D*d*ACP) field (see Domain ACP specification) in the appropriate register set:

- Peripheral ACP evaluation registers
- Memory ACP evaluation registers

If ACP evaluation determines that the transaction request has sufficient access rights to the target resource, XRDC allows the transaction to continue. Otherwise, it terminates the request with an error and captures the address and attribute information in the appropriate error registers.

The exact process depends on the target of the request (see Transaction targets).

XRDC optionally supports the inclusion of a hardware semaphore to dynamically alter the ACP of a memory region or peripheral (see Hardware semaphores and dynamic access rights).

### 18.5.6.2 Transaction targets

The ACP evaluation for an XRDC-protected transaction request depends on the target of the request.

**Table 71. Transaction targets**

| Request target | Topic | Brief description |
|---|---|---|
| Peripheral | Peripheral ACP evaluation example | Process for a transaction request to a target peripheral that is within a protected domain, with ACP evaluation configured by PDAC_W*w*_*s*. |
| Memory | Memory ACP evaluation example | Process for a transaction request to a target memory region that is within a protected domain, with ACP evaluation configured by MRGD_W*w*_*r*. |

### 18.5.6.3 Peripheral ACP evaluation registers

**Table 72. Peripheral ACP evaluation registers**

| Register | Index | Brief description |
|---|---|---|
| PDAC_W0_*s*[1] | *s* = peripheral slot | Specifies the ACP for an XRDC-protected peripheral, and an optional hardware semaphore. |
| PDAC_W1_*s*[1] | *s* = peripheral slot | Enables the set of PDAC registers for the associated peripheral and locks the set. Typically, you configure the peripheral by writing to the PDAC registers and then limiting their respective domains from making updates to the D*d*ACP fields or by locking the set for all updates. |

1. See PDAC register structure.

### 18.5.6.4 PDAC register structure

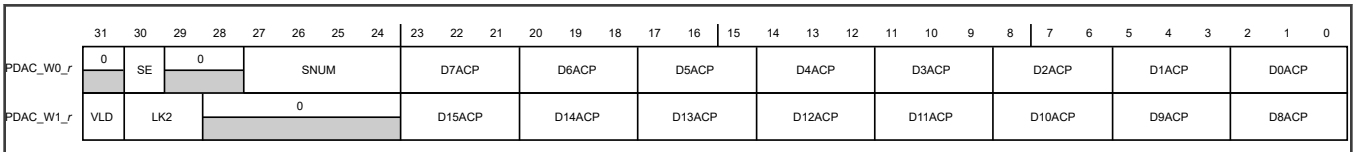



**Figure 41. PDAC register structure**

### 18.5.6.5 Memory ACP evaluation registers

When XRDC is enabled (CR[GVLD] = 1), you cannot access any XRDC-protected memory unless you configure at least one set of memory region descriptors (see MRGD_W*w*_*r*) that includes the targeted memory.

Table 73. Memory ACP evaluation registers

| Register | Index | Brief description |
|---|---|---|
| MRCFG*c* | *c* = memory controller instance | Indicates the number of memory regions per memory controller (NMRGD). Each memory region is configured by a set of memory region descriptor registers (MRGD_W*w*_*r*) described below. |
| MRGD_W0_*r*[1] | *r* = memory region | Specifies starting address for the memory region. |
| MRGD_W1_*r*[1] | *r* = memory region | Specifies ending address for the memory region. |
| MRGD_W2_*r*[1] | *r* = memory region | Specifies the ACP for each supported domain, and an optional hardware semaphore. |
| MRGD_W3_*r*[1] | *r* = memory region | Enables the set of MRGD registers for the associated region and locks the set. Typically, you define the memory region by writing to the MRGD registers and then limiting their respective domains from updating the D*d*ACP fields or by locking the set for all updates. |

1. See MRGD register structure.

### 18.5.6.6 MRGD register structure

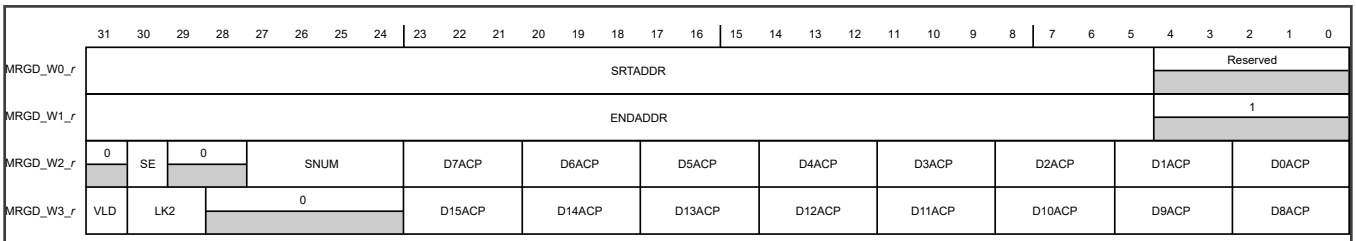



Figure 42. MRGD register structure

### 18.5.6.7 Access control model

XRDC supports a four-level hierarchical access control model. This model combines the traditional privileged (also known as supervisor) and user access levels with an additional signal for the secure attribute of each memory reference, as shown in Access control model levels.

Each level has different access control policies that specify the read and write accessibility for a target. XRDC combines the privileged and secure attributes with the DID assigned to each system bus transaction to form the hardware basis for the access control mechanism. You specify the ACP for target resources using the D*d*ACP fields (see Domain ACP specification) found in the configuration registers shown in Peripheral ACP evaluation registers and Memory ACP evaluation registers.

You can dynamically control access to shared memory regions and target peripherals with the optional inclusion of a hardware semaphore (see Hardware semaphores and dynamic access rights). If you enable this semaphore for a given address space or peripheral, XRDC allows writes to the target resource only if the requesting domain owns the semaphore.

For cores that support only the three-state access control model (Secure Privileged, Secure User, Nonsecure User), XRDC forces the nonsecure output signal from the MDAC submodule to 0 in privileged mode. This change enables precise state transitions between user and privileged modes. Specifically, the MDAC logic for initiator *m* generates the nonsecure attribute output signal as a function of the Three-State Model (PID*m*[TSM]) and PID Present (HWCFG2[PIDP*m*]) fields, as shown in Generation of secure attribute).

#### 18.5.6.7.1 Access control model levels

Table 74. Access control model levels

| Secure | Nonsecure | Privileged (supervisor) | Not privileged (user) | Level |
|---|---|---|---|---|
| Yes | Not applicable | Yes | Not applicable | Most restricted |
| Yes | Not applicable | Not applicable | Yes | More restricted |
| Not applicable | Yes | Yes | Not applicable | Less restricted |
| Not applicable | Yes | Not applicable | Yes | Least restricted |

#### 18.5.6.7.2 Generation of secure attribute

This table shows how XRDC generates the secure attribute for a transaction.

Table 75. Generation of secure attribute

| Access levels [1] | PID*m*[TSM] (b) | PID present [2] | Secure attribute determined by |
|---|---|---|---|
| 4 | 0 | No | Initiator secure attribute |
| 4 | 0 | Yes | Initiator secure attribute && ~initiator privileged attribute |
| 3 | 1 | No | PID*m*[5] && ~initiator privileged attribute |
| 3 | 1 | Yes | Local secure attribute && ~initiator privileged attribute |

1. XRDC assumes a core initiator supports the four-level access model. If a core supports only the three-state access control model, you must write 1 to PID*m*[TSM] before loading any nonsecure value into the PID.
2. Indicated in HWCFG2–3[PIDP*m*].

### 18.5.6.8 Domain ACP specification

Table 76. Domain ACP specification

| D*d*ACP | Allowable accesses | | | |
|---|---|---|---|---|
| | Secure Privileged | Secure User | Nonsecure Privileged | Nonsecure User |
| 111b | R, W | R, W | R, W | R, W |
| 110b | R, W | R, W | R, W | None |
| 101b | R, W | R, W | R | R |
| 100b | R, W | R, W | R | None |
| 011b | R, W | R, W | None | None |
| 010b | R, W | None | None | None |
| 001b | R | R | None | None |
| 000b | None | None | None | None |

#### 18.5.6.9 Memory region ACP evaluation

During ACP evaluation of a memory transaction request, if the target memory location falls within the address range specified by any XRDC memory region descriptor, then XRDC identifies the request as a memory region hit. For each memory region hit, XRDC compares the DID, privileged attribute, and secured attribute of the transaction to the associated domain ACP (MRGD_W{2,3}_*r*[D*d*ACP]) in the memory region descriptor. See ACP evaluation for additional details on this function.

The following conditions cause XRDC to report an access error:

- The target memory location does not fall within any defined memory regions; in other words, the transaction request is not a hit.
- The transaction request does not have the appropriate access permissions for the region, which triggers a domain violation.
- The transaction request hits multiple (overlapping) regions, and all of those regions signal access violations.

Unimplemented domain identifiers default to no access privileges, and therefore the access type of a D*d*ACP field for an unimplemented domain is read-as-zero/writes-ignored (RAZ/WI).

#### 18.5.6.10 Hardware semaphores and dynamic access rights

XRDC memory region descriptors and peripheral access control support an optional hardware semaphore in their access evaluations. This hardware semaphore allows hardware enforcement of dynamic access rights, based on the state of the semaphore for shared memory regions or shared peripherals.

If enabled, the state of the semaphore dynamically modifies the access control policies so that only the domain owning it has write access to the resource. The write permissions for all other domains are revoked based on the semaphore state. If no domain owns the semaphore, the PAC and MRC submodules evaluate DdACP normally.

If you enable a hardware semaphore (by writing 1 to MRGD_W2_*r*[SE] or PDAC_W0_*r*[SE]) then, before the normal DdACP evaluation, XRDC checks the state of the hardware semaphore specified in MRGD_W2_*r*[SNUM] or PDAC_W0_*r*[SNUM].

On a write transaction, if the semaphore is not idle (the semaphore state is non-zero) and the requesting domain does not own the semaphore, the memory or peripheral access terminates with an error. In other words, writes into a semaphore-enabled address space or peripheral are allowed only if the semaphore is idle or the requesting domain owns the semaphore.

### 18.5.7 XRDC transaction examples

To see the complete transaction process for an XRDC-protected transaction request from an initiator to target:

1. Follow one of these examples of domain assignment:
   - DFMT0 direct domain assignment example
   - DFMT0 PID-based domain assignment example
   - DFMT1 direct domain assignment example-single MDA
2. Then follow one of these examples of ACP evaluation:
   - Peripheral ACP evaluation example
   - Memory ACP evaluation example

#### 18.5.7.1 DFMT0 direct domain assignment example

This configuration assigns a specific domain to all incoming transactions.

- A core initiator has initiator number = 6.
- The core has one MDAC register (MDACFG6[NMDAR] = 1), configured as shown in Register settings for DFMT0 direct domain assignment transactions.

MDAC_W*w*_*m*_DFMT0 registers do not include fields for secure and privileged attributes. Those attributes are part of the transaction data from core initiators and are forwarded with the transaction after domain assignment.

#### 18.5.7.1.1 Register settings for DFMT0 direct domain assignment transactions

**Table 77. Register settings for DFMT0 direct domain assignment transactions**

| Field | MDA_W0_6_DFMT0 | Comments |
|---|---|---|
| VLD | 1 | Enables this register for use in domain assignment. |
| LK1 | 1 | Locks the settings in this register until the next module reset. |
| DFMT | 0 | Indicates that this is a DFMT0 register. |
| PID | 00_0000b | Not used because PE = 00b. |
| PIDM | 00_0000b | Not used because PE = 00b. |
| PE | 00b | Disables PID-based filtering. |
| DIDS | 00b | All incoming transactions are to be assigned to the domain specified by the DID field. |
| DID | 01b | Because PE = 00b and DIDS = 00b, all incoming transactions are to be assigned DID value 01b. |

#### 18.5.7.1.2 DFMT0 direct domain assignment process

The application configures XRDC as it boots. After application boot completes, a core issues a read request to a target peripheral.

1. XRDC intercepts the request and performs domain assignment using the configuration in MDA_W0_6_DFMT0. In this example, XRDC assigns DID = 01b to all incoming transactions.
2. The transaction proceeds with DID = 01b and the privileged and secure attributes provided by the core.

### 18.5.7.2 DFMT0 PID-based domain assignment example

This example configuration demonstrates PID-based domain assignment:

- A core initiator with initiator number = 4 processes both safety-critical tasks and routine tasks, using two domains:
  - Domain 0 is reserved for safety-critical tasks (PID = 0–15).
  - Domain 1 is reserved for routine tasks (PID > 15).
- The core has eight MDAC registers (MDACFG4[NMDAR] = 8), configured as follows (see Register settings for DFMT0 PID-based transactions):
  - MDA_W0_4_DFMT0 assigns safety-critical tasks (PID = 0–15) to domain 0.
  - MDA_W1_4_DFMT0 assigns routine tasks (PID = 16–31) to domain 1.
  - MDA_W2_4_DFMT0 assigns routine tasks (PID = 32–63) to domain 1.
  - MDA_W3_4_DFMT0 through MDA_W7_4_DFMT0 are not used.

MDAC_W*w*_*m*_DFMT0 registers do not include fields for secure and privileged attributes. Those attributes are part of the transaction data from core initiators and are forwarded with the transaction after domain assignment.

#### 18.5.7.2.1 Register settings for DFMT0 PID-based transactions

**Table 78. Register settings for PID-based transactions**

| Field | Registers | | | | |
|---|---|---|---|---|---|
| | MDA_W0_4_DFMT0 | MDA_W1_4_DFMT0 | MDA_W2_4_DFMT0 | MDA_W[3–7]_4_DFMT0 | Comments |
| VLD | 1 | 1 | 1 | 0 | Enables this register for use when assigning domains. |
| LK1 | 1 | 1 | 1 | 1 | Locks the settings in each register until the next device reset. |
| DFMT | 0 | 0 | 0 | 0 | Indicates that these registers apply to domain assignment for core initiators. |
| PID | 00_0000b | 01_0000b | 10_0000b | — | Constant match value to be used for PID-based filtering. |
| PIDM | 00_1111b | 10_1111b | 01_1111b | — | Each 0 bit causes the corresponding PID bit to be considered in domain assignment. |
| PE | 10b | 10b | 10b | — | Specifies the type of pattern matching used for PID evaluation. |
| DIDS | 00b | 00b | 00b | — | Assign all incoming transactions with PIDs that pass the filtering criteria to the domain specified by the DID field. |
| DID | 00b | 01b | 01b | — | This DID value is assigned to incoming transactions with PIDs that pass the filtering criteria. |

#### 18.5.7.2.2 DFMT0 PID-based domain assignment process

The application configures XRDC as it boots. After booting completes, a core issues a read request to a target peripheral. The task making the request has PID = 6, indicating that it is a safety-critical task.

1. XRDC intercepts the request and performs domain assignment using each enabled MDA_W*n*_4_DFMT0 register, regardless of whether it has already found a match. In this example, XRDC performs the domain assignments as shown in DFMT0 PID-based domain assignment evaluation.
2. After XRDC completes all domain assignment evaluations, it logically ORs the assigned DIDs to determine the final DID assigned to the transaction. In this example, only one evaluation results in a DID assignment, so there is no OR operation.
3. The transaction proceeds with DID 00b and the privilege and secure attributes provided by the core.

#### 18.5.7.2.3 DFMT0 PID-based domain assignment evaluation

**Table 79. DFMT0 PID-based domain assignment evaluation**

| Register | Evaluation steps | Boolean math | Result |
|---|---|---|---|
| MDA_W0_4_DFMT0 | 1. Bitwise AND of PID with inverted PIDM. | 00_0000b & 11_0000b | 00_0000b |
| | 2. Bitwise AND of transaction PID (PID4[PID]) with inverted PIDM. | 000110b & 11_0000b | 00_0000b |
| | 3. Compare the results of steps 1 and 2. | 00_0000b == 000000b | True: Assign DID 0 |

*Table continues on the next page...*

Table 79. DFMT0 PID-based domain assignment evaluation (continued)

| Register | Evaluation steps | Boolean math | Result |
|---|---|---|---|
| MDA_W1_4_DFMT0 | 1. Bitwise AND of PID with inverted PIDM. | 01_0000b & 01_0000b | 01_0000b |
| | 2. Bitwise AND of transaction PID (PID4[PID]) with inverted PIDM. | 00_0110b & 01_0000b | 00_0000b |
| | 3. Compare the results of steps 1 and 2. | 01_0000b == 00_0000b | False: No DID assignment |
| MDA_W2_4_DFMT0 | 1. Bitwise AND of PID with inverted PIDM. | 10_0000b & 10_0000b | 10_0000b |
| | 2. Bitwise AND of transaction PID (PID4[PID]) with inverted PIDM. | 00_0110b & 10_0000b | 00_0000b |
| | 3. Compare the results of steps 1 and 2. | 10_0000b == 00_0000b | False: No DID assignment |

#### 18.5.7.3 DFMT1 direct domain assignment example-single MDA

This configuration assigns a specific domain to all incoming transactions.

- An initiator has initiator number = 6.
- The initiator has one MDAC register (MDACFG6[NMDAR] = 1) as shown in Register settings for DFMT1 direct domain assignment transactions.

##### 18.5.7.3.1 Register settings for DFMT1 direct domain assignment transactions

Table 80. Register settings for DFMT1 direct domain assignment transactions

| Field | MDA_W0_6_DFMT1 | Comments |
|---|---|---|
| VLD | 1 | Enables this register for use in domain assignment. |
| LK1 | 1 | Locks the settings in this register until the next chip reset. |
| DFMT | 1 | Indicates that this is a DFMT1 register. |
| DIDB | 0b | All incoming transactions are to be assigned to the domain that the DID field specifies. |
| SA | 10b | All incoming transactions retain their Secure attribute value. |
| PA | 10b | All incoming transactions retain their Privileged attribute value. |
| DID | 01b | Because DIDB = 0b, all incoming transactions are to be assigned DID value 01b. |

##### 18.5.7.3.2 DFMT1 direct domain assignment process

The application configures XRDC as it boots. After application boot completes, an initiator issues a read request to a target peripheral.

1. XRDC intercepts the request and assigns a domain using the configuration in MDA_W0_6_DFMT1. In this example, XRDC assigns DID = 01b to all incoming transactions.
2. The transaction proceeds with DID = 01b and the privileged and secure attributes provided by the initiator.

#### 18.5.7.4 Peripheral ACP evaluation example

This example configuration demonstrates ACP evaluation for a target peripheral.

- The core initiator must have exclusive access to the ADC0 peripheral for safety-critical tasks in domain 0.
- The core initiator supports 8 domains (HWCFG0[NDID] = 111b).
- ADC occupies PDAC slot 40 in the chip, as defined in an example memory map file (see Finding the PDAC slot number for a peripheral).
- Given 8 domains and PDAC slot 40, the PDAC registers associated with ADC0 are PDAC_W0_40 and PDAC_W1_40.

The following sections describe the register configurations for this example.

#### 18.5.7.4.1 Finding the PDAC slot number for a peripheral

This topic shows how to find the PDAC slot number for a peripheral, but it is a generic example. Memory map file organization and appearance can vary.

To find the PDAC slot number for a peripheral:

1. Open the memory map file attached to this document and view the peripherals page.
2. Locate the peripheral in the "Instance" column.
3. The PDAC slot number for the peripheral is at the intersection of the "PDAC slot number" column and the peripheral row.

In this figure, for peripheral ADC_0, the PDAC slot number is 40. Therefore, the PDAC registers for ADC0 are PDAC_W*w*_40.

| | A | B | D | E | I | J |
|---|---|---|---|---|---|---|
| 1 | Instance | Description | Start address | End address | PDAC slot number | AIPS instance |
| 8 | LCU_0 | Logic Control Unit 0 | 0x40098000 | 0x4009BFFF | 38 | AIPS_0 |
| 9 | LCU_1 | Logic Control Unit 1 | 0x4009C000 | 0x4009FFFF | 39 | AIPS_0 |
| 10 | ADC_0 | Analog-to-digital converter 0 | 0x400A0000 | 0x400A3FFF | 40 | AIPS_0 |
| 11 | ADC_1 | Analog-to-digital converter 1 | 0x400A4000 | 0x400A7FFF | 41 | AIPS_0 |
| 12 | ADC_2 | Analog-to-digital converter 2 | 0x400A8000 | 0x400ABFFF | 42 | AIPS_0 |

**Figure 43. Finding the PDAC slot number for a peripheral**

#### 18.5.7.4.2 Register settings for peripheral ACP evaluation

This table shows the PDAC_W*w*_40 settings for a safety-critical task assigned to domain 0.

**Table 81. Register settings for peripheral ACP evaluation**

| Register | Field | Value (b) | Comments |
|---|---|---|---|
| PDAC_W0_40 | SE | 0 | The hardware semaphore (see the SEMA42 chapter) is disabled. |
| | SNUM | (don't care) | The hardware semaphore is not used in this example. |
| | D7ACP | 000 | Domain 7 has no access to the peripheral. |
| | D6ACP | 000 | Domain 6 has no access to the peripheral. |
| | D5ACP | 000 | Domain 5 has no access to the peripheral. |
| | D4ACP | 000 | Domain 4 has no access to the peripheral. |
| | D3ACP | 000 | Domain 3 has no access to the peripheral. |
| | D2ACP | 000 | Domain 2 has no access to the peripheral. |

*Table continues on the next page...*

Table 81. Register settings for peripheral ACP evaluation (continued)

| Register | Field | Value (b) | Comments |
|---|---|---|---|
| | D1ACP | 000 | Domain 1 has no access to the peripheral. |
| | D0ACP | 010 | Only privileged, secure transactions from domain 0 have access. |
| PDAC_W1_40 | VLD | 1 | Use this register set in domain ACP evaluations. |
| | LK2 | 11 | Lock the settings in this register until the next device reset. |

#### 18.5.7.4.3 Peripheral ACP evaluation process

XRDC performs the following process for ACP evaluation:

1. When the application is running, the core issues a read request to a target peripheral. The task making the request has PID = 0, indicating that it is a safety-critical task. The transaction request is privileged and secured.
2. Between the chip interconnect and ADC0, XRDC intercepts the request and compares its DID, privileged attribute, and secured attribute to the configuration in PDAC_W0_40 and PDAC_W1_40.
3. Because the ADC0 D0ACP field is 010b for privileged, secured access, XRDC grants access to the transaction.
4. The transaction proceeds normally without any further intervention from XRDC.

### 18.5.7.5 Memory ACP evaluation example

This example configuration demonstrates ACP evaluation for a target memory. Following are the desired features:

- The core initiator must have exclusive access to the memory region for safety-critical tasks in domain 0.
- The target memory is the address range 1B00_0000h–1B00_1FFFh, protected by memory controller 0 (MRC0), for example.
- Access to the entire memory range will be controlled by the memory region descriptor defined by MRGD_W*w*_0.
- The requested transaction is secure privileged.

With the configuration settings shown in Register settings for memory ACP evaluation, XRDC grants access and the transaction proceeds normally. There is no further XRDC intervention.

#### 18.5.7.5.1 Register settings for memory ACP evaluation

Table 82. Register settings for memory ACP evaluation

| Register | Field | Value | Comments |
|---|---|---|---|
| MRGD_W0_0 | SRTADDR | 1B00_0000h | Starting address of the memory region. |
| MRGD_W1_0 | ENDADDR | 1B00_1FFFh | Ending address of the memory region. |
| MRGD_W2_0 | SE | 0 | The hardware semaphore (see the SEMA42 chapter) is disabled. |
| | SNUM | (don't care) | The hardware semaphore is not used in this example. |
| | D7ACP | 000b | Domain 7 has no access to the peripheral. |
| | D6ACP | 000b | Domain 6 has no access to the peripheral. |
| | D5ACP | 000b | Domain 5 has no access to the peripheral. |
| | D4ACP | 000b | Domain 4 has no access to the peripheral. |
| | D3ACP | 000b | Domain 3 has no access to the peripheral. |

*Table continues on the next page...*

Table 82. Register settings for memory ACP evaluation (continued)

| Register | Field | Value | Comments |
|---|---|---|---|
| | D2ACP | 000b | Domain 2 has no access to the peripheral. |
| | D1ACP | 000b | Domain 1 has no access to the peripheral. |
| | D0ACP | 010b | Only privileged, secure transactions from domain 0 have access. |
| MRGD_W3_0 | VLD | 1 | Use this register set in domain ACP evaluations. |
| | LK2 | 11b | Lock the settings in this register until the next device reset. |

#### 18.5.7.5.2 Memory ACP evaluation process

In this example, XRDC performs the following process for ACP evaluation:

1. When the application is running, the core issues a read request to address 1B00_0100h. The transaction request is secure privileged.
2. Between the chip interconnect and the memory, XRDC intercepts the request and compares its DID, memory location of the address, privileged attribute, and secured attribute to the configuration in MRGD_W0_0, MRGD_W1_0, MRGD_W2_0 and MRGD_W3_0.
3. Because the address 1B00_0100h is in the memory range 1B00_0000h–1B00_1FFFh and its D0ACP field is 010b for privileged, secured access, XRDC grants access to the transaction.
4. The transaction proceeds normally without any no further intervention from XRDC.

### 18.5.8 Clocking

This module has no clocking considerations.

### 18.5.9 Interrupts

This module outputs an interrupt signal which can be connected to interrupt controller. Check chip-specific interrupt assignment for details. Interrupt is asserted on detection of access violation by any checker, and it remains asserted until DERRLOC registers are cleared.

## 18.6 Initialization information

Out of reset, XRDC is disabled (CR[GVLD] = 0), which allows secure privileged startup code to configure the entire programming model.

### 18.6.1 Initialization procedure

1. Read the hardware configuration registers to obtain the implemented XRDC hardware capabilities for the chip:
   - HWCFG0
   - HWCFG1
   - HWCFG2
   - MDACFG*m* (one for each supported bus initiator—number indicated in HWCFG0[NMSTR])
   - MRCFG*c* (one for each supported memory controller—number indicated in HWCFG0[NMRC])
2. Use the information retrieved in step 1 and the desired domain architecture to configure:
   - Domain assignments (MDA_W*w_m*_DFMT0 and MDA_W*w_m*_DFMT1)
   - Memory region descriptors (MRGD_W*w_n*)

- Peripheral access control (PDAC_W*w*_*s*)

Ensure that you enable the necessary registers and register sets using the appropriate VLD fields. Also, you can limit access to these registers or lock them after you configure them, by using the appropriate LK1 fields.

3. Enable XRDC (write 1 to CR[GVLD]).

XRDC is now fully operational.

### 18.6.2 Minimize access errors

When you configure and enable XRDC, it begins generating DIDs for transaction requests and evaluating access rights at the target memory and peripheral resources. Because of the distributed design hierarchy and the pipelined nature of the hardware system bus fabric, it can take multiple cycles for a generated DID to propagate. Until that happens, XRDC uses the initiator's default DID. Depending on the programmed ACPs, the default DID might generate an access error response.

If XRDC generates incorrect error responses, you can use the following approaches to minimize or eliminate these extraneous access errors:

- Minimize the amount of system bus traffic when XRDC is enabled (CR[GVLD] = 1).
- Ensure that all target memory addresses provide sufficient access rights for any default DIDs and for the newly programmed DIDs. After XRDC is fully operational, as confirmed by a read of HWCFG1, you can remove the permissions for the default DIDs.
- Try to have the bus initiator that programs and configures XRDC use the same DID, that is, its default DID, for both initialization and configuration, besides normal system operation. You do this when you define the DID assignments for the system.

## 18.7 Application information

### 18.7.1 Master domain assignments

The typical use case for master domain assignments is to include one or more core bus initiators in a single domain, possibly combined with other noncore bus initiator modules such as DMA. This configuration may be static or may be changed dynamically to select between a small number of domains. HWCFG0[NDID] indicates the maximum number of supported domains. XRDC also supports the optional use of PIDs to create multiple classes of cores, each in different domains.

For example, you can group critical tasks—safety-critical, performance-critical, and so on—into one domain and all other tasks into a second domain. Typically, you assign the DID at initialization, but you can also reconfigure domain assignment while the application is running.

A core bus initiator typically has multiple MDA_W*w*_*m*_DFMT0 registers associated with it.

A noncore bus initiator typically has a single MDA_W*w*_*m*_DFMT1 register associated with it.

The master domain assignment, memory region descriptor, and peripheral domain access control registers have lock fields that enable you to limit access to, or to lock, the registers. These actions protect the configuration.

### 18.7.2 Memory region descriptor management

There are two important concepts to consider for managing the memory region descriptors.

Each MRC*c* configuration is chip-specific. See the chip-specific XRDC information for the number of implemented memory region descriptors (MRGD_W*w*_*n*) in a given MRC*c* instance, and the specific port numbers associated with the target memories being monitored.

Second, as detailed in Memory region ACP evaluation, after you enable the XRDC, a memory reference must hit one or more of the configured regions. Otherwise, the transaction results in an access violation. Two other conditions also result in access errors:

- The access hits a single region descriptor and that region signals a domain violation.
- The access hits multiple (overlapping) regions and all regions signal violations.

The second condition reflects that XRDC gives priority to permission granting over access denying for overlapping regions. This approach provides more flexibility to system software in memory region descriptor assignments.

## 18.7.3 Domain error capture management

### 18.7.3.1 Domain error capture registers

When an MRC or PAC detects a domain access violation, XRDC captures information about the transaction in the following registers:

**Table 83. Domain error capture registers**

| Register[field] | Index | Information |
|---|---|---|
| DERRLOC*d*[MRCINST] | *d* = DID | Domain error location for MRC instances, with asserted bits indicating which MRC instance numbers are reporting an error |
| DERRLOC*d*[PACINST] | *d* = DID | Domain error location for PAC instances, with asserted bits indicating which PAC instance numbers are reporting an error |
| DERR_W0_*i* | *i* = instance number | Transaction target address |
| DERR_W1_*i* | *i* = instance number | Additional information about the transaction |
| DERR_W3_*i* | *i* = instance number | Reset and rearm domain error capture for the instance |

### 18.7.3.2 Handling domain access violation errors

When an MRC or PAC instance detects a domain access violation, it reports the error by asserting the associated bit in DERRLOCd[MRCINST] or [PACINST], and XRDC asserts the error interrupt output. To retrieve information about the error, the error handler must:

1. Read each DERRLOC*d* register until it finds a non-zero MRCINST or PACINST value.

   The index of the DERRLOC*d* register is the DID for the domain in which the error occurred.

2. Configure the domain assignment for the master executing the exception handler (MDA_Ww_m_DFMTf[DID]) to assign the DID that corresponds to the DERRLOC*d* register index, such that the DERR_Ww_i registers show the error for that domain.

3. Read HWCFG1[DID] to be sure the error handler is now operating in the correct domain. In other words, make sure HWCFG1[DID] equals the value written to MDA_Ww_m_DFMTf[DID].

4. Find the number of an MRC or PAC instance reporting an error by parsing DERRLOC*d*[MRCINST] and DERRLOC*d*[PACINST] for an asserted bit.

   There may be multiple access violations, across multiple MRC or PAC instances, pending for a given domain. To quickly find the lowest numbered instance reporting an access violation, execute a "find first one bit" instruction (alternatively known as "count leading zeroes") on the MRCINST and PACINST fields.

5. Retrieve the error address (DERR_W0_*i*[EADDR]).

6. Retrieve the error information (DERR_W1_*i*).

   More than one error may have occurred in the MRC or PAC instance, as indicated by the error status (DERR_W1_*i*[EST] = 11b). If more than one error has occurred in the instance, XRDC captures data only for the first error.

7. Use the error address and information to handle the error (whatever that may require).

8. Reset the DERR_W*w*_*i* registers and rearm error capture (write 1b to DERR_W3_*i*[RECR]).

   Rearming error capture deasserts the instance bit in DERRLOC*d*.

9. Repeat steps 1 and 8 for each asserted bit in PACINST and MRCINST until there are no more asserted bits.

Domain error retrieval illustrates an example error retrieval.

#### 18.7.3.3 Domain error retrieval

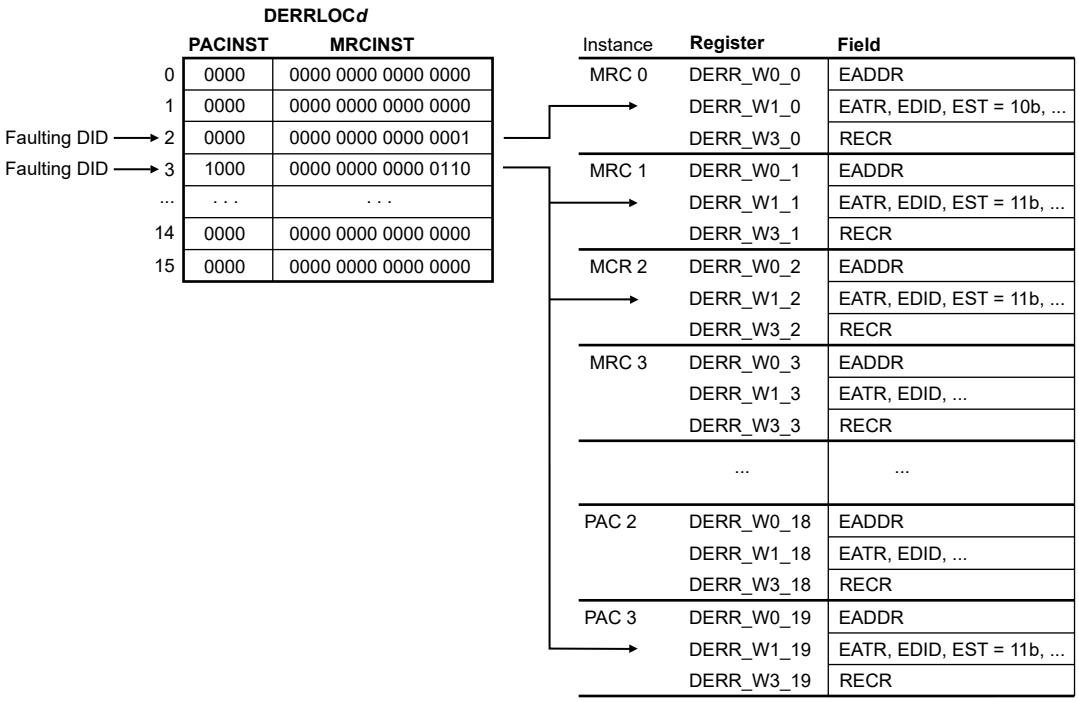



Figure 44. Domain error retrieval

## 18.8 Memory map and register definitions

### 18.8.1 Register organization

XRDC registers are partitioned into these groups:

- Basic hardware control and configuration
- Domain errors (including location and details)
- Master domain assignments
- Peripheral domain access controls
- Memory region descriptors

### 18.8.2 Register access guidelines

The following guidelines apply to XRDC register access:

- You can access the XRDC registers only in secure, privileged access mode.
- Unless stated otherwise, the registers support 8-, 16-, and 32-bit reads, and 32-bit writes.
- Unless stated otherwise, XRDC terminates the following access attempts with an error:
  - Accesses in a different access mode
  - Unsupported write data size
  - Writes to read-only resources

- Writes to reserved address spaces

- Accesses to these memory map holes return an error:
  - Any access to a register that does not exist
  - Holes in the 0–F0h and DERR to PID register space
  - For MDAC, gaps in the master domain assignment (MDA_W*w*_*m*_DFMT*n*) registers
  - For MRCs, any gap in the MRGD_W*w*_*r* registers (for example, if there are four memory region descriptors, attempted access to a fifth descriptor fails)
- Accesses to these memory map holes do not return a bus error:
  - MRCs: Memory region descriptors occupy only four words but have an additional four words of address available: words 4–7
  - PDAC: Registers associated with unimplemented PDAC slots
- Read accesses to these memory map holes do not return a bus error:
  - Offset F8h and FCh
  - Offset 100–13Fh
  - Offset 140–14Fh
  - Offset 200–23Ch

## 18.8.3 XRDC register descriptions

### 18.8.3.1 XRDC memory map

XRDC base address: 4027_8000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Control (CR) | 32 | RW | 0000_008Ah |
| F0h | Hardware Configuration 0 (HWCFG0) | 32 | R | 1202_0502h |
| F4h | Hardware Configuration 1 (HWCFG1) | 32 | R | See section |
| F8h | Hardware Configuration 2 (HWCFG2) | 32 | R | 0000_0000h |
| 100h | Master Domain Assignment Configuration (MDACFG0) | 8 | R | 01h |
| 101h | Master Domain Assignment Configuration (MDACFG1) | 8 | R | 81h |
| 102h | Master Domain Assignment Configuration (MDACFG2) | 8 | R | 81h |
| 103h | Master Domain Assignment Configuration (MDACFG3) | 8 | R | 01h |
| 105h | Master Domain Assignment Configuration (MDACFG5) | 8 | R | 81h |
| 140h | Memory Region Configuration (MRCFG0) | 8 | R | 10h |
| 141h | Memory Region Configuration (MRCFG1) | 8 | R | 10h |
| 142h | Memory Region Configuration (MRCFG2) | 8 | R | 04h |
| 200h - 208h | Domain Error Location (DERRLOC0 - DERRLOC2) | 32 | R | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 400h | Domain Error Word 0 (DERR_W0_0) | 32 | R | 0000_0000h |
| 404h | Domain Error Word 1 (DERR_W1_0) | 32 | R | 0000_0000h |
| 40Ch | Domain Error Word 3 (DERR_W3_0) | 32 | RW | 0000_0000h |
| 410h | Domain Error Word 0 (DERR_W0_1) | 32 | R | 0000_0000h |
| 414h | Domain Error Word 1 (DERR_W1_1) | 32 | R | 0000_0000h |
| 41Ch | Domain Error Word 3 (DERR_W3_1) | 32 | RW | 0000_0000h |
| 420h | Domain Error Word 0 (DERR_W0_2) | 32 | R | 0000_0000h |
| 424h | Domain Error Word 1 (DERR_W1_2) | 32 | R | 0000_0000h |
| 42Ch | Domain Error Word 3 (DERR_W3_2) | 32 | RW | 0000_0000h |
| 500h | Domain Error Word 0 (DERR_W0_16) | 32 | R | 0000_0000h |
| 504h | Domain Error Word 1 (DERR_W1_16) | 32 | R | 0000_0000h |
| 50Ch | Domain Error Word 3 (DERR_W3_16) | 32 | RW | 0000_0000h |
| 510h | Domain Error Word 0 (DERR_W0_17) | 32 | R | 0000_0000h |
| 514h | Domain Error Word 1 (DERR_W1_17) | 32 | R | 0000_0000h |
| 51Ch | Domain Error Word 3 (DERR_W3_17) | 32 | RW | 0000_0000h |
| 520h | Domain Error Word 0 (DERR_W0_18) | 32 | R | 0000_0000h |
| 524h | Domain Error Word 1 (DERR_W1_18) | 32 | R | 0000_0000h |
| 52Ch | Domain Error Word 3 (DERR_W3_18) | 32 | RW | 0000_0000h |
| 700h | Process Identifier (PID0) | 32 | RW | 0000_0000h |
| 70Ch | Process Identifier (PID3) | 32 | RW | 0000_0000h |
| 800h | Master Domain Assignment (MDA_W0_0_DFMT0) | 32 | RW | 0000_0000h |
| 820h | Master Domain Assignment (MDA_W0_1_DFMT1) | 32 | RW | 2000_0000h |
| 840h | Master Domain Assignment (MDA_W0_2_DFMT1) | 32 | RW | 2000_0000h |
| 860h | Master Domain Assignment (MDA_W0_3_DFMT0) | 32 | RW | 0000_0000h |
| 8A0h | Master Domain Assignment (MDA_W0_5_DFMT1) | 32 | RW | 2000_0000h |
| 1100h | Peripheral Domain Access Control Word 0 (PDAC_W0_32) | 32 | RW | 0000_0000h |
| 1104h | Peripheral Domain Access Control Word 1 (PDAC_W1_32) | 32 | RW | 0000_0000h |
| 1108h | Peripheral Domain Access Control Word 0 (PDAC_W0_33) | 32 | RW | 0000_0000h |
| 110Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_33) | 32 | RW | 0000_0000h |
| 1110h | Peripheral Domain Access Control Word 0 (PDAC_W0_34) | 32 | RW | 0000_0000h |
| 1114h | Peripheral Domain Access Control Word 1 (PDAC_W1_34) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1118h | Peripheral Domain Access Control Word 0 (PDAC_W0_35) | 32 | RW | 0000_0000h |
| 111Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_35) | 32 | RW | 0000_0000h |
| 1120h | Peripheral Domain Access Control Word 0 (PDAC_W0_36) | 32 | RW | 0000_0000h |
| 1124h | Peripheral Domain Access Control Word 1 (PDAC_W1_36) | 32 | RW | 0000_0000h |
| 1130h | Peripheral Domain Access Control Word 0 (PDAC_W0_38) | 32 | RW | 0000_0000h |
| 1134h | Peripheral Domain Access Control Word 1 (PDAC_W1_38) | 32 | RW | 0000_0000h |
| 1138h | Peripheral Domain Access Control Word 0 (PDAC_W0_39) | 32 | RW | 0000_0000h |
| 113Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_39) | 32 | RW | 0000_0000h |
| 1140h | Peripheral Domain Access Control Word 0 (PDAC_W0_40) | 32 | RW | 0000_0000h |
| 1144h | Peripheral Domain Access Control Word 1 (PDAC_W1_40) | 32 | RW | 0000_0000h |
| 1148h | Peripheral Domain Access Control Word 0 (PDAC_W0_41) | 32 | RW | 0000_0000h |
| 114Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_41) | 32 | RW | 0000_0000h |
| 1150h | Peripheral Domain Access Control Word 0 (PDAC_W0_42) | 32 | RW | 0000_0000h |
| 1154h | Peripheral Domain Access Control Word 1 (PDAC_W1_42) | 32 | RW | 0000_0000h |
| 1160h | Peripheral Domain Access Control Word 0 (PDAC_W0_44) | 32 | RW | 0000_0000h |
| 1164h | Peripheral Domain Access Control Word 1 (PDAC_W1_44) | 32 | RW | 0000_0000h |
| 1168h | Peripheral Domain Access Control Word 0 (PDAC_W0_45) | 32 | RW | 0000_0000h |
| 116Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_45) | 32 | RW | 0000_0000h |
| 1170h | Peripheral Domain Access Control Word 0 (PDAC_W0_46) | 32 | RW | 0000_0000h |
| 1174h | Peripheral Domain Access Control Word 1 (PDAC_W1_46) | 32 | RW | 0000_0000h |
| 1178h | Peripheral Domain Access Control Word 0 (PDAC_W0_47) | 32 | RW | 0000_0000h |
| 117Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_47) | 32 | RW | 0000_0000h |
| 1400h | Peripheral Domain Access Control Word 0 (PDAC_W0_128) | 32 | RW | 0000_0000h |
| 1404h | Peripheral Domain Access Control Word 1 (PDAC_W1_128) | 32 | RW | 0000_0000h |
| 1408h | Peripheral Domain Access Control Word 0 (PDAC_W0_129) | 32 | RW | 0000_0000h |
| 140Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_129) | 32 | RW | 0000_0000h |
| 1410h | Peripheral Domain Access Control Word 0 (PDAC_W0_130) | 32 | RW | 0000_0000h |
| 1414h | Peripheral Domain Access Control Word 1 (PDAC_W1_130) | 32 | RW | 0000_0000h |
| 1418h | Peripheral Domain Access Control Word 0 (PDAC_W0_131) | 32 | RW | 0000_0000h |
| 141Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_131) | 32 | RW | 0000_0000h |
| 1420h | Peripheral Domain Access Control Word 0 (PDAC_W0_132) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1424h | Peripheral Domain Access Control Word 1 (PDAC_W1_132) | 32 | RW | 0000_0000h |
| 1428h | Peripheral Domain Access Control Word 0 (PDAC_W0_133) | 32 | RW | 0000_0000h |
| 142Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_133) | 32 | RW | 0000_0000h |
| 1430h | Peripheral Domain Access Control Word 0 (PDAC_W0_134) | 32 | RW | 0000_0000h |
| 1434h | Peripheral Domain Access Control Word 1 (PDAC_W1_134) | 32 | RW | 0000_0000h |
| 1438h | Peripheral Domain Access Control Word 0 (PDAC_W0_135) | 32 | RW | 0000_0000h |
| 143Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_135) | 32 | RW | 0000_0000h |
| 1440h | Peripheral Domain Access Control Word 0 (PDAC_W0_136) | 32 | RW | 0000_0000h |
| 1444h | Peripheral Domain Access Control Word 1 (PDAC_W1_136) | 32 | RW | 0000_0000h |
| 1448h | Peripheral Domain Access Control Word 0 (PDAC_W0_137) | 32 | RW | 0000_0000h |
| 144Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_137) | 32 | RW | 0000_0000h |
| 1450h | Peripheral Domain Access Control Word 0 (PDAC_W0_138) | 32 | RW | 0000_0000h |
| 1454h | Peripheral Domain Access Control Word 1 (PDAC_W1_138) | 32 | RW | 0000_0000h |
| 1458h | Peripheral Domain Access Control Word 0 (PDAC_W0_139) | 32 | RW | 0000_0000h |
| 145Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_139) | 32 | RW | 0000_0000h |
| 1460h | Peripheral Domain Access Control Word 0 (PDAC_W0_140) | 32 | RW | 0000_0000h |
| 1464h | Peripheral Domain Access Control Word 1 (PDAC_W1_140) | 32 | RW | 0000_0000h |
| 1468h | Peripheral Domain Access Control Word 0 (PDAC_W0_141) | 32 | RW | 0000_0000h |
| 146Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_141) | 32 | RW | 0000_0000h |
| 1470h | Peripheral Domain Access Control Word 0 (PDAC_W0_142) | 32 | RW | 0000_0000h |
| 1474h | Peripheral Domain Access Control Word 1 (PDAC_W1_142) | 32 | RW | 0000_0000h |
| 1478h | Peripheral Domain Access Control Word 0 (PDAC_W0_143) | 32 | RW | 0000_0000h |
| 147Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_143) | 32 | RW | 0000_0000h |
| 1480h | Peripheral Domain Access Control Word 0 (PDAC_W0_144) | 32 | RW | 0000_0000h |
| 1484h | Peripheral Domain Access Control Word 1 (PDAC_W1_144) | 32 | RW | 0000_0000h |
| 1488h | Peripheral Domain Access Control Word 0 (PDAC_W0_145) | 32 | RW | 0000_0000h |
| 148Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_145) | 32 | RW | 0000_0000h |
| 1490h | Peripheral Domain Access Control Word 0 (PDAC_W0_146) | 32 | RW | 0000_0000h |
| 1494h | Peripheral Domain Access Control Word 1 (PDAC_W1_146) | 32 | RW | 0000_0000h |
| 1498h | Peripheral Domain Access Control Word 0 (PDAC_W0_147) | 32 | RW | 0000_0000h |
| 149Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_147) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 14A0h | Peripheral Domain Access Control Word 0 (PDAC_W0_148) | 32 | RW | 0000_0000h |
| 14A4h | Peripheral Domain Access Control Word 1 (PDAC_W1_148) | 32 | RW | 0000_0000h |
| 14A8h | Peripheral Domain Access Control Word 0 (PDAC_W0_149) | 32 | RW | 0000_0000h |
| 14ACh | Peripheral Domain Access Control Word 1 (PDAC_W1_149) | 32 | RW | 0000_0000h |
| 14B0h | Peripheral Domain Access Control Word 0 (PDAC_W0_150) | 32 | RW | 0000_0000h |
| 14B4h | Peripheral Domain Access Control Word 1 (PDAC_W1_150) | 32 | RW | 0000_0000h |
| 14B8h | Peripheral Domain Access Control Word 0 (PDAC_W0_151) | 32 | RW | 0000_0000h |
| 14BCh | Peripheral Domain Access Control Word 1 (PDAC_W1_151) | 32 | RW | 0000_0000h |
| 14C0h | Peripheral Domain Access Control Word 0 (PDAC_W0_152) | 32 | RW | 0000_0000h |
| 14C4h | Peripheral Domain Access Control Word 1 (PDAC_W1_152) | 32 | RW | 0000_0000h |
| 14C8h | Peripheral Domain Access Control Word 0 (PDAC_W0_153) | 32 | RW | 0000_0000h |
| 14CCh | Peripheral Domain Access Control Word 1 (PDAC_W1_153) | 32 | RW | 0000_0000h |
| 14D0h | Peripheral Domain Access Control Word 0 (PDAC_W0_154) | 32 | RW | 0000_0000h |
| 14D4h | Peripheral Domain Access Control Word 1 (PDAC_W1_154) | 32 | RW | 0000_0000h |
| 14D8h | Peripheral Domain Access Control Word 0 (PDAC_W0_155) | 32 | RW | 0000_0000h |
| 14DCh | Peripheral Domain Access Control Word 1 (PDAC_W1_155) | 32 | RW | 0000_0000h |
| 14E0h | Peripheral Domain Access Control Word 0 (PDAC_W0_156) | 32 | RW | 0000_0000h |
| 14E4h | Peripheral Domain Access Control Word 1 (PDAC_W1_156) | 32 | RW | 0000_0000h |
| 14E8h | Peripheral Domain Access Control Word 0 (PDAC_W0_157) | 32 | RW | 0000_0000h |
| 14ECh | Peripheral Domain Access Control Word 1 (PDAC_W1_157) | 32 | RW | 0000_0000h |
| 14F0h | Peripheral Domain Access Control Word 0 (PDAC_W0_158) | 32 | RW | 0000_0000h |
| 14F4h | Peripheral Domain Access Control Word 1 (PDAC_W1_158) | 32 | RW | 0000_0000h |
| 14F8h | Peripheral Domain Access Control Word 0 (PDAC_W0_159) | 32 | RW | 0000_0000h |
| 14FCh | Peripheral Domain Access Control Word 1 (PDAC_W1_159) | 32 | RW | 0000_0000h |
| 1500h | Peripheral Domain Access Control Word 0 (PDAC_W0_160) | 32 | RW | 0000_0000h |
| 1504h | Peripheral Domain Access Control Word 1 (PDAC_W1_160) | 32 | RW | 0000_0000h |
| 1508h | Peripheral Domain Access Control Word 0 (PDAC_W0_161) | 32 | RW | 0000_0000h |
| 150Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_161) | 32 | RW | 0000_0000h |
| 1510h | Peripheral Domain Access Control Word 0 (PDAC_W0_162) | 32 | RW | 0000_0000h |
| 1514h | Peripheral Domain Access Control Word 1 (PDAC_W1_162) | 32 | RW | 0000_0000h |
| 1518h | Peripheral Domain Access Control Word 0 (PDAC_W0_163) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 151Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_163) | 32 | RW | 0000_0000h |
| 1520h | Peripheral Domain Access Control Word 0 (PDAC_W0_164) | 32 | RW | 0000_0000h |
| 1524h | Peripheral Domain Access Control Word 1 (PDAC_W1_164) | 32 | RW | 0000_0000h |
| 1528h | Peripheral Domain Access Control Word 0 (PDAC_W0_165) | 32 | RW | 0000_0000h |
| 152Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_165) | 32 | RW | 0000_0000h |
| 1530h | Peripheral Domain Access Control Word 0 (PDAC_W0_166) | 32 | RW | 0000_0000h |
| 1534h | Peripheral Domain Access Control Word 1 (PDAC_W1_166) | 32 | RW | 0000_0000h |
| 1538h | Peripheral Domain Access Control Word 0 (PDAC_W0_167) | 32 | RW | 0000_0000h |
| 153Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_167) | 32 | RW | 0000_0000h |
| 1540h | Peripheral Domain Access Control Word 0 (PDAC_W0_168) | 32 | RW | 0000_0000h |
| 1544h | Peripheral Domain Access Control Word 1 (PDAC_W1_168) | 32 | RW | 0000_0000h |
| 1548h | Peripheral Domain Access Control Word 0 (PDAC_W0_169) | 32 | RW | 0000_0000h |
| 154Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_169) | 32 | RW | 0000_0000h |
| 1550h | Peripheral Domain Access Control Word 0 (PDAC_W0_170) | 32 | RW | 0000_0000h |
| 1554h | Peripheral Domain Access Control Word 1 (PDAC_W1_170) | 32 | RW | 0000_0000h |
| 1558h | Peripheral Domain Access Control Word 0 (PDAC_W0_171) | 32 | RW | 0000_0000h |
| 155Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_171) | 32 | RW | 0000_0000h |
| 1568h | Peripheral Domain Access Control Word 0 (PDAC_W0_173) | 32 | RW | 0000_0000h |
| 156Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_173) | 32 | RW | 0000_0000h |
| 1578h | Peripheral Domain Access Control Word 0 (PDAC_W0_175) | 32 | RW | 0000_0000h |
| 157Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_175) | 32 | RW | 0000_0000h |
| 1588h | Peripheral Domain Access Control Word 0 (PDAC_W0_177) | 32 | RW | 0000_0000h |
| 158Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_177) | 32 | RW | 0000_0000h |
| 1590h | Peripheral Domain Access Control Word 0 (PDAC_W0_178) | 32 | RW | 0000_0000h |
| 1594h | Peripheral Domain Access Control Word 1 (PDAC_W1_178) | 32 | RW | 0000_0000h |
| 1598h | Peripheral Domain Access Control Word 0 (PDAC_W0_179) | 32 | RW | 0000_0000h |
| 159Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_179) | 32 | RW | 0000_0000h |
| 15A0h | Peripheral Domain Access Control Word 0 (PDAC_W0_180) | 32 | RW | 0000_0000h |
| 15A4h | Peripheral Domain Access Control Word 1 (PDAC_W1_180) | 32 | RW | 0000_0000h |
| 15A8h | Peripheral Domain Access Control Word 0 (PDAC_W0_181) | 32 | RW | 0000_0000h |
| 15ACh | Peripheral Domain Access Control Word 1 (PDAC_W1_181) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 15B0h | Peripheral Domain Access Control Word 0 (PDAC_W0_182) | 32 | RW | 0000_0000h |
| 15B4h | Peripheral Domain Access Control Word 1 (PDAC_W1_182) | 32 | RW | 0000_0000h |
| 15B8h | Peripheral Domain Access Control Word 0 (PDAC_W0_183) | 32 | RW | 0000_0000h |
| 15BCh | Peripheral Domain Access Control Word 1 (PDAC_W1_183) | 32 | RW | 0000_0000h |
| 15C0h | Peripheral Domain Access Control Word 0 (PDAC_W0_184) | 32 | RW | 0000_0000h |
| 15C4h | Peripheral Domain Access Control Word 1 (PDAC_W1_184) | 32 | RW | 0000_0000h |
| 15D0h | Peripheral Domain Access Control Word 0 (PDAC_W0_186) | 32 | RW | 0000_0000h |
| 15D4h | Peripheral Domain Access Control Word 1 (PDAC_W1_186) | 32 | RW | 0000_0000h |
| 15D8h | Peripheral Domain Access Control Word 0 (PDAC_W0_187) | 32 | RW | 0000_0000h |
| 15DCh | Peripheral Domain Access Control Word 1 (PDAC_W1_187) | 32 | RW | 0000_0000h |
| 15E0h | Peripheral Domain Access Control Word 0 (PDAC_W0_188) | 32 | RW | 0000_0000h |
| 15E4h | Peripheral Domain Access Control Word 1 (PDAC_W1_188) | 32 | RW | 0000_0000h |
| 15F8h | Peripheral Domain Access Control Word 0 (PDAC_W0_191) | 32 | RW | 0000_0000h |
| 15FCh | Peripheral Domain Access Control Word 1 (PDAC_W1_191) | 32 | RW | 0000_0000h |
| 1608h | Peripheral Domain Access Control Word 0 (PDAC_W0_193) | 32 | RW | 0000_0000h |
| 160Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_193) | 32 | RW | 0000_0000h |
| 1610h | Peripheral Domain Access Control Word 0 (PDAC_W0_194) | 32 | RW | 0000_0000h |
| 1614h | Peripheral Domain Access Control Word 1 (PDAC_W1_194) | 32 | RW | 0000_0000h |
| 1618h | Peripheral Domain Access Control Word 0 (PDAC_W0_195) | 32 | RW | 0000_0000h |
| 161Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_195) | 32 | RW | 0000_0000h |
| 1620h | Peripheral Domain Access Control Word 0 (PDAC_W0_196) | 32 | RW | 0000_0000h |
| 1624h | Peripheral Domain Access Control Word 1 (PDAC_W1_196) | 32 | RW | 0000_0000h |
| 1628h | Peripheral Domain Access Control Word 0 (PDAC_W0_197) | 32 | RW | 0000_0000h |
| 162Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_197) | 32 | RW | 0000_0000h |
| 1630h | Peripheral Domain Access Control Word 0 (PDAC_W0_198) | 32 | RW | 0000_0000h |
| 1634h | Peripheral Domain Access Control Word 1 (PDAC_W1_198) | 32 | RW | 0000_0000h |
| 1648h | Peripheral Domain Access Control Word 0 (PDAC_W0_201) | 32 | RW | 0000_0000h |
| 164Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_201) | 32 | RW | 0000_0000h |
| 1650h | Peripheral Domain Access Control Word 0 (PDAC_W0_202) | 32 | RW | 0000_0000h |
| 1654h | Peripheral Domain Access Control Word 1 (PDAC_W1_202) | 32 | RW | 0000_0000h |
| 1658h | Peripheral Domain Access Control Word 0 (PDAC_W0_203) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 165Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_203) | 32 | RW | 0000_0000h |
| 1660h | Peripheral Domain Access Control Word 0 (PDAC_W0_204) | 32 | RW | 0000_0000h |
| 1664h | Peripheral Domain Access Control Word 1 (PDAC_W1_204) | 32 | RW | 0000_0000h |
| 1668h | Peripheral Domain Access Control Word 0 (PDAC_W0_205) | 32 | RW | 0000_0000h |
| 166Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_205) | 32 | RW | 0000_0000h |
| 1670h | Peripheral Domain Access Control Word 0 (PDAC_W0_206) | 32 | RW | 0000_0000h |
| 1674h | Peripheral Domain Access Control Word 1 (PDAC_W1_206) | 32 | RW | 0000_0000h |
| 1678h | Peripheral Domain Access Control Word 0 (PDAC_W0_207) | 32 | RW | 0000_0000h |
| 167Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_207) | 32 | RW | 0000_0000h |
| 1680h | Peripheral Domain Access Control Word 0 (PDAC_W0_208) | 32 | RW | 0000_0000h |
| 1684h | Peripheral Domain Access Control Word 1 (PDAC_W1_208) | 32 | RW | 0000_0000h |
| 1688h | Peripheral Domain Access Control Word 0 (PDAC_W0_209) | 32 | RW | 0000_0000h |
| 168Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_209) | 32 | RW | 0000_0000h |
| 16A0h | Peripheral Domain Access Control Word 0 (PDAC_W0_212) | 32 | RW | 0000_0000h |
| 16A4h | Peripheral Domain Access Control Word 1 (PDAC_W1_212) | 32 | RW | 0000_0000h |
| 16A8h | Peripheral Domain Access Control Word 0 (PDAC_W0_213) | 32 | RW | 0000_0000h |
| 16ACh | Peripheral Domain Access Control Word 1 (PDAC_W1_213) | 32 | RW | 0000_0000h |
| 16B0h | Peripheral Domain Access Control Word 0 (PDAC_W0_214) | 32 | RW | 0000_0000h |
| 16B4h | Peripheral Domain Access Control Word 1 (PDAC_W1_214) | 32 | RW | 0000_0000h |
| 16B8h | Peripheral Domain Access Control Word 0 (PDAC_W0_215) | 32 | RW | 0000_0000h |
| 16BCh | Peripheral Domain Access Control Word 1 (PDAC_W1_215) | 32 | RW | 0000_0000h |
| 16C0h | Peripheral Domain Access Control Word 0 (PDAC_W0_216) | 32 | RW | 0000_0000h |
| 16C4h | Peripheral Domain Access Control Word 1 (PDAC_W1_216) | 32 | RW | 0000_0000h |
| 16C8h | Peripheral Domain Access Control Word 0 (PDAC_W0_217) | 32 | RW | 0000_0000h |
| 16CCh | Peripheral Domain Access Control Word 1 (PDAC_W1_217) | 32 | RW | 0000_0000h |
| 16D8h | Peripheral Domain Access Control Word 0 (PDAC_W0_219) | 32 | RW | 0000_0000h |
| 16DCh | Peripheral Domain Access Control Word 1 (PDAC_W1_219) | 32 | RW | 0000_0000h |
| 16E0h | Peripheral Domain Access Control Word 0 (PDAC_W0_220) | 32 | RW | 0000_0000h |
| 16E4h | Peripheral Domain Access Control Word 1 (PDAC_W1_220) | 32 | RW | 0000_0000h |
| 16E8h | Peripheral Domain Access Control Word 0 (PDAC_W0_221) | 32 | RW | 0000_0000h |
| 16ECh | Peripheral Domain Access Control Word 1 (PDAC_W1_221) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 16F8h | Peripheral Domain Access Control Word 0 (PDAC_W0_223) | 32 | RW | 0000_0000h |
| 16FCh | Peripheral Domain Access Control Word 1 (PDAC_W1_223) | 32 | RW | 0000_0000h |
| 1700h | Peripheral Domain Access Control Word 0 (PDAC_W0_224) | 32 | RW | 0000_0000h |
| 1704h | Peripheral Domain Access Control Word 1 (PDAC_W1_224) | 32 | RW | 0000_0000h |
| 1708h | Peripheral Domain Access Control Word 0 (PDAC_W0_225) | 32 | RW | 0000_0000h |
| 170Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_225) | 32 | RW | 0000_0000h |
| 1710h | Peripheral Domain Access Control Word 0 (PDAC_W0_226) | 32 | RW | 0000_0000h |
| 1714h | Peripheral Domain Access Control Word 1 (PDAC_W1_226) | 32 | RW | 0000_0000h |
| 1718h | Peripheral Domain Access Control Word 0 (PDAC_W0_227) | 32 | RW | 0000_0000h |
| 171Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_227) | 32 | RW | 0000_0000h |
| 1728h | Peripheral Domain Access Control Word 0 (PDAC_W0_229) | 32 | RW | 0000_0000h |
| 172Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_229) | 32 | RW | 0000_0000h |
| 1738h | Peripheral Domain Access Control Word 0 (PDAC_W0_231) | 32 | RW | 0000_0000h |
| 173Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_231) | 32 | RW | 0000_0000h |
| 1740h | Peripheral Domain Access Control Word 0 (PDAC_W0_232) | 32 | RW | 0000_0000h |
| 1744h | Peripheral Domain Access Control Word 1 (PDAC_W1_232) | 32 | RW | 0000_0000h |
| 1760h | Peripheral Domain Access Control Word 0 (PDAC_W0_236) | 32 | RW | 0000_0000h |
| 1764h | Peripheral Domain Access Control Word 1 (PDAC_W1_236) | 32 | RW | 0000_0000h |
| 1800h | Peripheral Domain Access Control Word 0 (PDAC_W0_256) | 32 | RW | 0000_0000h |
| 1804h | Peripheral Domain Access Control Word 1 (PDAC_W1_256) | 32 | RW | 0000_0000h |
| 1808h | Peripheral Domain Access Control Word 0 (PDAC_W0_257) | 32 | RW | 0000_0000h |
| 180Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_257) | 32 | RW | 0000_0000h |
| 1820h | Peripheral Domain Access Control Word 0 (PDAC_W0_260) | 32 | RW | 0000_0000h |
| 1824h | Peripheral Domain Access Control Word 1 (PDAC_W1_260) | 32 | RW | 0000_0000h |
| 1828h | Peripheral Domain Access Control Word 0 (PDAC_W0_261) | 32 | RW | 0000_0000h |
| 182Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_261) | 32 | RW | 0000_0000h |
| 1830h | Peripheral Domain Access Control Word 0 (PDAC_W0_262) | 32 | RW | 0000_0000h |
| 1834h | Peripheral Domain Access Control Word 1 (PDAC_W1_262) | 32 | RW | 0000_0000h |
| 1838h | Peripheral Domain Access Control Word 0 (PDAC_W0_263) | 32 | RW | 0000_0000h |
| 183Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_263) | 32 | RW | 0000_0000h |
| 1840h | Peripheral Domain Access Control Word 0 (PDAC_W0_264) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1844h | Peripheral Domain Access Control Word 1 (PDAC_W1_264) | 32 | RW | 0000_0000h |
| 1848h | Peripheral Domain Access Control Word 0 (PDAC_W0_265) | 32 | RW | 0000_0000h |
| 184Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_265) | 32 | RW | 0000_0000h |
| 1850h | Peripheral Domain Access Control Word 0 (PDAC_W0_266) | 32 | RW | 0000_0000h |
| 1854h | Peripheral Domain Access Control Word 1 (PDAC_W1_266) | 32 | RW | 0000_0000h |
| 1858h | Peripheral Domain Access Control Word 0 (PDAC_W0_267) | 32 | RW | 0000_0000h |
| 185Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_267) | 32 | RW | 0000_0000h |
| 1860h | Peripheral Domain Access Control Word 0 (PDAC_W0_268) | 32 | RW | 0000_0000h |
| 1864h | Peripheral Domain Access Control Word 1 (PDAC_W1_268) | 32 | RW | 0000_0000h |
| 1868h | Peripheral Domain Access Control Word 0 (PDAC_W0_269) | 32 | RW | 0000_0000h |
| 186Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_269) | 32 | RW | 0000_0000h |
| 1870h | Peripheral Domain Access Control Word 0 (PDAC_W0_270) | 32 | RW | 0000_0000h |
| 1874h | Peripheral Domain Access Control Word 1 (PDAC_W1_270) | 32 | RW | 0000_0000h |
| 1878h | Peripheral Domain Access Control Word 0 (PDAC_W0_271) | 32 | RW | 0000_0000h |
| 187Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_271) | 32 | RW | 0000_0000h |
| 1880h | Peripheral Domain Access Control Word 0 (PDAC_W0_272) | 32 | RW | 0000_0000h |
| 1884h | Peripheral Domain Access Control Word 1 (PDAC_W1_272) | 32 | RW | 0000_0000h |
| 1888h | Peripheral Domain Access Control Word 0 (PDAC_W0_273) | 32 | RW | 0000_0000h |
| 188Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_273) | 32 | RW | 0000_0000h |
| 1890h | Peripheral Domain Access Control Word 0 (PDAC_W0_274) | 32 | RW | 0000_0000h |
| 1894h | Peripheral Domain Access Control Word 1 (PDAC_W1_274) | 32 | RW | 0000_0000h |
| 1898h | Peripheral Domain Access Control Word 0 (PDAC_W0_275) | 32 | RW | 0000_0000h |
| 189Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_275) | 32 | RW | 0000_0000h |
| 18A0h | Peripheral Domain Access Control Word 0 (PDAC_W0_276) | 32 | RW | 0000_0000h |
| 18A4h | Peripheral Domain Access Control Word 1 (PDAC_W1_276) | 32 | RW | 0000_0000h |
| 18A8h | Peripheral Domain Access Control Word 0 (PDAC_W0_277) | 32 | RW | 0000_0000h |
| 18ACh | Peripheral Domain Access Control Word 1 (PDAC_W1_277) | 32 | RW | 0000_0000h |
| 18B0h | Peripheral Domain Access Control Word 0 (PDAC_W0_278) | 32 | RW | 0000_0000h |
| 18B4h | Peripheral Domain Access Control Word 1 (PDAC_W1_278) | 32 | RW | 0000_0000h |
| 18B8h | Peripheral Domain Access Control Word 0 (PDAC_W0_279) | 32 | RW | 0000_0000h |
| 18BCh | Peripheral Domain Access Control Word 1 (PDAC_W1_279) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 18C0h | Peripheral Domain Access Control Word 0 (PDAC_W0_280) | 32 | RW | 0000_0000h |
| 18C4h | Peripheral Domain Access Control Word 1 (PDAC_W1_280) | 32 | RW | 0000_0000h |
| 18C8h | Peripheral Domain Access Control Word 0 (PDAC_W0_281) | 32 | RW | 0000_0000h |
| 18CCh | Peripheral Domain Access Control Word 1 (PDAC_W1_281) | 32 | RW | 0000_0000h |
| 18D8h | Peripheral Domain Access Control Word 0 (PDAC_W0_283) | 32 | RW | 0000_0000h |
| 18DCh | Peripheral Domain Access Control Word 1 (PDAC_W1_283) | 32 | RW | 0000_0000h |
| 18E8h | Peripheral Domain Access Control Word 0 (PDAC_W0_285) | 32 | RW | 0000_0000h |
| 18ECh | Peripheral Domain Access Control Word 1 (PDAC_W1_285) | 32 | RW | 0000_0000h |
| 1900h | Peripheral Domain Access Control Word 0 (PDAC_W0_288) | 32 | RW | 0000_0000h |
| 1904h | Peripheral Domain Access Control Word 1 (PDAC_W1_288) | 32 | RW | 0000_0000h |
| 1918h | Peripheral Domain Access Control Word 0 (PDAC_W0_291) | 32 | RW | 0000_0000h |
| 191Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_291) | 32 | RW | 0000_0000h |
| 1920h | Peripheral Domain Access Control Word 0 (PDAC_W0_292) | 32 | RW | 0000_0000h |
| 1924h | Peripheral Domain Access Control Word 1 (PDAC_W1_292) | 32 | RW | 0000_0000h |
| 1928h | Peripheral Domain Access Control Word 0 (PDAC_W0_293) | 32 | RW | 0000_0000h |
| 192Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_293) | 32 | RW | 0000_0000h |
| 1930h | Peripheral Domain Access Control Word 0 (PDAC_W0_294) | 32 | RW | 0000_0000h |
| 1934h | Peripheral Domain Access Control Word 1 (PDAC_W1_294) | 32 | RW | 0000_0000h |
| 1938h | Peripheral Domain Access Control Word 0 (PDAC_W0_295) | 32 | RW | 0000_0000h |
| 193Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_295) | 32 | RW | 0000_0000h |
| 1940h | Peripheral Domain Access Control Word 0 (PDAC_W0_296) | 32 | RW | 0000_0000h |
| 1944h | Peripheral Domain Access Control Word 1 (PDAC_W1_296) | 32 | RW | 0000_0000h |
| 1948h | Peripheral Domain Access Control Word 0 (PDAC_W0_297) | 32 | RW | 0000_0000h |
| 194Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_297) | 32 | RW | 0000_0000h |
| 1950h | Peripheral Domain Access Control Word 0 (PDAC_W0_298) | 32 | RW | 0000_0000h |
| 1954h | Peripheral Domain Access Control Word 1 (PDAC_W1_298) | 32 | RW | 0000_0000h |
| 1978h | Peripheral Domain Access Control Word 0 (PDAC_W0_303) | 32 | RW | 0000_0000h |
| 197Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_303) | 32 | RW | 0000_0000h |
| 1980h | Peripheral Domain Access Control Word 0 (PDAC_W0_304) | 32 | RW | 0000_0000h |
| 1984h | Peripheral Domain Access Control Word 1 (PDAC_W1_304) | 32 | RW | 0000_0000h |
| 1998h | Peripheral Domain Access Control Word 0 (PDAC_W0_307) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 199Ch | Peripheral Domain Access Control Word 1 (PDAC_W1_307) | 32 | RW | 0000_0000h |
| 19B8h | Peripheral Domain Access Control Word 0 (PDAC_W0_311) | 32 | RW | 0000_0000h |
| 19BCh | Peripheral Domain Access Control Word 1 (PDAC_W1_311) | 32 | RW | 0000_0000h |
| 19D0h | Peripheral Domain Access Control Word 0 (PDAC_W0_314) | 32 | RW | 0000_0000h |
| 19D4h | Peripheral Domain Access Control Word 1 (PDAC_W1_314) | 32 | RW | 0000_0000h |
| 19D8h | Peripheral Domain Access Control Word 0 (PDAC_W0_315) | 32 | RW | 0000_0000h |
| 19DCh | Peripheral Domain Access Control Word 1 (PDAC_W1_315) | 32 | RW | 0000_0000h |
| 2000h | Memory Region Descriptor Word 0 (MRGD_W0_0) | 32 | RW | 0000_0001h |
| 2004h | Memory Region Descriptor Word 1 (MRGD_W1_0) | 32 | RW | 0000_001Fh |
| 2008h | Memory Region Descriptor Word 2 (MRGD_W2_0) | 32 | RW | 0000_0000h |
| 200Ch | Memory Region Descriptor Word 3 (MRGD_W3_0) | 32 | RW | 0000_0000h |
| 2020h | Memory Region Descriptor Word 0 (MRGD_W0_1) | 32 | RW | 0000_0001h |
| 2024h | Memory Region Descriptor Word 1 (MRGD_W1_1) | 32 | RW | 0000_001Fh |
| 2028h | Memory Region Descriptor Word 2 (MRGD_W2_1) | 32 | RW | 0000_0000h |
| 202Ch | Memory Region Descriptor Word 3 (MRGD_W3_1) | 32 | RW | 0000_0000h |
| 2040h | Memory Region Descriptor Word 0 (MRGD_W0_2) | 32 | RW | 0000_0001h |
| 2044h | Memory Region Descriptor Word 1 (MRGD_W1_2) | 32 | RW | 0000_001Fh |
| 2048h | Memory Region Descriptor Word 2 (MRGD_W2_2) | 32 | RW | 0000_0000h |
| 204Ch | Memory Region Descriptor Word 3 (MRGD_W3_2) | 32 | RW | 0000_0000h |
| 2060h | Memory Region Descriptor Word 0 (MRGD_W0_3) | 32 | RW | 0000_0001h |
| 2064h | Memory Region Descriptor Word 1 (MRGD_W1_3) | 32 | RW | 0000_001Fh |
| 2068h | Memory Region Descriptor Word 2 (MRGD_W2_3) | 32 | RW | 0000_0000h |
| 206Ch | Memory Region Descriptor Word 3 (MRGD_W3_3) | 32 | RW | 0000_0000h |
| 2080h | Memory Region Descriptor Word 0 (MRGD_W0_4) | 32 | RW | 0000_0001h |
| 2084h | Memory Region Descriptor Word 1 (MRGD_W1_4) | 32 | RW | 0000_001Fh |
| 2088h | Memory Region Descriptor Word 2 (MRGD_W2_4) | 32 | RW | 0000_0000h |
| 208Ch | Memory Region Descriptor Word 3 (MRGD_W3_4) | 32 | RW | 0000_0000h |
| 20A0h | Memory Region Descriptor Word 0 (MRGD_W0_5) | 32 | RW | 0000_0001h |
| 20A4h | Memory Region Descriptor Word 1 (MRGD_W1_5) | 32 | RW | 0000_001Fh |
| 20A8h | Memory Region Descriptor Word 2 (MRGD_W2_5) | 32 | RW | 0000_0000h |
| 20ACh | Memory Region Descriptor Word 3 (MRGD_W3_5) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 20C0h | Memory Region Descriptor Word 0 (MRGD_W0_6) | 32 | RW | 0000_0001h |
| 20C4h | Memory Region Descriptor Word 1 (MRGD_W1_6) | 32 | RW | 0000_001Fh |
| 20C8h | Memory Region Descriptor Word 2 (MRGD_W2_6) | 32 | RW | 0000_0000h |
| 20CCh | Memory Region Descriptor Word 3 (MRGD_W3_6) | 32 | RW | 0000_0000h |
| 20E0h | Memory Region Descriptor Word 0 (MRGD_W0_7) | 32 | RW | 0000_0001h |
| 20E4h | Memory Region Descriptor Word 1 (MRGD_W1_7) | 32 | RW | 0000_001Fh |
| 20E8h | Memory Region Descriptor Word 2 (MRGD_W2_7) | 32 | RW | 0000_0000h |
| 20ECh | Memory Region Descriptor Word 3 (MRGD_W3_7) | 32 | RW | 0000_0000h |
| 2100h | Memory Region Descriptor Word 0 (MRGD_W0_8) | 32 | RW | 0000_0001h |
| 2104h | Memory Region Descriptor Word 1 (MRGD_W1_8) | 32 | RW | 0000_001Fh |
| 2108h | Memory Region Descriptor Word 2 (MRGD_W2_8) | 32 | RW | 0000_0000h |
| 210Ch | Memory Region Descriptor Word 3 (MRGD_W3_8) | 32 | RW | 0000_0000h |
| 2120h | Memory Region Descriptor Word 0 (MRGD_W0_9) | 32 | RW | 0000_0001h |
| 2124h | Memory Region Descriptor Word 1 (MRGD_W1_9) | 32 | RW | 0000_001Fh |
| 2128h | Memory Region Descriptor Word 2 (MRGD_W2_9) | 32 | RW | 0000_0000h |
| 212Ch | Memory Region Descriptor Word 3 (MRGD_W3_9) | 32 | RW | 0000_0000h |
| 2140h | Memory Region Descriptor Word 0 (MRGD_W0_10) | 32 | RW | 0000_0001h |
| 2144h | Memory Region Descriptor Word 1 (MRGD_W1_10) | 32 | RW | 0000_001Fh |
| 2148h | Memory Region Descriptor Word 2 (MRGD_W2_10) | 32 | RW | 0000_0000h |
| 214Ch | Memory Region Descriptor Word 3 (MRGD_W3_10) | 32 | RW | 0000_0000h |
| 2160h | Memory Region Descriptor Word 0 (MRGD_W0_11) | 32 | RW | 0000_0001h |
| 2164h | Memory Region Descriptor Word 1 (MRGD_W1_11) | 32 | RW | 0000_001Fh |
| 2168h | Memory Region Descriptor Word 2 (MRGD_W2_11) | 32 | RW | 0000_0000h |
| 216Ch | Memory Region Descriptor Word 3 (MRGD_W3_11) | 32 | RW | 0000_0000h |
| 2180h | Memory Region Descriptor Word 0 (MRGD_W0_12) | 32 | RW | 0000_0001h |
| 2184h | Memory Region Descriptor Word 1 (MRGD_W1_12) | 32 | RW | 0000_001Fh |
| 2188h | Memory Region Descriptor Word 2 (MRGD_W2_12) | 32 | RW | 0000_0000h |
| 218Ch | Memory Region Descriptor Word 3 (MRGD_W3_12) | 32 | RW | 0000_0000h |
| 21A0h | Memory Region Descriptor Word 0 (MRGD_W0_13) | 32 | RW | 0000_0001h |
| 21A4h | Memory Region Descriptor Word 1 (MRGD_W1_13) | 32 | RW | 0000_001Fh |
| 21A8h | Memory Region Descriptor Word 2 (MRGD_W2_13) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 21ACh | Memory Region Descriptor Word 3 (MRGD_W3_13) | 32 | RW | 0000_0000h |
| 21C0h | Memory Region Descriptor Word 0 (MRGD_W0_14) | 32 | RW | 0000_0001h |
| 21C4h | Memory Region Descriptor Word 1 (MRGD_W1_14) | 32 | RW | 0000_001Fh |
| 21C8h | Memory Region Descriptor Word 2 (MRGD_W2_14) | 32 | RW | 0000_0000h |
| 21CCh | Memory Region Descriptor Word 3 (MRGD_W3_14) | 32 | RW | 0000_0000h |
| 21E0h | Memory Region Descriptor Word 0 (MRGD_W0_15) | 32 | RW | 0000_0001h |
| 21E4h | Memory Region Descriptor Word 1 (MRGD_W1_15) | 32 | RW | 0000_001Fh |
| 21E8h | Memory Region Descriptor Word 2 (MRGD_W2_15) | 32 | RW | 0000_0000h |
| 21ECh | Memory Region Descriptor Word 3 (MRGD_W3_15) | 32 | RW | 0000_0000h |
| 2200h | Memory Region Descriptor Word 0 (MRGD_W0_16) | 32 | RW | 0000_0001h |
| 2204h | Memory Region Descriptor Word 1 (MRGD_W1_16) | 32 | RW | 0000_001Fh |
| 2208h | Memory Region Descriptor Word 2 (MRGD_W2_16) | 32 | RW | 0000_0000h |
| 220Ch | Memory Region Descriptor Word 3 (MRGD_W3_16) | 32 | RW | 0000_0000h |
| 2220h | Memory Region Descriptor Word 0 (MRGD_W0_17) | 32 | RW | 0000_0001h |
| 2224h | Memory Region Descriptor Word 1 (MRGD_W1_17) | 32 | RW | 0000_001Fh |
| 2228h | Memory Region Descriptor Word 2 (MRGD_W2_17) | 32 | RW | 0000_0000h |
| 222Ch | Memory Region Descriptor Word 3 (MRGD_W3_17) | 32 | RW | 0000_0000h |
| 2240h | Memory Region Descriptor Word 0 (MRGD_W0_18) | 32 | RW | 0000_0001h |
| 2244h | Memory Region Descriptor Word 1 (MRGD_W1_18) | 32 | RW | 0000_001Fh |
| 2248h | Memory Region Descriptor Word 2 (MRGD_W2_18) | 32 | RW | 0000_0000h |
| 224Ch | Memory Region Descriptor Word 3 (MRGD_W3_18) | 32 | RW | 0000_0000h |
| 2260h | Memory Region Descriptor Word 0 (MRGD_W0_19) | 32 | RW | 0000_0001h |
| 2264h | Memory Region Descriptor Word 1 (MRGD_W1_19) | 32 | RW | 0000_001Fh |
| 2268h | Memory Region Descriptor Word 2 (MRGD_W2_19) | 32 | RW | 0000_0000h |
| 226Ch | Memory Region Descriptor Word 3 (MRGD_W3_19) | 32 | RW | 0000_0000h |
| 2280h | Memory Region Descriptor Word 0 (MRGD_W0_20) | 32 | RW | 0000_0001h |
| 2284h | Memory Region Descriptor Word 1 (MRGD_W1_20) | 32 | RW | 0000_001Fh |
| 2288h | Memory Region Descriptor Word 2 (MRGD_W2_20) | 32 | RW | 0000_0000h |
| 228Ch | Memory Region Descriptor Word 3 (MRGD_W3_20) | 32 | RW | 0000_0000h |
| 22A0h | Memory Region Descriptor Word 0 (MRGD_W0_21) | 32 | RW | 0000_0001h |
| 22A4h | Memory Region Descriptor Word 1 (MRGD_W1_21) | 32 | RW | 0000_001Fh |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 22A8h | Memory Region Descriptor Word 2 (MRGD_W2_21) | 32 | RW | 0000_0000h |
| 22ACh | Memory Region Descriptor Word 3 (MRGD_W3_21) | 32 | RW | 0000_0000h |
| 22C0h | Memory Region Descriptor Word 0 (MRGD_W0_22) | 32 | RW | 0000_0001h |
| 22C4h | Memory Region Descriptor Word 1 (MRGD_W1_22) | 32 | RW | 0000_001Fh |
| 22C8h | Memory Region Descriptor Word 2 (MRGD_W2_22) | 32 | RW | 0000_0000h |
| 22CCh | Memory Region Descriptor Word 3 (MRGD_W3_22) | 32 | RW | 0000_0000h |
| 22E0h | Memory Region Descriptor Word 0 (MRGD_W0_23) | 32 | RW | 0000_0001h |
| 22E4h | Memory Region Descriptor Word 1 (MRGD_W1_23) | 32 | RW | 0000_001Fh |
| 22E8h | Memory Region Descriptor Word 2 (MRGD_W2_23) | 32 | RW | 0000_0000h |
| 22ECh | Memory Region Descriptor Word 3 (MRGD_W3_23) | 32 | RW | 0000_0000h |
| 2300h | Memory Region Descriptor Word 0 (MRGD_W0_24) | 32 | RW | 0000_0001h |
| 2304h | Memory Region Descriptor Word 1 (MRGD_W1_24) | 32 | RW | 0000_001Fh |
| 2308h | Memory Region Descriptor Word 2 (MRGD_W2_24) | 32 | RW | 0000_0000h |
| 230Ch | Memory Region Descriptor Word 3 (MRGD_W3_24) | 32 | RW | 0000_0000h |
| 2320h | Memory Region Descriptor Word 0 (MRGD_W0_25) | 32 | RW | 0000_0001h |
| 2324h | Memory Region Descriptor Word 1 (MRGD_W1_25) | 32 | RW | 0000_001Fh |
| 2328h | Memory Region Descriptor Word 2 (MRGD_W2_25) | 32 | RW | 0000_0000h |
| 232Ch | Memory Region Descriptor Word 3 (MRGD_W3_25) | 32 | RW | 0000_0000h |
| 2340h | Memory Region Descriptor Word 0 (MRGD_W0_26) | 32 | RW | 0000_0001h |
| 2344h | Memory Region Descriptor Word 1 (MRGD_W1_26) | 32 | RW | 0000_001Fh |
| 2348h | Memory Region Descriptor Word 2 (MRGD_W2_26) | 32 | RW | 0000_0000h |
| 234Ch | Memory Region Descriptor Word 3 (MRGD_W3_26) | 32 | RW | 0000_0000h |
| 2360h | Memory Region Descriptor Word 0 (MRGD_W0_27) | 32 | RW | 0000_0001h |
| 2364h | Memory Region Descriptor Word 1 (MRGD_W1_27) | 32 | RW | 0000_001Fh |
| 2368h | Memory Region Descriptor Word 2 (MRGD_W2_27) | 32 | RW | 0000_0000h |
| 236Ch | Memory Region Descriptor Word 3 (MRGD_W3_27) | 32 | RW | 0000_0000h |
| 2380h | Memory Region Descriptor Word 0 (MRGD_W0_28) | 32 | RW | 0000_0001h |
| 2384h | Memory Region Descriptor Word 1 (MRGD_W1_28) | 32 | RW | 0000_001Fh |
| 2388h | Memory Region Descriptor Word 2 (MRGD_W2_28) | 32 | RW | 0000_0000h |
| 238Ch | Memory Region Descriptor Word 3 (MRGD_W3_28) | 32 | RW | 0000_0000h |
| 23A0h | Memory Region Descriptor Word 0 (MRGD_W0_29) | 32 | RW | 0000_0001h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 23A4h | Memory Region Descriptor Word 1 (MRGD_W1_29) | 32 | RW | 0000_001Fh |
| 23A8h | Memory Region Descriptor Word 2 (MRGD_W2_29) | 32 | RW | 0000_0000h |
| 23ACh | Memory Region Descriptor Word 3 (MRGD_W3_29) | 32 | RW | 0000_0000h |
| 23C0h | Memory Region Descriptor Word 0 (MRGD_W0_30) | 32 | RW | 0000_0001h |
| 23C4h | Memory Region Descriptor Word 1 (MRGD_W1_30) | 32 | RW | 0000_001Fh |
| 23C8h | Memory Region Descriptor Word 2 (MRGD_W2_30) | 32 | RW | 0000_0000h |
| 23CCh | Memory Region Descriptor Word 3 (MRGD_W3_30) | 32 | RW | 0000_0000h |
| 23E0h | Memory Region Descriptor Word 0 (MRGD_W0_31) | 32 | RW | 0000_0001h |
| 23E4h | Memory Region Descriptor Word 1 (MRGD_W1_31) | 32 | RW | 0000_001Fh |
| 23E8h | Memory Region Descriptor Word 2 (MRGD_W2_31) | 32 | RW | 0000_0000h |
| 23ECh | Memory Region Descriptor Word 3 (MRGD_W3_31) | 32 | RW | 0000_0000h |
| 2400h | Memory Region Descriptor Word 0 (MRGD_W0_32) | 32 | RW | 0000_0001h |
| 2404h | Memory Region Descriptor Word 1 (MRGD_W1_32) | 32 | RW | 0000_001Fh |
| 2408h | Memory Region Descriptor Word 2 (MRGD_W2_32) | 32 | RW | 0000_0000h |
| 240Ch | Memory Region Descriptor Word 3 (MRGD_W3_32) | 32 | RW | 0000_0000h |
| 2420h | Memory Region Descriptor Word 0 (MRGD_W0_33) | 32 | RW | 0000_0001h |
| 2424h | Memory Region Descriptor Word 1 (MRGD_W1_33) | 32 | RW | 0000_001Fh |
| 2428h | Memory Region Descriptor Word 2 (MRGD_W2_33) | 32 | RW | 0000_0000h |
| 242Ch | Memory Region Descriptor Word 3 (MRGD_W3_33) | 32 | RW | 0000_0000h |
| 2440h | Memory Region Descriptor Word 0 (MRGD_W0_34) | 32 | RW | 0000_0001h |
| 2444h | Memory Region Descriptor Word 1 (MRGD_W1_34) | 32 | RW | 0000_001Fh |
| 2448h | Memory Region Descriptor Word 2 (MRGD_W2_34) | 32 | RW | 0000_0000h |
| 244Ch | Memory Region Descriptor Word 3 (MRGD_W3_34) | 32 | RW | 0000_0000h |
| 2460h | Memory Region Descriptor Word 0 (MRGD_W0_35) | 32 | RW | 0000_0001h |
| 2464h | Memory Region Descriptor Word 1 (MRGD_W1_35) | 32 | RW | 0000_001Fh |
| 2468h | Memory Region Descriptor Word 2 (MRGD_W2_35) | 32 | RW | 0000_0000h |
| 246Ch | Memory Region Descriptor Word 3 (MRGD_W3_35) | 32 | RW | 0000_0000h |

### 18.8.3.2 Control (CR)

#### Offset

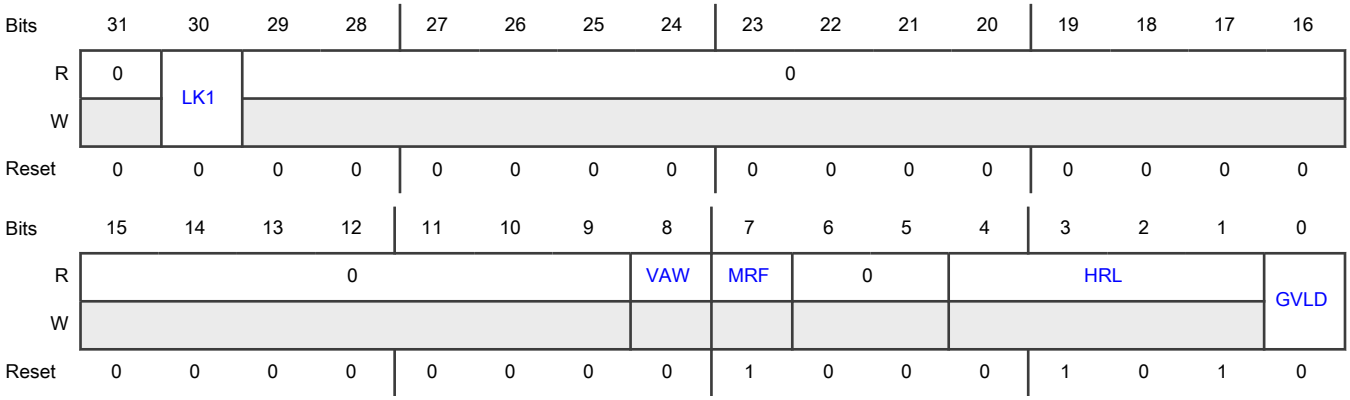
| Register | Offset |
|---|---|
| CR | 0h |

#### Function

Provides XRDC status and enables XRDC operation.

Access: Secure privileged read/write

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>LK1 | Lock<br>Prohibits writes to this register.<br>• If unlocked, this register accepts any secure privileged write.<br>• If locked, you cannot write to this register and it remains read-only until after the next reset.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Unlocked<br>1b - Locks<br>When writing |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No effect<br>1b - Locks |
| 29-9<br>— | Reserved |
| 8<br>VAW | Virtualization Aware<br>Indicates whether domain assignments support the optional inclusion of a logical partition identifier, which is also known as an operating system number or Arm virtual machine identifier (VMID).<br>0b - Not virtualization-aware<br>1b - Virtualization-aware |
| 7<br>MRF | Memory Region Format<br>Indicates the format of memory region descriptors.<br>0b - Reserved<br>1b - SMPU family format |
| 6-5<br>— | Reserved |
| 4-1<br>HRL | Hardware Revision Level<br>Indicates the XRDC hardware revision level, which is associated with a set of functional characteristics of the module. |
| 0<br>GVLD | Global Valid (XRDC Global Enable/Disable)<br>Enables XRDC. When XRDC is disabled, all bus initiators can access all targets.<br>0b - Disables<br>1b - Enables |

### 18.8.3.3 Hardware Configuration 0 (HWCFG0)

#### Offset

| Register | Offset |
|---|---|
| HWCFG0 | F0h |

#### Function

Indicates XRDC configuration details, including:

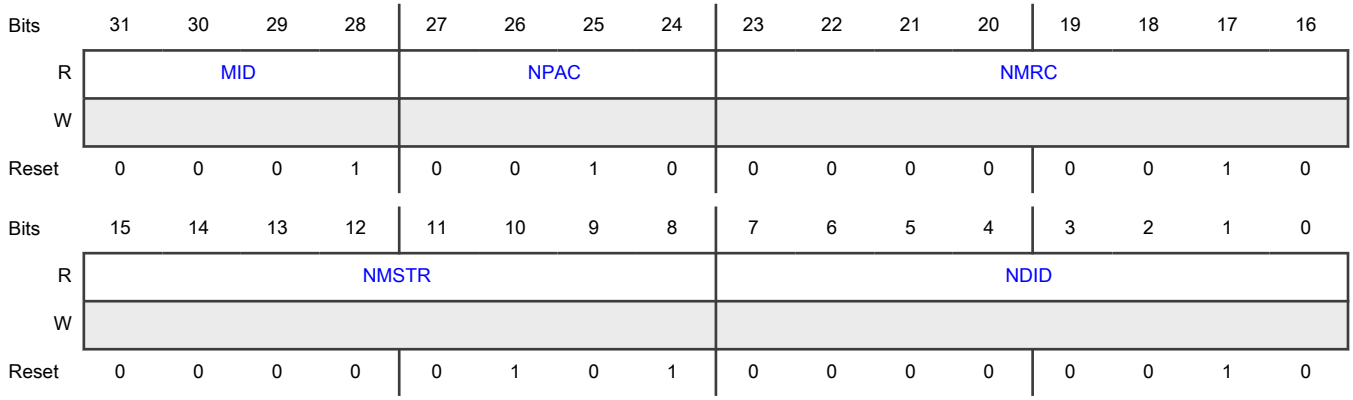
- XRDC module ID
- Number of implemented domains

- Number of bus initiators
- Number of MRCs
- Number of PACs

Attempting to write to this register causes an error.

Access: Secure privileged read

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-28<br>MID | Module ID |
| 27-24<br>NPAC | Number Of PACs<br>Indicates the number of PACs minus 1. In other words, the actual number of PACs is NPAC + 1. |
| 23-16<br>NMRC | Number of MRCs<br>Indicates the number of MRCs minus 1. In other words, the actual number of MRCs is NMRC + 1. |
| 15-8<br>NMSTR | Number Of Bus Initiators<br>Indicates the number of bus initiators minus 1. In other words, the actual number of bus initiators is NMSTR + 1. |
| 7-0<br>NDID | Number Of DIDs<br>Indicates the number of domains (DIDs) minus 1. In other words, the actual number of DIDs is NDID + 1. |

## 18.8.3.4 Hardware Configuration 1 (HWCFG1)

### Offset

| Register | Offset |
|---|---|
| HWCFG1 | F4h |

### Function

Indicates the DID of the bus initiator making the current transaction request. See Domain error capture management for information about typical usage.

Attempting to write to this register causes an error.

Access: Secure privileged read

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | DID | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | u[1] | u | u | u |

1. The reset value is determined by the current configuration of the accessing initiator.

### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>DID | Domain Identifier<br>Indicates the DID of the requesting bus initiator. |

## 18.8.3.5 Hardware Configuration 2 (HWCFG2)

### Offset

| Register | Offset |
|---|---|
| HWCFG2 | F8h |

### Function

For initiators 0–31, indicates whether a given initiator has a built-in PID register as part of its programming model. If not, you must use the corresponding PIDm register to mimic the functionality of a built-in PID register.

Each bit corresponds to the same numbered initiator. For example, if PIDP18 is 1, bus initiator 18 has its own PID register. If PIDP18 is 0, then initiator 18 does not have its own PID register and you must use PID18.

Attempting to write to this register causes an error.

Access: Secure privileged read

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PIDP31 | PIDP30 | PIDP29 | PIDP28 | PIDP27 | PIDP26 | PIDP25 | PIDP24 | PIDP23 | PIDP22 | PIDP21 | PIDP20 | PIDP19 | PIDP18 | PIDP17 | PIDP16 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PIDP15 | PIDP14 | PIDP13 | PIDP12 | PIDP11 | PIDP10 | PIDP9 | PIDP8 | PIDP7 | PIDP6 | PIDP5 | PIDP4 | PIDP3 | PIDP2 | PIDP1 | PIDP0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>PIDPn | Process Identifier Present<br>0b - Does not have PID register<br>1b - Has PID register |

## 18.8.3.6 Master Domain Assignment Configuration (MDACFG0 - MDACFG5)

### Offset

| Register | Offset |
|---|---|
| MDACFG0 | 100h |
| MDACFG1 | 101h |
| MDACFG2 | 102h |
| MDACFG3 | 103h |
| MDACFG5 | 105h |

### Function

Indicates the number of implemented master domain assignment registers (MDA_W*w*_*m*_DFMT0 or MDA_W*w*_*m*_DFMT1) for initiator *m*, where *m* ranges from 0 to 63. You can read these registers using 8-, 16-, or 32-bit accesses.

If NMDAR is 0, the associated initiator does not exist.

Attempting to write to this register causes an error.

Access: Secure privileged read

### Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | NCM | 0 | | | NMDAR | | | |
| W | | | | | | | | |
| Reset | See Register reset values. | | | | | | | |

### Register reset values

| Register | Reset value |
|---|---|
| MDACFG0 | 01h |
| MDACFG1–MDACFG2 | 81h |
| MDACFG3 | 01h |
| MDACFG4 | — |
| MDACFG5 | 81h |

### Fields

| Field | Function |
|---|---|
| 7<br>NCM | Noncore Master<br>If NMDAR is greater than zero, indicates whether initiator *m* uses MDA_W*w*_*m*_DFMT0 or MDA_W*w*_*m*_DFMT1 to configure domain assignment.<br>This field is 0 for a non-existent initiator.<br>0b - Core initiator or initiator does not exist<br>1b - Noncore initiator |
| 6-4<br>— | Reserved |
| 3-0<br>NMDAR | Number Of Master Domain Assignment Registers<br>Indicates the number of master domain assignment registers (MDA_W*w*_*m*_DFMT0 or MDA_W*w*_*m*_DFMT1) associated with initiator *m*.<br>0000b - Initiator does not exist<br>0001b-1000b - Number of registers<br>All other values are reserved. |

### 18.8.3.7 Memory Region Configuration (MRCFG0 - MRCFG2)

#### Offset

| Register | Offset |
|---|---|
| MRCFG0 | 140h |
| MRCFG1 | 141h |
| MRCFG2 | 142h |

#### Function

Indicates the number of memory region descriptors (*r*) for MRC*c*, from 4 to 16 in increments of four, with 0 indicating a non-existent MRC. These registers are organized as byte-sized data arrays and can be read using 8-, 16-, or 32-bit accesses.

Attempting to write to this register causes an error.

Access: Secure Privileged Read

#### Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | NMRGD | | | |
| W | | | | | | | | |
| Reset | See Register reset values. | | | | | | | |

#### Register reset values

| Register | Reset value |
|---|---|
| MRCFG0–MRCFG1 | 10h |
| MRCFG2 | 04h |

#### Fields

| Field | Function |
|---|---|
| 7-5<br>— | Reserved |
| 4-0<br>NMRGD | Number Of Memory Region Descriptors<br>Indicates the number of memory region descriptors associated with the MRC.<br>0_0000b - MRC does not exist<br>0_0100b - 4<br>0_1000b - 8 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0_1100b - 12<br>1_0000b - 16<br>All other values are reserved. |

### 18.8.3.8 Domain Error Location (DERRLOC0 - DERRLOC2)

#### Offset

| Register | Offset |
|---|---|
| DERRLOC0 | 200h |
| DERRLOC1 | 204h |
| DERRLOC2 | 208h |

#### Function

Indicates the MRC or PAC instance in domain *d* where an access violation has occurred. Each bit corresponds to the like-numbered instance. For more information, see Domain error capture management.

Attempting to write to this register causes an error.

Access: Secure privileged read

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 19-16<br>PACINST | PAC Instance<br>Indicates the presence of a detected access violation for domain *d* in a PAC instance. Each bit corresponds to the instance index for the DERR_W*w*_*i* registers: bit 16 of the register corresponds to the DERR_W*w*_*16* registers, which show error information for PAC 0, and so on. Multiple bits can be 1 at any time, indicating access violations have been detected across multiple PACs.<br>For each bit in this field:<br>0b - No access violation error or the PAC instance is not physically present<br>1b - Access violation detected |
| 15-0<br>MRCINST | MRC Instance<br>Indicates the presence of a detected access violation for domain *d* in an MRC instance. Each bit corresponds to the like-numbered MRC instance: bit 0 (bit 0 of the register) corresponds to MRC instance 0, and so on. Multiple bits can be 1 at any time, indicating access violations have been detected across multiple MRCs.<br>For each bit in this field:<br>0b - No access violation error or the MRC instance is not physically present<br>1b - Access violation detected |

### 18.8.3.9 Domain Error Word 0 (DERR_W0_0 - DERR_W0_18)

#### Offset

| Register | Offset |
|---|---|
| DERR_W0_0 | 400h |
| DERR_W0_1 | 410h |
| DERR_W0_2 | 420h |
| DERR_W0_16 | 500h |
| DERR_W0_17 | 510h |
| DERR_W0_18 | 520h |

#### Function

Indicates the address of an access violation detected by an MRC or a PAC, indexed by the MRC or PAC instance (*i*) that detected the violation, as indicated in DERRLOC*d*. This register is part of a 16-byte set:

- DERR_W0_*i*: Word 0, the first 4 bytes
- DERR_W1_*i*: Word 1, the second 4 bytes
- Word 2, 4 reserved bytes
- DERR_W3_*i*: Word 3, the fourth 4 bytes

The first 16 sets (*i* from 0 to 15) are associated with MRCs and the rest (starting with *i* = 16) are associated with PACs. For more information, see Domain error capture management.

The error capture registers in the memory region controller and peripheral access controller submodules contain physical registers for each domain, but are organized in the DERR_Wn registers to provide the information for the requesting domain only. The registers return the error information for the domain id that is used to read the registers. When no error occurred for this domain id, it returns 0 data.

When the error capture logic is rearmed by writing the required data pattern to DERR_W3_n, this register is cleared.

When XRDC detects an access violation, it captures the error information and disables subsequent updates to the error capture registers until you write to DERR_W3_*i*.

**NOTE**

If initiators with the same DID cause simultaneous error accesses, the error capture registers record only the error of the lowest target index.

Attempting to write to this register causes an error. Attempting to read the error registers for a non-existent MRC or PAC instance causes an error.

Access: Secure privileged read

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>EADDR | Error Address<br>Indicates the target address of the first transaction that causes an access violation after reset or after rearming error capture. |

## 18.8.3.10 Domain Error Word 1 (DERR_W1_0 - DERR_W1_18)

### Offset

| Register | Offset |
|---|---|
| DERR_W1_0 | 404h |
| DERR_W1_1 | 414h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| DERR_W1_2 | 424h |
| DERR_W1_16 | 504h |
| DERR_W1_17 | 514h |
| DERR_W1_18 | 524h |

### Function

Indicates the attributes of an access violation detected by an MRC or a PAC, indexed by the MRC or PAC instance (*i*) that detected the access violation, as indicated in DERRLOC*d*. For more information, see DERR_W0_*i* and Domain error capture management.

The error capture registers in the memory region controller and peripheral access controller submodules contain physical registers for each domain, but are organized in the DERR_Wn registers to provide the information for the requesting domain only. The registers return the error information for the domain id that is used to read the registers. When no error occurred for this domain id, it returns 0 data.

When the error capture logic is rearmed by writing the required data pattern to DERR_W3_n, this register is cleared.

**NOTE**

If initiators with the same DID cause simultaneous error accesses, the error capture registers record only the error of the lowest target index.

Attempting to write to this register causes an error. Attempting to read the error registers for a non-existent MRC or PAC instance causes an error.

Access: Secure privileged read

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | ERW | EATR | | | 0 | | | | EDID | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-30<br>EST | Error State |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Indicates the state of access violations for this domain in this instance of the MRC or PAC. After the first access violation to occur after reset or rearming of error capture, XRDC records subsequent errors as an overrun condition without any data captured.<br>00b-01b - No access violations detected<br>10b - A single access violation has been detected<br>11b - Multiple access violations have been detected |
| 29-27<br>— | Reserved |
| 26-24<br>EPORT | Error Port<br>Identifies the encoded port number of the MRC that detected the access violation. See the chip-specific configuration details for the MRC port connections. For access violations detected by a PAC, this field is zero. |
| 23-12<br>— | Reserved |
| 11<br>ERW | Error Read Or Write<br>Indicates whether the captured access violation occurred on a read or write access.<br>0b - Read access<br>1b - Write access |
| 10-8<br>EATR | Error Attributes<br>Indicates attributes of the access violation.<br>000b - Secure user mode, instruction fetch access<br>001b - Secure user mode, data access<br>010b - Secure privileged mode, instruction fetch access<br>011b - Secure privileged mode, data access<br>100b - Nonsecure user mode, instruction fetch access<br>101b - Nonsecure user mode, data access<br>110b - Nonsecure privileged mode, instruction fetch access<br>111b - Nonsecure privileged mode, data access |
| 7-4<br>— | Reserved |
| 3-0<br>EDID | Error Domain Identifier<br>Indicates the DID of the access violation. |

### 18.8.3.11 Domain Error Word 3 (DERR_W3_0 - DERR_W3_18)

#### Offset

| Register | Offset |
|---|---|
| DERR_W3_0 | 40Ch |
| DERR_W3_1 | 41Ch |
| DERR_W3_2 | 42Ch |
| DERR_W3_16 | 50Ch |
| DERR_W3_17 | 51Ch |
| DERR_W3_18 | 52Ch |

#### Function

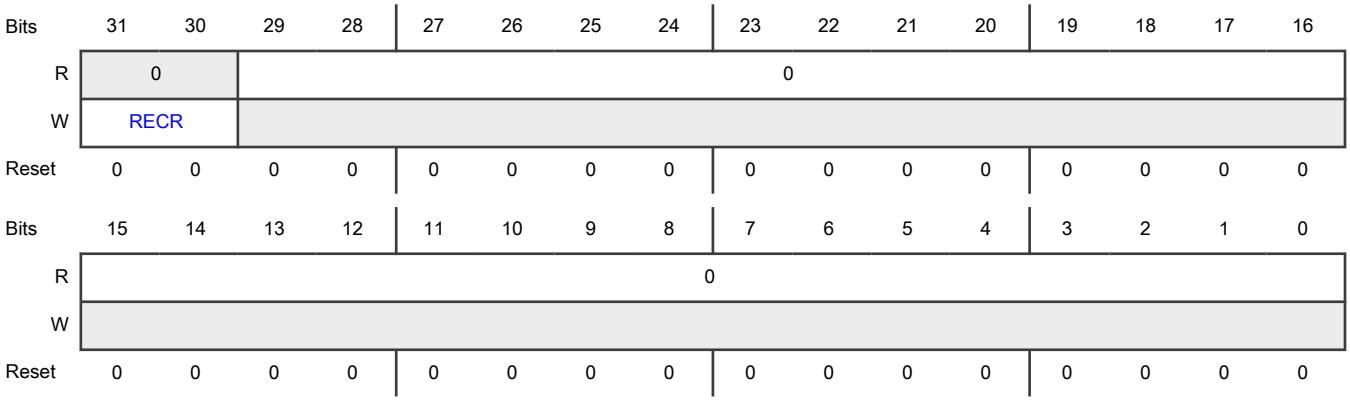
Rearms instance error capture, resets the error capture registers (DERR_W0_*d*, DERR_W1_*d*), and deasserts the instance bit in DERRLOC*d*. After reading the access violation error information, an error handler must write 1 to RECR.

Register write only rearms error capture registers for the domain id that is used to write this register.

This register returns 0000h when read. Attempted reads of an MRC or PAC instance that is not physically present cause an error.

For more information, see Domain error capture management.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-30 | Rearm Error Capture Registers |
| RECR | Resets and rearms the domain error capture registers for this instance, including deasserting the instance bit in DERRLOC*d*.<br>00b,10b,11b - No effect<br>01b - Rearms error capture, resets error capture registers, and deasserts instance bit in DERRLOCd |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 29-0<br>— | Reserved |

### 18.8.3.12 Process Identifier (PID0 - PID3)

#### Offset

| Register | Offset |
|---|---|
| PID0 | 700h |
| PID3 | 70Ch |

#### Function

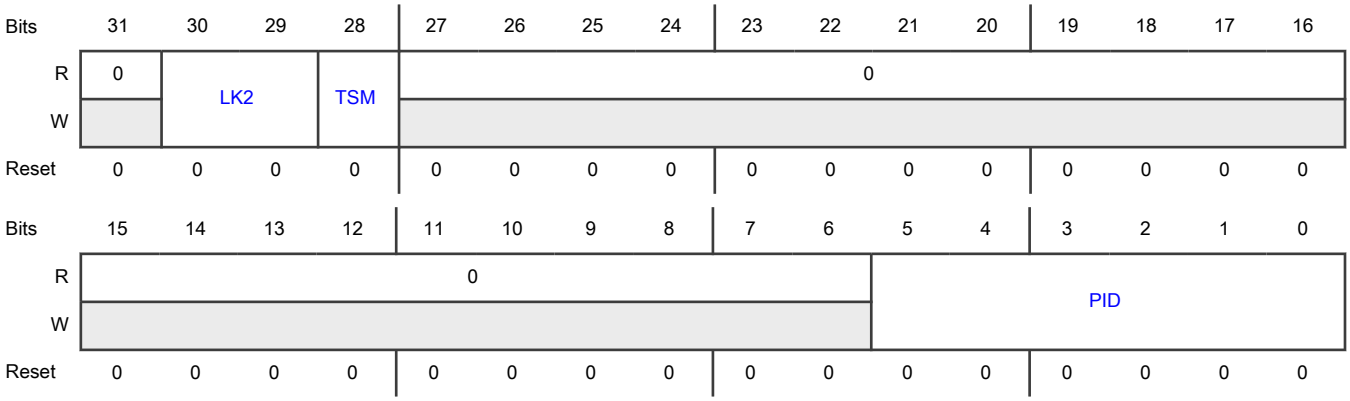
Specifies the PID for the associated core initiator *m*.

Some cores contain a built-in PID register. If the core has a built-in PID register, XRDC populates the PID field with the value from the core PID register. If the core does not have the built-in register, the XRDC PID register allows applications to mimic PID operation for that core by writing the desired PID to the associated register.

HWCFG2 provides a bitmap of the implemented PID*m* registers. Noncore initiators do not have an associated PID register.

For information about PID-based operation, see PID-based domain assignment.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-29 | Lock |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| LK2 | Limits or prohibits writes to this register.<br>When you assert a bit in this field, it remains asserted until the next module reset.<br>If the core initiator has a built-in PID register, as indicated in HWCFG2, then a secure privileged read returns 0 for this field.<br>00b,01b - Any secure privileged write<br>10b - Secure privileged writes from initiator only<br>11b - Locks |
| 28<br>TSM | Three-State Model<br>Specifies that the core initiator supports only the three-state access control model. If you write 1 to this field, it remains asserted until the next reset.<br>For cores that support only the three-state access control model, you must assert this field before loading any nonsecure value into the PID.<br>See Generation of secure attribute. |
| 27-6<br>— | Reserved |
| 5-0<br>PID | Process Identifier<br>Specifies the transaction PID for the corresponding core initiator.<br>If the core has a built-in PID register, then a secure privileged read returns the core's PID register value.<br>Bit 5 specifies the secure attribute (0 = secure, 1 = nonsecure) for the transaction. |

### 18.8.3.13 Master Domain Assignment (MDA_W0_0_DFMT0 - MDA_W0_3_DFMT0)

#### Offset

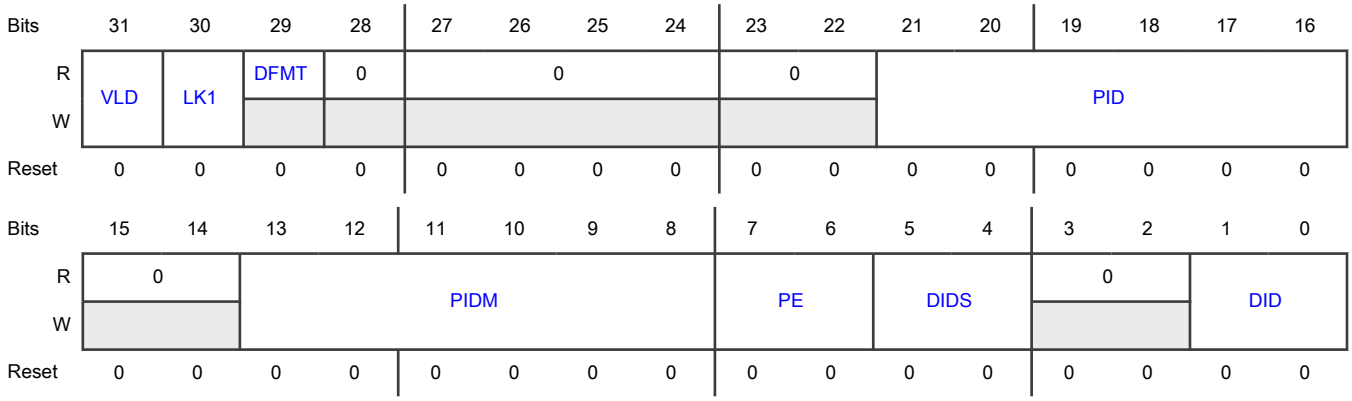
| Register | Offset |
|---|---|
| MDA_W0_0_DFMT0 | 800h |
| MDA_W0_3_DFMT0 | 860h |

#### Function

Specifies the information used by the MDAC to assign a core bus initiator to a specific domain (DID). For more information, see Master domain assignment controller (MDAC).

Access: Secure privileged read/write

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>VLD | Valid<br>Specifies whether this domain assignment is valid. In other words, if VLD and CR[GVLD] are both asserted, XRDC uses the configuration in this register in the domain assignment process. If CR[GVLD] is set to 1 and VLD is set to 0 then every transaction from this initiator will be assigned a DID of 0.<br>This field has no effect unless XRDC is enabled (CR[GVLD] = 1).<br>0b - Invalid<br>1b - Valid |
| 30<br>LK1 | Lock<br>Prohibits writes to this register.<br>• If unlocked, this register accepts any secure privileged write.<br>• If locked, you cannot write to this register and it remains read-only until after the next reset.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Unlocked<br>1b - Locks<br>When writing<br>0b - No effect<br>1b - Locks |
| 29<br>DFMT | Domain Format<br>Indicates the domain assignment format.<br>0b - Core bus initiator domain assignment (DFMT0) |
| 28 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 27-24<br>— | Reserved |
| 23-22<br>— | Reserved |
| 21-16<br>PID | Process Identifier<br>Specifies the PID to be combined with the PIDM field and included in the domain assignment process. This field applies only if PID is enabled (PE = 1). |
| 15-14<br>— | Reserved |
| 13-8<br>PIDM | Process Identifier Mask<br>Specifies a mask applied to the PID to support including multiple PIDs in the domain hit determination. For each bit asserted in PIDM, the corresponding bit of the PID is ignored in the comparison. This field applies only if PID is enabled (PE = 1). |
| 7-6<br>PE | Process Identifier Enable<br>Enables the optional inclusion of PID, qualified by PIDM, in the domain hit evaluation. This inclusion supports the definition of inclusive or exclusive sets of masked PID values.<br>00b-01b - No PID is included<br>10b - Partial domain hit = (PID & ~PIDM) == (PIDm[PID] & ~PIDM)<br>11b - Partial domain hit = ~((PID & ~PIDM) == (PIDm[PID] & ~PIDM)) |
| 5-4<br>DIDS | DID Select<br>Selects the source of the DID.<br>00b - Use the DID field of this register<br>01b - Use the input DID<br>10b - Concatenate bits 3–2 of this register with the least significant 2 bits of the input DID (DID_in[1:0])<br>11b - Reserved |
| 3-2<br>— | Reserved |
| 1-0<br>DID | Domain Identifier<br>Specifies the DID. DIDS controls whether and how this value is used. |

### 18.8.3.14 Master Domain Assignment (MDA_W0_1_DFMT1 - MDA_W0_5_DFMT1)

#### Offset

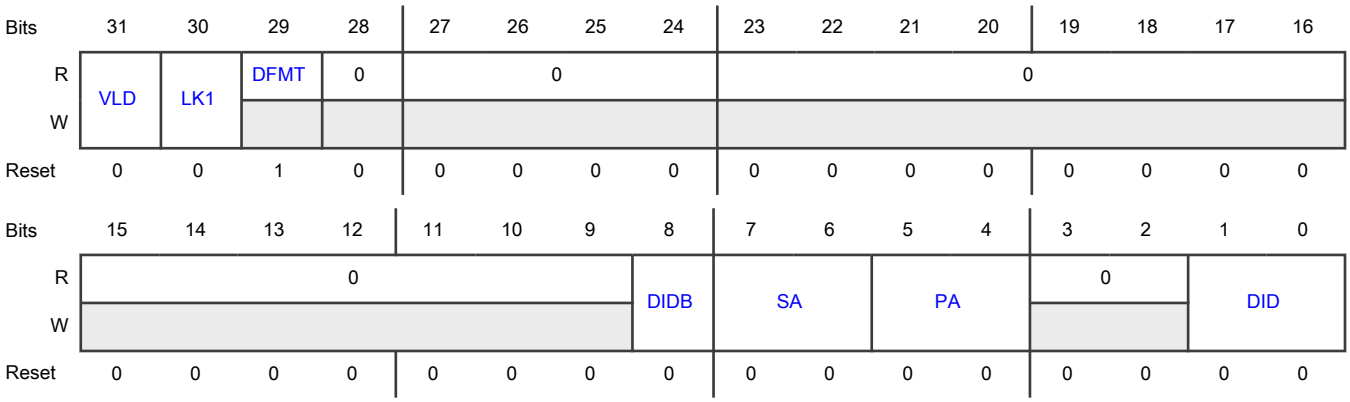
| Register | Offset |
|---|---|
| MDA_W0_1_DFMT1 | 820h |
| MDA_W0_2_DFMT1 | 840h |
| MDA_W0_5_DFMT1 | 8A0h |

#### Function

Specifies the information used by the MDAC to assign a bus initiator to a specific domain (DID). For more information, see Master domain assignment controller (MDAC).

Access: Secure privileged read/write

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31<br>VLD | Valid<br>Specifies whether this domain assignment is valid. In other words, if VLD and CR[GVLD] are both asserted, XRDC uses the configuration in this register in the domain assignment process. If CR[GVLD] is set to 1 and VLD is set to 0 then every transaction from this initiator will be assigned a DID of 0.<br>This field has no effect unless XRDC is enabled (CR[GVLD] = 1).<br>0b - Invalid<br>1b - Valid |
| 30<br>LK1 | Lock<br>Prohibits writes to this register.<br>• If unlocked, this register accepts any secure privileged write. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • If locked, you cannot write to this register and it remains read-only until after the next reset.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Unlocked<br>1b - Locks<br>When writing<br>0b - No effect<br>1b - Locks |
| 29<br>DFMT | Domain Format<br>Indicates the domain assignment format.<br>1b - Bus initiator domain assignment (DFMT1) |
| 28<br>— | Reserved |
| 27-24<br>— | Reserved |
| 23-9<br>— | Reserved |
| 8<br>DIDB | DID Bypass<br>Enables bypassing of an input DID as the domain identifier for this initiator. This capability allows noncore initiators (for example, a DMA) to appear as a core.<br>After this field is set to 1, it remains at that value until the next reset.<br>0b - Bypass DID input. Use DID<br>1b - Use DID input |
| 7-6<br>SA | Secure Attribute<br>Specifies the secure attribute.<br>**NOTE**<br>If SA = 1X or VLD = 0, use the secure attribute input from the initiator.<br>00b - Force to secure<br>01b - Force to nonsecure<br>1xb - Use secure attribute from the initiator |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 5-4<br>PA | Privileged Attribute<br>Specifies the privileged (supervisor/user) attribute.<br>**NOTE**<br>If PA = 1X or VLD = 0, use the privileged attribute input from the initiator.<br>00b - Force to user<br>01b - Force to privileged<br>1xb - Use privileged attribute from the initiator |
| 3-2<br>— | Reserved |
| 1-0<br>DID | Domain Identifier<br>Specifies the DID. |

### 18.8.3.15 Peripheral Domain Access Control Word 0 (PDAC_W0_32 - PDAC_W0_315)

**Offset**

| Register | Offset |
|---|---|
| PDAC_W0_32 | 1100h |
| PDAC_W0_33 | 1108h |
| PDAC_W0_34 | 1110h |
| PDAC_W0_35 | 1118h |
| PDAC_W0_36 | 1120h |
| PDAC_W0_38 | 1130h |
| PDAC_W0_39 | 1138h |
| PDAC_W0_40 | 1140h |
| PDAC_W0_41 | 1148h |
| PDAC_W0_42 | 1150h |
| PDAC_W0_44 | 1160h |
| PDAC_W0_45 | 1168h |
| PDAC_W0_46 | 1170h |
| PDAC_W0_47 | 1178h |
| PDAC_W0_128 | 1400h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| PDAC_W0_129 | 1408h |
| PDAC_W0_130 | 1410h |
| PDAC_W0_131 | 1418h |
| PDAC_W0_132 | 1420h |
| PDAC_W0_133 | 1428h |
| PDAC_W0_134 | 1430h |
| PDAC_W0_135 | 1438h |
| PDAC_W0_136 | 1440h |
| PDAC_W0_137 | 1448h |
| PDAC_W0_138 | 1450h |
| PDAC_W0_139 | 1458h |
| PDAC_W0_140 | 1460h |
| PDAC_W0_141 | 1468h |
| PDAC_W0_142 | 1470h |
| PDAC_W0_143 | 1478h |
| PDAC_W0_144 | 1480h |
| PDAC_W0_145 | 1488h |
| PDAC_W0_146 | 1490h |
| PDAC_W0_147 | 1498h |
| PDAC_W0_148 | 14A0h |
| PDAC_W0_149 | 14A8h |
| PDAC_W0_150 | 14B0h |
| PDAC_W0_151 | 14B8h |
| PDAC_W0_152 | 14C0h |
| PDAC_W0_153 | 14C8h |
| PDAC_W0_154 | 14D0h |
| PDAC_W0_155 | 14D8h |
| PDAC_W0_156 | 14E0h |
| PDAC_W0_157 | 14E8h |
| PDAC_W0_158 | 14F0h |
| PDAC_W0_159 | 14F8h |
| PDAC_W0_160 | 1500h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| PDAC_W0_161 | 1508h |
| PDAC_W0_162 | 1510h |
| PDAC_W0_163 | 1518h |
| PDAC_W0_164 | 1520h |
| PDAC_W0_165 | 1528h |
| PDAC_W0_166 | 1530h |
| PDAC_W0_167 | 1538h |
| PDAC_W0_168 | 1540h |
| PDAC_W0_169 | 1548h |
| PDAC_W0_170 | 1550h |
| PDAC_W0_171 | 1558h |
| PDAC_W0_173 | 1568h |
| PDAC_W0_175 | 1578h |
| PDAC_W0_177 | 1588h |
| PDAC_W0_178 | 1590h |
| PDAC_W0_179 | 1598h |
| PDAC_W0_180 | 15A0h |
| PDAC_W0_181 | 15A8h |
| PDAC_W0_182 | 15B0h |
| PDAC_W0_183 | 15B8h |
| PDAC_W0_184 | 15C0h |
| PDAC_W0_186 | 15D0h |
| PDAC_W0_187 | 15D8h |
| PDAC_W0_188 | 15E0h |
| PDAC_W0_191 | 15F8h |
| PDAC_W0_193 | 1608h |
| PDAC_W0_194 | 1610h |
| PDAC_W0_195 | 1618h |
| PDAC_W0_196 | 1620h |
| PDAC_W0_197 | 1628h |
| PDAC_W0_198 | 1630h |
| PDAC_W0_201 | 1648h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| PDAC_W0_202 | 1650h |
| PDAC_W0_203 | 1658h |
| PDAC_W0_204 | 1660h |
| PDAC_W0_205 | 1668h |
| PDAC_W0_206 | 1670h |
| PDAC_W0_207 | 1678h |
| PDAC_W0_208 | 1680h |
| PDAC_W0_209 | 1688h |
| PDAC_W0_212 | 16A0h |
| PDAC_W0_213 | 16A8h |
| PDAC_W0_214 | 16B0h |
| PDAC_W0_215 | 16B8h |
| PDAC_W0_216 | 16C0h |
| PDAC_W0_217 | 16C8h |
| PDAC_W0_219 | 16D8h |
| PDAC_W0_220 | 16E0h |
| PDAC_W0_221 | 16E8h |
| PDAC_W0_223 | 16F8h |
| PDAC_W0_224 | 1700h |
| PDAC_W0_225 | 1708h |
| PDAC_W0_226 | 1710h |
| PDAC_W0_227 | 1718h |
| PDAC_W0_229 | 1728h |
| PDAC_W0_231 | 1738h |
| PDAC_W0_232 | 1740h |
| PDAC_W0_236 | 1760h |
| PDAC_W0_256 | 1800h |
| PDAC_W0_257 | 1808h |
| PDAC_W0_260 | 1820h |
| PDAC_W0_261 | 1828h |
| PDAC_W0_262 | 1830h |
| PDAC_W0_263 | 1838h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| PDAC_W0_264 | 1840h |
| PDAC_W0_265 | 1848h |
| PDAC_W0_266 | 1850h |
| PDAC_W0_267 | 1858h |
| PDAC_W0_268 | 1860h |
| PDAC_W0_269 | 1868h |
| PDAC_W0_270 | 1870h |
| PDAC_W0_271 | 1878h |
| PDAC_W0_272 | 1880h |
| PDAC_W0_273 | 1888h |
| PDAC_W0_274 | 1890h |
| PDAC_W0_275 | 1898h |
| PDAC_W0_276 | 18A0h |
| PDAC_W0_277 | 18A8h |
| PDAC_W0_278 | 18B0h |
| PDAC_W0_279 | 18B8h |
| PDAC_W0_280 | 18C0h |
| PDAC_W0_281 | 18C8h |
| PDAC_W0_283 | 18D8h |
| PDAC_W0_285 | 18E8h |
| PDAC_W0_288 | 1900h |
| PDAC_W0_291 | 1918h |
| PDAC_W0_292 | 1920h |
| PDAC_W0_293 | 1928h |
| PDAC_W0_294 | 1930h |
| PDAC_W0_295 | 1938h |
| PDAC_W0_296 | 1940h |
| PDAC_W0_297 | 1948h |
| PDAC_W0_298 | 1950h |
| PDAC_W0_303 | 1978h |
| PDAC_W0_304 | 1980h |
| PDAC_W0_307 | 1998h |

*Table continues on the next page...*

*Table continued from the previous page...*

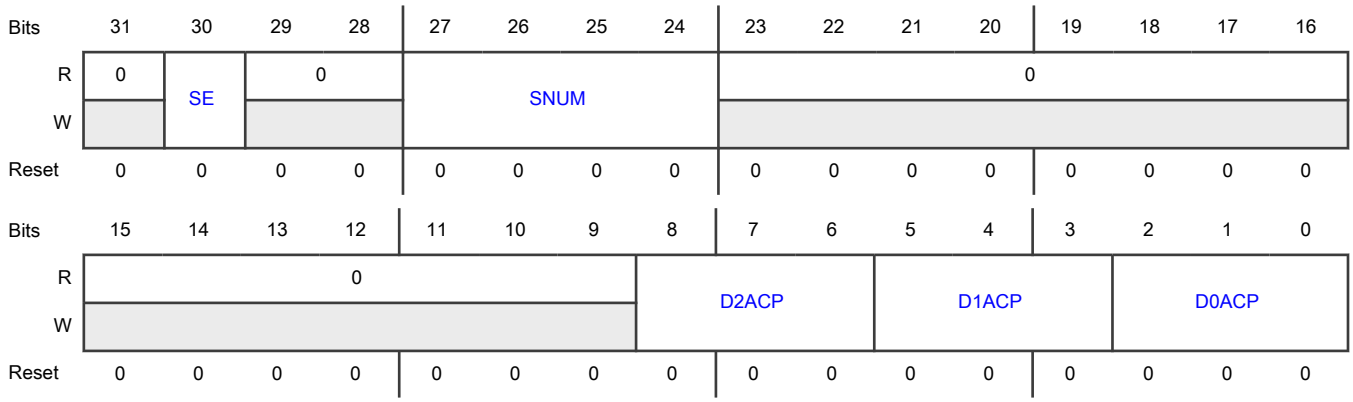
| Register | Offset |
|---|---|
| PDAC_W0_311 | 19B8h |
| PDAC_W0_314 | 19D0h |
| PDAC_W0_315 | 19D8h |

### Function

In conjunction with PDAC_W1_*s*, specifies the ACP configuration for peripheral slot *s*. The ACP controls access to the peripheral by all initiators within the domain. For the available ACPs, see Domain ACP specification. For more information, see Peripheral access controller (PAC).

Access: Secure privileged read/write

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>SE | Semaphore Enable<br>Enables the inclusion of the semaphore specified in SNUM in the D*d*ACP evaluation.<br>0b - Disables<br>1b - Enables |
| 29-28<br>— | Reserved |
| 27-24<br>SNUM | Semaphore Number<br>Specifies the hardware semaphore to include in the D*d*ACP access evaluation. This field applies only if you enable semaphore (write 1 to SE). |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 23-9<br>— | Reserved |
| 8-6: D2ACP<br>5-3: D1ACP<br>2-0: D0ACP | Domain Access Control Policy<br>Specifies the ACP for the associated domain. This field applies only for a supported DID; if the DID is not implemented, the field is read-only zero (no access rights). For field values, see Domain ACP specification. |

### 18.8.3.16 Peripheral Domain Access Control Word 1 (PDAC_W1_32 - PDAC_W1_315)

**Offset**

| Register | Offset |
|---|---|
| PDAC_W1_32 | 1104h |
| PDAC_W1_33 | 110Ch |
| PDAC_W1_34 | 1114h |
| PDAC_W1_35 | 111Ch |
| PDAC_W1_36 | 1124h |
| PDAC_W1_38 | 1134h |
| PDAC_W1_39 | 113Ch |
| PDAC_W1_40 | 1144h |
| PDAC_W1_41 | 114Ch |
| PDAC_W1_42 | 1154h |
| PDAC_W1_44 | 1164h |
| PDAC_W1_45 | 116Ch |
| PDAC_W1_46 | 1174h |
| PDAC_W1_47 | 117Ch |
| PDAC_W1_128 | 1404h |
| PDAC_W1_129 | 140Ch |
| PDAC_W1_130 | 1414h |
| PDAC_W1_131 | 141Ch |
| PDAC_W1_132 | 1424h |
| PDAC_W1_133 | 142Ch |
| PDAC_W1_134 | 1434h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| PDAC_W1_135 | 143Ch |
| PDAC_W1_136 | 1444h |
| PDAC_W1_137 | 144Ch |
| PDAC_W1_138 | 1454h |
| PDAC_W1_139 | 145Ch |
| PDAC_W1_140 | 1464h |
| PDAC_W1_141 | 146Ch |
| PDAC_W1_142 | 1474h |
| PDAC_W1_143 | 147Ch |
| PDAC_W1_144 | 1484h |
| PDAC_W1_145 | 148Ch |
| PDAC_W1_146 | 1494h |
| PDAC_W1_147 | 149Ch |
| PDAC_W1_148 | 14A4h |
| PDAC_W1_149 | 14ACh |
| PDAC_W1_150 | 14B4h |
| PDAC_W1_151 | 14BCh |
| PDAC_W1_152 | 14C4h |
| PDAC_W1_153 | 14CCh |
| PDAC_W1_154 | 14D4h |
| PDAC_W1_155 | 14DCh |
| PDAC_W1_156 | 14E4h |
| PDAC_W1_157 | 14ECh |
| PDAC_W1_158 | 14F4h |
| PDAC_W1_159 | 14FCh |
| PDAC_W1_160 | 1504h |
| PDAC_W1_161 | 150Ch |
| PDAC_W1_162 | 1514h |
| PDAC_W1_163 | 151Ch |
| PDAC_W1_164 | 1524h |
| PDAC_W1_165 | 152Ch |
| PDAC_W1_166 | 1534h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| PDAC_W1_167 | 153Ch |
| PDAC_W1_168 | 1544h |
| PDAC_W1_169 | 154Ch |
| PDAC_W1_170 | 1554h |
| PDAC_W1_171 | 155Ch |
| PDAC_W1_173 | 156Ch |
| PDAC_W1_175 | 157Ch |
| PDAC_W1_177 | 158Ch |
| PDAC_W1_178 | 1594h |
| PDAC_W1_179 | 159Ch |
| PDAC_W1_180 | 15A4h |
| PDAC_W1_181 | 15ACh |
| PDAC_W1_182 | 15B4h |
| PDAC_W1_183 | 15BCh |
| PDAC_W1_184 | 15C4h |
| PDAC_W1_186 | 15D4h |
| PDAC_W1_187 | 15DCh |
| PDAC_W1_188 | 15E4h |
| PDAC_W1_191 | 15FCh |
| PDAC_W1_193 | 160Ch |
| PDAC_W1_194 | 1614h |
| PDAC_W1_195 | 161Ch |
| PDAC_W1_196 | 1624h |
| PDAC_W1_197 | 162Ch |
| PDAC_W1_198 | 1634h |
| PDAC_W1_201 | 164Ch |
| PDAC_W1_202 | 1654h |
| PDAC_W1_203 | 165Ch |
| PDAC_W1_204 | 1664h |
| PDAC_W1_205 | 166Ch |
| PDAC_W1_206 | 1674h |
| PDAC_W1_207 | 167Ch |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| PDAC_W1_208 | 1684h |
| PDAC_W1_209 | 168Ch |
| PDAC_W1_212 | 16A4h |
| PDAC_W1_213 | 16ACh |
| PDAC_W1_214 | 16B4h |
| PDAC_W1_215 | 16BCh |
| PDAC_W1_216 | 16C4h |
| PDAC_W1_217 | 16CCh |
| PDAC_W1_219 | 16DCh |
| PDAC_W1_220 | 16E4h |
| PDAC_W1_221 | 16ECh |
| PDAC_W1_223 | 16FCh |
| PDAC_W1_224 | 1704h |
| PDAC_W1_225 | 170Ch |
| PDAC_W1_226 | 1714h |
| PDAC_W1_227 | 171Ch |
| PDAC_W1_229 | 172Ch |
| PDAC_W1_231 | 173Ch |
| PDAC_W1_232 | 1744h |
| PDAC_W1_236 | 1764h |
| PDAC_W1_256 | 1804h |
| PDAC_W1_257 | 180Ch |
| PDAC_W1_260 | 1824h |
| PDAC_W1_261 | 182Ch |
| PDAC_W1_262 | 1834h |
| PDAC_W1_263 | 183Ch |
| PDAC_W1_264 | 1844h |
| PDAC_W1_265 | 184Ch |
| PDAC_W1_266 | 1854h |
| PDAC_W1_267 | 185Ch |
| PDAC_W1_268 | 1864h |
| PDAC_W1_269 | 186Ch |

*Table continues on the next page...*

*Table continued from the previous page...*

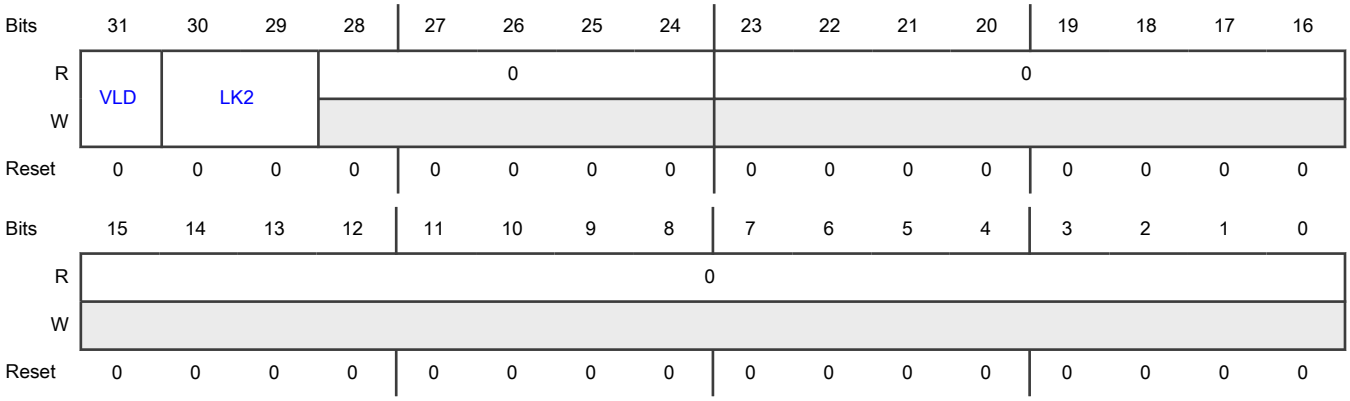
| Register | Offset |
|---|---|
| PDAC_W1_270 | 1874h |
| PDAC_W1_271 | 187Ch |
| PDAC_W1_272 | 1884h |
| PDAC_W1_273 | 188Ch |
| PDAC_W1_274 | 1894h |
| PDAC_W1_275 | 189Ch |
| PDAC_W1_276 | 18A4h |
| PDAC_W1_277 | 18ACh |
| PDAC_W1_278 | 18B4h |
| PDAC_W1_279 | 18BCh |
| PDAC_W1_280 | 18C4h |
| PDAC_W1_281 | 18CCh |
| PDAC_W1_283 | 18DCh |
| PDAC_W1_285 | 18ECh |
| PDAC_W1_288 | 1904h |
| PDAC_W1_291 | 191Ch |
| PDAC_W1_292 | 1924h |
| PDAC_W1_293 | 192Ch |
| PDAC_W1_294 | 1934h |
| PDAC_W1_295 | 193Ch |
| PDAC_W1_296 | 1944h |
| PDAC_W1_297 | 194Ch |
| PDAC_W1_298 | 1954h |
| PDAC_W1_303 | 197Ch |
| PDAC_W1_304 | 1984h |
| PDAC_W1_307 | 199Ch |
| PDAC_W1_311 | 19BCh |
| PDAC_W1_314 | 19D4h |
| PDAC_W1_315 | 19DCh |

### Function

In conjunction with PDAC_W0_*s*, specifies the ACP configuration for peripheral slot *s*.

Access: Secure privileged read/write

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>VLD | Valid<br>Specifies whether this domain assignment is valid. In other words, if VLD and CR[GVLD] are both asserted, XRDC uses the configuration in this pair of registers. If either CR[GVLD] or this field is 0, all accesses to the peripheral are allowed. To support a coherent register state, any write to PDAC_W0_*s* forces this field to zero.<br>This field has no effect unless XRDC is enabled (CR[GVLD] = 1).<br>0b - Invalid<br>1b - Valid |
| 30-29<br>LK2 | Lock<br>Limits or prohibits writes to the set of PDAC words (PDAC_W0_*s* and PDAC_W1_*s*) for this peripheral slot.<br>When you assert a bit in this field, it remains asserted until the next module reset.<br>00b-01b - Both words can be written to<br>10b - Domain d can update only its associated DdACP field—all other fields are read-only<br>11b - Locks (both words are read-only) |
| 28-24<br>— | Reserved |
| 23-0<br>— | Reserved |

## 18.8.3.17 Memory Region Descriptor Word 0 (MRGD_W0_0 - MRGD_W0_35)

### Offset

Registers in this array exist only for the following combinations of index values.

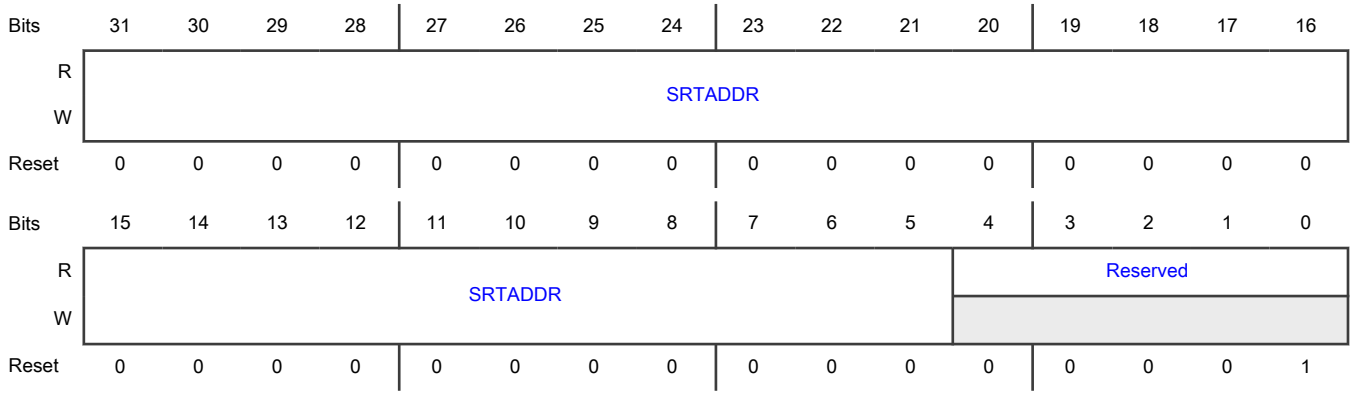
| Index *n* | Index *m* |
|---|---|
| 0–1 | 0–15 |
| 2 | 0–3 |

| Register | Offset |
|---|---|
| MRGD_W0_(n * 16 + m) | 2000h + (n × 200h) + (m × 20h) |

### Function

Specifies the starting address of memory region *r*.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-5<br>SRTADDR | Start Address<br>Specifies the most significant bits of the 0-modulo 32-byte start address of the memory region. The minimum region size is 32 bytes. |
| 4-0<br>— | Reserved |

## 18.8.3.18 Memory Region Descriptor Word 1 (MRGD_W1_0 - MRGD_W1_35)

### Offset

Registers in this array exist only for the following combinations of index values.

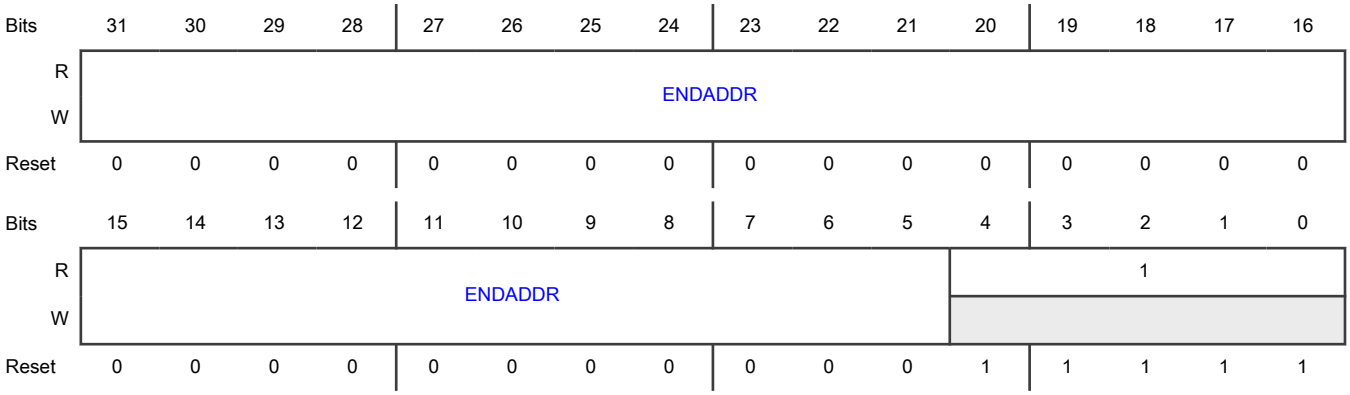
| Index *n* | Index *m* |
|---|---|
| 0–1 | 0–15 |
| 2 | 0–3 |

| Register | Offset |
|---|---|
| MRGD_W1_(n * 16 + m) | 2004h + (n × 200h) + (m × 20h) |

### Function

Specifies the ending address of memory region *r*.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-5<br>ENDADDR | End Address<br>Specifies the most significant bits of the 31-modulo 32-byte end address of memory region *r*. |
| 4-0<br>— | Reserved |

## 18.8.3.19 Memory Region Descriptor Word 2 (MRGD_W2_0 - MRGD_W2_35)

### Offset

Registers in this array exist only for the following combinations of index values.

| Index *n* | Index *m* |
|---|---|
| 0–1 | 0–15 |
| 2 | 0–3 |

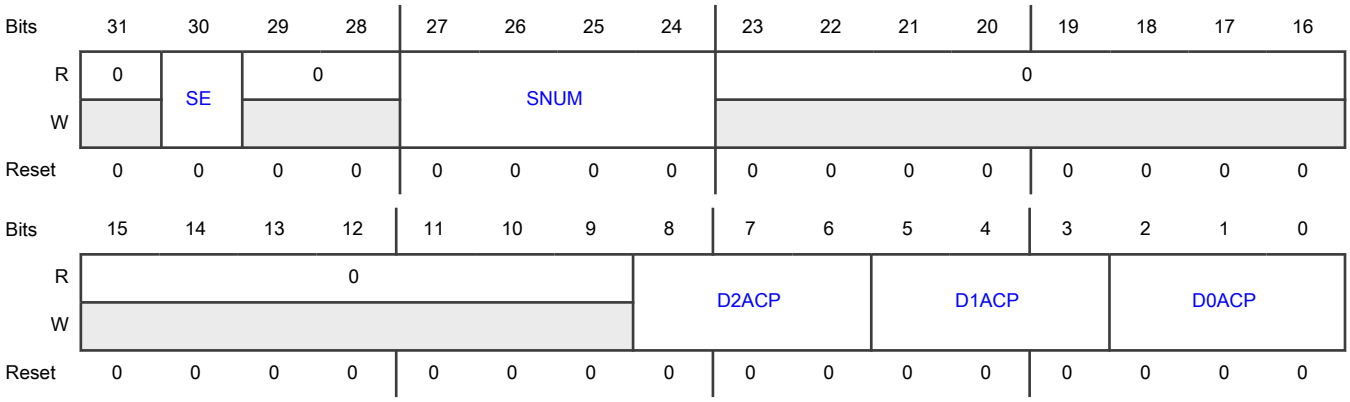
| Register | Offset |
|---|---|
| MRGD_W2_(n * 16 + m) | 2008h + (n × 200h) + (m × 20h) |

### Function

Specifies the ACP for the associated domain. The encodings specify read and write access capabilities based on the four operating states. This field applies only for a supported DID; if the DID is not implemented, the field is read-only zero (no access rights).

For field values, see Domain ACP specification.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>SE | Semaphore Enable<br>Enables the inclusion of the semaphore specified in SNUM in the D*d*ACP evaluation.<br>0b - Disables<br>1b - Enables |
| 29-28<br>— | Reserved |
| 27-24<br>SNUM | Semaphore Number<br>Specifies the hardware semaphore to include in the D*d*ACP access evaluation. This field applies only if you enable semaphore (write 1 to SE). |
| 23-9<br>— | Reserved |
| 8-6: D2ACP<br>5-3: D1ACP<br>2-0: D0ACP | Domain Access Control Policy<br>Specifies the ACP for the associated domain. This field applies only for a supported DID; if the DID is not implemented, the field is read-only zero (no access rights). For field values, see Domain ACP specification. |

### 18.8.3.20 Memory Region Descriptor Word 3 (MRGD_W3_0 - MRGD_W3_35)

#### Offset

Registers in this array exist only for the following combinations of index values.

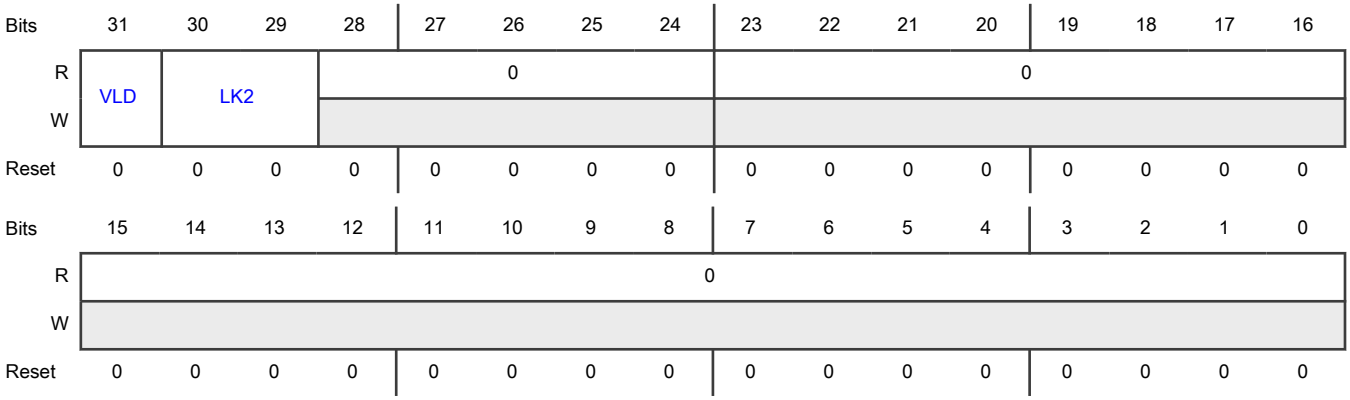
| Index *n* | Index *m* |
|---|---|
| 0–1 | 0–15 |
| 2 | 0–3 |

| Register | Offset |
|---|---|
| MRGD_W3_(n * 16 + m) | 200Ch + (n × 200h) + (m × 20h) |

#### Function

Specifies whether this memory region descriptor is enabled and limits or prohibits writes to it.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31<br>VLD | Valid<br>Specifies whether this domain assignment is valid. In other words, if VLD and CR[GVLD] are both asserted, XRDC uses the configuration in this set of registers. If CR[GVLD] is 0, all accesses to the memory are allowed. If CR[GVLD] is 1 and VLD is 0, all accesses are blocked. To support a coherent register state, a write to any of the MRGD W0–W2 registers forces this field to zero.<br>This field has no effect unless XRDC is enabled (CR[GVLD] = 1).<br>0b - Invalid<br>1b - Valid |
| 30-29 | Lock<br>Limits or prohibits writes to the set of MRGD words (MRGD_W*w_n*) for this memory region. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| LK2 | When you assert a bit in this field, it remains asserted until the next module reset.<br>00b - All words in the set can be written to<br>01b - Reserved<br>10b - Domain d can update only its associated DdACP field; all other fields are read-only<br>11b - Locks (all words are read-only) |
| 28-24<br>— | Reserved |
| 23-0<br>— | Reserved |

## 18.9 Glossary

**ACP** access control policy. The access limitations specified for memory and peripheral resources.

**DID** domain identifier. A numeric value that identifies a specific domain.

**domain** An access-controlled virtual group of on-chip initiators (cores and noncore initiators) and targets (memories and peripherals) that comprise an isolated computing environment. All initiators in a domain have the same access to chip resources within that domain.

See Introduction to domains for more information.

**initiator** A processor core or non-processor module, such as DMA or a communications channel, that can initiate transactions with memory and peripheral resources. This term replaces "master".

**LPID** logical partition identifier. Also called an operating system ID or VMID, the LPID identifies a virtual initiator (either core or noncore) that runs on a hypervisor.

**MDAC** Master Domain Assignment Controller. Manages resource assignments and DIDs.

**MGR** Manager. Manages accesses through the XRDC programming model.

**MRC** Memory Region Controller. Controls access to memories based on memory region descriptors.

**PAC** Peripheral Access Controller (also sometimes called PDAC). Controls access to peripherals.

**PDAC** See PAC.

**PID** process identifier. A numeric value provided by some core processors to identify the currently active process.

**SDAC** Deprecated. See MRC.

**target** A peripheral or memory resource that one or more initiators can access. This term replaces "slave".

**transaction** A read or write request made by an initiator to a target peripheral or memory.

**VMID** virtual machine identifier. See LPID.

# Chapter 19
# Memory and Memory Interfaces

## 19.1 Introduction

This chapter discusses the configuration of memories and memory interfaces, such as flash memory, flash memory controller, and SRAM.

## 19.2 Flash memory controller and flash memory modules

For information on this, see the following chapters:

- Flash Memory Controller (PFLASH)
- Embedded Flash Memory

## 19.3 Related information

**Table 84. Related information**

| Topic | Related chapters | For additional information |
|---|---|---|
| System memory map | Memory Map | See the memory map file attached to this document. |
| Clocking | • Clocking Overview<br>• Clock Generation Module (MC_CGM) | — |
| Arm Cortex-M7 core | Cortex-M7 Overview | |
| XRDC | Extended Resource Domain Controller | |
| Direct-memory access | • Direct Memory Access Multiplexer (DMAMUX)<br>• Enhanced Direct Memory Access (eDMA) | See the DMAMUX map file attached to this document. |
| EIM | Error Injection Module | — |
| ERM | Error Reporting Module | — |

## 19.4 SRAM access

In case a master accesses an SRAM with multi-bit ECC errors, the chip may respond as follows:

- Map all such faults to FCCU. The recommended reaction for the fault is to generate the functional reset.
- Map such faults to ERM. If the ERM interrupt is enabled, ERM generates an interrupt.

## 19.5 Memories

The following table provides information on the types of memories, with their associated configurations, available in MCXE31.

Table 85. Memory configuration

| Memory | Size | Configuration (words × bits) | ECC or parity | ECC or parity width | Diagnostic information or error report | Applicability |
|---|---|---|---|---|---|---|
| SRAM0 | 160 KB | 32 KB with STANDBY mode retention (4096 × (64 + 8)) | ECC (SECDED) | 8 | ERM | MCXE31B |
| | | 128 KB (16384 × (64 + 8)) | | | | |
| | 96 KB | 32 KB with STANDBY mode retention (4096 × (64 + 8)) | ECC (SECDED) | 8 | ERM | MCXE317 |
| | | 64 KB (8192 × (64 + 8)) | | | | |
| | 32 KB | 32 KB with STANDBY mode retention (4096 × (64 + 8)) | ECC (SECDED) | 8 | ERM | MCXE315/ MCXE316 |
| CM7_0 I-cache data | 8 KB | 4 KB (512 × (64 + 8)) | ECC (SECDED) | 8 | ERM | MCXE31B<br>MCXE317 |
| | | 4 KB (512 × (64 + 8)) | | | | |
| CM7_0 I-cache tag | 672 bytes | 336 bytes (128 × (21 + 7)) | ECC (SECDED) | 7 | ERM | MCXE315/ MCXE316 |
| | | 336 bytes (128 × (21 + 7)) | | | | |
| CM7_0 D-cache data | 8 KB | 1 KB (256 × (32 + 7)) | ECC (SECDED) | 7 | ERM | |
| | | 1 KB (256 × (32 + 7)) | | | | |
| | | 1 KB (256 × (32 + 7)) | | | | |
| | | 1 KB (256 × (32 + 7)) | | | | |
| | | 1 KB (256 × (32 + 7)) | | | | |
| | | 1 KB (256 × (32 + 7)) | | | | |
| | | 1 KB (256 × (32 + 7)) | | | | |
| | | 1 KB (256 × (32 + 7)) | | | | |
| CM7_0 D-cache tag | 800 bytes | 200 bytes (64 × (25 + 7)) | ECC (SECDED) | 7 | ERM | |
| | | 200 bytes (64 × (25 + 7)) | | | | |
| | | 200 bytes (64 × (25 + 7)) | | | | |
| | | 200 bytes (64 × (25 + 7)) | | | | |
| DMA TCD | 1 KB | 1 KB (128 × (64 + 8)) | ECC (SECDED) | 8 | ERM | MCXE31B |
| | 640 bytes | 640 bytes (80 × (64 + 8)) | ECC (SECDED) | 8 | ERM | MCXE317<br>MCXE315/ MCXE316 |
| FlexCAN_0 | 5 KB | 5 KB (640 × (64 + 40)) | ECC (SECDED) | 40 | FlexCAN_0 | MCXE31B<br>MCXE317 |

*Table continues on the next page...*

Table 85. Memory configuration (continued)

| Memory | Size | Configuration (words × bits) | ECC or parity | ECC or parity width | Diagnostic information or error report | Applicability |
|---|---|---|---|---|---|---|
| FlexCAN_1 | 1920 bytes | 1920 bytes (240 × (64 + 40)) | ECC (SECDED) | 40 | FlexCAN_1 | MCXE315/ MCXE316 |
| FlexCAN_2 | 1920 bytes | 1920 bytes (240 × (64 + 40)) | ECC (SECDED) | 40 | FlexCAN_2 | |
| FlexCAN_3 | 1152 bytes | 1152 bytes (144 × (64 + 40)) | ECC (SECDED) | 40 | FlexCAN_3 | MCXE31B<br>MCXE317 |
| FlexCAN_4 | 1152 bytes | 1152 bytes (144 × (64 + 40)) | ECC(SECDED) | 40 | FlexCAN_4 | MCXE31B<br>MCXE317 |
| FlexCAN_5 | 1152 bytes | 1152 bytes (144 × (64 + 40)) | ECC (SECDED) | 40 | FlexCAN_5 | |
| QuadSPI RAM | 1 KB | 1 KB (128×64) | No | 0 | Not applicable | MCXE31B |
| EMAC TX | 8960 bytes | 4480 bytes (1024 × (35 + 7)) | ECC (SECDED) | 7 | EMAC | |
| | | 4480 bytes (1024 × (35 + 7)) | | | | |
| EMAC RX | 8960 bytes | 4480 bytes (1024 × (35 + 7)) | ECC (SECDED) | 7 | EMAC | |
| | | 4480 bytes (1024 × (35 + 7)) | | | | |
| EMAC TSN | 1664 bytes | 1664 bytes (512 × (26 + 7)) | ECC (SECDED) | 7 | EMAC | |
| EMAC RXPARSER | 960 bytes | 960 bytes (80 × (96 + 8)) | ECC (SECDED) | 8 | EMAC | |
| QuadSPI TX | 320 bytes | 320 bytes (80×32) | No | 0 | Not applicable | |
| Cortex-M7 cluster ETF ETMI | 1 KB | 1 KB (128×64) | No | 0 | Not applicable | MCXE31B<br>MCXE317<br>MCXE315/ MCXE316 |
| Cortex-M7 cluster ETF ETMD | 2 KB | 2 KB (128×128) | No | 0 | Not applicable | |
| HTM ETF | 1 KB | 1 KB (128×64) | No | 0 | Not applicable | |
| Shared system ETF | 2 KB | 2 KB (256×64) | No | 0 | Not applicable | |
| HSE secure RAM | 64 KB | 64 KB (8192 × (64 + 8)) | ECC (SECDED) | 8 | HSE | |
| HSE PKC | 8 KB | 4 KB (512 × (64 + 8)) | ECC (SECDED) | 8 | HSE | |
| | | 4 KB (512 × (64 + 8)) | | | | |
| HSE MTB | 1 KB | 1 KB (256×32) | No | 0 | HSE | |

*Table continues on the next page...*

Table 85. Memory configuration (continued)

| Memory | Size | Configuration (words × bits) | ECC or parity | ECC or parity width | Diagnostic information or error report | Applicability |
|---|---|---|---|---|---|---|
| CM7_0 ITCM | 32 KB | 32 KB (4096 × (64 + 8)) | ECC (SECDED) | 8 | ERM | MCXE31B |
| CM7_0_DTCM | 64 KB | 32 KB (8192 × (32 + 8)) | ECC (SECDED) | 8 | ERM | MCXE317 |
| | | 32 KB (8192 × (32 + 8)) | | | | MCXE315/ MCXE316 |

## 19.6 Recommendations for ARM memories

As per ARM M-7 Safety manual, following considerations must be ensured for proper operation of ARM memories:

- ITCM and DTCM must be properly initialized with correct ECC before any read operation to avoid any code runaway or software malfucntion or core lockup.

**NOTE**

ITCM must be initialized with 64-bit writes whereas DTCM can be initialized with 32-bit writes also.

- To safely disable TCM:
  1. Clear ITCMCR[EN] or DTCMCR[EN] as required. See ARM Cortex-M7 Devices Generic User Guide for details on ITCMCR and DTCMCR register.
  2. Execute DSB instruction
  3. Execute ISB instruction

**NOTE**

Care must be taken if disabling the ITCM while executing from it. In this case, software must ensure that the switch-off code is stored in the L2 code memory region from where execution continues when the ITCM is disabled.

- To safely disable the I-cache:
  1. Clear CCR[IC]. See ARM Cortex-M7 Devices Generic User Guide for details on CCR register.
  2. Execute DSB instruction
  3. Execute ISB instruction
- To safely disable the D-cache:
  1. Clean and invalidate non-WT locations in D-cache
  2. Clear CCR[DC]
  3. Execute DSB instruction

See Table 86 for details on memory ECC initialization.

## 19.7 Memory ECC initialization summary

The table below summarizes memory ECC initialization.

Table 86. Memory ECC initialization summary

| Memory | Write access size | Masters |
|---|---|---|

*Table continues on the next page...*

Table 86. Memory ECC initialization summary (continued)

| | | CM7_ n | CM7_ m | eDMA |
|---|---|---|---|---|
| SRAM | 64-bits only | System | System | System |
| CM7_n ITCM | 64-bits only | Direct or Backdoor | Backdoor | Not possible |
| CM7_n DTCM | 32-bits or 64-bits | Direct or Backdoor | Backdoor | Backdoor |
| CM7_m ITCM | 64-bits only | Backdoor | Direct or Backdoor | Not possible |
| CM7_m DTCM | 32-bits or 64-bits | Backdoor | Direct or Backdoor | Backdoor |

## 19.8 Glossary

| | |
|---|---|
| **DTCM** | Data tightly coupled memory |
| **ECC** | Error code correction |
| **ETF** | Embedded trace FIFO |
| **ETMD** | Embedded trace macrocell-data |
| **ETMI** | Embedded trace macrocell-instruction |
| **ITCM** | Instruction tightly coupled memory |
| **MTB** | Macrocell trace buffer |
| **PKC** | Public key cryptography |
| **RXPARSER** | Receive parser memory |
| **SECDED** | Single error correction double error detection |
| **SRAM** | Static random access memory |
| **TSN** | Time sensitive network |

# Chapter 20
# Embedded Flash Memory (c40asf)

## 20.1 Chip-specific c40asf flash memory information

### 20.1.1 Flash memory configuration and register settings

Table 87 shows flash memory blocks and their associated configuration.

**Table 87. Flash block configuration**

| Block or address number | Block name | Start address (hex) | End address (hex) | Size | Applicability |
|---|---|---|---|---|---|
| 0 | Code flash memory 0 | 0040_0000 | 004F_FFFF | 1 MB | MCXE31B and MCXE317 |
| | | | | 512 KB | MCXE315 and MCXE316 |
| 1 | Code flash memory 1 | 0050_0000 | 005F_FFFF | 1 MB | MCXE31B and MCXE317 |
| | | | | 512 KB | MCXE315 and MCXE316 |
| 2 | Code flash memory 2 | 0060_0000 | 006F_FFFF | 1 MB | MCXE31B |
| 3 | Code flash memory 3 | 0070_0000 | 007F_FFFF | 1 MB | MCXE31B |
| 4 | Data flash memory | 1000_0000 | 1001_FFFF | 128 KB | MCXE31B |
| 2 | | | | 128 KB | MCXE317 |
| | | 1000_0000 | 1000_FFFF | 64 KB | MCXE315/MCXE316 |
| 0 [1] | UTest NVM | 1B00_0000 | 1B00_1FFF | 8 KB | MCXE31B, MCXE317, MCXE315 and MCXE316 |

1. The address region number is the same as block number for all the blocks except this one. Address region is called UTest NVM address region in this case.

No application cores except ELE_HSEB can access the address regions shown in Table 88.

**Table 88. Secure flash memory configuration**

| Description | MCXE315/MCXE316 | | MCXE317 | | MCXE31B | | Size |
|---|---|---|---|---|---|---|---|
| | Start address | End address | Start address | End address | Start address | End address | |
| Secure BAF code (SBAF_CODE) | 004F_4000h | 004F_FFFFh | 005F_4000h | 005F_FFFFh | 007F_4000h | 007F_FFFFh | 48 KB |

**NOTE**

For HSE memory configuration, see *'Memory map when HSE firmware usage feature flag is enabled'* section in *'Boot Overview'* chapter.

The flash memory can perform multiple reads in parallel (between different blocks), using the single-, dual-, or quad-read feature. It also includes an internal UTest mode that can generate single- and double-bit ECC errors.

The code flash memory lets you access three data paths simultaneously.

You can configure the flash memory sectors as erase- or write-protected by using the flash memory's sector and super-sector locking features.

Data flash and/or Code flash memory blocks can be altered by a code executing from different flash block or SRAM.

Data flash memory is the same as code flash memory—supporting program, erase, and read operations.

After the completion of the program or erase operations, the MCR and MCRS registers notify you as soon as the code flash memory code can be executed. At the end of reset recovery, MCRS[DONE] transitions from 0 to 1.

**NOTE**

MCRS[RWE] might become 1 during the program or erase operation because of speculative accesses by Cortex-M7 core(s). If there are memory regions where no speculative accesses must be initiated, Arm recommends to configure the on-core MPU to set those regions with these attributes:

- Device or strongly-ordered
- Execute never

**NOTE**

- ADR[A3] and ADR[A4] field are reserved for MCXE317, as code flash memory size in 2 MB for MCXE317.
- For MCXE317, reset value of FLASH_SSPELOCK and FLASH_XSSPELOCK register is 0x0FFF_FFFF.

### 20.1.2 MCRE register reset value

The reset value of the MCRE register depends on the chip-specific flash memory configuration. This table provides the flash memory configuration of the chip and the associated register reset values.

**Table 89. MCRE register reset value**

| MCRE fields | Reset value |
|---|---|
| n1M (bit position 23:21) | 100b – Four 1 MB blocks |
| | 010b – Two 1 MB blocks |
| n512K (bit position 15:14) | 00b – Zero 512 KB blocks |
| n256K (bit position 7:6) | 01b – One 256 KB blocks |
| MCRE register reset value | MCXE31B - 0080_0040h |
| | MCXE317 - 0040_0040h |

### 20.1.3 Debugger considerations while Flash Program/Erase

While flash programming is done via debugger, special considerations should be taken care off. See "Debug Subsystem" chapter for details.

## 20.2 Introduction

The primary function of the embedded flash memory serves as an electrically programmable and erasable non-volatile memory (NVM) that may be used for instruction or data storage.

### 20.2.1 Overview

The embedded flash memory is designed for use in embedded MCU and SoC applications. It supports a total memory size up to 8.5 MB of NVM in main space and 8 KB of UTest NVM space per module. Multiple flash modules may be instantiated within an SoC to achieve higher density. The embedded flash memory is addressable by page (256 bits) for read operation and double-word(s) and page and quad page for program operations. Flash memory reads always return 256 bits, although read page buffering may

be performed by the PFC. The flash memory is able to do multiple reads in parallel (between different blocks), utilizing the quad read feature.

For details on the embedded flash memory architecture and features, see Functional description.

The following figure shows the top-level diagram and functional organization of the flash memory unit.

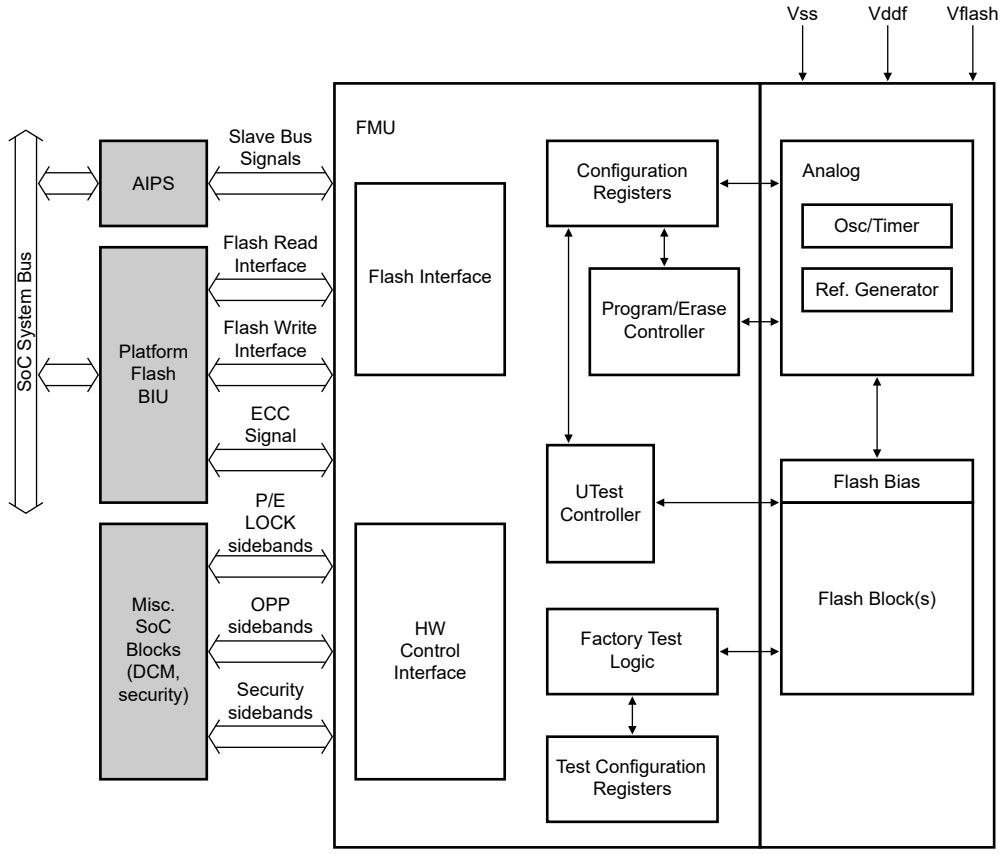



**Figure 45. Block diagram**

## 20.2.2 Features

The embedded flash memory includes these distinct features:

- Test information stored in a dedicated non-volatile sector (referred to as the UTest sector)
- OTP space made available in the UTest sector
- Read page size of 256 bits (8 words)
- ECC with single bit correction, double bit detection (all 1s valid) with 64-bit granularity
- Quad page programming (256-bit granularity)
- Hardware programmable sector and super sector program or erase restriction control
- Erasing of selected sector or block erase
- Independent programming of the UTest NVM sector
- Embedded hardware program and erase algorithms
- Support for reading while writing when the accesses are to different blocks
- UTest mode (user-accessible test modes) including array integrity and margin read

- Triple-voted flops for safety-critical flash memory functions (for example, internal trimming, redundancy, mode control, and others)

### 20.2.3 Modes of operation

Following is a brief description of the embedded flash memory operating modes.

- User mode is the default operating mode of the embedded flash memory. In this mode, it is possible to read and write registers (including register-based interlock writes), read the memory array, program the memory array, and erase the memory array. In this mode program and erase operations are initiated by doing register writes. Program and erase operations are controlled by an internal state machine.
- Low-Power mode turns off all DC current sources within the embedded flash memory and enables VFLASH and VDDF to be power gated. The embedded flash memory is not accessible for read, write, program, or erase when in Low-Power mode. This mode results in the lowest current draw that embedded flash memory can achieve.
- UTest mode is a tiered test mode strategy in which a portion of the factory test modes are made available. This mode is protected but accessible.

## 20.3 UTest NVM sector

The UTest NVM sector may be enabled by PEADR[PEASAD] during interlock writes. When the UTest NVM space is enabled, all program and erase operations are mapped to the UTest NVM sector. User-mode programming of the UTest NVM sector is enabled only when PEADR[PEASAD] is high.

The UTest NVM sector is an OTP sector (assuming Test mode disable seal is written) . Therefore, performing an erase operation is not permitted.

The UTest NVM sector supports RWW, and is grouped with the sectors in partition 0.

The UTest NVM sector may be locked against program by using hardware program and erase protection input to the flash module (MCRS[TSPELOCK]).

The UTest NVM sector contains specified data that is needed for embedded flash memory or SoC features.

Programming of the UTest NVM space has restrictions that are similar to those of the main space in terms of how ECC is calculated. Only one program operation is allowed per 64-bit ECC segment.

## 20.4 Test mode disable seal

The UTest NVM sector includes a mechanism to disable factory entry into Test mode selectable by block. Extreme care must be taken when using this feature, because blocks that are selected to be protected in this method will not have possible failures analyzed by factory failure analysts. Once sealed, the UTest sector becomes OTP (erase locked, and OPP enabled). The UTest sector is not accessible in Test Mode.

Protection of this sort prevents all high voltage operations to the flash memory executed by the internal state-machine, as well as reads through this state machine, and reads through the array integrity state machine when using Test mode interfaces.

UTest operations, margin read and array integrity, are also protected by preventing MISR updates on blocks selected for protection, although reads are still performed, and single bit corrections and double bit detections will be logged. Protection through other interfaces is provided by normal User mode protection mechanisms and is device-specific.

The method to disable factory entry into Test mode is to first program the Test mode disable seal location to be 5A4B_3C2Dh. After the next reset is asserted, Test mode is disabled.

It is possible to create a password to enable factory entry into Test mode. This can be programmed into the Test mode disable override passcode. The passcode may not be 0000_0000h, FFFF_FFFFh, or 5555_5555h. These are all invalid passcodes and will not be accepted to override. If it is desired, by the customer, that override never be possible, one of the three invalid passcodes must be put into the Test mode disable override passcode location. Passcode may be entered to authenticate entry, if enabled, by performing a 32-bit register write to register address 90h.

Even if the Test mode disable seal password is written, the UTest NVM sector is always protected and the UTest operations, margin read and array integrity, are always protected.

Only blocks selected in the Test mode disable block select field are controlled by the Test mode disable feature. Therefore, it is possible for customers to selectively pick blocks that have this type of protection and will not be eligible for factory failure analysis. Bits programmed to 0 in the Test mode disable block select field(s) designate blocks that are controlled by the Test mode disable feature.

In order to let two opportunities to select blocks for the Test mode disable, two regions are available, Test mode disable bock select - 1 and Test mode disable block select - 2. If either Test mode disable block select - 1 or Test mode disable block select - 2 has a 0 programmed, that block will be protected. The bits of these two regions are logically ANDed to define the blocks that are effectively selected for the Test mode disable feature. The bock select field is organized as shown in the next table:

**Table 90. Test mode disable block select**

| Block | Bits used to select blocks |
|---|---|
| Block 0 disable | Data[0] |
| Block 1 disable | Data[8] |
| Block 2 disable | Data[16] |
| Block 3 disable | Data[24] |
| Block 4 disable | Data[32] |
| Block 5 disable | Data[40] |

## 20.5 Functional description

The embedded flash memory module consists of blocks with the following options:

- 2 MB blocks (0, 1, 2, 3 or 4 blocks allowed per module)
- 1 MB blocks (0, 1, 2, 3 or 4 blocks allowed per module)
- 512 KB blocks (0, 1, 2 or 4 blocks allowed per module)
- 256 KB blocks (0, 1, 2 or 4 blocks allowed per module)
- A total of six blocks are allowed per module

Read while Write is allowed within a module, and is determined based on block boundaries. Read while Write partitions are used to determine locations for valid read-while-write (RWW) operations. While the embedded flash memory performs a write (program or erase) to a given partition, it can simultaneously perform a read from any other partition. For program and erase operations, only the address specified by an interlock write determines the partition being written (sector or super sector locking does not determine the RWW partitions being written).

The main address space is also divided into sectors and super sectors to implement independent erase and program protection. The UTest NVM space also exists as a sector and has independent program protection. The UTest NVM sector is included to support systems that require non-volatile memory (NVM) for security or to store system initialization information.

A number of MCR bits are protected against write when another field, or set of fields, is in a specific state. These write locks are covered on a bit-by-bit basis in Module Configuration Register. The write locks do not consider the effects of trying to write two or more bits simultaneously. This module does not allow fields to be written simultaneously that put the device into an illegal state. This is implemented through a priority mechanism among the fields, and the next table shows this mechanism.

Table 91. MCR field set and clear priority levels

| Priority level | MCR field(s) |
|---|---|
| 1 | ERS |
| 2 | PGM |
| 3 | EHV |

If two or more MCR fields are written simultaneously, only the field with the lowest priority number is accepted for modification. For example, writing 1 to ERS and PGM simultaneously results in 1 being written only to ERS.

Each read of the embedded flash memory retrieves a page, or eight consecutive words (256 bits), of information. The address for each word retrieved within a page differs from the other addresses in the page only by address bits [4:2]. The flash memory page read architecture easily supports both cache and burst mode at the PFC BIU level for high-speed read applications.

The embedded flash memory supports fault tolerance through error correction code (ECC) and error detection. ECC implemented within the embedded flash memory corrects single bit failures and detects double bit failures.

Program and erase of the embedded flash memory requires multiple system clock cycles to complete. Program and erase may be aborted.

The embedded flash memory may operate in various modes as described in the following sections.

## 20.5.1 User mode

In User mode, the embedded flash memory may be read and written (register writes and interlock writes), programmed or erased. The following sub-sections define all actions that may be performed in User mode.

### 20.5.1.1 Read and write

The default state of the embedded flash memory is read. The main and UTest NVM address spaces can be read only in the read state. The flash registers are always available for read, except when the embedded flash memory is in Low-Power mode. The module enters the read state on reset, and remains in the read state under two sets of conditions:

- The read state is active when the embedded flash memory is enabled (User mode read).
- The read state is active when MCR[PGM] or MCR[ERS] are 1 and a high voltage operation is ongoing (RWW).

**NOTE**

Reads done to the partition(s) being operated on (either erased or programmed) result in MCRS[RWE] becoming 1.

In embedded flash memory, flash core reads return 256 bits (1 page). Register reads return 32 bits of data.

**NOTE**

Flash core reads are performed through the PFC BIU. In many cases, the BIU does read page buffering to allow sequential reads to be done with higher performance. This could provide a data coherency issue that must be handled with software. Data coherency may be an issue after a program or erase operation, as well as UTest NVM sector operations.

In user mode, registers may be written (including interlock writes using registers).

Register reads to unmapped register address space return all 0s.

Register writes to unmapped register address space have no effect.

Interlock writes that are attempted during a high voltage operation (MCR[EHV] = 1 or MCRS[DONE] =0) result in the interlock write being ignored, and address and data will not be updated.

#### 20.5.1.2 Program

A flash memory program sequence operates on any page within the flash core. Within a page, up to eight words may be altered in a single program operation. Also, up to four pages can be altered in a single program operation. Whenever the array is programmed, the ECC bits also get programmed.

ECC is handled on a 64-bit boundary. Thus, if only one word in any given 64-bit ECC segment is programmed, the adjoining word in that segment should not be programmed, because ECC calculation is already complete for that 64-bit segment. Attempts to program the adjoining word may likely result in an operation failure. It is recommended that all programming operations ranger from 64 bits to 1024 bits, and be 64-bit aligned. The programming operation should completely fill selected ECC segments within the page. Only one program is allowed per 64-bit ECC segment between erases.

**Caution**

In rare cases, over-programming of a 64-bit ECC segment may be done (EEPROM emulation in data flash region). In this case, approved EEPROM emulation drivers must be used, and they must limit the number of over-program operations to three times (four total programs between erases).

Programming changes the value stored in an array bit from logic 1 to logic 0 only. Programming cannot change a stored logic 0 to a logic 1.

**NOTE**

If a logic 0 is attempted to be over-programmed by a logic 1, the resulting operation fails (MCRS[PEG] = 0), and the 0s that are interlocked are merged (ORed) with the 0s that are already present in the 64-bit ECC segment, unless the block is designated as an over-orogram protected block.

Addresses in locked sectors cannot be programmed. Values may be programmed in any or all of eight words, within a page, with a single program sequence. Up to four pages can be programmed at once on a quad-page boundary. The program operation consists of the following sequence of events:

1. Write the address to be programmed using logical address registers located in the PFC. The result of this write will also be reflected physically in the PEADR register.

   **NOTE**

   PEADR writes are initiated by writes that first occur in the PFC. Please see the platform flash memory controller chapter for more information. PEADR reads are allowed, and represent the flash memory physical address.

   **NOTE**

   Ensure the sector that contains the address to be programmed is unlocked. Sector and super sector lock status is latched at this step and does not change until the next program or erase sequence is started.

2. Data to be programmed must be written in the appropriate DATAX register (where X is 0 to 31). All unwritten data words default to FFFF_FFFFh.
3. Change the value in MCR[PGM] from a 0 to a 1.
4. Write a 1 to MCR[EHV] to start the internal program sequence or skip to step 8 to terminate.
5. Wait until MCRS[DONE] becomes 1.

   **NOTE**

   Since MCRS[DONE] clears with MCR[EHV] being set, it may not be possible for software to read MCRS[DONE] as 0 between step 4 and step 5, depending on the operation selected.

6. Confirm MCRS[PEG] = 1.
7. Write 0 to MCR[EHV].
8. Write 0 to MCR[PGM] to terminate the program sequence.

Program may be initiated with the writing of the PEADR register (enabled with a write done in the PFC block). The PEADR write to initiate the program sequence determines the quad-page address to be programmed. Data must also be written to the DATA register(s). This first write is referred to as an interlock write. Unwritten locations default to a data value of FFFF_FFFFh.

An interlock write must be performed before writing 1 to MCR[PGM], and MCR[PGM] must be set before writing 1 to MCR[EHV] during a program sequence. A program sequence may be terminated by writing 0 to MCR[PGM] prior to writing 1 to MCR[EHV].

While MCR[EHV] is 1 and MCRS[DONE] is 0, MCR[EHV] may be cleared, resulting in a program abort. A program abort forces the embedded flash memory to step 8 of the program sequence. An aborted program results in PEG becoming 0, indicating a failed operation. The data space being operated on before the abort contains indeterminate data.

**Caution**

Aborting a program operation leaves the FC addresses being programmed in an indeterminate data state. This may be recovered by executing an erase on the affected block.

#### 20.5.1.2.1 Program hardware locking

Hardware locking which affects the program and erase operations is available for UTest sector and all array blocks.

#### 20.5.1.2.2 Over-program protection enable

One-time programming can be enabled. If over-program protection is enabled, any double word that has already been programmed cannot be programmed again. The over-program protection enable does not affect the erase operation. Attempts to over-program result in MCRS[PEG] becoming 0. If any double word within a quad-page has an over-program protection violation, MCRS[PEG] becomes 0, and no double words are programmed.

One-time programming can be enabled for the UTest NVM sector.

### 20.5.1.3 Erase

An erase changes the value stored in all bits of the selected sector or block to logic 1, and operates on a sector or a block in the main address space. The erase sequence is fully automated within the flash memory. Locked sectors cannot be erased. The erase sequence consists of the following events:

1. Write to the sector or block address to be erased using logical address registers located in the PFC. The result of this write is reflected physically in the PEADR register. One, and only one, DATA register must also be written. This is referred to as an erase interlock write.

   **NOTE**

   PEADR writes are initiated by writes that first occur in the PFC. Please see the platform memory flash controller chapter for more information. PEADR reads are allowed, and represent the flash memory physical address.

   **NOTE**

   The selected block or sector must be unlocked prior to initiating an erase with the appropriate sector or super sector lock registers. Sector and super sector lock status is latched at this step and does not change until the next program or erase sequence starts. If a block is selected for erase, all the sectors and super sectors must be unlocked within that block.

2. Change the value in MCR[ERS] from a 0 to a 1, and at the same time select the size of erase using MCR[ESS].
3. Write a 1 to MCR[EHV] to start the internal erase sequence or skip to step 6 to terminate.
4. Wait until MCRS[DONE] becomes 1.

   **NOTE**

   Since MCRS[DONE] clears with MCR[EHV] being set, it may not be possible for software to read MCRS[DONE] as 0 between step 3 and step 4, depending on the operation selected.

5. Confirm MCRS[PEG] = 1.

6. Write 0 to MCR[EHV].
7. Write 0 to MCR[ERS] to terminate the erase, and if set MCR[ESS] should also be cleared (else the field auto clears).

Erase may be initiated with the writing of the PEADR register (enabled with a write done in the Platform Flash Controller block). The PEADR write to initiate the erase sequence determines the sector or block to be erased. One write must also occur to a DATA register. Data word written during an erase sequence interlock write is ignored. This write is referred to as an erase interlock write.

An erase interlock write must occur before writing 1 to MCR[ERS] (and optionally MCR[ESS]), and MCR[ERS] must be set before writing 1 to MCR[EHV] during an erase sequence. The erase sequence may be terminated by writing 0 to MCR[ERS] prior to writing 1 to MCR[EHV].

While MCR[EHV] is 1 and MCRS[DONE] is 0, MCR[EHV] may be cleared, resulting in an erase abort. An erase abort forces the embedded flash memory to step 7 of the erase sequence. An aborted erase results in PEG becoming 0, indicating a failed operation. The sector or block being operated on before the abort contain indeterminate data.

#### 20.5.1.3.1 Erase hardware locking

Hardware locking which affects the erase and program operations is available for UTest sector and all array blocks. For details, see Program hardware locking.

### 20.5.1.4 Express program

A mechanism is made available to provide an express method to program up to 1024 bits of flash memory. When activated, all ongoing program and erase operations through the register interface are automatically aborted (MCR[EHV] becomes 0 automatically) and the express interface is given priority. When using the express interface, sequence requirements are ignored, although a XPEADR write is required to start the express program sequence.

The flash express program mechanism provides a quicker way to get information programmed into a block of flash memory when an application needs an immediate diary access, or a crash log access.

Addresses in locked sectors cannot be express programmed.

The express program operation consists of the following sequence of events:

1. Write the address to be programmed using the logical address register located in the PFC. The result of this write will also be reflected physically in the XPEADR register.

**NOTE**

XPEADR writes are initiated by writes that first occur to the PFC. Please see the platform flash controller chapter for more information. XPEADR reads are allowed, and represent the flash memory physical address.

**NOTE**

Ensure the sector that contains the address to be programmed is unlocked. Sector and super sector lock status is latched at this step and does not change until the next program, erase or express program is started.

**NOTE**

If XMCR[XPGM] interrupts a program or erase setup after MCR[EHV] is written to 1, the MCR and MCRS registers indicate if the operation was automatically aborted (MCR[EHV] = 0, MCRS[PEG] = 0, MCR[PGM] =1 or MCR[ERS] = 1), or if the operation was successful (MCR[EHV] = 1, MCRS[PEG] = 1, MCR[PGM] =1 or MCR[ERS] = 1). If express program interrupts a program or erase setup prior to MCR[EHV] being written to a 1, the MCR and PEADR registers will be preserved to show the state of the setup upon interruption. If fields MCR[PGM] = 1 or MCR[ERS] = 1 at interruption, the user must terminate the event by writing 0 to these fields. However, if MCR[PGM] or MCR[ERS] were not 1 at interruption, PEADR is preserved and does not need to be rewritten, but the PDATA registers are required to be rewritten for the interrupted operation. After the PDATA write(s) are repeated to finish the interlock write, the remainder of the program and erase sequence can be done to restart the interrupted operation (including doing a terminate after PGM or ERS are written to 1, if desired).

**NOTE**

Before starting the express program, ensure that UTest operations (array integrity or user margin read) are not ongoing (or about to be started). UT0[UTE] must be low before requesting the express program, and remain low during express program operations. Hardware locks are in place to ensure this occurs.

2. Wait until XMCR[XDOK] goes high.
3. Write the DATA registers with the data desired to be programmed. Up to 1024b (128B) may be programmed with express program. Only DATA registers that are written will receive program pulses.
4. Change the value in XMCR[XPGM] from a 0 to a 1.
5. Write a 1 to XMCR[XEHV] to start the internal express program sequence.
6. Wait until XMCR[XDONE] becomes 1.
7. Confirm XMCR[XPEG] = 1.
8. Write 0 to XMCR[XEHV].
9. Write 0 to XMCR[XPGM] to terminate the express program sequence.

### 20.5.2 Low-Power mode

In Low-Power mode, the embedded flash memory is put into a state where VDDF power gating and VFLASH power gating is allowed. It is the lowest current mode for the flash memory. No reads from or writes to the embedded flash memory are possible when in this mode. Most power dissipation is due to leakage in this mode.

Prior to entering Low-Power mode, it is required that all program, erase, and UTest operations be stopped.

When in Low-Power mode, register access is prevented. Flash core accesses are also prevented until power mode is exited. Flash core reads and writes may occur after power mode is exited.

The embedded flash memory returns to a post-reset state in all cases.

### 20.5.3 Program and erase watchdog timer

The embedded flash memory contains an internal watchdog timer that is used to prevent denial of service issues and events.

The watchdog timer is active during two periods of a program and erase event: program/erase setup and program/erase cleanup. If it is desired that program and erase operations also be checked with a watchdog timer, a system resource may be used for this purpose.

Program/erase setup always starts at the time of the PEADR write. An express program setup starts with a write to XPEADR and XMCR[XDOK] = 1. In the event of an Express program interrupt of main or alternate interface setup, the timer starts at the time when the express program operation completes (write of XMCR[XPGM] = 0). The watchdog stops when either MCR[EHV] is set to start the operation, or when the operation is terminated. Termination can occur when MCR[PGM] or MCR[ERS] are written to 0 (prior to MCR[EHV] being written to 1), or MCRS[PES] written to 0. If the express program interrupts the program/erase setup, this also stops the timer on that interface, and express program maintains priority. The above statements also apply to the alternate interface.

**NOTE**

If PEADR writes occur while UT0[UTE] is 1, the watchdog feature does not activate for setup. Watchdog timer is not active during UTest operations.

Program/erase cleanup always starts at the time of MCRS[DONE] = 1 on completion or an abort. In the event of an express program auto-abort of main or alternate interface, the cleanup starts at the time of MCRS[DONE] becoming 1 and the express program operation completed (XMCR[XPGM] written to 0). The watchdog stops when MCR[PGM] or MCR[ERS] are written to 0 at the end of the operation. If express program interrupts the program/erase cleanup, the timer on that interface also stops. The above statements also apply to the alternate interface.

Watchdog timers exist for each interface (main, alternate and express).

If a watchdog timeout occurs, the watchdog interrupt will be reflected in MCRS[WDI], AMCRS[AWDI] or XMCR[XWDI]. Interrupts can be enabled on the main and alternate interfaces using MCR[WDIE] or AMCR[AWDIE]. When a watchdog timeout occurs and an interrupt register asserts, the PEID restrictions on that interface are released. At that point, any master can complete the operation, and PEID locking is no longer enforced. If the watchdog timeout occurs, the EHV bit on that interface (MCR[EHV], AMCR[AEHV], and XMCR[XEHV]) are locked for writing to a 1. In this instance, any master can take the interface to a program and erase terminate. This enables accepted MCRS[WDI], AMCRS[AWDI] or XMCR[XWDI] to become 0, and the operation may restart with a new interlock write.

**NOTE**

If the watchdog timeout occurs before PGM becomes 1 (on any interface), or ERS becomes 1 (on any interface), the sequence must be taken to the point that the PGM or ERS bit for that interface is written and then the field for that interface can be cleared to create the terminate event and to clear the watchdog interrupt.

By default, the watchdog is enabled. Through the alternate MCR, the watchdog may be disabled (AMCR[WDD]), and the watchdog timeouts maybe adjusted (AMCR[WDT]) for all timeouts and all interfaces.

### 20.5.4 UTest mode

UTest mode enables customers to do specific tests that check the integrity of the embedded flash memory.

**NOTE**

When entering UTest mode, a best practice is to ensure all error flags and address error reporting in ADR are cleared to their reset value to ensure robust software and procedure execution.

**NOTE**

UTest mode diagnostic features ECC Logic Check, EDC after ECC Logic Check, Address Encode Logic Check, and Read Reference and Voltage Check, require serialization of flash reads starting with mode setup and ending with mode exit. This may be accomplished by executing diagnostic code from system RAM (with single reads to flash tagged address) as well as ensuring any flash reads outside of this diagnostic code are idle (single core).

#### 20.5.4.1 Array integrity self check

Array integrity is checked using a predefined address sequence (based on UT0[AIS]), and this operation is executed on selected blocks.

The data to be read is customer-specific user code programmed into the flash memory and the correct MISR signature is calculated based on that code.

Any random or nonrandom code is valid. After the operation completes, the results of the reads can be checked by reading the MISR value, to determine if an incorrect read or ECC detection was noted. Array integrity MISR value is calculated after ECC detection and correction. Array integrity requires that the read wait states control registers in the CTL register be set to match the system frequency being used.

The array integrity self check consists of the following steps:

1. Enable UTest mode.
2. Select the block to receive the array integrity check by performing an interlock write to that block. Write logical address register in the PFC (which will be reflected in the PEADR register) and 1 DATA register (PDATA is ignored). The block selected for array integrity check does not need to be unlocked.

**NOTE**

Blocks protected with the Test mode disable seal are still read as part of the array integrity sequence. The resulting read on sealed blocks is not captured in the MISR, but single bit correction, double bit detection and breakpoints are still honored.

**NOTE**

It is not possible to perform array integrity operations on the UTest NVM sector.

3. If desired, write 1 to UT0[AIS] for sequential addressing only.

   **NOTE**

   For normal integrity checks of the flash memory, sequential addressing is recommended. If it is required to more thoroughly check the read path (in a diagnostic mode), AIS shall remain 0 to examine more read transitions. This sequence takes more time.

4. Seed the MISR registers (UM0 - UM9) with desired values.
5. If breakpoints are desired, write 1 to UT0[AIBPE], and ensure that MCRS[EER] and MCRS[SBC] are 0. If it is desired to break on a single bit correction, ensure that UT0[SBCE] = 1.
6. Write 1 to UT0[AIE].
   a. If desired, the array integrity operation may be aborted before UT0[AID] goes high. This may be done by writing 0 to UT0[AIE] and then continuing to the next step. Note that in the event of an aborted array integrity check, MISRs contain a signature for the portion of the operation that was completed prior to the abort and are not deterministic. Before performing another array integrity operation, the UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8, and UM9 registers may need to be initialized to the desired seed value by doing register writes.
   b. If desired, the array integrity operation may be suspended before UT0[AID] goes high. This may be done by writing 1 to UT0[AISUS] to request an array integrity suspend. After UT0[AISUS] becomes 1, the user should wait for UT0[AID] to become 1, which indicates that the flash memory has entered the suspend state, and normal reads to the flash memory may be done. After UT0[AID] becomes 1, UT0[AISUS] may be written to 0 to resume the array integrity sequence.

   **NOTE**

   User mode array reads requested during the array integrity test are ignored to ensure that the array integrity operation is not corrupted. The memory array does not respond to array read requests during this time. User mode array reads may be executed if suspended or at a breakpoint.

7. Wait until UT0[AID] becomes 1.
8. If breakpoints are enabled, check if UT0[NAIBP] = 1. If the value of this field is 1, MCRS[EER], MCRS[SBC], and ADDR may be checked to determine the cause of the break and the address of the break. Prior to resuming the operation, write 1 to clear MCRS[SBC] or MCRS[EER]. Then, the operation may be resumed by writing 0 to UT0[NAIBP]. Continue to wait until UT0[AID] becomes 1. If breakpoints are not enabled, or if UT0[NAIBP] = 0 when UT0[AID] becomes 1, the operation is complete. Continue to the next step.
9. Read values in MISRs (UM0 - UM9) to ensure correct signature.

   **NOTE**

   Array integrity reads may be done to unselected (or non-present) locations. Reads done to these locations do not update the MISRs, MCRS[EER] and MCRS[SBC].

10. Write 0 to UT0[AIE].

#### 20.5.4.2 User margin read

User margin read may be done using the array integrity interface, and has all the associated features of the array integrity interface (MISR and breakpoints).

User margin reads are done at a read margin level, checking for erased bits or programmed bits encroaching on the nominal read level.

User margin read requires that the Read Wait States control registers in the CTL register be written to match the system frequency being used. Margin ECC corrections and detections are noted during the user margin read test. Margin read MISR value is calculated after ECC detection and correction.

The data to be read is customer-specific user code programmed into the flash memory.

Any random or non-random code is valid. After the operation completes, the margin read results can be checked by reading MCRS[EER] and the MCRS[SBC] to determine if zero, one, or two bits are being detected by the margin read, as well as checking the MISR signature.

The use model for margin read is in the event of a user-detected single bit correction (through user reads), a margin read may be done to check for a possible second bit falling within the selected margin levels.

The user margin read consists of the following steps:

1. Enable UTest mode.
2. Select the block to receive the array integrity check by performing an interlock write to that block. Write logical address register in the PFC (which will be reflected in the PEADR register) and 1 DATA register (PDATA is ignored). The block selected for user margin read check does not need to be unlocked.

   **NOTE**

   Blocks protected with the Test mode disable seal are still read as part of the margin read sequence. The resulting read on sealed blocks is not captured in the MISR, but single bit correction, double bit detection and breakpoints are still honored.

   **NOTE**

   It is not possible to perform margin read operations on the UTest NVM sector.

3. Write 1 to UT0[AIS] for sequential addressing only.

   **NOTE**

   For margin read checks of the flash memory, sequential addressing is recommended. Writing 0 to UT0[AIS] is possible for margin reads, but using the sequence takes more time, and is not recommended.

4. Seed the MISR registers (UM0 - UM9) with desired values.
5. Ensure that MCRS[EER] and MCRS[SBC] are 0.
6. To enable single bit correction reporting during margin read, write UT0[SBCE] = 1.
7. Write 1 to UT0[MRE].
8. Write UT0[MRV] to the desired value depending on if it is desired to do one's margin or zero's margin.
9. Write 1 to UT0[AIE].

   **NOTE**

   User mode array reads requested during the margin read test are ignored to ensure that the margin read operation is not corrupted. The memory array does not respond to array read requests during this time. User mode array reads may be executed if suspended or at a breakpoint.

   **NOTE**

   During margin read operations, with UT0[AID] = 0, it is not recommended to attempt write operations to MCRS[SBC] and MCRS[EER]. It is recommended that these fields only be written during suspend, breakpoints or at the completion of the margin read operation.

10. Wait until UT0[AID] becomes 1.
11. If breakpoints are enabled or a suspend was requested during margin read, the operation may be at a breakpoint or a suspend state. See Array integrity self check for more information.
12. Read values in the MISRs (UM0 - UM9) to ensure correct signature.

**NOTE**

Margin reads may be done to unselected (or non-present) locations. Reads done to these locations do not update the MISRs, MCRS[EER] and MCRS[SBC].

13. Write 0 to UT0[AIE].

#### 20.5.4.3 ECC logic check

ECC logic can be checked by providing data to be read in UD0[EDATA], UD1[EDATA] and/or UD2[EDATAC]. Array reads can then be performed, ensuring expected results.

The ECC logic check consists of the following steps:

1. Enable UTest mode.
2. Write 1 to UT0[EIE].
3. Write to UD0[EDATA], UD1[EDATA] and/or UD2[EDATAC] to provide data and check bit values to be read. Single bit corrections or double bit detections can be simulated by properly selecting data and check bit combinations.
4. Write the page address to receive data provided in step 3 into ADR.
5. Write to UD2[ED3], UD2[ED2], UD2[ED1] and/or UD2[ED0] to select the double words on the page to receive the check.
6. Reads can now be done through the BIU using an array read request. In the event of a BIU read requested from an address that matches the address in ADR, expected data and corrections or detections are observed based on data written into UD0[EDATA], UD1[EDATA] and/or UD2[EDATAC]. MCRS[EER] and MCRS[SBC] can be checked to evaluate the status of reads done.

**NOTE**

In the event of an ECC error or single bit correction, during the ECC logic check (UT0[EIE] = 1), ADR is not loaded, and the address tagged to receive the ECC logic check values is preserved.

7. Once completed, write UT0[EIE] to 0.

#### 20.5.4.4 EDC after ECC logic check

EDC after ECC logic can be checked by providing data to be read in UD3[EDDATA], UD4[EDDATA] and/or UD5[EDDATAC]. Array reads can then be performed, ensuring expected results.

The EDC after ECC logic check consists of the following steps:

1. Enable UTest mode.
2. Write 1 to UT0[EDIE].
3. Write to UD3[EDDATA], UD4[EDDATA] and/or UD5[EDDATAC] to provide data and check bit values to be read.
4. Write the page address to receive data provided in step 3 into ADR.
5. Write to UD5[EDD3], UD5[EDD2], UD5[EDD1] and/or UD5[EDD0] to select the double words on the page to receive the check.
6. Reads can now be done through the BIU using an array read request. In the event of a BIU read requested from an address that matches the address in ADR, expected EDC after ECC errors are observed based on data written into UD3[EDDATA], UD4[EDDATA] and/or UD5[EDDATAC]. MCRS[EEE] can be checked to evaluate the status of reads done.
7. Once completed, write UT0[EDIE] to 0.

#### 20.5.4.5 Address encode logic check

Address encode logic can be checked by inverting the address encode information from the memory array in UA0[AEI] and UA1[AEI]. Array reads can then be performed, ensuring expected results.

The Address encode check consists of the following steps:

1. Enable UTest mode.
2. Write 1 to UT0[AEIE].
3. Write to UA0[AEI] and/or UA1[AEI] to provide address bit(s) to be inverted.
4. Write the page address to receive inverted addresses provided in step 3 into ADR.
5. Reads can now be done through the BIU using an array read request. In the event of a BIU read requested from an address that matches the address in ADR, expected address encode errors are observed based on address invert values written into UA0[AEI] and/or UA1[AEI]. MCRS[AEE] can be checked to evaluate the status of reads done.
6. Once completed, write UT0[AEIE] to 0.

#### 20.5.4.6 Read reference and voltage check

Read reference and voltage detection logic can be checked by writing 1 to UT0[RRIE]. Array reads can be then be performed, ensuring expected results.

The read reference and voltage check consists of the following steps:

1. Enable UTest mode.
2. Write 1 to UT0[RRIE].
3. Write the page address to receive a read reference error MCRS[RRE] and a read voltage error MCRS[RVE] into ADR.
4. Reads can now be done through the BIU using an array read request. In the event of a BIU read requested from an address that matches the address in ADR, expected read reference and read voltage errors are observed based on ADR registers. MCRS[RRE] and MCRS[RVE] can be checked to evaluate the status of reads done.
5. Once completed, write UT0[RRIE] to 0.

### 20.5.5 Data flash memory requirements for EEPROM emulation

The embedded flash memory may be used to emulate an EEPROM utilizing software drivers, strategic over-programming of double words in the flash memory and following the below requirement and best practices in EEPROM software drivers.

It is required that the EEPROM emulation driver must build in fault tolerance allowing for the ability to “skip” records. In the case of an unsuccessful program, the ability to retry programming in the next available record location is required (record retirement). In the case of an unsuccessful sector erase, the ability to retire a sector is required (sector retirement).

In addition, following are best practices for the EEPROM emulation software driver to be considered:

- Choose a record scheme which allows for grouping of EEPROM data contents that are updated at one time such that unchanged data is not needlessly copied while minimizing record qualifier and status overhead. A variable length record scheme may be used to allow for grouping of data that is written together and limiting total record overhead.
- Copy only the valid records during a sector change and re-constitution of EEPROM data set to avoid needless copying of data within flash memory. This avoids rebuilding the entire data set.
- Only redundantly storing critical data which cannot be recovered in two sectors if attempting to protect against record or sector failures.
- Load level total program/erase cycles applied to flash memory locations allocated for EEPROM emulation by using a round-robin sector scheme.

When combining round-robin with sector retirement requirement, three or more sectors must be allocated for EEPROM emulation inclusive of extra spare sector(s).

For details, see NXP application notes published on this topic.

## 20.6 Initialization information

A reset is the highest priority operation for the embedded flash memory and terminates all other operations.

The embedded flash memory uses reset to initialize register and status fields to their default reset values. If the embedded flash memory is executing a program or erase operation (MCR[PGM] = 1 or MCR[ERS] = 1) and a reset is issued, the operation aborts and the embedded flash memory disables the high voltage logic without causing any damage to the high voltage circuits. Reset aborts all operations and forces the embedded flash memory into User mode, ready to receive accesses.

After reset is requested, MCRS[DONE] becomes 0, and remains low during reset and reset recovery.

At the end of reset recovery, MCRS[DONE] transitions from 0 to 1.

After a reset completes, register reads may be performed.

**NOTE**

Registers that require updating from UTest NVM information, or other inputs, may not read updated values until MCRS[DONE] becomes 1.

During reset recovery, register writes are not allowed until MCRS[DONE] becomes 1 to indicate reset recovery is complete.

**Caution**

Resetting during a program or erase operation leaves the FC blocks being programmed or erased in an indeterminate data state. This may be recovered by executing an erase on the affected blocks.

## 20.7 Memory map and register description

### 20.7.1 c40asf flash memory register descriptions

The embedded flash memory map consists of a flash memory array (which includes main array space and UTest NVM space) and a region of registers associated with the programming model that enable flash memory array operation and modification.

The address space consists of up to 6 blocks (with restrictions), and blocks can be 2 MB, 1 MB, 512 KB or 256KB in size.

#### 20.7.1.1 c40asf_flash memory map

FLASH base address: 402E_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Module Configuration (MCR) | 32 | RW | 00FF_0000h |
| 4h | Module Configuration Status (MCRS) | 32 | RW | 0000_C100h |
| 8h | Extended Module Configuration (MCRE) | 32 | R | 0000_0000h |
| Ch | Module Control (CTL) | 32 | RW | 0000_0600h |
| 10h | Address (ADR) | 32 | RW | 0000_0000h |
| 14h | Program and Erase Address (PEADR) | 32 | R | 0000_0000h |
| 50h | Sector Program and Erase Hardware Lock (SPELOCK) | 32 | R | FFFF_FFFFh |
| 54h | Super Sector Program and Erase Hardware Lock (SSPELOCK) | 32 | R | 0000_0FFFh |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 70h | Express Sector Program and Erase Hardware Lock (XSPELOCK) | 32 | R | FFFF_FFFFh |
| 74h | Express Super Sector Program and Erase Hardware Lock (XSSPELOCK) | 32 | R | 0000_0FFFh |
| 90h | Test Mode Disable Password Check (TMD) | 32 | RW | 0000_0000h |
| 94h | UTest 0 (UT0) | 32 | RW | 0000_0001h |
| 98h - B8h | UMISRn (UM0 - UM8) | 32 | RW | 0000_0000h |
| BCh | UMISR9 (UM9) | 32 | RW | 0000_0000h |
| D0h | UTest Data 0 (UD0) | 32 | RW | 0000_0000h |
| D4h | UTest Data 1 (UD1) | 32 | RW | 0000_0000h |
| D8h | UTest Data 2 (UD2) | 32 | RW | 0000_0000h |
| DCh | UTest Data 3 (UD3) | 32 | RW | 0000_0000h |
| E0h | UTest Data 4 (UD4) | 32 | RW | 0000_0000h |
| E4h | UTest Data 5 (UD5) | 32 | RW | 0000_0000h |
| E8h | UTest Address 0 (UA0) | 32 | RW | 0000_0000h |
| ECh | UTest Address 1 (UA1) | 32 | RW | 0000_0000h |
| F0h | Express Module Configuration (XMCR) | 32 | RW | 00FF_C000h |
| F4h | Express Program Address (XPEADR) | 32 | R | 0000_0000h |
| 100h - 17Ch | Program Data (DATA0 - DATA31) | 32 | RW | FFFF_FFFFh |

### 20.7.1.2 Module Configuration (MCR)

#### Offset

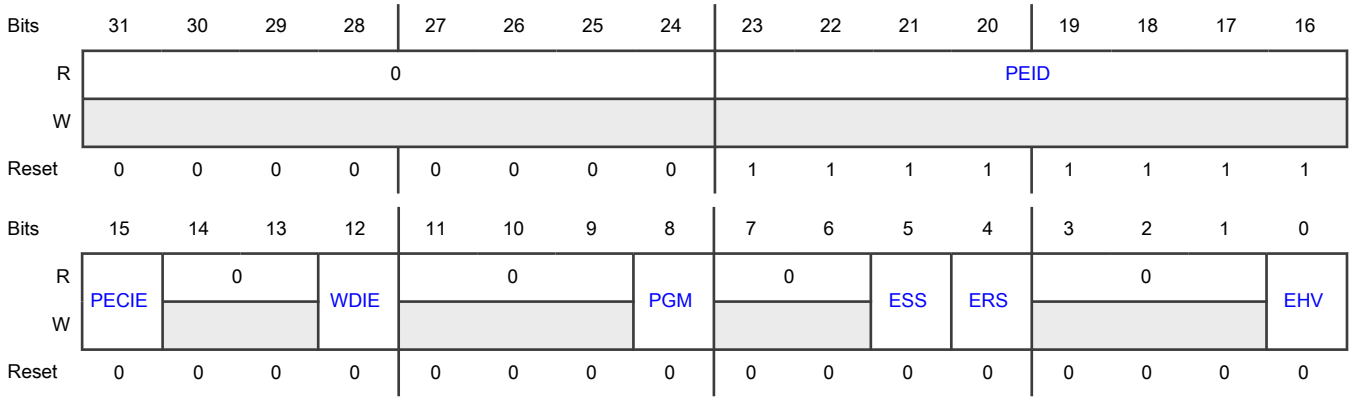
| Register | Offset |
|---|---|
| MCR | 0h |

#### Function

**NOTE**

- A number of Module Configuration Register (MCR) bits are locked against write by other bits. These locks are discussed in relationship to each bit in this section. Simultaneously writing bits which lock each other out is also discussed in Functional description.
- See Functional description for information about simultaneous MCR writes, and priority levels.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16<br>PEID | Program and Erase Master/Domain ID<br>This field shows the ID of the master that has started the Program or Erase sequence (as well as Array Integrity and User Margin Reads). The ID is latched when the sequence has started (writing of the PEADR register). Upon completion of an operation (Program (MCR[PGM] written to 0 and MCRS[PES] equals 0), Erase (MCR[ERS] written to 0 and MCRS[PES] equals 0), Array Integrity (UT0[AIE] written to 0), User Margin Read(UT0[AIE] written to 0)), or Program and Erase Sequence clear (MCRS[PES] cleared to 0) the PEID field will return to an all ones state. See the chip-specific information for a list of Master IDs. |
| 15<br>PECIE | Program/Erase Complete Interrupt Enable<br>PECIE provides a mechanism to trigger an interrupt request upon the assertion of the DONE status due to a high voltage event (program or erase) finishing (normal, abort). If PECIE is written while not in a high voltage event, the interrupt will not immediately trigger, but will trigger after the next high voltage event is completed. Writing (and reading) of this bit will be restricted to the master that updated the PEADR register as captured in the PEID register at the start of the Program or Erase sequence.<br>**NOTE**<br>See the chip-specific information for interrupt details<br>0b - Interrupt request not generated when MCRS[DONE] is 1<br>1b - Interrupt request generated when MCRS[DONE] is 1 |
| 14-13<br>— | Reserved |
| 12<br>WDIE | Watch Dog Interrupt Enable<br>WDIE provides a mechanism to trigger an interrupt request upon the assertion of the WDI bit on the main interface. If WDIE is asserted, when WDI asserts, an interrupt from the flash will trigger to the system. The interrupt from the flash will mirror exactly the WDI bit. WDIE does not affect the register bit WDI. Writing (and |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | reading) of this bit will be restricted to the master that updated the PEADR register as captured in the PEID register at the start of the Program or Erase sequence.<br>**NOTE**<br>See the chip-specific information for interrupt details<br>0b - Watchdog interrupt not enabled<br>1b - Watchdog interrupt enabled |
| 11-9<br>— | Reserved |
| 8<br>PGM | Program<br>PGM is used as part of the setup for a program operation. A 0 to 1 transition of PGM after program interlock write(s) is part of the program sequence. A 1 to 0 transition of PGM ends the program sequence. PGM can be set while in user mode read (ERS is low and UTE is low), and after the interlock write is completed. PES must also be low to enable PGM to be set.<br>PGM can be cleared only when EHV is low and DONE is high. PGM is cleared on reset. Writing (and reading) of this bit will be restricted to the master that updated the PEADR register as captured in the PEID register at the start of the Program or Erase sequence.<br>0b - Flash memory not executing a program sequence<br>1b - Flash memory executing a program sequence |
| 7-6<br>— | Reserved |
| 5<br>ESS | Erase Size Select<br>ESS is used to qualify the embedded flash memory erase operation for either sector erase or block erase. If ESS is selected for block erase, only the main array of that block will be erased.<br>**NOTE**<br>Block erase is for Factory use only, under controlled conditions. See Appendix A (Electrical Specifications) for more information on environmental condition requirements, and cycle limit.<br>**NOTE**<br>If interlock write occurs to UTest sector, ESS will be locked from setting to a 1. Block Erase of UTest sector is not supported, only sector erase is allowed. Erase of UTest space is not allowed if sealed, although ERS can be set and a PEG error will occur in response to the HV request.<br>ESS can only be written to a 1 at the same time that ERS is written to a 1. If ESS is set, ESS will auto clear with ERS clearing (independent of the value written to ESS). ESS can not be cleared by writing a zero. ESS is cleared on reset. Writing (and reading) of this bit will be restricted to the master that updated the PEADR register as captured in the PEID register at the start of the Program or Erase sequence. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Flash memory erase is on a sector<br>1b - Flash memory erase is on a block |
| 4<br>ERS | Erase<br>ERS is used as part of the setup for an erase operation. A 0 to 1 transition of ERS after an erase interlock write is part of the erase sequence. A 1 to 0 transition of ERS ends the erase sequence. ERS can only be set in user mode read (PGM is low and UTE is low), and after the interlock write is completed. PES must also be low to enable ERS to be set.<br>ERS can be cleared only when EHV is low and DONE is high. ERS is cleared on reset. Writing (and reading) of this bit will be restricted to the master that updated the PEADR register as captured in the PEID register at the start of the Program or Erase sequence.<br>0b - Flash memory not executing an erase sequence<br>1b - Flash memory executing an erase sequence |
| 3-1<br>— | Reserved |
| 0<br>EHV | Enable High Voltage<br>The EHV bit enables the embedded flash memory for a high voltage program/erase operation. EHV is cleared on reset. EHV must be set after the PGM or ERS bit is set in a program or erase sequence. Writing (and reading) of this bit will be restricted to the master that updated the PEADR register as captured in the PEID register at the start of the Program or Erase sequence. EHV may be set, initiating a program/erase under one of the following conditions:<br>• Erase (ERS = 1, PEP = 0, PES = 0, WDI = 0)<br>• Program (PGM = 1, PEP = 0, PES = 0, WDI = 0)<br>Clearing EHV while DONE is low will abort the current program/erase high voltage operation. An abort causes the value of PEG to be cleared, indicating a failed program/erase; address locations being operated on by the aborted operation contain indeterminate data after an abort. EHV may not be written to a 1 after an abort is requested (EHV being cleared) and before DONE transitions high. EHV may not be written to a 1 after it is cleared as part of a program or erase cleanup, until after the next PEADR write.<br>**NOTE**<br>Aborting a high voltage operation leaves addresses in an indeterminate data state. This may be recovered by executing an erase on the affected sectors.<br>0b - Flash memory is not enabled to perform a high voltage operation.<br>1b - Flash memory is enabled to perform a high voltage operation. |

### 20.7.1.3 Module Configuration Status (MCRS)

#### Offset

| Register | Offset |
|---|---|
| MCRS | 4h |

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EER | SBC | AEE | EEE | 0 | | RVE | RRE | 0 | | | RWE | 0 | | PEP | PES |
| W | W1C | W1C | W1C | W1C | | | W1C | W1C | | | | W1C | | | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DONE | PEG | 0 | WDI | 0 | | EPEG | TSPELOCK | 0 | | | | | | | RE |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31<br>EER | ECC Event Error<br>This bit provides information on previous reads (either user initiated reads or internally initiated reset reads).If a double bit detection occurred, the EER bit is set to a '1'. This bit must then be cleared, or a reset must occur before it returns to a 0 state. This bit may not be set by software. In the event of a single bit detection and correction, this bit is not set. If EER is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of EER) are correct. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect.<br>0b - Reads occurring normally<br>1b - ECC error occurred during a previous read |
| 30<br>SBC | Single Bit Correction<br>SBC provides information on previous reads, if the SBCE is set. If a single bit correction occurred, the SBC bit is set to a 1. This bit must then be cleared, or a reset must occur before this bit returns to a 0 state. If SBC is not set, or remains 0, this indicates that all previous reads (from the last reset, or clearing of SBC) did not require a correction. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect.<br>0b - Reads occurring without corrections<br>1b - Single bit correction occurred during a previous read |
| 29<br>AEE | Address Encode Error |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | AEE provides information on previous reads monitoring the address encode feature. On every read request to the flash, the incoming address is compared to an encoded address (row, column, and block) coming back from the memory array using the read data sense amplifier timing. If these two values do not match (zero selected, multiple selected, wrong selected), or the timing is incorrect, an address encode error will be recorded. If an address encode mismatch is detected, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect.<br>0b - Reads are occurring without address encode mismatches<br>1b - Previous read may be corrupted based on address encode mismatch |
| 28<br>EEE | EDC after ECC Error<br>EEE provides information on previous reads monitoring the EDC after ECC feature. On every read request to the flash, ECC is recalculated serially, and if there is a mismatch between the ECC calculations (taking into account corrections or detections) a late error will be reported. If an EDC after ECC mismatch is detected, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect.<br>0b - Reads are occurring without EDC after ECC mismatches<br>1b - Previous read may be corrupted based on ECC calculation errors |
| 27-26<br>— | Reserved |
| 25<br>RVE | Read Voltage Error<br>RVE provides information on previous reads monitoring the read voltage. If the read voltage is detected to be out of range, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect.<br>0b - Reads are occurring without voltage issues<br>1b - A previous read may have been corrupted due to read voltage being out of range |
| 24<br>RRE | Read Reference Error<br>RRE provides information on previous reads monitoring the read reference. If the read reference is detected to be out of range, this bit is set to indicate that previous reads requested may have been corrupted. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect.<br>0b - Reads occur without reference issues<br>1b - Previous read may be corrupted because of read reference being out of range |
| 23-21<br>— | Reserved |
| 20<br>RWE | Read-While-Write Event Error<br>This bit provides information on previous read-while-write (RWW) reads. If an RWW error occurs, this bit is set to 1. The bit must then be cleared, or a reset must occur before it returns to a 0 state. If RWE is not set, or |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | remains 0, this indicates that all previous RWW reads (from the last reset, or clearing of RWE) are correct. Since this bit is an error flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect.<br>0b - Reads occur normally<br>1b - RWW error occurred during a previous read |
| 19-18<br>— | Reserved |
| 17<br>PEP | Program and Erase Protection Error<br>PEP provides information about program and erase operations with respect to protection errors. A protection error occurs if a program is attempted to a locked sector or super sector, or if an erase is selected to a locked sector or super sector. This is evaluated prior to the operation beginning, and if an error is detected, high voltage operations (either a Program or Erase) will not be attempted for this request.<br>**NOTE**<br>If a location has both OTP and Lock protection, the response from the NVM will be PEP=1 only.<br>If PEP is asserted, it must be cleared prior to attempting another high voltage operation. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect.<br>0b - Program and erase protection errors do not exist<br>1b - Previous program or erase protection error encountered |
| 16<br>PES | Program and Erase Sequence Error<br>PES provides information about program and erase operations with respect to sequence errors. A sequence error occurs when the program or erase sequence is not followed exactly. If an “out-of-order” event is detected, PES will assert, and the remainder of the sequence will not be accepted. PES monitoring begins when the PEADR register is written and ends when EHV is set to start the operation. PES monitoring only applies to Program and Erase (Sector and Block). It does not apply to Express Program, or UTest operations. Following is a complete list of sequence error conditions:<br>• Any specific DATA register written twice.<br>• More than one DATA register write for ERS operations.<br>• Attempts to write PGM or ERS register out of sequence.<br>• Attempts to write PGM after ERS, or ERS after PGM.<br>• Attempts to write DATA registers after PGM or ERS are written.<br>• Attempts to write EHV out of sequence.<br>Writes to the MCR that are protected by the MCR Priority Levels (i.e. simultaneous PGM, ERS and EHV writes) will not result in a PES condition.<br>Attempts to write MCR or DATA registers by a master that does not match the PEID, will not result in a PES condition. Those reads and writes will be blocked, and the master in control of the interface may continue with its' sequence. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Clearing of PES (due to a sequence error) has the same effect as the clearing of PGM or ERS after a program or erase operation. PEADR and DATA registers are cleared, PEID is cleared, and the operation must be re-started from the beginning. If PGM or ERS are already set, they must be cleared before PES is cleared. PES can only be cleared once the PGM or ERS bit for the sequence in progress is cleared. Since this bit is a status flag, it must be cleared to a 0 by writing a 1 to the register location. A write of 0 has no effect.<br>0b - Program and erase sequence errors do not exist<br>1b - Previous program or erase sequence encountered an error |
| 15<br>DONE | State Machine Status<br>DONE indicates whether the embedded flash memory is performing a high voltage operation. DONE is set to a 1 on termination of the embedded flash memory reset. DONE is read only. DONE is set to a 1 at the end of program and erase high voltage sequences.<br>**NOTE**<br>This field transitions from a 0 to 1 during reset and remains at 1 after reset.<br>0b - Performing a high voltage operation<br>1b - Not executing a high voltage operation |
| 14<br>PEG | Program/Erase Good<br>The value of PEG is updated automatically during the program and erase high voltage operations. Aborting a program/erase high voltage operation causes PEG to be cleared, indicating the sequence failed. PEG is set to a 1 when the embedded flash memory is reset. PEG is read only.<br>The value of PEG is valid only when PGM = 1 or ERS = 1 and after DONE transitions from 0 to 1 due to an abort or the completion of a program/erase operation. PEG is valid until PGM/ERS makes a 1 to 0 transition or EHV makes a 0 to 1 transition.<br>**NOTE**<br>1. If program or erase operations are attempted on sector(s) that are locked, the response from embedded flash memory is PEG = 1, indicating that the operation was successful, and the contents of the sector(s) are properly protected from the program or erase operation. PEG = 1 is also true if an abort occurs during an HV request to a locked sector.<br>2. If a program or erase operation is attempted to a location marked as OTP, the response from the embedded flash memory is PEG = 0, indicating that the operation was not allowed. The value interlocked is not programmed, since desired double word was already programmed with a previous program operation. Erase is prevented on OTP locations.<br>0b - Program or erase operation failed<br>1b - Program or erase operation successful |
| 13<br>— | Reserved |
| 12 | Watch Dog Interrupt |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| WDI | WDI is a status register to indicate that the Watchdog Timer for Program or Erase had expired. WDI is status only, and will be automatically cleared once the operation that caused the watchdog timeout is terminated or clean up from the operation is completed (clearing of PGM (with PES low), ERS (with PES low) or PES bit).<br>0b - Normal Operation, Watchdog Timer has not expired.<br>1b - Program Watchdog Timer has expired. |
| 11-10<br>— | Reserved |
| 9<br>EPEG | ECC Enabled Program/Erase Good<br>EPEG is a qualifier to PEG to indicate if a passing program or erase required ECC Enabled verifies to pass. In the event of a failed operation (PEG=0), EPEG will always remain 0. The value of EPEG is updated automatically during the program and erase high voltage operations. EPEG is set to a 0 when the embedded flash memory is reset. EPEG is read only.<br>The value of EPEG is valid only when PGM = 1 or ERS = 1 and after DONE transitions from 0 to 1 due to an abort or the completion of a program/erase operation. EPEG is valid until PGM/ERS makes a 1 to 0 transition or EHV makes a 0 to 1 transition.<br>0b - Program or erase operation did not require ECC Enabled verifies.<br>1b - Program or erase operation required ECC Enabled verifies to pass. |
| 8<br>TSPELOCK | UTest NVM Program and Erase Lock<br>TSPELOCK reflects the status of the hardware program and erase protection input to the flash module.<br>The TSPELOCK register is latched on the PEADR[PEASAD] write (UTest Sector Interlock), is not writable, and is status only. During high voltage events, the status is locked. When not interlocked this bit will read 1.<br>The default value of the TSPELOCK bits is program and erase protected.<br>0b - Corresponding sector not locked, and may be programmed or erased<br>1b - Corresponding sector protected from the program and erase sequences |
| 7-1<br>— | Reserved |
| 0<br>RE | Reset Error<br>This bit provides information on previous resets. Checks are done within the flash and if a coherency issue or ECC error is detected during the reset reads used for trimming on the flash, this status flag will assert. This bit is status only, and is not writable.<br>0b - Reset occurred without errors<br>1b - Reset error encountered |

### 20.7.1.4 Extended Module Configuration (MCRE)

#### Offset

| Register | Offset |
|---|---|
| MCRE | 8h |

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | n1M | | | | 0 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset [1] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | n512K | | 0 | | | | | | n256K | | 0 | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1. The reset value of this register is set at the factory and varies among chips. See the chip-specific c40asf information for details.

#### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-21<br>n1M | Number of 1 MB Blocks<br>000b - Zero 1 MB blocks<br>001b - One 1 MB block<br>010b - Two 1 MB blocks<br>011b - Three 1 MB blocks<br>100b - Four 1 MB blocks<br>101b - Reserved<br>110b - Reserved<br>111b - Reserved |
| 20-16<br>— | Reserved |
| 15-14 | Number of 512 KB Blocks |

*Table continues on the next page...*

*Table continued from the previous page...*

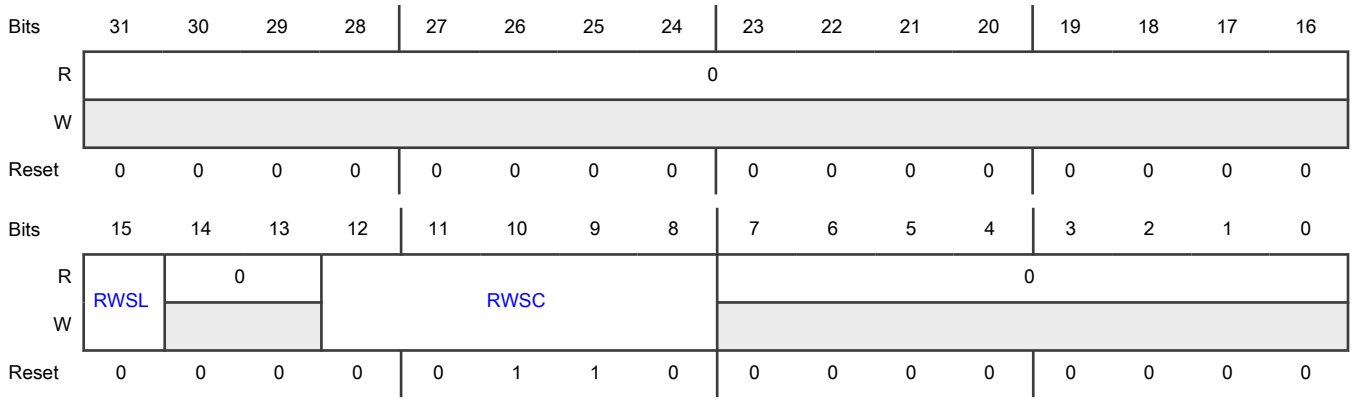
| Field | Function |
|---|---|
| n512K | 00b - Zero 512 KB blocks<br>01b - One 512 KB block<br>10b - Two 512 KB blocks<br>11b - Four 512 KB blocks |
| 13-8<br>— | Reserved |
| 7-6<br>n256K | Number of 256 KB Blocks<br>00b - Zero 256 KB blocks<br>01b - One 256 KB block<br>10b - Two 256 KB blocks<br>11b - Four 256 KB blocks |
| 5-0<br>— | Reserved |

### 20.7.1.5 Module Control (CTL)

#### Offset

| Register | Offset |
|---|---|
| CTL | Ch |

#### Diagram



### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>RWSL | Read Wait State Lock<br>Read Wait State Lock. The RWSL bit locks the read wait state control register (RWSC). If RWSL is set, then the RWSC field can not be written. RWSL is not clearable by a register write, it is only cleared upon reset. Once RWSL is written to a 1, it will remain a 1 until the next reset and RWSC will be locked from writing. RWSC and RWSL are allowed to be written simultaneously. RWSL does not affect readability of RWSC.<br>0b - RWSC not locked and available for writing<br>1b - RWSC locked and unavailable for writing |
| 14-13<br>— | Reserved |
| 12-8<br>RWSC | Wait State Control<br>This field controls the number of wait states to be added to the best-case flash array access time for reads. RWSC is only writable when RWSL is a 0. The best case flash array access time for reads is one cycle.<br>This field must be set to a value corresponding to the operation frequency of the Flash and the actual read access time of the Flash to the flash boundary. Total read latency will be SoC dependant based on the how the cores are coupled within the platform. The frequency information, and total latency details are documented in the SoC specification.<br>Higher operating frequencies require non-zero settings for this field for proper flash operation.<br>**NOTE**<br>• Updates to this configuration field that control reads must take place only when the flash memory is idle. Changing configuration settings while flash memory access is in progress can lead to non-deterministic behavior.<br>• Values of RWSC[4:0] not listed here are reserved.<br>**NOTE**<br>0_0000b - Reserved<br>0_0001b - One additional wait state is added.<br>0_0010b - Two additional wait states are added.<br>0_0011b - Three additional wait states are added.<br>0_0100b - Four additional wait states are added.<br>0_0101b - Five additional wait states are added.<br>0_0110b - Six additional wait states are added.<br>0_0111b - Seven additional wait states are added.<br>0_1000b - Eight additional wait states are added. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0_1001b - Reserved<br>1_1111b - Reserved |
| 7-0<br>— | Reserved |

### 20.7.1.6 Address (ADR)

#### Offset

| Register | Offset |
|---|---|
| ADR | 10h |

#### Function

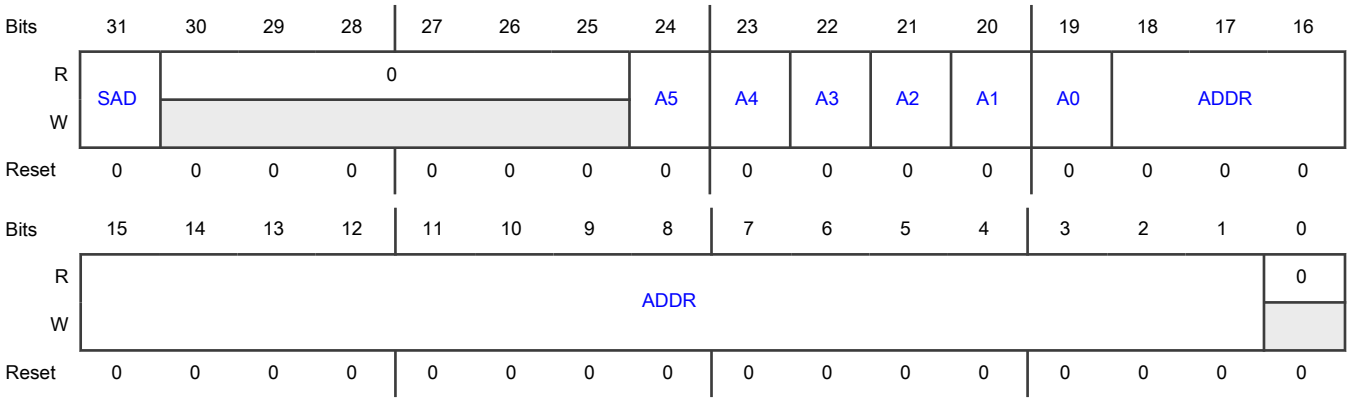
The Address register (ADR) provides the first failing address in the event of a failure (ECC or PGM/Erase state machine), as well as single bit correction information. The address register is also writable for UTest Mode operations. The address is identified using a combination of bits that identify the memory region and offset address.

**NOTE**

- The address given is an offset from the base address of the address space.
- The block numbering scheme that corresponds to the offset address is based on the flash memory's *internal* addressing scheme, which may be different than the chip's system addressing scheme.

Understanding the mapping between the flash address, which is specified relative to the flash module's internal addressing scheme, and the location within the chip's system memory map requires a clear understanding of the flash memory layout. See the chip-specific information for a detailed explanation of the chip's flash memory layout and how to map system addresses to the flash module's internal addressing.

#### Diagram



### Fields

| Field | Function |
|---|---|
| 31<br>SAD | UTest NVM Address<br>Qualifies the address captured during an ECC event error, single bit correction, or state machine operation.<br>See the description of the ADDR field for more information. This field is not writeable if UTE = 1, and is ignored if it's in UTest mode.<br>0b - Address captured or to be accessed is from the main array space<br>1b - Address captured or to be accessed is from the UTest NVM array space |
| 30-25<br>— | Reserved<br>Reserved; reset to 0 |
| 24<br>A5 | Address Region 5<br>Qualifies the address field (ADDR) to region 5<br>If the value of this field is 1, the ADDR field maps to region 5. See the description of the ADDR field for more information. If the region is not present, this field is locked to 0.<br>0b - Address captured or to be accessed is not from region 5<br>1b - Address captured or to be accessed is from region 5 |
| 23<br>A4 | Address Region 4<br>Qualifies the address field (ADDR) to region 4<br>If the value of this field is 1, the ADDR field maps to region 4. See the description of the ADDR field for more information. If the region is not present, this field is locked to 0.<br>0b - Address captured or to be accessed is not from region 4<br>1b - Address captured or to be accessed is from region 4 |
| 22<br>A3 | Address Region 3<br>Qualifies the address field (ADDR) to region 3<br>If the value of this field is 1, the ADDR field maps to region 3. See the description for the ADDR field for more information. If the region is not present, this field is locked to 0.<br>0b - Address captured or to be accessed is not from region 3<br>1b - Address captured or to be accessed is from region 3 |
| 21<br>A2 | Address Region 2<br>Qualifies the address field (ADDR) to region 2<br>If the value of this field is 1, the ADDR field maps to region 2. See the description of the ADDR field for more information. If the region is not present, this field is locked to 0.<br>0b - Address captured or to be accessed is not from region 2<br>1b - Address captured or to be accessed is from region 2 |
| 20 | Address Region 1 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| A1 | Qualifies the address field (ADDR) to the region<br>If the value of this field is 1, the ADDR field maps to region 1. See the description for the ADDR field for more information. If the region is not present, this field is locked to 0.<br>0b - Address captured or to be accessed is not from region 1<br>1b - Address captured or to be accessed is from region 1 |
| 19<br>A0 | Address Region 0<br>Qualifies the ADDR field to region 0. If A0 = 1, then the ADDR field maps to that region. If the region is not present, this field is locked to 0. For details, see the field description for ADDR.<br>0b - Address captured or to be accessed is not from region 0<br>1b - Address captured or to be accessed is from region 0 |
| 18-1<br>ADDR | Address<br>The ADR register provides the first failing address in the event of ECC event error (EER set), single bit correction (SBC set), as well as providing the address of a failure that may have occurred in a state machine operation (PEG cleared). ECC event errors take priority over single bit corrections, which take priority over state machine errors. This is especially valuable in the event of an RWW operation, where the read senses an ECC error or single bit correction, and the state machine fails simultaneously. The failing address for ECC event error is held until EER is cleared. The failing address for single bit correction is held until an ECC event error, or until SBC is cleared. State machine address is held until an ECC event error or single bit correction event occurs, or until the next state machine event (PEG cleared) occurs. This address is always a double word address that selects 64 bits.<br>The ADR register is writable, and can be used in the UTest ECC Logic Test, UTest EDC after ECC Logic Test, and UTest Address Encode Logic Test. If any of these tests are enabled (UT0[EIE] = 1, UT0[EDIE] = 1, or UT0[AEIE] = 1) then the ADR register will not update for ECC event error, single bit correction or state machine errors.<br>If MCRS[EER] or MCRS[SBC] are set, the ADR register is locked from writing. MCRS[PEG] does not affect the writeability of the ADR register.<br>ADDR[18:1] is an offset from a base address of 0h for each block region. The A0, A1, A2, A3, A4, and A5 bits qualify the block size region the ADDR field. |
| 0<br>— | Reserved<br>Reserved; reset to 0 |

### 20.7.1.7 Program and Erase Address (PEADR)

**Offset**

| Register | Offset |
|---|---|
| PEADR | 14h |

### Function

The Program and Erase Address register (PEADR) is used to provide the address to be programmed or the location of the sector or block to be erased. The program and erase address register is read only in user mode (user mode writes occur with writes done to the Platform Flash Controller). The address is identified using a combination of bits that identify the memory region and offset address.

**NOTE**

- The address provided is an offset from the base address of the address space.
- The block numbering scheme that corresponds to the offset address is based on the flash memory's *internal* addressing scheme, which may be different from the chip's system addressing scheme.

Understanding the mapping between the flash address, which is specified relative to the flash module's internal addressing scheme, and the location within the chip's system memory map requires a clear understanding of the flash memory layout. See the chip-specific information for a detailed explanation of the chip's flash memory layout and how to map system addresses to the flash module's internal addressing.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PEASAD | 0 | | | | | | PEA5 | PEA4 | PEA3 | PEA2 | PEA1 | PEA0 | PEADDR | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31<br>PEASAD | UTest NVM Program and Erase Address<br>Qualifies the address field (PEADDR) to the region.<br>If PESAD = 1, the PEADDR field maps to that region. See the description of the PEADDR field for more information.<br>This field is not writeable if UTE = 1 and is ignored if it becomes 1 while in UTest mode.<br>0b - Address accessed is from the main array space<br>1b - Address accessed is from the UTest NVM array space |
| 30-25<br>— | Reserved<br>Reserved; reset to 0 |
| 24<br>PEA5 | Program and Erase Address Region 5<br>Qualifies the address field (PEADDR) to region 5. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | If PEA5 = 1, the PEADDR field maps to region 5. See the description of the PEADDR field for more information. If region 5 is not present, this field is locked to 0.<br>0b - Address accessed is not from region 5<br>1b - Address accessed is from region 5 |
| 23<br>PEA4 | Program and Erase Address Region 4<br>Qualifies the address field (PEADDR) to region 4.<br>If PEA4 = 1, the PEADDR field maps to region 4. See the description of the PEADDR field for more information. If region 4 is not present, this field is locked to 0.<br>0b - Address accessed is not from region 4<br>1b - Adrress accessed is from region 4 |
| 22<br>PEA3 | Program and Erase Address Region 3<br>Qualifies the address field (PEADDR) to region 3.<br>If PEA3 = 1, the PEADDR field maps to region 3. See the description for PEADDR for more information. If region 3 is not present, this field is locked to 0.<br>0b - Address accessed is not from region 3<br>1b - Address accessed is from region 3 |
| 21<br>PEA2 | Program and Erase Address Region 2<br>Qualifies the address field (PEADDR) to the region.<br>If PEA2 = 1, the PEADDR field maps to region 2. See the description of the PEADDR field for more information. If region 2 is not present, this field is locked to 0.<br>0b - Address accessed is not from region 2<br>1b - Address accessed is from region 2 |
| 20<br>PEA1 | Program and Erase Address Region 1<br>Qualifies the address field (PEADDR) to the region.<br>If PEA1 = 1, the PEADDR field maps to region 1. See the description of the PEADDR field for more information. If region 1 is not present, this field is locked to 0.<br>0b - Address accessed is not from region 1<br>1b - Address accessed is from region 1 |
| 19<br>PEA0 | Program and Erase Address Region 0<br>Qualifies the address field (PEADDR) to the region.<br>If PEA0 = 1, the PEADDR field maps to region 0. See the description of the PEADDR field for more information. If region 0 is not present, this field is locked to 0.<br>0b - Address accessed is not from region 0<br>1b - Address accessed is from region 0 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 18-5<br>PEADDR | Program and Erase Address<br>The PEADDR register provides offset address location to be programmed or the sector to be erased in the case of sector erase. This address is always a quad page address that selects 1024 bits.<br>The PEADDR register is read only in user mode and represents the flash physical address status. PEADDR may be updated once (and only once) per Program or Erase event. PEADDR writes initiate a Program, Erase, Array Integrity or User Margin Read request.<br>Once PEADDR is updated, it will be locked for the full program and erase operation, until MCR[PGM] or MCR[ERS] is cleared at the end of the operation (as long as MCRS[PES] is zero). If MCRS[PES] is set, PEADR is locked until this bit is also cleared.<br>**NOTE**<br>If Express Program has been initiated (XPEADR written), the PEADR is locked from writing until Express Program is completed (XPGM written to 0).<br>Upon completion of an operation (Program (MCR[PGM] written to 0 and MCRS[PES] equals 0), Erase (MCR[ERS] written to 0 and MCRS[PES] equals 0), Array Integrity (UT0[AIE] written to 0), User Margin Read(UT0[AIE] written to 0)), or Program and Erase Sequence clear (MCRS[PES] written to 0) the PEADR registers will return to an all zeros state.<br>When MCR[EHV] or UT0[AIE] are set, PEADDR bits will not be writeable.<br>Once PEADDR is updated, it will have read restrictions based on the PEID master that did the original write in the Platform Flash Controller.<br>PEADDR[18:5] is an offset from a base address of 0h for each block region. The PEA0, PEA1, PEA2, PEA3, PEA4 and PEA5 qualify the block size region the PEADDR field. |
| 4-0<br>— | Reserved<br>Reserved; reset to 0 |

### 20.7.1.8 Sector Program and Erase Hardware Lock (SPELOCK)

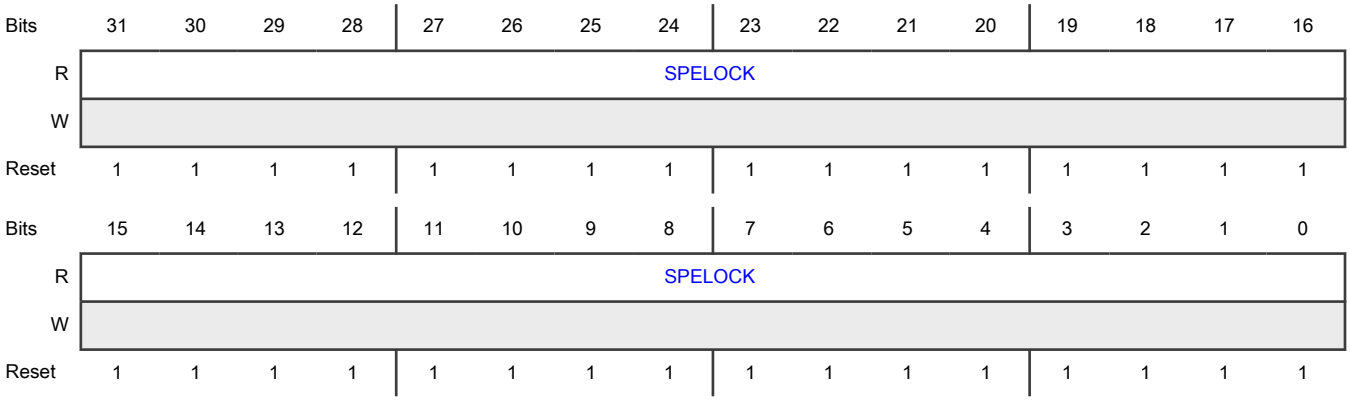
#### Offset

| Register | Offset |
|---|---|
| SPELOCK | 50h |

#### Function

The Sector Program and Erase Lock register provides a means to protect sectors from being programed and erased on the main interface. The last 256 KB of all blocks have sector lock protection capability. The rest of the block (if applicable) is protected with super sector protection capability. This register shows the status of the pelock hardware input for the block that is interlocked. For more information on hardware program and erase protection see Program hardware locking and Erase hardware locking .

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>SPELOCK | Sector Program and Erase Lock [31:0]<br>SPELOCK reflects the status of the hardware program and erase protection input to the flash module for the block interlocked on the main interface.<br>The SPELOCK register is latched on the PEADR write, is not writable, and is status only. SPELOCK will return back to all ones at the same time that PEADR is cleared. When not interlocked this register will read all ones. During high voltage events, the status is locked.<br>The default value of the SPELOCK bits is program and erase protected. In the event that sectors are not present (due to configuration or total memory size), the SPELOCK bits default to locked, and will remain 1.<br>0 - The corresponding sector is not locked, and may be programed or erased.<br>1 - The corresponding sector is protected from program or erase. |

## 20.7.1.9 Super Sector Program and Erase Hardware Lock (SSPELOCK)

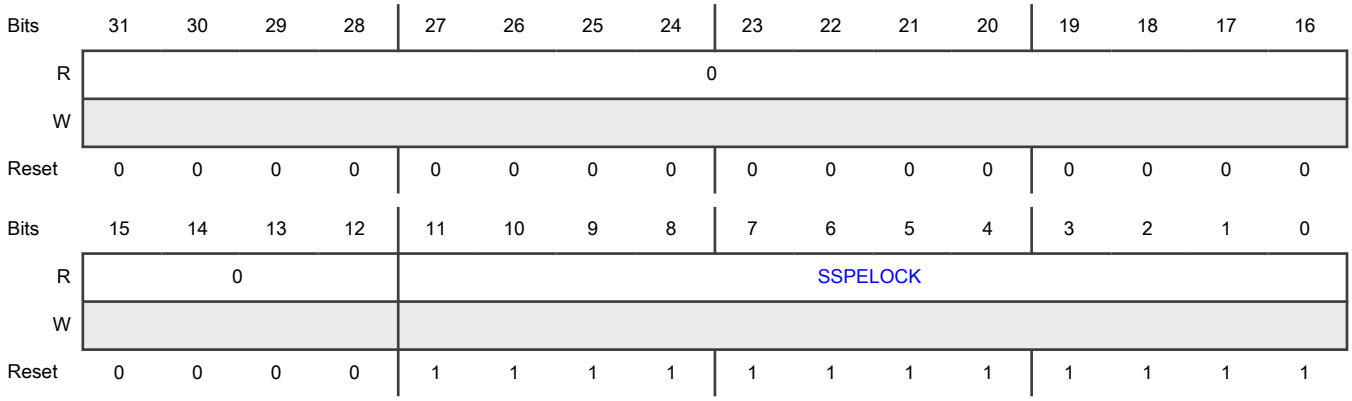
### Offset

| Register | Offset |
|---|---|
| SSPELOCK | 54h |

### Function

The Program and Erase Lock register provides a means to protect super sectors from being programed and erased on the main interface. Super Sector protection is available on block space greater than 256 KB. For 256 KB blocks, this register is not applicable. For 512 KB blocks, the first half of the block is protected with super sector granularity. For 1 MB blocks, the first 768 KB is protected with super sector granularity. For 2 MB blocks, the first 1792 KB is protected with super sector granularity. This register shows the status of the pelock hardware input for the block interlocked. For more information on hardware program and erase protection see Program hardware locking and Erase hardware locking .

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved<br>Reserved; reset to 0 |
| 11-0<br>SSPELOCK | Super Sector Program and Erase Lock [11:0]<br>SSPELOCK reflects the status of the hardware program and erase protection input to the flash module for the block interlocked on the main interface.<br>The SSPELOCK register is latched on the PEADR write, is not writable, and is status only. SSPELOCK will return back to all ones at the same time that PEADR is cleared. When not interlocked this register will read all ones. During high voltage events, the status is locked.<br>The default value of the SSPELOCK bits is program and erase protected. In the event that super sectors are not present (due to configuration or total memory size), or super sectors not present due to block size, the SSPELOCK bits default locked, and will remain 1.<br>0 - The corresponding super sector is not locked, and may be programmed or erased.<br>1 - The corresponding super sector is protected from program or erase. |

## 20.7.1.10 Express Sector Program and Erase Hardware Lock (XSPELOCK)

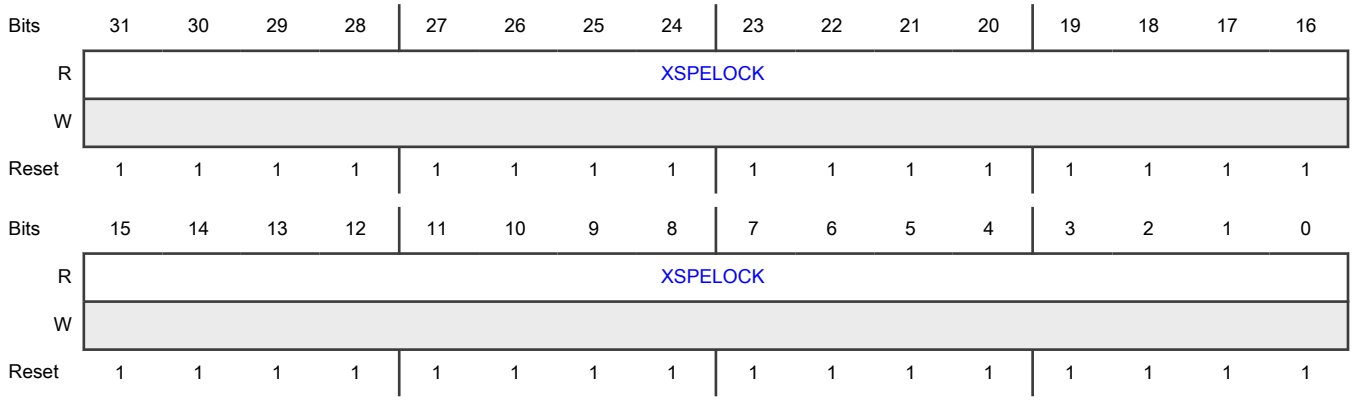
### Offset

| Register | Offset |
|---|---|
| XSPELOCK | 70h |

### Function

The Express Program and Erase Lock register provides a means to protect sectors from being programed and erased on the express program interface. The last 256 KB of all blocks have sector lock protection capability. The rest of the block (if applicable) is protected with super sector protection capability. This register shows the status of the pelock hardware input for the block that is interlocked. For more information on hardware program and erase protection see Program hardware locking and Erase hardware locking .

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>XSPELOCK | Express Sector Program and Erase Lock [31:0]<br>XSPELOCK reflects the status of the hardware program and erase protection input to the flash module for the block interlocked on the express program interface.<br>The XSPELOCK register is latched on the XPEADR write, is not writable, and is status only. XSPELOCK will return back to all ones at the same time that XPEADR is cleared. When not interlocked this register will read all ones. During high voltage events, the status is locked.<br>The default value of the XSPELOCK bits is program and erase protected. In the event that sectors are not present (due to configuration or total memory size), the XSPELOCK bits default to locked, and will remain 1.<br>0 - The corresponding sector is not locked, and may be programmed or erased.<br>1 - The corresponding sector is protected from program or erase. |

## 20.7.1.11 Express Super Sector Program and Erase Hardware Lock (XSSPELOCK)

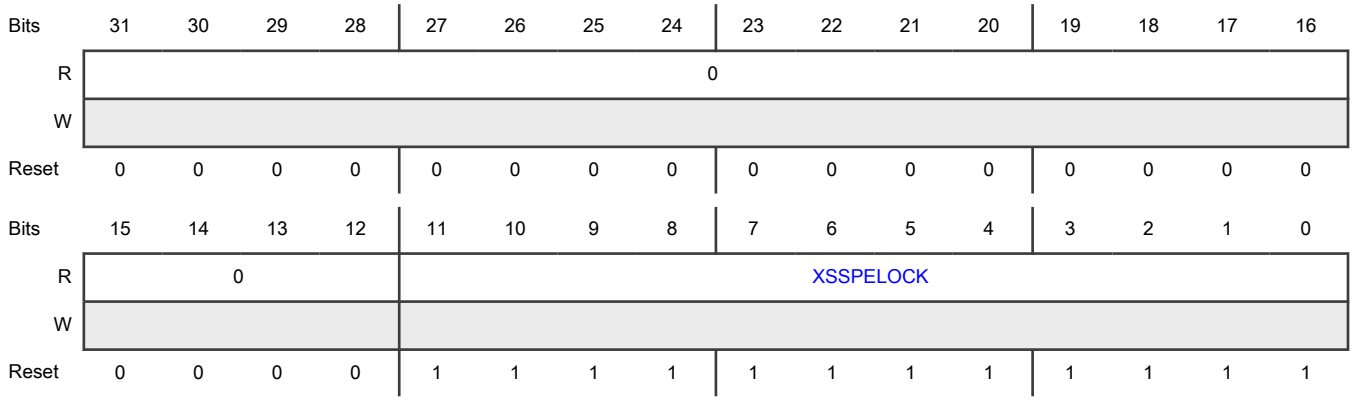
### Offset

| Register | Offset |
|---|---|
| XSSPELOCK | 74h |

### Function

The Express Program and Erase Lock register provides a means to protect super sectors from being programed and erased on the express program interface. Super Sector protection is available on block space greater than 256 KB. For 256 KB blocks, this register is not applicable. For 512 KB blocks, the first half of the block is protected with super sector granularity. For 1 MB blocks, the first 768 KB is protected with super sector granularity. For 2 MB blocks, the first 1792 KB is protected with super sector granularity. This register shows the status of the pelock hardware input for the block interlocked. For more information on hardware program and erase protection see Program hardware locking and Erase hardware locking .

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved<br>Reserved; reset to 0 |
| 11-0<br>XSSPELOCK | Express Super Sector Program and Erase Lock [11:0]<br>XSSPELOCK reflects the status of the hardware program and erase protection input to the flash module for the block interlocked on the express program interface.<br>The XSSPELOCK register is latched on the XPEADR write, is not writable, and is status only. XSSPELOCK will return back to all ones at the same time that XPEADR is cleared. When not interlocked this register will read all ones. During high voltage events, the status is locked.<br>The default value of the XSSPELOCK bits is program and erase protected. In the event that super sectors are not present (due to configuration or total memory size), or super sectors not present due to block size, the XSSPELOCK bits default to locked, and will remain 1.<br>0 - The corresponding super sector is not locked, and may be programmed or erased.<br>1 - The corresponding super sector is protected from program or erase. |

## 20.7.1.12 Test Mode Disable Password Check (TMD)

### Offset

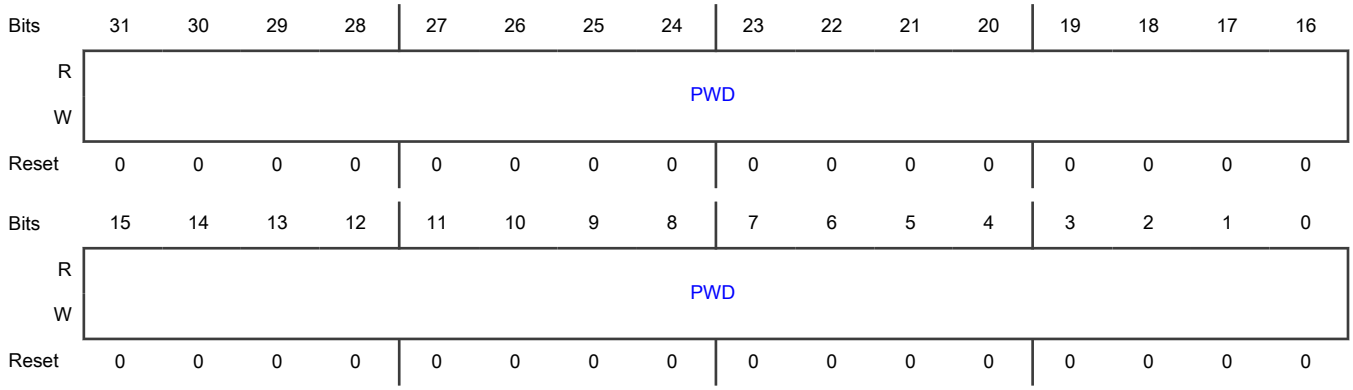
| Register | Offset |
|---|---|
| TMD | 90h |

### Function

Provides a means to supply a challenge password to disable the Test mode disable seal block select.

Writes to the register have no effect except for a password challenge. Reads to this register always return 0.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>PWD | Password Challenge |

## 20.7.1.13 UTest 0 (UT0)

### Offset

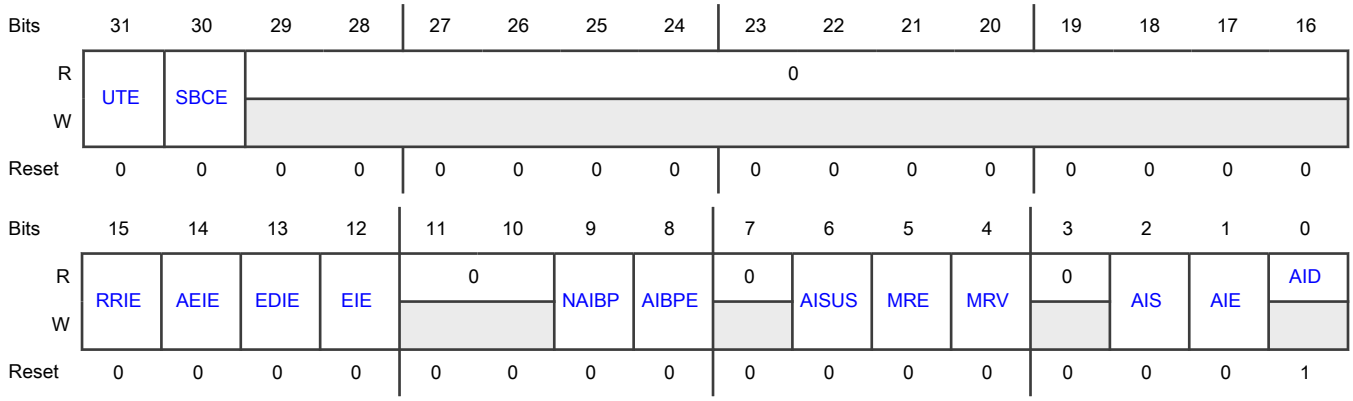
| Register | Offset |
|---|---|
| UT0 | 94h |

### Function

Provides a means to control UTest.

UTest mode provides the ability to perform test features on the flash memory. This register is only writeable when the flash memory is put into UTest mode by writing a passcode.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31<br>UTE | UTest Enable<br>This status bit indicates when U-Test is enabled. All bits in UT0, UM0, UM1, UM2, UM3, UM4, UM5, UM6, UM7, UM8, UM9, UD0, UD1, UD2, UD3, UD4, UD5, UA0 and UA1 registers are locked when this bit is 0. This bit is not writeable to a 1, but may be cleared. The reset value is 0. The method to set this bit is to provide a password, and if the password matches, the UTE bit is set to reflect a status of enabled, and is enabled until it is cleared by a register write. The UTE password (set UTE=1) and UTE clearing to 0 will only be accepted if Program or Erase are not in progress (PEADR write through clearing of PGM/ERS/PES), Alternate Program or Erase are not in progress (APEADR write through clearing of APGM/AERS/APES), AI/MR Operation are not in progress (PEADR write through clearing of AIE), and an Express Program operation is not in progress (XPEADR write through the clearing of XPGM). UTE can only be cleared if AID = 1, MRE, AIE, EIE, EDIE, AEIE, and RRIE are all equal to 0. While successfully clearing UTE, writes to set AIE, MRE, EIE, EDIE, AEIE or RRIE will be ignored. For UTE, the password F9F9_9999h must be written to the UT0 register, and this must be a 32bit write.<br>0b - U-Test mode is not enabled.<br>1b - U-Test mode is enabled. |
| 30<br>SBCE | Single Bit Correction Enable<br>SBCE enables single bit correction results to be observed in SBC. ECC corrections that occur when SBCE is cleared will not be logged.<br>0b - Disabled<br>1b - Enabled |
| 29-16<br>— | Reserved<br>Reserved; reset to 0 |
| 15<br>RRIE | Read Reference Input Enable<br>RRIE enables the force of an error on the read reference detection circuit. This is useful in the Read Reference Error check. If this bit is set, the read reference error register MCRS[RRE] will set when an address match is achieved to the ADR register. RRIE is not simultaneously writable to a 1 as UTE is being successfully cleared to a 0.<br>0b - Read reference input disabled<br>1b - Read reference input enabled |
| 14<br>AEIE | Address Encode Invert Enable<br>AEIE enables the input register AEI to invert the address encode value received from the array, and force a mismatch in the address encode comparison. This is useful in the Address Encode logic check. If this bit is set, address encode information from the memory array will be inverted based on values in the AEI register when an address match is achieved to the ADR register. AEIE is not simultaneously writable to a 1 as UTE is being successfully cleared to a 0.<br>0b - Address encode invert is disabled<br>1b - Address encode values are inverted based on UA0[AEI] |
| 13 | EDC after ECC Data Input Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| EDIE | EDIE enables the input registers EDDATA and EDDATAC to be the source of data to the EDC after ECC comparator or EDC after ECC encoder. This is useful in the EDC after ECC logic check. If this bit is set, data read via a BIU read request will be from the EDDATA and EDDATAC registers when an address match is achieved to the ADR register. EDIE is not simultaneously writable to a 1 as UTE is being successfully cleared to a 0.<br>0b - EDC after ECC data input is disabled<br>1b - Data read is from UD3[EDDATA] and UD5[EDDATAC] |
| 12<br>EIE | ECC Data Input Enable<br>EIE enables the input registers EDATA and EDATAC to be the source of data to the ECC logic (instead of from the memory array). This is useful in the ECC logic check. If this bit is set, data read via a BIU read request will be from the EDATA and EDATAC registers when an address match is achieved to the ADR register. EIE is not simultaneously writable to a 1 as UTE is being successfully cleared to a 0.<br>0b - ECC data input is disabled<br>1b - Data read is from UD0[EDATA] and UD2[EDATAC] |
| 11-10<br>— | Reserved |
| 9<br>NAIBP | Next Array Integrity Break Point<br>If AIBPE is set, NAIBP will be set once a single bit correction (if enabled) or double bit detection is noted during the Array Integrity test. NAIBP is not writable to 1, but may be cleared to 0. Clearing NAIBP to 0, will enable the Array Integrity operation to be re-started after a breakpoint is encountered. NAIPB may only be cleared to 0 if both EER = 0 and SBC = 0. If the Array Integrity operation completes without encountering another correction or detection, AID will be set with NAIBP remaining 0.<br>0b - Array integrity state machine is not currently at a breakpoint<br>1b - Array integrity state machine is at a breakpoint |
| 8<br>AIBPE | Array Integrity Break Point Enable<br>To enable breakpoints during an array integrity test, AIBPE may be set. See NAIBP description for more information about array integrity breakpoints.<br>0b - Array integrity breakpoints disabled<br>1b - Array integrity breakpoints enabled during array integrity checks |
| 7<br>— | Reserved<br>Reserved; reset to 0 |
| 6<br>AISUS | Array Integrity Suspend<br>AISUS enables a suspend of an Array Integrity Operation. Array Integrity may be suspend by setting AISUS, and resumed by clearing AISUS. AISUS is writeable to a 1 only when AID is low. AISUS is clearable to a 0 only when AID is high. Attempting to write AISUS and AIE on the same clock cycle will result in only AIE getting written. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Array integrity sequence not suspended.<br>1b - Array integrity sequence is suspended. |
| 5<br>MRE | Margin Read Enable<br>MRE combined with MRV enables user margin reads to be done. Normal user reads are not affected by MRE, although user reads while the margin read operation is ongoing are not supported. MRE is not writable if AID is low, or if AISUS is high, or if NAIBP is high. MRE is not simultaneously writable to a 1 as UTE is being successfully cleared to a 0.<br>0b - Margin reads are not enabled.<br>1b - Margin reads are enabled. |
| 4<br>MRV | Margin Read Value<br>MRV selects the margin level that is being checked. Margin can be checked to an erased level (MRV=1) or to a programmed level (MRV=0). In order for this value to be valid, MRE must also be set. MRV is not writable if AID is low, or if AISUS is high, or if NAIBP is high.<br>0b - Zero's margin reads are requested.<br>1b - One's margin reads are requested. |
| 3<br>— | Reserved |
| 2<br>AIS | Array Integrity Sequence<br>AIS determines the address sequence to be used during array integrity checks. The default sequence (AIS = 0) is meant to replicate sequences that normal user code follows, and thoroughly checks the read propagation paths. This sequence is proprietary. The alternative sequence (AIS = 1) is logically sequential.<br>It should be noted that the time to run a sequential sequence is shorter than the time to run the proprietary sequence. AIS is not writeable if AIE is high.<br>0b - Array integrity sequence is proprietary sequence<br>1b - Array integrity sequence is sequential |
| 1<br>AIE | Array Integrity Enable<br>AIE set to one starts the array integrity check done on all selected blocks. The address sequence is determined by AIS, and the MISR (UM0 through UM9) can be checked after the operation is complete, to determine if a correct signature has been obtained. Once an array integrity operation is requested (AIE=1), it may be terminated by clearing AIE if the operation has finished (AID = 1) or aborted by clearing AIE if the operation is ongoing (AID = 0). AIE is locked from writing unless an interlock write occurred (PEADR write and DATA write). AIE is not simultaneously writable to a 1 as UTE is being successfully cleared to a 0.<br>0b - Array integrity checks not enabled<br>1b - Array integrity checks enabled |
| 0 | Array Integrity Done |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| AID | AID is cleared upon an array integrity check being enabled (to signify the operation is ongoing). Once completed, AID is set to indicate that the array integrity check is complete. At this time the MISR (UMR registers) can be checked. AID may also assert if breakpoints are enabled (AIBPE is set), an abort is requested or a suspend is requested. AID cannot be written, and is status only.<br>0b - Array integrity check ongoing<br>1b - Array integrity check complete |

### 20.7.1.14 UMISRn (UM0 - UM8)

#### Offset

For a = 0 to 8:

| Register | Offset |
|---|---|
| UMa | 98h + (a × 4h) |

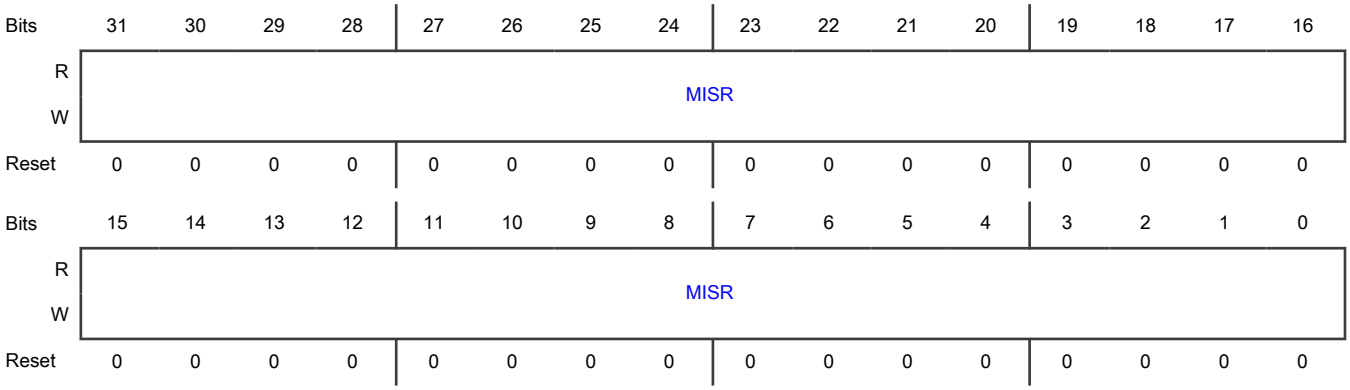
#### Function

Provides a means to evaluate array integrity. UM9 to UM0 are a set of multiple-input signature registers (MISRs) that hold the 289-bit MISR value as shown in the next table.

**Table 92. MISR mapping**

| UM register | MISR bits |
|---|---|
| UM0 | MISR[31:0] |
| UM1 | MISR[63:32] |
| UM2 | MISR[95:64] |
| UM3 | MISR[127:96] |
| UM4 | MISR[159:128] |
| UM5 | MISR[191:160] |
| UM6 | MISR[223:192] |
| UM7 | MISR[255:224] |
| UM8 | MISR[287:256] |
| UM9 | MISR[288] |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>MISR | MISR[31:0]<br>The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required).<br>The MISR can be seeded to any value by writing the MISR registers.<br>**NOTE**<br>Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way. Writing the MISR registers prior to an Array Integrity operation (to seed the MISR) is allowed.<br>The MISR register provides a means to calculate a MISR during array integrity operations.<br>The MISR can be represented by the following polynomial:<br>$x^{289} + x^{7} + x^{6} + x^{5} + x^{4} + x^{2} + 1$<br>The MISR is calculated by taking the previous MISR value and then "exclusive ORing" the new data. In addition the most significant bit (in this case it is MISR[288]), is then "exclusive ORed" into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2] and MISR[0]. The result of the "exclusive OR" is shifted left on each read.<br>The MISR register is used in array integrity operations.<br>If during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size) then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers are not recalculated, and the previous value is retained. Once the AIE register is cleared, the MISR register will return to an all 0's state. |

### 20.7.1.15 UMISR9 (UM9)

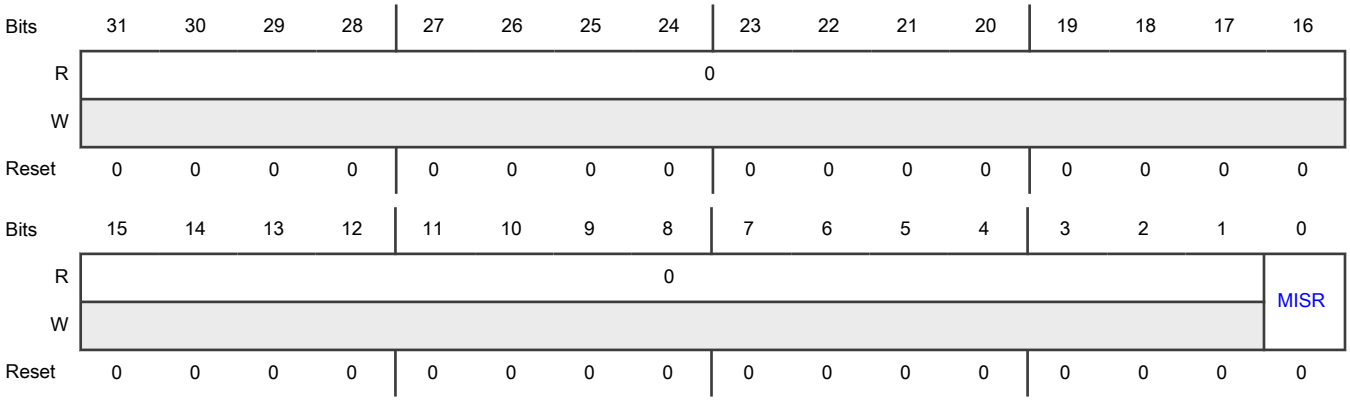
#### Offset

| Register | Offset |
|---|---|
| UM9 | BCh |

#### Function

Provides a means to evaluate array integrity.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved<br>Reserved; reset to 0 |
| 0<br>MISR | MISR[288]<br>The MISR registers accumulate a signature from an array integrity event. The MISR captures all data fields, as well as ECC fields, and the ECC error signal. Data (read and ECC) captured in the MISR is after the ECC logic, and represents corrected data (if required).<br>The MISR can be seeded to any value by writing the MISR registers.<br>**NOTE**<br>Writing the MISR register during an array integrity operation (including suspend or breakpoint) although possible, is not recommended. A write of the MISR registers at this point may alter the final signature in an unpredictable way. Writing the MISR registers prior to an Array Integrity operation (to seed the MISR) is allowed.<br>The MISR register provides a means to calculate a MISR during array integrity operations.<br>The MISR can be represented by the following polynomial:<br>$x^{289} + x^7 + x^6 + x^5 + x^4 + x^2 + 1$ |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | The MISR is calculated by taking the previous MISR value and then "exclusive ORing" the new data. In addition the most significant bit (in this case it is MISR[288]), is then "exclusive ORed" into input of MISR[7], MISR[6], MISR[5], MISR[4], MISR[2] and MISR[0]. The result of the "exclusive OR" is shifted left on each read.<br>The MISR register is used in array integrity operations.<br>If during address sequencing, reads extend into an invalid address location (in other words, greater than the maximum address for a given array size) then reads are still executed to the array, but the results from the array read are not deterministic. In this instance, the MISR registers are not recalculated, and the previous value is retained. Once the AIE register is cleared, the MISR register will return to an all 0's state. |

### 20.7.1.16 UTest Data 0 (UD0)

#### Offset

| Register | Offset |
|---|---|
| UD0 | D0h |

#### Function

Provides a means to enable data to be substituted during UTest procedures.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EDATA | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EDATA | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-0<br>EDATA | ECC Data [31:0]<br>Enables checks of ECC logic by allowing data bits to be input into the ECC logic and then read out by doing array reads or array integrity checks. EDATA corresponds to the 32 array bits representing word 0 of all double words in the page selected in ADR. |

### 20.7.1.17 UTest Data 1 (UD1)

#### Offset

| Register | Offset |
|---|---|
| UD1 | D4h |

#### Function

Provides a means to enable data to be substituted during UTest procedures.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EDATA | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EDATA | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-0<br>EDATA | ECC Data [63:32]<br>Enables checks of ECC logic by allowing data bits to be input into the ECC logic and then read out by doing array reads or array integrity checks. UD0[EDATA] corresponds to the 32 array bits representing word 1 of all double words in the page selected in ADR. |

### 20.7.1.18 UTest Data 2 (UD2)

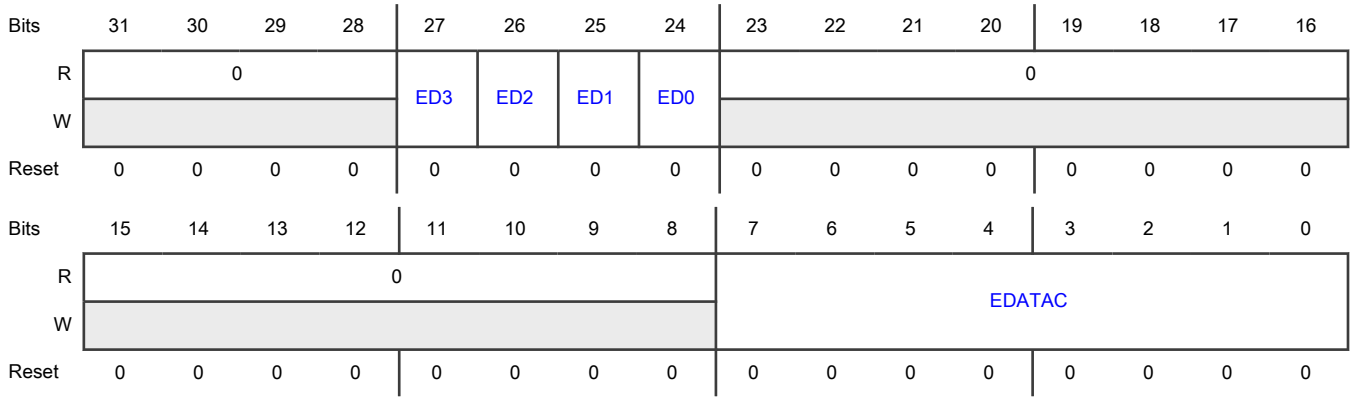
#### Offset

| Register | Offset |
|---|---|
| UD2 | D8h |

#### Function

Provides a means to enable data to be substituted during UTest procedures.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27<br>ED3 | ECC Logic Check Double Word 3<br>Enables checks of ECC logic by allowing data bits to be input into the ECC logic and then read out by doing array reads or array integrity checks.<br>If this field's value becomes 1, the data replacement occurs on DW3 on the page selected in ADR. |
| 26<br>ED2 | ECC Logic Check Double Word 2<br>Enables checks of ECC logic by allowing data bits to be input into the ECC logic and then read out by doing array reads or array integrity checks.<br>If this field's value becomes 1, the data replacement occurs on DW2 on the page selected in ADR. |
| 25<br>ED1 | ECC Logic Check Double Word 1<br>Enables checks of ECC logic by allowing data bits to be input into the ECC logic and then read out by doing array reads or array integrity checks.<br>If this field's value becomes 1, the data replacement occurs on DW1 on the page selected in ADR. |
| 24<br>ED0 | ECC Logic Check Double Word 0<br>Enables checks of ECC logic by allowing data bits to be input into the ECC logic and then read out by doing array reads or array integrity checks.<br>If this field's value becomes 1, the data replacement occurs on DW0 on the page selected in ADR. |
| 23-8<br>— | Reserved |
| 7-0<br>EDATAC | ECC Data Check Bits [7:0]<br>Enables checks of ECC logic by allowing data bits to be input into the ECC logic and then read out by doing array reads or array integrity checks. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | This field corresponds to the eight array bits representing the check bits of all double words in the page selected in ADR. |

### 20.7.1.19 UTest Data 3 (UD3)

#### Offset

| Register | Offset |
|---|---|
| UD3 | DCh |

#### Function

Provides a means to enable data to be substituted during UTest procedures.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EDDATA | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EDDATA | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-0<br>EDDATA | EDC After ECC Data [31:0]<br>Enables checks of EDC after ECC logic by allowing data bits to be input into the EDC after ECC logic and then read out by doing array reads or array integrity checks.<br>This field corresponds to the 32 array bits representing word 0 of all double words in the page selected in ADR. |

### 20.7.1.20 UTest Data 4 (UD4)

#### Offset

| Register | Offset |
|---|---|
| UD4 | E0h |

#### Function

Provides a means to enable data to be substituted during UTest procedures.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EDDATA | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EDDATA | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-0<br>EDDATA | EDC After ECC Data [63:31]<br>Enables checks of EDC after ECC logic by allowing data bits to be input into the EDC after ECC logic and then read out by doing array reads or array integrity checks.<br>This field corresponds to the 32 array bits representing word 1 of all double words in the page selected in ADR. |

### 20.7.1.21 UTest Data 5 (UD5)

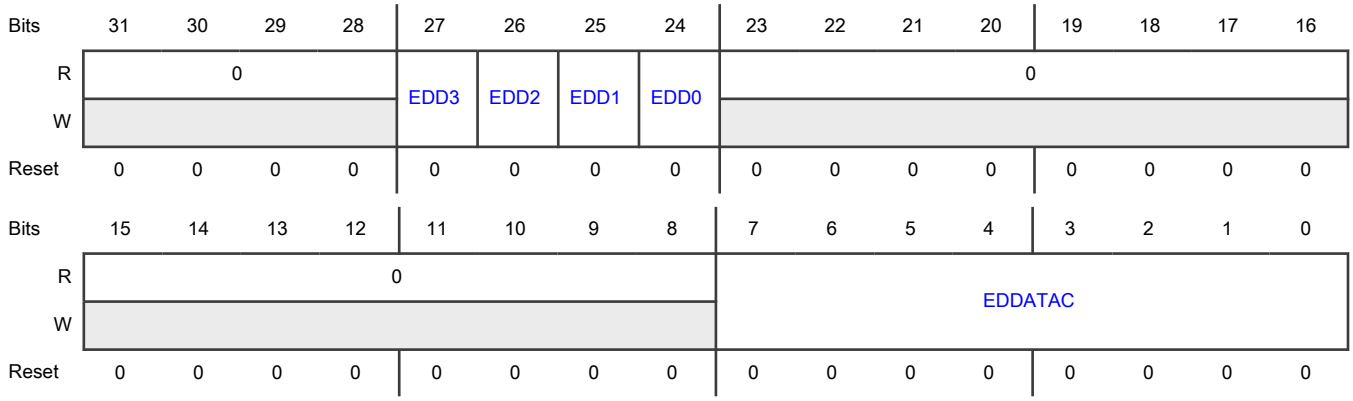
#### Offset

| Register | Offset |
|---|---|
| UD5 | E4h |

#### Function

Provides a means to enable data to be substituted during UTest procedures.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27<br>EDD3 | EDC After ECC Logic Check Double Word 3<br>Enables checks of EDC after ECC logic by allowing data bits to be input into the EDC after ECC logic and then read out by doing array reads or array integrity checks.<br>If this field's value becomes 1, the data replacement occurs on DW3 on the page selected in ADR. |
| 26<br>EDD2 | EDC after ECC Logic Check Double Word 2<br>Enables checks of EDC after ECC logic by allowing data bits to be input into the EDC after ECC logic and then read out by doing array reads or array integrity checks.<br>If this field's value becomes 1, the data replacement occurs on DW2 on the page selected in ADR. |
| 25<br>EDD1 | EDC After ECC Logic Check Double Word 1<br>Enables checks of EDC after ECC logic by allowing data bits to be input into the EDC after ECC logic and then read out by doing array reads or array integrity checks.<br>If this field's value becomes 1, the data replacement occurs on DW1 on the page selected in ADR. |
| 24<br>EDD0 | EDC After ECC Logic Check Double Word 0<br>Enables checks of EDC after ECC logic by allowing data bits to be input into the EDC after ECC logic and then read out by doing array reads or array integrity checks.<br>If this field's value becomes 1, the data replacement occurs on DW0 on the page selected in ADR. |
| 23-8<br>— | Reserved |
| 7-0<br>EDDATAC | EDC After ECC Data Check Bits [7:0]<br>Enables checks of EDC after ECC logic by allowing data bits to be input into the EDC after ECC logic and then read out by doing array reads or array integrity checks. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | This field corresponds to the eight array bits representing the check bits of all double words in the page selected in ADR. |

### 20.7.1.22 UTest Address 0 (UA0)

#### Offset

| Register | Offset |
|---|---|
| UA0 | E8h |

#### Function

Provides a means to enable address to be inverted during UTest procedures.

**NOTE**

- UTest Address 0 (UA0) and UTest Address 1 (UA1) combine to represent a 52-bit wide field, which matches the width of the address internally to the flash memory that is used for the address encode comparison.
- Writing 1 to any one of the fields in UA0 or UA1 leads to flagging an AEE error when causing the error injection.
- When the value of a field within the range of bits 18:3 is 1, the corresponding address signal to the flash memory controller or BIU is inverted.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | AEI | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | AEI | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-0<br>AEI | Address Encode Invert [31:0]<br>Enables checks of address encode logic by allowing address bits to be inverted into the address encode compare logic, forcing a miscompare. Performing array reads on the page selected in ADR allows the address encode invert(s) to be active. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | Additionally, in the event of an address encode error during normal user mode operation, the page address provided to the PFC is inverted when an address encode error is encountered. AEI[18:3], which reflects the page address, can be used with the Address Encode Logic Check to control this inversion feature (aligning with ADR). Any bit in AEI[18:3] written to a 1 also enables the inversion of the corresponding signal of that address, allowing checks to be performed on the comparison logic that exists in the PFC.<br>**NOTE**<br>Multiple bits in AEI[18:3] may be written to a 1, which causes these bits to miscompare on the address encode comparison logic, as well as causing multiple bits to have their inversion enabled as part of the address provided to the BIU. |

### 20.7.1.23 UTest Address 1 (UA1)

#### Offset

| Register | Offset |
|---|---|
| UA1 | ECh |

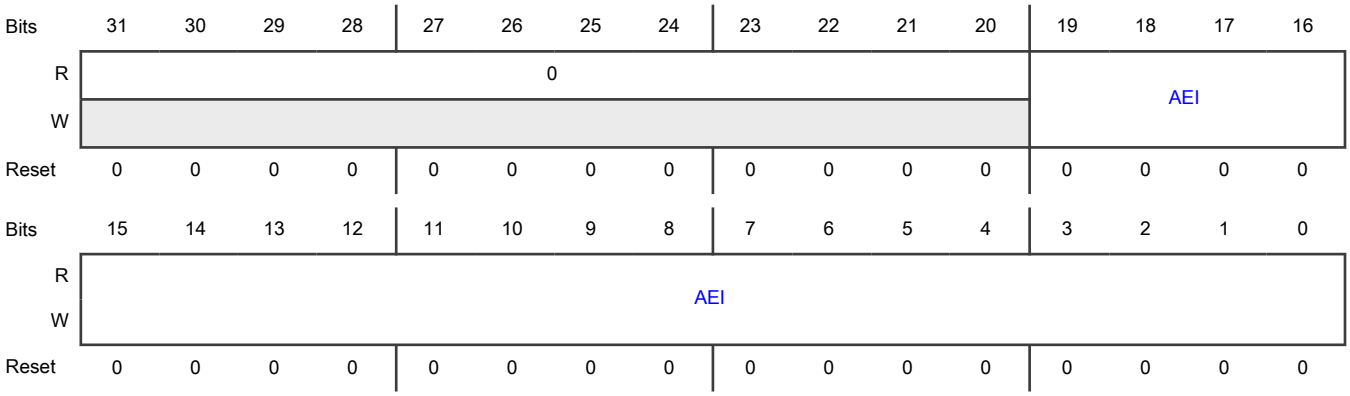
#### Function

Provides a means to enable address to be inverted during UTest procedures.

**NOTE**

- UTest Address 0 (UA0) and UTest Address 1 (UA1) combine to represent a 52-bit wide field, which matches the width of the address internally to the flash that is used for the address encode comparison.
- Writing 1 to any one of the fields in UA0 or UA1 leads to flagging an AEE error when causing the error injection.

#### Diagram



### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19-0<br>AEI | Address Encode Invert [51:32]<br>Enables checks of address encode logic by allowing address bits to be inverted into the address encode compare logic, forcing a miscompare. Performing array reads on the page selected in ADR allows the address encode invert(s) to be active. |

## 20.7.1.24 Express Module Configuration (XMCR)

### Offset

| Register | Offset |
|---|---|
| XMCR | F0h |

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | XPEID | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | XDONE | XPEG | XDOK | XWDI | XWDIE | 0 | XEPEG | XPGM | 0 | | | | | | | XEHV |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16<br>XPEID | Express Program Master/Domain ID<br>This field shows the ID of the master that has started the express program sequence. The ID is latched when the sequence has started (writing of the XPEADR register). It is set back to the reset value (set to all '1') when the operation is completed. See the chip-specific information for a list of Master IDs. |
| 15 | Express State Machine Status |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| XDONE | XDONE indicates whether the embedded flash memory is performing an express program operation. XDONE is set to a 1 on completion of the express program operation.<br>0b - Executing an express program operation<br>1b - Not executing an express program operation |
| 14<br>XPEG | Express Program Good<br>The XPEG field indicates the completion status of the last flash memory express program operation. The value of XPEG is updated automatically after the program operations. XPEG will be 0 if the program operation failed. See MCRS[PEG] for more information, as the features of that bit apply to XPEG as well.<br>0b - Program operation failed<br>1b - Program operation successful |
| 13<br>XDOK | Express Data OK<br>This field is a status signal from the flash to indicate that it is OK to load Data into the DATA registers. It will assert in response to a XPEADR load (once ongoing operations are aborted, if required). XDOK will clear once the Express Program operation is complete (XPGM written to 0).<br>0b - Flash memory not ready to accept writes to the DATA registers<br>1b - Writes to DATA registers allowed |
| 12<br>XWDI | Express Watch Dog Interrupt<br>XWDI is a status register to indicate that the Watchdog Timer for Express Program had expired. XWDI is status only, and will be automatically cleared once the operation that caused the watchdog timeout is terminated or clean up from the operation is completed (clearing of XPGM bit).<br>0b - Normal Operation, Watchdog Timer has not expired.<br>1b - Express Program Watchdog Timer has expired. |
| 11<br>XWDIE | Express Watch Dog Interrupt Enable<br>XWDIE provides a mechanism to trigger an interrupt request upon the assertion of the XWDI bit on the express interface. If XWDIE is asserted, when XWDI asserts an interrupt from the flash will trigger to the system. The interrupt from the flash will mirror exactly the XWDI bit. XWDIE does not affect the register bit XWDI. Writing (and reading) of this bit will be restricted to the master that updated the XPEADR register as captured in the XPEID register at the start of the Express Program sequence.<br>0b - Express watchdog interrupt disabled<br>1b - Express watchdog interrupt enabled |
| 10<br>— | Reserved |
| 9<br>XEPEG | Express ECC Enabled Program Good<br>XEPEG is a qualifier to XPEG to indicate if a passing program required ECC Enabled verifies to pass. In the event of a failed operation (XPEG=0), XEPEG will always remain 0. The value of XEPEG is updated |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | automatically after the program operations. See MCRS[EPEG] for more information, as the features of that bit apply to XEPEG as well.<br>0b - Program operation did not require ECC-enabled verifies<br>1b - Program operation required ECC-enabled verifies to pass |
| 8<br>XPGM | Express Program<br>XPGM is used as part of the setup for an express program operation.<br>A 0 to 1 transition of XPGM after program interlock write(s) is part of the setup for an express program sequence. A 1 to 0 transition of XPGM ends the express program sequence.<br>XPGM can be set at any time if UT0[UTE] is low, and after the interlock write is completed (XPEADR written, XDOK asserted, and DATAn register(s) written).<br>XPGM may be cleared when XEHV is low. Writing (and reading) of this bit will be restricted to the master that updated the XPEADR register as captured in the XPEID register at the start of the Program sequence.<br>0b - Flash memory not executing an express program sequence<br>1b - Flash memory executing an express program sequence |
| 7-1<br>— | Reserved |
| 0<br>XEHV | Express Enable High Voltage<br>The XEHV field enables the embedded flash memory for a high voltage program/erase operation. XEHV is cleared on reset. XEHV must be set after DATA writes to start an express program operation and XPGM is set. This field may be set when XPGM = 1, XDOK = 1, and XWDI = 0. XEHV may not be written to a 1 after it is cleared as part of an express program cleanup, until after the next XPEADR write.<br>Express Program operations may not be aborted. XEHV may only be cleared once XDONE is set to 1. Writing (and reading) of this bit will be restricted to the master that updated the XPEADR register as captured in the XPEID register at the start of the Program sequence.<br>0b - Flash memory is not enabled to perform an express high voltage operation.<br>1b - Flash memory is enabled to perform an express high voltage operation. |

### 20.7.1.25 Express Program Address (XPEADR)

#### Offset

| Register | Offset |
|---|---|
| XPEADR | F4h |

#### Function

The Express Program Address register (XPEADR) is used to provide the address to be programmed. The express program address register is read only in user mode (user mode writes occur with writes done to the Platform Flash Controller). The address is identified using a combination of bits that identify the memory region and offset address.

**NOTE**

- The address given is an offset from the base address of the address space.
- The block numbering scheme that corresponds to the offset address is based on the flash memory's *internal* addressing scheme, which may be different than the chip's system addressing scheme.

Understanding the mapping between the flash address, which is specified relative to the flash module's internal addressing scheme, and the location within the chip's system memory map requires a clear understanding of the flash memory layout. See the chip-specific information for a detailed explanation of the chip's flash memory layout and how to map system addresses to the flash module's internal addressing.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | XPEA5 | XPEA4 | XPEA3 | XPEA2 | XPEA1 | XPEA0 | XPEADDR | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved<br>Reserved; reset to 0 |
| 24<br>XPEA5 | Express Program Address Region 5<br>Qualifies the address field (XPEADDR) to the region.<br>If XPEA5 = 1, the XPEADDR field maps to that region. See the description of the XPEADDR field for more information. If this region is not present, the field is locked to 0.<br>0b - Address accessed is not from region 5<br>1b - Address accessed is from region 5 |
| 23<br>XPEA4 | Express Program Address Region 4<br>Qualifies the address field (XPEADDR) to the region.<br>If XPEA4 = 1, the XPEADDR field maps to that region. See the description of the XPEADDR field for more information. If this region is not present, the field is locked to 0.<br>0b - Address accessed is not from region 4<br>1b - Address accessed is from region 4 |
| 22<br>XPEA3 | Express Program Address Region 3<br>Qualifies the address field (XPEADDR) to the region. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | If XPEA3 = 1, the XPEADDR field maps to that region. See the description of the XPEADDR field for more information. If this region is not present, the field is locked to 0.<br>0b - Address accessed is not from region 3<br>1b - Address accessed is from region 3 |
| 21<br>XPEA2 | Express Program Address Region 2<br>Qualifies the address field (XPEADDR) to the region.<br>If XPEA2 = 1, the XPEADDR field maps to that region. See the description of the XPEADDR field for more information. If this region is not present, the field is locked to 0.<br>0b - Address accessed is not from region 2<br>1b - Address accessed is from region 2 |
| 20<br>XPEA1 | Express Program Address Region 1<br>Qualifies the address field (XPEADDR) to the region.<br>If XPEA1 = 1, the XPEADDR field maps to that region. See the description of the XPEADDR field for more information. If this region is not present, the field is locked to 0.<br>0b - Address accessed is not from region 1<br>1b - Address accessed is from region 1 |
| 19<br>XPEA0 | Express Program Address Region 0<br>Qualifies the address field (XPEADDR) to the region.<br>If XPEA0 = 1, the PEADDR field maps to that region. See the description of the XPEADDR field for more information. If this region is not present, the field is locked to 0.<br>0b - Address accessed is not from region 0<br>1b - Address accessed is from region 0 |
| 18-5<br>XPEADDR | Express Program Address<br>The XPEADDR register provides offset address location to be programmed. This address is always a quad page address that selects 1024 bits.<br>The XPEADDR register is read only in user mode and represents the flash physical address status. XPEADDR may be updated once (and only once) per Express Program event. XPEADDR writes initiate an Express Program request.<br>XPEADDR may be updated four times between reset events. Upon completing the fourth Express Program request (clearing of XPGM register), the XPEADDR register will return to all zeros, and remain locked from future XPEADDR updates and thus future Express Program requests will be ignored until the next reset.<br>Once XPEADDR is updated, it will be locked for the full express program operation, until XMCR[XPGM] is cleared at the end of the operation.<br>Writing of XPEADDR has the effect of blocking other registers used for program and erase to be written (PEADR/MCR registers as well as APEADR/APDATA/AMCR registers) until the express program operation is completed. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Upon completion of an Express Program operation (XMCR[XPGM] written to 0) the XPEADR registers will return to an all zeros state.<br>When UT0[UTE] is set or XMCR[XEHV] is set, XPEADDR bits will not be writeable.<br>Once XPEADDR is updated, it will have read restrictions based on the PEID master that did the original write in the Platform Flash Controller.<br>XPEADDR[18:5] is an offset from a base address of 0h for each block region. The XPEA0, XPEA1, XPEA2, XPEA3, XPEA4 and XPEA5 qualify the block size region the XPEADDR field. |
| 4-0<br>— | Reserved<br>Reserved; reset to 0 |

### 20.7.1.26 Program Data (DATA0 - DATA31)

#### Offset

For a = 0 to 31:

| Register | Offset |
|---|---|
| DATAa | 100h + (a × 4h) |

#### Function

Enables data to be provided for program (or express program). This set of 32 data registers enables 1024 bits of data to be programmed.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PDATA | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PDATA | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### Fields

| Field | Function |
|---|---|
| 31-0 | Program Data |

*Table continues on the next page...*

| Field | Function |
|---|---|
| PDATA | PDATA registers will reset to all ones.<br>PDATA is not writable before PEADR is written or updated to start a program, express program, erase or UTest operation, and will read a value of all ones.<br>Once PEADR is written, all PDATA registers will be made available for writing. Writing (and reading) of the PDATA registers will be restricted to the master that updated the PEADR register as captured in the PEID register.<br>**NOTE**<br>PDATA restrictions related to PEID will be disabled if UT0[UTE] is set.<br>**NOTE**<br>For Express Program a similar behavior exists on the PDATA registers. Once XPEADR is written, and XDOK comes back as a 1, all PDATA registers will switch to all 1's (program safe state) and be made available for writing (and reading) to the master's ID that matches XPEID captured during the XPEADR update. In the event of an interrupted operation, XPEID permissions will override PEID permissions until the express program operation is completed (XPGM written to a 0).<br>Upon completion of an operation (Program (MCR[PGM] written to 0 and MCRS[PES] equals 0), Erase (MCR[ERS] written to 0 and MCRS[PES] equals 0), Express Program (XMCR[XPGM] written to 0), Array Integrity (UT0[AIE] written to 0), User Margin Read(UT0[AIE] written to 0)) or Program and Erase Sequence clear (MCRS[PES] written to 0) the PDATA registers will return to an all ones state.<br>When UT0[AIE] is set, PDATA bits will not be writeable.<br>When MCRS[PES] is set, PDATA bits will not be writeable except when an express program request occurs and XMCR[XDOK] is set.<br>Also, when MCR[PGM] or MCR[ERS] are set, PDATA bits will not be writeable except when an express program request occurs and XMCR[XDOK] is set. When an express program operation is in process, PDATA bits will not be writeable once XMCR[XEHV] is set until XMCR[XPGM] is cleared.<br>Only PDATA Bits that are written will receive program or express program pulses. |

## 20.8 Glossary

**Abort** Premature end of a mode, sequence, state or operation. An abort may leave the embedded flash memory in a state in which it contains indeterminate or corrupted data.

**BIU** Bus interface unit(Contains all system-level customization required to make the embedded flash memory part of an SoC)

**Block** Subdivision of the Flash Module containing NVM memory bits. Block sizes range from 256KB to 2MB. Each block is made up of 8KB sectors.

**Double word** Two words, or 64 bits.

**ECC** Error correction code(Internally used to correct single bit errors, or detect double errors within a 64-bit double word.)

**ECC segment** 64 bit double word, representing the size of data used to calculate ECC information.

**EDC** Error detection code

**FC** Flash memory core

| | |
|---|---|
| **FMU** | Flash Management Unit - Logic that enables program, erase, read, and other flash functions. |
| **Interlock Write** | A write to the PEADR register followed by a write to any DATA register performed while initiating a program, erase, array integrity, or user margin read sequence. Interlock in cleared once the operation is finished by clearing the appropriate control bit, causing PEADR register to clear. |
| **MCU** | Microcontroller unit |
| **Module** | Flash memory instance that contains flash block(s), FMU, and required analog support structures. |
| **NVM** | Nonvolatile memory |
| **OTP** | One time programmable |
| **PFC** | Platform flash memory controller(Contains all system-level customization required to make the embedded flash memory part of an SoC) |
| **Page** | 8 words of data (256 bits). When doing reads to the embedded flash memory, this is the width of data read. Up to 4 pages can be programmed at a time. |
| **Partition** | Subdivision of the Flash Module. Used for read-while-write operation, where addresses in the partition being written may not be read. Partitions is always equal to 1 block, and contains multiple sectors. |
| **RWW** | Read-while-write (operation) |
| **Sector** | Subdivision of the Flash Block that is independently erasable. Sector Size is always 8 KB. |
| **Super sector** | Subdivision of the flash block that includes a group of sectors. Super Sector Size is always 64 KB, and consists of 8 sectors. |
| **SoC** | System-on-chip |
| **UTest NVM sector** | 8 KB sector outside the main address space |
| **Word** | 32 bits |

# Chapter 21
# Flash Memory Controller (PFLASH)

## 21.1 Chip-specific PFLASH information

### 21.1.1 Flash memory architecture

The flash memory on the chip consists of a flash memory controller and a flash memory array module. The flash memory controller provides flash–memory configuration and control functions and manages the interface between the flash memory array and the chip's crossbar switch.

This chip implements three 64-bit AHB buses.

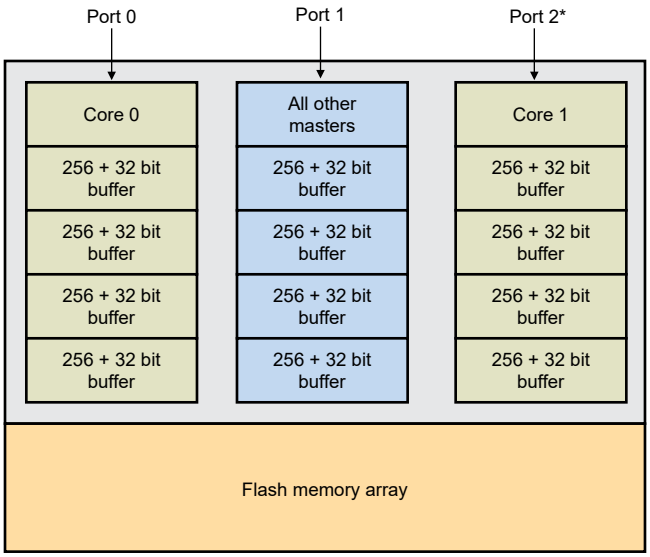



**Figure 46. Flash memory architecture**

### 21.1.2 Flash memory controller

The flash memory controller:

- Acts as an interface between the system bus and flash memory array.
- Is a triple-ported controller with a dedicated line buffering per port and per master ID. This enables you to use the line buffers more efficiently because various masters have dedicated buffers that are not compromised when other masters perform read operations.

Also, by having separate ports, you can have separate connections for each CPU instruction bus and a single port for all data accesses as shown in Figure 46.

In general, the flash memory controller registers affect the global flash memory behavior (for example, read buffering and access control).

**NOTE**

In MCXE317:

- PFLASH block 3 and block 4 are not present, so both the sector and super sector registers are not available.
- For PFLASH block 2, the super sector registers are not available.

### 21.1.3 Platform flash configuration registers (PFCR*n*)

The table below defines the PFCR*n* fields that the chip uses.

**Table 93. Platform flash configuration registers (PFCR*n*)**

| Master | Buffer enable field | Prefetch enable field |
|---|---|---|
| Cortex–M7_0 | P0_CBFEN, P0_DBFEN | P0_CPFEN, P0_DPFEN |
| eDMA + ELE_HSEB + EMAC[1] | P1_CBFEN, P1_DBFEN | P1_CPFEN, P1_DPFEN |

1. MCXE315/MCXE316 and MCXE317 do not support EMAC.

### 21.1.4 Platform flash access protection register (PFAPR)

This table defines the PFAPR[M*n*AP] fields that the chip uses. This chip does not use the other master access protection fields, but those fields are readable and writable.

**Table 94. Platform flash access protection register (PFAPR)**

| Master number | Master name | Access protection field |
|---|---|---|
| 0 | Cortex–M7_0 | M0AP |
| 2 | eDMA | M2AP |
| 3 | ELE_HSEB | M3AP |
| 4 | EMAC | M4AP |
| 5 | Reserved | |
| 6 | | |
| 7 | | |

## 21.2 Overview

PFLASH acts as an interface between the system bus (AHB-Lite 2.v6) and flash memory array.

PFLASH supports three 64-bit AHB buses and a 256-bit read data interface from each flash memory array. The slave port assignments and buffer organization are organized to offer maximum performance of code execution in a multicore architecture. The buffer mechanism serves to deliver flash memory read data with zero-wait state response on lines that reside in cache. AHB requests that miss the prefetch cache generate the needed flash memory array access and are forwarded to the AHB upon completion. Each cache entry is 256 bits, matching the flash memory array page size and providing 512 bytes of high-speed local storage.

### 21.2.1 Block diagrams

The following figure provides a block diagram showing PFLASH and the attached flash memory array.

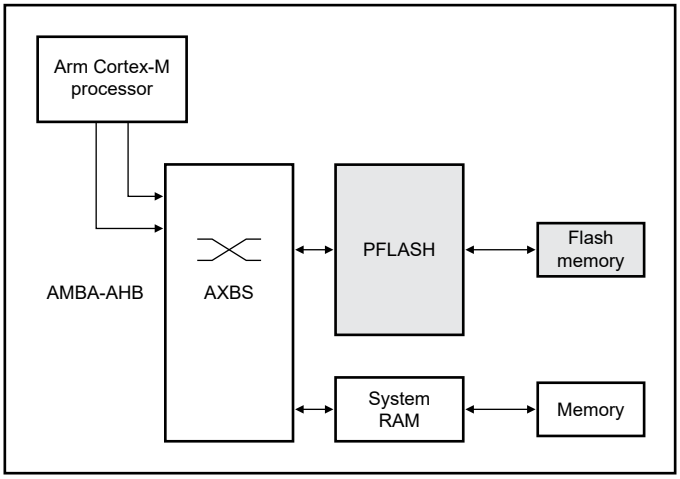



Figure 47. Platform-centric simple block diagram with PFLASH

### 21.2.2 Features

- Two 64-bit AHB interface ports (p0, p1) allowing simultaneous access to dedicated prefetch mini-cache per slave port
- 256-bit read data bus and 64-bit write data bus
- Configurable read buffering and line prefetching support via a mini-cache, plus a prefetch controller for each AHB port to provide single-cycle buffer hit read response
- Configurable access control based on read/write and AHB master ID attributes
- Support for reporting single-bit and multi-bit flash memory ECC events on a 64-bit doubleword boundary

## 21.3 Functional description

As shown in Figure 47, PFLASH interfaces between:

- The AHB system bus ports
- The flash memory array

For accesses targeting flash memory, the PFLASH generates as inputs to the flash memory array:

- Read and write enables
- Block selects
- Array address
- Write size
- Write data

PFLASH captures read data from the flash memory array and drives it onto the AHB system bus. Up to four pages of data (256-bit page size) may be buffered in each prefetch buffer for AHB Port0 and Port1 . Lines may be prefetched in advance of being requested, allowing single-cycle (zero AHB wait-states) read data responses on buffer hits.

Access protections may be applied on a per-master basis for both reads and writes to support security and privilege mechanisms.

### 21.3.1 Read transactions

On an incoming AHB read request, a mini-cache lookup and access privilege evaluation are performed during the AHB address phase. If a buffer hit occurs, the requested data is retrieved from the previously loaded prefetch buffer entry and returned on the system bus with a zero wait-state response. If a buffer miss occurs, a flash memory access is initiated.

Read accesses are terminated under control of the appropriate wait state settings. Access timing can be varied to account for the operating conditions of the chip (for example, frequency, voltage, temperature, and so on) by appropriately setting the read wait state field in flash memory.

### 21.3.2 Write transactions

An interlock write on a program or erase sequence is initiated by first writing to Platform Flash Memory Program Erase Address Logical (PFCPGM_PEADR_L), Platform Flash Memory Express Program Erase Address Logical (PFCPGM_XPEADR_L)(see the Flash Memory chapter for write sequence details).

### 21.3.3 Access protections

#### 21.3.3.1 PFAPR

PFLASH provides programmable, configurable access protections for read cycles on a per-master basis in PFAPR[M*n*AP]. This field restricts read access on a per-master basis. This functionality is described in Platform Flash Memory Access Protection (PFAPR). Detection of a protection violation based on PFAPR settings results in an error response from PFLASH on the AHB transfer.

### 21.3.4 Error termination

PFLASH can invoke a system bus error termination in the following scenarios:

- Access privilege violation (see Access protections for details)
- Attempted access by an AHB master to a reserved region in the flash memory map
- Multi-bit ECC error detection on AHB read, and PFCR4[DMEEE] = 0 (for data flash memory, it is further qualified by PFCR4[DERR_SUP] = 0)

### 21.3.5 Line read buffers and prefetch operation

The PFLASH AHB ports of the each contain a mini-cache. PFLASH uses the buffers for both prefetch and normal demand fetches. Also, the buffers are shared for code and data fetches, and can be controlled independently for code and data from control registers.

Prefetch triggering is controllable on a per-port basis. A PFLASH read access may trigger a prefetch to the next sequential line of array data on the cycle following the request. The access address is incremented by 32 bytes, and a subsequent flash memory access is initiated. A flash memory array prefetch is initiated if the data is not already resident in a line read buffer. Prefetched data is always loaded into the least-recently-used buffer.

For Port0 and Port1 there are four line buffer entries in their respective prefetch mini-caches that follow a fully associative, least-recently-used replacement policy.

For prefetching to occur, you must set the following configuration fields, where *n* corresponds to the port number and *m* corresponds to *C* (code) or *D* (data):

PFCRn[Pn_mBFEN] = 1 and PFCRn[Pn_mPFEN] = 1

### 21.3.6 Array integrity considerations

During an array integrity sequence, the flash memory array ignores any incoming read requests. When a flash memory array integrity check is in progress, PFLASH terminates all flash memory access requests with an error. More specifically, it aborts the incoming flash memory access requests and terminates the system bus transfer with an error.

### 21.3.7 Safety considerations

#### 21.3.7.1 Flash memory address generation check

Functional safety coverage of the address path and control within PFLASH rely on a feedback path between PFLASH and flash memory. Remember that on a requested access to flash memory, PFLASH must decode the system AHB bus signals to generate the corresponding flash memory interface signals to invoke a flash memory lookup. In addition to providing the requested read data, the flash memory also provides output sidebands reflecting the encoded address and block selects used to perform the actual row lookup.

PFLASH uses this sideband information to verify the expected transaction. If a mismatch is detected, indicating a failure in the address generation or control logic within PFLASH or the transmission path between PFLASH and the flash memory array, then the event is forwarded to the chip fault collection module and the corresponding buffer is invalidated.

### 21.3.8 ECC error handling on data flash block

When PFCR4[DERR_SUP] is enabled, ECC errors on data flash blocks are handled specially.

If there is a noncorrectable error detection, a fixed, illegal opcode value is returned to the requesting master along with the associated ECC checkbits as determined by the requesting address.

For noncorrectable error detection, PFLASH returns a value of 1555_1555h to the requesting master.

This is mainly used for EEPROM emulation applications.

### 21.3.9 Read cycles—buffer miss

On an incoming AHB read request, a mini-cache lookup and access privilege evaluation are performed during the AHB address phase. If a buffer miss occurs, a flash memory access is initiated.

If the flash memory access was the direct result of an AHB transaction, the corresponding page buffer is loaded and marked as the most-recently-used. If the flash memory access was the result of a speculative prefetch to the next sequential line, it is loaded into the least-recently-used buffer. The status of this buffer is not changed to most-recently-used until a subsequent buffer hit occurs as a result of an AHB read request.

### 21.3.10 Read cycles—buffer hit

PFLASH allows single-cycle read responses to the AHB when the requested read access was previously loaded into one of the page buffers. In these cases of a buffer hit, read data is returned on the system bus with a zero wait-state response.

### 21.3.11 Flash memory error response operation

The flash memory array may signal an error response to terminate a requested access because of improper sequencing during program/erase operations and improper sequencing during array integrity testing. When an error response is received, PFLASH does not update or validate a page read buffer. An error response may be signaled on a read or interlock write operation. For more information on the specifics related to signaling of flash memory errors, including flash memory ECC events, array integrity testing, and read-while-write events, see the flash memory chapter.

### 21.3.12 Clocking

This module has no clocking considerations.

### 21.3.13 Interrupts

This module has no interrupts.

## 21.4 External signals

This module has no external signals.

## 21.5 PFLASH register descriptions

PFLASH provides an IPS programming model mapped to a standard 16 KB on-platform peripheral slot. The programming model consists of flash memory access configuration.

You can reference the programming model only by using a 32-bit (word) access. References that are attempted using different access sizes, or to undefined (reserved) addresses, or in User mode generate an IPS error termination. PFLASH allows access to the programming model by all system bus masters.

Write to read only registers don't generate error termination.

You can only access the programming model in Supervisor mode, except *PEADR* registers which can be accessed in Supervisor or User mode.

Attempted updates to the programming model when PFLASH is in the middle of an operation results in non-deterministic behavior. You must architect software to avoid this scenario. The recommended flow for multicore devices is:

1. Start only one core.
2. Execute initialization code until it is complete.
3. Start the remaining cores.

If you need to reconfigure the flash memory, code execution must be temporarily moved to system RAM.

### 21.5.1 PFLASH memory map

PFLASH base address: 4026_8000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Platform Flash Memory Configuration 0 (PFCR0) | 32 | RW | 0000_0003h |
| 4h | Platform Flash Memory Configuration 1 (PFCR1) | 32 | RW | 0000_0003h |
| 10h | Platform Flash Memory Configuration 4 (PFCR4) | 32 | RW | 0000_0000h |
| 14h | Platform Flash Memory Access Protection (PFAPR) | 32 | RW | FFFF_FFFFh |
| 300h | Platform Flash Memory Program Erase Address Logical (PFCPGM_PEADR_L) | 32 | RW | 0000_0000h |
| 304h | Platform Flash Memory Program Erase Address Physical (PFCPGM_PEADR_P) | 32 | R | 0000_0000h |
| 308h | Platform Flash Memory Express Program Erase Address Logical (PFCPGM_XPEADR_L) | 32 | RW | 0000_0000h |
| 30Ch | Platform Flash Memory Express Program Erase Address Physical (PFCPGM_XPEADR_P) | 32 | R | 0000_0000h |
| 340h - 350h | Block n Sector Program Erase Lock (PFCBLK0_SPELOCK - PFCBLK4_SPELOCK) | 32 | RW | FFFF_FFFFh |
| 358h | Block UTEST Sector Program Erase Lock (PFCBLKU_SPELOCK) | 32 | RW | 0000_0001h |
| 35Ch - 368h | Block n Super Sector Program Erase Lock (PFCBLK0_SSPELOCK - PFCBLK3_SSPELOCK) | 32 | RW | 0000_0FFFh |
| 380h - 390h | Block n Set Sector Lock (PFCBLK0_SETSLOCK - PFCBLK4_SETSLOCK) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 398h | Block UTEST Set Sector Lock (PFCBLKU_SETSLOCK) | 32 | RW | 0000_0000h |
| 39Ch - 3A8h | Block n Set Super Sector Lock (PFCBLK0_SSETSLOCK - PFCBLK3_SSETSLOCK) | 32 | RW | 0000_0000h |
| 3C0h - 45Ch | Block a Lock Master Sector b (PFCBLK0_LOCKMASTER_S0 - PFCBLK4_LOCKMASTER_S7) | 32 | R | FFFF_FFFFh |
| 480h | Block UTEST Lock Master Sector (PFCBLKU_LOCKMASTER_S) | 32 | R | 0000_00FFh |
| 484h | Block m Lock Master Super Sector n (PFCBLK0_LOCKMASTER_SS0) | 32 | R | FFFF_FFFFh |
| 488h | Block m Lock Master Super Sector n (PFCBLK0_LOCKMASTER_SS1) | 32 | R | FFFF_FFFFh |
| 48Ch | Block m Lock Master Super Sector n (PFCBLK0_LOCKMASTER_SS2) | 32 | R | FFFF_FFFFh |
| 494h | Block m Lock Master Super Sector n (PFCBLK1_LOCKMASTER_SS0) | 32 | R | FFFF_FFFFh |
| 498h | Block m Lock Master Super Sector n (PFCBLK1_LOCKMASTER_SS1) | 32 | R | FFFF_FFFFh |
| 49Ch | Block m Lock Master Super Sector n (PFCBLK1_LOCKMASTER_SS2) | 32 | R | FFFF_FFFFh |
| 4A4h | Block m Lock Master Super Sector n (PFCBLK2_LOCKMASTER_SS0) | 32 | R | FFFF_FFFFh |
| 4A8h | Block m Lock Master Super Sector n (PFCBLK2_LOCKMASTER_SS1) | 32 | R | FFFF_FFFFh |
| 4ACh | Block m Lock Master Super Sector n (PFCBLK2_LOCKMASTER_SS2) | 32 | R | FFFF_FFFFh |
| 4B4h | Block m Lock Master Super Sector n (PFCBLK3_LOCKMASTER_SS0) | 32 | R | FFFF_FFFFh |
| 4B8h | Block m Lock Master Super Sector n (PFCBLK3_LOCKMASTER_SS1) | 32 | R | FFFF_FFFFh |
| 4BCh | Block m Lock Master Super Sector n (PFCBLK3_LOCKMASTER_SS2) | 32 | R | FFFF_FFFFh |

### 21.5.2 Platform Flash Memory Configuration 0 (PFCR0)

**Offset**

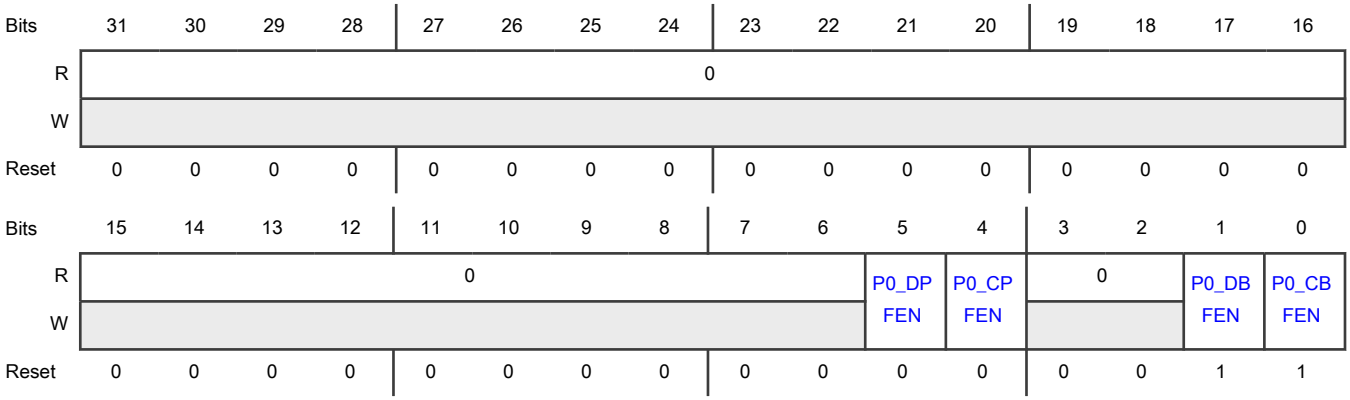
| Register | Offset |
|---|---|
| PFCR0 | 0h |

### Function

Specifies the operation of PFLASH Port0.

See the chip-specific PFLASH information for details about the actual masters available on the chip.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved |
| 5<br>P0_DPFEN | Port0 Data Prefetch Enable<br>Enables or disables data prefetching initiated by a read access on Port0. Prefetching can only be enabled if the buffers are enabled by writing 1 to DBFEN. Hardware reset returns this field to 0.<br>0b - Disable<br>1b - Enable |
| 4<br>P0_CPFEN | Port0 Code Prefetch Enable<br>Enables or disables code prefetching initiated by a read access on Port0. Prefetching can only be enabled if the buffers are enabled by writing 1 to CBFEN. Hardware reset returns this field to 0.<br>0b - Disable<br>1b - Enable |
| 3-2<br>— | Reserved |
| 1<br>P0_DBFEN | Port0 PFLASH Line Read Data Buffers Enable<br>Enables or disables line read data buffer hits. It is also used to invalidate the buffers.<br>If this field is 0, the line read buffers are disabled from satisfying read requests, and all buffer valid bits are set to 0. If this field is enabled, the line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disable<br>1b - Enable |
| 0<br>P0_CBFEN | Port0 PFLASH Line Read Code Buffers Enable<br>Enables or disables line read code buffer hits. It is also used to invalidate the buffers.<br>If disabled, the line read buffers are disabled from satisfying read requests, and all buffer valid bits are set to 0. If enabled, the line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.<br>0b - Disable<br>1b - Enable |

## 21.5.3 Platform Flash Memory Configuration 1 (PFCR1)

### Offset

| Register | Offset |
|---|---|
| PFCR1 | 4h |

### Function

Specifies the operation of PFLASH Port1.

See the chip-specific PFLASH information for details about the actual masters available on the chip.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | P1_DPFEN | P1_CPFEN | 0 | | P1_DBFEN | P1_CBFEN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

### Fields

| Field | Function |
|---|---|
| 31-6 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 5<br>P1_DPFEN | Port1 Data Prefetch Enable<br>Enables or disables data prefetching initiated by a read access on Port1. Prefetching can only be enabled if the buffers are enabled by writing 1 to DBFEN. Hardware reset returns this field to 0.<br>0b - Disable<br>1b - Enable |
| 4<br>P1_CPFEN | Port1 Code Prefetch Enable<br>Enables or disables code prefetching initiated by a read access on Port1. Prefetching can only be enabled if the buffers are enabled by writing 1 to CBFEN. Hardware reset returns this field to 0.<br>0b - Disable<br>1b - Enable |
| 3-2<br>— | Reserved |
| 1<br>P1_DBFEN | Port1 PFLASH Line Read Data Buffers Enable<br>Enables or disables line read data buffer hits. It is also used to invalidate the buffers.<br>If this field is 0, the line read buffers are disabled from satisfying read requests, and all buffer valid bits are set to 0. If this field is enabled, the line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.<br>0b - Disable<br>1b - Enable |
| 0<br>P1_CBFEN | Port1 PFLASH Line Read Code Buffers Enable<br>Enables or disables line read code buffer hits. It is also used to invalidate the buffers.<br>If disabled, the line read buffers are disabled from satisfying read requests, and all buffer valid bits are set to 0. If enabled, the line read buffers are enabled to satisfy read requests on hits. Buffer valid bits may be set when the buffers are successfully filled.<br>0b - Disable<br>1b - Enable |

### 21.5.4 Platform Flash Memory Configuration 4 (PFCR4)

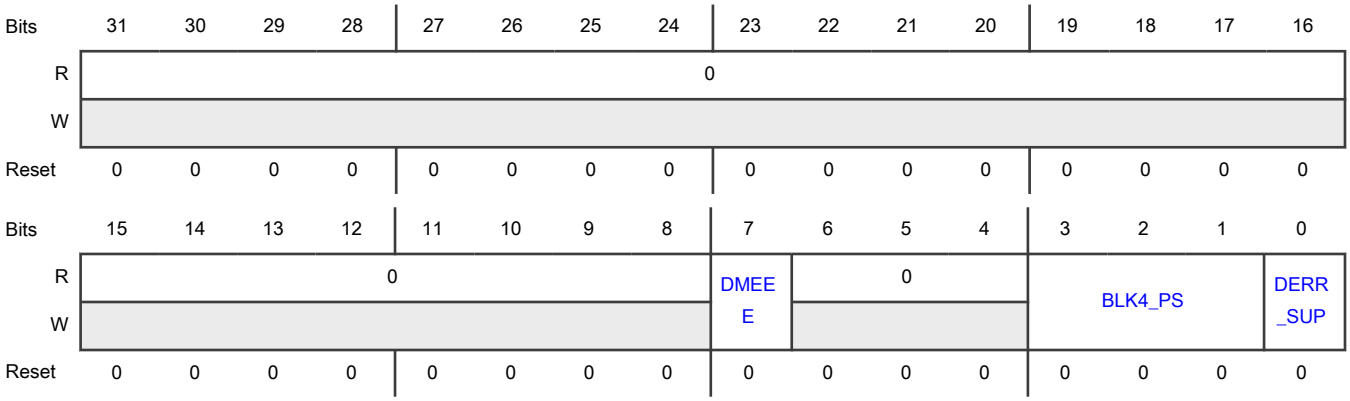
**Offset**

| Register | Offset |
|---|---|
| PFCR4 | 10h |

### Function

Specifies operation of the flash memory controller buffer.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7<br>DMEEE | Disable Multi-Bit ECC Error Exception<br>Enables or disables system bus error response on multi-bit ECC error. Hardware reset returns this field to 0.<br>0b - Error response sent on system bus for multi-bit ECC error<br>1b - Error response not sent on system bus for multi-bit ECC error |
| 6-4<br>— | Reserved |
| 3-1<br>BLK4_PS | Block 4 Pipe Select<br>Selects the active pipe for flash memory block 4 access.<br>PFLASH has four independent command pipes to issue four parallel read commands to different flash memory blocks. Reads from flash memory block 0–3 are always done through command pipe 0–3, respectively. However, the access to block 4 is not fixed and can be through any of the command pipes. You must only change this field when there is no ongoing block 4 access.<br>A special round-robin arbitration scheme snoops the availability of a command pipe during block 4 access. If any of the command pipes are idle during the first read request to block 4, the ownership of that command pipe gets shared between block 4 and the respective block. If none of the command pipes are idle during a block 4 read request, block 4 gets associated with each of the command pipes in round-robin fashion. When a command pipe acquires ownership of block 4, it keeps that ownership until all the commands to block 4 from all the masters are served.<br>000b - Block 4 access is always through pipe0<br>001b - Block 4 access is always through pipe1 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 010b - Block 4 access is always through pipe2<br>011b - Block 4 access is always through pipe3<br>1xxb - Block 4 access can be through any of the command pipes, based on which command pipe is available for block 4 access |
| 0<br>DERR_SUP | Data Error Suppression<br>See the Embedded Flash Memory configuration information or system memory map for which flash memory blocks are affected by this field.<br>0b - Reports ECC events on data flash memory accesses<br>1b - Single-bit and multi-bit ECC events on data flash memory accesses are suppressed |

## 21.5.5 Platform Flash Memory Access Protection (PFAPR)

### Offset

| Register | Offset |
|---|---|
| PFAPR | 14h |

### Function

Controls read accesses to the flash memory array.

See the chip-specific PFLASH information for details about the actual masters available on the chip.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | M0AP | | M1AP | | M2AP | | M3AP | | M4AP | | M5AP | | M6AP | | M7AP | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | M8AP | | M9AP | | M10AP | | M11AP | | M12AP | | M13AP | | M14AP | | M15AP | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### Fields

| Field | Function |
|---|---|
| 31-30: M0AP | Master n Access Protection |

*Table continues on the next page...*

| Field | Function |
|---|---|
| 29-28: M1AP<br>27-26: M2AP<br>25-24: M3AP<br>23-22: M4AP<br>21-20: M5AP<br>19-18: M6AP<br>17-16: M7AP<br>15-14: M8AP<br>13-12: M9AP<br>11-10: M10AP<br>9-8: M11AP<br>7-6: M12AP<br>5-4: M13AP<br>3-2: M14AP<br>1-0: M15AP | Controls whether read accesses to the flash memory are allowed based on the master ID of a requesting master. These fields are initialized by hardware reset. The field M3AP is reserved.<br>x0b - This master cannot perform any read accesses<br>x1b - This master can perform read accesses |

### 21.5.6 Platform Flash Memory Program Erase Address Logical (PFCPGM_PEADR_L)

#### Offset

| Register | Offset |
|---|---|
| PFCPGM_PEADR_L | 300h |

#### Function

Provides the flash memory address to be programmed, or the location of the sector or block to be erased through main flash memory (pgm/erase) queue. Write access to this register is domain-ID aware.

The respective bus master must have program/erase permission to the flash memory address written to this register. Otherwise a transfer error results. For further information on flash memory address restrictions see the XRDC chapter.

A write to this register is managed via three-cycle access. Before updating the register, you must ensure that no ongoing high-voltage operation is executing through the flash memory main queue.

Unauthorized flash memory address writes result in a transfer error.

Writes to this register during an ongoing high-voltage operation (initiated through the flash memory main queue) or during express program operation are ignored.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>PEADR_L | Program Erase Address Logical<br>Contains the system logical address for flash memory program/erase. |

## 21.5.7 Platform Flash Memory Program Erase Address Physical (PFCPGM_PEADR_P)

### Offset

| Register | Offset |
|---|---|
| PFCPGM_PEADR_P | 304h |

### Function

Reflects the flash memory block number and offset address corresponding to Platform Flash Memory Program Erase Address Logical (PFCPGM_PEADR_L). This register has the same format as the PEADR register in the Flash Memory chapter—see it for details.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>PEADR_P | Program Erase Address Physical<br>Contains the flash block select and offset address for flash memory program/erase. |

## 21.5.8 Platform Flash Memory Express Program Erase Address Logical (PFCPGM_XPEADR_L)

### Offset

| Register | Offset |
|---|---|
| PFCPGM_XPEADR_L | 308h |

### Function

Provides the flash memory address to be programmed using the flash memory express program feature. Write access to this register is domain-ID aware.

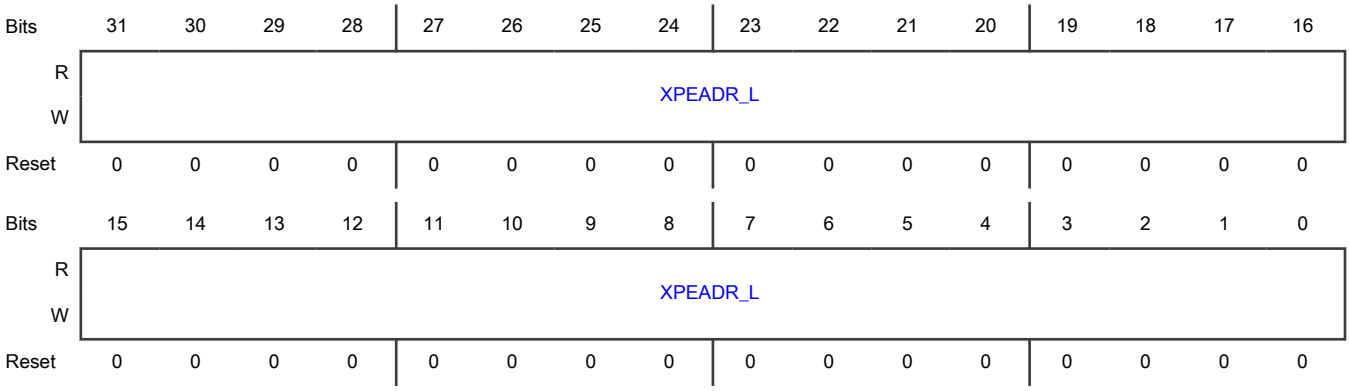
The respective bus master must have program/erase permission to the flash memory address written to this register. Otherwise a transfer error results. See the XRDC chapter for further information on flash memory address restrictions.

A write to this register is managed via three-cycle access. Before updating the register, you must ensure that no ongoing high-voltage operation is executing through the flash memory main queue.

Unauthorized flash memory address writes result in a transfer error.

Writes to this register during an ongoing express program operation are ignored.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>XPEADR_L | Express Program Erase Address Logical<br>Contains the system logical address for express flash program/erase. |

## 21.5.9 Platform Flash Memory Express Program Erase Address Physical (PFCPGM_XPEADR_P)

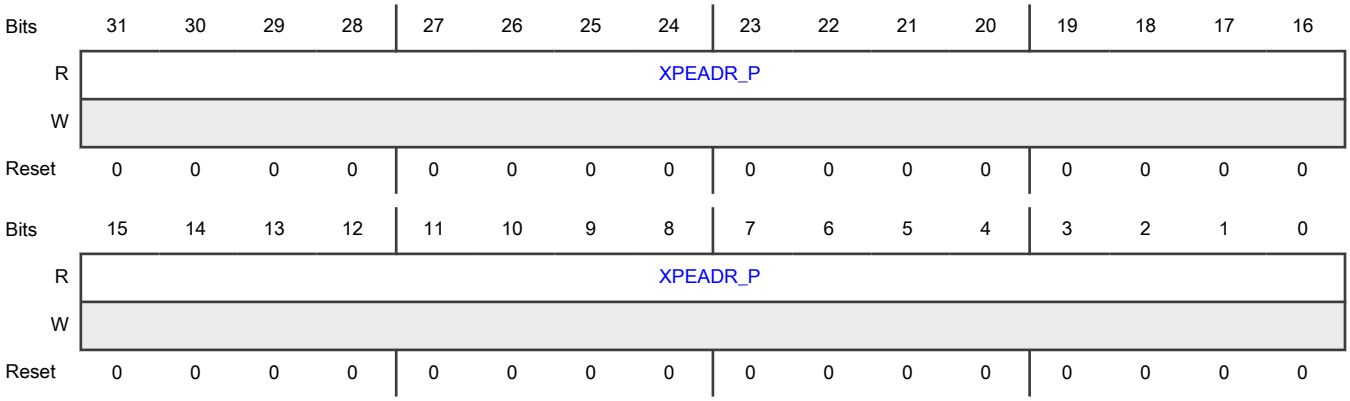
### Offset

| Register | Offset |
|---|---|
| PFCPGM_XPEADR_P | 30Ch |

### Function

Reflects the flash memory block number and offset address corresponding to PFCPGM_XPEADR_L. This register has the same format as the XPEADR register in the Flash Memory chapter—see it for details.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>XPEADR_P | Express Program Erase Address Physical<br>Contains the flash memory block select and offset address for flash memory express program/erase. |

## 21.5.10 Block n Sector Program Erase Lock (PFCBLK0_SPELOCK - PFCBLK4_SPELOCK)

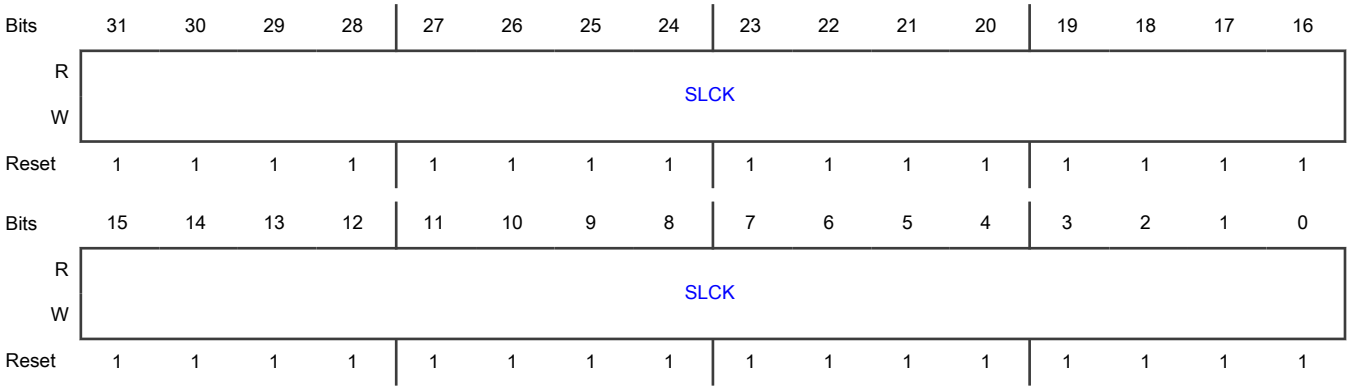
### Offset

| Register | Offset |
|---|---|
| PFCBLK0_SPELOCK | 340h |
| PFCBLK1_SPELOCK | 344h |
| PFCBLK2_SPELOCK | 348h |
| PFCBLK3_SPELOCK | 34Ch |
| PFCBLK4_SPELOCK | 350h |

### Function

Provides a way to protect sectors from being modified. Sector protection is available on the last 256 KB of every block (for 256 KB blocks, all sectors are available for protection). Each lock bit can be associated with a particular domain ID by writing 1 to the appropriate bit in PFCBLKn_SETSLOCK[SETSLCK]. After the lock is acquired, only a master having the same domain ID can modify the corresponding lock bit. If the corresponding PFCBLKn_SETSLOCK[SETSLCK] bit is not equal to 1, any master can modify the appropriate SLCK bit. If a lock bit is already acquired by a particular domain ID, any effort to modify (1 to 0, or 0 to 1) the lock bit by a master with a different domain ID results in a transfer error.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>SLCK | Sector Lock<br>Locks selected sector. If vector bit value = 0, the sector is available for program and erase operations. If vector bit value = 1, the sector is locked and not available for program and erase operations. |

## 21.5.11 Block UTEST Sector Program Erase Lock (PFCBLKU_SPELOCK)

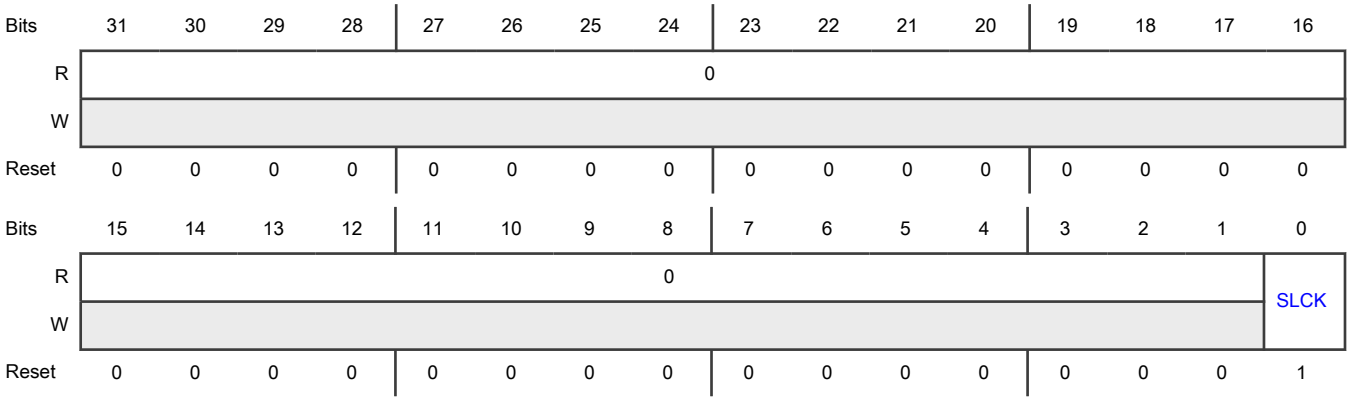
### Offset

| Register | Offset |
|---|---|
| PFCBLKU_SPELOCK | 358h |

### Function

Provides a way to protect sectors from being modified. Sector protection is available on the last 256 KB of every block (for 256 KB blocks, all sectors are available for protection). Each lock bit can be associated with a particular domain ID by writing 1 to PFCBLKU_SETSLOCK[SETSLCK]. After the lock is acquired, only a master having the same domain ID can modify the corresponding lock bit. If the corresponding PFCBLKU_SETSLOCK[SETSLCK] bit is not equal to 1, any master can modify the SLCK bit. If a lock bit is already acquired by a particular domain ID, any effort to modify (1 to 0, or 0 to 1) the lock bit by a master with a different domain ID results in a transfer error.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>SLCK | Sector Lock<br>Locks selected sector. If vector bit value = 0, the sector is available for program and erase operations. If vector bit value = 1, the sector is locked and not available for program and erase operations. |

## 21.5.12 Block n Super Sector Program Erase Lock (PFCBLK0_SSPELOCK - PFCBLK3_SSPELOCK)

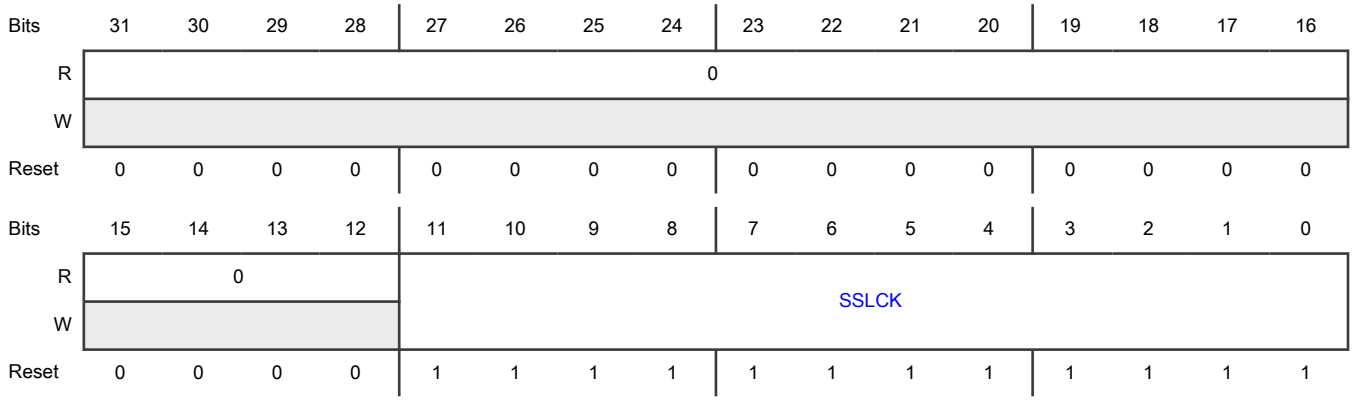
### Offset

| Register | Offset |
|---|---|
| PFCBLK0_SSPELOCK | 35Ch |
| PFCBLK1_SSPELOCK | 360h |
| PFCBLK2_SSPELOCK | 364h |
| PFCBLK3_SSPELOCK | 368h |

### Function

Provides a way to protect super sectors from being modified. Super sector protection is available on block space larger than 256 KB. For 256 KB blocks, this register is not applicable. For 512 KB blocks, the first half of the block is protected with super sector granularity. For 1 MB blocks, the first 768 KB is protected with super sector granularity. Each lock bit can be associated with a particular domain ID by writing 1 to the appropriate bit in PFCBLKn_SSETSLOCK[SSETSLCK]. After the lock is acquired, only a master having the same domain ID can modify the corresponding lock bit. If the corresponding PFCBLKn_SSETSLOCK[SSETSLCK] bit is not equal to 1, any master can modify the SSLCK bit. If a lock bit is already acquired by a particular domain ID, any effort to modify (1 to 0, or 0 to 1) the lock bit by a master with a different domain ID results in a transfer error.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11-0<br>SSLCK | Super Sector Lock<br>Locks selected super sector. If vector bit value = 0, the super sector is available for program and erase operations. If vector bit value = 1, the super sector is locked and not available for program and erase operations. |

## 21.5.13 Block n Set Sector Lock (PFCBLK0_SETSLOCK - PFCBLK4_SETSLOCK)

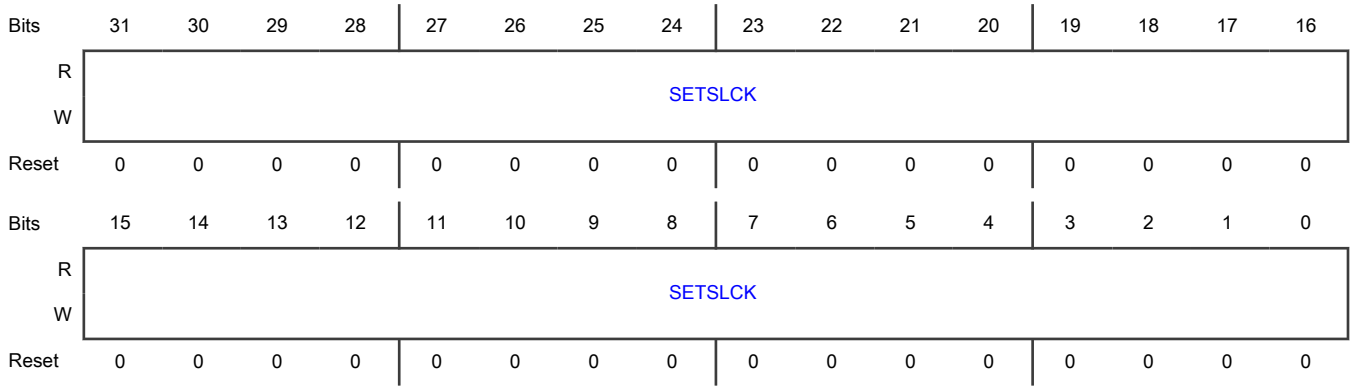
### Offset

| Register | Offset |
|---|---|
| PFCBLK0_SETSLOCK | 380h |
| PFCBLK1_SETSLOCK | 384h |
| PFCBLK2_SETSLOCK | 388h |
| PFCBLK3_SETSLOCK | 38Ch |
| PFCBLK4_SETSLOCK | 390h |

### Function

Provides a mechanism for the masters to gain the ownership of the corresponding PFCBLKn_SPELOCK lock bit based on domain id . After it is equal to 1, the bit is returned to 0 at next reset. If any SETSLOCK bit is not equal to 1, the corresponding LOCK bit can be modified by any master. If a bit is already acquired by a particular domain ID, any effort to modify the lock bit by a master with a different domain ID is ignored without transfer error.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>SETSLCK | If the vector bit value = 0, the corresponding lock bit is not owned by any master. If the vector bit value = 1, the lock bit is owned by the masters having the domain ID stored in PFCBLKn_LOCKMASTER_Sm. |

## 21.5.14 Block UTEST Set Sector Lock (PFCBLKU_SETSLOCK)

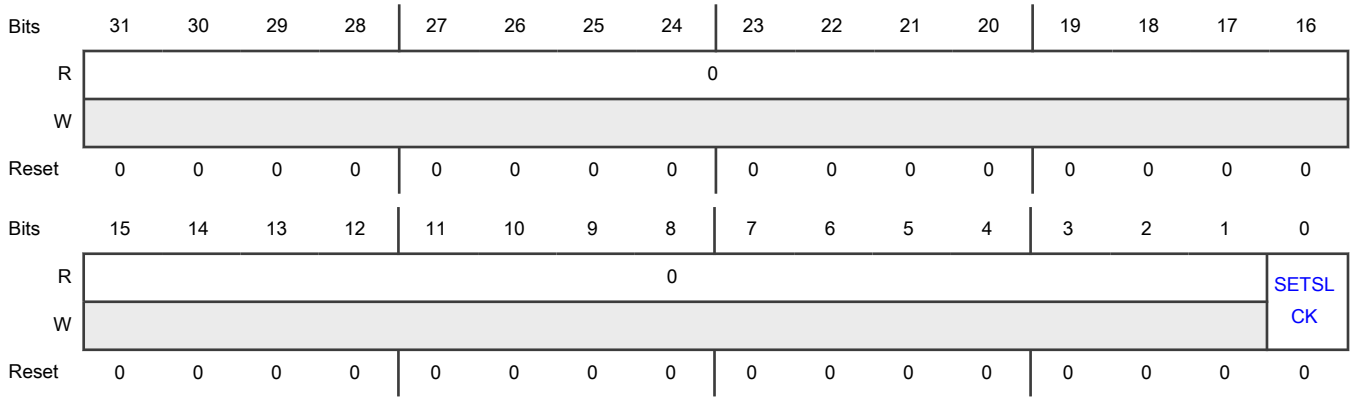
### Offset

| Register | Offset |
|---|---|
| PFCBLKU_SETSLOCK | 398h |

### Function

Provides a mechanism for the masters to gain ownership of the corresponding PFCBLKU_SPELOCK lock bit based on domain id. After it is equal to 1, the bit is returned to 0 at next reset. If any SETSLOCK bit is not equal to 1, the corresponding LOCK bit can be modified by any master. If a bit is already acquired by a particular domain ID, any effort to modify the lock bit by a master with a different domain ID is ignored without transfer error.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>SETSLCK | Set Sector Lock<br>Locks selected sector. If vector bit value = 0, the corresponding lock bit is not owned by any master. If vector bit value = 1, the lock bit is owned by the masters having the domain ID stored in PFCBLKn_LOCKMASTER_SSm. |

## 21.5.15 Block n Set Super Sector Lock (PFCBLK0_SSETSLOCK - PFCBLK3_SSETSLOCK)

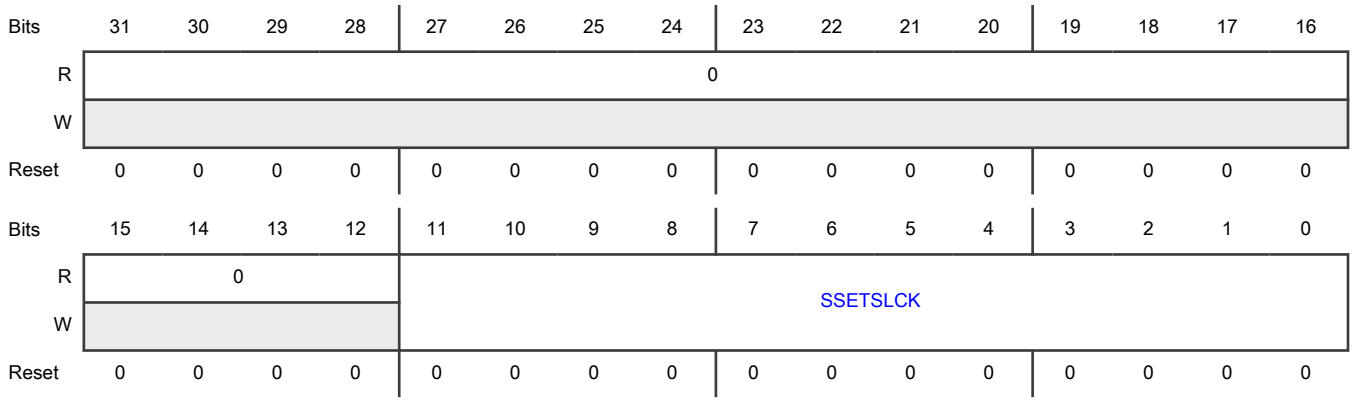
### Offset

| Register | Offset |
|---|---|
| PFCBLK0_SSETSLOCK | 39Ch |
| PFCBLK1_SSETSLOCK | 3A0h |
| PFCBLK2_SSETSLOCK | 3A4h |
| PFCBLK3_SSETSLOCK | 3A8h |

### Function

Provides a mechanism for the masters to gain ownership of the corresponding PFCBLKn_SPELOCK lock bit based on domain id. After it is equal to 1, the bit is returned to 0 at next reset. If any SSETSLOCK bit is not equal to 1, the corresponding LOCK bit can be modified by any master. If a bit is already acquired by a particular domain ID, any effort to modify the lock bit by a master with a different domain ID is ignored without transfer error.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11-0<br>SSETSLCK | Set Super Sector Lock<br>Locks selected super sector. If vector bit value = 0, the corresponding lock bit is not owned by any master. If vector bit value = 1, the lock bit is owned by the masters having the domain ID stored in Block a Lock Master Sector b (PFCBLK0_LOCKMASTER_S0 - PFCBLK4_LOCKMASTER_S7). |

## 21.5.16 Block a Lock Master Sector b (PFCBLK0_LOCKMASTER_S0 - PFCBLK4_LOCKMASTER_S7)

### Offset

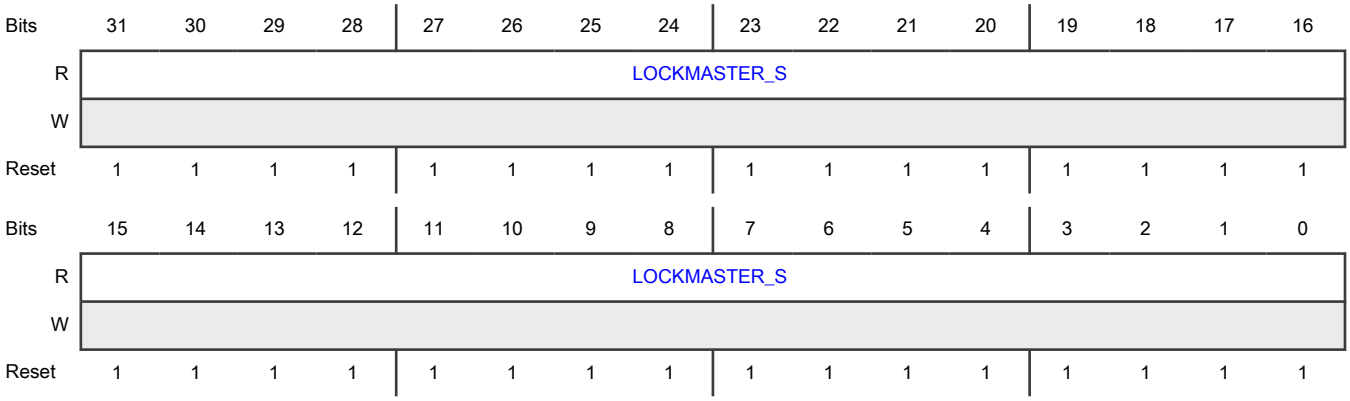
For a = 0 to 4; b = 0 to 7:

| Register | Offset |
|---|---|
| PFCBLKa_LOCKMASTER_Sb | 3C0h + (a × 20h) + (b × 4h) |

### Function

Provides the domain ID information of the master currently acquiring the lock bit. The domain ID is represented by an 8-bit field. This is a read-only register.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0 | Block a Lock Master Sector b<br>Contains domain ID of the master currently acquiring the lock bit. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| LOCKMASTER_S | PFCBLK0_LOCKMASTER_S0[LOCKMASTER_S[7:0]] holds the domain ID information of PFCBLK0_SPELOCK[0].<br>PFCBLK0_LOCKMASTER_S0[LOCKMASTER_S[15:8]] holds the domain ID information of PFCBLK0_SPELOCK[1], and so on in incremental order. |

## 21.5.17 Block UTEST Lock Master Sector (PFCBLKU_LOCKMASTER_S)

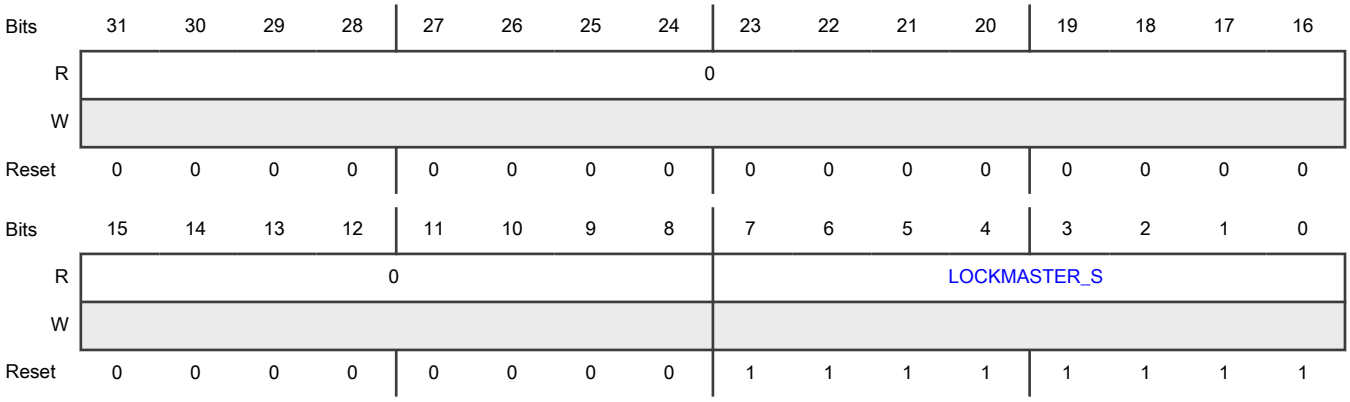
### Offset

| Register | Offset |
|---|---|
| PFCBLKU_LOCKMASTER_S | 480h |

### Function

Provides the domain ID information of the master currently acquiring the lock bit. The domain ID is represented by an 8-bit field. This is a read-only register.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>LOCKMASTER_S | Lock Master Sector<br>Contains domain ID of the master currently acquiring the lock bit.<br>PFCBLKU_LOCKMASTER_S[LOCKMASTER_S[7:0]] holds the domain ID information of PFCBLKU_SPELOCK[0]. |

### 21.5.18 Block m Lock Master Super Sector n (PFCBLK0_LOCKMASTER_SS0 - PFCBLK3_LOCKMASTER_SS2)

#### Offset

For a = 0 to 3; b = 0 to 2:

| Register | Offset |
|---|---|
| PFCBLKa_LOCKMASTER_SSb | 484h + (a × 10h) + (b × 4h) |

#### Function

Provides the domain ID information of the master currently acquiring the lock bit. The domain ID is represented by an 8-bit field. This is a read-only register.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LOCKMASTER_SS | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

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

#### Fields

| Field | Function |
|---|---|
| 31-0<br>LOCKMASTER_SS | Block a Lock Master Super Sector b<br>Contains domain ID of the master currently acquiring the lock bit.<br>PFCBLK0_LOCKMASTER_SS0[LOCKMASTER_SS[7:0]] holds the domain ID information of PFCBLK0_SSPELOCK[0].<br>PFCBLK0_LOCKMASTER_SS0[LOCKMASTER_SS[15:8]] holds the domain ID information of PFCBLK0_SSPELOCK[1], and so on in incremental order. |

## 21.6 Glossary

| | |
|---|---|
| **AHB** | Advanced high-performance bus |
| **ECC** | Error correcting code |
| **HSE** | Hardware security engine |
| **IPS** | Internal peripheral system |
| **ID** | Identification |

# Chapter 22
# RAM Controller (PRAMC)

## 22.1 Chip-specific PRAMC information

### 22.1.1 PRAMC instances and configuration

Table 95. PRAMC instances and configuration

| Instance | SRAM array | Size (KB) | Applicability |
|---|---|---|---|
| PRAMC_0 | SRAM 0 | 32 | MCXE31B, MCXE31/MCXE315/MCXE316 |
| | SRAM 0 | 128 | MCXE31B (128KB support)<br>MCXE317 (64KB support) |
| PRAMC_1 | SRAM 1 | 32 | MCXE31B |
| | SRAM 1 | 128 | MCXE31B |

### 22.1.2 RAM initialization

After chip power-on reset, you must initialize RAM to a known value using a 64-bit master before 32-bit masters can read or write to RAM. If you do not initialize RAM this way, any attempt to execute a 32-bit read or write access terminates with an uncorrectable ECC error event on this chip.

### 22.1.3 SRAM0 behavior while XRDC is configured to block access to SRAM0

While XRDC is configured to restrict access to SRAM0, the below behaviors should be considered:

1. In case if XRDC is configured to restrict access to SRAM0 and a read transaction to SRAM0 is performed: To ensure faster SRAM0 accesses, the access is routed to SRAM0 and blocked by PRAMC if XRDC is configured to block access. In such a case, if an error is present on the SRAM0 data, (that is error on data or error injected by EIM), the same is latched by ERM as well. This data is not used by application core since PRAMC gives a bus error in this case.
2. In case if XRDC is configured to restrict access to SRAM0 and a 32-bit write access is performed: Since SRAM is 64-bit wide, so a 32-bit write transaction is performed as a read-modify-write. In the read cycle, again as described in previous scenario, the PRAMC will initiate an error response. In this case also, if there is any error on SRAM0 data bus (i.e., error or error injected via EIM), the same is captured in ERM as well.

## 22.2 Overview

The RAM controller is an interface between the system bus (AHB-Lite 2.v6) and the integrated system RAM. It converts the protocols between the system bus and the dedicated RAM array interface.

The RAM controller supports one 64-bit AHB interface and a 64-bit RAM array interface. The AHB port provides a connection to the platform crossbar for direct RAM access from the various crossbar masters.

### 22.2.1 Block diagram

Shown below in Figure 48 is a simplified block diagram depicting the position of the RAM controller within a typical platform architecture.

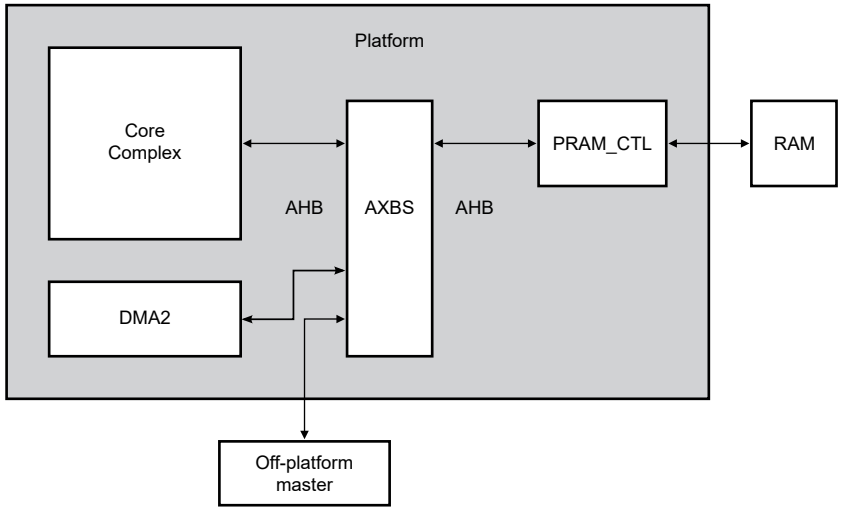



Figure 48. Simplified platform block diagram

### 22.2.2 Features

The following list summarizes the key features of the RAM controller:

- System bus supports 64-bit data
- Array interface supports a 64-bit data + 8-bit ECC interface
- Late-write support via 64-bit data + 8-bit ECC storage buffer supports single-cycle write accesses
- Configurable read access timing (zero or one wait state programmable) allowing use in large range of frequency targets
- Read-modify-write operation supports array write size smaller than a double-word
- Hardware EDC after ECC feature to check the ECC logic.

## 22.3 Functional description

This section describes the functions of the RAM controller.

### 22.3.1 Read and write operations

The RAM controller processes read and write requests to on-chip RAM and provides a register interface for access to performance tuning functions.

**NOTE**

The RAM controller register interface is accessible only in supervisor mode. Accesses in user mode return a transfer error.

#### 22.3.1.1 Reads

Read transfers, of any size, can be configured to complete with a zero or one additional wait state response on the system bus.

#### 22.3.1.2 Optional read wait state

The RAM controller can be optionally programmed to register RAM read data prior to returning it on the system bus. Setting the PRCRx[FT_DIS] field inserts a register in the read data path for use when operating the controller at high frequencies. The state of PRCRx[FT_DIS] field has no effect on writes.

A random, initial access takes two clock cycles (2T) to complete if PRCRx[FT_DIS]=0, and a WRAP4 burst will have an access time of 2-2-2-2. A random, initial access takes three clock cycles (3T) to complete if PRCRx[FT_DIS]=1, and a WRAP4 burst has an access time of 3-2-2-2.

**NOTE**

The number of cycles taken for a RAM access can vary ±1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes two clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either one or two cycles depending on RAM speed relative to the PRAMC controller clock frequency.

## 22.3.2 Writes

This section discusses various write operations of the RAM controller.

### 22.3.2.1 64-bit writes

Aligned 64-bit writes execute in a single AHB data phase cycle, resulting in zero wait states on back-to-back writes of these sizes. If, during the data phase of a write, a read is requested on the AHB, the write is registered in the late-write buffer, enabling the read to take place without a wait state. The valid buffered or late-write data is stored on the next available array address phase.

Back-to-back 64-bit writes execute slightly differently. The first write bypasses the late-write buffer—the write data is stored directly to the array in the same cycle in which it is valid on the AHB.

### 22.3.2.2 Less than 64-bit writes

Writes of size less than 64 bits incur a read-modify-write action as a consequence of the ECC coding scheme's 64-bit granularity. In the case of a read-modify-write action, the RAM controller performs SEC/DED on the read data. The write data is merged into the appropriate byte lanes along with the potentially corrected read data. A new codeword is generated based on the newly formed double-word. This double-word and its associated check-bits are subsequently written to RAM. Figure 49 provides details on the read-modify-write data path.

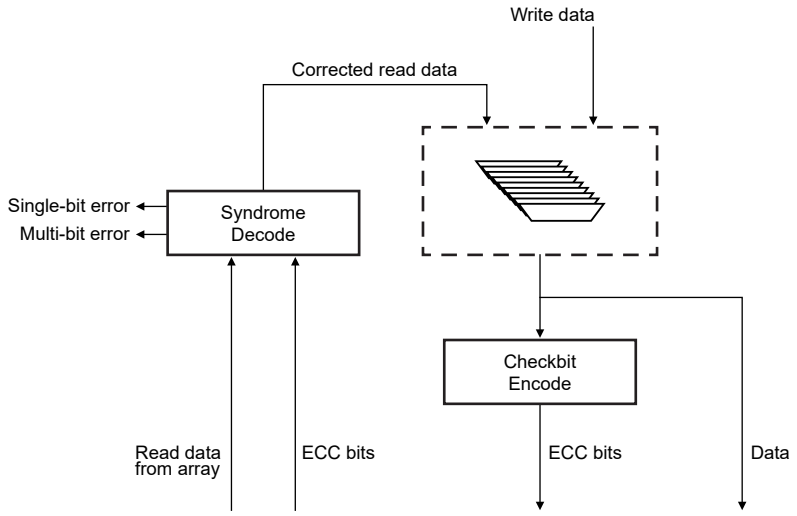



**Figure 49. Read-modify-write data path**

On a read-modify-write operation, the array read data is registered after it is decoded and before it is merged with the AHB write data. Therefore, writes of size less than 64 bits require the insertion of one wait state before the data phase can be completed.

#### 22.3.2.3 Unaligned writes

The RAM controller is compliant with the AMBA-AHB2.v6 Extensions specification with regard to byte strobes. The size of the transfer is sufficient to cover all bytes being written, and covers more bytes in the case of an unaligned transfer. The address of the transfer is rounded down to the nearest boundary of the size of the transaction.

**NOTE**

Unaligned writes always generate read-modify-write operations in the RAM controller in order to preserve the validity of the ECC codeword.

### 22.3.3 Clocking

The following table describes clocks used in this module.

**Table 96. Clock sources**

| Clock name | Description |
|---|---|
| System clock | This is the system clock used for accessing the PRAM controller over the system bus interconnect. |

### 22.3.4 Reset

**Table 97. Reset types**

| Reset | Description |
|---|---|
| System reset | This is the asynchronous system reset used for resetting the PRAM controller. |

### 22.3.5 Interrupts

This module has no interrupts.

## 22.4 External signals

There are no external signals for this module.

## 22.5 Initialization/application information

Each memory address must be written to a known value before it is read, to initialize the ECC. This includes reads generated from the read-modify-write operation that occurs when a write transfer of less than 64 bits or an unaligned write is requested. Without first writing an address to a known value, a read from the address will most likely generate an uncorrectable ECC event.

## 22.6 PRAMC register descriptions

The RAM controller module provides an IPS programming model mapped to a standard on-platform peripheral slot.

**NOTE**

1. The programming model can only be referenced using a 32-bit (word) access. Attempted references using different access sizes or to undefined (reserved) addresses generate an IPS error termination.
2. Attempted updates to the PRAMC programming model while a PRAMC operation is in progress will result in non-deterministic behavior. Software architect must be such as to avoid this scenario by ensuring that PRAMC configuration changes are made only during system boot or when only one master is enabled. In multi-core devices, update the PRAMC configuration only when one core is active and no other masters, for example, DMA or communications modules, are enabled. Move any instruction execution or memory references outside the system RAM while updating the PRAMC configuration, for example, to the core local memory space.

## 22.6.1 PRAMC memory map

PRAMC_0 base address: 4026_4000h

PRAMC_1 base address: 4046_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Platform RAM Configuration register 1 (PRCR1) | 32 | RW | 0000_0100h |

## 22.6.2 Platform RAM Configuration register 1 (PRCR1)

### Offset

| Register | Offset |
|---|---|
| PRCR1 | 0h |

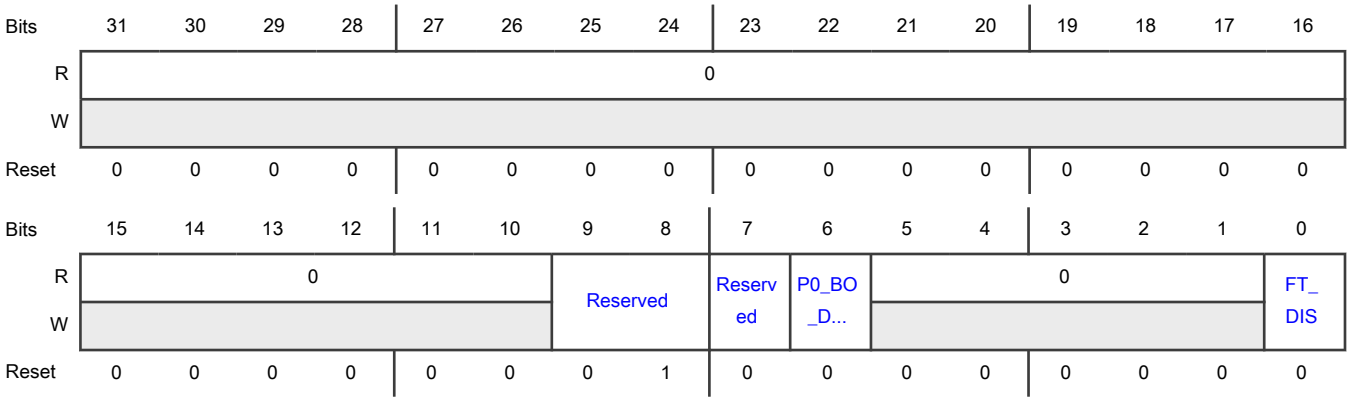
### Function

The Platform RAM Configuration register 1 (PRCR1) is used to specify operation of the RAM controller.

**NOTE**

This register is accessible only in supervisor mode. Accesses in user mode return a transfer error.

### Diagram



**Fields**

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9-8<br>— | Reserved |
| 7<br>— | Reserved |
| 6<br>P0_BO_DIS | Port p0 read burst optimization disable.<br>**NOTE**<br>The number of cycles taken for a RAM access can vary ± 1 clock cycle depending on the RAM speed relative to the PRAMC controller clock frequency. If system RAM is running at the same frequency as the PRAMC controller, a random initial access takes 2 clock cycles. If system RAM is running at a slower frequency, a random initial access may take 3 clock cycles. Subsequent burst beats take either 1 or 2 cycles depending on RAM speed relative to the PRAMC controller clock frequency.<br>0b - 64-bit WRP4 read bursts are optimized such that the controller returns a 2-1-1-1 response when PRCR1[FT_DIS ]=1<br>1b - 64-bit WRP4 read bursts are not optimized; the controller returns a 2-2-2-2 response when PRCR1[FT_DIS ]=1 |
| 5-1<br>— | Reserved |
| 0<br>FT_DIS | Flow-through disabled<br>This field defines the AHB response on reads. The state of this field has no impact on the response latency on writes. This bit is cleared by hardware reset.<br>**NOTE**<br>Do not change the FT_DIS bit value while accessing system RAM. Relocate code programming the FT_DIS bit to another memory area, for example, local core memory.<br>0b - RAM read data is passed directly to the system bus, incurring no additional latency<br>1b - RAM read data is registered prior to returning on the system bus, incurring one extra cycle of latency |

## 22.7 Glossary

| | |
|---|---|
| **AHB** | Arm advanced high-performance bus |
| **DED** | Double error detection |
| **DMA** | Direct memory access |
| **MUX** | Multiplexer |

| | |
|---|---|
| **SEC** | Single error correction |

# Chapter 23
# Clocking

## 23.1 Introduction

This chapter describes the clocking architecture and includes the following information:

- System clock specifics
- Clock sources
- Clock architecture
- Clock control registers
- Clock monitoring
- Clock gating
- Module clocking

This chapter discusses the clocks generated on the chip. Peripheral-specific protocol clocks are described in the corresponding peripheral chapters.

## 23.2 Features

- Multiple clock sources supported for clock generation:
  - Fast internal RC oscillator (FIRC)
  - Slow internal RC oscillator (SIRC)
  - Fast external crystal oscillator (FXOSC)
  - Slow external crystal oscillator (SXOSC)

    **NOTE**

    SXOSC is not available in 100-MAX QFP, 48-pin LQFP, and 100-pin LQFP packages. For details, see 'MCXE31 chip's feature comparison' table in Introduction chapter.

  - Phase-locked loop (PLL)
- Frequency-modulated PLL output clock to reduce electromagnetic emissions
- Precise clocks for timers and communication functions
- Glitchless clock switching Clock Generation module (MC_CGM) clock selectors
- System clock progressive clock frequency switching (PCFS)
- Clock monitoring units (CMU_FC, CMU_FM) to check clock integrity
- Core and peripheral clock gating using the Mode Entry module's (MC_ME) partition process configuration registers

## 23.3 Clocking overview

The MCXE31 clocking architecture consists of multiple:

- Clock sources
- Monitors
- Multiplexers
- Dividers

The blocks in the above bullet list provide the required clocking domains for the different functional blocks.

The sections in the following list show the clocking configuration of the chip:

- Clock source generation: PLL, FXOSC, FIRC, SIRC, and SXOSC
- MC_CGM mux 0 clocks: MC_CGM mux 0 generated clocks (not including EMAC clock signals)
- Clockout overview: CLKOUT_STANDBY and CLKOUT_RUN
- Other clocks
- EMAC clocking: Ethernet utilized clocking

The figures shown in MCXE31B clock system diagram, MCXE317 clock system diagram, and MCXE315/MCXE316 clock system diagram show the overall clock tree for the different chip variants of the MCXE31 family, which are a combination of the sections mentioned in the bullet list above.

### 23.3.1 Clock source generation

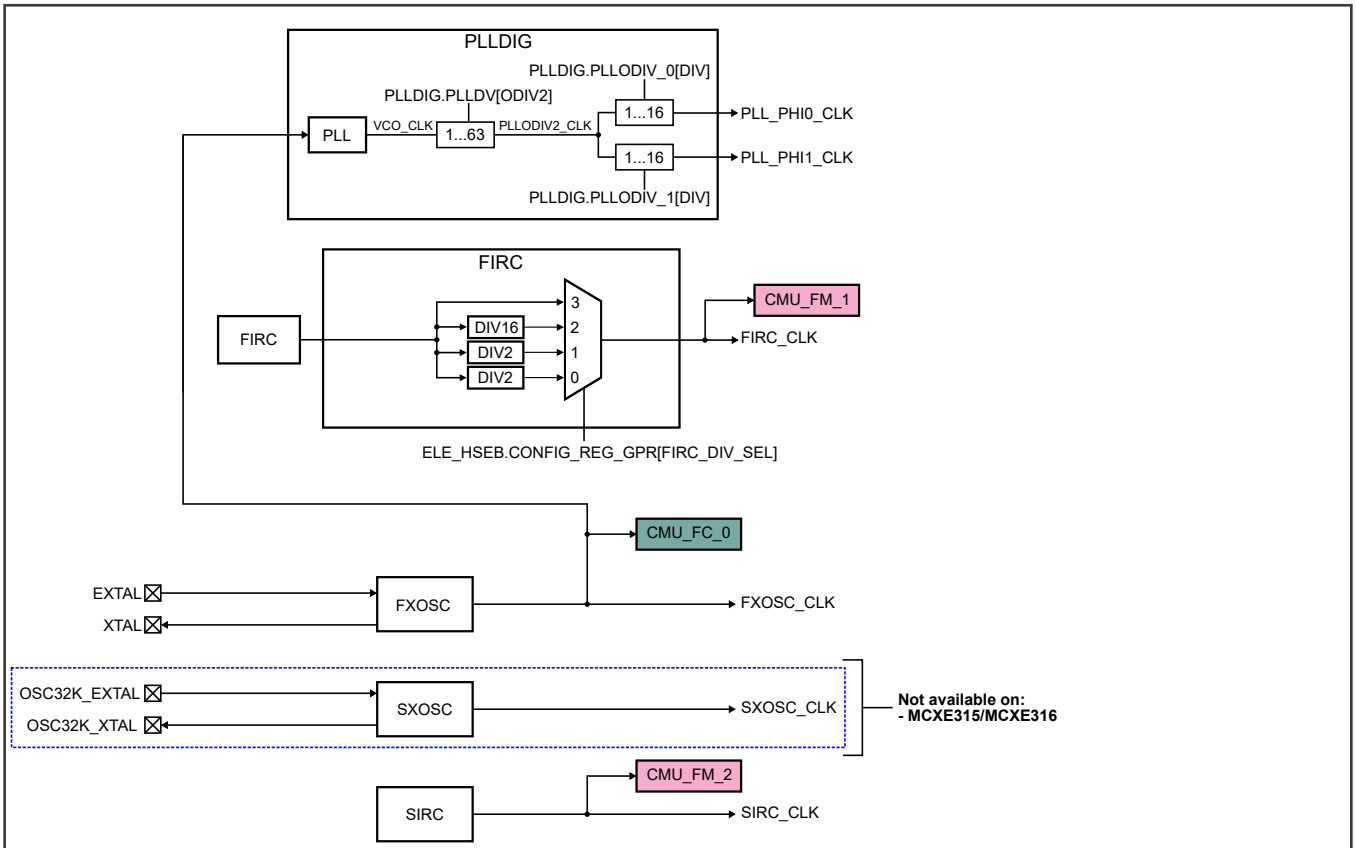



Figure 50. Clock source generation

### 23.3.2 MC_CGM mux 0 clocks

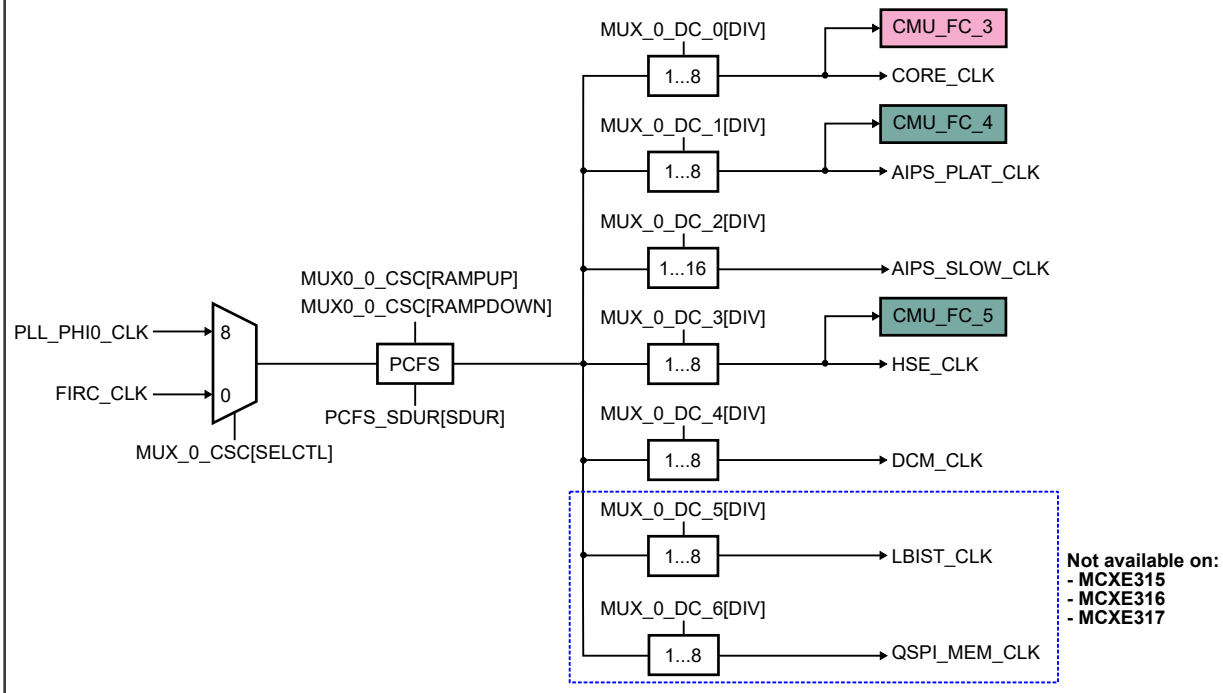



**Figure 51. MC_CGM mux 0 clocks**

**NOTE**

The clock frequency relationship between TCK and HSE_CLK clocks for ELE_HSEB must be a minimum ratio of 1:1.5. For example, if HSE_CLK equals 80 MHz, then TCK must be less than or equal to 53 MHz (80 MHz ÷ 1.5).

### 23.3.3 Clockout overview

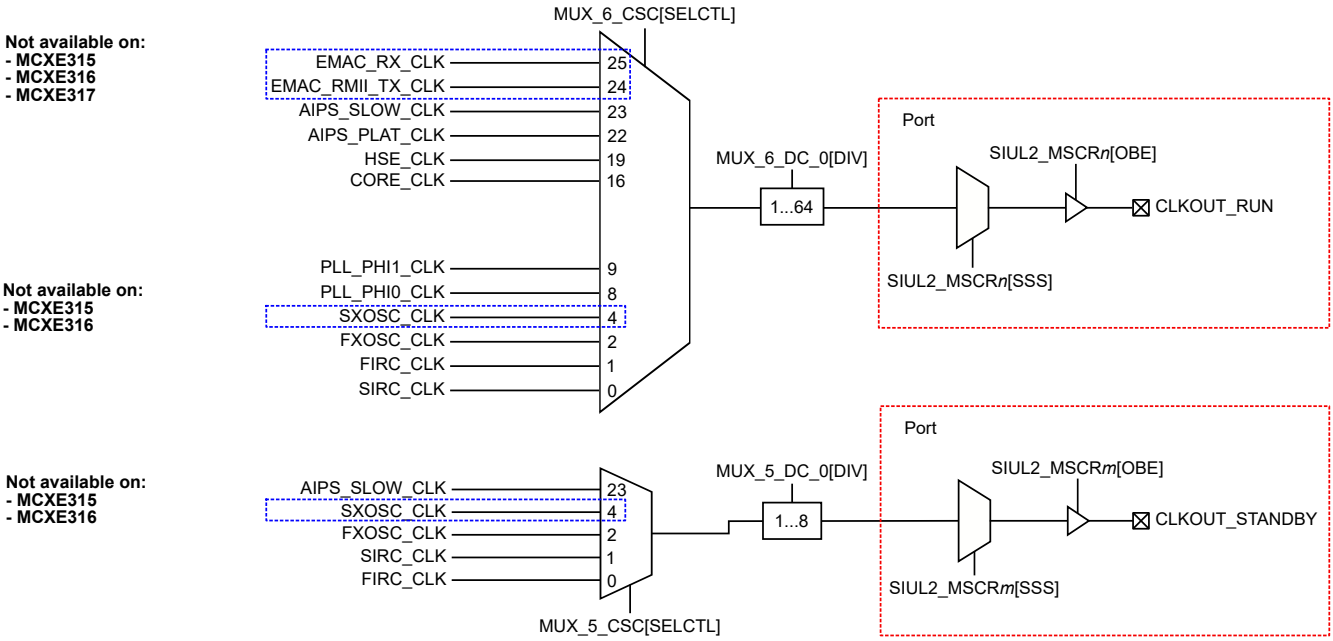



Figure 52. Clockout overview

**NOTE**

CLKOUT_RUN is not available during Standby mode.

#### 23.3.3.1 SIUL2 options for CLKOUT_RUN

Table 98. SIUL2 options for CLKOUT_RUN

| Source | Destination | Port | MSCR*n* | MSCR fields | | |
|---|---|---|---|---|---|---|
| | | | | OBE | IBE | SSS |
| MUX_6_DC_0 divider output | CLKOUT_RUN | PTB5 | 37 | 1 | 0 | 0101b |
| | | PTD10 | 106 | 1 | 0 | 0110b |
| | | PTD14 | 110 | 1 | 0 | 0111b |

#### 23.3.3.2 SIUL2 options for CLKOUT_STANDBY

Table 99. SIUL2 options for CLKOUT_STANDBY

| Source | Destination | Port | MSCR*m* | MSCR fields | | |
|---|---|---|---|---|---|---|
| | | | | OBE | IBE | SSS |
| MUX_5_DC_0 divider output | CLKOUT_STANDBY | PTA12 | 12 | 1 | 0 | 0011b |
| | | PTE10 | 138 | 1 | 0 | 0101b |

### 23.3.4 Other clocks

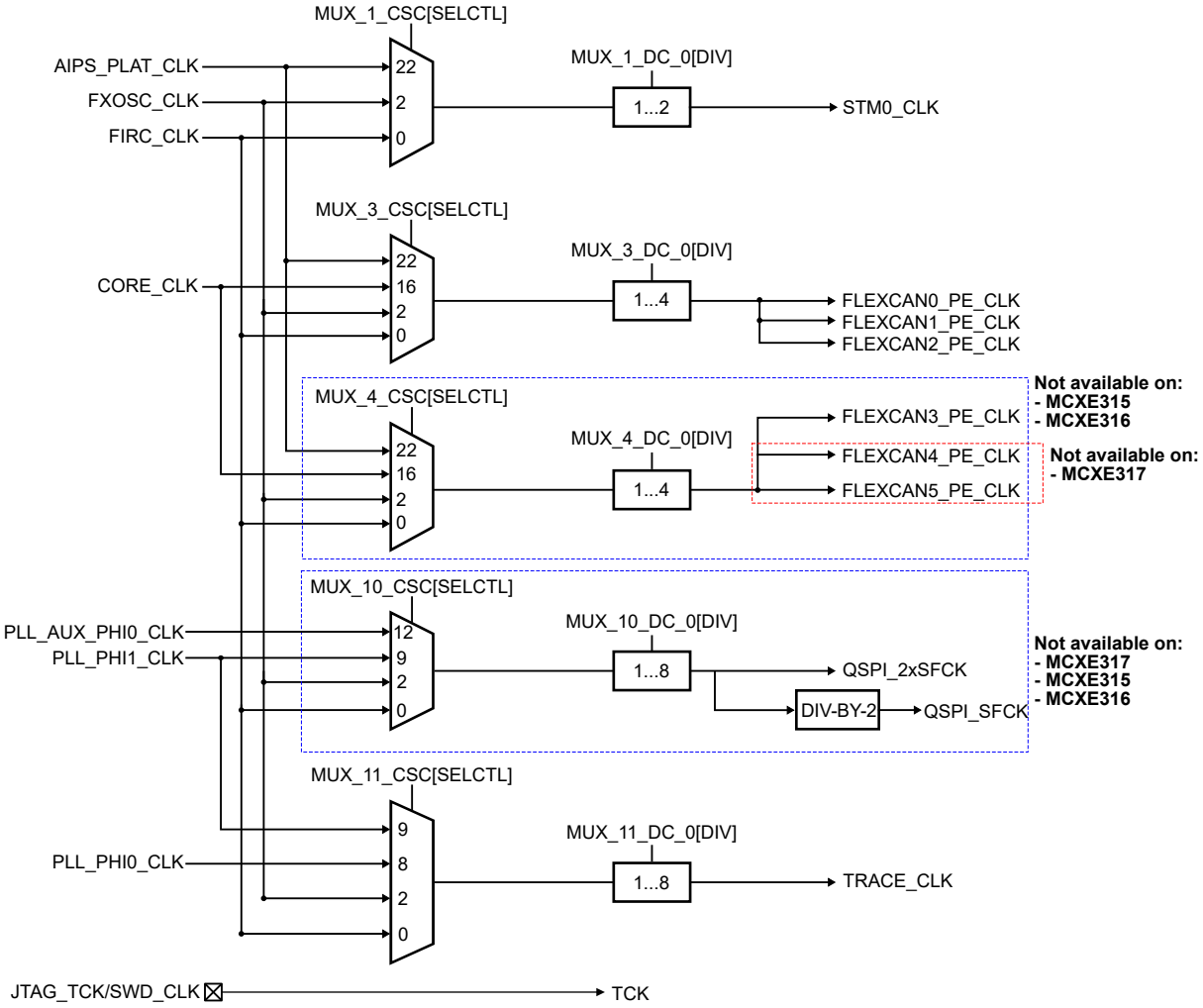



Figure 53. Other clocks

## 23.3.5 MCXE31B clock system diagram

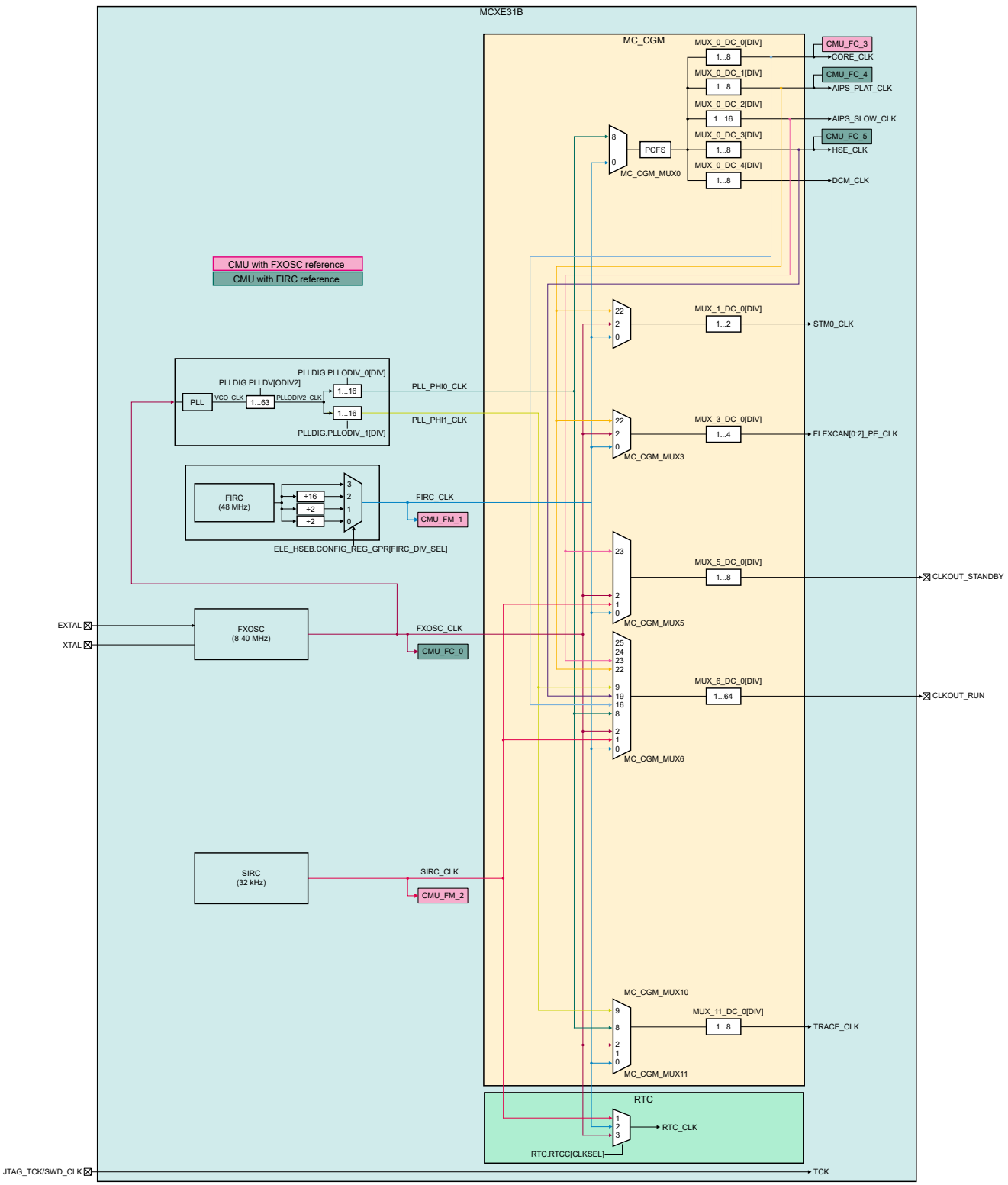



Figure 54. MCXE31B clock system diagram

## 23.3.6 MCXE317 clock system diagram

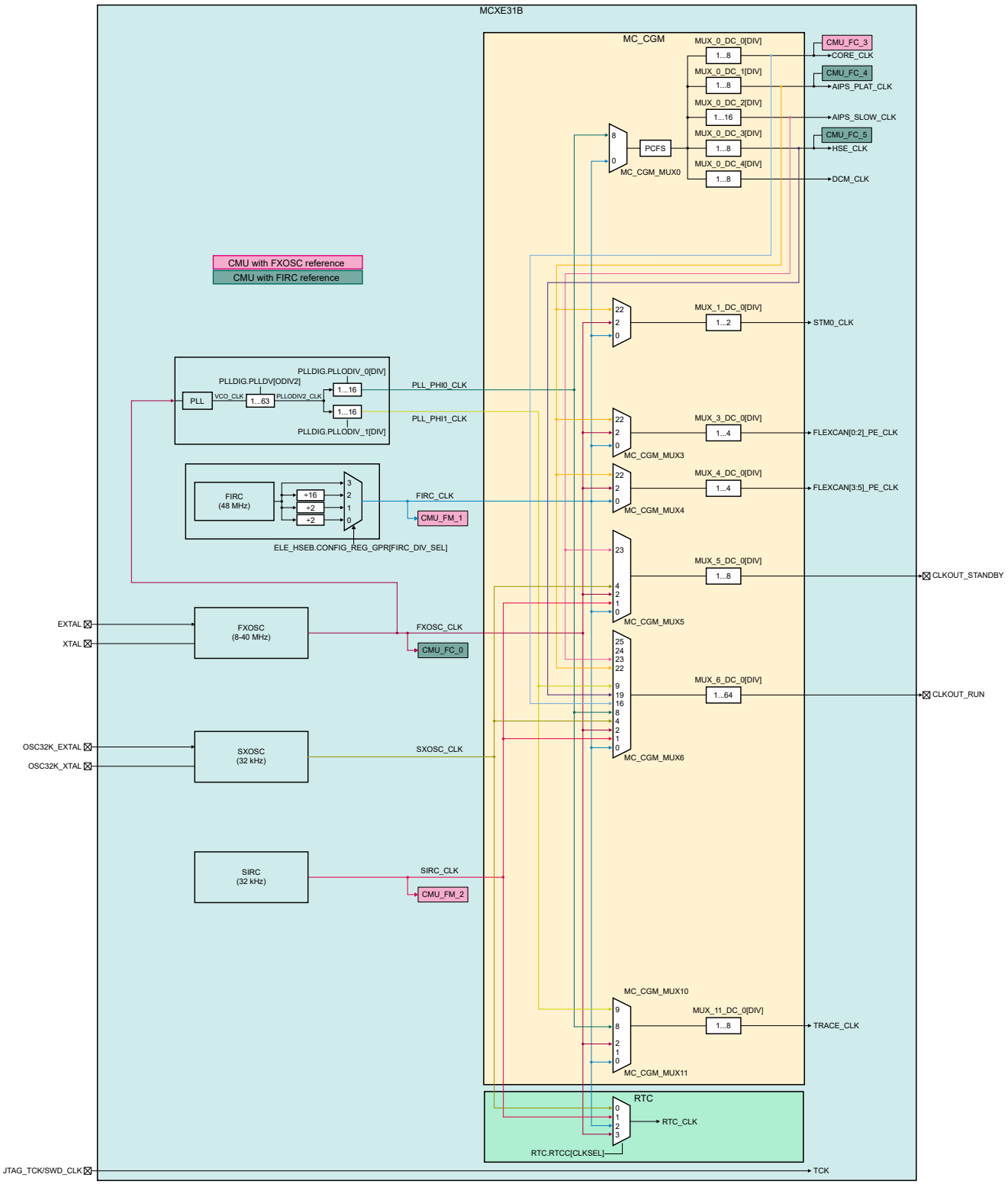



Figure 55. MCXE317 clock system diagram

## 23.3.7 MCXE315/MCXE316 clock system diagram

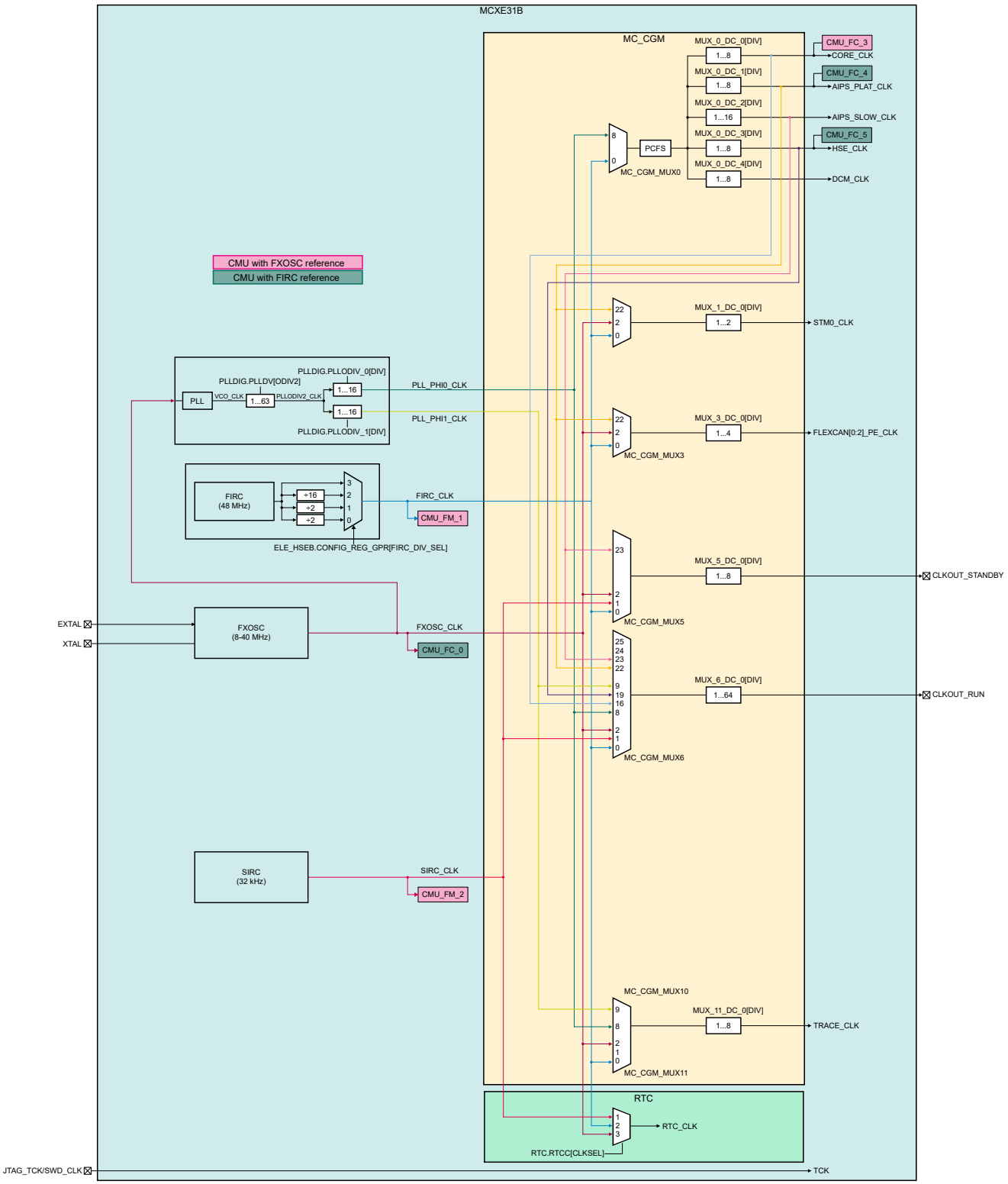



Figure 56. MCXE315/MCXE316 clock system diagram

## 23.4 Clock sources

### 23.4.1 Introduction

The chip contains the following clock sourcing modules:

- FIRC
  - FIRC_CLK is the default system clock source.
- SIRC
- PLL
- FXOSC
- SXOSC (not available on the MCXE315/MCXE316)

The following list shows some of the clock system features:

- All clock sources support software configurability for enabling or disabling. [3]
- All clock sources, except SXOSC_CLK, are initialized to their default state on functional reset.
- The SXOSC_CLK supports RTC applications across functional reset and is reset on destructive reset.

Only SIRC_CLK and FIRC_CLK are enabled out of reset and are enabled on any functional reset. The other clock sources are disabled on reset.

#### 23.4.1.1 Chip clock sources

**Table 100. Chip clock sources**

| Clock source | Divider | Default state | Reset | Uses |
|---|---|---|---|---|
| FIRC_CLK (48 MHz) | 1, 2, 16 | On | POR (enabled on functional and destructive reset)<br>POR assertion - FIRC_CLK disabled asynchronously<br>POR deassertion - FIRC_CLK enabled | • Boot clock<br>• Default system clock source<br>• Safe clock for safety modules FCCU and FOSU<br>• SIUL2 filter clock<br>• MC_RGM clock source |
| PLL_PHI*n*_CLK (640-1280 MHz) | 1...16 | Off | Functional (disabled on functional reset) | • Optional system clock source<br>• Communication modules (FlexCAN, EMAC, QuadSPI, and so on) |
| FXOSC_CLK (8-40 MHz) | — | Off | Functional (disabled on functional reset) | • Reference clock source for PLL<br>• Communication modules (FlexCAN, EMAC, QuadSPI, and so on) |
| SXOSC_CLK (32.768 KHz) | — | Off | Destructive (disabled on destructive reset) | • RTC source for operation across functional reset (SXOSC_CLK is not available on the MCXE315/MCXE316) |

*Table continues on the next page...*

[3] FIRC_CLK and SIRC_CLK cannot be disabled during Run mode.

Table 100. Chip clock sources (continued)

| Clock source | Divider | Default state | Reset | Uses |
|---|---|---|---|---|
| SIRC_CLK (32 KHz) | — | On | POR (enabled on functional reset) | • Safe clock along with FIRC_CLK<br>• SWT clock source<br>• POR_WDG source clock |

## 23.4.2 Chip input clocks

Table 101. Chip input clocks

| Pin | Description |
|---|---|
| XTAL | FXOSC crystal pins |
| EXTAL | |
| OSC32K_XTAL | SXOSC crystal pins (not available on the MCXE315/MCXE316) |
| OSC32K_EXTAL | |
| EMAC_MII_RMII_TX_CLK | EMAC transmitter clock/EMAC RMII clock (not availalble on the MCXE315/MCXE316 or MCXE317) |
| EMAC_MII_RX_CLK | EMAC receiver clock (not availalble on the MCXE315/MCXE316 or MCXE317) |
| JTAG_TCLK/SWD_CLK | JTAG/SWD clock |
| SAI*n*_MCLK | SAI_*n* clock in slave mode (not availalble on the MCXE315/MCXE316 or MCXE317) |
| SAI*n*_BCLK | SAI_*n* bit clock in slave mode (not availalble on the MCXE315/MCXE316 or MCXE317) |
| LPSPI*n*_SCK | LPSPI*n* serial clock in slave mode |
| LPI2C*n*_SCL | LPI2C_*n* clock |
| LPI2C*n*_SCLS | LPI2C_*n* secondary clock |

## 23.4.3 Chip output clocks

Table 102. Chip output clocks

| Pin | Description |
|---|---|
| CLKOUT_RUN | Available during Run mode, unavailable during Standby mode |
| CLKOUT_STANDBY | Available during both Run and Standby modes |
| LPSPI*n*_SCK | LPSPI*n* serial clock in master mode |
| LPI2C*n*_SCL | LPI2C_*n* clock |
| LPI2C*n*_SCLS | LPI2C_*n* secondary clock |
| EMAC_MII_RMII_MDC | EMAC clock for control data transfer to PHY (not availalble on MCXE315/MCXE316 or MCXE317) |
| EMAC_MII_RMII_TX_CLK | EMAC transmit clock (not availalble on MCXE315/MCXE316 or MCXE317) |

*Table continues on the next page...*

Table 102. Chip output clocks (continued)

| Pin | Description |
|---|---|
| TRACE_ETM_CLKOUT | ETM trace clock (not availalble on MCXE315/MCXE316, MCXE317) |
| SAI*n*_BCLK | SAI_*n* bit clock in master mode (not availalble on MCXE315/MCXE316 or MCXE317) |
| QuadSPI_SCKFA | QuadSPI serial clock for serial flash device A (fast) (not availalble on MCXE315/MCXE316 or MCXE317) |

### 23.4.4 Fast internal RC oscillator (FIRC)

The chip has an FIRC with the following features:

- Acts as the system clock source on power-up and after any reset event.
  - Important to detect FIRC_CLK failure and recovery
- Acts as the chip's safe clock for safety-relevant applications.
- Is always enabled in Run mode and can be optionally enabled in Standby mode.
- Used as clock source for the following:
  - MC_RGM
  - FCCU and FOSU
  - SIUL2 filters

#### 23.4.4.1 FIRC failure detection

The FIRC_CLK is the safe clock source used as the FCCU and FOSU clock source. The chip supports FIRC_CLK failure detection and recovery by the mechanisms described in the following cases:

- Case 1 - FIRC_CLK not used as system clock and goes out of range:
  - CMU_FM_1 continuously measures the FIRC_CLK clock frequency using FXOSC_CLK as the reference. On each metering window completion, CMU_FM_1 asserts an interrupt (if configured, CMU_FM_1.IER[FMCIE] is 1). You store a configured reference clock count CMU_FM_1.RCCR[REF_CNT] (for example, number of FXOSC_CLK cycles in metering window). Your software checks the FIRC_CLK clock counts by reading CMU_FM_1.SR[MET_CNT].
  - In the event FIRC_CLK goes out of range, the software detects the frequency variation after the subsequent metering window by checking CMU_FM_1.SR[MET_CNT] and takes necessary action by either of the recommended options:
    - SBC-driven power cycle: The SBC receives indication from the chip (through GPIO toggle, QuadSPI communication, and so on). The SBC initiates a power cycle sequence.
    - Software-driven functional reset: The chip executes a functional reset as configured by your software.
- Case 2 - FIRC_CLK not used as system clock and fails (becomes stuck):
  - CMU_FM_1 continuously measures the FIRC_CLK frequency with FXOSC_CLK as the reference. Application software must check FIRC_CLK after a defined time limit by reading CMU_FM_1.SR[MET_CNT] and CMU_FM_1.SR[FMTO].
  - If there is an FIRC_CLK failure, the CMU_FM_1 writes a 1 to the timeout status flag CMU_FM_1.SR[FMTO].
  - When CMU_FM_1.SR[FMTO] is 1, application software takes necessary action by an SBC-driven power cycle wherein the chip provides an indication to the SBC (through GPIO toggle, QuadSPI communication, and so on). The SBC then initiates a power cycle sequence.
- Case 3 - FIRC_CLK used as system clock and goes out of range or fails (becomes stuck):
  - CMU_FC_3, CMU_FC_4, and CMU_FC_5 continuously monitor the system clock nodes with FXOSC_CLK as reference for FLL or FHH events.

- In the event of FIRC_CLK failure (when FIRC_CLK acts as the system clock source), these CMUs report the FLL event, acting as a destructive reset source.
- The system then undergoes the reset sequence, during which the FIRC_CLK is reinitialized.
- The MC_RGM.DES fields indicate the source of the reset event.
- In addition, CMU_FM_1 monitors FIRC_CLK. If the software is not able to service the CMU_FM_1 interrupt before POR_WDG timeout, POR_WDG treats this as a critical FIRC_CLK failure and initiates a POR_WDG recovery. Therefore, you must ensure that, if enabled, the CMU_FM_1 interrupt must be serviced within the POR_WDG timeout duration.

#### 23.4.4.2 FIRC_CLK behavior in Standby mode

FIRC_CLK can be optionally enabled in Standby mode by configuring FIRC.STDBY_ENABLE[STDBY_EN].

When the PMC acknowledges the Standby mode entry, the FIRC_CLK switches from the On state to the Standby mode configuration selected by the FIRC.STDBY_ENABLE[STDBY_EN] configuration. On wakeup from Standby mode, the FIRC_CLK configuration switches from the Standby mode configured state to the On state.

**NOTE**

When the trims are being applied, the FIRC will appear as momentarily disabled as shown in Reset timing diagram in the Reset overview chapter.

### 23.4.5 SIRC

The chip has a SIRC having the following features:

- Is always enabled in Run mode and can be optionally enabled in Standby mode. Having the SIRC always enabled improves system robustness by ensuring that a clock is always available for various SWTs when reducing the chip power consumption in Standby mode.
- Used as clock source for the following:
  - SWT
  - POR_WDG

#### 23.4.5.1 SIRC failure detection

Like the FIRC_CLK, the SIRC_CLK is a safe clock for the design. The SIRC_CLK is used as clock source for SWTs and POR_WDG and therefore it is important to detect SIRC failure and ensure its recovery. The chip supports SIRC_CLK failure detection and recovery by the mechanism described below.

- Case 1 - SIRC_CLK goes out of range:
  - CMU_FM_2 continuously measures the SIRC_CLK clock frequency with FXOSC_CLK as reference. On each metering window completion, the CMU_FM_2 raises an interrupt. The software checks the SIRC clock counts by CMU_FM_2.SR[MET_CNT] with respect to the reference clock counts CMU_FM_2.RCCR[REF_CNT].
  - In the event of clock going out of range, the software detects the frequency variation after the subsequent metering window by checking CMU_FM_2.SR[MET_CNT] and takes necessary action by either of the recommended options:
    1. SBC-driven power cycle. The chip gives an indication to the SBC (through GPIO toggle, QuadSPI communication, and so on.). The SBC then initiates a power cycle sequence.
    2. SW-driven functional reset. The chip executes a functional reset by software.
- Case 2 - SIRC_CLK fails (becomes stuck):
  - CMU_FM_2 continuously measures the SIRC clock frequency with FXOSC_CLK as reference. Software needs to check this reference after a predefined time by checking CMU_FM_2.SR[MET_CNT] and CMU_FM_2.SR[FMTO].

— In the event of SIRC_CLK clock failure, the CMU_FM_2 writes 1 to the timeout status flag field in its status register, namely, CMU_FM_2.SR[FMTO].

— When CMU_FM_2.SR[FMTO] is 1, the software takes necessary action by either of the recommended options:

1. SBC-driven power cycle: The chip gives an indication to the SBC (through GPIO toggle, QuadSPI communication, and so on). The SBC then initiates a power cycle sequence.
2. Software-driven functional reset: User software executes a functional reset.

#### 23.4.5.2 SIRC behavior in Standby mode

SIRC can optionally be enabled in Standby mode by configuring SIRC.MISCELLANEOUS_IN[STANDBY_ENABLE].

When PMC acknowledges the Standby mode entry, the SIRC switches from the On state to the standby configuration selected by SIRC.MISCELLANEOUS_IN[STANDBY_ENABLE] configuration. On wake-up from Standby mode, the SIRC configuration switches back from the standby-configured state to the On state.

**NOTE**

When the trims are being applied, the SIRC will appear as momentarily disabled similar to FIRC, as shown in section Reset timing diagram.

### 23.4.6 FXOSC

The chip supports an 8–40 MHz fast crystal oscillator which has following features:

- Acts as the reference for PLL.
- Supports crystal input mode and bypass mode if using an external oscillator.
- Acts as a clock source for communication modules:
  — FlexCAN
  — QuadSPI[4]
  — EMAC (EMAC_CLK_TS)[4]

### 23.4.7 SXOSC

The chip supports a slow crystal oscillator (SXOSC) which has the following features (not available on the MCXE315/MCXE316):

- Supports crystal input mode and bypass mode.
- Acts as a clock source for RTC. As SXOSC is not affected by functional reset, it supports RTC operation across functional reset. SXOSC is only reset on destructive reset.

### 23.4.8 PLL

The chip contains up to two PLL to provide precision clock source with the following features:

- Optional system clock source (in high performance applications)
- System clock source in safety applications
- Can be used as clock source for communication modules, when configured as system clock source:
  — FlexCAN (in SYNC mode operation)
  — EMAC (not available on MCXE315/MCXE316 or MCXE317)
  — QuadSPI (not available on MCXE315/MCXE316 or MCXE317)
  — LPSPI

[4] Not present in MCXE315/MCXE316 and MCXE317.

- LPI2C
- FlexIO
- LPUART

- Supports frequency modulation
- Contains lock status monitoring logic which supports loss-of-lock indication

**NOTE**

See PLL Digital Interface (PLLDIG) for PLL configuration details. See the *MCXE31 Data Sheet* for PLL limitations.

#### 23.4.8.1 PLL configurations

The PLL output predivider frequency depends on the PLLDIG.PLLDV[RDIV] and PLLDIG.PLLDV[MFI] configurations. The PLL VCO clock can be divided further by configuring PLLDIG.PLLODIV_0[DIV] for PLL_PHI0_CLK and PLLDIG_PLLODIV_1[DIV] for PLL_PHI1_CLK (see the PLL Digital Interface (PLLDIG) chapter for configuration details).

##### 23.4.8.1.1 PLL configuration sequence

Before enabling the PLL, you must enable FXOSC_CLK and wait until it is stable. FXOSC.STAT[OSC_STAT] must be monitored to determine the FXOSC_CLK status.

To disable the PLL, the software must disable PLL first and only then disable FXOSC (if required).

### 23.4.9 Chip clock outputs

The chip supports two CLKOUT_*x* pins for allowing viewing of some internal clocks as follows:

- CLKOUT_STANDBY
  - Used for showing clocks available in Run and Standby modes.
- CLKOUT_RUN
  - Used for showing only Run mode clocks.

See the Clockout overview section and the Clock Generation Module (MC_CGM) chapter for details on available clocks and configuration.

**NOTE**

The CLKOUT_STANDBY registers are latched when the chip enters Standby mode and are reset in Standby mode sequence. Therefore, the CLKOUT_STANDBY signal needs to be reconfigured on Standby mode exit.

**NOTE**

CLKOUT_STANDBY is available on two pads GPIO[12] and GPIO[138] but CLKOUT across functional reset and standby is supported only on GPIO[12] and OBE(output buffer enable) is controlled by DCM GPR bit. Please refer to DCMRWP1[3] bit for detail.

## 23.5 MC_CGM

The MC_CGM controls the clock functionality of the chip. See the Clock Generation Module (MC_CGM) chapter for details on MC_CGM clocking controls.

### 23.5.1 MC_CGM clock multiplexer types

In this chip, CLKOUT_RUN, CLKOUT_STANDBY, and TRACE_CLK multiplexers are software-controlled multiplexers. The rest are hardware-controlled multiplexers (see the Clock Generation Module (MC_CGM) chapter for details on software and hardware multiplexers).

### 23.5.2 MC_CGM clock multiplexers for MCXE315/MCXE316 and MCXE317

Table 103. MC_CGM clock multiplexers for MCXE315/MCXE316 and MCXE317

| Clock mux | Register description | Source inputs [1] | Register | Selector output | Divider output |
|---|---|---|---|---|---|
| Clock mux 0 | Select Control | FIRC_CLK<br>PLL_PHI0_CLK | MUX_0_CSC | | — |
| | Select Status | | MUX_0_CSS | | |
| | Divider 0 Control | — | MUX_0_DC_0 | — | CORE_CLK |
| | Divider 1 Control | | MUX_0_DC_1 | | AIPS_PLAT_CLK |
| | Divider 2 Control | | MUX_0_DC_2 | | AIPS_SLOW_CLK |
| | Divider 3 Control | | MUX_0_DC_3 | | HSE_CLK |
| | Divider 4 Control | | MUX_0_DC_4 | | DCM_CLK |
| Clock mux 3 | Select Control | FIRC_CLK<br>FXOSC_CLK<br>AIPS_PLAT_CLK | MUX_3_CSC | — | — |
| | Select Status | | MUX_3_CSS | | |
| | Divider 0 Control | — | MUX_3_DC_0 | | FLEXCAN0_PE_CLK<br>FLEXCAN1_PE_CLK<br>FLEXCAN2_PE_CLK |
| Clock mux 4 [2] | Select Control | FIRC_CLK<br>FXOSC_CLK<br>AIPS_PLAT_CLK | MUX_4_CSC | — | — |
| | Select Status | | MUX_4_CSS | | |
| | Divider 0 Control | — | MUX_4_DC_0 | | FLEXCAN3_PE_CLK<br>FLEXCAN4_PE_CLK<br>FLEXCAN5_PE_CLK |
| Clock mux 5 [3] | Select Control | FIRC_CLK<br>SIRC_CLK<br>FXOSC_CLK<br>SXOSC_CLK<br>AIPS_SLOW_CLK | MUX_5_CSC | — | — |
| | Select Status | | MUX_5_CSS | | |
| | Divider 0 Control | — | MUX_5_DC_0 | | CLKOUT_STANDBY |
| Clock mux 6 [4] | Select Control | FIRC_CLK<br>SIRC_CLK<br>FXOSC_CLK<br>SXOSC_CLK<br>PLL_PHI0_CLK<br>PLL_PHI1_CLK<br>CORE_CLK<br>HSE_CLK<br>AIPS_PLAT_CLK<br>AIPS_SLOW_CLK | MUX_6_CSC | — | — |
| | Select Status | | MUX_6_CSS | | |

*Table continues on the next page...*

Table 103. MC_CGM clock multiplexers for MCXE315/MCXE316 and MCXE317 (continued)

| Clock mux | Register description | Source inputs [1] | Register | Selector output | Divider output |
|---|---|---|---|---|---|
| | | EMAC_RMII_TX_CLK<br>EMAC_RX_CLK | | | |
| | Divider 0 Control | — | MUX_6_DC_0 | | CLKOUT_RUN |
| Clock mux 11 | Select Control | FIRC_CLK<br>FXOSC_CLK<br>PLL_PHI0_CLK<br>PLL_PHI1_CLK | MUX_11_CSC | — | — |
| | Select Status | | MUX_11_CSS | | |
| | Divider 0 Control | — | MUX_11_DC_0 | | TRACE_CLK |

1. The default clock selected for all clock mux selectors is FIRC_CLK (out of reset).
2. Clock mux 4 and FLEXCAN[3:5]_PE_CLK are not available on the MCXE315/MCXE316.
3. SXOSC as source for clock mux 5 is not available on the MCXE315/MCXE316.
4. SXOSC as source for clock mux 6 is not available on the MCXE315/MCXE316.

### 23.5.3 MC_CGM clock multiplexers (excluding MCXE315/MCXE316 and MCXE317)

Table 104. MC_CGM clock multiplexers (MCXE31B)

| Clock mux | Register description | Source inputs [1] | Register | Selector output | Divider output |
|---|---|---|---|---|---|
| Clock mux 0 | Select Control | FIRC_CLK<br>PLL_PHI0_CLK | MUX_0_CSC | — | — |
| | Select Status | | MUX_0_CSS | | |
| | Divider 0 Control | — | MUX_0_DC_0 | | CORE_CLK |
| | Divider 1 Control | | MUX_0_DC_1 | | AIPS_PLAT_CLK |
| | Divider 2 Control | | MUX_0_DC_2 | | AIPS_SLOW_CLK |
| | Divider 3 Control | | MUX_0_DC_3 | | HSE_CLK |
| | Divider 4 Control | | MUX_0_DC_4 | | DCM_CLK |
| | Divider 5 Control | | MUX_0_DC_5 | | LBIST_CLK |
| | Divider 6 Control | | MUX_0_DC_6 | | QSPI_MEM_CLK |
| Clock mux 1 | Select Control | FIRC_CLK<br>FXOSC_CLK<br>AIPS_PLAT_CLK | MUX_1_CSC | — | — |
| | Select Status | | MUX_1_CSS | | |
| Clock mux 2 | Select Control | FIRC_CLK<br>FXOSC_CLK<br>AIPS_PLAT_CLK | MUX_2_CSC | — | — |
| | Select Status | | MUX_2_CSS | | |
| Clock mux 3 | Divider 0 Control | — | MUX_3_DC_0 | — | FLEXCAN0_PE_CLK<br>FLEXCAN1_PE_CLK<br>FLEXCAN2_PE_CLK |

*Table continues on the next page...*

Table 104. MC_CGM clock multiplexers (MCXE31B) (continued)

| Clock mux | Register description | Source inputs [1] | Register | Selector output | Divider output |
|---|---|---|---|---|---|
| Clock mux 4 | Select Control | FIRC_CLK<br>FXOSC_CLK<br>AIPS_PLAT_CLK | MUX_4_CSC | — | — |
| | Select Status | | MUX_4_CSS | | |
| | Divider 0 Control | — | MUX_4_DC_0 | | FLEXCAN3_PE_CLK |
| Clock mux 5 | Select Control | FIRC_CLK<br>SIRC_CLK<br>FXOSC_CLK<br>SXOSC_CLK<br>AIPS_SLOW_CLK | MUX_5_CSC | — | — |
| | Select Status | | MUX_5_CSS | | |
| | Divider 0 Control | — | MUX_5_DC_0 | | CLKOUT_STANDBY |
| Clock mux 6 | Select Control | FIRC_CLK<br>SIRC_CLK<br>FXOSC_CLK<br>SXOSC_CLK<br>PLL_PHI0_CLK<br>PLL_PHI1_CLK<br><br>CORE_CLK<br>HSE_CLK<br>AIPS_PLAT_CLK<br>AIPS_SLOW_CLK<br>EMAC_RMII_TX_CLK<br>EMAC_RX_CLK | MUX_6_CSC | — | — |
| | Select Status | | MUX_6_CSS | | |
| | Divider 0 Control | — | MUX_6_DC_0 | | CLKOUT_RUN |
| Clock mux 7 | Select Control | FIRC_CLK<br>EMAC_RMII_MII_TX_CLK (pin)<br>EMAC_RX_CLK (pin) | MUX_7_CSC | — | — |
| | Select Status | | MUX_7_CSS | | |
| | Divider 0 Control | — | MUX_7_DC_0 | | EMAC_RX_CLK |
| Clock mux 8 | Select Control | FIRC_CLK<br>EMAC_RMII_MII_TX_CLK (pin) | MUX_8_CSC | — | — |
| | Select Status | | MUX_8_CSS | | |
| | Divider 0 Control | — | MUX_8_DC_0 | | EMAC_TX_CLK |
| Clock mux 9 | Select Control | FIRC_CLK<br>FXOSC_CLK<br>PLL_PHI0_CLK<br>EMAC_RMII_TX_CLK<br>EMAC_RX_CLK | MUX_9_CSC | — | — |
| | Select Status | | MUX_9_CSS | | |

*Table continues on the next page...*

Table 104. MC_CGM clock multiplexers (MCXE31B) (continued)

| Clock mux | Register description | Source inputs [1] | Register | Selector output | Divider output |
|---|---|---|---|---|---|
| | Divider 0 Control | — | MUX_9_DC_0 | | EMAC_TS_CLK |
| Clock mux 10 | Select Control | FIRC_CLK<br>FXOSC_CLK<br>PLL_PHI1_CLK | MUX_10_CSC | — | — |
| | Select Status | | MUX_10_CSS | | |
| | Divider 0 Control | — | MUX_10_DC_0 | | QSPI_SFCK |
| Clock mux 11 | Select Control | FIRC_CLK<br>FXOSC_CLK<br>PLL_PHI0_CLK<br>PLL_PHI1_CLK | MUX_11_CSC | — | — |
| | Select Status | | MUX_11_CSS | | |
| | Divider 0 Control | — | MUX_11_DC_0 | | TRACE_CLK |

1. The default clock selected for all clock mux selectors is FIRC_CLK (out of reset).

### 23.5.4 MC_CGM clock sources mapping

Table 105. MC_CGM clock sources mapping

| Clock selector index [1] | MC_CGM clock source | Clock source |
|---|---|---|
| 0 | clk_src_0 | FIRC_CLK |
| 1 | clk_src_1 | SIRC_CLK |
| 2 | clk_src_2 | FXOSC_CLK |
| 3 | Reserved | Reserved |
| 4 | clk_src_4 | SXOSC_CLK [2] |
| 5–7 | Reserved | Reserved |
| 8 | clk_src_8 | PLL_PHI0_CLK |
| 9 | clk_src_9 | PLL_PHI1_CLK |
| 10–11 | Reserved | Reserved |
| 12 | clk_src_12 | PLL_AUX_PHI0_CLK |
| 13 | clk_src_13 | PLL_AUX_PHI1_CLK |
| 15 | Reserved | Reserved |
| 16 | clk_src_16 | CORE_CLK |
| 17–18 | Reserved | Reserved |
| 19 | clk_src_19 | HSE_CLK |
| 20–21 | Reserved | Reserved |
| 22 | clk_src_22 | AIPS_PLAT_CLK |
| 23 | clk_src_23 | AIPS_SLOW_CLK |

*Table continues on the next page...*

Table 105. MC_CGM clock sources mapping (continued)

| Clock selector index [1] | MC_CGM clock source | Clock source |
|---|---|---|
| 24 | clk_src_24 | EMAC_RMII_TX_CLK [3] |
| 25 | clk_src_25 | EMAC_RX_CLK [3] |
| 26–49 | Reserved | Reserved |
| 50 | clk_src_50 | CLKOUT_RUN |

1. All clock selector indexes not shown are reserved.
2. SXOSC_CLK is not available on the MCXE315/MCXE316.
3. EMAC_RMII_TX_CLK and EMAC_RX_CLK are not available on the MCXE315/MCXE316 or MCXE317.

## 23.6 Peripheral clocking

The module clocking diagrams for the peripherals are shown in the following subsections (see Peripheral clock gating for peripheral clock gating possibilities).

### 23.6.1 Module clocking

The following sections show how the chip modules use MODULE_CLK and REG_INTF_CLK to control their functionality.

#### 23.6.1.1 Communication modules

Figure 57 shows the REG_INTF_CLK and MODULE_CLK connections, and Table 106 shows the REG_INTF_CLK and MODULE_CLK signals used by these modules. Any module diagram that does not explicitly show a REG_INTF_CLK uses the same source for REG_INTF_CLK as used by MODULE_CLK.

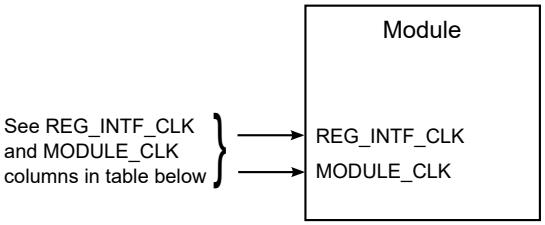



Figure 57. Communication module clocks

Table 106. Communication module clocking

| Module | MODULE_CLK | REG_INTF_CLK |
|---|---|---|
| LPSPI | See LPSPI*n* clocking. | |
| LPI$^2$C | See LPI2C_*n* clocking. | |
| FlexIO | See FlexIO clocking. | |
| FlexCAN | See FlexCAN_*n* clocking. | |
| SAI | See SAI_*n* clocking. | |
| EMAC | See EMAC clocking. | |
| LPUART_[0,8] | AIPS_PLAT_CLK | AIPS_PLAT_CLK |
| LPUART_[1:7,9:15] | AIPS_SLOW_CLK | AIPS_SLOW_CLK |
| QuadSPI | See QuadSPI clocking. | |

#### 23.6.1.1.1 FlexCAN_*n* clocking

The figure below shows the FlexCAN_*n* clocking.

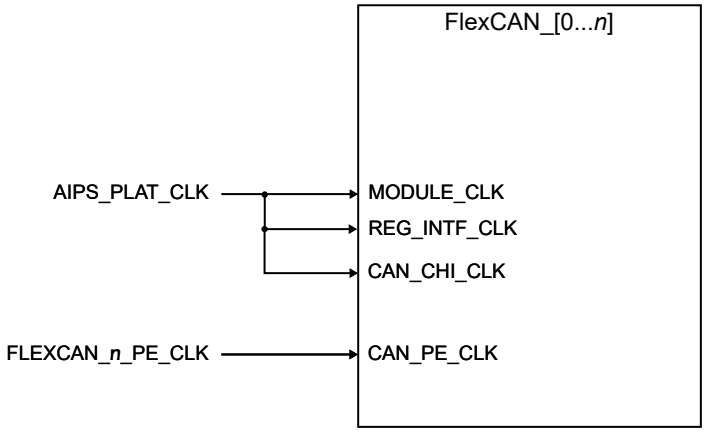



**Figure 58. FlexCAN_*n* clocking**

FlexCAN has the following unique clocks:

- CAN_CHI_CLK—FlexCAN controller host interface clock
- CAN_PE_CLK—FlexCAN protocol engine clock

FlexCAN supports up to an 8 Mbps data rate using a 40 MHz CAN_PE_CLK. With a 16 MHz crystal source, 3.2 Mbps is achievable for baud rate calculations (see the section Protocol timing in the "CAN (FlexCAN)" chapter for details).

For MCXE317, the maximum data rate supported in FlexCAN_0 instance is 5.7 Mbps.

**NOTE**

See Feature comparison for details on this module's availability on your chip variant.

##### 23.6.1.1.1.1 FlexCAN timestamp implementation

The following figure shows the FlexCAN timestamping implementation. The related table shows the timestamp sources and corresponding clock nodes.

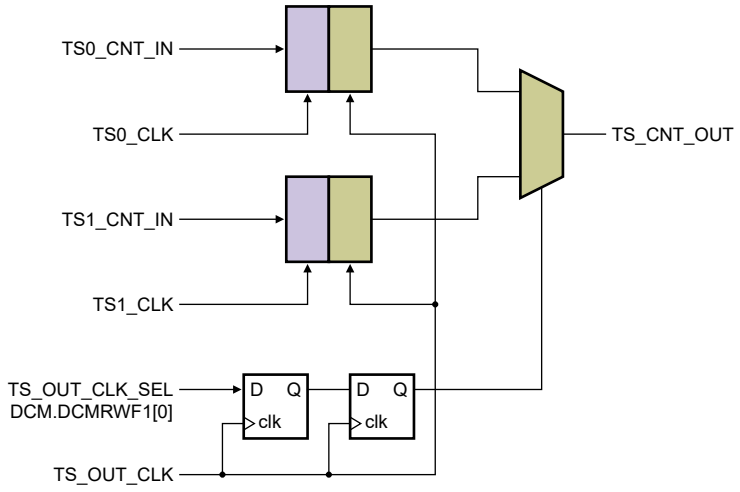



**Figure 59. FlexCAN timestamping**

Table 107. Timestamp sources and clock nodes

| Timestamp | Module | TS clock domain |
|---|---|---|
| TS0_CLK | EMAC | TS0_CLK—EMAC_TS_CLK |
| TS1_CLK | STM0 | TS1_CLK—AIPS_PLAT_CLK |
| TS_OUT_CLK | FlexCAN_*n* | TS_OUT_CLK—EMAC_TS_CLK |

**NOTE**

The timestamp clock (TS_OUT_CLK, EMAC_TS_CLK for MCXE31) must be greater than or equal to FLEXCAN_*n*_PE_CLK. When using STM0 as the timestamp source, the FlexCAN timestamp clock (TS_OUT_CLK, EMAC_TS_CLK for MCXE31) must be greater than or equal to STM0_CLK.

#### 23.6.1.1.2 LPI2C_*n* clocking

The following figure shows LPI2C_*n* clocking, and the related table shows the LPI2C SIUL2 configuration.

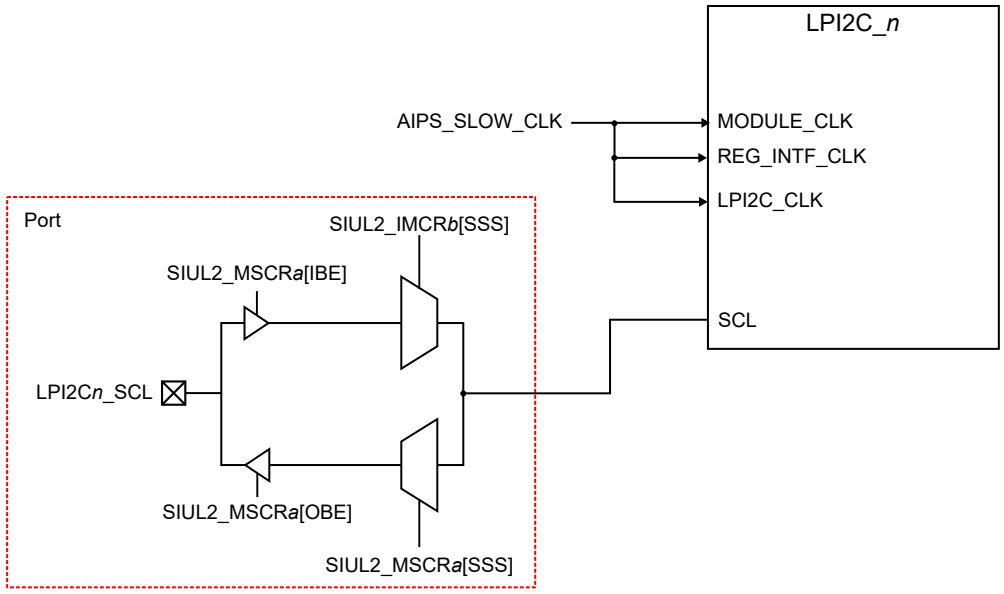



Figure 60. LPI2C clocking

**NOTE**

See Feature comparison for details on this module's availability on your chip variant.

Table 108. SIUL2 options for LPI2C*n*_SCL clock

| $I^2C$_*n* signal | $I^2C$ mode | I/O signal | Port | MSCR*a* | MSCR fields | | | IMCR*b* | IMCR[SSS] |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | OBE | IBE | SSS | | |
| I2C_0_SCL | Slave | LPI2C0_SCL | PTC8 | 72 | 0 | 1 | X | 212 | 0001b |
| | Master | | | | 1 | 0 | 001b | | X |
| | Slave | | PTD14 | 110 | 0 | 1 | X | | 0010b |
| | Master | | | | 1 | 0 | 100b | | X |

*Table continues on the next page...*

Table 108. SIUL2 options for LPI2C*n*_SCL clock (continued)

| I²C_*n* signal | I²C mode | I/O signal | Port | MSCR*a* | MSCR fields | | | IMCR*b* | IMCR[SSS] |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | OBE | IBE | SSS | | |
| | Slave | | PTF20[1] | 180 | 0 | 1 | X | | 0011b |
| | Master | | | | 1 | 0 | 100b | | X |
| I2C_0_SCLS | Slave | LPI2C0_SCLS | PTB1 | 33 | 0 | 1 | X | 213 | 0001b |
| | Master | | | | 1 | 0 | 001b | | X |
| | | | | | | | | | |
| I2C_1_SCL | Slave | LPI2C1_SCL | PTC7 | 71 | 0 | 1 | X | 217 | 0001b |
| | Master | | | | 1 | 0 | 011b | | X |
| | Slave | | PTC15 | 79 | 0 | 1 | X | | 0110b |
| | Master | | | | 1 | 0 | 111b | | X |
| | Slave | | PTC28 | 92 | 0 | 1 | X | | 0100b |
| | Master | | | | 1 | 0 | 101b | | X |
| | Slave | | PTD9 | 105 | 0 | 1 | X | | 0010b |
| | Master | | | | 1 | 0 | 010b | | X |
| | Slave | | PTF7[2] | 167 | 0 | 1 | X | | 0101b |
| | Master | | | | 1 | 0 | 011b | | X |
| I2C_1_SCLS | Slave | LPI2C1_SCLS | PTC17 | 81 | 0 | 1 | X | 218 | 0010b |
| | Master | | | | 1 | 0 | 100b | | X |

1. PTF20 pad is not present in MCXE315/MCXE316 and MCXE317
2. PTF7 pad is not present in MCXE315/MCXE316 and MCXE317.

#### 23.6.1.1.3 EMAC clocking

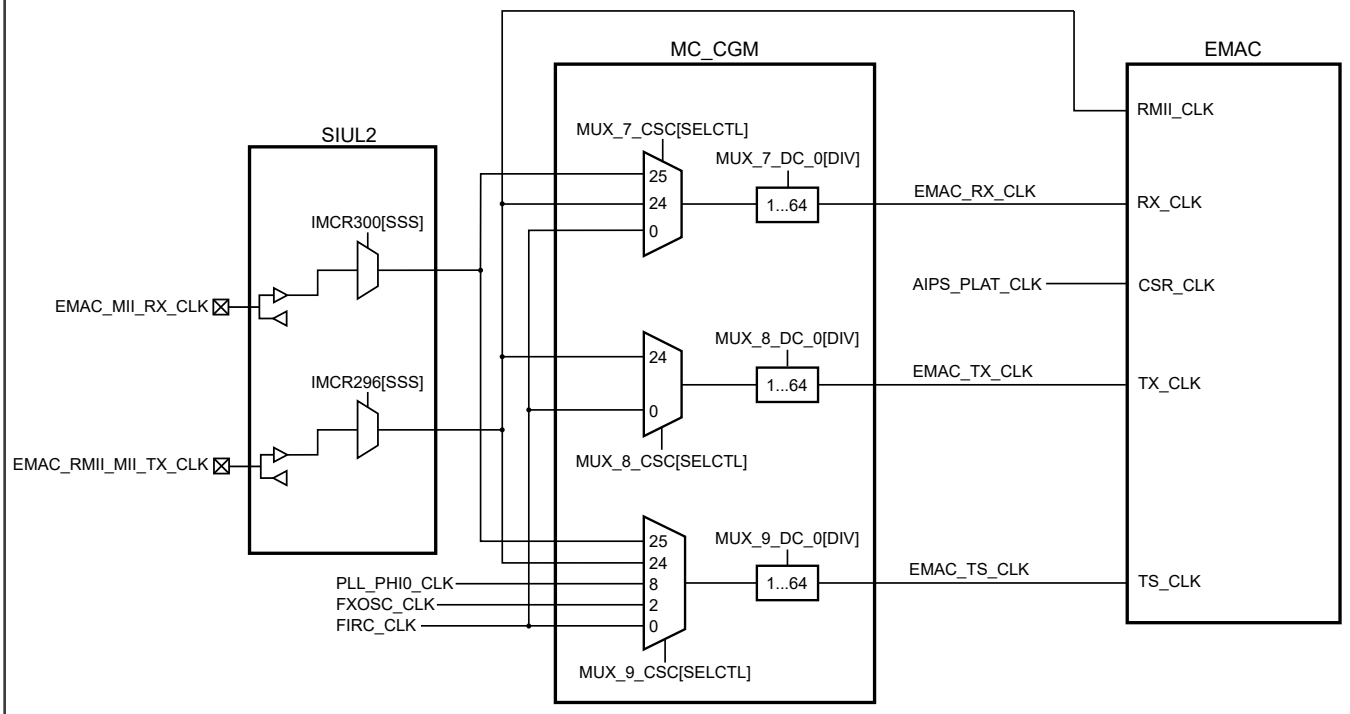



Figure 61. EMAC clocking

**NOTE**

EMAC operates only in Clock options A and B, since the module clock becomes lower than the protocol clock (RMII/MII clocks) in other modes.

**NOTE**

See Feature comparison for details on this module's availability on your chip variant.

#### 23.6.1.1.3.1 EMAC RMII clocking

The following table shows the EMAC RMII clocking, and the related table shows the SIUL2 clock signal configuration for RMII.

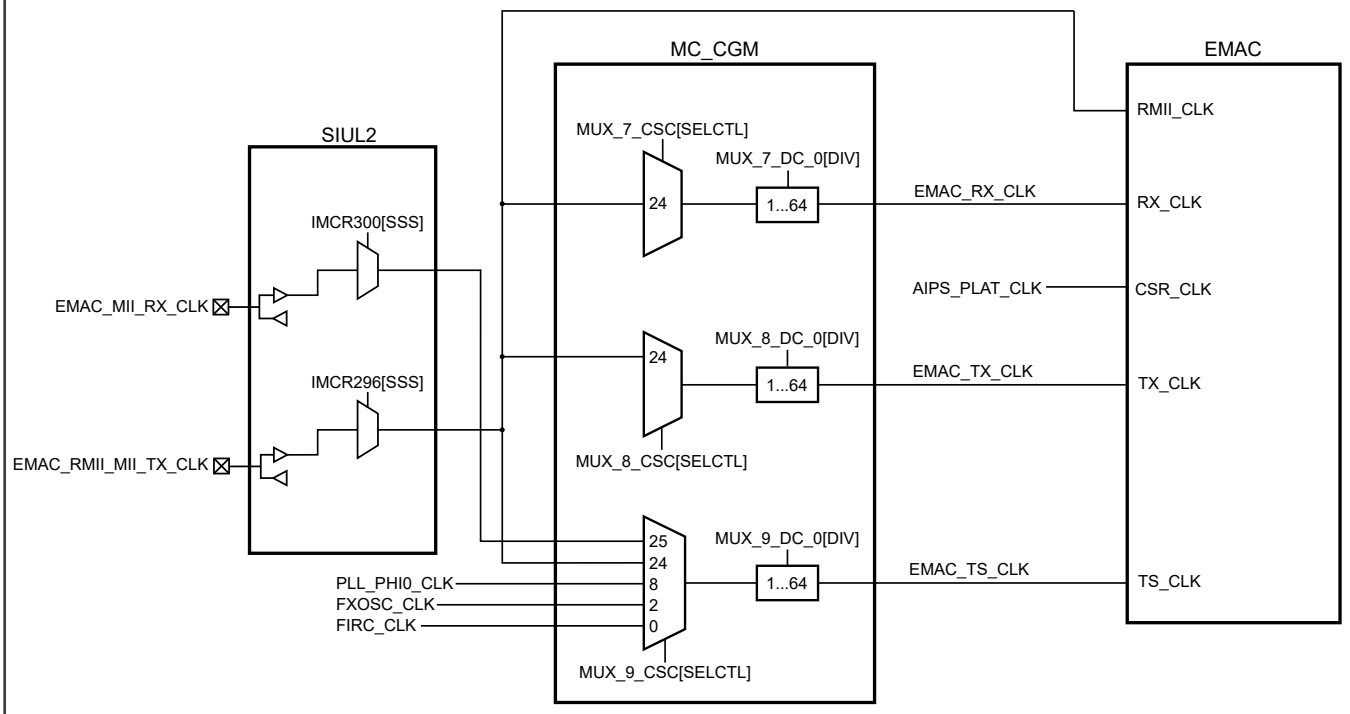



Figure 62. EMAC RMII clocking

Table 109. EMAC RMII clock configuration

| Source clock | Destination clock | Port | SIUL2 | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | MSCR*a* | MSCR fields | | | IMCR*b* | IMCR[SSS] |
| | | | | OBE | IBE | SSS | | |
| EMAC_RMII_MII_TX_CLK | EMAC_RX_CLK | PTC0 | 64 | 0 | 1 | X | 296 | 0100b |
| | EMAC_TX_CLK | | | | | | | |
| | EMAC_TX_CLK | PTD6 | 102 | 0 | 1 | X | | 0010b |
| | EMAC_RX_CLK | | | | | | | |
| | EMAC_TX_CLK | PTD11 | 107 | 0 | 1 | X | | 0001b |
| | EMAC_RX_CLK | | | | | | | |
| | EMAC_TX_CLK | PTD12 | 108 | 0 | 1 | X | | 0011b |
| | EMAC_RX_CLK | | | | | | | |
| | | | | | | | | |
| EMAC_RMII_MII_TX_CLK (MC_CGM.MUX_9_CSC[SELCTL] = 18h) | EMAC_TS_CLK | PTC0 | 64 | 0 | 1 | X | 296 | 0100b |
| | | PTD6 | 102 | 0 | 1 | X | | 0010b |
| | | PTD11 | 107 | 0 | 1 | X | | 0001b |
| | | PTD12 | 108 | 0 | 1 | X | | 0011b |

*Table continues on the next page...*

Table 109. EMAC RMII clock configuration (continued)

| Source clock | Destination clock | Port | SIUL2 | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | MSCR*a* | MSCR fields | | | IMCR*b* | IMCR[SSS] |
| | | | | OBE | IBE | SSS | | |
| EMAC_MII_RX_CLK (MC_CGM.MUX_9_CSC[SELCTL] = 19h) | EMAC_TS_CLK | PTC1 | 65 | 0 | 1 | X | 300 | 0011b |
| | | PTD5 | 101 | 0 | 1 | X | | 0010b |
| | | PTD10 | 106 | 0 | 1 | X | | 0001b |

**NOTE**

The EMAC time-stamp clock (EMAC_TS_CLK) can use either of the source clocks show in the table above, but can also select other clocks as selected in MC_CGM.MUX_9_CSC[SELCTL].

### 23.6.1.1.3.2 EMAC MII clocking

The following figure shows the EMAC MII clocking, and the related table shows the SIUL2 clock signal configuration for MII.

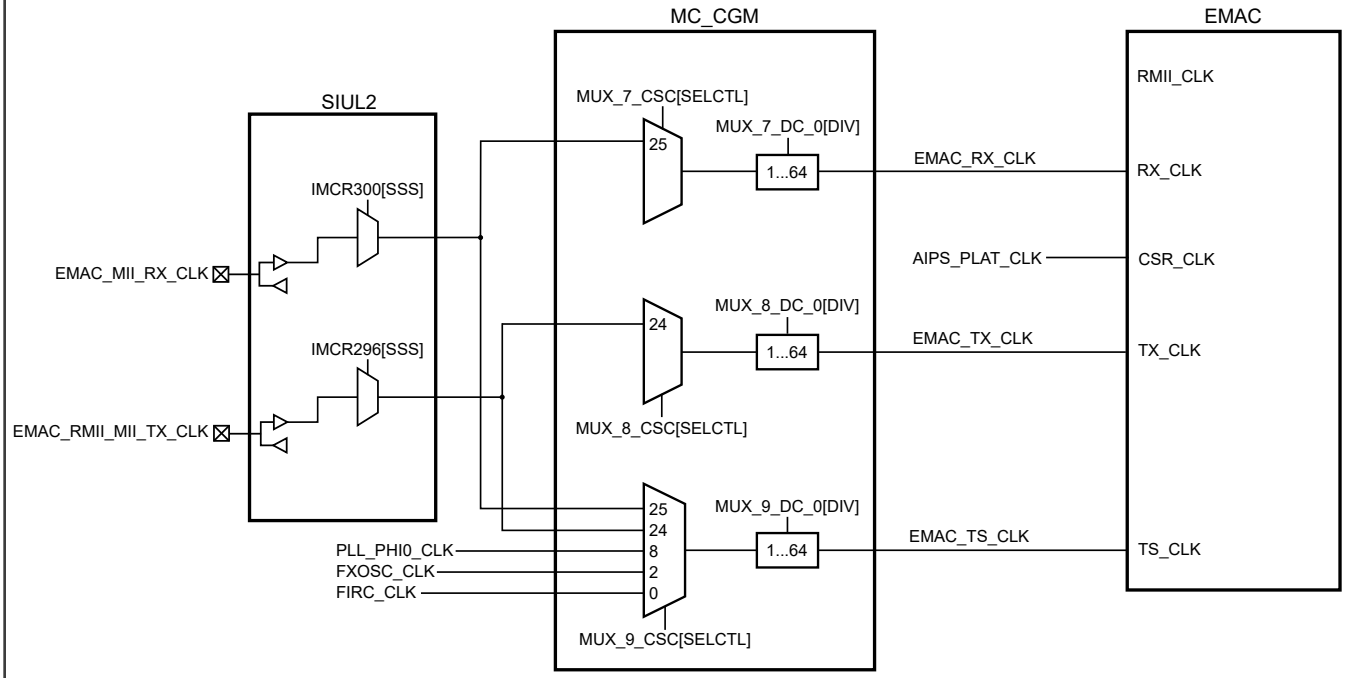



Figure 63. EMAC MII clocking

Table 110. SIUL2 EMAC MII clock configuration

| Source clock | Destination | Port | SIUL2 | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | MSCR*a* | MSCR fields | | | IMCR*b* | IMCR[SSS] |
| | | | | OBE | IBE | SSS | | |
| EMAC_RMII_MII_TX_CLK | EMAC_TX_CLK | PTC0 | 64 | 0 | 1 | X | 296 | 0100b |
| | | PTD6 | 102 | 0 | 1 | X | | 0010b |

*Table continues on the next page...*

Table 110. SIUL2 EMAC MII clock configuration (continued)

| Source clock | Destination | Port | SIUL2 | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | MSCR*a* | MSCR fields | | | IMCR*b* | IMCR[SSS] |
| | | | | OBE | IBE | SSS | | |
| | | PTD11 | 107 | 0 | 1 | X | | 0001b |
| | | PTD12 | 108 | 0 | 1 | X | | 0011b |
| | | | | | | | | |
| EMAC_MII_RX_CLK | EMAC_RX_CLK | PTC1 | 65 | 0 | 1 | X | | 0011b |
| | | PTD5 | 101 | 0 | 1 | X | 300 | 0010b |
| | | PTD10 | 106 | 0 | 1 | X | | 0001b |
| | | | | | | | | |
| EMAC_RMII_MII_TX_CLK (MC_CGM.MUX_9_CSC[SELCTL] = 18h) | EMAC_TS_CLK | PTC0 | 64 | 0 | 1 | X | | 0100b |
| | | PTD6 | 102 | 0 | 1 | X | | 0010b |
| | | PTD11 | 107 | 0 | 1 | X | 296 | 0001b |
| | | PTD12 | 108 | 0 | 1 | X | | 0011b |
| EMAC_MII_RX_CLK (MC_CGM.MUX_9_CSC[SELCTL] = 19h) | EMAC_TS_CLK | PTC1 | 65 | 0 | 1 | X | | 0011b |
| | | PTD5 | 101 | 0 | 1 | X | 300 | 0010b |
| | | PTD10 | 106 | 0 | 1 | X | | 0001b |

**NOTE**

The EMAC time-stamp clock (EMAC_TS_CLK) can use any of the signals shown in the figure above which includes either EMAC_RX_CLK or EMAC_RMII_MII_TX_CLK source clocks. MC_CGM.MUX_9_CSC[SELCTL] selects the specific source clock for EMAC_TS_CLK.

#### 23.6.1.1.4 LPSPI*n* clocking

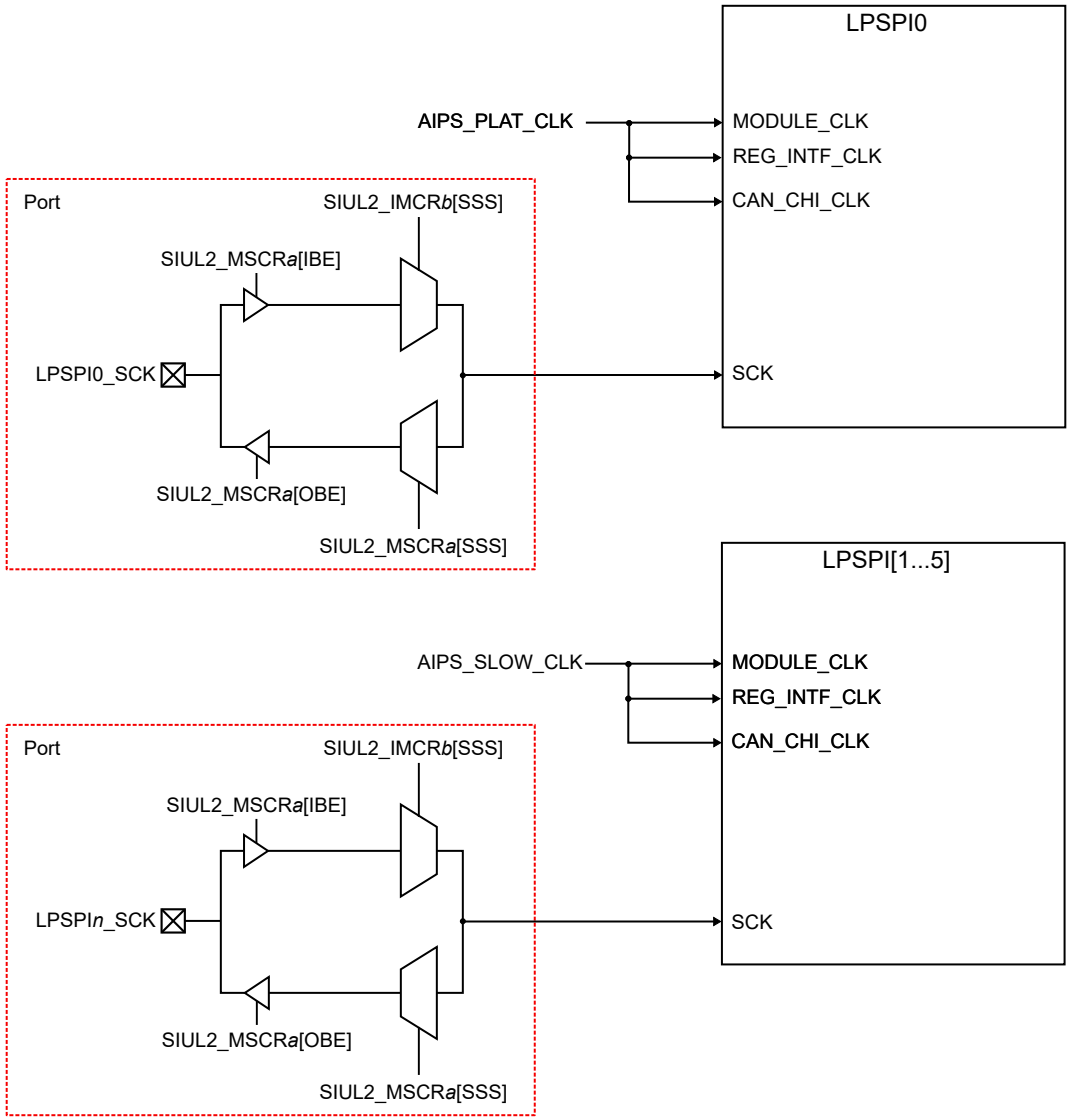



**Figure 64. LPSPI*n* clocking**

**NOTE**

See Feature comparison for details on this module's availability on your chip variant.

**Table 111. SIUL2 options for LPSPI*n* SCK**

| Source | Destination | SPI mode | Port | MSCR *a* | MSCR fields | | | IMCR *b* | IMCR[SSS] |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | OBE | IBE | SSS | | |
| LPSPI0_SCK | LPSPI0 SCK | Slave | PTC8 | 72 | 0 | 1 | X | 229 | 0001b |
| | | Master | | | 1 | 0 | 0110b | | X |
| | | Slave | PTD11 | 107 | 0 | 1 | X | | 0101b |

*Table continues on the next page...*

Table 111. SIUL2 options for LPSPI*n* SCK (continued)

| Source | Destination | SPI mode | Port | MSCR *a* | MSCR fields | | | IMCR *b* | IMCR[SSS] |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | OBE | IBE | SSS | | |
| | | Master | | | 1 | 0 | 0110b | | X |
| | | Slave | PTD15 | 111 | 0 | 1 | X | | 0011b |
| | | Master | | | 1 | 0 | 0100b | | X |
| | | Slave | PTE1 | 129 | 0 | 1 | X | | 0010b |
| | | Master | | | 1 | 0 | 0010b | | X |
| | | | | | | | | | |
| LPSPI1_SCK | LPSPI1 SCK | Slave | PTA3 | 3 | 0 | 1 | X | 238 | 0001b |
| | | Master | | | 1 | 0 | 0011b | | X |
| | | Slave | PTA19 | 19 | 0 | 1 | X | | 0011b |
| | | Master | | | 1 | 0 | 0100b | | X |
| | | Slave | PTA28 | 28 | 0 | 1 | X | | 0100b |
| | | Master | | | 1 | 0 | 0011b | | X |
| | | Slave | PTB14 | 46 | 0 | 1 | X | | 0010b |
| | | Master | | | 1 | 0 | 0011b | | X |
| | | | | | | | | | |
| LPSPI2_SCK | LPSPI2 SCK | Slave | PTB29 | 61 | 0 | 1 | X | 245 | 0011b |
| | | Master | | | 1 | 0 | 0101b | | X |
| | | Slave | PTC15 | 79 | 0 | 1 | X | | 0010b |
| | | Master | | | 1 | 0 | 0011b | | X |
| | | Slave | PTE15 | 143 | 0 | 1 | X | | 0001b |
| | | Master | | | 1 | 0 | 0011b | | X |
| | | Slave | PTF0 | 160 | 0 | 1 | X | | 0100b |
| | | Master | | | 1 | 0 | 011b | | X |
| | | | | | | | | | |
| LPSPI3_SCK | LPSPI3 SCK | Slave | PTC17 | 81 | 0 | 1 | X | 252 | 0011b |
| | | Master | | | 1 | 0 | 0001b | | X |
| | | Slave | PTD1 | 97 | 0 | 1 | X | | 0001b |
| | | Master | | | 1 | 0 | 0011b | | X |
| | | Slave | PTE7 | 135 | 0 | 1 | X | | 0010b |
| | | Master | | | 1 | 0 | 0110b | | X |
| | | Slave | PTF13 | 173 | 0 | 1 | X | | 0100b |
| | | Master | | | 1 | 0 | 0100b | | X |
| | | | | | | | | | |
| LPSPI4_SCK | LPSPI4 SCK | Slave | PTB10 | 42 | 0 | 1 | X | 259 | 0010b |

*Table continues on the next page...*

Table 111. SIUL2 options for LPSPI*n* SCK (continued)

| Source | Destination | SPI mode | Port | MSCR *a* | MSCR fields | | | IMCR *b* | IMCR[SSS] |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | OBE | IBE | SSS | | |
| | | Master | | | 1 | 0 | 0001b | | X |
| | | Slave | PTC27 | 91 | 0 | 1 | X | | 0001b |
| | | Master | | | 1 | 0 | 0111b | | X |
| | | Slave | PTE22 | 150 | 0 | 1 | X | | 0011b |
| | | Master | | | 1 | 0 | 0110b | | X |
| | | | | | | | | | |
| LPSPI5_SCK | LPSPI5 SCK | Slave | PTA3 | 3 | 0 | 1 | X | 266 | 0010b |
| | | Master | | | 1 | 0 | 0111b | | X |
| | | Slave | PTD14 | 110 | 0 | 1 | X | | 0001b |
| | | Master | | | 1 | 0 | 0001b | | X |
| | | Slave | PTD26 | 122 | 0 | 1 | X | | 0011b |
| | | Master | | | 1 | 0 | 0110b | | X |

#### 23.6.1.1.5 LPUART clocking

The following figure shows the LPUART clocking configuration, and the related table shows LPUART use case baud rates.

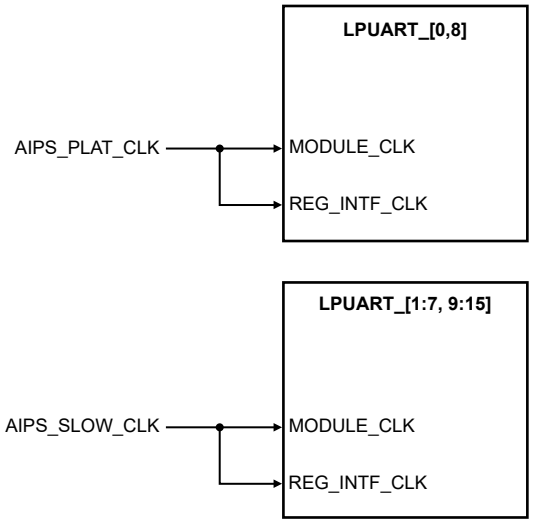



Figure 65. LPUART clocking

**NOTE**

See Feature comparison for details on this module's availability on your chip variant.

Table 112. LPUART baud rate calculation

| Required baud rate (bps) | LPUART_CLK (MHz) | OSR | SBR[12:0] | Calculated baud rate (bps) [1] |
|---|---|---|---|---|
| 8192 | 40 | 4 | 976 | 8196 |
| 8192 | 80 | 15 | 610 | 8196 |
| 8192 | 48 | 15 | 366 | 8196 |
| 115200 | 48 | 15 | 26 | 115384 |
| 115200 | 40 | 7 | 43 | 116279 |
| 19200 | 80 | 4 | 833 | 19207 |
| 19200 | 40 | 7 | 260 | 19230 |
| 38400 | 40 | 7 | 130 | 38461 |

1. MODULE_CLK ÷ (LPUART.BAUD[SBR] × (LPUART.BAUD[OSR] + 1))

#### 23.6.1.1.6 FlexIO clocking

The following figure shows the FlexIO clocking interface. The related two tables show the FlexIO baud rate use cases.

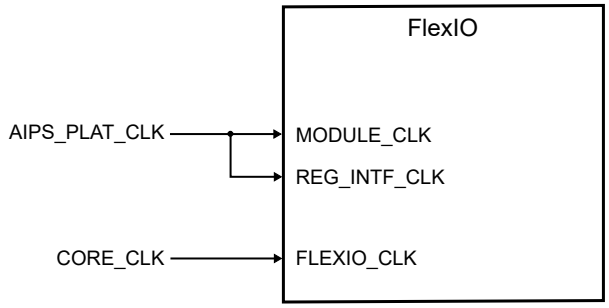



Figure 66. FlexIO clocking

**NOTE**

See Feature comparison for details on this module's availability on your chip variant.

Table 113. FlexIO baud rate calculation (FlexIO.TIMCFG*n*[TIMDEC] = 101b)

| FLEXIO_CLK (CORE_CLK) | Required baud rate | TIMCMP*n*[CMP] | | Theoretical baud rate [1] | Bit duration | Observed baud rate |
|---|---|---|---|---|---|---|
| | | Hex | Decimal | | | |
| 88 MHz | 9600 | 0010h | 16 | 10110.29 | 101.33 µs | 9868 |
| 88 MHz | 19200 | 0007h | 7 | 21484.37 | 47.44 µs | 21079 |
| 88 MHz | 57600 | 0001h | 1 | 85937.50 | 11.66 µs | 85763 |
| 88 MHz | 115200 | 0000h | 0 | 171875.00 | 5.88 µs | 170068 |

1. Theoretical baud rate = Frequency ÷ (256 × 2 × (TIMCMP*n*[CMP] + 1))

Table 114. FlexIO baud rate calculation (FlexIO.TIMCFG*n*[TIMDEC] = 100b)

| FLEXIO_CLK (CORE_CLK) | Required baud rate | TIMCMP*n*[CMP] | | Theoretical baud rate [1] | Bit duration | Observed baud rate |
|---|---|---|---|---|---|---|
| | | Hex | Decimal | | | |
| 88 MHz | 9600 | — | — | — | — | — |
| 88 MHz | 19200 | 8Eh | 142 | 19230.77 | 51.88 µs | 19275 |
| 88 MHz | 57600 | 2Eh | 46 | 58510.64 | 16.89 µs | 59206 |
| 88 MHz | 115200 | 16h | 22 | 119565.21 | 8.44 µs | 118483 |

### 23.6.1.1.7 QuadSPI clocking

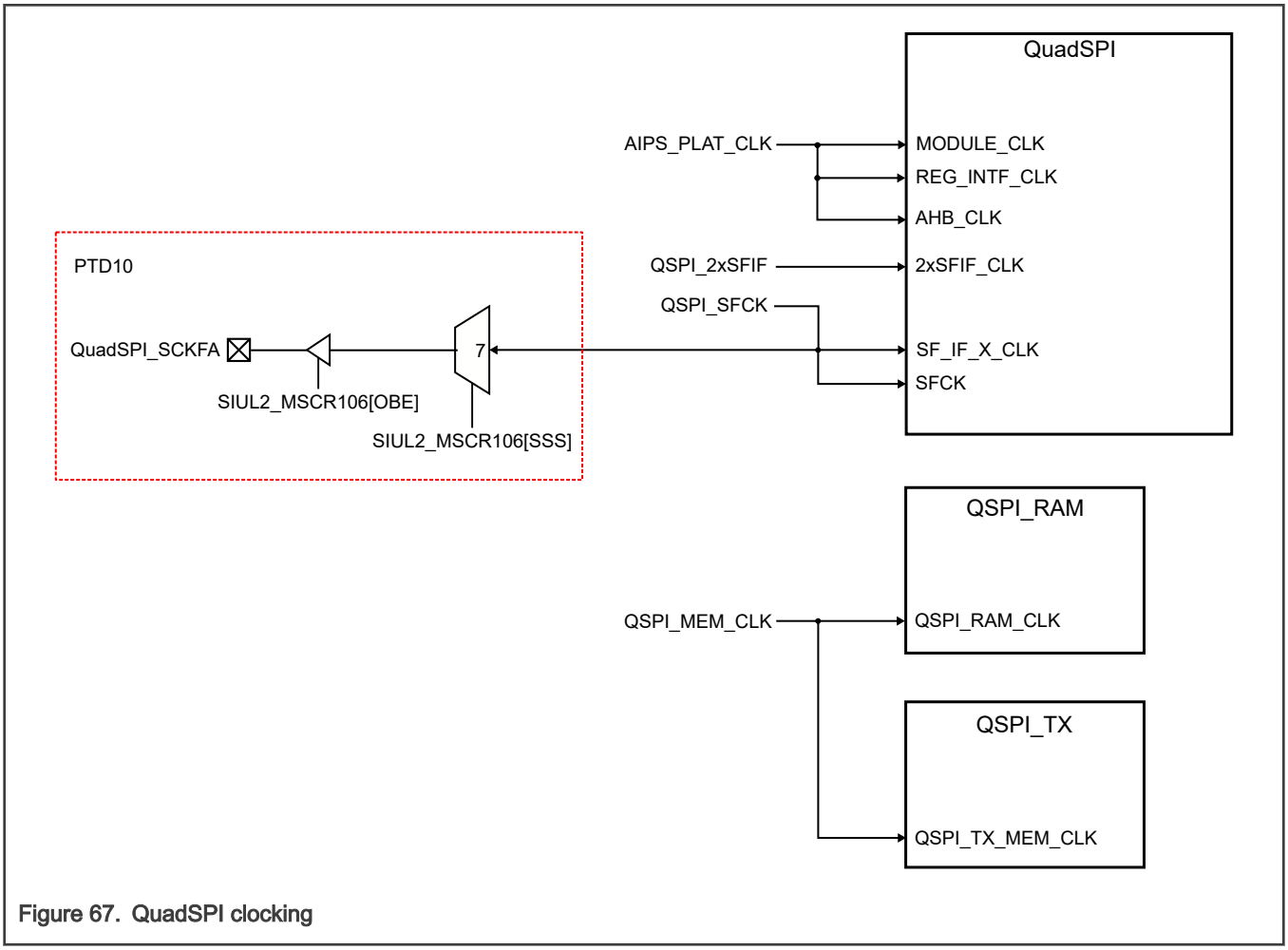



Figure 67. QuadSPI clocking

**NOTE**

See Feature comparison for details on this module's availability on your chip variant.

Table 115. QuadSPI clocking

| Clock option | AIPS_PLAT_CLK | QSPI_SFCK | QSPI_MEM_CLK |
|---|---|---|---|
| See Option A - High Performance mode (CORE_CLK @ 160 MHz) | 80 MHz | 120 MHz | 160 MHz |
| See Option B - Reduced Speed mode (CORE_CLK @ 120 MHz) | 60 MHz | 120 MHz or 80 MHz | 120 MHz |
| See Option F - Operation in 1:1 mode with CORE_CLK and AXBS_CLK at same speed | 80 MHz | 80 MHz | 160 MHz |

#### 23.6.1.1.8 SAI_*n* clocking

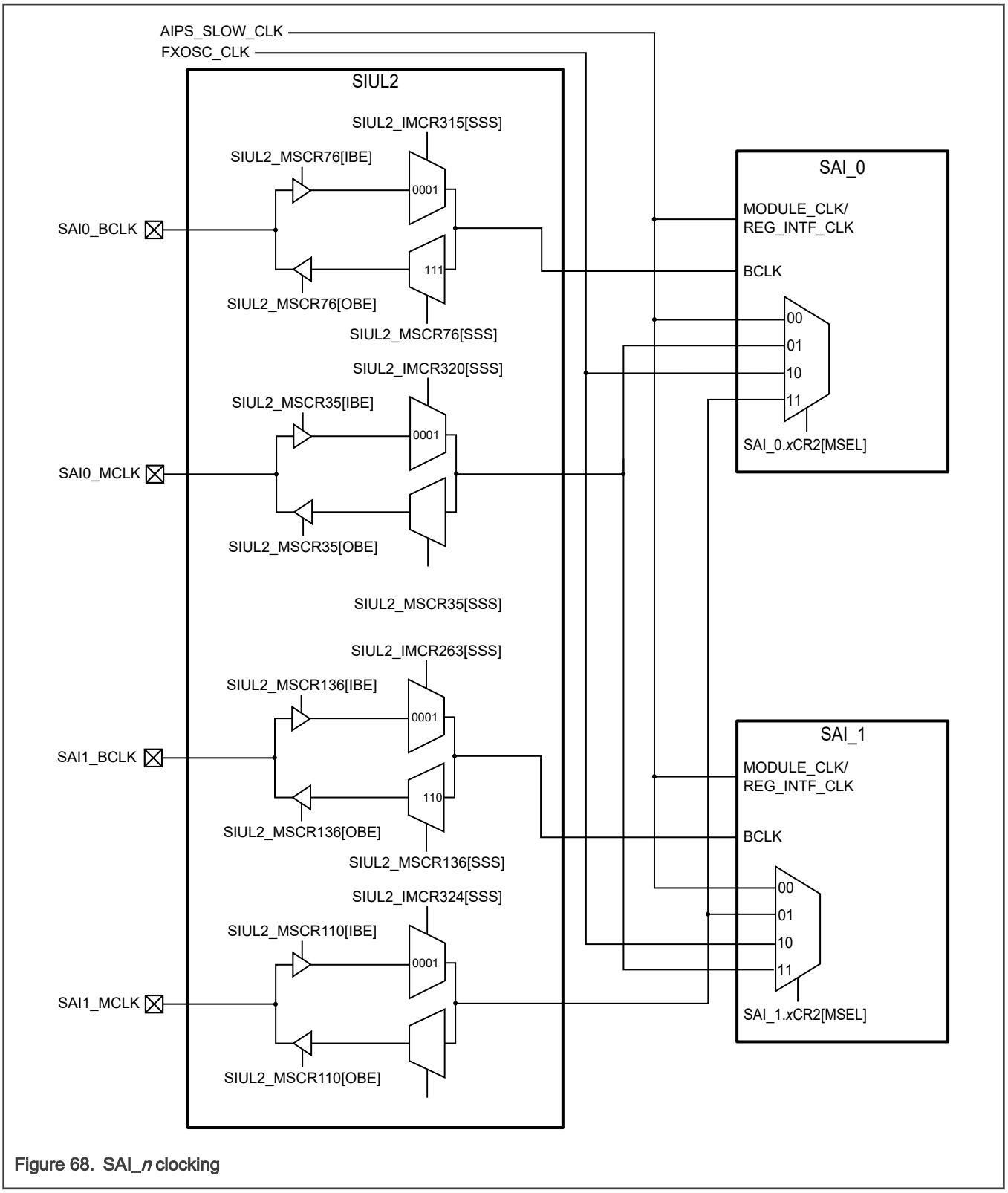



Figure 68. SAI_*n* clocking

**NOTE**

See Feature comparison for details on this module's availability on your chip variant.

Table 116. SIUL2 options for SAI_*n*

| Source | Destination | Port | MSCR *a* | MSCR fields | | | IMCR *b* | IMCR[SSS] |
|---|---|---|---|---|---|---|---|---|
| | | | | OBE | IBE | SSS | | |
| SAI0_MCLK | SAI_0.*x*CR2[MSEL] = 01b<br>SAI_1.*x*CR2[MSEL] = 11b | PTB3 | 35 | 0 | 1 | X | 320 | 0001b |
| | | | | — | — | Not available | | X |
| SAI1_MCLK | SAI_0.*x*CR2[MSEL] = 11b<br>SAI_1.*x*CR2[MSEL] = 01b | PTD14 | 110 | 0 | 1 | X | 324 | 0001b |
| | | | | 1 | 0 | Not available | | X |
| SAI0_BCLK | SAI0 BCLK | PTC12 | 76 | 0 | 1 | X | 315 | 0001b |
| SAI0 BCLK | SAI0_BCLK | | | 1 | 0 | 111b | | X |
| SAI1_BCLK | SAI1 BCLK | PTE8 | 136 | 0 | 1 | X | 322 | 0001b |
| SAI1 BCLK | SAI1_BCLK | | | 1 | 0 | 110b | | X |

Internally generated MCLK is not supported on the MCXE31 chip family.

### 23.6.1.2 System modules

Figure 69 shows the REG_INTF_CLK and MODULE_CLK connections, and Table 117 shows the REG_INTF_CLK and MODULE_CLK signals used by these modules. Any module diagram that does not explicitly show a REG_INTF_CLK uses the same source for REG_INTF_CLK as used by MODULE_CLK.

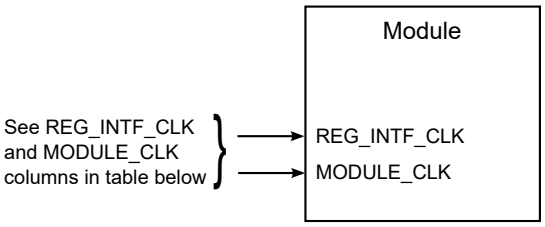



Figure 69. System module clocks

Table 117. System module clocking

| Module | MODULE_CLK | REG_INTF_CLK |
|---|---|---|
| MSCM | AIPS_PLAT_CLK | AIPS_PLAT_CLK |
| MCM | AIPS_SLOW_CLK | AIPS_SLOW_CLK |
| SIUL2 | See SIUL2 clocking. | |
| VIRT_WRAPPER | AIPS_SLOW_CLK | AIPS_SLOW_CLK |
| AXBS | CORE_CLK | AIPS_PLAT_CLK |

*Table continues on the next page...*

Table 117. System module clocking (continued)

| Module | MODULE_CLK | REG_INTF_CLK |
|---|---|---|
| DMAMUX | CORE_CLK | CORE_CLK |
| eDMA | CORE_CLK | AIPS_PLAT_CLK |
| INTM | AIPS_PLAT_CLK | AIPS_PLAT_CLK |
| SEMA42 | AIPS_PLAT_CLK | AIPS_PLAT_CLK |
| XBIC | CORE_CLK | AIPS_PLAT_CLK |
| XRDC | CORE_CLK | AIPS_PLAT_CLK |

#### 23.6.1.2.1 SIUL2 clocking

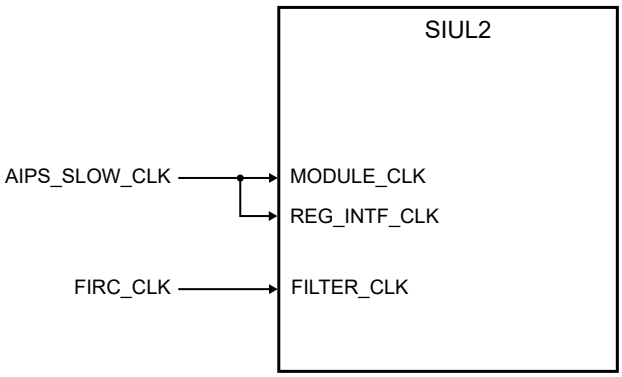



Figure 70. SIUL2 clocking

### 23.6.1.3 Clocking modules

Figure 71 shows the REG_INTF_CLK and MODULE_CLK connections, and Table 118 shows the REG_INTF_CLK and MODULE_CLK signals used by these modules. Any module diagram that does not explicitly show a REG_INTF_CLK uses the same source for REG_INTF_CLK as used by MODULE_CLK.

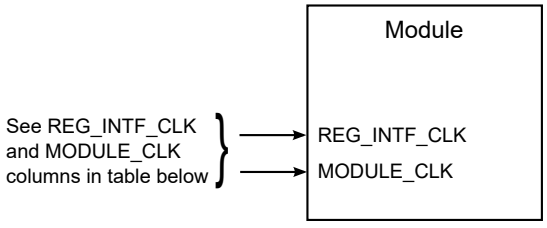



Figure 71. Clocking module clocks

Table 118. Clocking module clocking

| Module | MODULE_CLK | REG_INTF_CLK |
|---|---|---|
| FXOSC | See FXOSC clocking. | |
| SXOSC | See SXOSC clocking. | |

*Table continues on the next page...*

Table 118. Clocking module clocking (continued)

| Module | MODULE_CLK | REG_INTF_CLK |
|---|---|---|
| SIRC | See SIRC clocking. | |
| FIRC | See FIRC clocking. | |
| PLLDIG | See PLLDIG clocking. | |
| MC_CGM | — | AIPS_SLOW_CLK |

#### 23.6.1.3.1 FIRC clocking

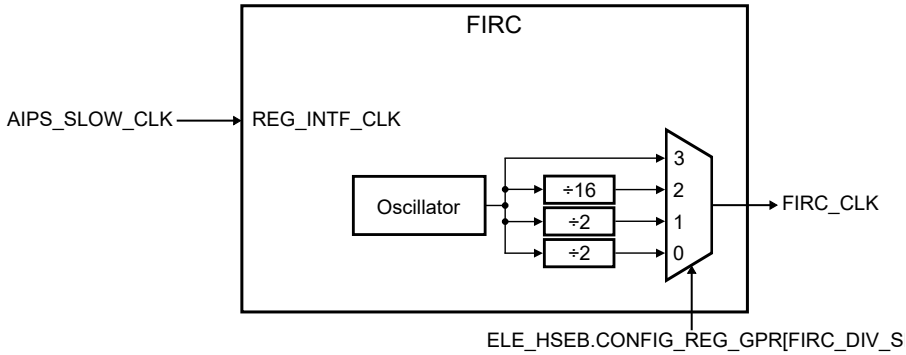



Figure 72. FIRC clocking

#### 23.6.1.3.2 SIRC clocking

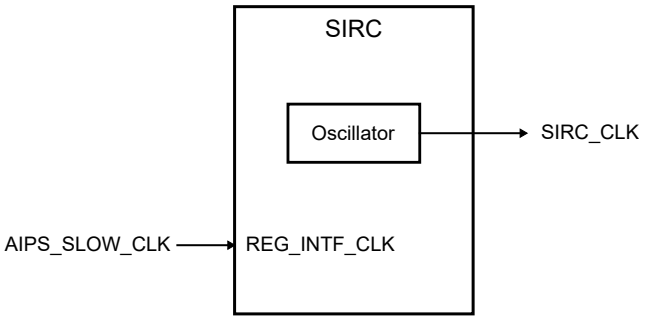



Figure 73. SIRC clocking

#### 23.6.1.3.3 FXOSC clocking

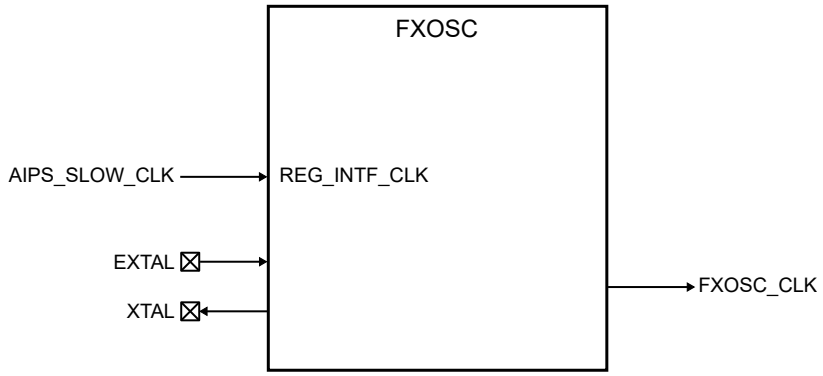



Figure 74. FXOSC clocking

#### 23.6.1.3.4 SXOSC clocking

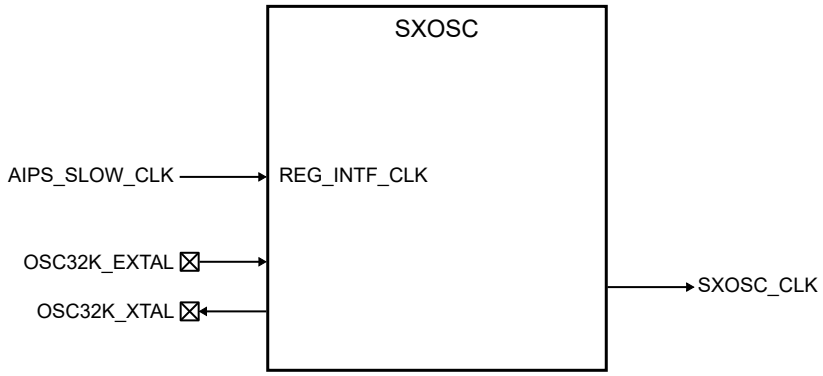



Figure 75. SXOSC clocking

#### 23.6.1.3.5 PLLDIG clocking

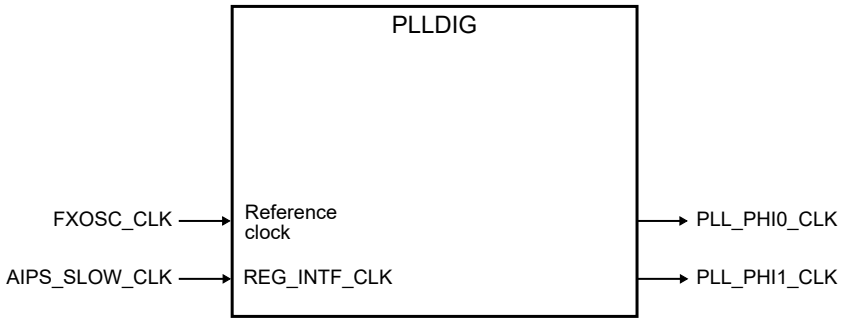



Figure 76. PLLDIG clocking

### 23.6.1.4 Reset modules

Figure 77 shows the REG_INTF_CLK and MODULE_CLK connections, and Table 119 shows the REG_INTF_CLK and MODULE_CLK signals used by these modules. Any module diagram that does not explicitly show a REG_INTF_CLK uses the same source for REG_INTF_CLK as used by MODULE_CLK.

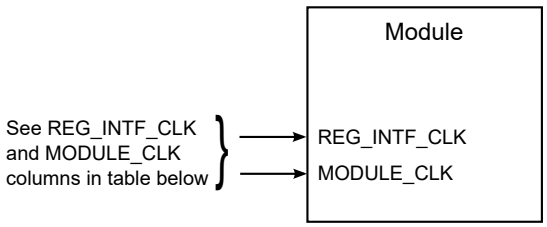



**Figure 77. Reset module clocks**

**Table 119. Reset module clocking**

| Module | MODULE_CLK | REG_INTF_CLK |
|---|---|---|
| MC_RGM | FIRC_CLK | FIRC_CLK |

### 23.6.1.5 Security modules

Figure 78 shows the REG_INTF_CLK and MODULE_CLK connections, and Table 120 shows the REG_INTF_CLK and MODULE_CLK signals used by these modules. Any module diagram that does not explicitly show a REG_INTF_CLK uses the same source for REG_INTF_CLK as used by MODULE_CLK.

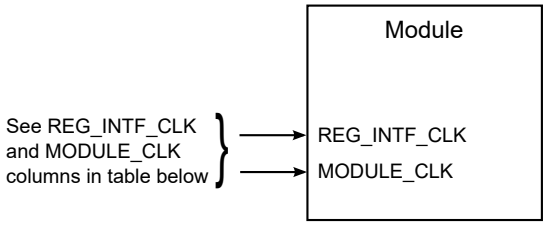



**Figure 78. Security module clocks**

**Table 120. Security module clocking**

| Module | MODULE_CLK | REG_INTF_CLK |
|---|---|---|
| ELE_HSEB | See ELE_HSEB clocking . | |
| MU_*n* | AIPS_SLOW_CLK | AIPS_SLOW_CLK |
| DCM | DCM_CLK | DCM_CLK |

#### 23.6.1.5.1 ELE_HSEB clocking

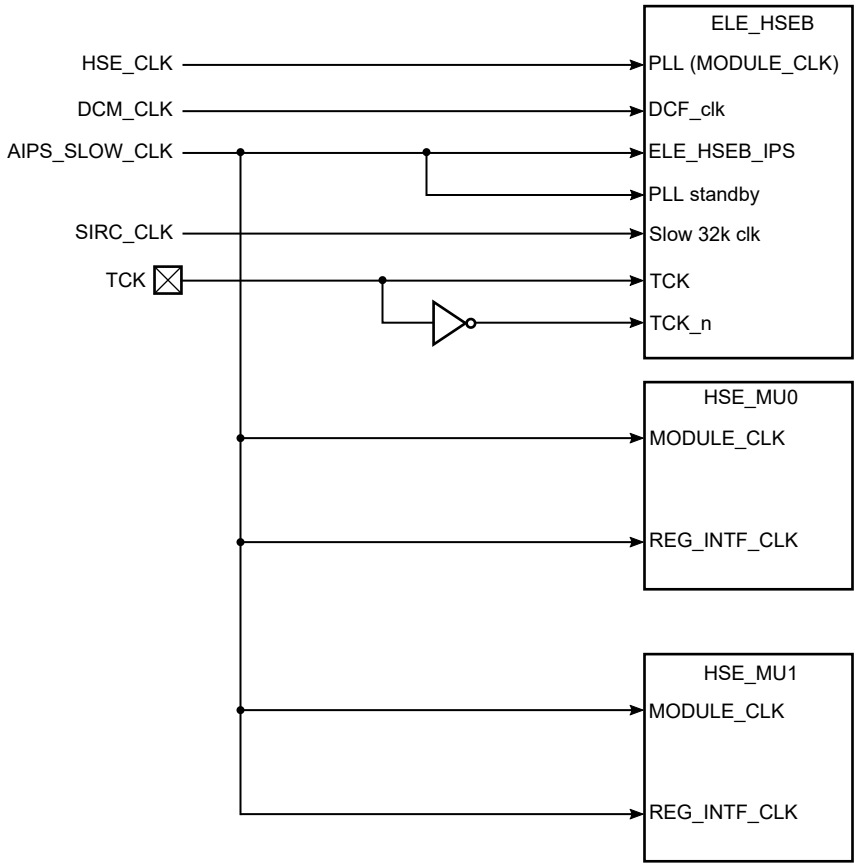



Figure 79. ELE_HSEB clocking

### 23.6.1.6 Power-management modules

Figure 80 shows the REG_INTF_CLK and MODULE_CLK connections, and Table 121 shows the REG_INTF_CLK and MODULE_CLK signals used by these modules. Any module diagram that does not explicitly show a REG_INTF_CLK uses the same source for REG_INTF_CLK as used by MODULE_CLK.

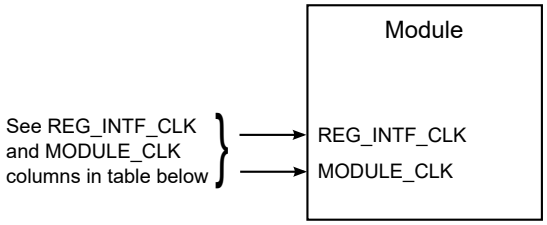



Figure 80. Power-management module clocks

Table 121. Power-management module clocking

| Module | MODULE_CLK | REG_INTF_CLK |
|---|---|---|
| PMC | AIPS_SLOW_CLK | AIPS_SLOW_CLK |
| MC_ME | AIPS_SLOW_CLK | AIPS_SLOW_CLK |
| MC_PCU | FIRC_CLK | FIRC_CLK |
| WKPU | AIPS_SLOW_CLK | AIPS_SLOW_CLK |

#### 23.6.1.7 Safety modules

Figure 81 shows the REG_INTF_CLK and MODULE_CLK connections, and Table 122 shows the REG_INTF_CLK and MODULE_CLK signals used by these modules. Any module diagram that does not explicitly show a REG_INTF_CLK uses the same source for REG_INTF_CLK as used by MODULE_CLK.

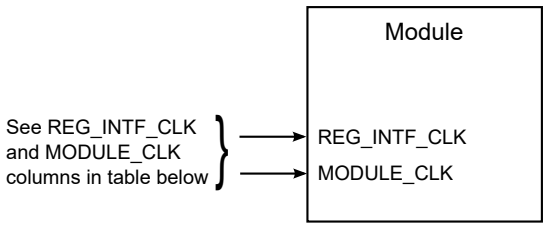



Figure 81. Safety module clocks

Table 122. Safety module clocking

| Module | MODULE_CLK | REG_INTF_CLK |
|---|---|---|
| EIM | AIPS_PLAT_CLK | AIPS_PLAT_CLK |
| ERM | See ERM clocking. | |
| FCCU | See FCCU clocking. | |
| STCU2 | See STCU2 clocking. | |
| REG_PROT | AIPS_SLOW_CLK | AIPS_SLOW_CLK |
| CMU_FC | AIPS_SLOW_CLK | AIPS_SLOW_CLK |
| CMU_FM | AIPS_SLOW_CLK | AIPS_SLOW_CLK |
| CRC | AIPS_PLAT_CLK | AIPS_PLAT_CLK |

#### 23.6.1.7.1 FCCU clocking

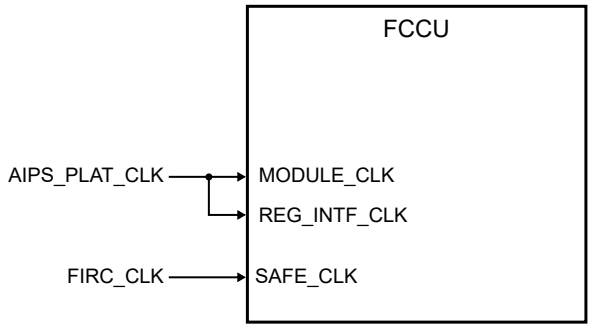



Figure 82. FCCU clocking

#### 23.6.1.7.2 STCU2 clocking

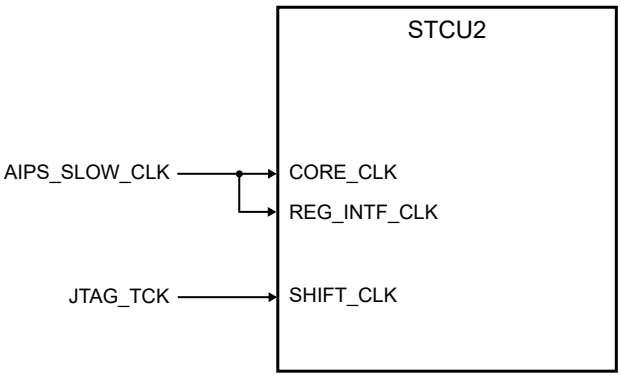



Figure 83. STCU2 clocking

#### 23.6.1.7.3 ERM clocking

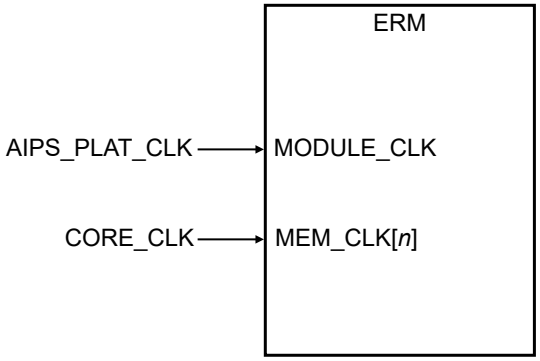



Figure 84. ERM clocking

**NOTE**

MEM_CLK[20:23] are not used. Source clock for MEM_CLK[0:19] is CORE_CLK.

#### 23.6.1.8 ADC and motor control modules

Figure 85 shows the REG_INTF_CLK and MODULE_CLK connections, and Table 123 shows the REG_INTF_CLK and MODULE_CLK signals used by these modules. Any module diagram that does not explicitly show a REG_INTF_CLK uses the same source for REG_INTF_CLK as used by MODULE_CLK.

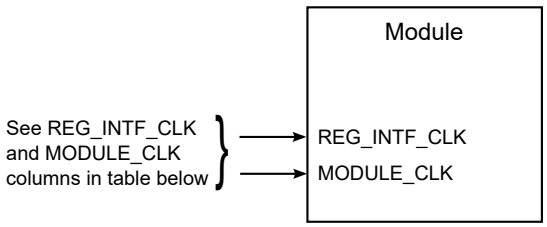



**Figure 85. Motor control module clocks**

**Table 123. Motor control module clocking**

| Module | MODULE_CLK | REG_INTF_CLK |
|---|---|---|
| ADC | See ADC_*n* clocking. | |
| LCU | CORE_CLK | |
| eMIOS | See eMIOS_*n* clocking. | |
| BCTU | See BCTU clocking. | |
| TRGMUX | AIPS_SLOW_CLK | |
| TSPC | AIPS_SLOW_CLK | |

##### 23.6.1.8.1 ADC_*n* clocking

The following figure shows ADC_*n* clocking configuration.

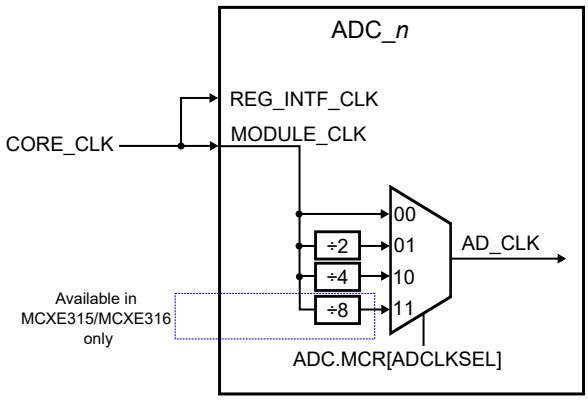



**Figure 86. ADC_*n* clocking**

**NOTE**

See Feature comparison for details on this module's availability on your chip variant.

The prescaler can be bypassed when using FIRC_CLK as the source (see MC_CGM mux 0 clocks for FIRC_CLK use details). The prescaler must be controlled such that the AD_CLK frequency is less than or equal to 120 MHz for MCXE317/MCXE315/MCXE316. For other MCXE31 products the frequency is 80 MHz.

The minimum operating speed of AD_CLK is 6 MHz using the following configuration:

1. Use FIRC_CLK (48 MHz) as clock source (MC_CGM.MUX_0_CSC[SELCTL] equal to 0000b).
2. Divide FIRC_CLK by 2 for CORE_CLK speed (MC_CGM.MUX_0_DC_0[DIV] equal to 1 (FIRC_CLK divide by 2 = 24 MHz)).
3. Write ADC.MCR[ADCCLKSEL] equal to 10b to divide the FIRC_CLK further by 4 (AD_CLK = 6 MHz).

However, at this lower speed, the ADC_*n* results will be degraded.

#### 23.6.1.8.2 eMIOS_*n* clocking

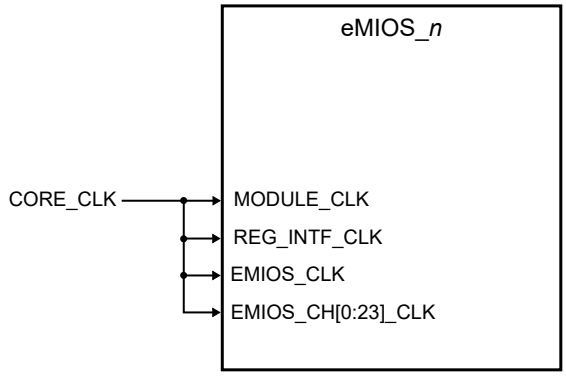



Figure 87. eMIOS_*n* clocking

**NOTE**

See Feature comparison for details on this module's availability on your chip variant.

#### 23.6.1.8.3 BCTU clocking

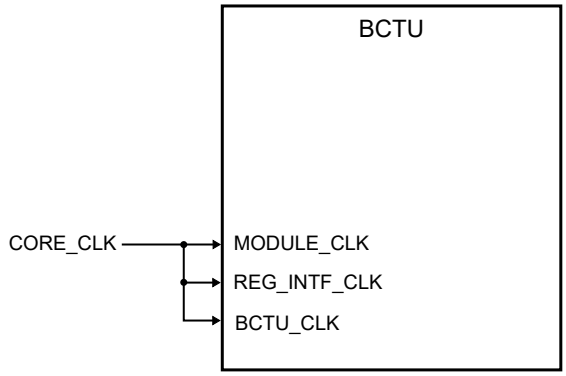



Figure 88. BCTU clocking

### 23.6.1.9 Timer modules

Figure 89 shows the REG_INTF_CLK and MODULE_CLK connections, and Table 124 shows the REG_INTF_CLK and MODULE_CLK signals used by these modules. Any module diagram that does not explicitly show a REG_INTF_CLK uses the same source for REG_INTF_CLK as used by MODULE_CLK.

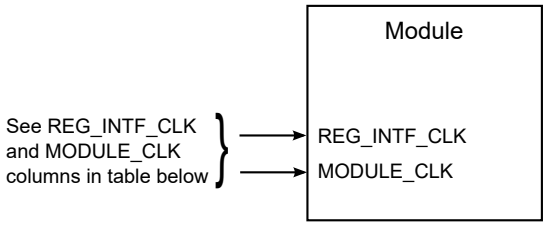



Figure 89. Timer module clocks

Table 124. Timer module clocking

| Module | MODULE_CLK | REG_INTF_CLK |
|---|---|---|
| PIT | See PIT_*n* clocking. | |
| SWT | See SWT_*n* clocking. | |
| STM_*n* | STM0_CLK | STM0_CLK |
| RTC | See RTC clocking. | |

#### 23.6.1.9.1 PIT_*n* clocking

The following figure shows the PIT_*n* clocking configuration. The related tables show the use case configuration.

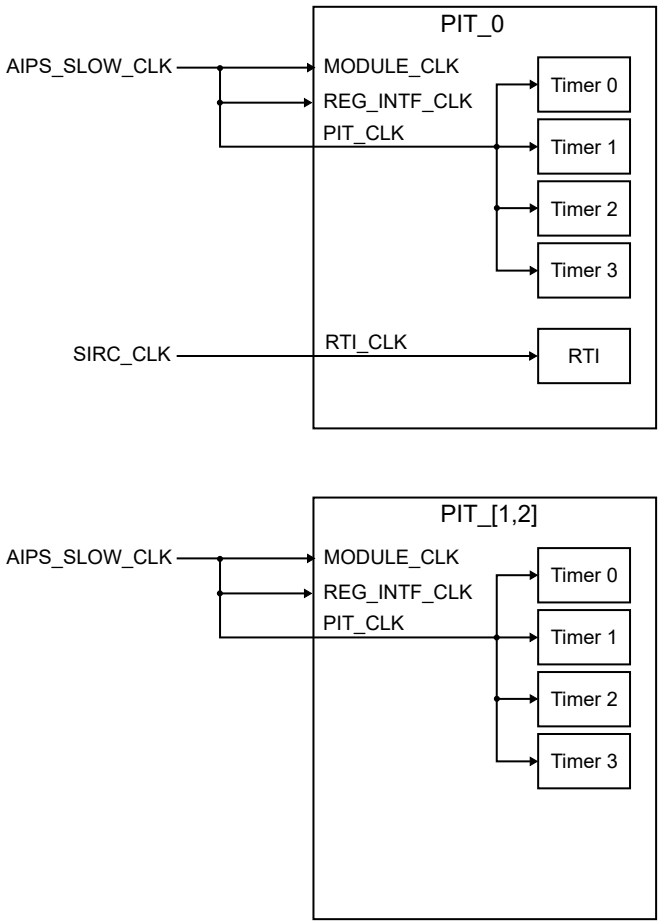



Figure 90. PIT_*n* clocking

**NOTE**

See Feature comparison for details on this module's availability on your chip variant.

Table 125. PIT_0 modes of operation

| MC_ME.PRTN0_COFB1_CLKEN[REQ44] | PIT_0.MCR[MDIS] | PIT_0.MCR[MDIS_RTI] | Mode | Application |
|---|---|---|---|---|
| 0 | X | X | PIT_0 clock gated (minimum power) | Module clock gated and unused / Standby mode, PIT and RTI unused |
| 1 | 0 | 0 | Both PIT and RTI enabled | Run mode |
| 1 | 0 | 1 | PIT running, RTI disabled | Run mode with only PIT active |
| 1 | 1 | 0 | PIT disabled, RTI enabled | Standby mode with RTI enabled |
| 1 | 1 | 1 | Both PIT and RTI disabled | Standby mode with RTI disabled |

Table 126. PIT_[1,2] modes of operation

| MC_ME.PRTN*n*_COFB1_CLKEN[REQ[45,63]] [1] | PIT_[1,2].MCR[MDIS] | Mode | Application |
|---|---|---|---|
| 0 | X | PIT_[1,2] clock gated (minimum power) | Module clock gated and unused, Standby mode |
| 1 | 0 | PIT_[1,2] enabled | Run mode |
| 1 | 0 | PIT_[1,2] running | Run mode with PIT active |
| 1 | 1 | PIT_[1,2] disabled | Standby mode |

1. PIT_1 and PIT_2 MC_ME partition registers used are:
   - PIT_1 - MC_ME.PRTN0_COFB1_CLKEN[REQ45]
   - PIT_2 - MC_ME.PRTN1_COFB1_CLKEN[REQ63]

### 23.6.1.9.2 SWT_*n* clocking

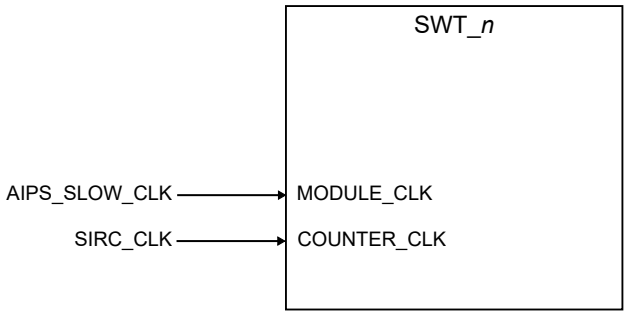



Figure 91. SWT_*n* clocking

**NOTE**

See Feature comparison for details on this module's availability on your chip variant.

#### 23.6.1.9.3 RTC clocking

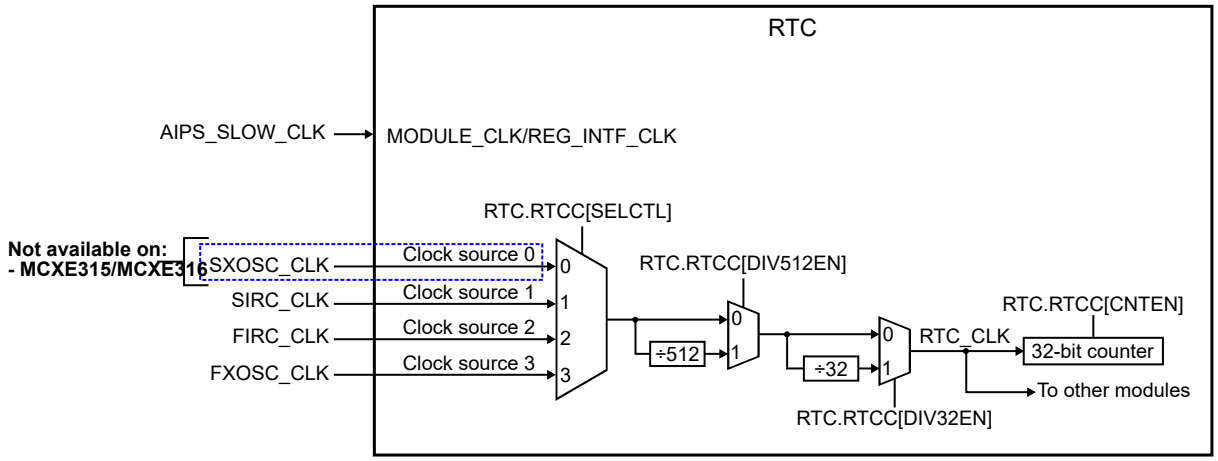



Figure 92. RTC clocking

**NOTE**

The RTC is available in Standby mode. Although bus clock is gated, the RTC can run on FIRC_CLK, SIRC_CLK, FXOSC_CLK, or SXOSC_CLK.

### 23.6.1.10 Debug modules

Figure 93 shows the REG_INTF_CLK and MODULE_CLK connections, and Table 127 shows the REG_INTF_CLK and MODULE_CLK signals used by these modules. Any module diagram that does not explicitly show a REG_INTF_CLK uses the same source for REG_INTF_CLK as used by MODULE_CLK.

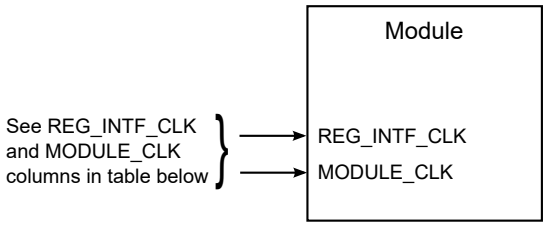



Figure 93. Debug module clocks

Table 127. Debug module clocking

| Module | MODULE_CLK | REG_INTF_CLK |
|---|---|---|
| JTAGC | See JTAGC clocking. | |
| JDC | See JDC clocking. | |

#### 23.6.1.10.1 JTAGC clocking

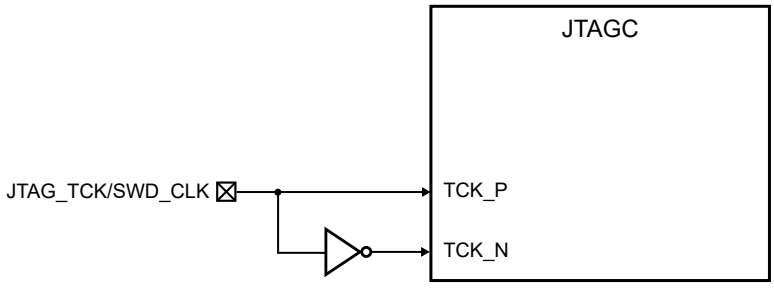



Figure 94. JTAGC clocking

#### 23.6.1.10.2 JDC clocking

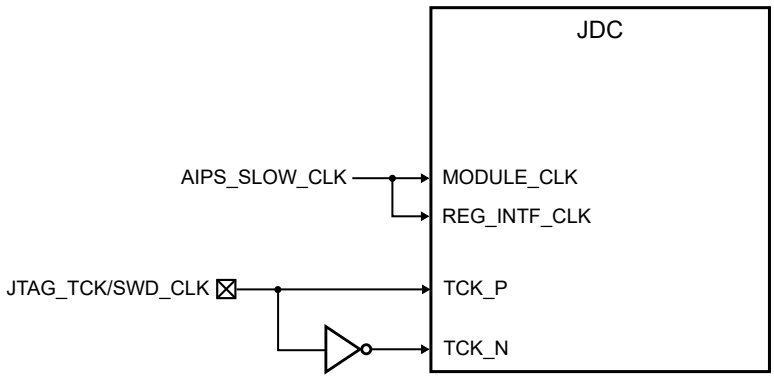



Figure 95. JDC clocking

### 23.6.1.11 Analog modules

Figure 96 shows the REG_INTF_CLK and MODULE_CLK connections, and Table 128 shows the REG_INTF_CLK and MODULE_CLK signals used by these modules. Any module diagram that does not explicitly show a REG_INTF_CLK uses the same source for REG_INTF_CLK as used by MODULE_CLK.

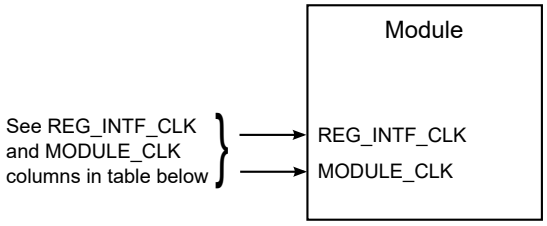



Figure 96. Analog module clocks

Table 128. Analog module clocking

| Module | MODULE_CLK | REG_INTF_CLK |
|---|---|---|
| LPCMP | See LPCMP_*n* clocking. | |
| Temperature Sensor | AIPS_SLOW_CLK | AIPS_SLOW_CLK |

#### 23.6.1.11.1 LPCMP_*n* clocking

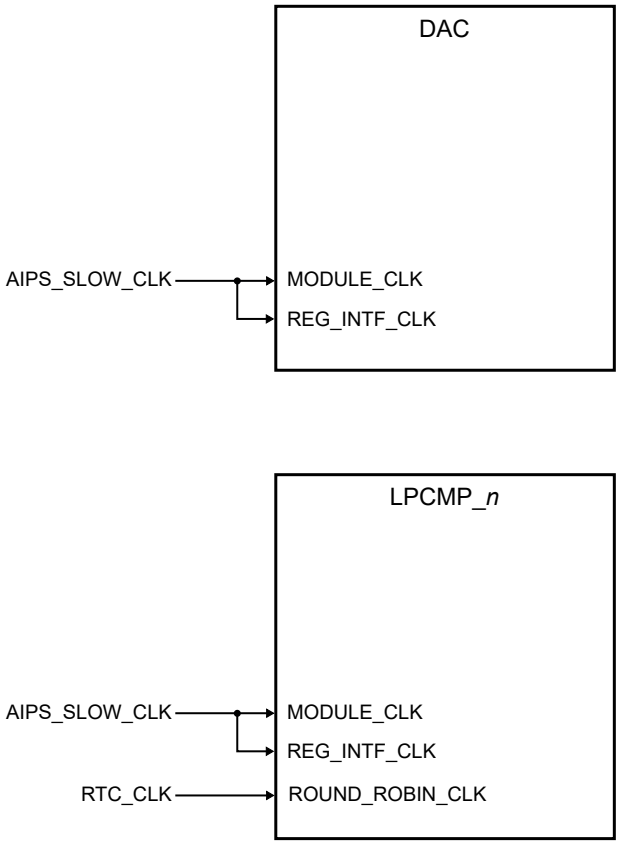



**Figure 97. LPCMP_*n* clocking**

**NOTE**

See Feature comparison for details on this module's availability on your chip variant.

### 23.6.1.12 Memory modules

Figure 98 shows the REG_INTF_CLK and MODULE_CLK connections, and Table 129 shows the REG_INTF_CLK and MODULE_CLK signals used by these modules. Any module diagram that does not explicitly show a REG_INTF_CLK uses the same source for REG_INTF_CLK as used by MODULE_CLK.

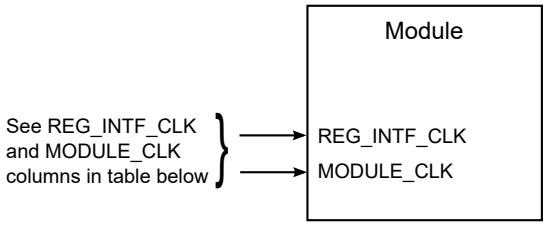



**Figure 98. Memory module clocks**

Table 129. Memory module clocking

| Module | MODULE_CLK | REG_INTF_CLK |
|---|---|---|
| PFLASH/ FLASH | CORE_CLK | AIPS_SLOW_CLK |
| PRAM/ SRAM | CORE_CLK | AIPS_SLOW_CLK |

### 23.6.2 Peripheral data rates

Table 130. Peripheral data rates

| Peripheral | Maximum data rate | |
|---|---|---|
| | MCXE31B | MCXE315/MCXE316 and MCXE317 |
| ADC | See the *MCXE31 Data Sheet* for details. | See the *MCXE31 Data Sheet* for details. |
| eMIOS[1] | Able to shift PWM edge by 1 ÷ (160 MHz) = 6.25 ns. | Able to shift PWM edge by 1 ÷ (120 MHz) = 8.33 ns. |
| BCTU[2] | BCTU to generate triggers at 160 MHz. | BCTU to generate triggers at 120 MHz. |
| LCU | Same domain as EMIOS and BCTU. | Same domain as EMIOS and BCTU. |
| QuadSPI | • Flash memory interface: SDR 120 MHz<br>• DDR and hyperflash not supported | • Flash memory interface: SDR 120 MHz<br>• DDR and hyperflash not supported |
| EMAC | See the section EMAC instance and configuration in the "Ethernet Media Access Controller (EMAC)" chapter for details. | See the section EMAC instance and configuration in the "Ethernet Media Access Controller (EMAC)" chapter for details. |
| FlexCAN[3] | 8 Mbps | 8 Mbps |
| LPI2C[4] | 400 Kbps in fast mode. | 400 Kbps in fast mode. |
| LPSPI[5] | • LPSPI0 is to have a high clock rate of 20 Mbps<br>• LPSPI1–LPSPI5 can be 10 Mbps | • LPSPI0 is to have a high clock rate of 15 Mbps<br>• LPSPI1–LPSPI3 can be 7.5 Mbps |
| SAI0/SAI1 (I2S)[6] | • Bit rate = 12.288 MHz (12.288 Mbps—bit clock frequency governs the bit rate)<br>• Master clock = 24.576 MHz<br>• SAI0 and SAI1 operate asynchronously to each other | • Bit rate = 12.288 MHz (12.288 Mbps—bit clock frequency governs the bit rate)<br>• Master clock = 24.576 MHz<br>• SAI0 and SAI1 operate asynchronously to each other |
| LPUART | See the section Baud rate generation in the "Low Power Universal Asynchronous Receiver/ Transmitter (LPUART)" chapter and Table 112 for details | See the section Baud rate generation in the "Low Power Universal Asynchronous Receiver/ Transmitter (LPUART)" chapter and Table 112 for details |
| FlexIO[7] | The different protocol data rates supported by FlexIO are listed below. For master mode, max baud rate is FLEXIO_CLK ÷ 4. For slave mode, max baud rate is FLEXIO_CLK ÷ 6. The baud rate is controlled by TIMCMP (lower 8 bits in 8-bit mode and 16 bits in 16-bit mode).<br>• UART: 19200 bps | The different protocol data rates supported by FlexIO are listed below. For master mode, max baud rate is FLEXIO_CLK ÷ 4. For slave mode, max baud rate is FLEXIO_CLK ÷ 6. The baud rate is controlled by TIMCMP (lower 8 bits in 8-bit mode and 16 bits in 16-bit mode).<br>• UART: 19200 bps |

*Table continues on the next page...*

Table 130. Peripheral data rates (continued)

| Peripheral | Maximum data rate | |
|---|---|---|
| | MCXE31B | MCXE315/MCXE316 and MCXE317 |
| | • I2C: 400 Kbps<br>• SPI: 10 Mbps<br>• I2S: 12.288 Mbps | • I2C: 400 Kbps<br>• SPI: 7.5 Mbps<br>• I2S: 12.288 Mbps |
| Trace | • Fast-speed pins: 120 MHz<br>• Medium-speed pins: 48 MHz | • SWO trace |

1. See section Global clock prescaler (GCP) in the "Enhanced Modular IO Subsystem (eMIOS)" chapter for details.
2. See section Triggers in the "Enhanced Modular IO Subsystem (eMIOS)" chapter.
3. See the section Protocol timing in the "CAN (FlexCAN)" chapter for data rate calculation details.
4. See the section Clocking in the "Low Power Inter-Integrated Circuit (LPI2C)" chapter for LPI2C_CLK frequency details.
5. See the section Clocking in the "Low Power Serial Peripheral Interface (LPSPI)" chapter.
6. See the section Clocking in the "Synchronous Audio Interface (SAI)" chapter and SAI_*n* clocking for details.
7. See the section Application information and FlexIO instances and configuration in the "Flexible I/O (FlexIO)" chapter and FlexIO clocking for baud configuration details.

## 23.6.3 Core and peripheral clock control

The chip provides provisions for core and peripheral clock gating. The next sections describe the details on clock gating possibilities and controls (see Power Management Controller (PMC) and Mode Entry Module (MC_ME) for details).

### 23.6.3.1 Clock gating

Application core clocks are gated by individual MC_ME core clock enable bits. Additionally, application cores can be clock gated by executing WFI (see the "Mode Entry Module (MC_ME)" chapter for details).

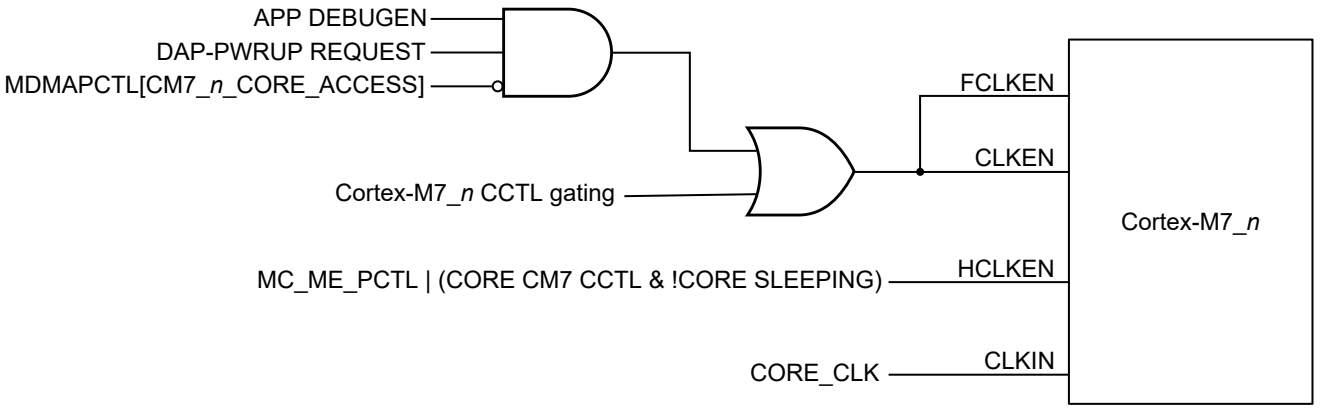



Figure 99. Cortex-M7 core clock gating

To support core debug across functional reset, challenge response done (application debug enable) gating is done so that the debugger can access core debug logic. When the debugger completes its programming with core debug logic, it must program the MDM_AP DAP control bit to shift the control of CLKIN and FCLK to CCTL.

There are two cases in which other masters can access the TCM of each core:

- When the TCM is used as system memory—HCLK will always remain on if TCM PCTL is 1.

- In applications where TCM is not used as system memory, TCM PCTL is written to 0. TCM will then function as the core's memory and HCLK will be gated on WFI.

See the section "MDM_AP register descriptions" in the Debug Subsystem chapter and the Memory Map chapter for details.

#### 23.6.3.2 Peripheral clock gating

See the tables in section "Peripheral clock gating" for the chip partitions, plus peripheral initialization and shutdown details.

## 23.7 Clocking details

### 23.7.1 System clock frequency limitations

**Table 131. System clock frequency limitations**

| System clock node | System clock divider | Maximum frequency allowed | Remarks |
|---|---|---|---|
| CORE_CLK | MC_CGM.MUX_0_DC_0[DIV] | • 160 MHz: MCXE31B<br>• 120 MHz: MCXE315/ MCXE316 and MCXE317 | CORE_CLK is always greater than or equal to AIPS_PLAT_CLK. |
| AIPS_PLAT_CLK | MC_CGM.MUX_0_DC_1[DIV] | 80 MHz | AIPS_PLAT_CLK is always less than or equal to CORE_CLK. |
| AIPS_SLOW_CLK | MC_CGM.MUX_0_DC_2[DIV] | 40 MHz | AIPS_SLOW_CLK is always less than or equal to AIPS_PLAT_CLK. |
| HSE_CLK | MC_CGM.MUX_0_DC_3[DIV] | 120 MHz | When CORE_CLK is equal to or less than 120 MHz, HSE_CLK can be equal to CORE_CLK. When CORE_CLK is higher than 120 MHz, HSE_CLK must be half of the CORE_CLK. |
| DCM_CLK | MC_CGM.MUX_0_DC_4[DIV] | 48 MHz | DCM_CLK |
| LBIST_CLK | MC_CGM.MUX_0_DC_5[DIV] | 48 MHz | LBIST clock |
| QSPI_MEM_CLK | MC_CGM.MUX_0_DC_6[DIV] | 160 MHz | QSPI_MEM_CLK is always equal to CORE_CLK except in 1:1 mode (see Option F - Operation in 1:1 mode with CORE_CLK and AXBS_CLK at same speed). |

**NOTE**

The chip supports 1:1 clocking mode, whereby the core(s) are clocked at the same frequency as the slave ports (flash memory, PRAM controller, AIPS controller). Option F - Operation in 1:1 mode with CORE_CLK and AXBS_CLK at same speed supports this requirement.

The frequencies in the table above are maximum frequencies for a specific clock. However, any clock frequency selected must adhere to the same clock divider ratios shown in Clocking use case examples.

### 23.7.2 Clocking use case examples

The chip supports the clocking modes shown in the subsequent sections, as follows:

- Option A - High Performance mode (CORE_CLK @ 160 MHz) (only available in Run mode)
- Option B - Reduced Speed mode (CORE_CLK @ 120 MHz) (only available in Run mode)

- Option C - Boot Standby mode (CORE_CLK @ 24 MHz)
- Option D - Low-Speed RUN mode (CORE_CLK @ 48 MHz)
- Option E - Low-Speed Run mode (CORE_CLK @ 3 MHz)
- Option E2 - Very-Low-Speed Run mode (CORE_CLK @ 750 kHz)
- Option F - Operation in 1:1 mode with CORE_CLK and AXBS_CLK at same speed (only available in Run mode)

The following list are requirements when configuring the clocking system.

- Configure PRAM/SRAM wait states and flash memory read/write wait cycles in clock configurations where CORE_CLK equals AIPS_PLAT_CLK. See "Option B" in Gasket configurations in various clocking modes for PRAM/SRAM wait states and gasket configurations for clocking options.
- Enable PLL_PHI*n*_CLK only if using it as the system clock source.
- Disable PLL_PHI*n*_CLK when using FIRC_CLK as the system clock, you must.
- Configure the PMC last mile regulator before enabling the PLL. Configure PMC.CONFIG[LMEN] and PMC.CONFIG[LMBCTLEN] (if using an external BJT) to enable the last mile regulator.
- Disable the PLL before disabling the last mile regulator (see PMC last-mile regulator auto-enable feature (for MCXE31B) for last mile regulator details).

### 23.7.2.1 Option A - High Performance mode (CORE_CLK @ 160 MHz)

**Table 132. Option A - High Performance mode (CORE_CLK @ 160 MHz)**

| Clocking options | Clock frequencies | |
|---|---|---|
| | MCXE31B[1] | |
| PLL VCO frequency | 960 MHz | |
| PLLODIV2_CLK<br>(PLLDIG.PLLDV[ODIV2]) | 480 MHz<br>(02h) | |
| FIRC_CLK<br>(ELE_HSEB.CONFIG_REG_GPR[FIRC_DIV_SEL]) | — | |
| **PLL_PHI1_CLK-related clocks** [2] | | |
| PLL_PHI1_CLK<br>(PLLDIG.PLLODIV_1[DIV]) | 240 MHz<br>(0001b) | 160 MHz<br>(0010b) |
| QSPI_SFCK<br>(MC_CGM.MUX_10_DC_0[DIV]) | 120 MHz<br>(0001b) | 80 MHz<br>(0001b) |
| TRACE_CLK<br>(MC_CGM.MUX_11_DC_0[DIV]) | **For fast pads** | |
| | 120 MHz<br>(0001b) | 80 MHz<br>(0001b) |
| | **For medium pads** | |
| | 48 MHz<br>(0100b) | 40 MHz<br>(011b) |
| **PLL_PHI0_CLK-related clocks** [3] | | |

*Table continues on the next page...*

Table 132. Option A - High Performance mode (CORE_CLK @ 160 MHz) (continued)

| Clocking options | Clock frequencies |
|---|---|
| | MCXE31B[1] |
| PLL_PHI0_CLK<br>(PLLDIG.PLLODIV_0[DIV]) | 160 MHz<br>(0010b) |
| CORE_CLK<br>• Application cores<br>• AXBS<br>• SRAM<br>• Flash memory controller port clock<br>• AIPS0 (high-speed peripheral clock)<br>(MC_CGM.MUX_0_DC_0[DIV]) | 160 MHz<br>(0000b) |
| QSPI_MEM_CLK<br>(MC_CGM.MUX_0_DC_6[DIV]) | 160 MHz<br>(0000b) |
| AIPS_PLAT_CLK (medium-speed peripheral clock)<br>(MC_CGM.MUX_0_DC_1[DIV]) | 80 MHz<br>(0001b) |
| AIPS_SLOW_CLK (slow-speed peripheral clock)<br>(MC_CGM.MUX_0_DC_2[DIV]) | 40 MHz<br>(0011b) |
| DCM_CLK<br>(MC_CGM.MUX_0_DC_4[DIV]) | 40 MHz<br>(0011b) |
| HSE_CLK<br>(MC_CGM.MUX_0_DC_3[DIV]) | 80 MHz<br>(0001b) |
| LBIST_CLK<br>(MC_CGM.MUX_0_DC_5[DIV]) | 40 MHz<br>(0011b) |

1. This table does not apply to MCXE315/MCXE316 or MCXE317.
2. MC_CGM.MUX_10_CSC[SELCTL] and MC_CGM.MUX_11_CSC[SELCTL] must equal 1001b.
3. MC_CGM.MUX_0_CSC[SELCTL] must equal 1000b.

#### 23.7.2.2 Option B - Reduced Speed mode (CORE_CLK @ 120 MHz)

Table 133. Option B - Reduced Speed mode (CORE_CLK @ 120 MHz)

| Clocking options | Clock frequencies | |
|---|---|---|
| | MCXE31B | MCXE315/MCXE316 and MCXE317 |
| PLL VCO frequency | 960 MHz | 960 MHz |
| PLLODIV2_CLK<br>(PLLDIG.PLLDV[ODIV2]) | 480 MHz<br>(02h) | 240 MHz<br>(04h) |

*Table continues on the next page...*

Table 133. Option B - Reduced Speed mode (CORE_CLK @ 120 MHz) (continued)

| Clocking options | Clock frequencies | | |
|---|---|---|---|
| | MCXE31B | | MCXE315/MCXE316 and MCXE317 |
| FIRC_CLK (ELE_HSEB.CONFIG_REG_GPR[FIRC_DIV_SEL]) | — | | |
| **PLL_PHI1_CLK-related clocks** [1], [2] | | | |
| PLL_PHI1_CLK (PLLDIG.PLLODIV_1[DIV]) | 240 MHz (0001b) | 160 MHz (0010b) | 48 MHz) (0100b) |
| QSPI_SFCK (MC_CGM.MUX_10_DC_0[DIV]) | 120 MHz (001b) | 80 MHz (001b) | — |
| TRACE_CLK (MC_CGM.MUX_11_DC_0[DIV]) | **For fast pads** | | — |
| | 120 MHz (001b) | 80 MHz (001b) | — |
| | **For medium pads** | | — |
| | 48 MHz (100b) | 40 MHz (011b) | — |
| **PLL_PHI0_CLK-related clocks** [3] | | | |
| PLL_PHI0_CLK (PLLDIG.PLLODIV_0[DIV]) | 120 MHz (011b) | | 120 MHz (001b) |
| CORE_CLK<br>• Application cores<br>• AXBS<br>• SRAM<br>• Flash memory controller port clock<br>• AIPS0 (high-speed peripheral clock)<br>(MC_CGM.MUX_0_DC_0[DIV]) | 120 MHz (000b) | | 120 MHz (000b) |
| QSPI_MEM_CLK (MC_CGM.MUX_0_DC_6[DIV]) | 120 MHz (000b) | | — |
| AIPS_PLAT_CLK (medium-speed peripheral clock) (MC_CGM.MUX_0_DC_1[DIV]) | 60 MHz (001b) | | 60 MHz (001b) |
| AIPS_SLOW_CLK (slow-speed peripheral clock) (MC_CGM.MUX_0_DC_2[DIV]) | 30 MHz (011b) | | 30 MHz (011b) |

*Table continues on the next page...*

Table 133. Option B - Reduced Speed mode (CORE_CLK @ 120 MHz) (continued)

| Clocking options | Clock frequencies | | | |
|---|---|---|---|---|
| | MCXE31B | | MCXE315/MCXE316 and MCXE317 | |
| DCM_CLK<br>(MC_CGM.MUX_0_DC_4[DIV]) | 30 MHz<br>(011b) | | 30 MHz<br>(011b) | |
| HSE_CLK<br>(MC_CGM.MUX_0_DC_3[DIV]) | 120 MHz<br>(000b) | 60 MHz<br>(001b) | 120 MHz<br>(000b) | 60 MHz<br>(001b) |
| LBIST_CLK<br>(MC_CGM.MUX_0_DC_5[DIV]) | 30 MHz<br>(011b) | | 30 MHz<br>(011b) | |

1. MC_CGM.MUX_10_CSC[SELCTL] and MC_CGM.MUX_11_CSC[SELCTL] must equal 1001b.
2. Do not use the combination of different frequencies mentioned below across the 2x2 matrix. The values shown in each cell are valid and must not be clubbed with the values mentioned in any other cells.
3. MC_CGM.MUX_0_CSC[SELCTL] must equal 1000b.

#### 23.7.2.3 Option C - Boot Standby mode (CORE_CLK @ 24 MHz)

Table 134. Option C - Boot Standby mode (CORE_CLK @ 24 MHz)

| Clocking options | Clock frequencies |
|---|---|
| | MCXE31 |
| PLL VCO frequency | — |
| PLLODIV2_CLK<br>(PLLDIG.PLLDV[ODIV2]) | — |
| FIRC_CLK<br>(ELE_HSEB.CONFIG_REG_GPR[FIRC_DIV_SEL]) | 24 MHz[1]<br>(00b) |
| **PLL_PHI1_CLK-related clocks [2]** | |
| PLL_PHI1_CLK<br>(PLLDIG.PLLODIV_1[DIV]) | — |
| QSPI_SFCK<br>(MC_CGM.MUX_10_DC_0[DIV]) | — |
| TRACE_CLK<br>(MC_CGM.MUX_11_DC_0[DIV]) | — |
| **PLL_PHI0_CLK-related clocks [3]** | |
| PLL_PHI0_CLK<br>(PLLDIG.PLLODIV_0[DIV]) | — |
| LBIST_CLK<br>(MC_CGM.MUX_0_DC_5[DIV]) | — |

*Table continues on the next page...*

Table 134. Option C - Boot Standby mode (CORE_CLK @ 24 MHz) (continued)

| Clocking options | Clock frequencies |
|---|---|
| | MCXE31 |
| QSPI_MEM_CLK<br>(MC_CGM.MUX_0_DC_6[DIV]) | — |
| CORE_CLK<br>• Application cores<br>• AXBS<br>• SRAM<br>• Flash memory controller port clock<br>• AIPS0 (high-speed peripheral clock)<br>(MC_CGM.MUX_0_DC_0[DIV]) | 24 MHz<br>(0000b) |
| AIPS_PLAT_CLK (medium-speed peripheral clock)<br>(MC_CGM.MUX_0_DC_1[DIV]) | 24 MHz<br>(0000b) |
| AIPS_SLOW_CLK (slow-speed peripheral clock)<br>(MC_CGM.MUX_0_DC_2[DIV]) | 12 MHz<br>(0001b) |
| DCM_CLK<br>(MC_CGM.MUX_0_DC_4[DIV]) | 24 MHz<br>(0000b) |
| HSE_CLK<br>(MC_CGM.MUX_0_DC_3[DIV]) | 24 MHz<br>(0000b) |

1. The FIRC_DIV_SEL is configured by the sBAF code. It is set to 11b after reset or normal standby exit and FIRC_CLK is 48 MHz. In case of fast standby exit, FIRC_DIV_SEL is 00b and FIRC_CLK is 24 MHz.
2. MC_CGM.MUX_10_CSC[SELCTL] and MC_CGM.MUX_11_CSC[SELCTL] values are don't care since QSPI_SFCK and TRACE_CLK are not used in this use case.
3. MC_CGM.MUX_0_CSC[SELCTL] must equal 0.

#### 23.7.2.4 Option D - Low-Speed RUN mode (CORE_CLK @ 48 MHz)

Table 135. Option D - Low-Speed RUN mode (CORE_CLK @ 48 MHz)

| Clocking options<br>(SYS_CLK = FIRC_CLK) | Clock frequencies |
|---|---|
| | MCXE31 |
| PLL VCO frequency | — |
| PLLODIV2_CLK<br>(PLLDIG.PLLDV[ODIV2]) | — |
| FIRC_CLK<br>(ELE_HSEB.CONFIG_REG_GPR[FIRC_DIV_SEL]) | 48 MHz<br>(11b) |
| **PLL_PHI1_CLK-related clocks** [1] | |

*Table continues on the next page...*

Table 135. Option D - Low-Speed RUN mode (CORE_CLK @ 48 MHz) (continued)

| Clocking options (SYS_CLK = FIRC_CLK) | Clock frequencies |
|---|---|
| | MCXE31 |
| PLL_PHI1_CLK (PLLDIG.PLLODIV_1[DIV]) | — |
| QSPI_SFCK (MC_CGM.MUX_10_DC_0[DIV]) | — |
| TRACE_CLK (MC_CGM.MUX_11_DC_0[DIV]) | — |
| **PLL_PHI0_CLK-related clocks** [2] | |
| PLL_PHI0_CLK (PLLDIG.PLLODIV_0[DIV]) | — |
| LBIST_CLK (MC_CGM.MUX_0_DC_5[DIV]) | — |
| QSPI_MEM_CLK (MC_CGM.MUX_0_DC_6[DIV]) | — |
| CORE_CLK<br>• Application cores<br>• AXBS<br>• SRAM<br>• Flash memory controller port clock<br>• AIPS0 (high-speed peripheral clock)<br>(MC_CGM.MUX_0_DC_0[DIV]) | 48 MHz (0000b) |
| AIPS_PLAT_CLK (medium-speed peripheral clock) (MC_CGM.MUX_0_DC_1[DIV]) | 48 MHz (0000b) |
| AIPS_SLOW_CLK (slow-speed peripheral clock) (MC_CGM.MUX_0_DC_2[DIV]) | 24 MHz (0001b) |
| DCM_CLK (MC_CGM.MUX_0_DC_4[DIV]) | 48 MHz (0000b) |
| HSE_CLK (MC_CGM.MUX_0_DC_3[DIV]) | 48 MHz (0000b) |

1. MC_CGM.MUX_10_CSC[SELCTL] and MC_CGM.MUX_11_CSC[SELCTL] values are don't care since QSPI_SFCK and TRACE_CLK are not used in this use case.
2. MC_CGM.MUX_0_CSC[SELCTL] must equal 0.

### 23.7.2.5 Option E - Low-Speed Run mode (CORE_CLK @ 3 MHz)

Table 136. Option E - Low-Speed Run mode (CORE_CLK @ 3 MHz)

| Clocking options | Clock frequencies |
|---|---|
| | MCXE31 |
| PLL VCO frequency | — |
| PLLODIV2_CLK<br>(PLLDIG.PLLDV[ODIV2]) | — |
| FIRC_CLK<br>(ELE_HSEB.CONFIG_REG_GPR[FIRC_DIV_SEL]) | 3 MHz<br>(10b) |
| **PLL_PHI1_CLK-related clocks** [1] | |
| PLL_PHI1_CLK<br>(PLLDIG.PLLODIV_1[DIV]) | — |
| QSPI_SFCK<br>(MC_CGM.MUX_10_DC_0[DIV]) | — |
| TRACE_CLK<br>(MC_CGM.MUX_11_DC_0[DIV]) | — |
| **PLL_PHI0_CLK-related clocks** [2] | |
| PLL_PHI0_CLK<br>(PLLDIG.PLLODIV_0[DIV]) | — |
| LBIST_CLK<br>(MC_CGM.MUX_0_DC_5[DIV]) | — |
| QSPI_MEM_CLK<br>(MC_CGM.MUX_0_DC_6[DIV]) | — |
| CORE_CLK<br>• Application cores<br>• AXBS<br>• SRAM<br>• Flash memory controller port clock<br>• AIPS0 (high-speed peripheral clock)<br>(MC_CGM.MUX_0_DC_0[DIV]) | 3 MHz<br>(0000b) |
| AIPS_PLAT_CLK (medium-speed peripheral clock)<br>(MC_CGM.MUX_0_DC_1[DIV]) | 3 MHz<br>(0000b) |
| AIPS_SLOW_CLK (slow-speed peripheral clock)<br>(MC_CGM.MUX_0_DC_2[DIV]) | 1.5 MHz<br>(0001b) |

*Table continues on the next page...*

Table 136. Option E - Low-Speed Run mode (CORE_CLK @ 3 MHz) (continued)

| Clocking options | Clock frequencies |
|---|---|
| | MCXE31 |
| DCM_CLK<br>(MC_CGM.MUX_0_DC_4[DIV]) | 3 MHz<br>(0000b) |
| HSE_CLK<br>(MC_CGM.MUX_0_DC_3[DIV]) | 3 MHz<br>(0000b) |

1. MC_CGM.MUX_10_CSC[SELCTL] and MC_CGM.MUX_11_CSC[SELCTL] values are don't care since QSPI_SFCK and TRACE_CLK are not used in this use case.
2. MC_CGM.MUX_0_CSC[SELCTL] must equal 0.

**NOTE**

For FIRC_CLK frequency modes less than 24 MHz, safety modules like the CMU_F*x_n* must be disabled for safety applications, because safety applications are to run on the PLL clocks. The CMU_F*x_n* will cause erroneous FHH events if not disabled.

### 23.7.2.6 Option E2 - Very-Low-Speed Run mode (CORE_CLK @ 750 kHz)

Table 137. Option E2 - Very-Low-Speed Run mode (CORE_CLK @ 750 kHz)

| Clocking options<br>(SYS_CLK = (FIRC_CLK ÷ 8) ÷ 8) | Clock frequencies |
|---|---|
| | MCXE31 |
| PLL VCO frequency | — |
| PLLODIV2_CLK<br>(PLLDIG.PLLDV[ODIV2]) | — |
| FIRC_CLK<br>(ELE_HSEB.CONFIG_REG_GPR[FIRC_DIV_SEL]) | 3 MHz<br>(10b) |
| **PLL_PHI1_CLK-related clocks** [1] | |
| PLL_PHI1_CLK<br>(PLLDIG.PLLODIV_1[DIV]) | — |
| QSPI_SFCK<br>(MC_CGM.MUX_10_DC_0[DIV]) | — |
| TRACE_CLK<br>(MC_CGM.MUX_11_DC_0[DIV]) | — |
| **PLL_PHI0_CLK-related clocks** [2] | |
| PLL_PHI0_CLK<br>(PLLDIG.PLLODIV_0[DIV]) | — |

*Table continues on the next page...*

Table 137. Option E2 - Very-Low-Speed Run mode (CORE_CLK @ 750 kHz) (continued)

| Clocking options (SYS_CLK = (FIRC_CLK ÷ 8) ÷ 8) | Clock frequencies |
|---|---|
| | MCXE31 |
| LBIST_CLK (MC_CGM.MUX_0_DC_5[DIV]) | — |
| QSPI_MEM_CLK (MC_CGM.MUX_0_DC_6[DIV]) | — |
| CORE_CLK<br>• Application cores<br>• AXBS<br>• SRAM<br>• Flash memory controller port clock<br>• AIPS0 (high-speed peripheral clock)<br>(MC_CGM.MUX_0_DC_0[DIV]) | 750 KHz (0011b) |
| AIPS_PLAT_CLK (medium-speed peripheral clock) (MC_CGM.MUX_0_DC_1[DIV]) | 750 KHz (0011b) |
| AIPS_SLOW_CLK (slow-speed peripheral clock) (MC_CGM.MUX_0_DC_2[DIV]) | 375 KHz (0111b) |
| DCM_CLK (MC_CGM.MUX_0_DC_4[DIV]) | 750 KHz (0011b) |
| HSE_CLK (MC_CGM.MUX_0_DC_3[DIV]) | 750 KHz (0011b) |

1. MC_CGM.MUX_10_CSC[SELCTL] and MC_CGM.MUX_11_CSC[SELCTL] values are don't care since QSPI_SFCK and TRACE_CLK are not used in this use case.
2. MC_CGM.MUX_0_CSC[SELCTL] must equal 0.

**NOTE**

For FIRC_CLK frequency modes less than 24 MHz, safety modules like the CMU_F*x_n* must be disabled for safety applications, because safety applications are to run on the PLL clocks. The CMU_F*x_n* will cause erroneous FHH events if not disabled.

### 23.7.2.7 Option F - Operation in 1:1 mode with CORE_CLK and AXBS_CLK at same speed

Table 138. Option F - Operation in 1:1 mode with CORE_CLK and AXBS_CLK at same speed

| Clocking options | Clock frequencies | |
|---|---|---|
| | MCXE31B | MCXE315/MCXE316 and MCXE317 |
| PLL VCO frequency | 960 MHz | 960 MHz |

*Table continues on the next page...*

Table 138. Option F - Operation in 1:1 mode with CORE_CLK and AXBS_CLK at same speed (continued)

| Clocking options | Clock frequencies | | |
|---|---|---|---|
| | MCXE31B | | MCXE315/MCXE316 and MCXE317 |
| PLLODIV2_CLK (PLLDIG.PLLDV[ODIV2]) | 480 MHz (02h) | | 240 MHz (04h) |
| FIRC_CLK (ELE_HSEB.CONFIG_REG_GPR[FIRC_DIV_SEL]) | — | | |
| **PLL_PHI1_CLK-related clocks** [1] | | | |
| PLL_PHI1_CLK (PLLDIG.PLLODIV_1[DIV]) | 240 MHz (0001b) | 160 MHz (0010b) | 48 MHz (0100b) |
| QSPI_SFCK (MC_CGM.MUX_10_DC_0[DIV]) | 120 MHz (001b) | 80 MHz (001b) | — |
| TRACE_CLK (MC_CGM.MUX_11_DC_0[DIV]) | **For fast pads** | | — |
| | 120 MHz (001b) | 80 MHz (001b) | — |
| | **For medium pads** | | — |
| | 48 MHz (100b) | 40 MHz (011b) | — |
| **PLL_PHI0_CLK-related clocks** [2] | | | |
| PLL_PHI0_CLK (PLLDIG.PLLODIV_0[DIV]) | 160 MHz (010b) | | 80 MHz (010b) |
| CORE_CLK<br>• Application cores<br>• AXBS<br>• SRAM<br>• Flash memory controller port clock<br>• AIPS0 (high-speed peripheral clock)<br>(MC_CGM.MUX_0_DC_0[DIV]) | 80 MHz (001b) | | 80 MHz (000b) |
| QSPI_MEM_CLK (MC_CGM.MUX_0_DC_6[DIV]) | 160 MHz (000b) | | — |
| AIPS_PLAT_CLK (medium-speed peripheral clock) (MC_CGM.MUX_0_DC_1[DIV]) | 80 MHz (001b) | | 80 MHz (000b) |

*Table continues on the next page...*

Table 138. Option F - Operation in 1:1 mode with CORE_CLK and AXBS_CLK at same speed (continued)

| Clocking options | Clock frequencies | |
|---|---|---|
| | MCXE31B | MCXE315/MCXE316 and MCXE317 |
| AIPS_SLOW_CLK (slow-speed peripheral clock) (MC_CGM.MUX_0_DC_2[DIV]) | 40 MHz (011b) | 40 MHz (001b) |
| DCM_CLK (MC_CGM.MUX_0_DC_4[DIV]) | 40 MHz (011b) | 40 MHz (001b) |
| HSE_CLK (MC_CGM.MUX_0_DC_3[DIV]) | 80 MHz (001b) | 80 MHz (000b) |
| LBIST_CLK (MC_CGM.MUX_0_DC_5[DIV]) | 40 MHz (011b) | 40 MHz (001b) |

1. MC_CGM.MUX_10_CSC[SELCTL] and MC_CGM.MUX_11_CSC[SELCTL] must equal 1001b.
2. MC_CGM.MUX_0_CSC[SELCTL] must equal 1000b.

### 23.7.3 Gasket configurations in various clocking modes

Table 139. Gasket configurations in various clocking modes (for MCXE31B)

| Gasket configurations | Option A[1] | Option B[2] | Option C[3] | Option D[4] | Option E[5] | Option E2[6] | Option F[7] |
|---|---|---|---|---|---|---|---|
| eDMA (S0) | 1:1 | Bypass | | | | | |
| eDMA (S1) | Bypass | | | | | | |
| ELE_HSEB | 1:2 | 1:1, 1:2 | Bypass | | | | |
| AIPS1/AIPS2 | 2:1 | | Bypass | | | | |
| QuadSPI | 2:1 | | Bypass | | | | |
| PRAM/SRAM | WS enabled | | WS disabled | | | | |
| EMAC 32:64 | 1:1 | | | | | | |
| BDRAM 64:32 | 1:1 | | | | | | |

1. See Option A - High Performance mode (CORE_CLK @ 160 MHz) for details.
2. See Option B - Reduced Speed mode (CORE_CLK @ 120 MHz) for details.
3. See Option C - Boot Standby mode (CORE_CLK @ 24 MHz) for details.
4. See Option D - Low-Speed RUN mode (CORE_CLK @ 48 MHz) for details.
5. See Option E - Low-Speed Run mode (CORE_CLK @ 3 MHz) for details.
6. See Option E2 - Very-Low-Speed Run mode (CORE_CLK @ 750 kHz) for details.
7. See Option F - Operation in 1:1 mode with CORE_CLK and AXBS_CLK at same speed for details.

Table 140. Gasket configurations in various clocking modes (for MCXE315/MCXE316 and MCXE317)

| Gasket configurations | Option B[1] | Option C[2] | Option D[3] | Option E[4] | Option E2[5] | Option F[6] |
|---|---|---|---|---|---|---|
| ELE_HSEB | 1:1, 1:2 | Bypass | | | | |

*Table continues on the next page...*

Table 140. Gasket configurations in various clocking modes (for MCXE315/MCXE316 and MCXE317) (continued)

| Gasket configurations | Option B[1] | Option C[2] | Option D[3] | Option E[4] | Option E2[5] | Option F[6] |
|---|---|---|---|---|---|---|
| AIPS1 | 2:1 | Bypass | | | | |
| PRAM/SRAM | WS disabled | | | | | |
| BDRAM 64:32 (TCM WS) | 1:1 | | | | | |

1. See Option B - Reduced Speed mode (CORE_CLK @ 120 MHz) for details.
2. See Option C - Boot Standby mode (CORE_CLK @ 24 MHz) for details.
3. See Option D - Low-Speed RUN mode (CORE_CLK @ 48 MHz) for details.
4. See Option E - Low-Speed Run mode (CORE_CLK @ 3 MHz) for details.
5. See Option E2 - Very-Low-Speed Run mode (CORE_CLK @ 750 kHz) for details.
6. See Option F - Operation in 1:1 mode with CORE_CLK and AXBS_CLK at same speed for details.

### 23.7.4 Default clock configuration

At reset recovery, the chip runs on the FIRC_CLK as the default configuration as shown in Option C - Boot Standby mode (CORE_CLK @ 24 MHz). Clocking configuration Option_C is the default configuration out of reset with the ELE_HSEB core as the boot core. The Cortex-M7_*n* application core clocks are gated by default. You need to enable the core clocks by the configuring the corresponding core clock enable bits shown in Core Clock Gating.

### 23.7.5 PCFS

The chip supports software-controllable PCFS for MC_CGM MUX_0 (see section "Progressive Clock Frequency Switching (PCFS)" in the "Clock Generation Module (MC_CGM)" chapter for details). PCFS increases and decreases the frequency in steps, avoiding any overshoots or undershoots. When a functional reset event occurs with PCFS enabled, the PCFS process runs and is then followed by the divider configuration updates.

### 23.7.6 Updating dividers: crossbar halt handshake

The clock divider update process consists of the crossbar halt handshake sequence (see section "Clock dividers update" in the "Clock Generation Module (MC_CGM)" chapter for the clock divider update process). A divider update asserts a request to the crossbar switch to halt any transaction that is in process. The dividers are updated when the crossbar switch acknowledges the request for halt. The halt request disables the crossbar switch gaskets in the following order:

1. Core gaskets (ELE_HSEB gaskets)
2. Crossbar switch (AXBS)
3. Flash AXBS bridge
4. PRAM/SRAM gasket

Dividers are updated after the gaskets acknowledge the halt request.

### 23.7.7 Changing system clock configurations

Software sequence for switching clock configurations from FIRC to PLL or PLL to PLL:

1. Before changing clock dividers/clock switching, the communication modules must be disabled by software. This ensures that erroneous communications do not occur during the clock transition.
2. All peripherals must be clock-gated using the MC_ME PCTL configurations.
3. All cores must be clock-gated, except the core being used to control the clock switching.
4. MC_CGM divider configurations can be done to change divider values as described above. The requisite system divider frequency relations must be ensured.

5. Configure MC_CGM_MUX_0_DIV_TRIG_CTRL (enable HHEN and TCTL).
6. Configure MC_CGM_MUX_0_DIV_TRIG.
7. System clock switching can be done by configuration of MC_CGM_MUX_0_CSC[SELCTL]

Software sequence for switching clock configurations from PLL to FIRC clock switching:

1. Before changing clock dividers/clock switching, communication modules must be disabled by software. This ensures that erroneous communications do not occur during the clock transition.
2. All the peripherals must be clock-gated using the MC_ME PCTL configurations.
3. All cores must be clock-gated, except the core being used to control the clock switching.
4. System clock switching can be done by configuration of MC_CGM_MUX_0_CSC[SELCTL].
5. MC_CGM divider configurations can be done to change divider values as described above. The requisite system divider frequency relations must be ensured.
6. Configure MC_CGM_MUX_0_DIV_TRIG_CTRL (enable HHEN and TCTL).
7. Configure MC_CGM_MUX_0_DIV_TRIG.

**NOTE**

When enabling PLL, the PMC last mile regulator should be enabled first by configuring PMC_CONFIG[LMEN] and PMC_CONFIG[LMBCTLEN] (if using an external BJT). The last mile regulator must be disabled after PLL is disabled.

## 23.8 Clock monitoring

The chip contains an independent clock monitoring mechanism which signals malfunctions in the clocking system. This chip consists of six Clock Monitoring Units (CMU_F*x*_[0:5]) for monitoring system clock and clocking module outputs. Figure 100 and Figure 101 show a lower-level view of the clocking monitoring system. Table 141 describes each CMU instance. Each CMU instance provides an independent interrupt or reset indication when the clock signal is out of range or lost. The CMU_FM_*n* provides a timeout indication in case there is a loss of metered clock. Your software must periodically check the CMU_FM_1 and CMU_FM_2 status within the chip FTTI (as specified in the Safety Manual).

**NOTE**

You must disable the CMU corresponding to the system clocks if the application changes the system clock source or changes the system clock divider configuration.

You must disable the CMU monitoring a clock source before disabling the clock source, then enable it after enabling the clock source.

The CMUs should be turned ON only after device has moved to PLL source (wherein LMR is ON).

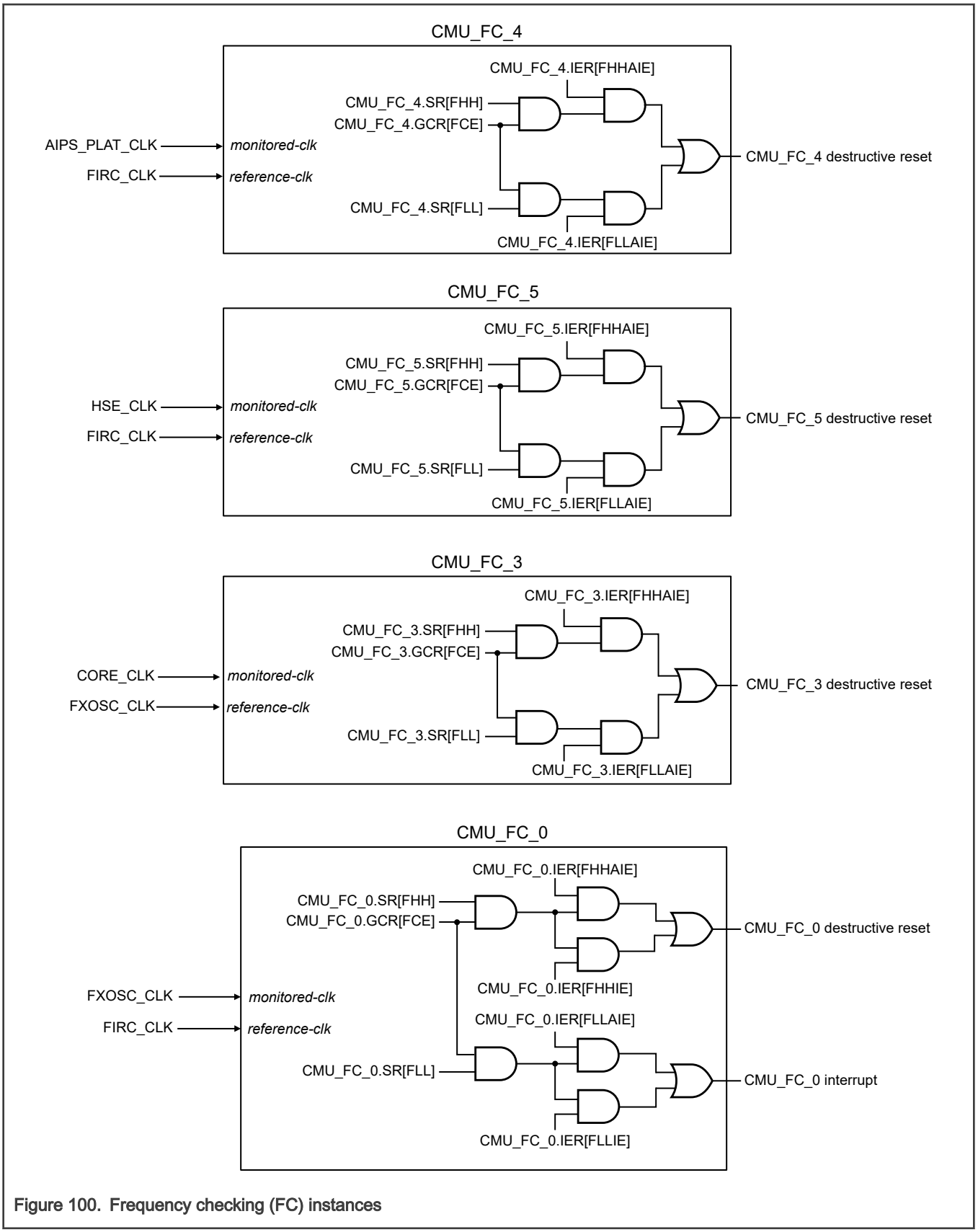



Figure 100. Frequency checking (FC) instances

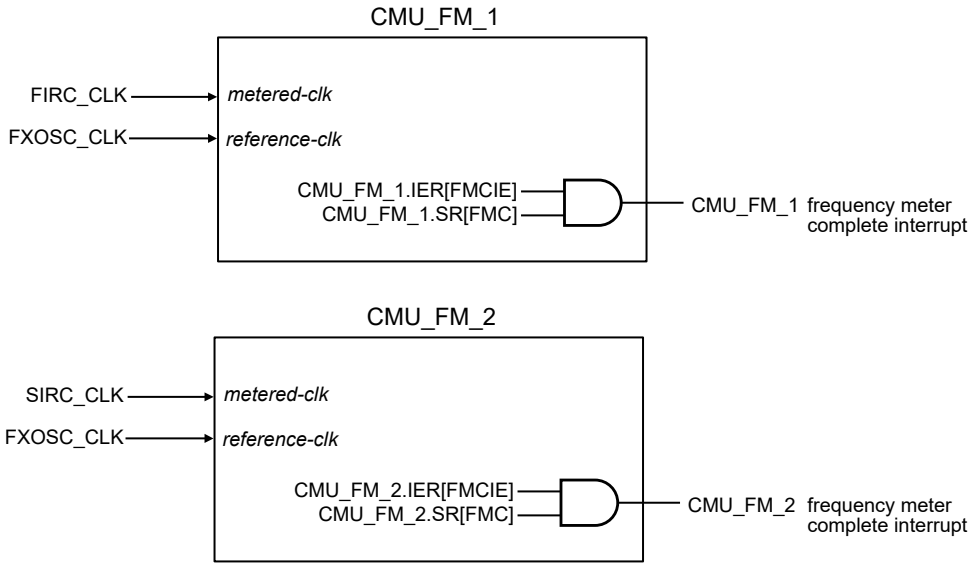



Figure 101. Frequency metering (FM) instances

Table 141. System clock monitors

| CMU | Reference clock | Monitored or metered clock | Failure reaction | Monitoring type description |
|---|---|---|---|---|
| CMU_FC_0 | FIRC_CLK | FXOSC_CLK | Destructive reset or interrupt | Precision over and under frequency |
| CMU_FM_1 | FXOSC_CLK | FIRC_CLK | Interrupt | Current frequency measurement periodically triggered by software |
| CMU_FM_2 | FXOSC_CLK | SIRC_CLK | Interrupt | Current frequency measurement periodically triggered by software |
| CMU_FC_3 | FXOSC_CLK | CORE_CLK | Destructive reset | Precision over and under frequency |
| CMU_FC_4 | FIRC_CLK | AIPS_PLAT_CLK | Destructive reset | Precision over and under frequency |
| CMU_FC_5 | FIRC_CLK | HSE_CLK | Destructive reset | Precision over and under frequency |

## 23.9 Glossary

| | |
|---|---|
| **MODULE_CLK** | Module operating clock |
| **REG_INTF_CLK** | Module register interface clock used for register read and write |
| **PCFS** | Progressive Clock Frequency Switching (see section PCFS for details) |
| **POR** | Power On Reset |
| **SBC** | System Basis Chip (see NXP SBC portfolio) |
| **FLL** | Frequency lower than low frequency reference |
| **FHH** | Frequency higher than high frequency reference |

| | |
|---|---|
| **FTTI** | Fault Tolerance Time Interval |

# Chapter 24
# Clock Generation Module (MC_CGM)

## 24.1 Chip-specific MC_CGM information

### 24.1.1 Associated content references

See the Clocking chapter for details pertaining to these:

- Chip clocking
- MC_CGM clock source mapping (see the "MC_CGM clock sources mapping" section for this)
- MC_CGM clock multiplexers (see the "MC_CGM clock multiplexers" section for this)

**CAUTION**

MC_CGM clock multiplexer configurations to non-supported and reserved clock sources are prohibited and can result in chip malfunctioning.

## 24.2 Introduction

The clock generation module (MC_CGM) is used to set up the configurable clock domains used by various chip blocks as per the application needs. It includes the clock multiplexers that allow software to select the desired clock sources for these domains. This is managed by the MC_CGM to ensure that the changing of the clock selection from one source to another occurs in a glitch-less fashion. In addition, the MC_CGM includes the clock dividers that can be configured by software.

See following figure for the MC_CGM block diagram:

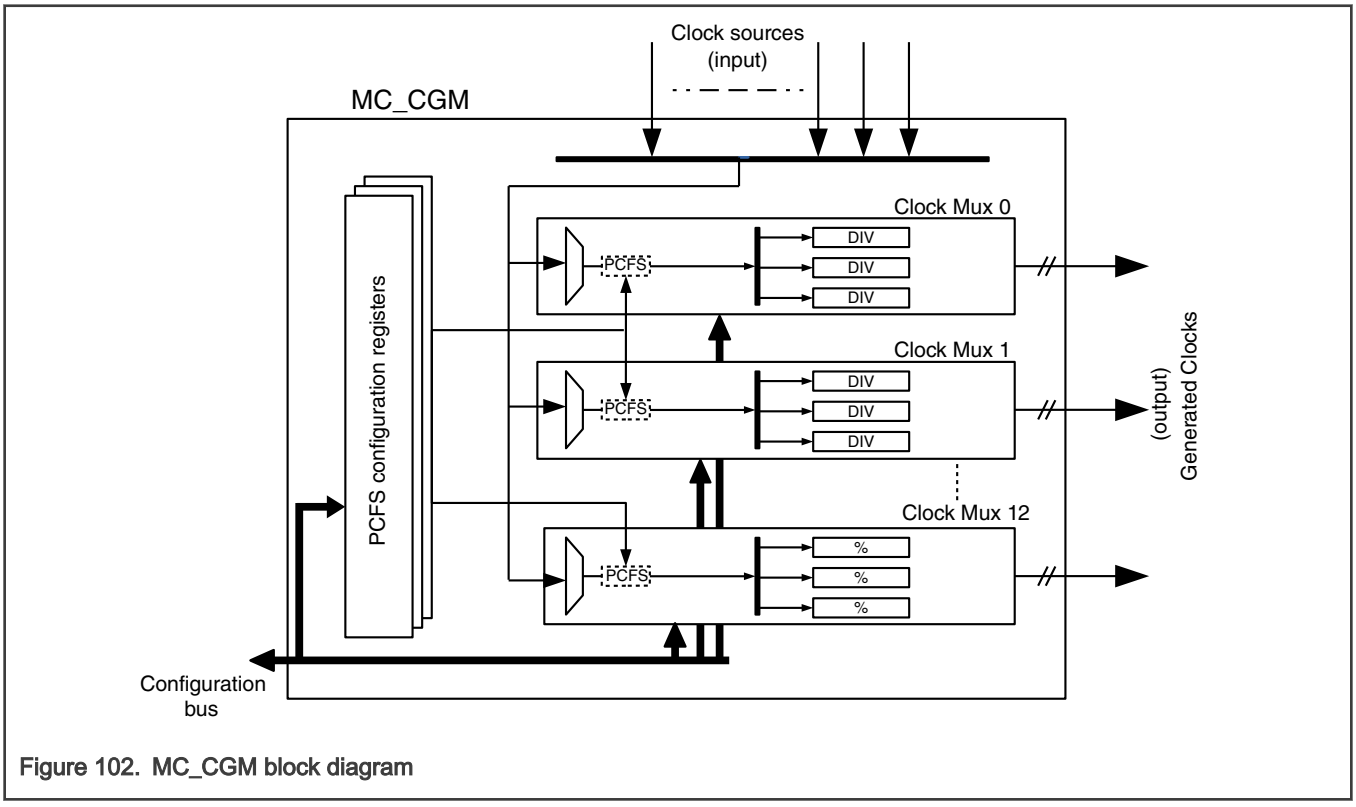



Figure 102. MC_CGM block diagram

**NOTE**

The block diagram is generic and does not necessarily reflect any specific MC_CGM implementation.

## 24.3 Features

MC_CGM includes the following features:

- Implements software configurable clock multiplexers for selecting from various clock sources
- Provides hardware-controlled multiplexers that guarantee glitchless transition, while the software-controlled multiplexers need a software sequence to ensure such a transition
- Provides software configurable automatic PCFS on certain clocks to minimize the impact of a sudden power consumption change through a gentle ramp-down and -up of the clock frequency when switching clock sources
- Implements software configurable clock dividers

## 24.4 Functional description

### 24.4.1 Clock selection multiplexer

Each of the clock multiplexers inside the MC_CGM either implements a fully hardware-controlled clock multiplexer or a software-controlled clock multiplexer. The following sections describe the two variants of the clock multiplexer.

#### 24.4.1.1 Hardware-controlled clock multiplexer

In hardware-based clock multiplexing, the underlying assumption is that under some conditions, error or reset states, software may not be active. Therefore, the clock switching is fully hardware based and is glitchless. To facilitate clock switching requests with software, the MUX_n_CSC and MUX_n_CSS registers implement request and status for the clock multiplexer, the rest is managed in hardware. Using these registers, the software can monitor the state of the hardware-based clock multiplexer and also make clock switching requests. It is recommended that a new clock switch request is given only when there are no pending/ongoing clock switching requests. However, a switch to the safe clock, that is, FIRC, can be performed at any instance of time. A switch to the safe clock is always completed. Software should ensure that while making a software-based switch to safe clock, the register configuration clock should be available, at least for completing the write to the MUX_n_CSC register. This means that the clock should be running for the register write to complete. Hardware clock multiplexer also supports hardware-based switch to safe clock, which is requested externally to MC_CGM (for example, by MC_RGM). For a hardware-based switch to safe clock, it is not required to have a register configuration clock for MC_CGM.

**NOTE**

- "Switch to safe clock" from software is requested by programming the MUX_n_CSC[SAFE_SW] bit field only and not by combining the MUX_n_CSC[CLK_SW] and MUX_n_CSC[SELCTL] bit fields of the MUX_n_CSC register. Writes to other fields are ignored when requesting switch for safe clock.
- After the switch to the safe clock requested by the MC_RGM has completed, the MC_RGM also requests the clock dividers to switch to their default values. This hardware-triggered divider update can be monitored in the same manner as for software-configured updates, and software should use it to ensure that the configuration update has completed before reconfiguring the clocks.
- Write accesses to the MUX_n_CSC register with clock select pointing to the "reserved" input clock source are aborted with bus transfer errors.

See Figure 103 that shows the flowchart representation for the sequence of steps to be followed when making a clock switch request to hardware-controlled clock multiplexer. Switch to safe clock can be requested at any instance of time, and for clarity reasons, it is not shown in Figure 103.

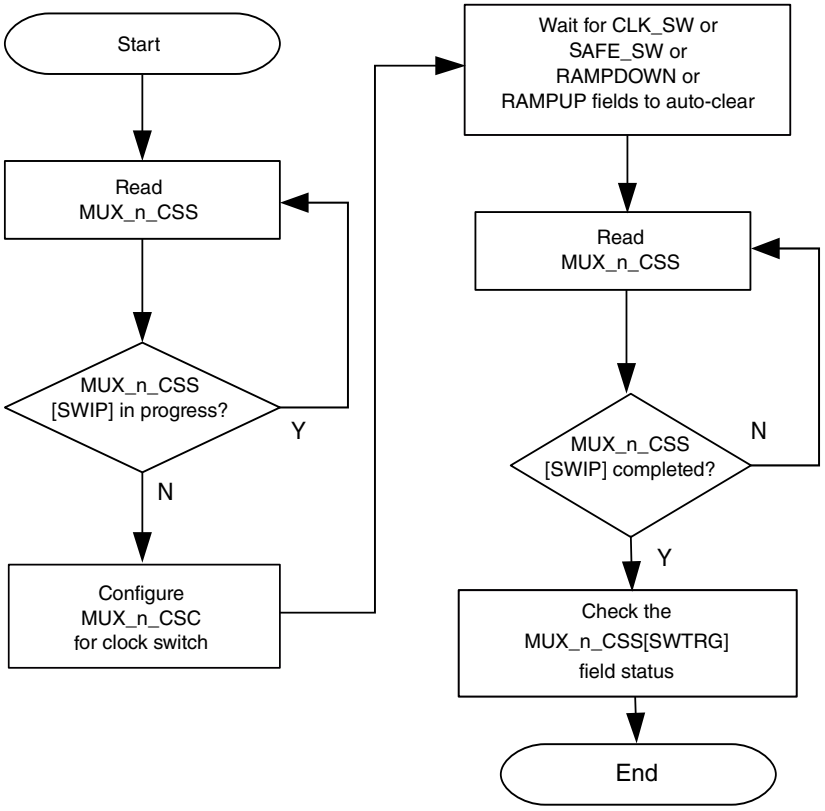



**Figure 103. Flow for clock switch request on hardware-controlled clock multiplexer**

**NOTE**

- A switch to the safe clock command always leads to a ramp-down from the currently selected clock and then a switch to "safe clock", except when there is an ongoing clock switch requested by the software without ramp-up and ramp-down. A safe clock switch request when there is an ongoing clock switch without ramp-up or ramp-down results in a switch to the safe clock without performing a ramp-down (either by MC_RGM or provided using the MUX_n_CSC register) does not perform a ramp-down before switching to "safe clock".
- The above flowchart steps can be preceded by points 1 and 3 below:
  1. Disable the divider.
  2. Switch clocks through hardware multiplexer.
  3. Enable and configure the dividers (atomic write instruction).

### 24.4.1.2 Software-controlled clock multiplexer

In software-based clock multiplexing, the underlying assumption is that the software is always available and there are no error or reset conditions in the chip. This implies that a glitchless switch between input clock sources of MC_CGM MUX can be achieved by following a sequence of steps in software. The software-based clock multiplexer implements a clock gate at the output of the clock mux. The software can gate the selected clock of MC_CGM MUX using a synchronous/graceful clock gate bit (that is, MUX_n_CSC[CG]) or a forced clock gate bit (that is, MUX_n_CSC[FCG]). The hardware does not guarantee that any glitches will escape when using forced clock gating. When a forced clock gate bit is written to 1, the internal clock gate forcefully gates the selected clock to logic-0, therefore, to avoid clock glitches, it should be ensured that the selected clock source is not running. See Figure 104 that shows the flowchart representation for the sequence of steps to be followed when making a clock switch request to software-controlled clock multiplexer. No switch to safe clock is supported in software-controlled clock multiplexer.

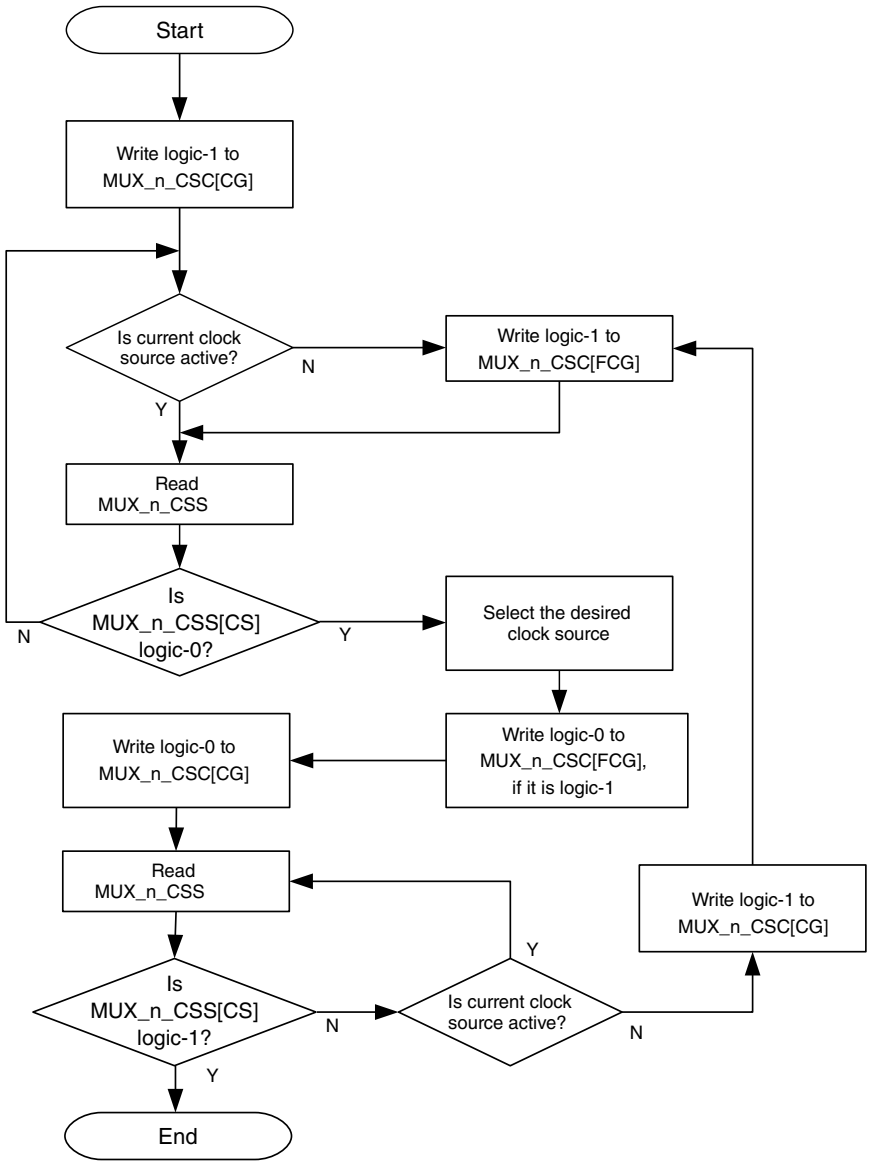



**Figure 104. Flow for clock switch request on software controlled clock multiplexer**

**NOTE**

- Ensure that before using the force gate bit, any IP or other logic using the clock of MC_CGM MUX is in the inactive state or a clock glitch resulting from usage of forced gating does not effect the IP (that is, it is clock gated after the MC_CGM).
- In Figure 104, the clock source to be selected should be active at the time of clock switch, else the MUX_n_CSS[CS] field will remain set to logic-0. In case the clock source to be selected becomes inactive (that is, loss of clock, and so on) during the switching operation, the switch to another clock source can be initiated by writing both the MUX_n_CSC[CG] and MUX_n_CSC[FCG] fields to logic-1.
- Writing a 'reserved' value for the MUX_n_CSC[SELCTL] field may result in an unpredictable clock at the output of the clock multiplexer.

**NOTE**

The above flowchart steps can be preceded by points 1 and 3 below:

1. Disable the divider.
2. Switch clocks through software multiplexer.
3. Enable and configure the dividers (atomic write instruction).

### 24.4.2 PCFS

MC_CGM implements PCFS when changing clock source at an MC_CGM clock mux. The PCFS is only implemented in a hardware-controlled clock multiplexer and not in a software-controlled clock multiplexer. It allows a gradual load change for a power/voltage supply unit by employing a gradual frequency changeover from one to another. The frequency changeover is achieved by clock division of input clock source with a sequence of division values, where frequency ramp down and ramp up is achieved when the sequence of division values are ascending and descending in nature, respectively. As the clock division (fractional) is implemented in digital logic, therefore, it is a coarse-level clock division rather than being an accurate-level division, implying that the duty cycle of the progressively divided clock can vary with time.

The PCFS feature is utilized when a drain current to frequency relationship is known, that is, for a given drain current what is the maximum allowed change in frequency ($f_{chg}$). The $f_{chg}$ is the first input parameter for PCFS and other parameters for PCFS are specified or calculated in relation to FIRC.

The PCFS hardware generates the clock division factors based on certain values that are programmed into the PCFS configuration register. The following pseudo codes represent the generation of clock division value sequence ($d_i$).

```
/* ramp down division values (d_n) with k steps*/
      delta_1 = RATE/1000;
      delta_2 = RATE/1000;
      d_0     = 1.0;
      for i=1 to k-1 do
        d_i = d_i-1 + delta_1;
        delta_1 = delta_1 + delta_2;
      endfor
```

```
/* ramp up division values (d_n) with k steps*/
      delta_1 = RATE*k/1000;
      delta_2 = RATE/1000;
      d_0     = 1.0 + RATE*k*(k+1)/2;
      for i=1 to k-1 do
        d_i = d_i-1 - delta_1;
        delta_1 = delta_1 - delta_2;
    endfor
```

As the generation of clock division values is not a closed-bounded function, calculating RATE for a given $f_{chg}$ is an iterative process. Find a value of RATE that produces a clock division sequence, which when used does not lead to a frequency change greater than $f_{chg}$. See Table 142 that tabulates some of the RATE values against $a_{max}$, where

$$a_{max}=f_{chg}/F_i$$

where, $F_i$ is the frequency of $i^{th}$ input clock source of the clock mux.

**Table 142. PCFS RATE values**

| $a_{max}$ | PCFS rate |
|---|---|
| 0.005 | 12 |
| 0.01 | 48 |

*Table continues on the next page...*

Table 142. PCFS RATE values (continued)

| $a_{max}$ | PCFS rate |
|---|---|
| 0.15 | 112 |
| 0.20 | 184 |

The last clock division factor in case of ramp-down or the first clock in case of ramp-up (following a ramp-down procedure), should be such that clock switch from any input clock source to another should be termed as safe, indicating that load changes have sustainable power effects. This frequency level is referred to as "safe frequency", equivalent to frequency of FIRC referred to as "safe clock" with frequency "safe clock frequency" ($f_{safe}$). The last clock division factor in the sequence of clock division factors happens after "k" steps. The factor k (=steps) is calculated by using the following formula:

```
k = ceil(0.5 + sqrt(0.25 - (2000 * (1 -(Fi/fsafe)) / RATE)))
```

Using the above formula, all the PCFS register configuration values can be calculated for a given frequency of a clock source:

```
PCFS_DIVEi.DIV = (Fi/fsafe)*1000-1;
        PCFS_DIVCi.INIT = RATE * k;
        PCFS_DIVCi.RATE = RATE;
        PCFS_DIVSi.DIV  = 999 + (RATE * k * (k+1)/2);
```

See Figure 105 that shows a graphical representation of the change in frequency, which happens during PCFS ramp-down and ramp-up.

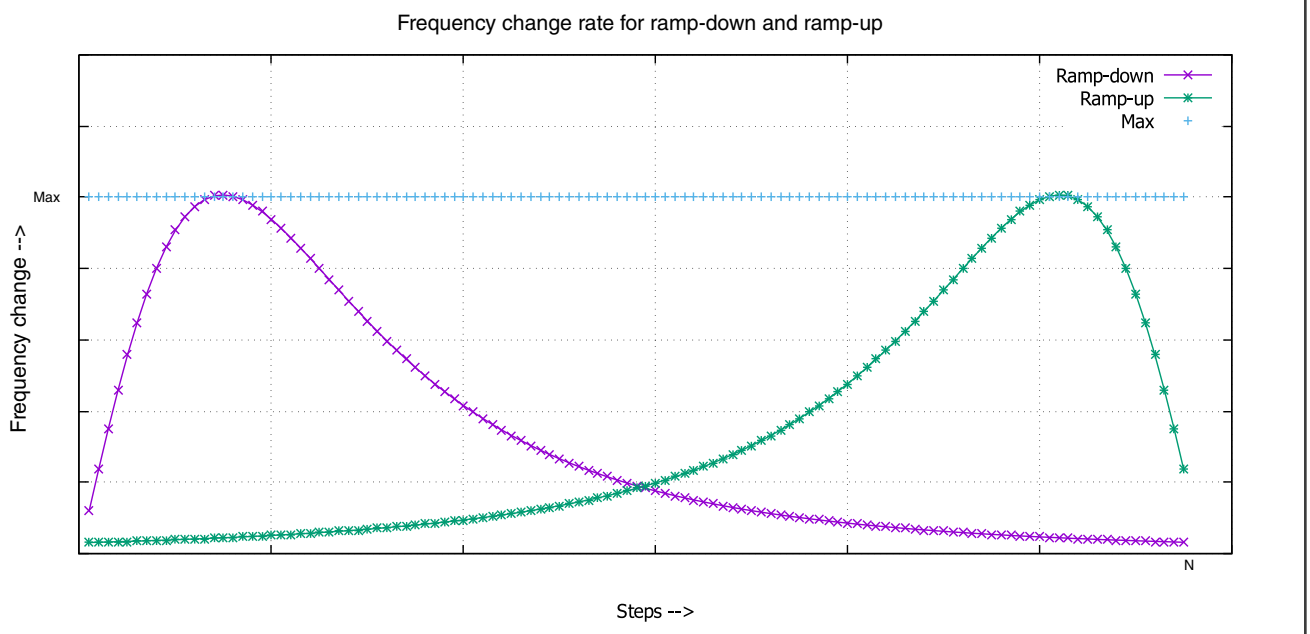



Figure 105. PCFS steps vs frequency changes

For any given clock source, if its frequency is less than that of FIRC, then its corresponding registers should be programmed to default values, where the default values are such that PCFS divider start and PCFS divider end values are same and equal to divide-by-1. The default values ensure that no progressive clock division is done when a clock switch request is given to switch from or to that source.

**NOTE**

Calculate the minimum frequency during the PCFS RAMPDOWN and RAMPUP operations by using the formula:

```
(FIRC/((PCFS_DIVE+1)/1000)) MHz
```

#### 24.4.2.1 PCFS control

The PCFS operation is configured by a set of configuration registers. One set pertains to the calculation of the clock division factors, which are PCFS_DIVC, PCFS_DIVE, PCFS_DIVS, and PCFS_SDUR, while the registers MUX_n_CSC and MUX_n_CSS implement trigger and status of the PCFS operation. The clock division factors are expected to be programmed before any other MC_CGM operation is initiated and remain unchanged. All the registers corresponding to the clock division factors should be programmed with FIRC as the configuration clock and before doing any clock switch or PCFS operation on any of the MC_CGM mux. It should be noted that the default values of the register corresponding to clock division factors are such that only the clock division factor is calculated by hardware that is divide-by-1. The PCFS operation is always triggered when the safe clock request is generated except when there is an on-going clock switch without ramp-up or ramp-down. Therefore, the software needs to ensure that the PCFS configuration is complete and correct.

While configuring MUX_n_CSC, only valid PCFS and clock switch requests should be provided. PCFS or clock switch requests should only be provided if the PCFS operation is in the idle state. If there is an ongoing PCFS operation, it is recommended not to provide any new PCFS triggers (except switch to safe clock) until the ongoing operation is completed. Switch to safe clock via hardware or by register configuration can be provided at any instance of time and is always completed. Valid combinations of PCFS and clock switches triggers are listed in Table 143. All the PCFS commands should be atomic in nature, which means a single register write should provide complete PCFS sequence to be executed that is ramp-down, clock switch, and ramp-up.

**Table 143. Valid PCFS and clock switch requests**

| PCFS operation state | Command |
|---|---|
| Idle | Ramp-down, clock switch, and ramp-up |
| Idle | Clock switch only (without ramp-up or ramp-down) |

When a switch to safe clock is provided by writing to MUX_n_CSC, then writes to other register fields are ignored.

#### 24.4.2.2 Clock source power-up and selection

This section provides guidelines for powering up a given clock source and selecting it at the MC_CGM clock multiplexer.

Following is the power-up procedure for a clock source:

1. Configure the parameter, if any.
2. Configure the power-up (or power-down) control field.
3. Wait for the power-up status indication.

After a power-up indication, the clock source can be selected to provide output clock at an MC_CGM clock multiplexer. A clock-monitoring setup can also be activated on the powered-up clock source.

See Figure 106 that shows a flow chart representation of this sequence.

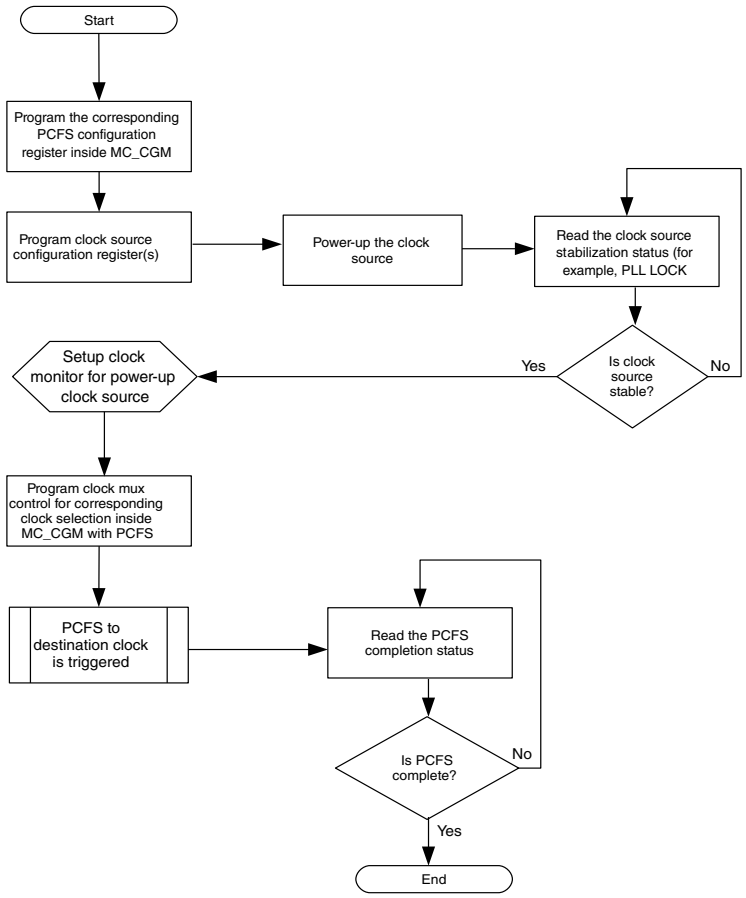



**Figure 106. Clock source power-up and selection procedure**

### 24.4.2.3 Clock source power-down and deselection

This section provides guidelines for powering down a given clock source and deselecting it at MC_CGM clock multiplexer.

Following is the power-down procedure for a clock source:

1. Deselect the clock source at all MC_CGM clock multiplexers.
2. Configure the power-up (or power-down) control field.
3. Wait for the power-down status indication.

See Figure 107 that shows a flow chart representation of this sequence.

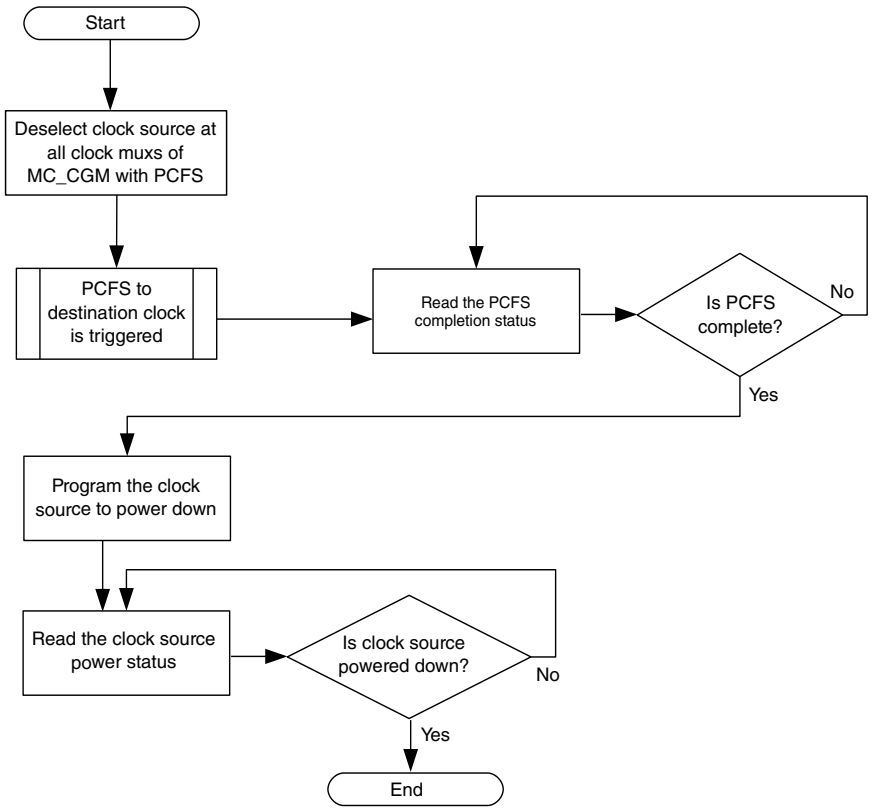



Figure 107. Clock source power-down and clock source deselection procedure

#### 24.4.2.4 Clock switch with load change

This section provides guidelines to switch between two high-frequency clock sources along with load changes in the system. Load change is referred to switching ON or OFF of logic/peripherals in the system, which has an effect of significant capacitance changes on the chip. This triggers the voltage regulation for the chip.

When a large number of peripherals or digital logic is enabled or disabled, it is recommended that this step should be performed at a low frequency. Independent of whether a clock switch is required, this criteria needs to be met. When a clock switch is required at two high frequencies, the recommended sequence is as follows:

1. Intermediately switch to FIRC.
2. Change load (that is, enable/disable peripherals).
3. Switch to the target clock.

See the following figure that shows a flow chart representation of this sequence.

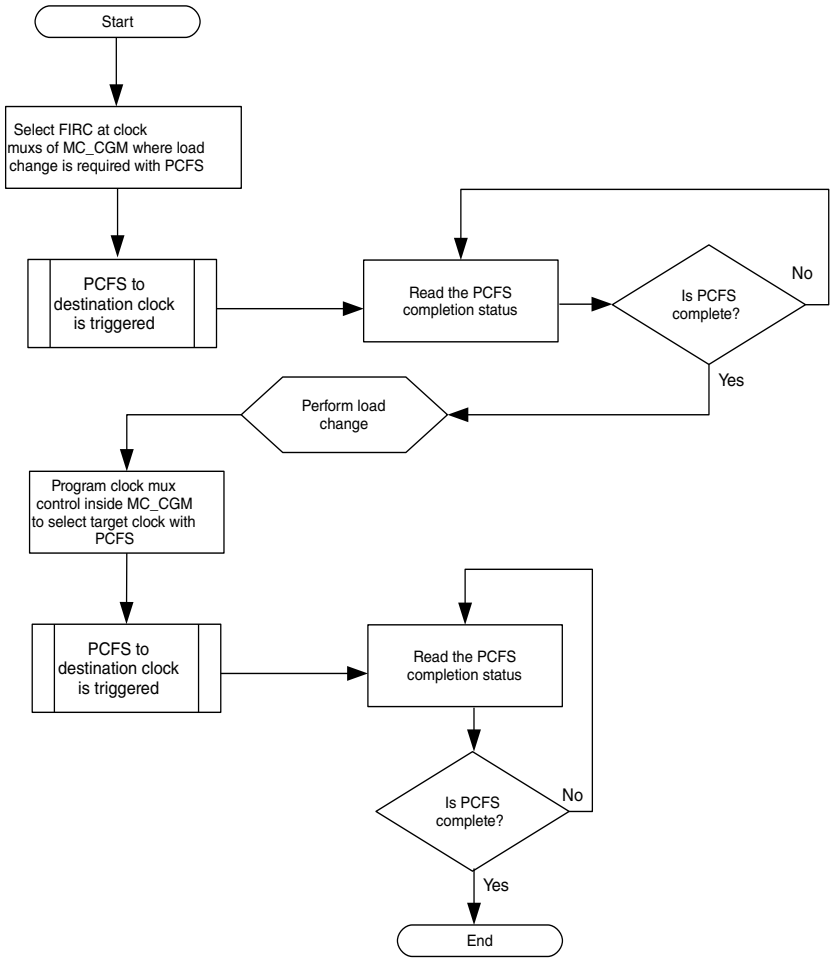



**Figure 108. Clock switch with load change procedure**

## 24.4.3 Clock dividers

Clock dividers are used for the generation of a divided clock that is used for running IPs or peripherals. The MC_CGM provides the following built-in dividers at each clock mux:

- 50% clock dividers

Each divider can be controlled by the Divider Enable (DE) bit and the Division Value (DIV) field. If a divider has its DE bit set to logic-0 in the respective configuration register, then that divider is disabled and the output divided clock is held to logic-0. If the DE bit is logic-1, the divider is enabled and provides a divided clock according to the value set in the DIV field.

### 24.4.3.1 50% clock divider

50% duty cycle dividers generate a real divided clock. The division factor is always an integer but is not restricted to even numbers. The rising edge of the divided clock is always synchronous to the rising edge of the divider source clock, but the falling edge is synchronous to the rising edge or the falling edge depending on whether the division factor is even or odd, respectively. If the input clock has a duty cycle of 50%, the divided output clock maintains the same 50% duty cycle. See Figure 109 that shows the 50% clock divider operation and its associated signals.

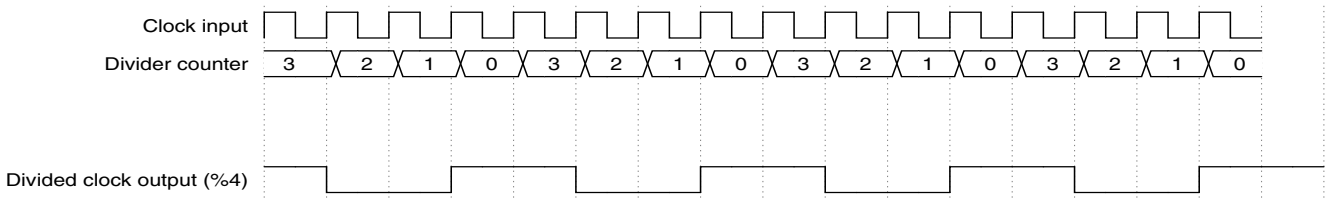



**Figure 109. 50% clock divider programmed for divide by 4 (DIV=3)**

A 50% divided clock can be considered asynchronous (not edge aligned) to other divided clocks from dividers at the same clock mux. 50% clock dividers are implemented without an active closed loop, and are expected not to get stuck if the input clock glitches for a single cycle.

### 24.4.3.2 Clock dividers update

To update the division value or the divider enable, the software should follow the procedure as shown in Figure 110. These updates happen only after the current division cycle has elapsed. However, if the phase of the clock divider is updated, the update happens independently from the state of the division cycle. Any update to the clock divider fields does not result in clock glitch either at the divided clock output or the phase-divided clock output.

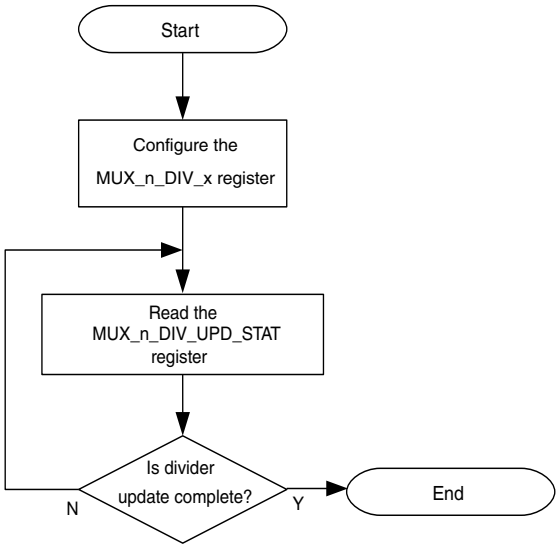



**Figure 110. Flow for clock divider programming update**

Following is the procedure for updating the dividers using the common trigger update:

1. Configure the MUX_n_DIV_TRIG_CTRL register.
2. Wait for the update to finish (until MUX_n_DIV_UPD_STAT is 0).
3. Update the clock dividers (only 50%) per the divider update procedure.
4. After the divider update is finished, perform a write operation on the MUX_n_DIV_TRIG register.
5. Wait for the update to finish, that is, until MUX_n_DIV_UPD_STAT is 0. During this period, the following process takes place:
   - Halt handshake is initiated if configured in step 1.
   - Clock dividers is updated only when AXBS is halted (that is, halt acknowledgment is received by MC_CGM). It is initiated, else the dividers are updated at alignment.
   - After the clock dividers are updated, MUX_n_DIV_UPD_STAT is asserted to 0.

- When the bit fields MUX_x_DIV_TRIG_CTRL[TCTL] and MUX_x_DIV_TRIG_CTRL[HHEN] are set to 1, then any write operation on trigger register will assert (MUX_n_DIV_UPD_STAT). Once the dividers are updated and aligned (MUX_n_DIV_UPD_STAT) will be deasserted.

**NOTE**

- The MUX_x_DIV_TRIG_CTRL[HHEN] bit should only be set when MUX_x_DIV_TRIG_CTRL[TCTL] is set, otherwise it may lead to misalignment of the dividers.
- In case of divider initialization by MC_RGM, a halt handshake protocol is initiated if the corresponding register bit is set and the clock dividers are initialized after the halt handshake protocol completion.

6. This completes the divider update.
7. When multiple writes to the dividers of same clock MUX is made without waiting for the previous update status signal to finish may lead to misalignment of the dividers.

For aligned dividers, the LCM of the division values programmed in the dividers of respective clock mux should be less than 100.

**NOTE**

Performing multiple writes to the divider without waiting for the earlier update to complete can lead to misalignment of the dividers.

Recommended software sequence for ensuring no undivided output at MC_CGM:

1. Reset is de-asserted
2. MC_RGM goes to IDLE
3. Enable the clock dividers of MC_CGM to provide FIRC clock so that reset of fixed dividers can be lifted.
4. Program the MC_CGM as per use case division values.
5. Switch the clock of MC_CGM Mux to desired one, and run the system.

## 24.5 MC_CGM register descriptions

MC_CGM implements a set of clock multiplexers that share PCFS configuration registers. MC_CGM registers have the following properties:

- All registers are 32-bit wide.
- Only 32-bit read and write accesses are supported.
- Read/write accesses of less than 32 bits terminate with an error.
- Writes to read-only register fields in writable registers are ignored and do not provide an error response.
- Writes to read-only registers are aborted with an error response.

### 24.5.1 MC_CGM memory map

MC_CGM base address: 402D_8000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | PCFS Step Duration (PCFS_SDUR) | 32 | RW | 0000_0000h |
| 58h | PCFS Divider Change 8 Register (PCFS_DIVC8) | 32 | RW | 0000_0000h |
| 5Ch | PCFS Divider End 8 Register (PCFS_DIVE8) | 32 | RW | 0000_03E7h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 60h | PCFS Divider Start 8 Register (PCFS_DIVS8) | 32 | RW | 0000_03E7h |
| 300h | Clock Mux 0 Select Control Register (MUX_0_CSC) | 32 | RW | 0000_0000h |
| 304h | Clock Mux 0 Select Status Register (MUX_0_CSS) | 32 | R | 0008_0000h |
| 308h | Clock Mux 0 Divider 0 Control Register (MUX_0_DC_0) | 32 | RW | 8000_0000h |
| 30Ch | Clock Mux 0 Divider 1 Control Register (MUX_0_DC_1) | 32 | RW | 8000_0000h |
| 310h | Clock Mux 0 Divider 2 Control Register (MUX_0_DC_2) | 32 | RW | 8001_0000h |
| 314h | Clock Mux 0 Divider 3 Control Register (MUX_0_DC_3) | 32 | RW | 8000_0000h |
| 318h | Clock Mux 0 Divider 4 Control Register (MUX_0_DC_4) | 32 | RW | 8000_0000h |
| 31Ch | Clock Mux 0 Divider 5 Control Register (MUX_0_DC_5) | 32 | RW | 8003_0000h |
| 320h | Clock Mux 0 Divider 6 Control Register (MUX_0_DC_6) | 32 | RW | 8000_0000h |
| 334h | Clock Mux 0 Divider Trigger Control Register (MUX_0_DIV_TRIG_CTRL) | 32 | RW | 0000_0000h |
| 338h | Clock Mux 0 Divider Trigger Register (MUX_0_DIV_TRIG) | 32 | RW | 0000_0000h |
| 33Ch | Clock Mux 0 Divider Update Status Register (MUX_0_DIV_UPD_STAT) | 32 | R | 0000_0000h |
| 340h | Clock Mux 1 Select Control Register (MUX_1_CSC) | 32 | RW | 0000_0000h |
| 344h | Clock Mux 1 Select Status Register (MUX_1_CSS) | 32 | R | 0008_0000h |
| 348h | Clock Mux 1 Divider 0 Control Register (MUX_1_DC_0) | 32 | RW | 0000_0000h |
| 37Ch | Clock Mux 1 Divider Update Status Register (MUX_1_DIV_UPD_STAT) | 32 | R | 0000_0000h |
| 380h | Clock Mux 2 Select Control Register (MUX_2_CSC) | 32 | RW | 0000_0000h |
| 384h | Clock Mux 2 Select Status Register (MUX_2_CSS) | 32 | R | 0008_0000h |
| 388h | Clock Mux 2 Divider 0 Control Register (MUX_2_DC_0) | 32 | RW | 0000_0000h |
| 3BCh | Clock Mux 2 Divider Update Status Register (MUX_2_DIV_UPD_STAT) | 32 | R | 0000_0000h |
| 3C0h | Clock Mux 3 Select Control Register (MUX_3_CSC) | 32 | RW | 0000_0000h |
| 3C4h | Clock Mux 3 Select Status Register (MUX_3_CSS) | 32 | R | 0008_0000h |
| 3C8h | Clock Mux 3 Divider 0 Control Register (MUX_3_DC_0) | 32 | RW | 0000_0000h |
| 3FCh | Clock Mux 3 Divider Update Status Register (MUX_3_DIV_UPD_STAT) | 32 | R | 0000_0000h |
| 400h | Clock Mux 4 Select Control Register (MUX_4_CSC) | 32 | RW | 0000_0000h |
| 404h | Clock Mux 4 Select Status Register (MUX_4_CSS) | 32 | R | 0008_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 408h | Clock Mux 4 Divider 0 Control Register (MUX_4_DC_0) | 32 | RW | 0000_0000h |
| 43Ch | Clock Mux 4 Divider Update Status Register (MUX_4_DIV_UPD_STAT) | 32 | R | 0000_0000h |
| 440h | Clock Mux 5 Select Control Register (MUX_5_CSC) | 32 | RW | 0000_0000h |
| 444h | Clock Mux 5 Select Status Register (MUX_5_CSS) | 32 | R | 0002_0000h |
| 448h | Clock Mux 5 Divider 0 Control Register (MUX_5_DC_0) | 32 | RW | 8001_0000h |
| 47Ch | Clock Mux 5 Divider Update Status Register (MUX_5_DIV_UPD_STAT) | 32 | R | 0000_0000h |
| 480h | Clock Mux 6 Select Control Register (MUX_6_CSC) | 32 | RW | 0000_0000h |
| 484h | Clock Mux 6 Select Status Register (MUX_6_CSS) | 32 | R | 0002_0000h |
| 488h | Clock Mux 6 Divider 0 Control Register (MUX_6_DC_0) | 32 | RW | 8001_0000h |
| 4BCh | Clock Mux 6 Divider Update Status Register (MUX_6_DIV_UPD_STAT) | 32 | R | 0000_0000h |
| 4C0h | Clock Mux 7 Select Control Register (MUX_7_CSC) | 32 | RW | 0000_0000h |
| 4C4h | Clock Mux 7 Select Status Register (MUX_7_CSS) | 32 | R | 0008_0000h |
| 4C8h | Clock Mux 7 Divider 0 Control Register (MUX_7_DC_0) | 32 | RW | 0000_0000h |
| 4FCh | Clock Mux 7 Divider Update Status Register (MUX_7_DIV_UPD_STAT) | 32 | R | 0000_0000h |
| 500h | Clock Mux 8 Select Control Register (MUX_8_CSC) | 32 | RW | 0000_0000h |
| 504h | Clock Mux 8 Select Status Register (MUX_8_CSS) | 32 | R | 0008_0000h |
| 508h | Clock Mux 8 Divider 0 Control Register (MUX_8_DC_0) | 32 | RW | 0000_0000h |
| 53Ch | Clock Mux 8 Divider Update Status Register (MUX_8_DIV_UPD_STAT) | 32 | R | 0000_0000h |
| 540h | Clock Mux 9 Select Control Register (MUX_9_CSC) | 32 | RW | 0000_0000h |
| 544h | Clock Mux 9 Select Status Register (MUX_9_CSS) | 32 | R | 0008_0000h |
| 548h | Clock Mux 9 Divider 0 Control Register (MUX_9_DC_0) | 32 | RW | 0000_0000h |
| 57Ch | Clock Mux 9 Divider Update Status Register (MUX_9_DIV_UPD_STAT) | 32 | R | 0000_0000h |
| 580h | Clock Mux 10 Select Control Register (MUX_10_CSC) | 32 | RW | 0000_0000h |
| 584h | Clock Mux 10 Select Status Register (MUX_10_CSS) | 32 | R | 0008_0000h |
| 588h | Clock Mux 10 Divider 0 Control Register (MUX_10_DC_0) | 32 | RW | 0000_0000h |
| 5BCh | Clock Mux 10 Divider Update Status Register (MUX_10_DIV_UPD_STAT) | 32 | R | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 5C0h | Clock Mux 11 Select Control Register (MUX_11_CSC) | 32 | RW | 0000_0000h |
| 5C4h | Clock Mux 11 Select Status Register (MUX_11_CSS) | 32 | R | 0002_0000h |
| 5C8h | Clock Mux 11 Divider 0 Control Register (MUX_11_DC_0) | 32 | RW | 8000_0000h |
| 5FCh | Clock Mux 11 Divider Update Status Register (MUX_11_DIV_UPD_STAT) | 32 | R | 0000_0000h |

## 24.5.2 PCFS Step Duration (PCFS_SDUR)

### Offset

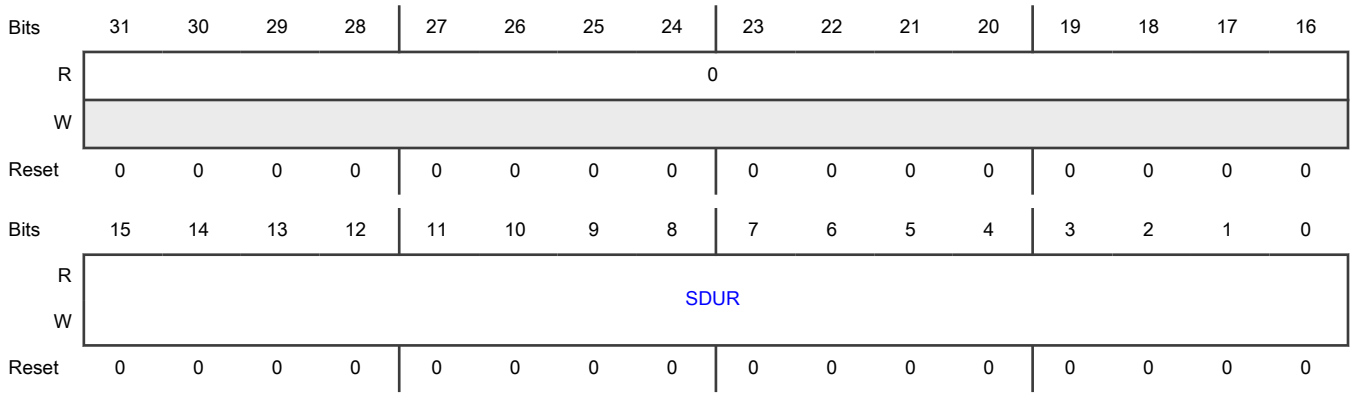
| Register | Offset |
|---|---|
| PCFS_SDUR | 0h |

### Function

This register specifies the step duration of each PCFS step. The value provided in this register specifies the PCFS step duration in terms of the number of cycles of FIRC.

This register is reset only by a destructive reset. For details, see PCFS.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>— | This field is reserved and reads return zeros. |
| 15-0<br>SDUR | Step duration<br>Count value of the step duration |

## 24.5.3 PCFS Divider Change 8 Register (PCFS_DIVC8)

### Offset

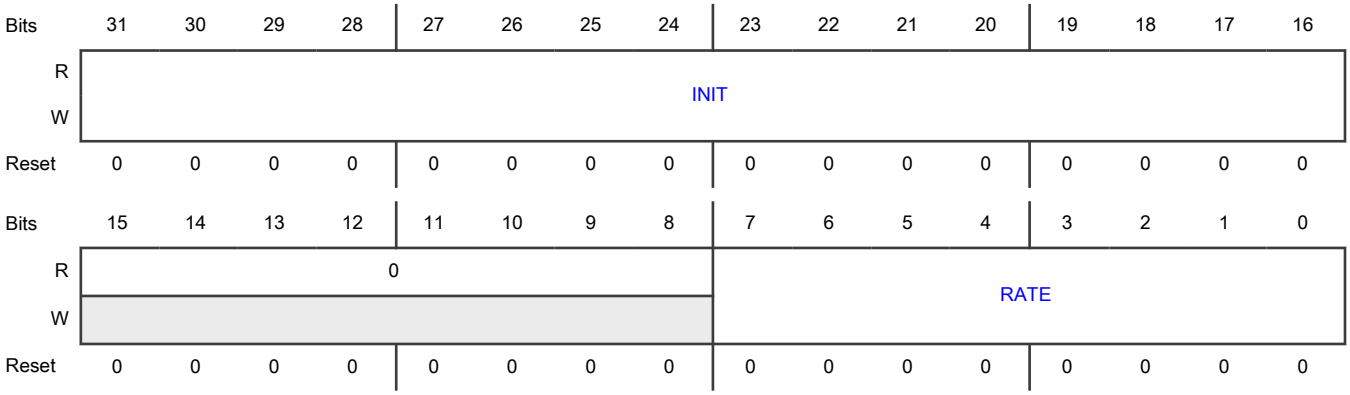
| Register | Offset |
|---|---|
| PCFS_DIVC8 | 58h |

### Function

This register defines the rate of frequency change and initial change value on frequency ramp-up for the Progressive Clock Frequency switching of PLL_PHI0_CLK.

This register is reset only on destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>INIT | Divider change initial value<br>This field provides the initial change value of the clock divider for the clock ramp-up phase of PLL_PHI0_CLK. |
| 15-8<br>— | This field is reserved and reads return zeros. |
| 7-0<br>RATE | Divider change rate<br>This value controls the change value of the clock divider for the clock ramp-up and ramp-down phase of PLL_PHI0_CLK. |

## 24.5.4 PCFS Divider End 8 Register (PCFS_DIVE8)

### Offset

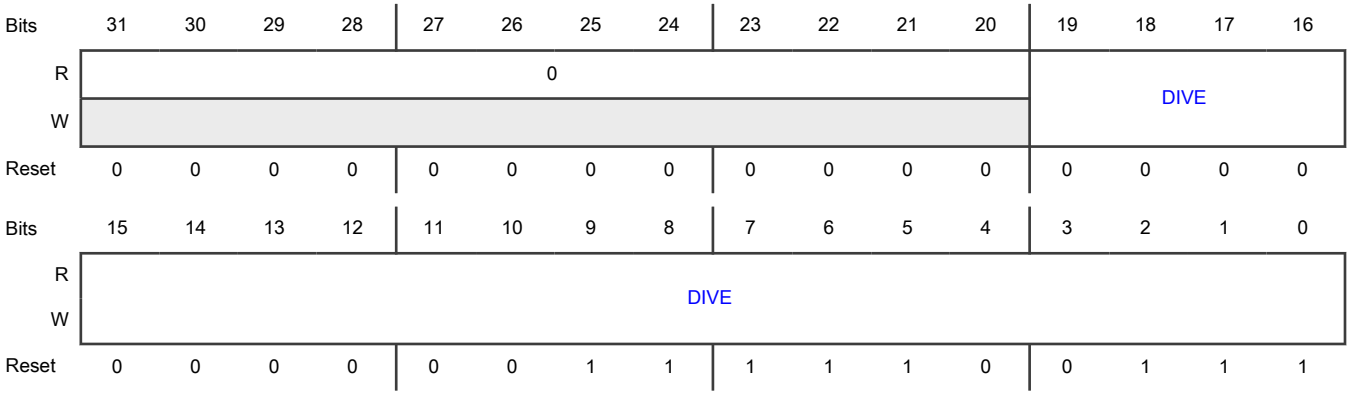
| Register | Offset |
|---|---|
| PCFS_DIVE8 | 5Ch |

### Function

This register defines the final division value on frequency ramp-down for the progressive system clock switching of PLL_PHI0_CLK.

This registers is reset only on destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-20<br>— | This field is reserved and reads return zeros. |
| 19-0<br>DIVE | Divider end value<br>This field provides the end value of the clock divider for the PLL_PHI0_CLK ramp-down phase. |

## 24.5.5 PCFS Divider Start 8 Register (PCFS_DIVS8)

### Offset

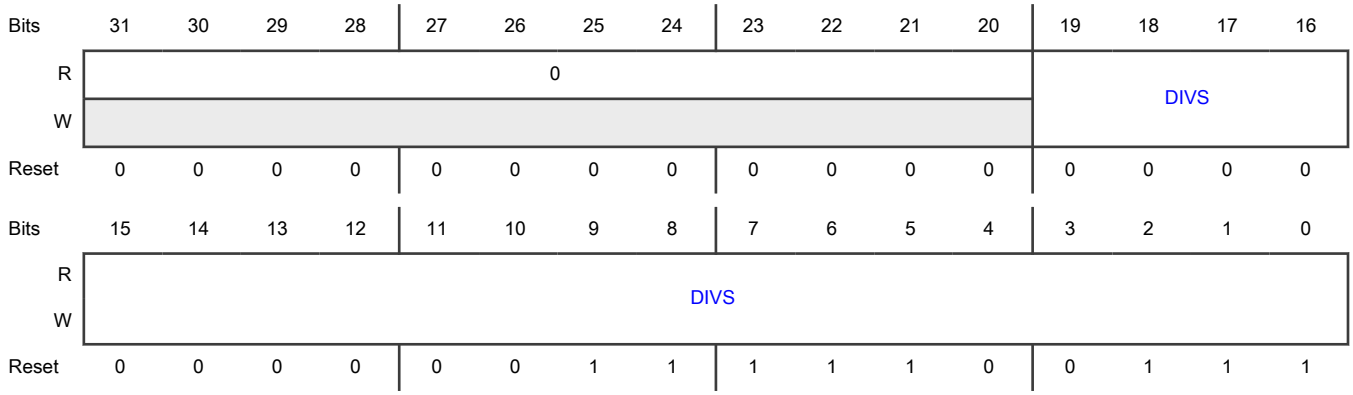
| Register | Offset |
|---|---|
| PCFS_DIVS8 | 60h |

### Function

This register defines the initial division value on frequency ramp-up for the progressive system clock switching of PLL_PHI0_CLK.

This register is reset only on destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-20<br>— | This field is reserved and reads return zeros. |
| 19-0<br>DIVS | Divider start value<br>This field provides the start value of the clock divider for the PLL_PHI0_CLK ramp-up phase |

## 24.5.6 Clock Mux 0 Select Control Register (MUX_0_CSC)

### Offset

| Register | Offset |
|---|---|
| MUX_0_CSC | 300h |

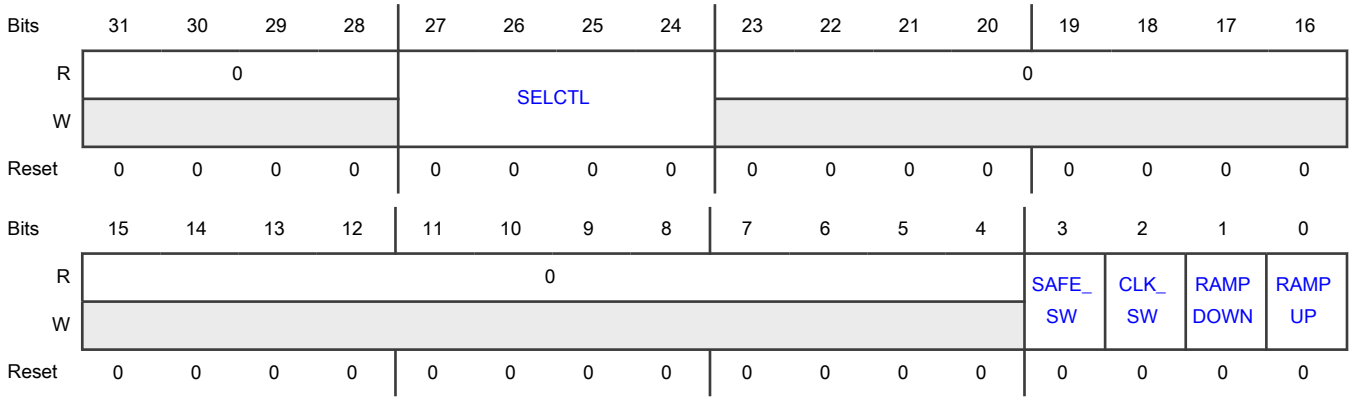
### Function

This register provides the clock source selection control for clock mux 0. Clock mux 0 implements hardware control clock switching ensuring that the clock switch happens in a graceful manner (without glitches). See the "Hardware-controlled clock multiplexer" section for details.

This register is reset on destructive reset only.

An update to all the PCFS-related fields of this register must be an atomic write, which means a single write must update the CLK_SW, RAMPDOWN, and RAMPUP fields. It is necessary to set both RAMPUP and RAMPDOWN bits together even if you want to trigger either RAMPUP or RAMPDOWN process otherwise the desired PCFS sequence will not be executed.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-28<br>— | This field is reserved and reads return zeros. |
| 27-24<br>SELCTL | Clock source selection control<br>Selects the source clock for clock mux 0. The reserved values are not displayed.<br>0000b - FIRC<br>1000b - PLL_PHI0_CLK |
| 23-4<br>— | This field is reserved and reads return zeros. |
| 3<br>SAFE_SW | Safe clock request<br>Writing 1 to this bit makes a safe clock switch request to FIRC. After a safe clock switch operation is requested, this bit is auto cleared and a corresponding bit in the status register is set. |
| 2<br>CLK_SW | Clock switch<br>Writing 1 to this bit makes a clock switch request to clock mux 0. After a clock switch operation is requested, this bit is auto cleared and a corresponding bit in the status register is set. |
| 1<br>RAMPDOWN | PCFS ramp-down<br>Writing 1 to this bit makes a PCFS ramp-down request to clock mux 0. After a PCFS ramp-down operation is requested, this bit is auto cleared and a corresponding bit in the status register is set. |
| 0<br>RAMPUP | PCFS ramp-up<br>Writing 1 to this bit makes a PCFS ramp-up request to clock mux 0. After a PCFS ramp-up operation is requested, this bit is auto cleared and a corresponding bit in the status register is set. |

## 24.5.7 Clock Mux 0 Select Status Register (MUX_0_CSS)

### Offset

| Register | Offset |
|---|---|
| MUX_0_CSS | 304h |

### Function

This register provides the current clock source selection status for clock mux 0.

This register is reset on destructive reset only.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | SELSTAT | | | | 0 | | | | SWTRG | | | SWIP |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | SAFE_SW | CLK_SW | RAMP DOWN | RAMP UP |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-28<br>— | This field is reserved and reads return zeros. |
| 27-24<br>SELSTAT | Clock source selection status<br>This value indicates the current source selected for clock mux 0. The reserved values are not displayed.<br>0000b - FIRC<br>1000b - PLL_PHI0_CLK |
| 23-20<br>— | This field is reserved and reads return zeros. |
| 19-17<br>SWTRG | Switch trigger cause<br>This value indicates the cause for the latest clock source switch. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>If the clock fails, followed by multiple safe clock switch requests for MC_CGM hardware clock mux, the value of the SWTRG field can be either 4 or 5.<br>000b - Reserved<br>001b - Switch after request succeeded.<br>010b - Switch after the request failed because of an inactive target clock and the current clock is FIRC.<br>011b - Switch after the request failed because of an inactive current clock and the current clock is FIRC.<br>100b - Switch to FIRC because of a safe clock request or reset succeeded.<br>101b - Switch to FIRC because of a safe clock request or reset succeeded, but the previous current clock source was inactive.<br>110b - Reserved<br>111b - Reserved |
| 16<br>SWIP | Switch in progress<br>**NOTE**<br>New clock switch request can only be given three clock cycles after the completion of the previous request.<br>0b - Clock source switching is complete.<br>1b - Clock source switching is in progress. |
| 15-4<br>— | This field is reserved and reads return zeros. |
| 3<br>SAFE_SW | Safe clock request<br>This field provides an indication of whether a switch to safe clock operation was requested during the previous/ongoing request on clock mux 0.<br>0b - No safe clock switch operation was requested.<br>1b - Safe clock switch operation was requested. |
| 2<br>CLK_SW | Clock switch<br>This field provides an indication of whether a clock switch operation was requested during the previous/ongoing request on clock mux 0.<br>0b - No clock switch operation was requested.<br>1b - Clock switch operation was requested. |
| 1<br>RAMPDOWN | PCFS ramp-down |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | This field provides an indication of whether a PCFS ramp-down operation was requested during the previous/ongoing request on clock mux 0.<br>**NOTE**<br>In case of safe clock switching, the ramp-down operation runs internally, but the value of the corresponding status field is not set to 1.<br>0b - No ramp-down operation was requested.<br>1b - Ramp-down operation was requested. |
| 0<br>RAMPUP | PCFS ramp-up<br>This field provides an indication of whether a PCFS ramp-up operation was requested during the previous/ ongoing request on clock mux 0.<br>0b - No ramp-up operation was requested.<br>1b - Ramp-up operation was requested. |

## 24.5.8 Clock Mux 0 Divider 0 Control Register (MUX_0_DC_0)

### Offset

| Register | Offset |
|---|---|
| MUX_0_DC_0 | 308h |

### Function

This register controls the clock divider 0 for clock mux 0.

This divider is a 50% duty cycle divider.

**NOTE**

The update to the fields of this register should be an atomic write, that is, one single write should update the complete register.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DE | 0 | | | | | | | | | | | DIV | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31<br>DE | Divider enable<br>0b - Unused<br>1b - Divider is enabled. |
| 30-19<br>— | This field is reserved and reads return zeros. |
| 18-16<br>DIV | Division value<br>This field provides the division value for the clock divider. The clock period of the clock after division is 'DIV+1' times the time period of the current input clock to the divider. |
| 15-0<br>— | This field is reserved and reads return zeros. |

## 24.5.9 Clock Mux 0 Divider 1 Control Register (MUX_0_DC_1)

### Offset

| Register | Offset |
|---|---|
| MUX_0_DC_1 | 30Ch |

### Function

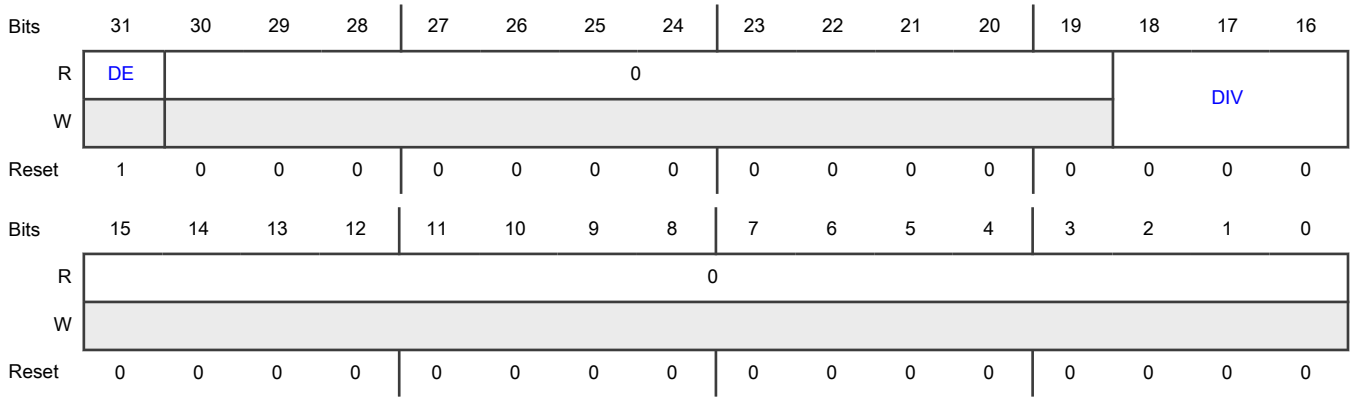
This register controls the clock divider 1 for clock mux 0.

This divider is a 50% duty cycle divider.

**NOTE**

The update to the fields of this register should be an atomic write, that is, one single write should update the complete register.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31<br>DE | Divider enable<br>0b - Unused<br>1b - Divider is enabled. |
| 30-19<br>— | This field is reserved and reads return zeros. |
| 18-16<br>DIV | Division value<br>This field provides the division value for the clock divider. The clock period of the clock after division is 'DIV+1' times the time period of the current input clock to the divider. |
| 15-0<br>— | This field is reserved and reads return zeros. |

## 24.5.10 Clock Mux 0 Divider 2 Control Register (MUX_0_DC_2)

### Offset

| Register | Offset |
|---|---|
| MUX_0_DC_2 | 310h |

### Function

This register controls the clock divider 2 for clock mux 0.

This divider is a 50% duty cycle divider.

**NOTE**

The update to the fields of this register should be an atomic write, that is, one single write should update the complete register.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DE | 0 | | | | | | | | | | | DIV | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

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

### Fields

| Field | Function |
|---|---|
| 31<br>DE | Divider enable<br>0b - Unused<br>1b - Divider is enabled. |
| 30-20<br>— | This field is reserved and reads return zeros. |
| 19-16<br>DIV | Division value<br>This field provides the division value for the clock divider. The clock period of the clock after division is 'DIV+1' times the time period of the current input clock to the divider. |
| 15-0<br>— | This field is reserved and reads return zeros. |

## 24.5.11 Clock Mux 0 Divider 3 Control Register (MUX_0_DC_3)

### Offset

| Register | Offset |
|---|---|
| MUX_0_DC_3 | 314h |

### Function

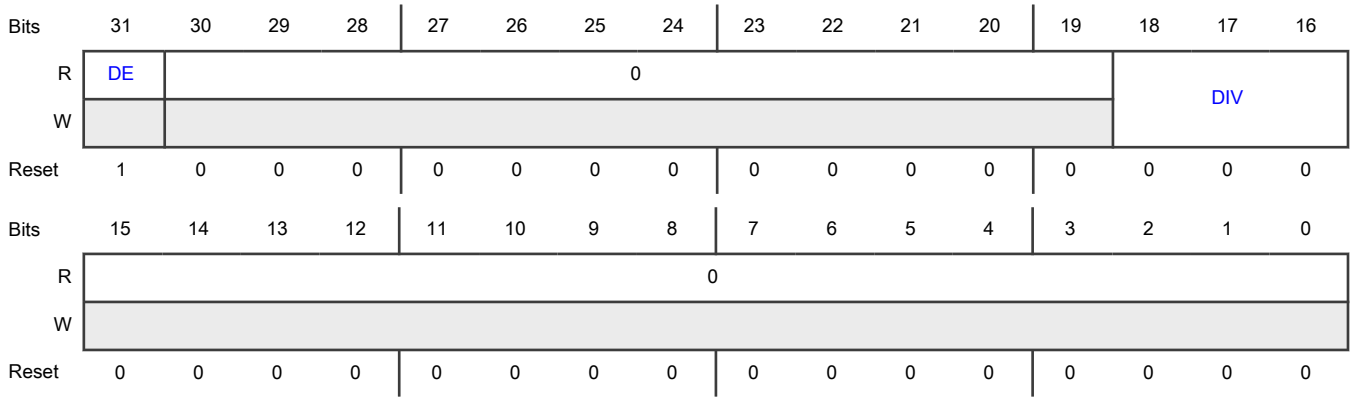
This register controls the clock divider 3 for clock mux 0.

This divider is a 50% duty cycle divider.

**NOTE**

The update to the fields of this register should be an atomic write, that is, one single write should update the complete register.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31<br>DE | Divider enable<br>0b - Unused<br>1b - Divider is enabled. |
| 30-19<br>— | This field is reserved and reads return zeros. |
| 18-16<br>DIV | Division value<br>This field provides the division value for the clock divider. The clock period of the clock after division is 'DIV+1' times the time period of the current input clock to the divider. |
| 15-0<br>— | This field is reserved and reads return zeros. |

## 24.5.12 Clock Mux 0 Divider 4 Control Register (MUX_0_DC_4)

### Offset

| Register | Offset |
|---|---|
| MUX_0_DC_4 | 318h |

### Function

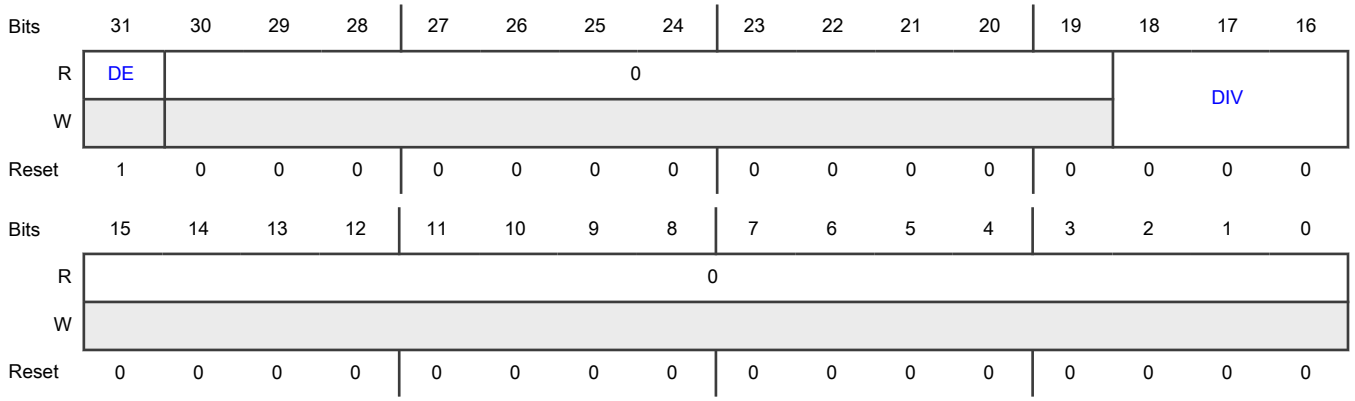
This register controls the clock divider 4 for clock mux 0.

This divider is a 50% duty cycle divider.

**NOTE**

The update to the fields of this register should be an atomic write, that is, one single write should update the complete register.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31<br>DE | Divider enable<br>0b - Unused<br>1b - Divider is enabled. |
| 30-19<br>— | This field is reserved and reads return zeros. |
| 18-16<br>DIV | Division value<br>This field provides the division value for the clock divider. The clock period of the clock after division is 'DIV+1' times the time period of the current input clock to the divider. |
| 15-0<br>— | This field is reserved and reads return zeros. |

## 24.5.13 Clock Mux 0 Divider 5 Control Register (MUX_0_DC_5)

### Offset

| Register | Offset |
|---|---|
| MUX_0_DC_5 | 31Ch |

### Function

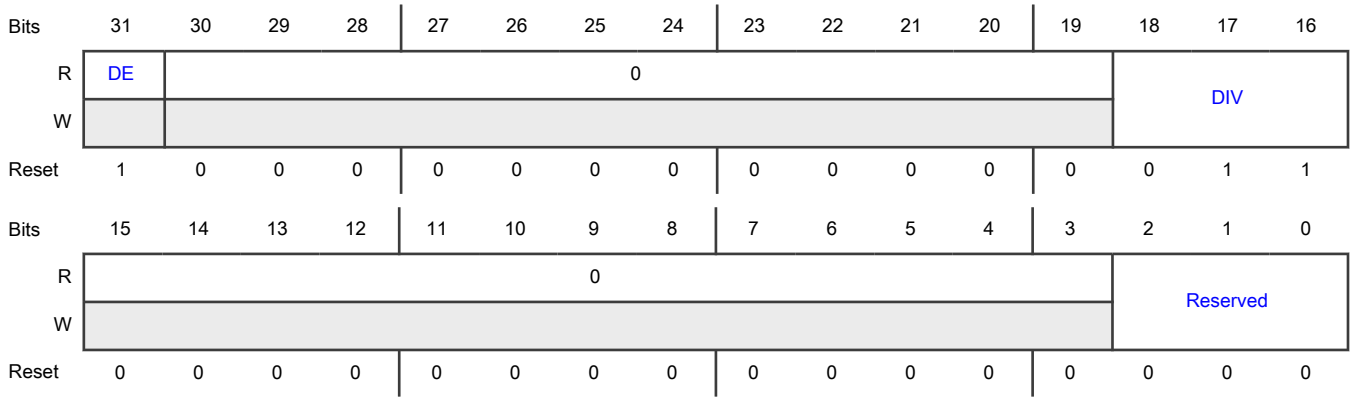
This register controls the clock divider 5 for clock mux 0.

This divider is a 50% duty cycle divider.

**NOTE**

The update to the fields of this register should be an atomic write, that is, one single write should update the complete register.

### Diagram



**Fields**

| Field | Function |
|---|---|
| 31<br>DE | Divider enable<br>0b - Unused<br>1b - Divider is enabled. |
| 30-19<br>— | This field is reserved and reads return zeros. |
| 18-16<br>DIV | Division value<br>This field provides the division value for the clock divider. The clock period of the clock after division is 'DIV+1' times the time period of the current input clock to the divider. |
| 15-3<br>— | Reserved |
| 2-0<br>— | Reserved |

### 24.5.14 Clock Mux 0 Divider 6 Control Register (MUX_0_DC_6)

**Offset**

| Register | Offset |
|---|---|
| MUX_0_DC_6 | 320h |

**Function**

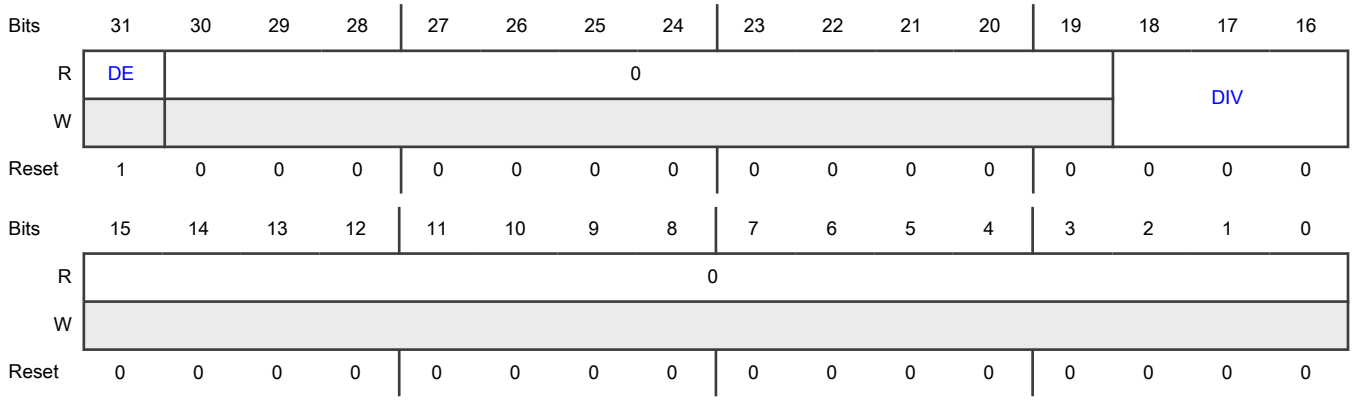
This register controls the clock divider 6 for clock mux 0.

This divider is a 50% duty cycle divider.

**NOTE**

The update to the fields of this register should be an atomic write, that is, one single write should update the complete register.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>DE | Divider enable<br>0b - Unused<br>1b - Divider is enabled. |
| 30-19<br>— | This field is reserved and reads return zeros. |
| 18-16<br>DIV | Division value<br>This field provides the division value for the clock divider. The clock period of the clock after division is 'DIV+1' times the time period of the current input clock to the divider. |
| 15-0<br>— | This field is reserved and reads return zeros. |

## 24.5.15 Clock Mux 0 Divider Trigger Control Register (MUX_0_DIV_TRIG_CTRL)

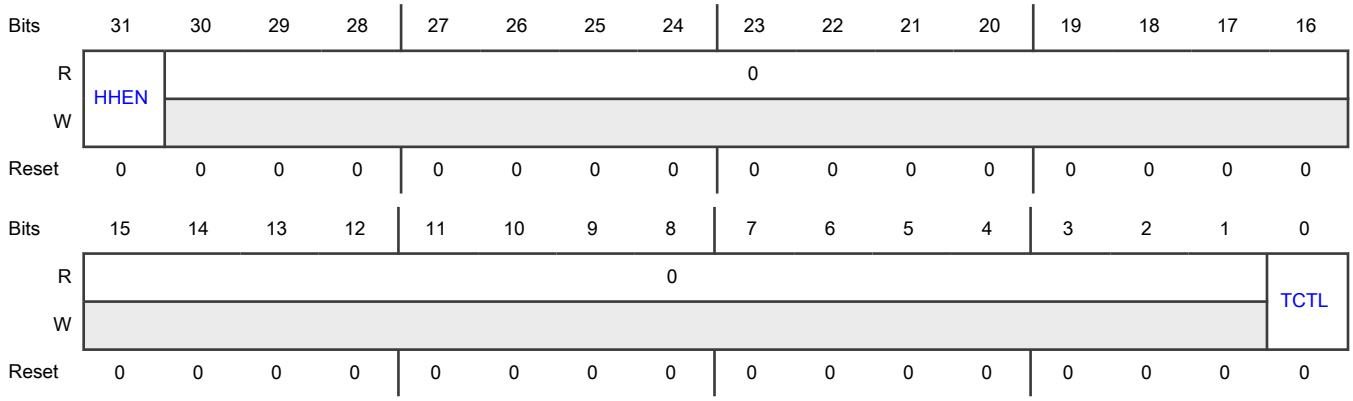
### Offset

| Register | Offset |
|---|---|
| MUX_0_DIV_TRIG_CTRL | 334h |

### Function

This register selects whether the dividers associated with clock mux 0 are updated immediately on writing to the corresponding divider configuration register (referred to as immediate divider update) or only on writing to the MC_CGM_MUX_0_DIV_TRIG register (referred to as common trigger update). When common trigger update is configured, this register also controls initiation of the halt handshake protocol with the on-chip AXBS. Software is required to configure HHEN field for handshaking with on-chip AXBS when the ratio of division value among the clock dividers need to be changed.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>HHEN | Halt handshake enable<br>This field controls the initiation of the halt handshake protocol with AXBS when a common trigger divider update is initiated.<br>0b - No halt handshake protocol is initiated.<br>1b - Halt handshake protocol is initiated. |
| 30-1<br>— | This field is reserved and reads return zeros. |
| 0<br>TCTL | Trigger control<br>This field controls the divider update configuration between immediate and common update.<br>0b - Immediate divider update<br>1b - Common trigger divider update |

## 24.5.16 Clock Mux 0 Divider Trigger Register (MUX_0_DIV_TRIG)

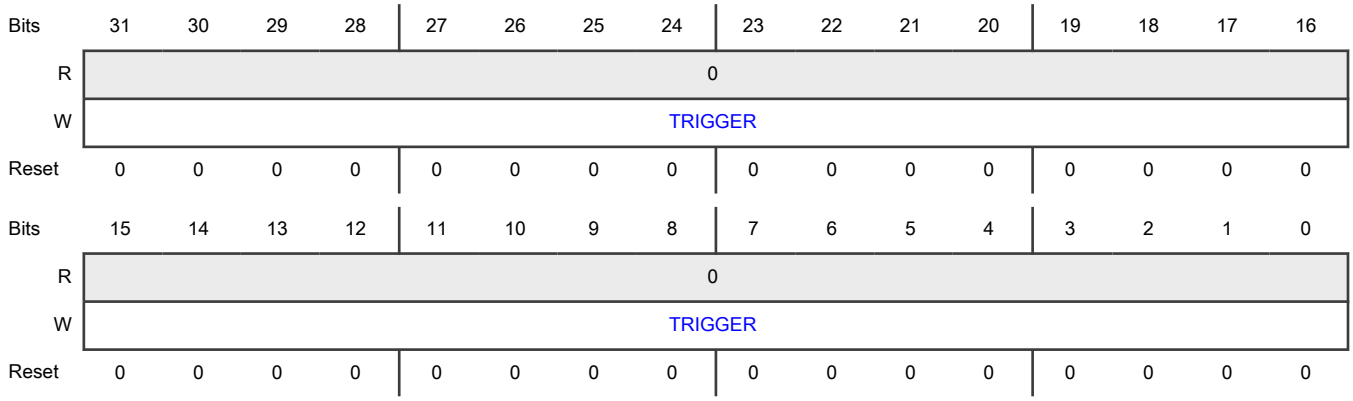
### Offset

| Register | Offset |
|---|---|
| MUX_0_DIV_TRIG | 338h |

### Function

This register provides a common trigger for the clock dividers (only 50% duty cycle dividers) of clock mux 0. Writing any value to this register provides a trigger to the dividers. This register should only be written after appropriately configuring the MC_CGM_MUX_0_DIV_TRIG_CTRL register.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>TRIGGER | Trigger for divider update |

## 24.5.17 Clock Mux 0 Divider Update Status Register (MUX_0_DIV_UPD_STAT)

### Offset

| Register | Offset |
|---|---|
| MUX_0_DIV_UPD_STAT | 33Ch |

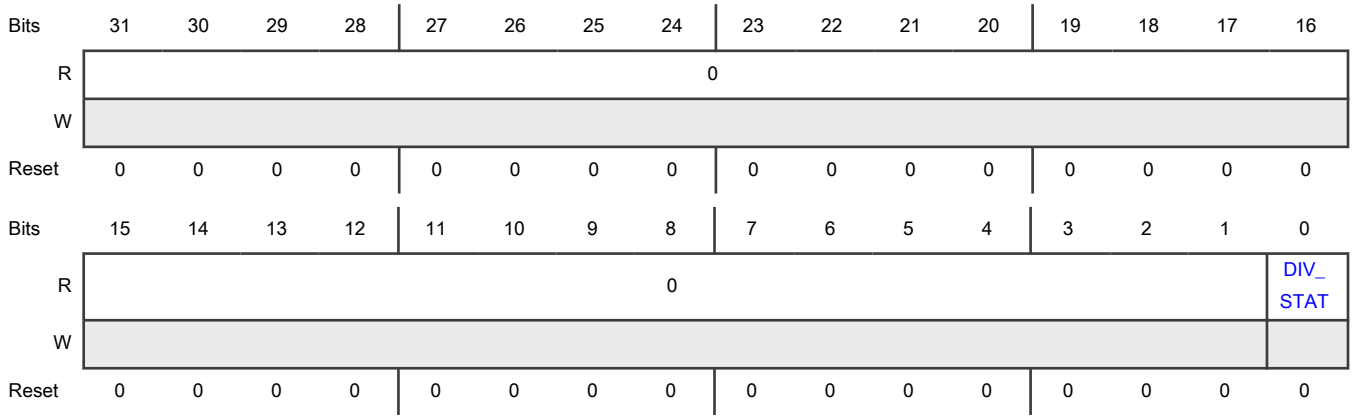
### Function

This register provides the update status of the clock dividers corresponding to clock mux 0. When a write operation on any divider control register is performed, the divider status bit in this register is set to logic-1. The bit is set to logic-0 when the divider has sampled the new divider configuration. Performing multiple writes without tracking the status bit on same or other clock dividers inside the same clock mux leads to inconsistent reporting, that is, the divider status maybe be set to logic-0 when the corresponding divider update is pending.

**NOTE**

Read accesses to MUX_n_DIV_UPD_STAT always complete without returning bus transfer error independent of whether any divider(s) are implemented inside MC_CGM clock mux.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | This field is reserved and reads return zeros. |
| 0<br>DIV_STAT | Divider status for clock mux 0<br>On reading MUX_n_DIV_UPD_STAT after updating a divider control register, if the value of this field is fixed to 1 because of an error in the selected clock source, perform the following steps to switch the mux to a new clock source:<br>1. Switch the mux to a working clock source without polling this field.<br>2. Update MUX_n_DC_m and poll this field.<br>**NOTE**<br>This field clears once divider configuration is updated or on destructive reset. If functional reset comes when this field is 1 then it can remain fixed to 1 until divider input clock is restored.<br>0b - No divider configuration update is pending.<br>1b - Divider configuration update on at least one divider associated with this multiplexer is pending. |

## 24.5.18 Clock Mux 1 Select Control Register (MUX_1_CSC)

### Offset

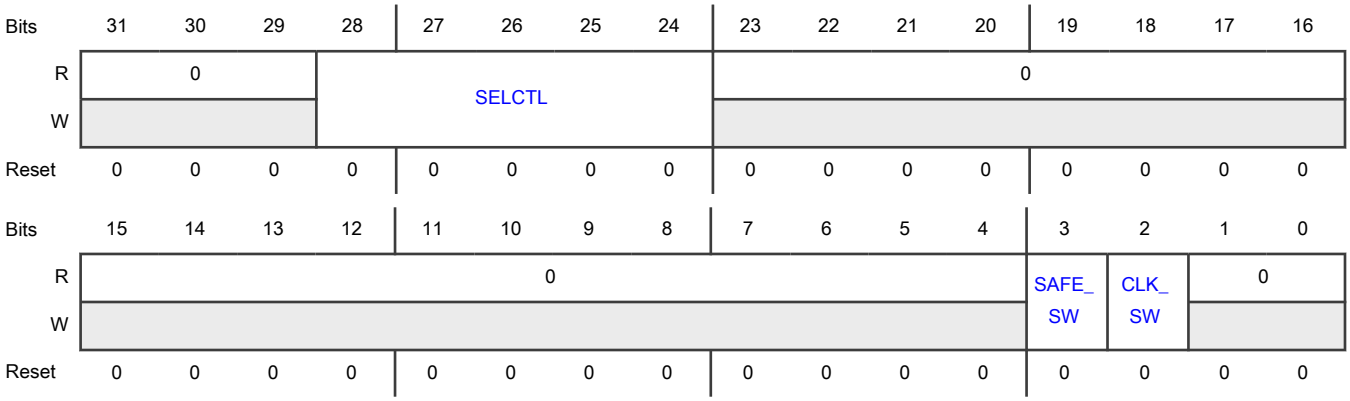
| Register | Offset |
|---|---|
| MUX_1_CSC | 340h |

### Function

This register provides the clock source selection control for clock mux 1. Clock mux 1 implements hardware control clock switching ensuring that the clock switch happens in a graceful manner (without glitches). See the "Hardware-controlled clock multiplexer" section for details.

This register is reset on destructive reset only.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-29<br>— | This field is reserved and reads return zeros. |
| 28-24<br>SELCTL | Clock source selection control<br>Selects the source clock for clock mux 1. The reserved values are not displayed.<br>0_0000b - FIRC<br>0_0010b - FXOSC<br>1_0110b - AIPS_PLAT_CLK |
| 23-4<br>— | This field is reserved and reads return zeros. |
| 3<br>SAFE_SW | Safe clock request<br>Writing 1 to this bit makes a safe clock switch request to FIRC. After a safe clock switch operation is requested, this bit is auto cleared and a corresponding bit in the status register is set. |
| 2<br>CLK_SW | Clock switch<br>Writing 1 to this bit makes a clock switch request to clock mux 1. After a clock switch operation is requested, this bit is auto cleared and a corresponding bit in the status register is set. |
| 1-0<br>— | This field is reserved and reads return zeros. |

## 24.5.19 Clock Mux 1 Select Status Register (MUX_1_CSS)

### Offset

| Register | Offset |
|---|---|
| MUX_1_CSS | 344h |

### Function

This register provides the current clock source selection status for clock mux 1.

This register is reset on destructive reset only.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | SELSTAT | | | | | 0 | | | | SWTRG | | | SWIP |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31-29<br>— | This field is reserved and reads return zeros. |
| 28-24<br>SELSTAT | Clock source selection status<br>This value indicates the current source selected for clock mux 1. The reserved values are not displayed.<br>0_0000b - FIRC<br>0_0010b - FXOSC<br>1_0110b - AIPS_PLAT_CLK |
| 23-20<br>— | This field is reserved and reads return zeros. |
| 19-17<br>SWTRG | Switch trigger cause<br>This value indicates the cause for the latest clock source switch. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>If the clock fails, followed by multiple safe clock switch requests for MC_CGM hardware clock mux, the value of the SWTRG field can be either 4 or 5.<br>000b - Reserved<br>001b - Switch after request succeeded.<br>010b - Switch after the request failed because of an inactive target clock and the current clock is FIRC.<br>011b - Switch after the request failed because of an inactive current clock and the current clock is FIRC.<br>100b - Switch to FIRC because of a safe clock request or reset succeeded.<br>101b - Switch to FIRC because of a safe clock request or reset succeeded, but the previous current clock source was inactive.<br>110b - Reserved<br>111b - Reserved |
| 16<br>SWIP | Switch in progress<br>**NOTE**<br>New clock switch request can only be given three clock cycles after the completion of the previous request.<br>0b - Clock source switching is complete.<br>1b - Clock source switching is in progress. |
| 15-4<br>— | This field is reserved and reads return zeros. |
| 3<br>SAFE_SW | Safe clock request<br>This field provides an indication of whether a switch to safe clock operation was requested during the previous/ongoing request on clock mux 1.<br>0b - No safe clock switch operation was requested.<br>1b - Safe clock switch operation was requested. |
| 2<br>CLK_SW | Clock switch<br>This field provides an indication of whether a clock switch operation was requested during the previous/ongoing request on clock mux 1.<br>0b - No clock switch operation was requested.<br>1b - Clock switch operation was requested. |
| 1-0<br>— | This field is reserved and reads return zeros. |

## 24.5.20 Clock Mux 1 Divider 0 Control Register (MUX_1_DC_0)

### Offset

| Register | Offset |
|---|---|
| MUX_1_DC_0 | 348h |

### Function

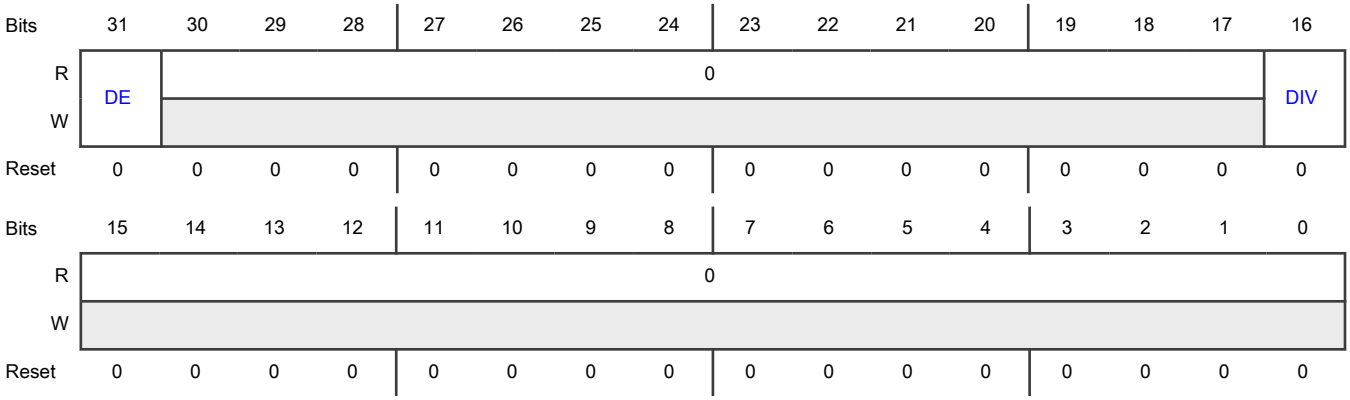
This register controls the clock divider 0 for clock mux 1.

This divider is a 50% duty cycle divider.

**NOTE**

The update to the fields of this register should be an atomic write, that is, one single write should update the complete register.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>DE | Divider enable<br>0b - Divider is disabled.<br>1b - Divider is enabled. |
| 30-17<br>— | This field is reserved and reads return zeros. |
| 16<br>DIV | Division value<br>This field provides the division value for the clock divider. The clock period of the clock after division is 'DIV+1' times the time period of the current input clock to the divider. |
| 15-0<br>— | This field is reserved and reads return zeros. |

## 24.5.21 Clock Mux 1 Divider Update Status Register (MUX_1_DIV_UPD_STAT)

### Offset

| Register | Offset |
|---|---|
| MUX_1_DIV_UPD_STAT | 37Ch |

### Function

This register provides the update status of the clock dividers corresponding to clock mux 1. When a write operation on any divider control register is performed, the divider status bit in this register is set to logic-1. The bit is set to logic-0 when the divider has sampled the new divider configuration. Performing multiple writes without tracking the status bit on same or other clock dividers inside the same clock mux leads to inconsistent reporting, that is, the divider status maybe be set to logic-0 when the corresponding divider update is pending.

**NOTE**

Read accesses to MUX_n_DIV_UPD_STAT always complete without returning bus transfer error independent of whether any divider(s) are implemented inside MC_CGM clock mux.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-1<br>— | This field is reserved and reads return zeros. |
| 0<br>DIV_STAT | Divider status for clock mux 1<br>On reading MUX_n_DIV_UPD_STAT after updating a divider control register, if the value of this field is fixed to 1 because of an error in the selected clock source, perform the following steps to switch the mux to a new clock source:<br>1. Switch the mux to a working clock source without polling this field.<br>2. Update MUX_n_DC_m and poll this field. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>This field clears once divider configuration is updated or on destructive reset. If functional reset comes when this field is 1 then it can remain fixed to 1 until divider input clock is restored.<br>0b - No divider configuration update is pending.<br>1b - Divider configuration update on at least one divider associated with this multiplexer is pending. |

## 24.5.22 Clock Mux 2 Select Control Register (MUX_2_CSC)

### Offset

| Register | Offset |
|---|---|
| MUX_2_CSC | 380h |

### Function

This register provides the clock source selection control for clock mux 2. Clock mux 2 implements hardware control clock switching ensuring that the clock switch happens in a graceful manner (without glitches). See the "Hardware-controlled clock multiplexer" section for details.

This register is reset on destructive reset only.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-29 | This field is reserved and reads return zeros. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 28-24<br>SELCTL | Clock source selection control<br>Selects the source clock for clock mux 2. The reserved values are not displayed.<br>0_0000b - FIRC<br>0_0010b - FXOSC<br>1_0110b - AIPS_PLAT_CLK |
| 23-4<br>— | This field is reserved and reads return zeros. |
| 3<br>SAFE_SW | Safe clock request<br>Writing 1 to this bit makes a safe clock switch request to FIRC. After a safe clock switch operation is requested, this bit is auto cleared and a corresponding bit in the status register is set. |
| 2<br>CLK_SW | Clock switch<br>Writing 1 to this bit makes a clock switch request to clock mux 2. After a clock switch operation is requested, this bit is auto cleared and a corresponding bit in the status register is set. |
| 1-0<br>— | This field is reserved and reads return zeros. |

### 24.5.23 Clock Mux 2 Select Status Register (MUX_2_CSS)

#### Offset

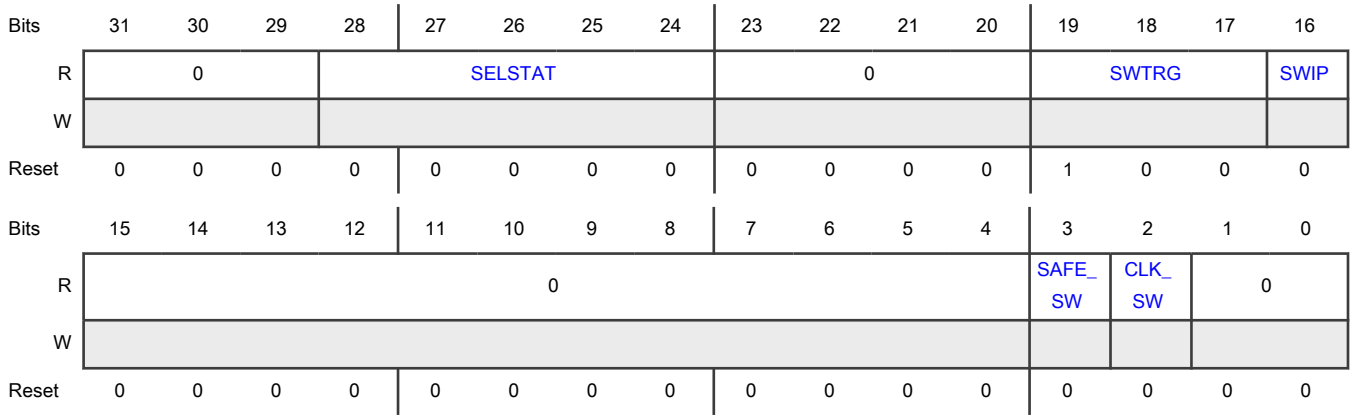
| Register | Offset |
|---|---|
| MUX_2_CSS | 384h |

#### Function

This register provides the current clock source selection status for clock mux 2.

This register is reset on destructive reset only.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-29<br>— | This field is reserved and reads return zeros. |
| 28-24<br>SELSTAT | Clock source selection status<br>This value indicates the current source selected for clock mux 2. The reserved values are not displayed.<br>0_0000b - FIRC<br>0_0010b - FXOSC<br>1_0110b - AIPS_PLAT_CLK |
| 23-20<br>— | This field is reserved and reads return zeros. |
| 19-17<br>SWTRG | Switch trigger cause<br>This value indicates the cause for the latest clock source switch.<br>**NOTE**<br>If the clock fails, followed by multiple safe clock switch requests for MC_CGM hardware clock mux, the value of the SWTRG field can be either 4 or 5.<br>000b - Reserved<br>001b - Switch after request succeeded.<br>010b - Switch after the request failed because of an inactive target clock and the current clock is FIRC.<br>011b - Switch after the request failed because of an inactive current clock and the current clock is FIRC.<br>100b - Switch to FIRC because of a safe clock request or reset succeeded.<br>101b - Switch to FIRC because of a safe clock request or reset succeeded, but the previous current clock source was inactive. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 110b - Reserved<br>111b - Reserved |
| 16<br>SWIP | Switch in progress<br>**NOTE**<br>New clock switch request can only be given three clock cycles after the completion of the previous request.<br>0b - Clock source switching is complete.<br>1b - Clock source switching is in progress. |
| 15-4<br>— | This field is reserved and reads return zeros. |
| 3<br>SAFE_SW | Safe clock request<br>This field provides an indication of whether a switch to safe clock operation was requested during the previous/ongoing request on clock mux 2.<br>0b - No safe clock switch operation was requested.<br>1b - Safe clock switch operation was requested. |
| 2<br>CLK_SW | Clock switch<br>This field provides an indication of whether a clock switch operation was requested during the previous/ongoing request on clock mux 2.<br>0b - No clock switch operation was requested.<br>1b - Clock switch operation was requested. |
| 1-0<br>— | This field is reserved and reads return zeros. |

### 24.5.24 Clock Mux 2 Divider 0 Control Register (MUX_2_DC_0)

**Offset**

| Register | Offset |
|---|---|
| MUX_2_DC_0 | 388h |

**Function**

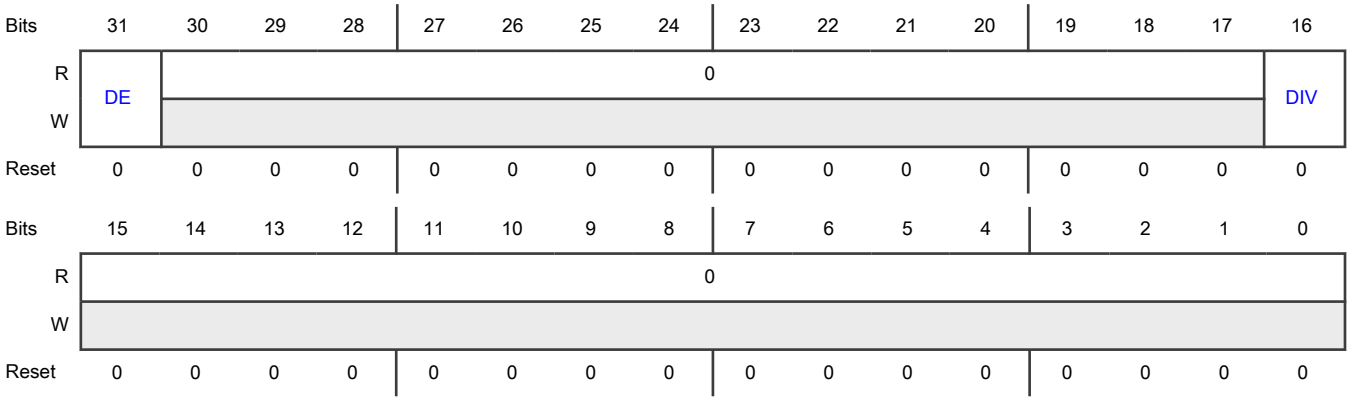
This register controls the clock divider 0 for clock mux 2.

This divider is a 50% duty cycle divider.

**NOTE**

The update to the fields of this register should be an atomic write, that is, one single write should update the complete register.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>DE | Divider enable<br>0b - Divider is disabled.<br>1b - Divider is enabled. |
| 30-17<br>— | This field is reserved and reads return zeros. |
| 16<br>DIV | Division value<br>This field provides the division value for the clock divider. The clock period of the clock after division is 'DIV+1' times the time period of the current input clock to the divider. |
| 15-0<br>— | This field is reserved and reads return zeros. |

## 24.5.25 Clock Mux 2 Divider Update Status Register (MUX_2_DIV_UPD_STAT)

### Offset

| Register | Offset |
|---|---|
| MUX_2_DIV_UPD_STAT | 3BCh |

### Function

This register provides the update status of the clock dividers corresponding to clock mux 2. When a write operation on any divider control register is performed, the divider status bit in this register is set to logic-1. The bit is set to logic-0 when the divider has sampled the new divider configuration. Performing multiple writes without tracking the status bit on same or other

clock dividers inside the same clock mux leads to inconsistent reporting, that is, the divider status maybe be set to logic-0 when the corresponding divider update is pending.

**NOTE**

Read accesses to MUX_n_DIV_UPD_STAT always complete without returning bus transfer error independent of whether any divider(s) are implemented inside MC_CGM clock mux.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-1<br>— | This field is reserved and reads return zeros. |
| 0<br>DIV_STAT | Divider status for clock mux 2<br>On reading MUX_n_DIV_UPD_STAT after updating a divider control register, if the value of this field is fixed to 1 because of an error in the selected clock source, perform the following steps to switch the mux to a new clock source:<br>1. Switch the mux to a working clock source without polling this field.<br>2. Update MUX_n_DC_m and poll this field.<br>**NOTE**<br>This field clears once divider configuration is updated or on destructive reset. If functional reset comes when this field is 1 then it can remain fixed to 1 until divider input clock is restored.<br>0b - No divider configuration update is pending.<br>1b - Divider configuration update on at least one divider associated with this multiplexer is pending. |

## 24.5.26 Clock Mux 3 Select Control Register (MUX_3_CSC)

### Offset

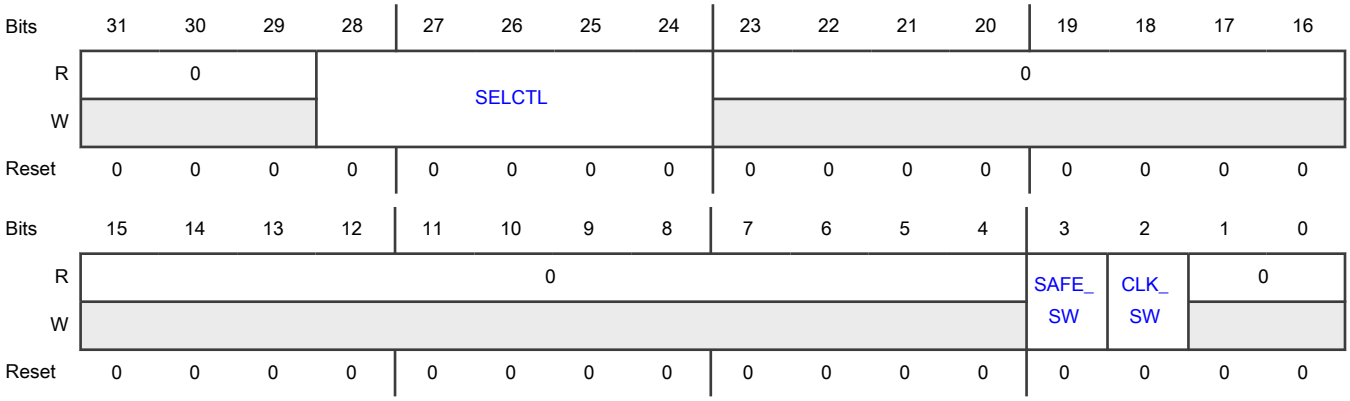
| Register | Offset |
|---|---|
| MUX_3_CSC | 3C0h |

### Function

This register provides the clock source selection control for clock mux 3. Clock mux 3 implements hardware control clock switching ensuring that the clock switch happens in a graceful manner (without glitches). See the "Hardware-controlled clock multiplexer" section for details.

This register is reset on destructive reset only.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-29<br>— | This field is reserved and reads return zeros. |
| 28-24<br>SELCTL | Clock source selection control<br>Selects the source clock for clock mux 3. The reserved values are not displayed.<br>0_0000b - FIRC<br>0_0010b - FXOSC<br>1_0110b - AIPS_PLAT_CLK |
| 23-4<br>— | This field is reserved and reads return zeros. |
| 3<br>SAFE_SW | Safe clock request |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Writing 1 to this bit makes a safe clock switch request to FIRC. After a safe clock switch operation is requested, this bit is auto cleared and a corresponding bit in the status register is set. |
| 2<br>CLK_SW | Clock switch<br>Writing 1 to this bit makes a clock switch request to clock mux 3. After a clock switch operation is requested, this bit is auto cleared and a corresponding bit in the status register is set. |
| 1-0<br>— | This field is reserved and reads return zeros. |

## 24.5.27 Clock Mux 3 Select Status Register (MUX_3_CSS)

### Offset

| Register | Offset |
|---|---|
| MUX_3_CSS | 3C4h |

### Function

This register provides the current clock source selection status for clock mux 3.

This register is reset on destructive reset only.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | SELSTAT | | | | 0 | | | | SWTRG | | | SWIP |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31-29<br>— | This field is reserved and reads return zeros. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 28-24<br>SELSTAT | Clock source selection status<br>This value indicates the current source selected for clock mux 3. The reserved values are not displayed.<br>0_0000b - FIRC<br>0_0010b - FXOSC<br>1_0110b - AIPS_PLAT_CLK |
| 23-20<br>— | This field is reserved and reads return zeros. |
| 19-17<br>SWTRG | Switch trigger cause<br>This value indicates the cause for the latest clock source switch.<br>**NOTE**<br>If the clock fails, followed by multiple safe clock switch requests for MC_CGM hardware clock mux, the value of the SWTRG field can be either 4 or 5.<br>000b - Reserved<br>001b - Switch after request succeeded.<br>010b - Switch after the request failed because of an inactive target clock and the current clock is FIRC.<br>011b - Switch after the request failed because of an inactive current clock and the current clock is FIRC.<br>100b - Switch to FIRC because of a safe clock request or reset succeeded.<br>101b - Switch to FIRC because of a safe clock request or reset succeeded, but the previous current clock source was inactive.<br>110b - Reserved<br>111b - Reserved |
| 16<br>SWIP | Switch in progress<br>**NOTE**<br>New clock switch request can only be given three clock cycles after the completion of the previous request.<br>0b - Clock source switching is complete.<br>1b - Clock source switching is in progress. |
| 15-4<br>— | This field is reserved and reads return zeros. |
| 3<br>SAFE_SW | Safe clock request |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | This field provides an indication of whether a switch to safe clock operation was requested during the previous/ongoing request on clock mux 3.<br>0b - No safe clock switch operation was requested.<br>1b - Safe clock switch operation was requested. |
| 2<br>CLK_SW | Clock switch<br>This field provides an indication of whether a clock switch operation was requested during the previous/ ongoing request on clock mux 3.<br>0b - No clock switch operation was requested.<br>1b - Clock switch operation was requested. |
| 1-0<br>— | This field is reserved and reads return zeros. |

## 24.5.28 Clock Mux 3 Divider 0 Control Register (MUX_3_DC_0)

### Offset

| Register | Offset |
|---|---|
| MUX_3_DC_0 | 3C8h |

### Function

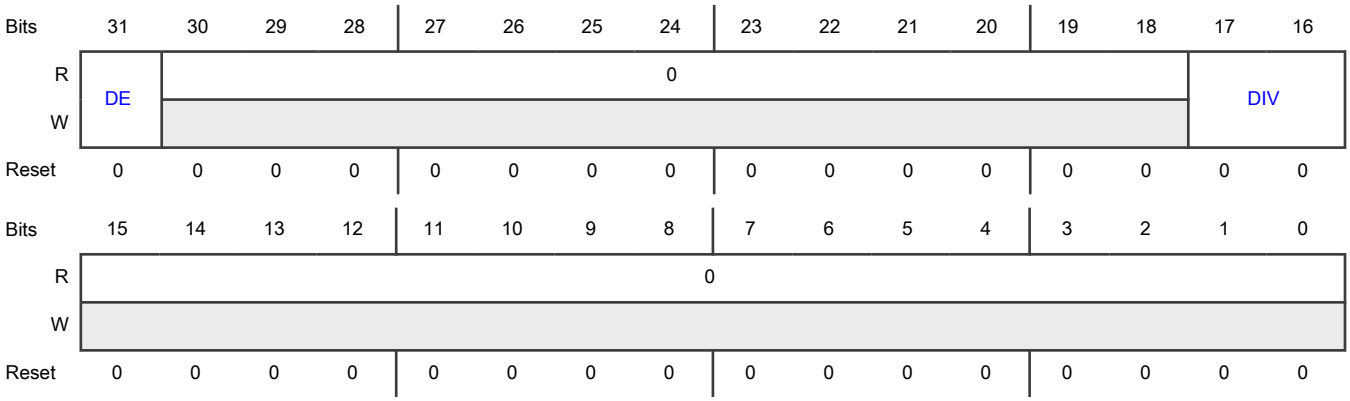
This register controls the clock divider 0 for clock mux 3.

This divider is a 50% duty cycle divider.

**NOTE**

The update to the fields of this register should be an atomic write, that is, one single write should update the complete register.

### Diagram



**Fields**

| Field | Function |
|---|---|
| 31<br>DE | Divider enable<br>0b - Divider is disabled.<br>1b - Divider is enabled. |
| 30-18<br>— | This field is reserved and reads return zeros. |
| 17-16<br>DIV | Division value<br>This field provides the division value for the clock divider. The clock period of the clock after division is 'DIV+1' times the time period of the current input clock to the divider. |
| 15-0<br>— | This field is reserved and reads return zeros. |

### 24.5.29 Clock Mux 3 Divider Update Status Register (MUX_3_DIV_UPD_STAT)

**Offset**

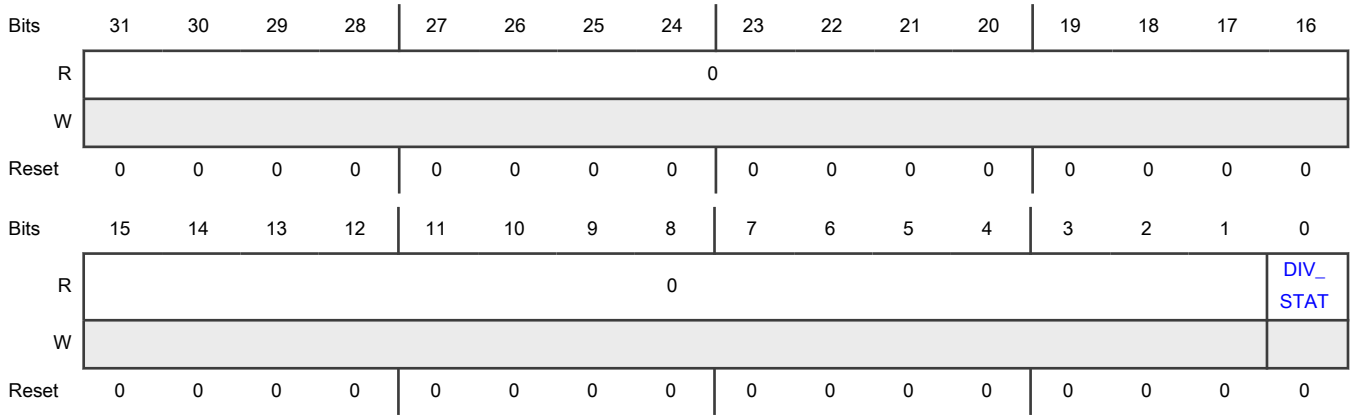
| Register | Offset |
|---|---|
| MUX_3_DIV_UPD_STAT | 3FCh |

**Function**
This register provides the update status of the clock dividers corresponding to clock mux 3. When a write operation on any divider control register is performed, the divider status bit in this register is set to logic-1. The bit is set to logic-0 when the divider has sampled the new divider configuration. Performing multiple writes without tracking the status bit on same or other clock dividers inside the same clock mux leads to inconsistent reporting, that is, the divider status maybe be set to logic-0 when the corresponding divider update is pending.

**NOTE**

Read accesses to MUX_n_DIV_UPD_STAT always complete without returning bus transfer error independent of whether any divider(s) are implemented inside MC_CGM clock mux.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | This field is reserved and reads return zeros. |
| 0<br>DIV_STAT | Divider status for clock mux 3<br>On reading MUX_n_DIV_UPD_STAT after updating a divider control register, if the value of this field is fixed to 1 because of an error in the selected clock source, perform the following steps to switch the mux to a new clock source:<br>1. Switch the mux to a working clock source without polling this field.<br>2. Update MUX_n_DC_m and poll this field.<br>**NOTE**<br>This field clears once divider configuration is updated or on destructive reset. If functional reset comes when this field is 1 then it can remain fixed to 1 until divider input clock is restored.<br>0b - No divider configuration update is pending.<br>1b - Divider configuration update on at least one divider associated with this multiplexer is pending. |

## 24.5.30 Clock Mux 4 Select Control Register (MUX_4_CSC)

### Offset

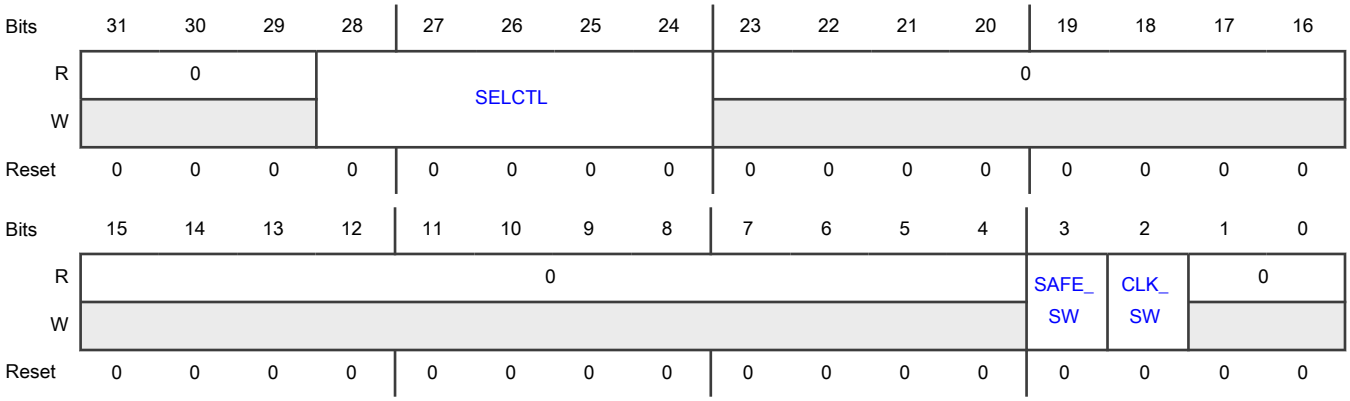
| Register | Offset |
|---|---|
| MUX_4_CSC | 400h |

### Function

This register provides the clock source selection control for clock mux 4. Clock mux 4 implements hardware control clock switching ensuring that the clock switch happens in a graceful manner (without glitches). See the "Hardware-controlled clock multiplexer" section for details.

This register is reset on destructive reset only.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-29<br>— | This field is reserved and reads return zeros. |
| 28-24<br>SELCTL | Clock source selection control<br>Selects the source clock for clock mux 4. The reserved values are not displayed.<br>0_0000b - FIRC<br>0_0010b - FXOSC<br>1_0110b - AIPS_PLAT_CLK |
| 23-4<br>— | This field is reserved and reads return zeros. |
| 3<br>SAFE_SW | Safe clock request<br>Writing 1 to this bit makes a safe clock switch request to FIRC. After a safe clock switch operation is requested, this bit is auto cleared and a corresponding bit in the status register is set. |
| 2<br>CLK_SW | Clock switch<br>Writing 1 to this bit makes a clock switch request to clock mux 4. After a clock switch operation is requested, this bit is auto cleared and a corresponding bit in the status register is set. |
| 1-0<br>— | This field is reserved and reads return zeros. |

## 24.5.31 Clock Mux 4 Select Status Register (MUX_4_CSS)

### Offset

| Register | Offset |
|---|---|
| MUX_4_CSS | 404h |

### Function

This register provides the current clock source selection status for clock mux 4.

This register is reset on destructive reset only.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | SELSTAT | | | | 0 | | | | SWTRG | | | SWIP |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31-29<br>— | This field is reserved and reads return zeros. |
| 28-24<br>SELSTAT | Clock source selection status<br>This value indicates the current source selected for clock mux 4. The reserved values are not displayed.<br>0_0000b - FIRC<br>0_0010b - FXOSC<br>1_0110b - AIPS_PLAT_CLK |
| 23-20<br>— | This field is reserved and reads return zeros. |
| 19-17<br>SWTRG | Switch trigger cause<br>This value indicates the cause for the latest clock source switch. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>If the clock fails, followed by multiple safe clock switch requests for MC_CGM hardware clock mux, the value of the SWTRG field can be either 4 or 5.<br>000b - Reserved<br>001b - Switch after request succeeded.<br>010b - Switch after the request failed because of an inactive target clock and the current clock is FIRC.<br>011b - Switch after the request failed because of an inactive current clock and the current clock is FIRC.<br>100b - Switch to FIRC because of a safe clock request or reset succeeded.<br>101b - Switch to FIRC because of a safe clock request or reset succeeded, but the previous current clock source was inactive.<br>110b - Reserved<br>111b - Reserved |
| 16<br>SWIP | Switch in progress<br>**NOTE**<br>New clock switch request can only be given three clock cycles after the completion of the previous request.<br>0b - Clock source switching is complete.<br>1b - Clock source switching is in progress. |
| 15-4<br>— | This field is reserved and reads return zeros. |
| 3<br>SAFE_SW | Safe clock request<br>This field provides an indication of whether a switch to safe clock operation was requested during the previous/ongoing request on clock mux 4.<br>0b - No safe clock switch operation was requested.<br>1b - Safe clock switch operation was requested. |
| 2<br>CLK_SW | Clock switch<br>This field provides an indication of whether a clock switch operation was requested during the previous/ongoing request on clock mux 4.<br>0b - No clock switch operation was requested.<br>1b - Clock switch operation was requested. |
| 1-0<br>— | This field is reserved and reads return zeros. |

## 24.5.32 Clock Mux 4 Divider 0 Control Register (MUX_4_DC_0)

### Offset

| Register | Offset |
|---|---|
| MUX_4_DC_0 | 408h |

### Function

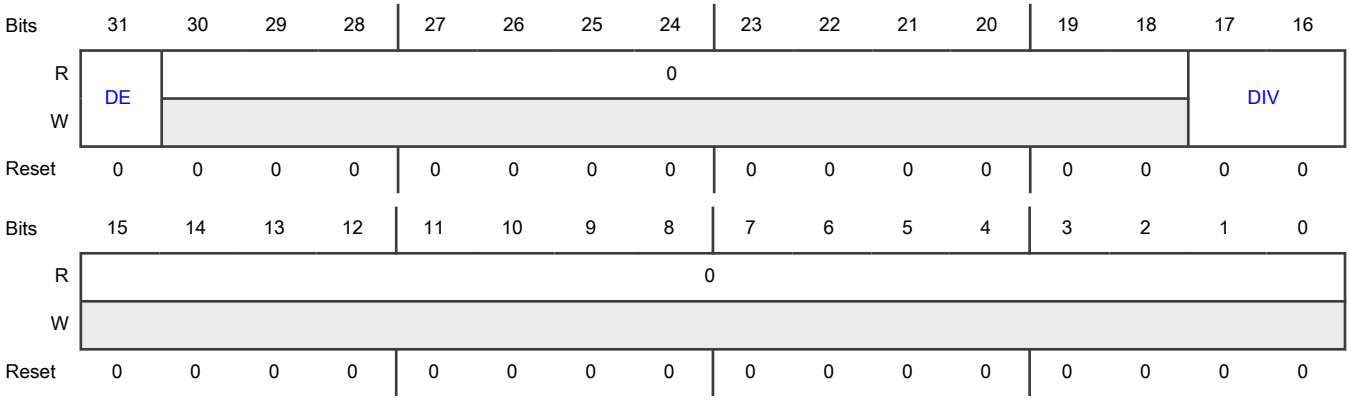
This register controls the clock divider 0 for clock mux 4.

This divider is a 50% duty cycle divider.

**NOTE**

The update to the fields of this register should be an atomic write, that is, one single write should update the complete register.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>DE | Divider enable<br>0b - Divider is disabled.<br>1b - Divider is enabled. |
| 30-18<br>— | This field is reserved and reads return zeros. |
| 17-16<br>DIV | Division value<br>This field provides the division value for the clock divider. The clock period of the clock after division is 'DIV+1' times the time period of the current input clock to the divider. |
| 15-0<br>— | This field is reserved and reads return zeros. |

### 24.5.33 Clock Mux 4 Divider Update Status Register (MUX_4_DIV_UPD_STAT)

#### Offset

| Register | Offset |
|---|---|
| MUX_4_DIV_UPD_STAT | 43Ch |

#### Function

This register provides the update status of the clock dividers corresponding to clock mux 4. When a write operation on any divider control register is performed, the divider status bit in this register is set to logic-1. The bit is set to logic-0 when the divider has sampled the new divider configuration. Performing multiple writes without tracking the status bit on same or other clock dividers inside the same clock mux leads to inconsistent reporting, that is, the divider status maybe be set to logic-0 when the corresponding divider update is pending.

**NOTE**

Read accesses to MUX_n_DIV_UPD_STAT always complete without returning bus transfer error independent of whether any divider(s) are implemented inside MC_CGM clock mux.

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-1<br>— | This field is reserved and reads return zeros. |
| 0<br>DIV_STAT | Divider status for clock mux 4<br>On reading MUX_n_DIV_UPD_STAT after updating a divider control register, if the value of this field is fixed to 1 because of an error in the selected clock source, perform the following steps to switch the mux to a new clock source:<br>1. Switch the mux to a working clock source without polling this field.<br>2. Update MUX_n_DC_m and poll this field. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>This field clears once divider configuration is updated or on destructive reset. If functional reset comes when this field is 1 then it can remain fixed to 1 until divider input clock is restored.<br>0b - No divider configuration update is pending.<br>1b - Divider configuration update on at least one divider associated with this multiplexer is pending. |

## 24.5.34 Clock Mux 5 Select Control Register (MUX_5_CSC)

### Offset

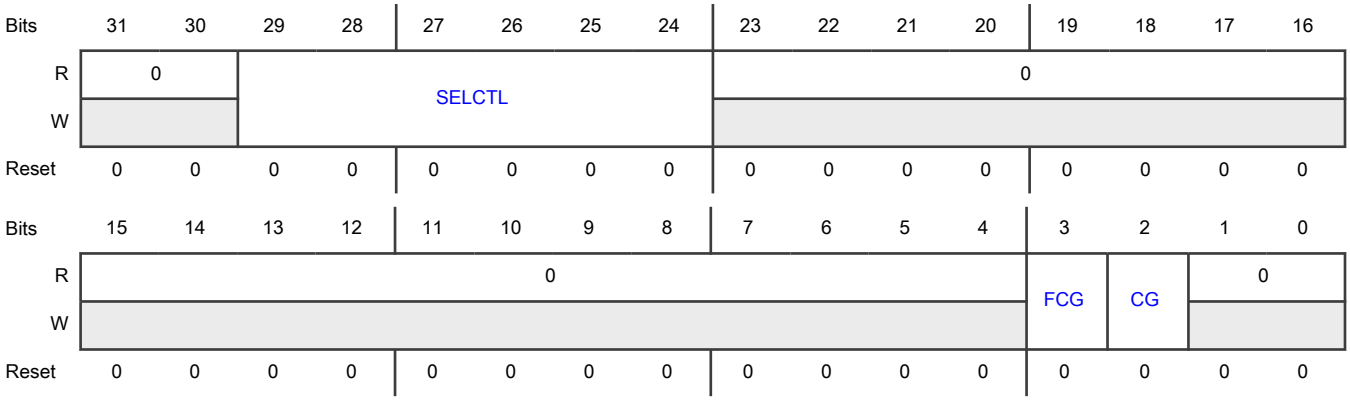
| Register | Offset |
|---|---|
| MUX_5_CSC | 440h |

### Function

This register provides the clock source selection control of clock mux 5. Clock mux 5 implements software control clock switching, and a graceful clock switch can be performed by executing a sequence of steps in software. See "Software-controlled clock multiplexer" section for details.

This register is reset on destructive reset only.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30 | This field is reserved and reads return zeros. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 29-24<br>SELCTL | Clock source selection control<br>Selects the source clock for clock mux 5. The reserved values are not displayed.<br>00_0000b - FIRC<br>00_0001b - SIRC<br>00_0010b - FXOSC<br>00_0100b - SXOSC<br>01_0111b - AIPS_SLOW_CLK<br>11_0010b - Reserved |
| 23-4<br>— | This field is reserved and reads return zeros. |
| 3<br>FCG | Force clock gate<br>Writing 1 to this bit gates the clock at the output of clock mux 5 to logic-0 irrespective of the logic level of the currently selected clock. Clock gating using this bit should only be performed when it is insured that current clock source is inactive. |
| 2<br>CG | Clock gate<br>Writing 1 to this bit gates the clock at the output of clock mux 5 to logic-0. Using this bit it is insured that no glitches are resulted when gating the clock. |
| 1-0<br>— | This field is reserved and reads return zeros. |

### 24.5.35 Clock Mux 5 Select Status Register (MUX_5_CSS)

#### Offset

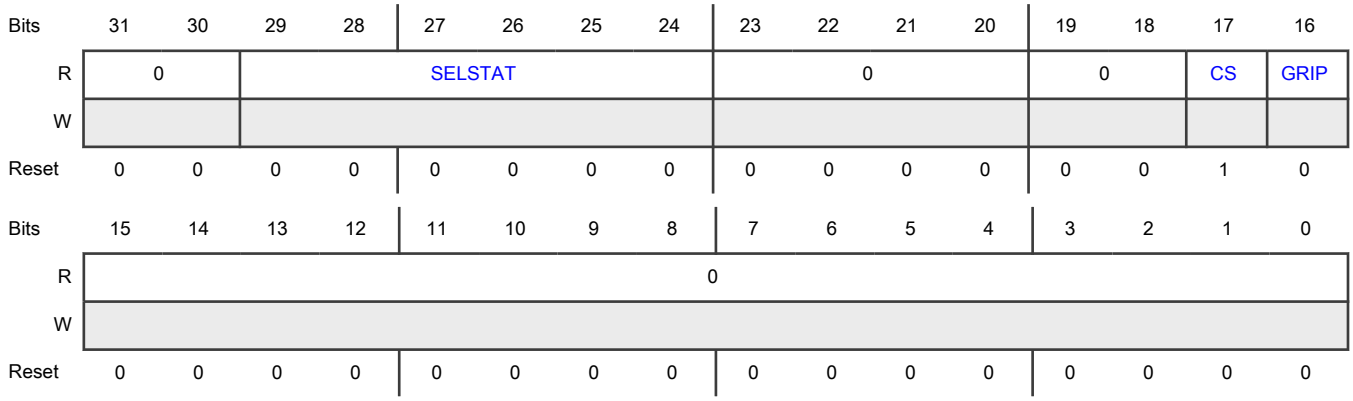
| Register | Offset |
|---|---|
| MUX_5_CSS | 444h |

#### Function

This register provides the current clock source selection status for clock mux 5.

This register is reset on destructive reset only.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | This field is reserved and reads return zeros. |
| 29-24<br>SELSTAT | Clock source selection status<br>This value indicates the current source selected for clock mux 5. The reserved values are not displayed.<br>00_0000b - FIRC<br>00_0001b - SIRC<br>00_0010b - FXOSC<br>00_0100b - SXOSC<br>01_0111b - AIPS_SLOW_CLK<br>11_0010b - Reserved |
| 23-20<br>— | This field is reserved and reads return zeros. |
| 19-18<br>— | This field is reserved and reads return zeros. |
| 17<br>CS | Clock status<br>This field indicates state of the clock at the output of the clock mux.<br>0b - Clock is gated to logic-0 at output of clock mux<br>1b - Clock mux is transparent. Active clock pulses at input of clock mux results in same number of pulses at its output |
| 16<br>GRIP | Gating request is in progress.<br>When a clock gate request is given this bit indicates if the clock gating at the output of mux has completed or not. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Clock source gating or ungating has completed.<br>1b - Clock source gating or ungating is in progress. |
| 15-0<br>— | This field is reserved and reads return zeros. |

## 24.5.36 Clock Mux 5 Divider 0 Control Register (MUX_5_DC_0)

### Offset

| Register | Offset |
|---|---|
| MUX_5_DC_0 | 448h |

### Function

This register controls the clock divider 0 for clock mux 5.

This divider is a 50% duty cycle divider.

**NOTE**

The update to the fields of this register should be an atomic write, that is, one single write should update the complete register.

**NOTE**

Software-controlled clock multiplexer dividers are not expected to return to the default state on the hardware transitions and handshakes occurring as part of the functional reset entry sequence (only hardware-controlled clock multiplexer dividers return to the default state).

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DE | 0 | | | | | | | | | | | DIV | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

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

**Fields**

| Field | Function |
|---|---|
| 31<br>DE | Divider enable<br>0b - Divider is disabled.<br>1b - Divider is enabled. |
| 30-19<br>— | This field is reserved and reads return zeros. |
| 18-16<br>DIV | Division value<br>This field provides the division value for the clock divider. The clock period of the clock after division is 'DIV+1' times the time period of the current input clock to the divider. |
| 15-0<br>— | This field is reserved and reads return zeros. |

### 24.5.37 Clock Mux 5 Divider Update Status Register (MUX_5_DIV_UPD_STAT)

**Offset**

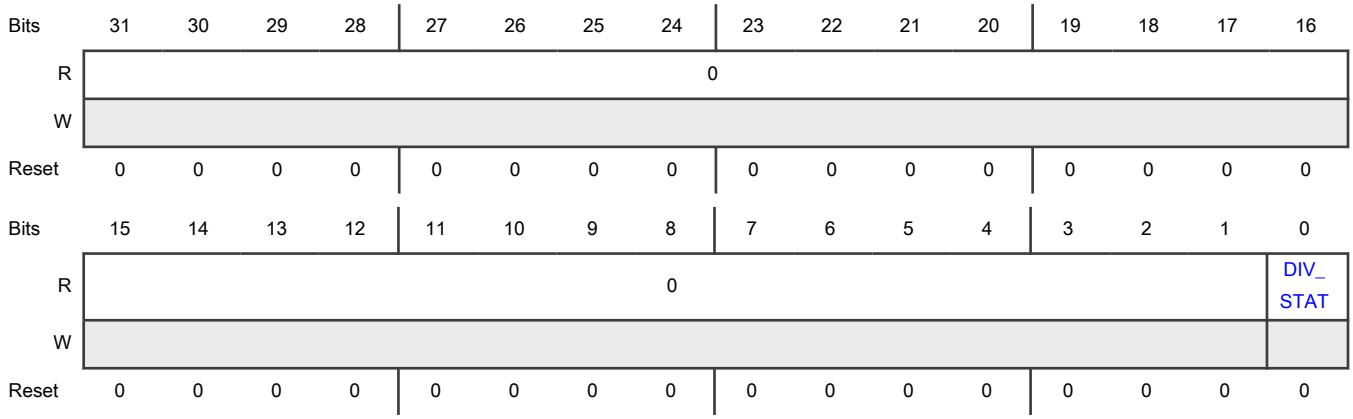
| Register | Offset |
|---|---|
| MUX_5_DIV_UPD_STAT | 47Ch |

**Function**
This register provides the update status of the clock dividers corresponding to clock mux 5. When a write operation on any divider control register is performed, the divider status bit in this register is set to logic-1. The bit is set to logic-0 when the divider has sampled the new divider configuration. Performing multiple writes without tracking the status bit on same or other clock dividers inside the same clock mux leads to inconsistent reporting, that is, the divider status maybe be set to logic-0 when the corresponding divider update is pending.

**NOTE**

Read accesses to MUX_n_DIV_UPD_STAT always complete without returning bus transfer error independent of whether any divider(s) are implemented inside MC_CGM clock mux.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | This field is reserved and reads return zeros. |
| 0<br>DIV_STAT | Divider status for clock mux 5<br>On reading MUX_n_DIV_UPD_STAT after updating a divider control register, if the value of this field is fixed to 1 because of an error in the selected clock source, perform the following steps to switch the mux to a new clock source:<br>1. Switch the mux to a working clock source without polling this field.<br>2. Update MUX_n_DC_m and poll this field.<br>**NOTE**<br>This field clears once divider configuration is updated or on destructive reset. If functional reset comes when this field is 1 then it can remain fixed to 1 until divider input clock is restored.<br>0b - No divider configuration update is pending.<br>1b - Divider configuration update on at least one divider associated with this multiplexer is pending. |

## 24.5.38 Clock Mux 6 Select Control Register (MUX_6_CSC)

### Offset

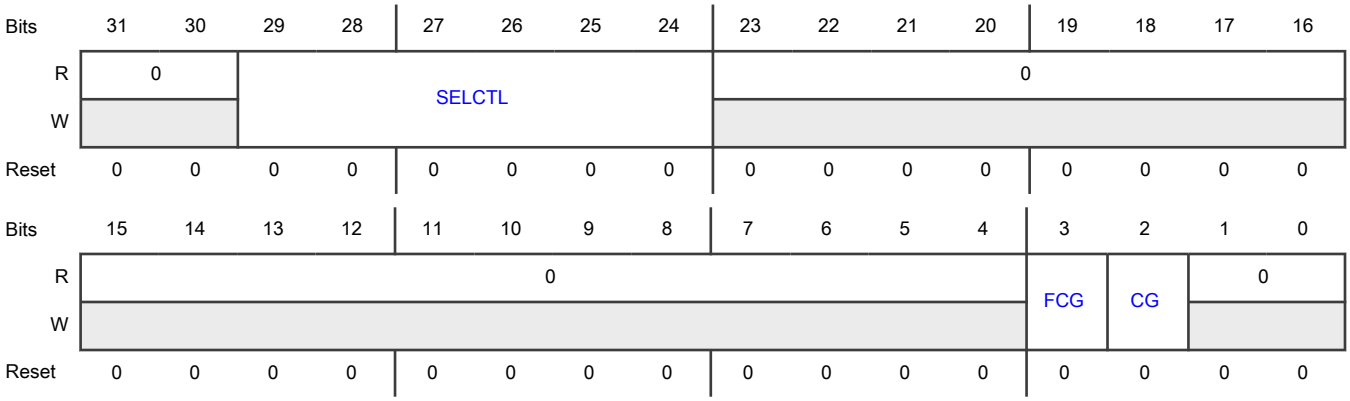
| Register | Offset |
|---|---|
| MUX_6_CSC | 480h |

### Function

This register provides the clock source selection control of clock mux 6. Clock mux 6 implements software control clock switching, and a graceful clock switch can be performed by executing a sequence of steps in software. See "Software-controlled clock multiplexer" section for details.

This register is reset on destructive reset only.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | This field is reserved and reads return zeros. |
| 29-24<br>SELCTL | Clock source selection control<br>Selects the source clock for clock mux 6. The reserved values are not displayed.<br>00_0000b - FIRC<br>00_0001b - SIRC<br>00_0010b - FXOSC<br>00_0100b - SXOSC<br>00_1000b - PLL_PHI0_CLK<br>00_1001b - PLL_PHI1_CLK<br>01_0000b - CORE_CLK<br>01_0011b - HSE_CLK<br>01_0110b - AIPS_PLAT_CLK<br>01_0111b - AIPS_SLOW_CLK<br>01_1000b - EMAC_RMII_TX_CLK<br>01_1001b - EMAC_RX_CLK<br>11_0011b - Reserved<br>11_0100b - Reserved<br>11_0101b - Reserved<br>11_0110b - Reserved<br>11_0111b - Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 11_1000b - Reserved<br>11_1001b - Reserved<br>11_1010b - Reserved<br>11_1011b - Reserved<br>11_1100b - Reserved<br>11_1101b - Reserved<br>11_1110b - Reserved |
| 23-4<br>— | This field is reserved and reads return zeros. |
| 3<br>FCG | Force clock gate<br>Writing 1 to this bit gates the clock at the output of clock mux 6 to logic-0 irrespective of the logic level of the currently selected clock. Clock gating using this bit should only be performed when it is insured that current clock source is inactive. |
| 2<br>CG | Clock gate<br>Writing 1 to this bit gates the clock at the output of clock mux 6 to logic-0. Using this bit it is insured that no glitches are resulted when gating the clock. |
| 1-0<br>— | This field is reserved and reads return zeros. |

## 24.5.39 Clock Mux 6 Select Status Register (MUX_6_CSS)

### Offset

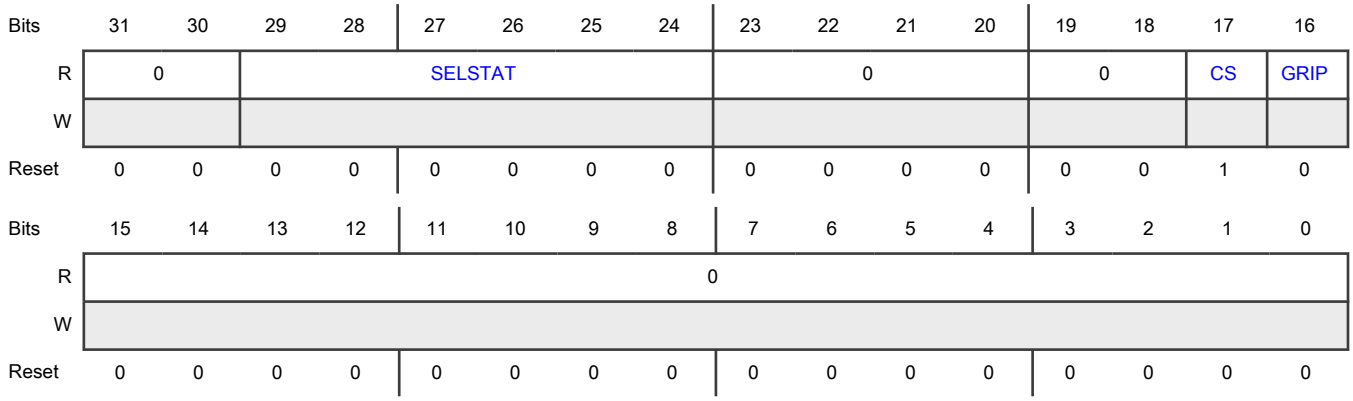
| Register | Offset |
|---|---|
| MUX_6_CSS | 484h |

### Function

This register provides the current clock source selection status for clock mux 6.

This register is reset on destructive reset only.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | This field is reserved and reads return zeros. |
| 29-24<br>SELSTAT | Clock source selection status<br>This value indicates the current source selected for clock mux 6. The reserved values are not displayed.<br>00_0000b - FIRC<br>00_0001b - SIRC<br>00_0010b - FXOSC<br>00_0100b - SXOSC<br>00_1000b - PLL_PHI0_CLK<br>00_1001b - PLL_PHI1_CLK<br>01_0000b - CORE_CLK<br>01_0011b - HSE_CLK<br>01_0110b - AIPS_PLAT_CLK<br>01_0111b - AIPS_SLOW_CLK<br>01_1000b - EMAC_RMII_TX_CLK<br>01_1001b - EMAC_RX_CLK<br>11_0011b - Reserved<br>11_0100b - Reserved<br>11_0101b - Reserved<br>11_0110b - Reserved<br>11_0111b - Reserved<br>11_1000b - Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 11_1001b - Reserved<br>11_1010b - Reserved<br>11_1011b - Reserved<br>11_1100b - Reserved<br>11_1101b - Reserved<br>11_1110b - Reserved |
| 23-20<br>— | This field is reserved and reads return zeros. |
| 19-18<br>— | This field is reserved and reads return zeros. |
| 17<br>CS | Clock status<br>This field indicates state of the clock at the output of the clock mux.<br>0b - Clock is gated to logic-0 at output of clock mux<br>1b - Clock mux is transparent. Active clock pulses at input of clock mux results in same number of pulses at its output |
| 16<br>GRIP | Gating request is in progress.<br>When a clock gate request is given this bit indicates if the clock gating at the output of mux has completed or not.<br>0b - Clock source gating or ungating has completed.<br>1b - Clock source gating or ungating is in progress. |
| 15-0<br>— | This field is reserved and reads return zeros. |

### 24.5.40 Clock Mux 6 Divider 0 Control Register (MUX_6_DC_0)

#### Offset

| Register | Offset |
|---|---|
| MUX_6_DC_0 | 488h |

#### Function

This register controls the clock divider 0 for clock mux 6.

This divider is a 50% duty cycle divider.

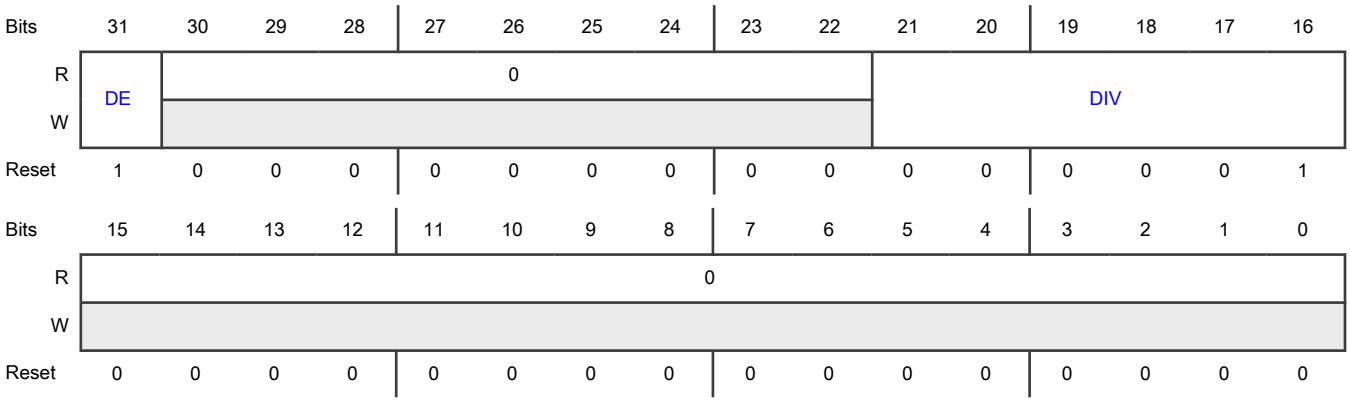
**NOTE**

The update to the fields of this register should be an atomic write, that is, one single write should update the complete register.

**NOTE**

Software-controlled clock multiplexer dividers are not expected to return to the default state on the hardware transitions and handshakes occurring as part of the functional reset entry sequence (only hardware-controlled clock multiplexer dividers return to the default state).

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>DE | Divider enable<br>0b - Divider is disabled.<br>1b - Divider is enabled. |
| 30-22<br>— | This field is reserved and reads return zeros. |
| 21-16<br>DIV | Division value<br>This field provides the division value for the clock divider. The clock period of the clock after division is 'DIV+1' times the time period of the current input clock to the divider. |
| 15-0<br>— | This field is reserved and reads return zeros. |

## 24.5.41 Clock Mux 6 Divider Update Status Register (MUX_6_DIV_UPD_STAT)

### Offset

| Register | Offset |
|---|---|
| MUX_6_DIV_UPD_STAT | 4BCh |

### Function

This register provides the update status of the clock dividers corresponding to clock mux 6. When a write operation on any divider control register is performed, the divider status bit in this register is set to logic-1. The bit is set to logic-0 when the divider has sampled the new divider configuration. Performing multiple writes without tracking the status bit on same or other clock dividers inside the same clock mux leads to inconsistent reporting, that is, the divider status maybe be set to logic-0 when the corresponding divider update is pending.

**NOTE**

Read accesses to MUX_n_DIV_UPD_STAT always complete without returning bus transfer error independent of whether any divider(s) are implemented inside MC_CGM clock mux.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-1<br>— | This field is reserved and reads return zeros. |
| 0<br>DIV_STAT | Divider status for clock mux 6<br>On reading MUX_n_DIV_UPD_STAT after updating a divider control register, if the value of this field is fixed to 1 because of an error in the selected clock source, perform the following steps to switch the mux to a new clock source:<br>1. Switch the mux to a working clock source without polling this field.<br>2. Update MUX_n_DC_m and poll this field.<br>**NOTE**<br>This field clears once divider configuration is updated or on destructive reset. If functional reset comes when this field is 1 then it can remain fixed to 1 until divider input clock is restored.<br>0b - No divider configuration update is pending.<br>1b - Divider configuration update on at least one divider associated with this multiplexer is pending. |

## 24.5.42 Clock Mux 7 Select Control Register (MUX_7_CSC)

### Offset

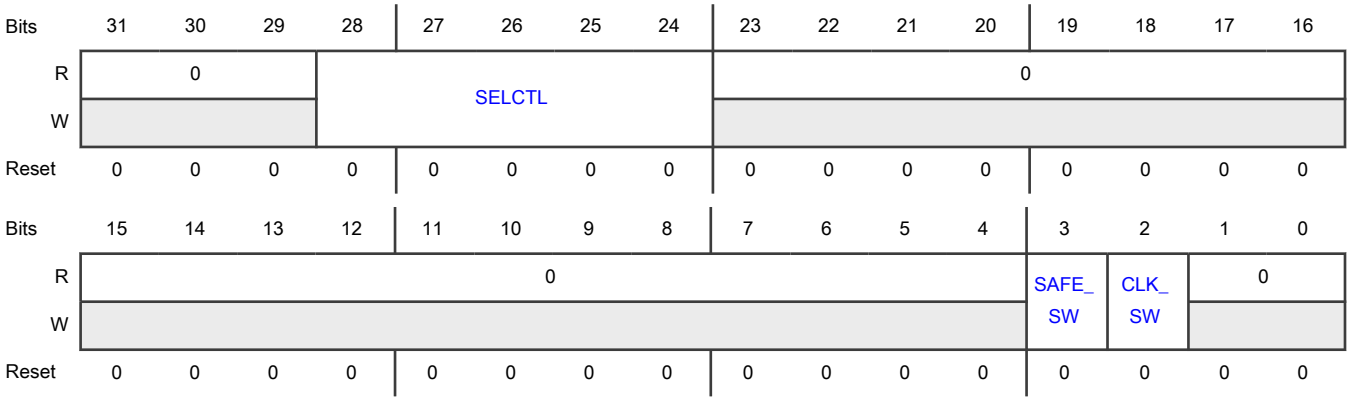
| Register | Offset |
|---|---|
| MUX_7_CSC | 4C0h |

### Function

This register provides the clock source selection control for clock mux 7. Clock mux 7 implements hardware control clock switching ensuring that the clock switch happens in a graceful manner (without glitches). See the "Hardware-controlled clock multiplexer" section for details.

This register is reset on destructive reset only.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-29<br>— | This field is reserved and reads return zeros. |
| 28-24<br>SELCTL | Clock source selection control<br>Selects the source clock for clock mux 7. The reserved values are not displayed.<br>0_0000b - FIRC<br>1_1000b - EMAC_RMII_TX_CLK<br>1_1001b - EMAC_RX_CLK |
| 23-4<br>— | This field is reserved and reads return zeros. |
| 3<br>SAFE_SW | Safe clock request |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Writing 1 to this bit makes a safe clock switch request to FIRC. After a safe clock switch operation is requested, this bit is auto cleared and a corresponding bit in the status register is set. |
| 2<br>CLK_SW | Clock switch<br>Writing 1 to this bit makes a clock switch request to clock mux 7. After a clock switch operation is requested, this bit is auto cleared and a corresponding bit in the status register is set. |
| 1-0<br>— | This field is reserved and reads return zeros. |

## 24.5.43 Clock Mux 7 Select Status Register (MUX_7_CSS)

### Offset

| Register | Offset |
|---|---|
| MUX_7_CSS | 4C4h |

### Function

This register provides the current clock source selection status for clock mux 7.

This register is reset on destructive reset only.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | SELSTAT | | | | 0 | | | | SWTRG | | | SWIP |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31-29<br>— | This field is reserved and reads return zeros. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 28-24<br>SELSTAT | Clock source selection status<br>This value indicates the current source selected for clock mux 7. The reserved values are not displayed.<br>0_0000b - FIRC<br>1_1000b - EMAC_RMII_TX_CLK<br>1_1001b - EMAC_RX_CLK |
| 23-20<br>— | This field is reserved and reads return zeros. |
| 19-17<br>SWTRG | Switch trigger cause<br>This value indicates the cause for the latest clock source switch.<br>**NOTE**<br>If the clock fails, followed by multiple safe clock switch requests for MC_CGM hardware clock mux, the value of the SWTRG field can be either 4 or 5.<br>000b - Reserved<br>001b - Switch after request succeeded.<br>010b - Switch after the request failed because of an inactive target clock and the current clock is FIRC.<br>011b - Switch after the request failed because of an inactive current clock and the current clock is FIRC.<br>100b - Switch to FIRC because of a safe clock request or reset succeeded.<br>101b - Switch to FIRC because of a safe clock request or reset succeeded, but the previous current clock source was inactive.<br>110b - Reserved<br>111b - Reserved |
| 16<br>SWIP | Switch in progress<br>**NOTE**<br>New clock switch request can only be given three clock cycles after the completion of the previous request.<br>0b - Clock source switching is complete.<br>1b - Clock source switching is in progress. |
| 15-4<br>— | This field is reserved and reads return zeros. |
| 3<br>SAFE_SW | Safe clock request |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | This field provides an indication of whether a switch to safe clock operation was requested during the previous/ongoing request on clock mux 7.<br>0b - No safe clock switch operation was requested.<br>1b - Safe clock switch operation was requested. |
| 2<br>CLK_SW | Clock switch<br>This field provides an indication of whether a clock switch operation was requested during the previous/ ongoing request on clock mux 7.<br>0b - No clock switch operation was requested.<br>1b - Clock switch operation was requested. |
| 1-0<br>— | This field is reserved and reads return zeros. |

## 24.5.44 Clock Mux 7 Divider 0 Control Register (MUX_7_DC_0)

### Offset

| Register | Offset |
|---|---|
| MUX_7_DC_0 | 4C8h |

### Function

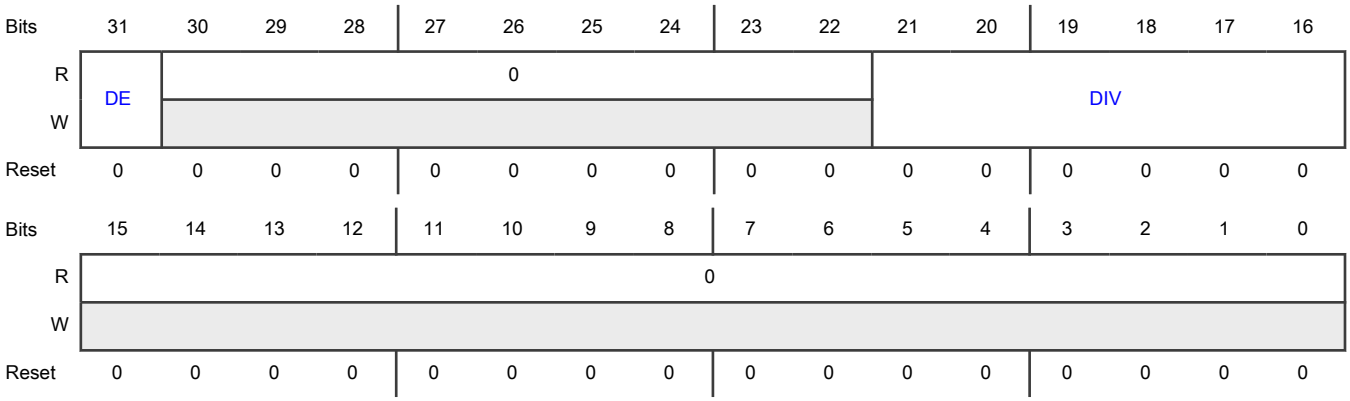
This register controls the clock divider 0 for clock mux 7.

This divider is a 50% duty cycle divider.

**NOTE**

The update to the fields of this register should be an atomic write, that is, one single write should update the complete register.

### Diagram



**Fields**

| Field | Function |
|---|---|
| 31<br>DE | Divider enable<br>0b - Divider is disabled.<br>1b - Divider is enabled. |
| 30-22<br>— | This field is reserved and reads return zeros. |
| 21-16<br>DIV | Division value<br>This field provides the division value for the clock divider. The clock period of the clock after division is 'DIV+1' times the time period of the current input clock to the divider. |
| 15-0<br>— | This field is reserved and reads return zeros. |

### 24.5.45 Clock Mux 7 Divider Update Status Register (MUX_7_DIV_UPD_STAT)

**Offset**

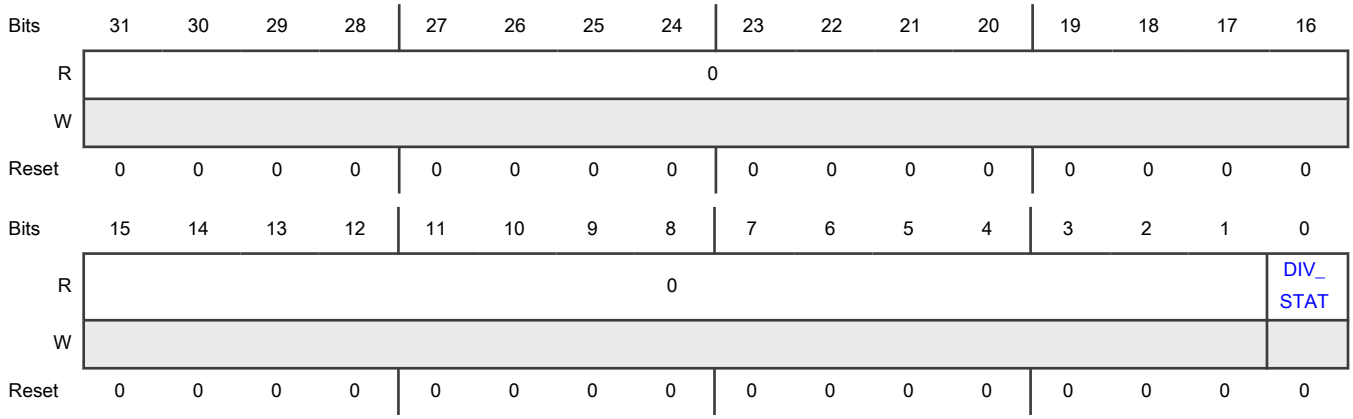
| Register | Offset |
|---|---|
| MUX_7_DIV_UPD_STAT | 4FCh |

**Function**
This register provides the update status of the clock dividers corresponding to clock mux 7. When a write operation on any divider control register is performed, the divider status bit in this register is set to logic-1. The bit is set to logic-0 when the divider has sampled the new divider configuration. Performing multiple writes without tracking the status bit on same or other clock dividers inside the same clock mux leads to inconsistent reporting, that is, the divider status maybe be set to logic-0 when the corresponding divider update is pending.

**NOTE**

Read accesses to MUX_n_DIV_UPD_STAT always complete without returning bus transfer error independent of whether any divider(s) are implemented inside MC_CGM clock mux.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | This field is reserved and reads return zeros. |
| 0<br>DIV_STAT | Divider status for clock mux 7<br>On reading MUX_n_DIV_UPD_STAT after updating a divider control register, if the value of this field is fixed to 1 because of an error in the selected clock source, perform the following steps to switch the mux to a new clock source:<br>1. Switch the mux to a working clock source without polling this field.<br>2. Update MUX_n_DC_m and poll this field.<br>**NOTE**<br>This field clears once divider configuration is updated or on destructive reset. If functional reset comes when this field is 1 then it can remain fixed to 1 until divider input clock is restored.<br>0b - No divider configuration update is pending.<br>1b - Divider configuration update on at least one divider associated with this multiplexer is pending. |

## 24.5.46 Clock Mux 8 Select Control Register (MUX_8_CSC)

### Offset

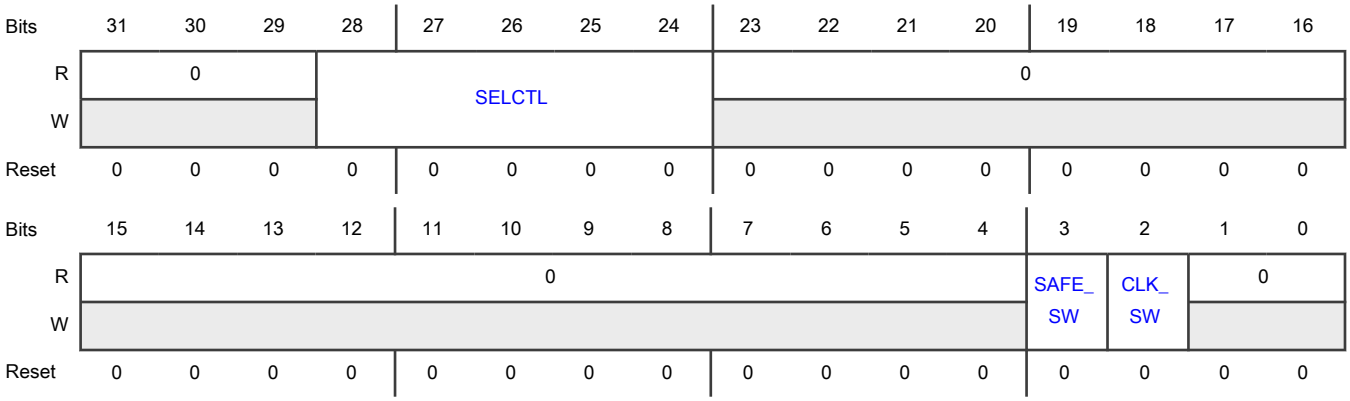
| Register | Offset |
|---|---|
| MUX_8_CSC | 500h |

### Function

This register provides the clock source selection control for clock mux 8. Clock mux 8 implements hardware control clock switching ensuring that the clock switch happens in a graceful manner (without glitches). See the "Hardware-controlled clock multiplexer" section for details.

This register is reset on destructive reset only.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-29<br>— | This field is reserved and reads return zeros. |
| 28-24<br>SELCTL | Clock source selection control<br>Selects the source clock for clock mux 8. The reserved values are not displayed.<br>0_0000b - FIRC<br>1_1000b - EMAC_RMII_TX_CLK |
| 23-4<br>— | This field is reserved and reads return zeros. |
| 3<br>SAFE_SW | Safe clock request<br>Writing 1 to this bit makes a safe clock switch request to FIRC. After a safe clock switch operation is requested, this bit is auto cleared and a corresponding bit in the status register is set. |
| 2<br>CLK_SW | Clock switch<br>Writing 1 to this bit makes a clock switch request to clock mux 8. After a clock switch operation is requested, this bit is auto cleared and a corresponding bit in the status register is set. |
| 1-0<br>— | This field is reserved and reads return zeros. |

## 24.5.47 Clock Mux 8 Select Status Register (MUX_8_CSS)

### Offset

| Register | Offset |
|---|---|
| MUX_8_CSS | 504h |

### Function

This register provides the current clock source selection status for clock mux 8.

This register is reset on destructive reset only.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | SELSTAT | | | | 0 | | | | SWTRG | | | SWIP |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31-29<br>— | This field is reserved and reads return zeros. |
| 28-24<br>SELSTAT | Clock source selection status<br>This value indicates the current source selected for clock mux 8. The reserved values are not displayed.<br>0_0000b - FIRC<br>1_1000b - EMAC_RMII_TX_CLK |
| 23-20<br>— | This field is reserved and reads return zeros. |
| 19-17<br>SWTRG | Switch trigger cause<br>This value indicates the cause for the latest clock source switch. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>If the clock fails, followed by multiple safe clock switch requests for MC_CGM hardware clock mux, the value of the SWTRG field can be either 4 or 5.<br>000b - Reserved<br>001b - Switch after request succeeded.<br>010b - Switch after the request failed because of an inactive target clock and the current clock is FIRC.<br>011b - Switch after the request failed because of an inactive current clock and the current clock is FIRC.<br>100b - Switch to FIRC because of a safe clock request or reset succeeded.<br>101b - Switch to FIRC because of a safe clock request or reset succeeded, but the previous current clock source was inactive.<br>110b - Reserved<br>111b - Reserved |
| 16<br>SWIP | Switch in progress<br>**NOTE**<br>New clock switch request can only be given three clock cycles after the completion of the previous request.<br>0b - Clock source switching is complete.<br>1b - Clock source switching is in progress. |
| 15-4<br>— | This field is reserved and reads return zeros. |
| 3<br>SAFE_SW | Safe clock request<br>This field provides an indication of whether a switch to safe clock operation was requested during the previous/ongoing request on clock mux 8.<br>0b - No safe clock switch operation was requested.<br>1b - Safe clock switch operation was requested. |
| 2<br>CLK_SW | Clock switch<br>This field provides an indication of whether a clock switch operation was requested during the previous/ongoing request on clock mux 8.<br>0b - No clock switch operation was requested.<br>1b - Clock switch operation was requested. |
| 1-0<br>— | This field is reserved and reads return zeros. |

## 24.5.48 Clock Mux 8 Divider 0 Control Register (MUX_8_DC_0)

### Offset

| Register | Offset |
|---|---|
| MUX_8_DC_0 | 508h |

### Function

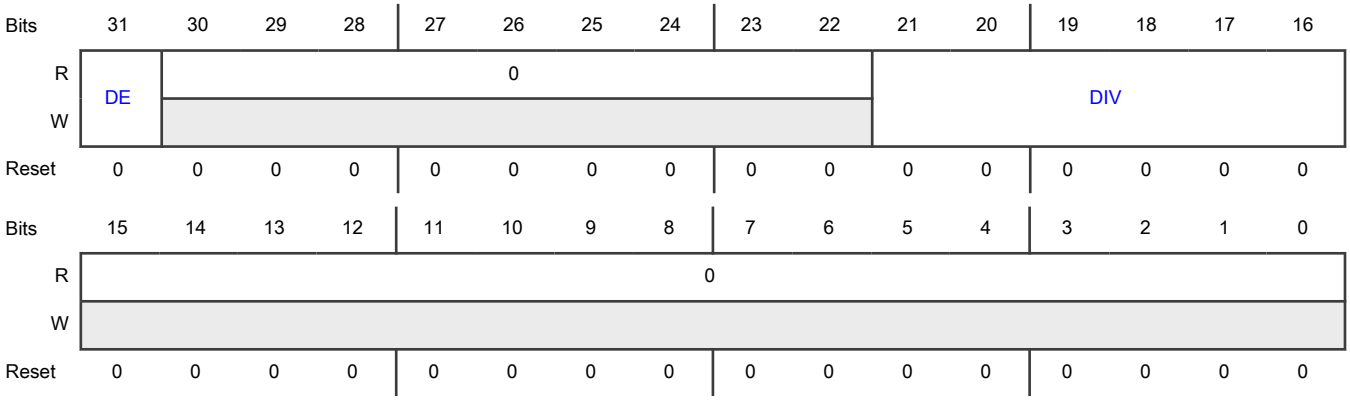
This register controls the clock divider 0 for clock mux 8.

This divider is a 50% duty cycle divider.

**NOTE**

The update to the fields of this register should be an atomic write, that is, one single write should update the complete register.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>DE | Divider enable<br>0b - Divider is disabled.<br>1b - Divider is enabled. |
| 30-22<br>— | This field is reserved and reads return zeros. |
| 21-16<br>DIV | Division value<br>This field provides the division value for the clock divider. The clock period of the clock after division is 'DIV+1' times the time period of the current input clock to the divider. |
| 15-0<br>— | This field is reserved and reads return zeros. |

### 24.5.49 Clock Mux 8 Divider Update Status Register (MUX_8_DIV_UPD_STAT)

#### Offset

| Register | Offset |
|---|---|
| MUX_8_DIV_UPD_STAT | 53Ch |

#### Function

This register provides the update status of the clock dividers corresponding to clock mux 8. When a write operation on any divider control register is performed, the divider status bit in this register is set to logic-1. The bit is set to logic-0 when the divider has sampled the new divider configuration. Performing multiple writes without tracking the status bit on same or other clock dividers inside the same clock mux leads to inconsistent reporting, that is, the divider status maybe be set to logic-0 when the corresponding divider update is pending.

**NOTE**

Read accesses to MUX_n_DIV_UPD_STAT always complete without returning bus transfer error independent of whether any divider(s) are implemented inside MC_CGM clock mux.

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-1<br>— | This field is reserved and reads return zeros. |
| 0<br>DIV_STAT | Divider status for clock mux 8<br>On reading MUX_n_DIV_UPD_STAT after updating a divider control register, if the value of this field is fixed to 1 because of an error in the selected clock source, perform the following steps to switch the mux to a new clock source:<br>1. Switch the mux to a working clock source without polling this field.<br>2. Update MUX_n_DC_m and poll this field. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>This field clears once divider configuration is updated or on destructive reset. If functional reset comes when this field is 1 then it can remain fixed to 1 until divider input clock is restored.<br>0b - No divider configuration update is pending.<br>1b - Divider configuration update on at least one divider associated with this multiplexer is pending. |

## 24.5.50 Clock Mux 9 Select Control Register (MUX_9_CSC)

### Offset

| Register | Offset |
|---|---|
| MUX_9_CSC | 540h |

### Function

This register provides the clock source selection control for clock mux 9. Clock mux 9 implements hardware control clock switching ensuring that the clock switch happens in a graceful manner (without glitches). See the "Hardware-controlled clock multiplexer" section for details.

This register is reset on destructive reset only.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-29 | This field is reserved and reads return zeros. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 28-24<br>SELCTL | Clock source selection control<br>Selects the source clock for clock mux 9. The reserved values are not displayed.<br>0_0000b - FIRC<br>0_0010b - FXOSC<br>0_1000b - PLL_PHI0_CLK<br>1_1000b - EMAC_RMII_TX_CLK<br>1_1001b - EMAC_RX_CLK |
| 23-4<br>— | This field is reserved and reads return zeros. |
| 3<br>SAFE_SW | Safe clock request<br>Writing 1 to this bit makes a safe clock switch request to FIRC. After a safe clock switch operation is requested, this bit is auto cleared and a corresponding bit in the status register is set. |
| 2<br>CLK_SW | Clock switch<br>Writing 1 to this bit makes a clock switch request to clock mux 9. After a clock switch operation is requested, this bit is auto cleared and a corresponding bit in the status register is set. |
| 1-0<br>— | This field is reserved and reads return zeros. |

### 24.5.51 Clock Mux 9 Select Status Register (MUX_9_CSS)

#### Offset

| Register | Offset |
|---|---|
| MUX_9_CSS | 544h |

#### Function

This register provides the current clock source selection status for clock mux 9.

This register is reset on destructive reset only.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | SELSTAT | | | | 0 | | | | SWTRG | | | SWIP |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31-29<br>— | This field is reserved and reads return zeros. |
| 28-24<br>SELSTAT | Clock source selection status<br>This value indicates the current source selected for clock mux 9. The reserved values are not displayed.<br>0_0000b - FIRC<br>0_0010b - FXOSC<br>0_1000b - PLL_PHI0_CLK<br>1_1000b - EMAC_RMII_TX_CLK<br>1_1001b - EMAC_RX_CLK |
| 23-20<br>— | This field is reserved and reads return zeros. |
| 19-17<br>SWTRG | Switch trigger cause<br>This value indicates the cause for the latest clock source switch.<br>**NOTE**<br>If the clock fails, followed by multiple safe clock switch requests for MC_CGM hardware clock mux, the value of the SWTRG field can be either 4 or 5.<br>000b - Reserved<br>001b - Switch after request succeeded.<br>010b - Switch after the request failed because of an inactive target clock and the current clock is FIRC.<br>011b - Switch after the request failed because of an inactive current clock and the current clock is FIRC. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 100b - Switch to FIRC because of a safe clock request or reset succeeded.<br>101b - Switch to FIRC because of a safe clock request or reset succeeded, but the previous current clock source was inactive.<br>110b - Reserved<br>111b - Reserved |
| 16<br>SWIP | Switch in progress<br>**NOTE**<br>New clock switch request can only be given three clock cycles after the completion of the previous request.<br>0b - Clock source switching is complete.<br>1b - Clock source switching is in progress. |
| 15-4<br>— | This field is reserved and reads return zeros. |
| 3<br>SAFE_SW | Safe clock request<br>This field provides an indication of whether a switch to safe clock operation was requested during the previous/ongoing request on clock mux 9.<br>0b - No safe clock switch operation was requested.<br>1b - Safe clock switch operation was requested. |
| 2<br>CLK_SW | Clock switch<br>This field provides an indication of whether a clock switch operation was requested during the previous/ongoing request on clock mux 9.<br>0b - No clock switch operation was requested.<br>1b - Clock switch operation was requested. |
| 1-0<br>— | This field is reserved and reads return zeros. |

## 24.5.52 Clock Mux 9 Divider 0 Control Register (MUX_9_DC_0)

### Offset

| Register | Offset |
|---|---|
| MUX_9_DC_0 | 548h |

### Function

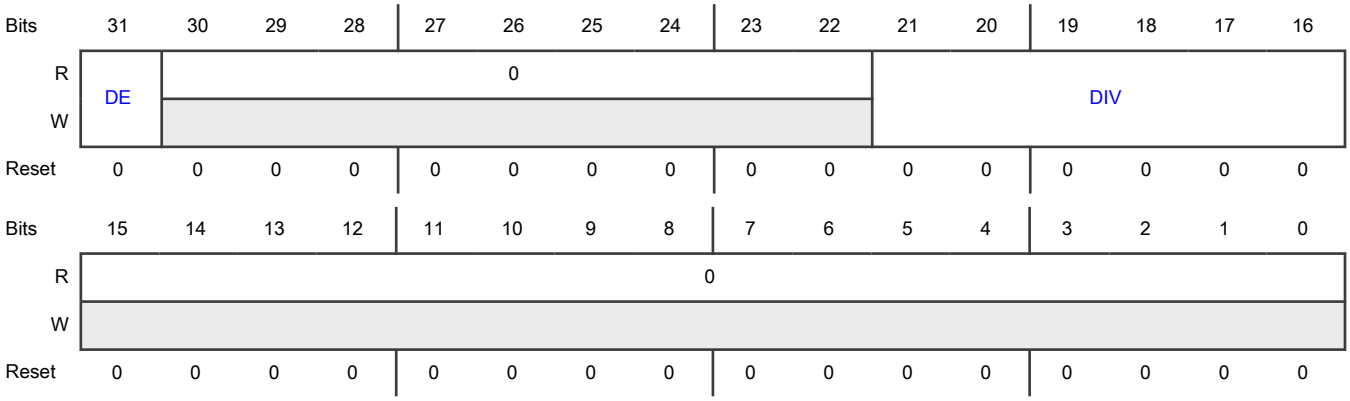
This register controls the clock divider 0 for clock mux 9.

This divider is a 50% duty cycle divider.

**NOTE**

The update to the fields of this register should be an atomic write, that is, one single write should update the complete register.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>DE | Divider enable<br>0b - Divider is disabled.<br>1b - Divider is enabled. |
| 30-22<br>— | This field is reserved and reads return zeros. |
| 21-16<br>DIV | Division value<br>This field provides the division value for the clock divider. The clock period of the clock after division is 'DIV+1' times the time period of the current input clock to the divider. |
| 15-0<br>— | This field is reserved and reads return zeros. |

## 24.5.53 Clock Mux 9 Divider Update Status Register (MUX_9_DIV_UPD_STAT)

### Offset

| Register | Offset |
|---|---|
| MUX_9_DIV_UPD_STAT | 57Ch |

### Function

This register provides the update status of the clock dividers corresponding to clock mux 9. When a write operation on any divider control register is performed, the divider status bit in this register is set to logic-1. The bit is set to logic-0 when the

divider has sampled the new divider configuration. Performing multiple writes without tracking the status bit on same or other clock dividers inside the same clock mux leads to inconsistent reporting, that is, the divider status maybe be set to logic-0 when the corresponding divider update is pending.

**NOTE**

Read accesses to MUX_n_DIV_UPD_STAT always complete without returning bus transfer error independent of whether any divider(s) are implemented inside MC_CGM clock mux.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-1<br>— | This field is reserved and reads return zeros. |
| 0<br>DIV_STAT | Divider status for clock mux 9<br>On reading MUX_n_DIV_UPD_STAT after updating a divider control register, if the value of this field is fixed to 1 because of an error in the selected clock source, perform the following steps to switch the mux to a new clock source:<br>1. Switch the mux to a working clock source without polling this field.<br>2. Update MUX_n_DC_m and poll this field.<br>**NOTE**<br>This field clears once divider configuration is updated or on destructive reset. If functional reset comes when this field is 1 then it can remain fixed to 1 until divider input clock is restored.<br>0b - No divider configuration update is pending.<br>1b - Divider configuration update on at least one divider associated with this multiplexer is pending. |

## 24.5.54 Clock Mux 10 Select Control Register (MUX_10_CSC)

### Offset

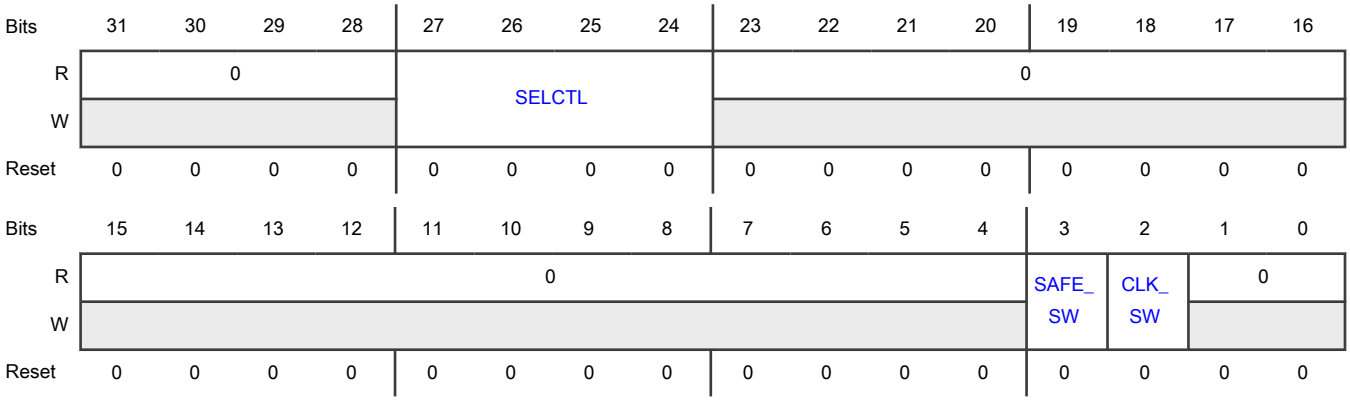
| Register | Offset |
|---|---|
| MUX_10_CSC | 580h |

### Function

This register provides the clock source selection control for clock mux 10. Clock mux 10 implements hardware control clock switching ensuring that the clock switch happens in a graceful manner (without glitches). See the "Hardware-controlled clock multiplexer" section for details.

This register is reset on destructive reset only.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-28<br>— | This field is reserved and reads return zeros. |
| 27-24<br>SELCTL | Clock source selection control<br>Selects the source clock for clock mux 10. The reserved values are not displayed.<br>0000b - FIRC<br>0010b - FXOSC<br>1001b - PLL_PHI1_CLK |
| 23-4<br>— | This field is reserved and reads return zeros. |
| 3<br>SAFE_SW | Safe clock request |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Writing 1 to this bit makes a safe clock switch request to FIRC. After a safe clock switch operation is requested, this bit is auto cleared and a corresponding bit in the status register is set. |
| 2<br>CLK_SW | Clock switch<br>Writing 1 to this bit makes a clock switch request to clock mux 10. After a clock switch operation is requested, this bit is auto cleared and a corresponding bit in the status register is set. |
| 1-0<br>— | This field is reserved and reads return zeros. |

## 24.5.55 Clock Mux 10 Select Status Register (MUX_10_CSS)

### Offset

| Register | Offset |
|---|---|
| MUX_10_CSS | 584h |

### Function

This register provides the current clock source selection status for clock mux 10.

This register is reset on destructive reset only.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | SELSTAT | | | | 0 | | | | SWTRG | | | SWIP |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31-28<br>— | This field is reserved and reads return zeros. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 27-24<br>SELSTAT | Clock source selection status<br>This value indicates the current source selected for clock mux 10. The reserved values are not displayed.<br>0000b - FIRC<br>0010b - FXOSC<br>1001b - PLL_PHI1_CLK |
| 23-20<br>— | This field is reserved and reads return zeros. |
| 19-17<br>SWTRG | Switch trigger cause<br>This value indicates the cause for the latest clock source switch.<br>**NOTE**<br>If the clock fails, followed by multiple safe clock switch requests for MC_CGM hardware clock mux, the value of the SWTRG field can be either 4 or 5.<br>000b - Reserved<br>001b - Switch after request succeeded.<br>010b - Switch after the request failed because of an inactive target clock and the current clock is FIRC.<br>011b - Switch after the request failed because of an inactive current clock and the current clock is FIRC.<br>100b - Switch to FIRC because of a safe clock request or reset succeeded.<br>101b - Switch to FIRC because of a safe clock request or reset succeeded, but the previous current clock source was inactive.<br>110b - Reserved<br>111b - Reserved |
| 16<br>SWIP | Switch in progress<br>**NOTE**<br>New clock switch request can only be given three clock cycles after the completion of the previous request.<br>0b - Clock source switching is complete.<br>1b - Clock source switching is in progress. |
| 15-4<br>— | This field is reserved and reads return zeros. |
| 3 | Safe clock request |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| SAFE_SW | This field provides an indication of whether a switch to safe clock operation was requested during the previous/ongoing request on clock mux 10.<br>0b - No safe clock switch operation was requested.<br>1b - Safe clock switch operation was requested. |
| 2<br>CLK_SW | Clock switch<br>This field provides an indication of whether a clock switch operation was requested during the previous/ ongoing request on clock mux 10.<br>0b - No clock switch operation was requested.<br>1b - Clock switch operation was requested. |
| 1-0<br>— | This field is reserved and reads return zeros. |

## 24.5.56 Clock Mux 10 Divider 0 Control Register (MUX_10_DC_0)

### Offset

| Register | Offset |
|---|---|
| MUX_10_DC_0 | 588h |

### Function

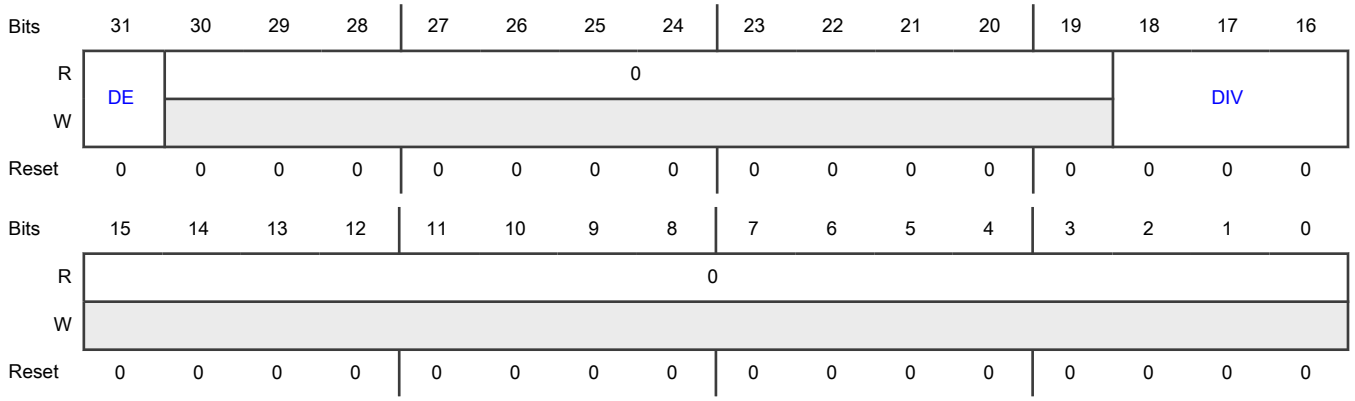
This register controls the clock divider 0 for clock mux 10.

This divider is a 50% duty cycle divider.

**NOTE**

The update to the fields of this register should be an atomic write, that is, one single write should update the complete register.

### Diagram



**Fields**

| Field | Function |
|---|---|
| 31<br>DE | Divider enable<br>0b - Divider is disabled.<br>1b - Divider is enabled. |
| 30-19<br>— | This field is reserved and reads return zeros. |
| 18-16<br>DIV | Division value<br>This field provides the division value for the clock divider. The clock period of the clock after division is 'DIV+1' times the time period of the current input clock to the divider. |
| 15-0<br>— | This field is reserved and reads return zeros. |

### 24.5.57 Clock Mux 10 Divider Update Status Register (MUX_10_DIV_UPD_STAT)

**Offset**

| Register | Offset |
|---|---|
| MUX_10_DIV_UPD_STAT | 5BCh |

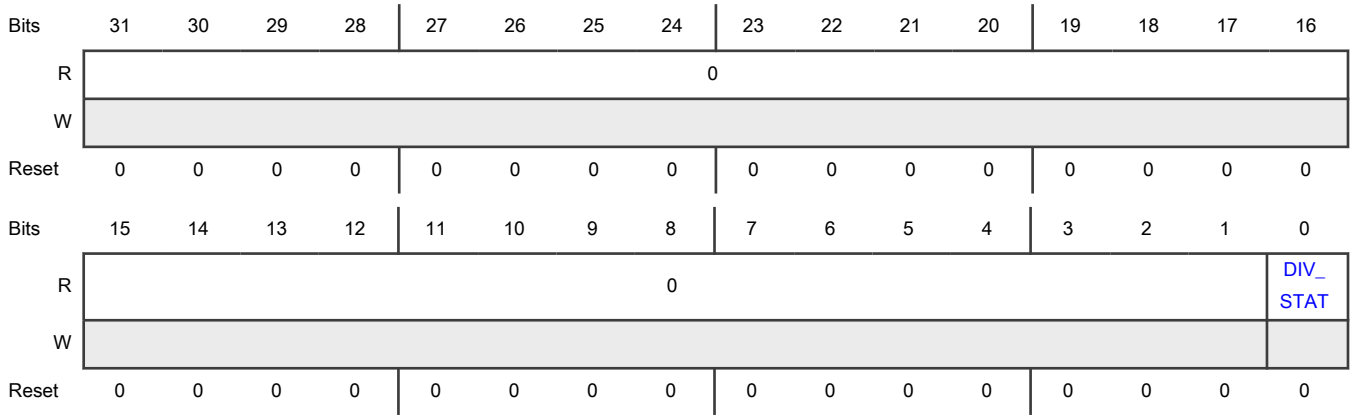
**Function**

This register provides the update status of the clock dividers corresponding to clock mux 10. When a write operation on any divider control register is performed, the divider status bit in this register is set to logic-1. The bit is set to logic-0 when the divider has sampled the new divider configuration. Performing multiple writes without tracking the status bit on same or other clock dividers inside the same clock mux leads to inconsistent reporting, that is, the divider status maybe be set to logic-0 when the corresponding divider update is pending.

**NOTE**

Read accesses to MUX_n_DIV_UPD_STAT always complete without returning bus transfer error independent of whether any divider(s) are implemented inside MC_CGM clock mux.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | This field is reserved and reads return zeros. |
| 0<br>DIV_STAT | Divider status for clock mux 10<br>On reading MUX_n_DIV_UPD_STAT after updating a divider control register, if the value of this field is fixed to 1 because of an error in the selected clock source, perform the following steps to switch the mux to a new clock source:<br>1. Switch the mux to a working clock source without polling this field.<br>2. Update MUX_n_DC_m and poll this field.<br>**NOTE**<br>This field clears once divider configuration is updated or on destructive reset. If functional reset comes when this field is 1 then it can remain fixed to 1 until divider input clock is restored.<br>0b - No divider configuration update is pending.<br>1b - Divider configuration update on at least one divider associated with this multiplexer is pending. |

## 24.5.58 Clock Mux 11 Select Control Register (MUX_11_CSC)

### Offset

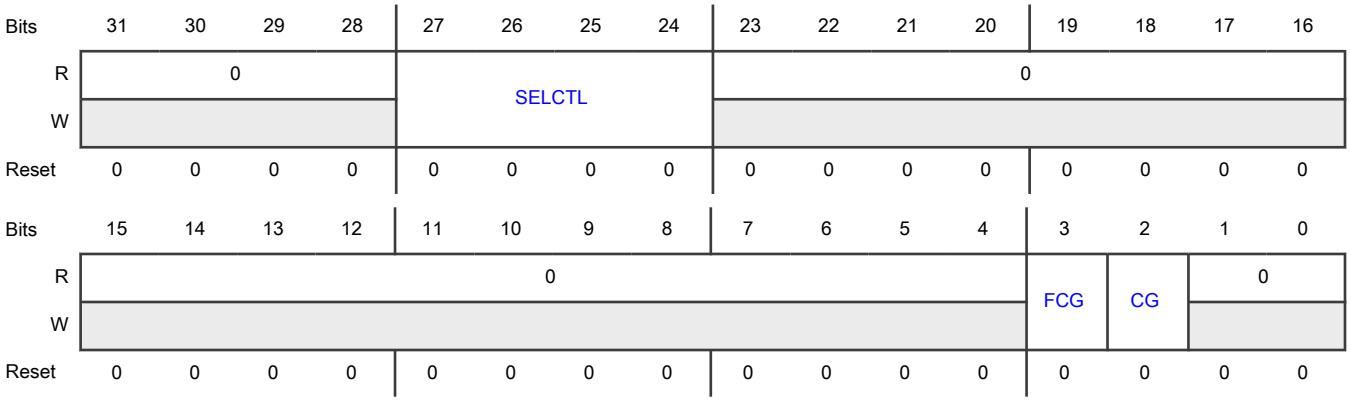
| Register | Offset |
|---|---|
| MUX_11_CSC | 5C0h |

### Function

This register provides the clock source selection control of clock mux 11. Clock mux 11 implements software control clock switching, and a graceful clock switch can be performed by executing a sequence of steps in software. See "Software-controlled clock multiplexer" section for details.

This register is reset on destructive reset only.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-28<br>— | This field is reserved and reads return zeros. |
| 27-24<br>SELCTL | Clock source selection control<br>Selects the source clock for clock mux 11. The reserved values are not displayed.<br>0000b - FIRC<br>0010b - FXOSC<br>1000b - PLL_PHI0_CLK<br>1001b - PLL_PHI1_CLK |
| 23-4<br>— | This field is reserved and reads return zeros. |
| 3<br>FCG | Force clock gate<br>Writing 1 to this bit gates the clock at the output of clock mux 11 to logic-0 irrespective of the logic level of the currently selected clock. Clock gating using this bit should only be performed when it is insured that current clock source is inactive. |
| 2<br>CG | Clock gate<br>Writing 1 to this bit gates the clock at the output of clock mux 11 to logic-0. Using this bit it is insured that no glitches are resulted when gating the clock. |
| 1-0<br>— | This field is reserved and reads return zeros. |

## 24.5.59 Clock Mux 11 Select Status Register (MUX_11_CSS)

### Offset

| Register | Offset |
|---|---|
| MUX_11_CSS | 5C4h |

### Function

This register provides the current clock source selection status for clock mux 11.

This register is reset on destructive reset only.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | SELSTAT | | | | 0 | | | | 0 | | CS | GRIP |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31-28<br>— | This field is reserved and reads return zeros. |
| 27-24<br>SELSTAT | Clock source selection status<br>This value indicates the current source selected for clock mux 11. The reserved values are not displayed.<br>0000b - FIRC<br>0010b - FXOSC<br>1000b - PLL_PHI0_CLK<br>1001b - PLL_PHI1_CLK |
| 23-20<br>— | This field is reserved and reads return zeros. |
| 19-18<br>— | This field is reserved and reads return zeros. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 17<br>CS | Clock status<br>This field indicates state of the clock at the output of the clock mux.<br>0b - Clock is gated to logic-0 at output of clock mux<br>1b - Clock mux is transparent. Active clock pulses at input of clock mux results in same number of pulses at its output |
| 16<br>GRIP | Gating request is in progress.<br>When a clock gate request is given this bit indicates if the clock gating at the output of mux has completed or not.<br>0b - Clock source gating or ungating has completed.<br>1b - Clock source gating or ungating is in progress. |
| 15-0<br>— | This field is reserved and reads return zeros. |

## 24.5.60 Clock Mux 11 Divider 0 Control Register (MUX_11_DC_0)

### Offset

| Register | Offset |
|---|---|
| MUX_11_DC_0 | 5C8h |

### Function

This register controls the clock divider 0 for clock mux 11.

This divider is a 50% duty cycle divider.

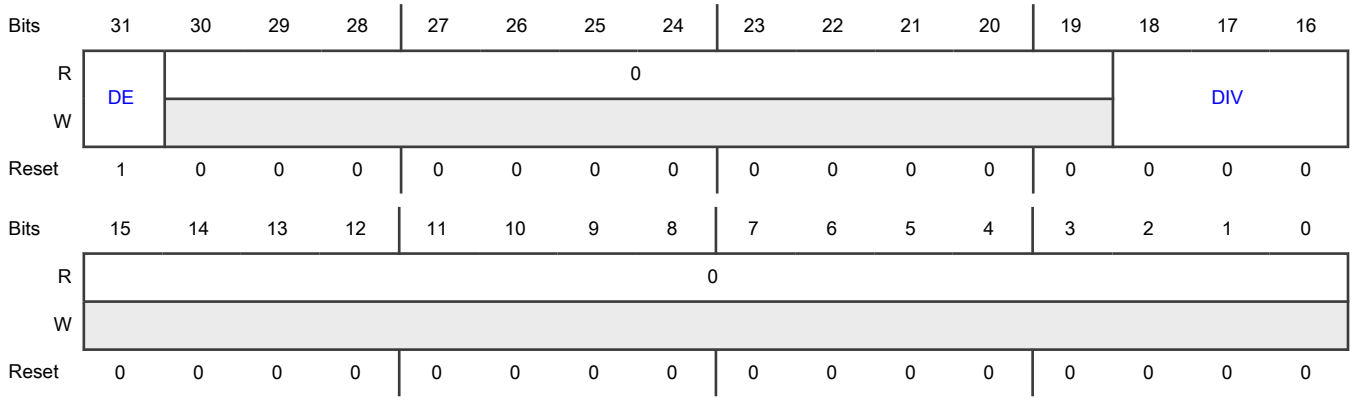
**NOTE**

The update to the fields of this register should be an atomic write, that is, one single write should update the complete register.

**NOTE**

Software-controlled clock multiplexer dividers are not expected to return to the default state on the hardware transitions and handshakes occurring as part of the functional reset entry sequence (only hardware-controlled clock multiplexer dividers return to the default state).

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>DE | Divider enable<br>0b - Divider is disabled.<br>1b - Divider is enabled. |
| 30-19<br>— | This field is reserved and reads return zeros. |
| 18-16<br>DIV | Division value<br>This field provides the division value for the clock divider. The clock period of the clock after division is 'DIV+1' times the time period of the current input clock to the divider. |
| 15-0<br>— | This field is reserved and reads return zeros. |

## 24.5.61 Clock Mux 11 Divider Update Status Register (MUX_11_DIV_UPD_STAT)

### Offset

| Register | Offset |
|---|---|
| MUX_11_DIV_UPD_STAT | 5FCh |

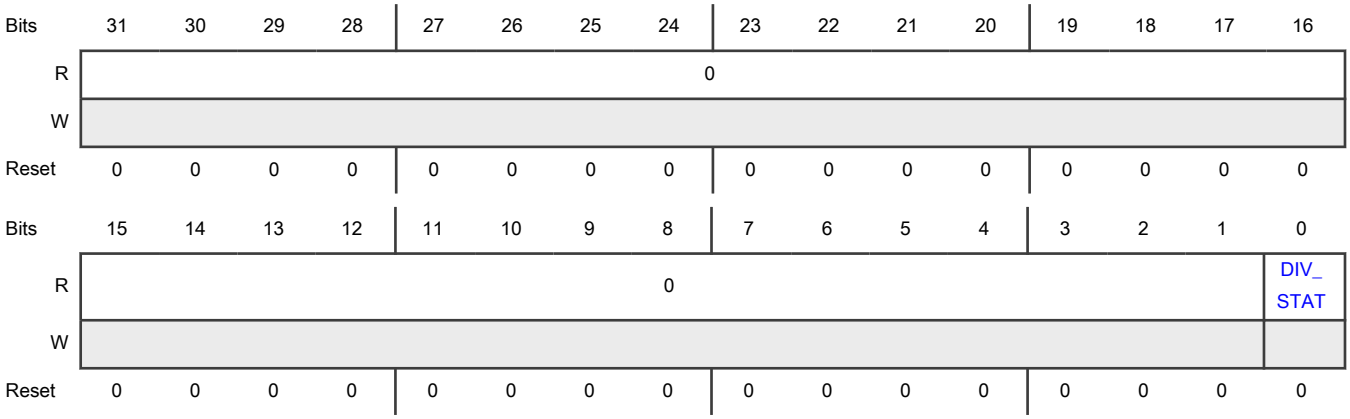
### Function

This register provides the update status of the clock dividers corresponding to clock mux 11. When a write operation on any divider control register is performed, the divider status bit in this register is set to logic-1. The bit is set to logic-0 when the divider has sampled the new divider configuration. Performing multiple writes without tracking the status bit on same or other clock dividers inside the same clock mux leads to inconsistent reporting, that is, the divider status maybe be set to logic-0 when the corresponding divider update is pending.

**NOTE**

Read accesses to MUX_n_DIV_UPD_STAT always complete without returning bus transfer error independent of whether any divider(s) are implemented inside MC_CGM clock mux.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | This field is reserved and reads return zeros. |
| 0<br>DIV_STAT | Divider status for clock mux 11<br>On reading MUX_n_DIV_UPD_STAT after updating a divider control register, if the value of this field is fixed to 1 because of an error in the selected clock source, perform the following steps to switch the mux to a new clock source:<br>1. Switch the mux to a working clock source without polling this field.<br>2. Update MUX_n_DC_m and poll this field.<br>**NOTE**<br>This field clears once divider configuration is updated or on destructive reset. If functional reset comes when this field is 1 then it can remain fixed to 1 until divider input clock is restored.<br>0b - No divider configuration update is pending.<br>1b - Divider configuration update on at least one divider associated with this multiplexer is pending. |

## 24.6 Glossary

| | |
|---|---|
| **PCFS** | Progressive clock frequency switching |
| **LCM** | Least common multiple |

# Chapter 25
# Fast Internal RC Oscillator (FIRC)

## 25.1 Overview

The FIRC digital interface controls the internal 48 MHz RC oscillator system.

### 25.1.1 Features

FIRC can be disabled in Standby mode via software.

- Status register provides the current operating state:
  - On and stable
  - Off or on but not stable

## 25.2 External signals

This module has no external signals.

## 25.3 Initialization

This module does not require initialization.

## 25.4 FIRC register descriptions

### 25.4.1 FIRC memory map

FIRC base address: 402D_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 4h | Status Register (Status_Register) | 32 | R | 0000_0001h |
| 8h | Standby Enable Register (STDBY_ENABLE) | 32 | RW | 0000_0000h |

### 25.4.2 Status Register (Status_Register)

**Offset**

| Register | Offset |
|---|---|
| Status_Register | 4h |

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>STATUS | Status bit for FIRC<br>0b - FIRC is off or unstable.<br>1b - FIRC is on and stable. |

## 25.4.3 Standby Enable Register (STDBY_ENABLE)

### Offset

| Register | Offset |
|---|---|
| STDBY_ENABLE | 8h |

### Function

This register enables or disables FIRC in chip’s Standby mode.

### Diagram

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

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

**Fields**

| Field | Function |
|---|---|
| 31-1<br>— | RESERVED |
| 0<br>STDBY_EN | Enables or disables FIRC in chip's Standby mode.<br>0b - Disabled<br>1b - Enabled |

## 25.5 Glossary

| | |
|---|---|
| **Standby mode** | Power saving mode of the chip |

# Chapter 26
# Slow Internal RC Oscillator (SIRC)

## 26.1 Overview

The SIRC digital interface controls the slow internal on-chip 32 KHz RC oscillator system.

### 26.1.1 Features

The SIRC module:

- Status register provides the current operating state:
  - On and stable
  - Off or on but not stable
- Operates at a frequency of 32 kHz in Functional mode

## 26.2 Operating mode

Only a POR reset will initialize the SIRC. Destructive or Functional resets do not impact the SIRC functionality.

SIRC stabilization occurs after 96 SIRC_CLK cycles.

The SIRC output clock remains invalid until the analog SIRC stabilizes. The output clock does not glitch or overshoot its frequency during enabling or disabling. Also, the clock does not get stuck or produce glitches on a very short hardware disable pulse.

## 26.3 External signals

This module has no external signals.

## 26.4 Initialization

This module does not require initialization.

## 26.5 SIRC register descriptions

### 26.5.1 SIRC memory map

Access to registers use 8-bit, 16-bit, or 32-bit addressing.

SIRC base address: 402C_8000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 4h | Status Register (SR) | 32 | R | 0000_0001h |
| Ch | Miscellaneous input (MISCELLANEOUS_IN) | 32 | RW | 0000_0000h |

## 26.5.2 Status Register (SR)

### Offset

| Register | Offset |
|---|---|
| SR | 4h |

### Diagram

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

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

### Fields

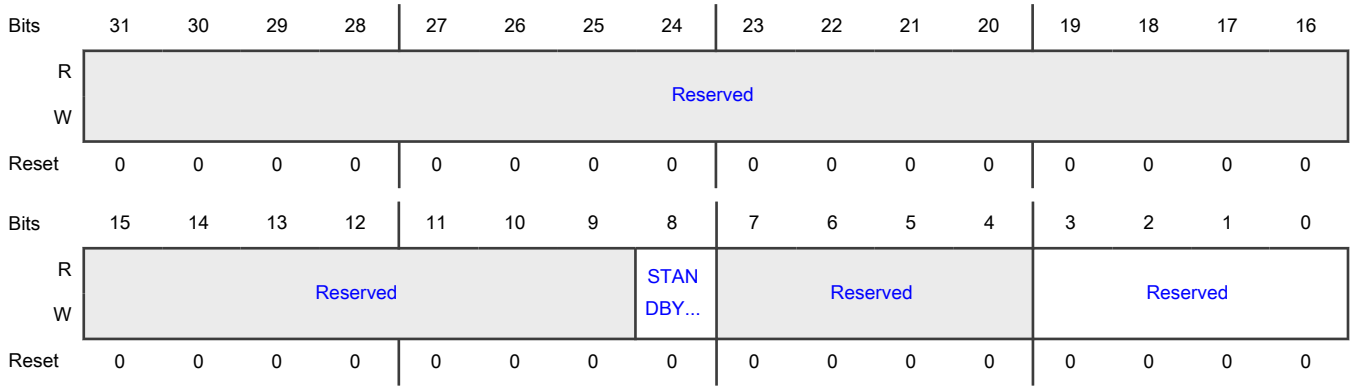
| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>STATUS | Status bit for SIRC<br>0b - SIRC is off or unstable<br>1b - SIRC is on and stable |

## 26.5.3 Miscellaneous input (MISCELLANEOUS_IN)

### Offset

| Register | Offset |
|---|---|
| MISCELLANEOUS_IN | Ch |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>STANDBY_ENABLE | Standby Enable for SIRC<br>0b - SIRC disables in Standby mode<br>1b - SIRC enables in Standby mode |
| 7-4<br>— | Reserved |
| 3-0<br>— | Reserved |

# Chapter 27
# Fast Crystal Oscillator Digital Controller (FXOSC)

## 27.1 Chip-specific FXOSC information

### 27.1.1 Chip-specific FXOSC information

**For MCXE31B:** For bypass mode applications, the EXTAL pin should be driven low when FXOSC is in off/disabled state.

- While initializing FXOSC: When the FXOSC is used in Bypass mode, the external clock source can only be enabled after the FXOSC is enabled.
- While disabling FXOSC: When the FXOSC is used in Bypass mode, the external clock source must already be inactive before disabling the FXOSC.

## 27.2 Overview

The Fast Crystal Oscillator (FXOSC) generates a clock which can be used at the SoC level. The FXOSC has a digital interface to control and configure the oscillator. When FXOSC is powered down at any time, it is designed not to generate any glitch at the output clock. A counter inside FXOSC handles different stabilization times. CG cell is clock gating cell, it gates the clock till stabilization time.

### 27.2.1 Block diagram

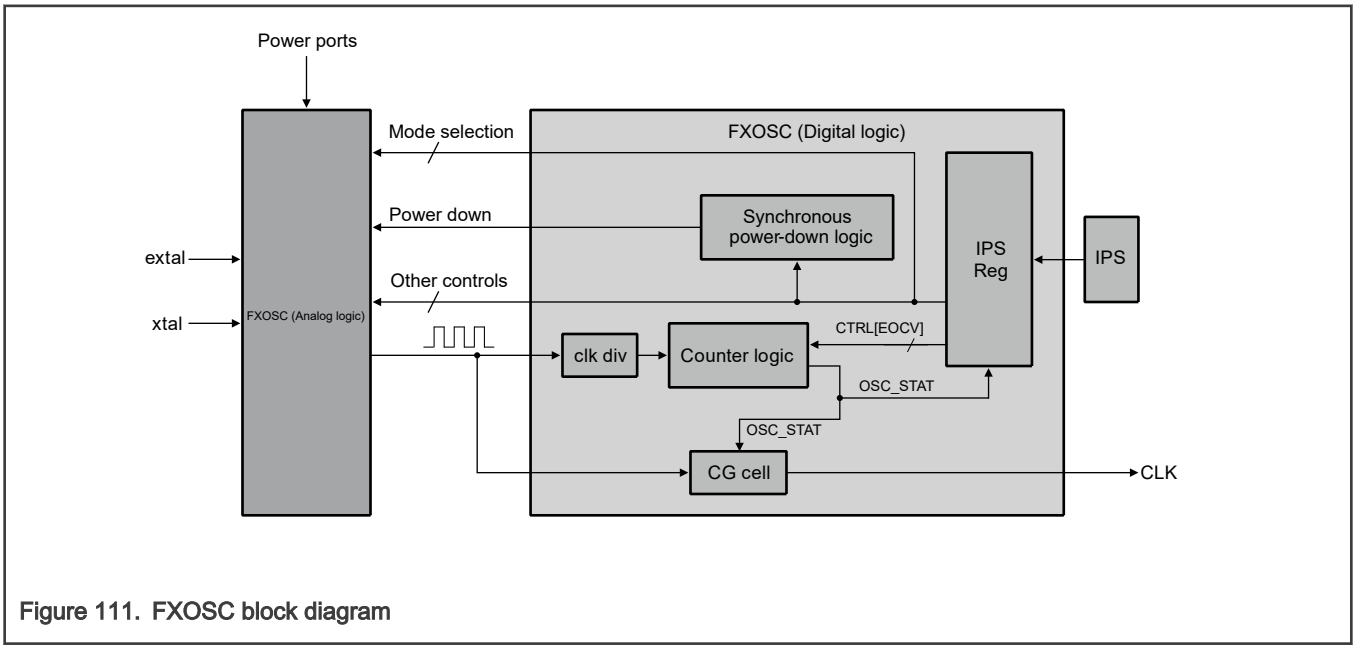



Figure 111. FXOSC block diagram

### 27.2.2 Features

FXOSC features are as follows:

- Status register shows current module state.
- Control register can:
  - Select a mode of operation:
    - Crystal mode
    - Single-Input Bypass mode using EXTAL clock input

— Disable the module (Power-Down mode)

## 27.3 Functional description

The table below shows configuration settings for the different FXOSC modes set by the CTRL register:

**Table 144. FXOSC operation mode settings**

| Mode | CTRL[OSCON] | CTRL[OSC_BYP] | CTRL[COMP_EN] | FXOSC_CLK |
|---|---|---|---|---|
| Power-Down mode | 0 | X | X | 0 |
| Crystal mode | 1 | 0 | 1 | Crystal clock |
| Single-Input Bypass mode | 1 | 1 | 0 | EXTAL |

Power-Down mode is the FXOSC default condition after any reset: POR, Destructive, or Functional.

### 27.3.1 Clock generation in Crystal mode

FXOSC_CLK start when counter value reaches FXOSC_CTRL[EOCV] × 128.

### 27.3.2 Clock generation in Single-Input Bypass mode

This mode bypasses the oscillator and uses a single-input external clock (EXTAL input) for FXOSC_CLK. When FXOSC enters Bypass mode:

- Enable the external clock after enabling FXOSC during FXOSC initialization.
- Disable the external clock beforehand when disabling FXOSC.

### 27.3.3 Clocking

This module has no clocking considerations.

### 27.3.4 Interrupts

This module has no interrupts.

## 27.4 External signals

This module has no external signals.

## 27.5 Initialization

#### Initializing FXOSC

Initialize FXOSC as follows:

1. Write the desired value to CTRL[OSC_BYP] and CTRL[COMP_EN] to select an operation mode as shown in Table 144.

   **NOTE**

   FXOSC must be disabled when the operation mode is modified.

2. Configure CTRL[GM_SEL].
   - In Crystal mode configure the transconductance based on the module specification in the chip data sheet.
   - In Single-Input Bypass mode write 0000b to this field.

**NOTE**

In Crystal mode FXOSC will not function with zero transconductance (GM_SEL = 0000b).

3. Set CTRL[EOCV] calculating the value as follows:
   - EOCV (in decimal) = (stabilization time in ns) ÷ (4×128×(period of clock in ns))
4. Write 1 to CRTL[OSCON] to enable FXOSC.

**NOTE**

When the FXOSC is used in Bypass mode, the clock from the external source can only be used after the FXOSC is enabled

5. Confirm the clock is stable (STAT[OSC_STAT] = 1) before using it.

**NOTE**

See Hardware design guide for further details and the recommended circuit for each mode.

#### Disabling FXOSC

Write 0 to CRTL[OSCON] to disable FXOSC when FXOSC_CLK is running and stable.

FXOSC enters Power-down mode after at least four crystal clocks. No glitches occur during the transition to Power-Down mode because synchronizers are used.

**NOTE**

After disabling FXOSC:

- Wait for at least 2µs before enabling FXOSC again.
- You must not change other values in FXOSC registers for at least 16 FXOSC_CLK cycles.

## 27.6 FXOSC register descriptions

This section provides the descriptions of all registers used for configuring the FXOSC.

### 27.6.1 FXOSC memory map

Use 8-bit, 16-bit, or 32-bit addressing to access registers.

FXOSC base address: 402D_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | FXOSC Control Register (CTRL) | 32 | RW | 019D_00C0h |
| 4h | Oscillator Status Register (STAT) | 32 | R | 0000_0000h |

### 27.6.2 FXOSC Control Register (CTRL)

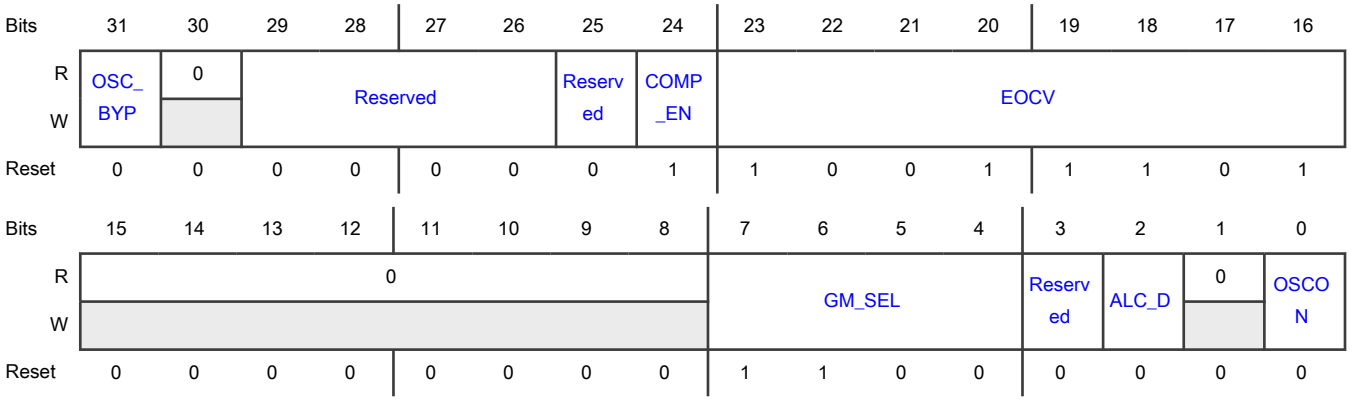
#### Offset

| Register | Offset |
|---|---|
| CTRL | 0h |

### Function

Configures FXOSC operation.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>OSC_BYP | Oscillator bypass<br>Bypasses the internal oscillator.<br>0b - Internal oscillator not bypassed<br>1b - Internal oscillator bypassed |
| 30<br>— | Reserved |
| 29-26<br>— | Reserved |
| 25<br>— | Reserved |
| 24<br>COMP_EN | Comparator enable<br>Enables or disables the comparator.<br>• For Crystal mode set this field to 1.<br>• For Single-Input Bypass mode set this field to 0.<br>0b - Comparator disabled<br>1b - Comparator enabled |
| 23-16<br>EOCV | End of count value<br>Specifies the end-of-count. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | The oscillator counter runs on the crystal clock divided by 4 and counts up to EOCV × 128. This counting period ensures that the external oscillator clock signal is stable before the system selects FXOSC as a source.<br>**NOTE**<br>• You must set EOCV to the appropriate value to allow clock and duty cycle to stabilize and guarantee that OSC_STAT becomes set within the crystal startup time.<br>— In Crystal mode, EOCV value must to be calculated to appropriate value based on the crystal specification using the equation in Initializing FXOSC.<br>— In Single-Input Bypass mode, EOCV value is irrelevant. FXOSC holds the counter in reset.<br>• Before modifying EOCV, FXOSC must be disabled. |
| 15-8<br>— | Reserved |
| 7-4<br>GM_SEL | Crystal overdrive protection<br>Selects the transconductance applied by the FXOSC amplifier. This setting depends on crystal specification.<br>**NOTE**<br>• In Crystal mode FXOSC will not function with zero transconductance (GM_SEL = 0000b).<br>• For details on how to set this field, see Initializing FXOSC.<br>0000b - 0x<br>0001b - 0.1004x<br>0010b - 0.2009x<br>0011b - 0.3013x<br>0100b - 0.2343x<br>0101b - 0.3348x<br>0110b - 0.4345x<br>0111b - 0.5349x<br>1000b - 0.4679x<br>1001b - 0.5684x<br>1010b - 0.6681x<br>1011b - 0.7678x<br>1100b - 0.7016x<br>1101b - 0.8013x |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1110b - 0.9003x<br>1111b - 1x |
| 3<br>— | Reserved |
| 2<br>ALC_D | Automatic level controller enable<br>The ALC feature of the FXOSC internal circuit automatically adjusts the bias current of the crystal oscillator's amplifier. It must be used in crystal oscillator mode. During startup the EXTAL/XTAL amplitude would be lower, so the ALC circuit dynamically increases the bias current of the internal amplifier to provide higher current and make crystal oscillator startup faster in normal oscillator mode. After EXTAL/XTAL amplitude has increased to a steady state, the ALC circuit automatically reduces the amplifier's bias current to save power.<br>0b - Enables automatic level controller<br>1b - Disables automatic level controller |
| 1<br>— | Reserved |
| 0<br>OSCON | Crystal oscillator power-down control<br>Enables or disables FXOSC<br>0b - Disables FXOSC<br>1b - Enables FXOSC |

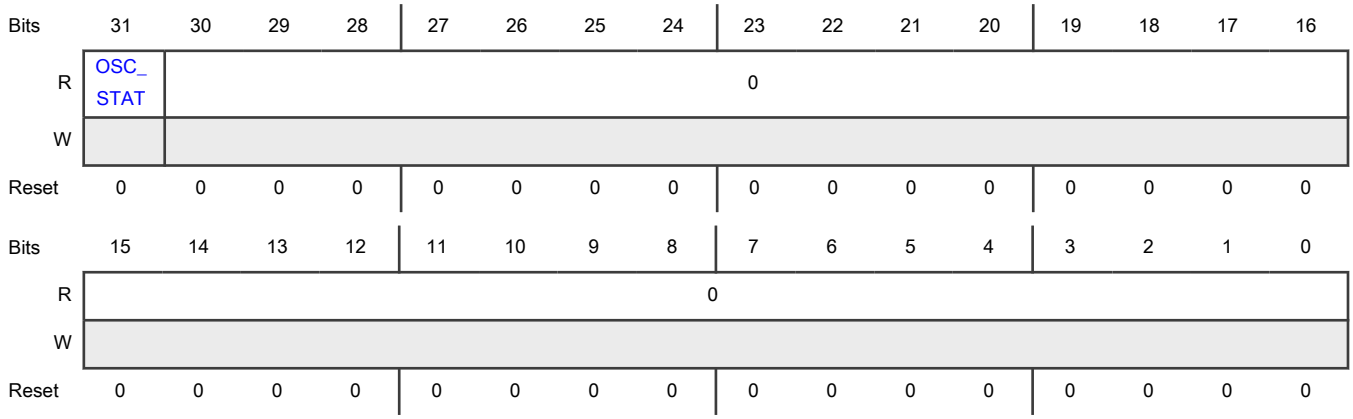
## 27.6.3 Oscillator Status Register (STAT)

**Offset**

| Register | Offset |
|---|---|
| STAT | 4h |

**Function**
Shows current state of FXOSC.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>OSC_STAT | Crystal oscillator status<br>Indicates the crystal oscillator status.<br>**NOTE**<br>OSC_STAT value is not valid if transconductance is set to 0.<br>0b - Crystal oscillator is off or on but not stable.<br>1b - Crystal oscillator is on and providing a stable clock. |
| 30-0<br>— | Reserved |

# Chapter 28
# Slow Crystal Oscillator Digital Controller (SXOSC)

## 28.1 Overview

The Slow crystal oscillator (SXOSC) generates a clock which can be used at the SoC level. The SXOSC has a digital interface to control and configure the oscillator. When SXOSC is powered down at any time, it is designed not to generate any glitch at the output clock. A counter inside SXOSC handles different stabilization times. CG cell is clock gating cell, it gates the clock till stabilization time.

### 28.1.1 Block diagram

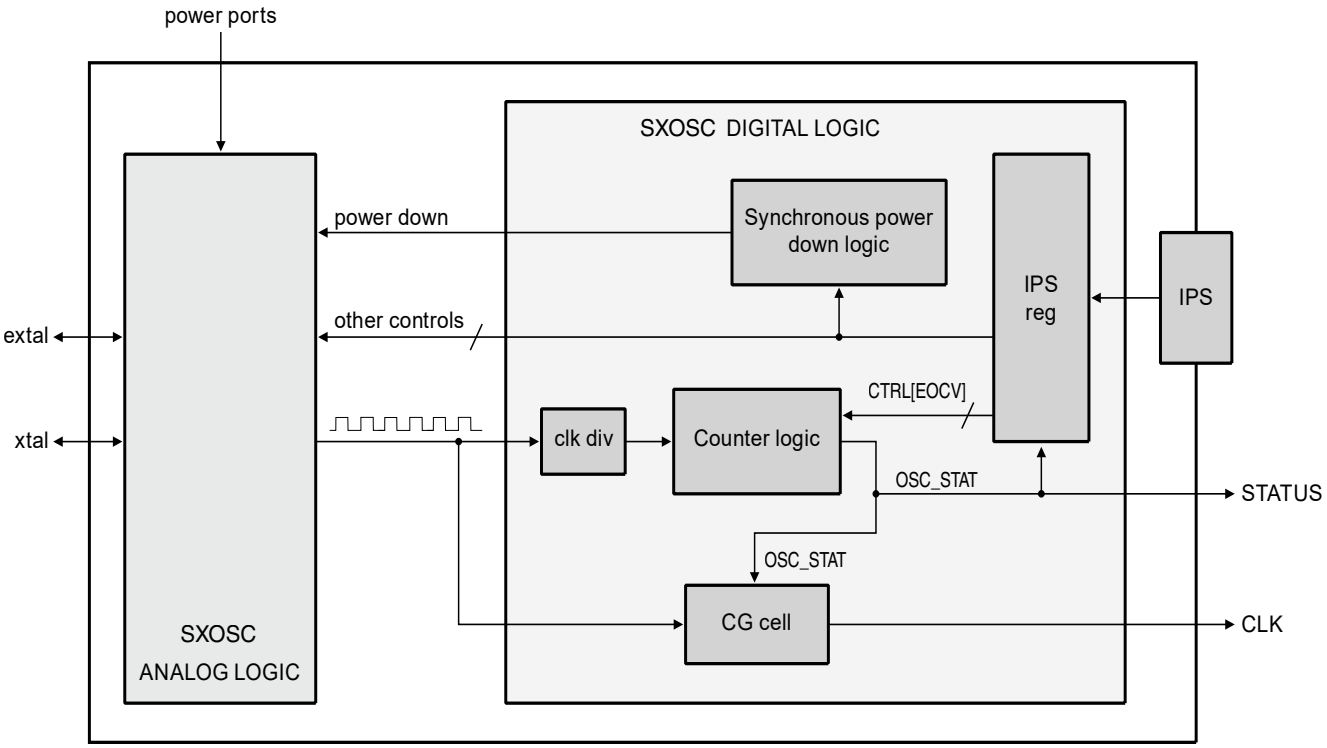



Figure 112. Block diagram

## 28.2 Features

- SXOSC generates a 32 KHz clock output in crystal mode
- SXOSC contains a status register, the value of which becomes 1 when the crystal stabilization time is complete
- SXOSC can be powered down through software bit.

## 28.3 Functional description

SXOSC generates control signals to configure the analog module to operate in specific modes.

The following table shows the mode of operation available for selection and its settings.

Table 145. Operation mode settings

| Mode | Value of SXOSC_CTRL[OSCON] | Output clock |
|---|---|---|
| Functional Oscillator | 0 (oscillator switched off) | 0 (indicates no output) |
| | 1 (oscillator switched on) | Crystal clock |

### 28.3.1 Clock generation in crystal mode

After hard reset, the crystal oscillator is switched off by default. For clock generation in crystal mode, see Table 145. The counter logic starts counting and the stable clock starts running one clock cycle after reaching the value of SXOSC_CTRL[EOCV] x 128 counter value. The module writes 1 to SXOSC_STAT[OSC_STAT] after two module clock cycles.

### 28.3.2 Clock stopping in crystal mode

To stop a stable, running clock, configure the power down mode as specified in Table 145. A glitch does not occur because synchronizers are used.

### 28.3.1 Modes of operation

The SXOSC has following modes of operation.

#### 28.3.1.1 Crystal mode

In this mode crystal is connected between extal and xtal ports, to select crystal mode see Table 145

#### 28.3.1.2 Bypass mode

The bypass mode is handled outside the DA Wrapper.

**NOTE**

In this mode crystal is removed from extal and xtal ports and extal is driven by external clock and xtal is unconnected.

### 28.3.2 Clocking

This module has no clocking considerations.

### 28.3.3 Interrupts

This module has no interrupts.

## 28.4 External signals

This module has no external signals.

## 28.5 Initialization information

To enter into any mode the following sequences must be followed. By default IP is disabled.

- Power-down Mode:
  - When SXOSC is running in any mode, de-assert SXOSC_CTRL[OSCON]
- Crystal Mode:
  1. Disable the IP by de-asserting SXOSC_CTRL[OSCON] bit
  2. Connect the crystal between extal and xtal ports

3. Write an appropriate value to SXOSC_CTRL[EOCV]
4. Program recommended value in SXOSC_CTRL[GM_SEL]
5. Enable the IP by asserting SXOSC_CTRL[OSCON] bit
6. SXOSC_STAT[OSC_STAT] bit will be set after counter runs as per programming of SXOSC_CTRL[EOCV] and clock will be released to SoC

## 28.6 SXOSC register descriptions

This section provides the description of all registers for configuring the SXOSC.

### 28.6.1 SXOSC memory map

Addresses are given as offsets from the module base address. All registers can be accessed using 8-bit, 16-bit or 32-bit addressing.

**NOTE**

Some of the register reset values are specifically configured for each unique device by external configuration signals or parameters.

SXOSC base address: 402C_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Oscillator Control Register (SXOSC_CTRL) | 32 | RW | 007D_0000h |
| 4h | Oscillator Status Register (SXOSC_STAT) | 32 | R | 0000_0000h |

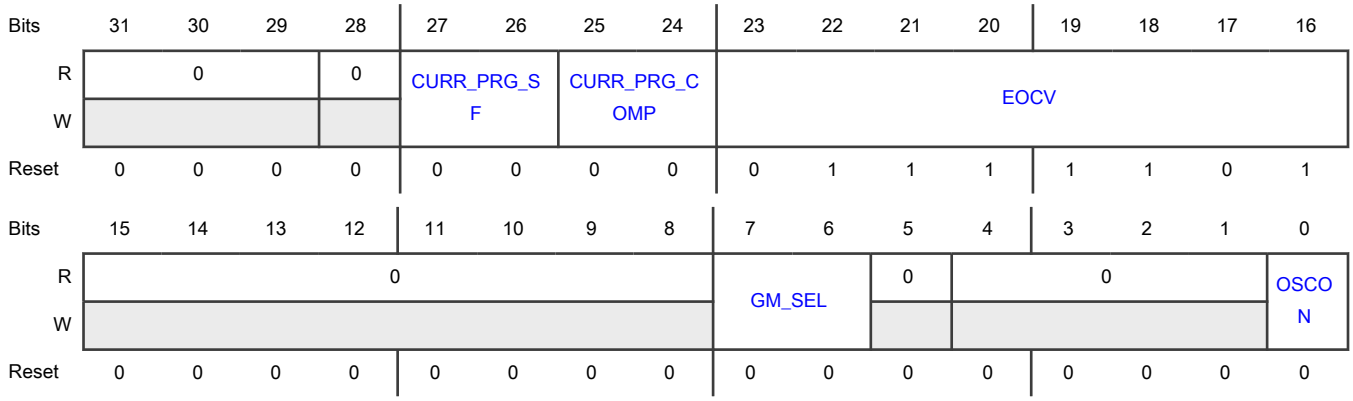
### 28.6.2 Oscillator Control Register (SXOSC_CTRL)

**Offset**

| Register | Offset |
|---|---|
| SXOSC_CTRL | 0h |

**Function**
Oscillator Control Register

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |
| 28<br>— | Reserved |
| 27-26<br>CURR_PRG_SF | These bits specify programmability of level shifter current.<br>00b - 3x<br>01b - 2x<br>10b - 3.5x<br>11b - 4x |
| 25-24<br>CURR_PRG_COMP | These bits specify programmability of comparator current.<br>00b - 1x<br>01b - 2x<br>10b - 3x<br>11b - 4x |
| 23-16<br>EOCV | End of count value<br>These bits specify the end of count value. This value is used by the oscillator Stabilization counter for comparison whenever it is switched On. This counting period ensures that the external oscillator clock signal is stable before it can be selected by the system. Oscillator counter runs on crystal clock divide by 4, and counts value upto EOCV * 128.<br>**NOTE**<br>In order to find the appropriate EOCV value, ensure that the internal counter is running for at least the stabilization time of the crystal as given in the Data Sheet. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>It is recommended to change the value of EOCV only when the IP is in disabled state. |
| 15-8<br>— | Reserved |
| 7-6<br>GM_SEL | Crystal overdrive protection This field setting decides the trans-conductance applied by SXOSC amplifier, and it will depend on crystal specification.<br>00b - 1x<br>01b - 1.25x<br>10b - 1.3x<br>11b - 1.6x |
| 5<br>— | Reserved |
| 4-1<br>— | Reserved |
| 0<br>OSCON | Crystal oscillator power-down control<br>**NOTE**<br>When disabling the IP through software, program 0 to this bit-field, and ensure to not change any other values in the registers for at least 16 SXOSC clock cycles.<br>0b - Crystal oscillator is switched OFF<br>1b - Crystal oscillator is switched ON |

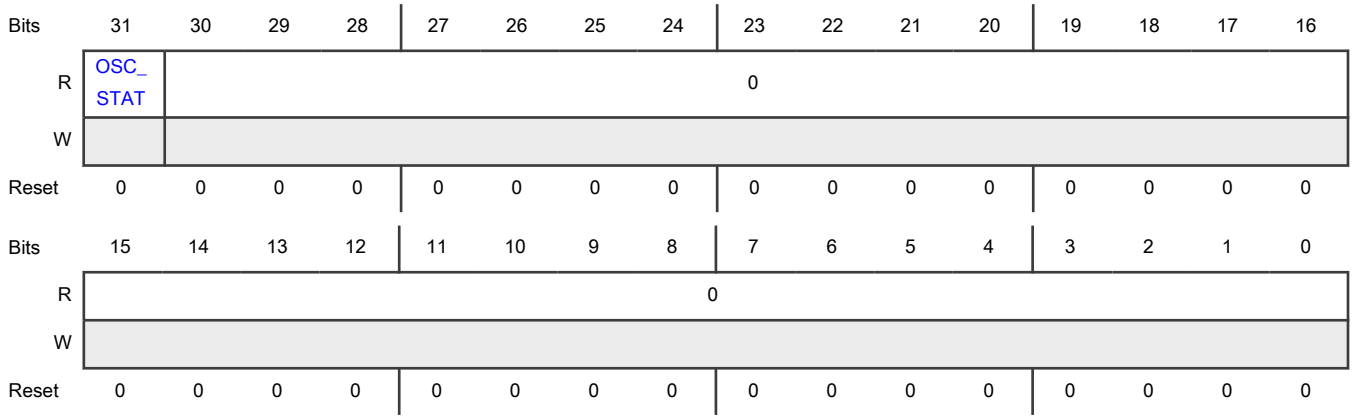
## 28.6.3 Oscillator Status Register (SXOSC_STAT)

**Offset**

| Register | Offset |
|---|---|
| SXOSC_STAT | 4h |

**Function**
Oscillator Status Register

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>OSC_STAT | Crystal oscillator status<br>0b - Crystal oscillator output clock is not stable.<br>1b - Crystal oscillator is providing a stable clock. |
| 30-0<br>— | Reserved |

# Chapter 29
# PLL Digital Interface (PLLDIG)

## 29.1 Chip-specific PLLDIG information

### 29.1.1 PLL-supported accesses and frequencies

The PLLODIV_0 and PLLODIV_1 registers support only word accesses. When you write to these registers, you must retain the default values of the reserved fields.

PLLDIG supports a down-spread modulation of up to 160 MHz PLL PHI clock output only.

### 29.1.2 Register implementation

In MCXE315/MCXE316 there are additional registers as compared to what is mentioned in section 'PLLDIG memory map'. See following table for details.

**Table 146. Register details**

| Register/Bitfield | Offset | Availability |
|---|---|---|
| PLLCLKMUX[REFCLKSEL][1] | 20h | Only available in MCXE315/MCXE316 |

1. See section 'PLLCLKMUX definition' for register definition

#### 29.1.2.1 PLLCLKMUX definition

Register PLLCLKMUX (PLL Clock Multiplexer) is available at offset 20h. This register selects the PLL clock source. Bitfield definition of this register is as shown below:

**Table 147. PLLCLKMUX Definition**

| Bitfield offset | Bitfield name | Bitfield description |
|---|---|---|
| 0 | REFCLKSEL | Reference Clock Select: Selects the PLL clock source.<br>• 0b-FXOSC_CLK<br>• 1b-FIRC_DIV2_CLK |
| 1-31 | Reserved | |

### 29.1.3 Initialization information for MCXE315/MCXE316

Perform the following steps to initialize PLL:

1. Confirm that PLLODIV_*n*[DE] is 0 for all dividers.
2. Confirm that PLLCR[PLLPD] is 1.
3. Program PLLCLKMUX to select the appropriate reference clock.
4. Program the following as needed:
   - PLLDV
   - PLLFD
   - PLLFM to the desired value
5. Program PLLDV[ODIV2] and PLLODIV_*n*[DIV] to the desired values.
6. Wait for the PLL reference clock to be stable.

7. Write 0 to PLLCR[PLLPD].
8. Wait for PLLSR[LOCK] to be 1.
9. Write 1 to PLLODIV_*n*[DE].

Perform the following steps to shut down PLL:

1. Write 0 to PLLODIV_n[DE] for all dividers.
2. Write 1 to PLLCR[PLLPD].

## 29.2 Overview

PLL can multiply or divide the frequency of a given clock input.

### 29.2.1 Block diagram

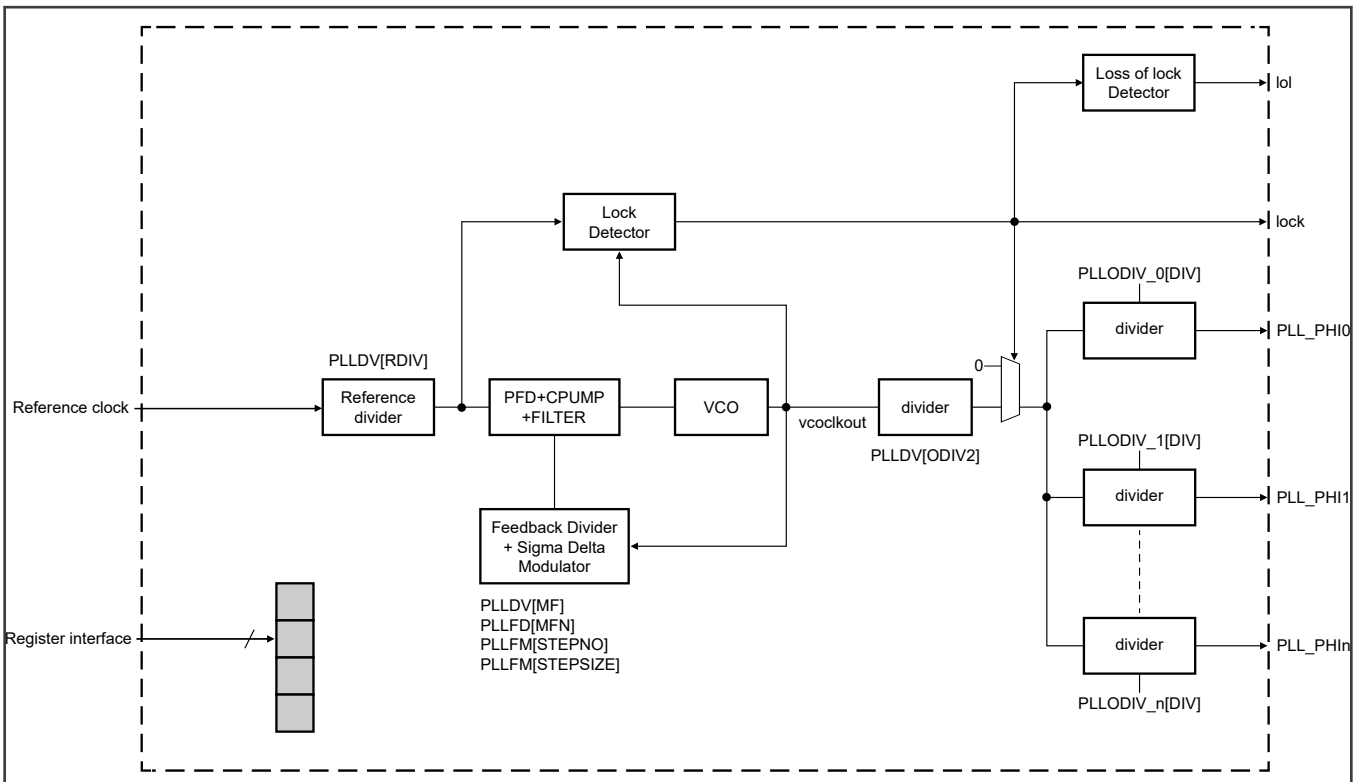



Figure 113. Block diagram

The number of output dividers can vary with the module instance. See the Clocking chapter to confirm the number of PLL output dividers.

### 29.2.2 Features

PLL includes the following features:

- Programmable frequency modulation
- Multiple integer dividers on PLL outputs
- Lock detection circuitry reports when PLL achieves frequency lock
- Continuous monitoring of lock status to report Loss of Lock (LOL) condition
- Powering down the module for low-power operation (Power-Down mode)

## 29.3 Functional description

This section explains PLL operation and configuration.

### 29.3.1 Modes of operation

**Table 148. Modes of operation**

| PLLCR[PLLPD] | PLLFD[SDMEN] | PLLFM[SSCGBYP] | Description |
|---|---|---|---|
| 1 | x | x | PLL is disabled. |
| 0 | 0 | 1 | Functional mode – PLL operates in integer-only mode. See Clock configuration. |
| 0 | 1 | 1 | Functional mode – PLL operates in Fractional mode (non-Frequency modulation). See Clock configuration. |
| 0 | 1 | 0 | Functional mode – PLL operates in Frequency Modulation mode. See Frequency modulation. |

### 29.3.2 Input clock frequency

PLL is designed to operate over a specified input clock frequency range. PLL source frequency limits are discussed in this chip's data sheet.

### 29.3.3 Clock configuration

See the equations below and the corresponding register configuration that determine the relationship between VCO frequency ($f_{VCO}$) and PLL reference frequency.

- Integer-only mode:
  - When PLLDV[RDIV] is 0:

$$f_{\text{pll_VCO}} = f\text{pll_ref} \times \text{PLLDV[MFI]}$$

**Equation 1. PLL VCO frequency in integer-only mode when PLLDV[RDIV] is 0**

  - When PLLDV[RDIV] is not 0:

$$f_{\text{pll_VCO}} = \frac{f_{\text{pll_ref}}}{\text{PLLDV[RDIV]}} \times \text{PLLDV[MFI]}$$

**Equation 2. PLL VCO frequency in integer-only mode when PLLDV[RDIV] is not 0**

- Fractional mode:
  - When PLLDV[RDIV] is 0:

$$f_{\text{pll_VCO}} = f_{\text{pll_ref}} \times \left(\text{PLLDV[MFI]} + \frac{\text{PLLFD[MFN]}}{18432}\right)$$

**Equation 3. PLL VCO frequency in Fractional mode when PLLDV[RDIV] is 0**

  - When PLLDV[RDIV] is not 0:

$$f_{\text{pll_VCO}} = \frac{f_{\text{pll_ref}}}{\text{PLLDV[RDIV]}} \times \left(\text{PLLDV[MFI]} + \frac{\text{PLLFD[MFN]}}{18432}\right)$$

**Equation 4. PLL VCO frequency in Fractional mode when PLLDV[RDIV] is not 0**

See the equation below and the corresponding register configuration that determine the relationship between reference and PLL_PHI*n* output frequencies.

$$f_{\text{pll_phi}} = \frac{f_{\text{pll_VCO}}}{\text{PLLDV[ODIV2]} \times \left(\text{PLLDIV_}n\text{[DIV]} + 1\right)}$$

**Equation 5. PLL PHI output frequency**

When configuring PLL, you must not violate the maximum system clock frequency or maximum and minimum VCO frequency specification of PLL (see this chip's data sheet for frequency limits).

You must disable PLL by writing 1 to PLLCR[PLLPD] before any PLL configuration or input clock are modified.

You must disable PLL by writing 1 to PLLCR[PLLPD] for at least 5 µs before writing 0 to PLLCR[PLLPD] to enable PLL.

The recommended procedure to program PLL and enter Normal mode is shown in Initialization information.

### 29.3.4 Loss of lock (LOL)

PLL provides LOL indication. The LOL indication can only be generated when PLL is in Functional mode (see Modes of operation). When PLL detects a LOL, it asserts its LOL event output.

PLL does not detect loss of reference clock. If the reference clock stops after PLL achieves lock, PLL continues to indicate lock. It is assumed that monitoring of PLL's reference clock that is done outside PLL is enabled while PLL is in operation.

PLL LOL is intended for detection of gross failures. Use CMUs for accurate frequency monitoring.

### 29.3.5 Frequency modulation

In Frequency Modulation mode, PLL generates a frequency-modulated clock. The modulation depth and modulation frequency are calculated using the equations shown in Frequency modulation programming.

Write 1 to PLLFM[SPREADCTL] to select down-spread modulation. See Figure 114 that shows an example of down-spread modulation.

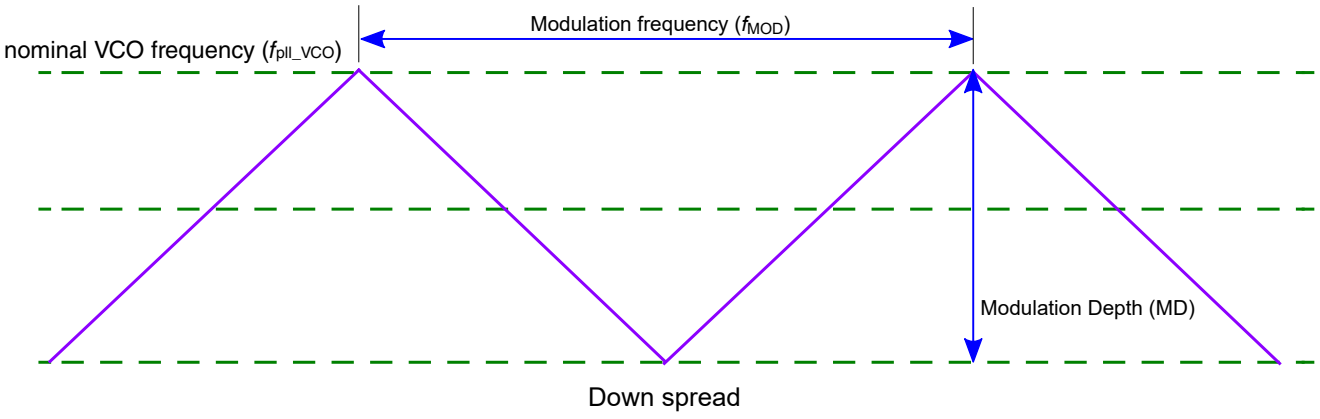



**Figure 114. Frequency modulation**

### 29.3.5.1 Frequency modulation programming

Modulation depth and modulation frequency programming uses step number (PLLFM[STEPNO]) and step size (PLLFM[STEPSIZE]). The table below shows variables used during calculations when programming PLL for frequency modulation.

**Table 149. Variables for configuring modulation depth and frequency**

| Variable | Description |
|---|---|
| $f_{REF}$ | Input clock frequency |
| $f_{MOD}$ | Expected modulation frequency |
| MD | Expected modulation depth in percentage |
| LDF | Loop division factor |
| $f_{pll_VCO}$ | Nominal VCO frequency |

Use the following equations to configure PLL for frequency modulation.

$$LDF = PLLDV[MFI] + \frac{PLLFD[MFN]}{18432}$$

**Equation 6. LDF**

$$PLLFM[STEPNO] = \frac{f_{REF}}{\left(2 \times f_{MOD} \times PLLDV[RDIV]\right)}$$

**Equation 7. Step number**

$$PLLFM[STEPSIZE] = \frac{MD \times LDF}{100 \times PLLFM[STEPNO]} \times 18432$$

**Equation 8. Step size**

Frequency modulation is only possible if the condition shown in Equation 9 is met.

$$(PLLFM[STEPSIZE] \times PLLFM[STEPNO]) < 18432$$

**Equation 9. Requirement to achieve FM**

You must write 0 to PLLFM[SSCGBYP] and write 1 to PLLFD[SDMEN] to enable frequency modulation.

$$Max\ (MD\ \%) = \frac{(f_{REF} \times 100)}{f_{pll_VCO}}$$

**Equation 10. Maximum possible modulation depth when PLL[RDIV] is 0**

$$Max\ (MD\ \%) = \frac{(f_{REF} \times 100)}{PLLDV[RDIV] \times f_{pll_VCO}}$$

**Equation 11. Maximum possible modulation depth when PLL[RDIV] is not 0**

**CAUTION**

The effective modulation depth may differ from the intended modulation depth because of rounding operations applied to PLLFM[STEPSIZE] and PLLFM[STEPNO].

### 29.3.6 Interrupt signals

This module has no interrupt signals.

## 29.4 External signals

This module has no external signals.

## 29.5 Initialization information

Perform the following steps to initialize PLL:

1. Confirm that PLLODIV_*n*[DE] is 0 for all dividers.
2. Confirm that PLLCR[PLLPD] is 1.
3. Program the following as needed:
   - PLLDV
   - PLLFD
   - PLLFM to the desired value
4. Program PLLDV[ODIV2] and PLLODIV_*n*[DIV] to the desired values.
5. Wait for the PLL reference clock to be stable.
6. Write 0 to PLLCR[PLLPD].
7. Wait for PLLSR[LOCK] to be 1.
8. Write 1 to PLLODIV_*n*[DE].

Perform the following steps to shut down PLL:

1. Write 0 to PLLODIV_n[DE] for all dividers.
2. Write 1 to PLLCR[PLLPD].

## 29.6 PLLDIG register descriptions

This section provides the memory map and detailed descriptions of registers used for configuring PLL. The table below shows the memory map. Addresses are given as offsets from the module base address. All registers are accessed using 8-bit, 16-bit, or 32-bit addressing.

### 29.6.1 PLLDIG memory map

PLL base address: 402E_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | PLL Control (PLLCR) | 32 | RW | 8000_0000h |
| 4h | PLL Status (PLLSR) | 32 | RW | 0000_0300h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 8h | PLL Divider (PLLDV) | 32 | RW | 0C3F_1032h |
| Ch | PLL Frequency Modulation (PLLFM) | 32 | RW | 4000_0000h |
| 10h | PLL Fractional Divider (PLLFD) | 32 | RW | 0000_0000h |
| 18h | PLL Calibration Register 2 (PLLCAL2) | 32 | RW | 0006_0000h |
| 80h - 84h | PLL Output Divider (PLLODIV_0 - PLLODIV_1) | 32 | RW | 0000_0000h |

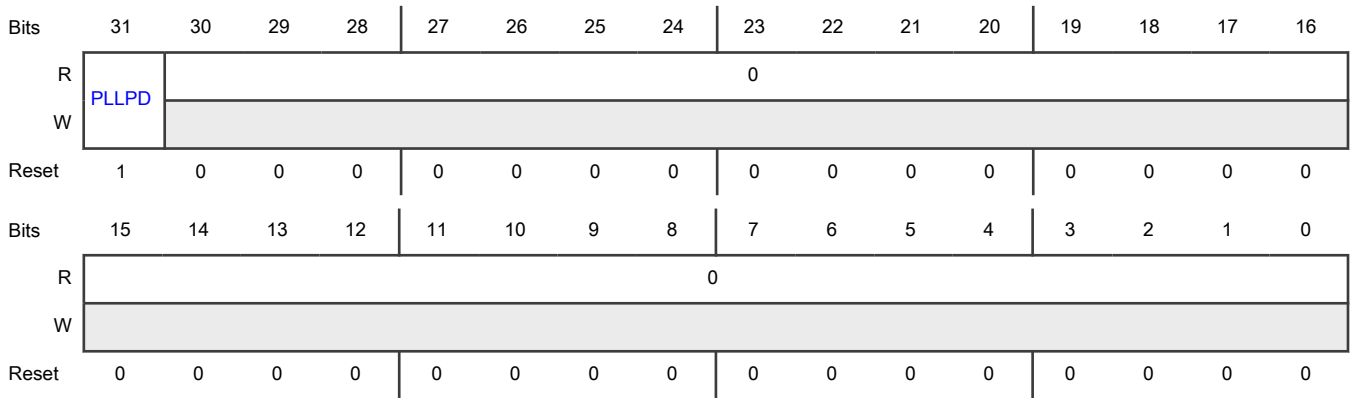
## 29.6.2 PLL Control (PLLCR)

### Offset

| Register | Offset |
|---|---|
| PLLCR | 0h |

### Function
Configures PLL functionality.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>PLLPD | PLL Power Down<br>Powers down or powers up PLL.<br>0b - Powered up<br>1b - Powered down |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 30-0<br>— | Reserved |

## 29.6.3 PLL Status (PLLSR)

### Offset

| Register | Offset |
|---|---|
| PLLSR | 4h |

### Function

Shows the PLL status.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | Reserved | | | | 0 | | | | LOL | LOCK | 0 | |
| W | | | | | | | | | | | | | W1C | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-11<br>— | Reserved |
| 10-8<br>— | Reserved |
| 7-4<br>— | Reserved |
| 3<br>LOL | Loss-Of-Lock Flag<br>Indicates the current PLL lock status. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No loss of lock detected<br>1b - Loss of lock detected |
| 2<br>LOCK | Lock Status<br>Indicates that PLL has acquired lock.<br>0b - Unlocked<br>1b - Locked |
| 1-0<br>— | Reserved |

## 29.6.4 PLL Divider (PLLDV)

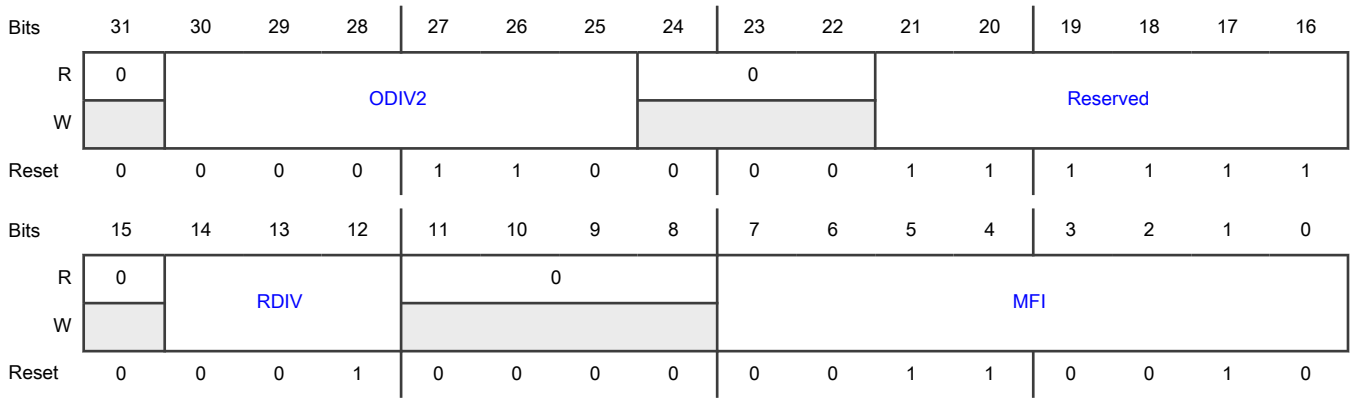
### Offset

| Register | Offset |
|---|---|
| PLLDV | 8h |

### Function

Divides input clocks for PLL output generation.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 30-25<br>ODIV2 | Output frequency divider for raw PLL clock.<br>6-bit field determining the VCO clock post divider for driving the PHI output clock.<br>000000 – Divide by 1<br>000001 – Divide by 1<br>000010 – Divide by 2<br>000011 – Divide by 3<br>000100 – Divide by 4<br>.....<br>111111 – Divide by 63 |
| 24-22<br>— | Reserved |
| 21-16<br>— | Reserved |
| 15<br>— | Reserved |
| 14-12<br>RDIV | Input Clock Predivider<br>Sets the input clock divider.<br>The output of the predivider circuit generates the PLL loop reference clock.<br>000b - Divide by 1<br>001b - Divide by 1<br>010b - Divide by 2<br>011b - Divide by 3<br>100b - Divide by 4<br>101b - Divide by 5<br>110b - Divide by 6<br>111b - Divide by 7 |
| 11-8<br>— | Reserved |
| 7-0<br>MFI | Integer Portion Of Loop Divider<br>Sets the value of the divider in the PLL feedback loop.<br>The value specified establishes the multiplication factor applied to the reference frequency. Write the divider value to this field, where the chosen value does not violate VCO frequency specifications. |

## 29.6.5 PLL Frequency Modulation (PLLFM)

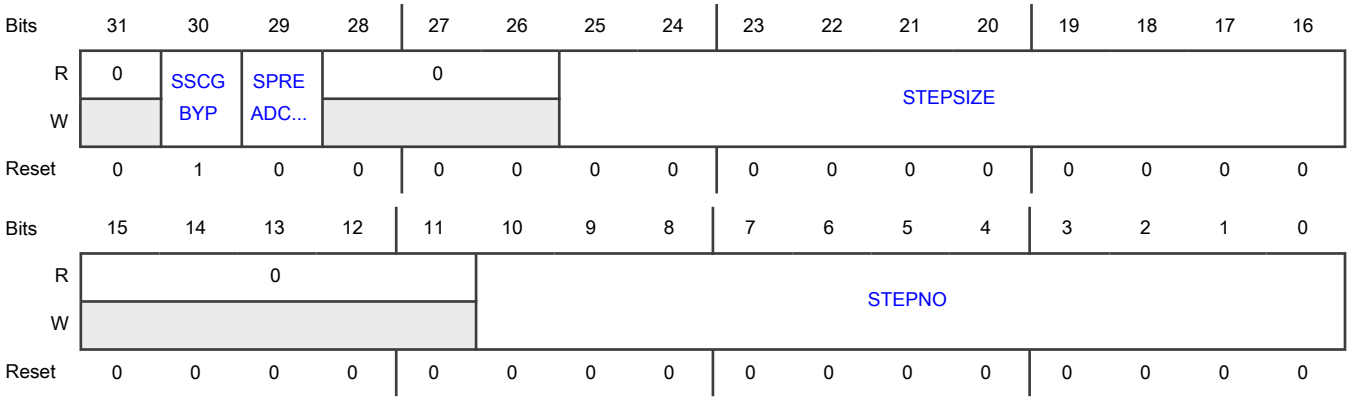
### Offset

| Register | Offset |
|---|---|
| PLLFM | Ch |

### Function

Configures PLL frequency modulation parameters.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>SSCGBYP | Frequency Modulation (Spread Spectrum Clock Generation) Bypass<br>Bypasses frequency modulation.<br>0b - Not bypassed<br>1b - Bypassed |
| 29<br>SPREADCTL | Modulation Type Selection<br>Indicates that the modulation is spread below the nominal frequency. You must write 1 to this field.<br>0b - Reserved<br>1b - Spread below nominal frequency |
| 28-26<br>— | Reserved |
| 25-16 | Frequency Modulation Step Size |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| STEPSIZE | Provides the step size for modulation depth and frequency in Frequency Modulation mode (see Frequency modulation). |
| 15-11<br>— | Reserved |
| 10-0<br>STEPNO | Number Of Steps Of Modulation Period Or Frequency Modulation<br>Provides the number of steps to achieve modulation depth in Frequency Modulation mode (see Frequency modulation). |

## 29.6.6 PLL Fractional Divider (PLLFD)

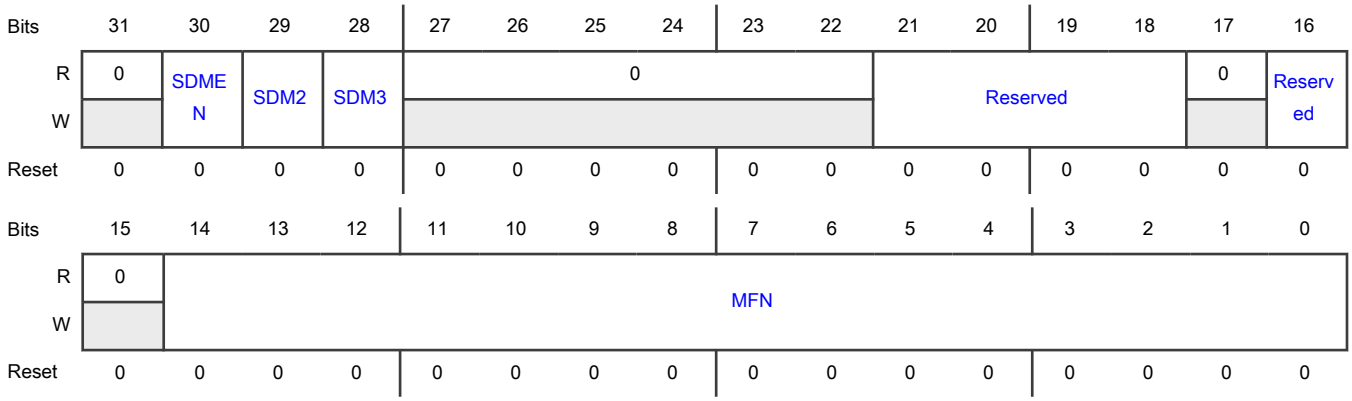
### Offset

| Register | Offset |
|---|---|
| PLLFD | 10h |

### Function

Enables and configures frequency modulation.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>SDMEN | Fractional Mode Enable<br>Enables Fractional mode. |

*Table continues on the next page...*

*Table continued from the previous page...*

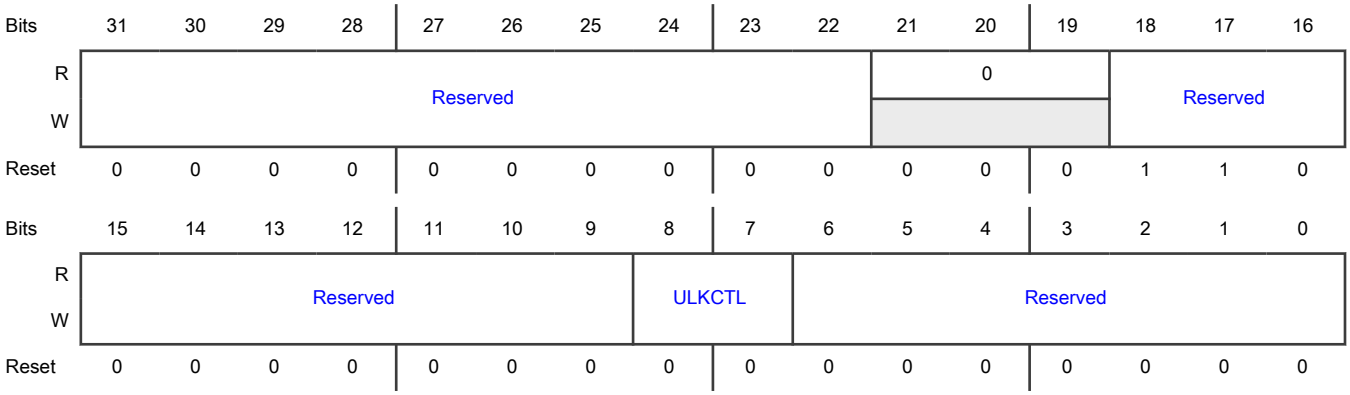
| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |
| 29<br>SDM2 | Fractional Mode Configuration<br>When you are in the fractional mode (SDMEN = 1), write 1 to this field.<br>**NOTE**<br>If SDMEN = 1, this field must be written 1. |
| 28<br>SDM3 | Fractional Mode Configuration<br>When you are in the fractional mode (SDMEN = 1), write 1 to this field.<br>**NOTE**<br>If SDMEN = 1, this field must be written 1. |
| 27-22<br>— | Reserved |
| 21-18<br>— | Reserved |
| 17<br>— | Reserved |
| 16<br>— | Reserved |
| 15<br>— | Reserved |
| 14-0<br>MFN | Numerator Of Fractional Loop Division Factor<br>Sets the numerator of the fractional loop division factor.<br>You must write a value of less than 18432 to this field. When Fractional mode is disabled, you must write 000_0000_0000_0000b to this field. |

## 29.6.7 PLL Calibration Register 2 (PLLCAL2)

**Offset**

| Register | Offset |
|---|---|
| PLLCAL2 | 18h |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-22<br>— | Reserved |
| 21-19<br>— | Reserved |
| 18-16<br>— | Reserved |
| 15-9<br>— | Reserved |
| 8-7<br>ULKCTL | Unlock Control Accuracy<br>Defines the accuracy necessary to achieve unlock.<br>The lock counter determines unlock if the number of VCO clock cycles in the window of reference cycles is outside the number of cycles defined by this field.<br>00b - Unlock range = Expected value ± 9 (recommended when PLLFM[SSCGBYP] = 1). Unlock range = Expected value ± 9 (recommended when PLLFM[SSCGBYP] = 1)<br>01b - Unlock range = Expected value ± 17 (recommended when PLLFM[SSCGBYP] = 0). Unlock range = Expected value ± 17 (recommended when PLLFM[SSCGBYP] = 0)<br>10b - Unlock range = Expected value ± 33<br>11b - Unlock range = Expected value ± 5 |
| 6-0<br>— | Reserved |

## 29.6.8 PLL Output Divider (PLLODIV_0 - PLLODIV_1)

### Offset

| Register | Offset |
|---|---|
| PLLODIV_0 | 80h |
| PLLODIV_1 | 84h |

### Function

Controls the PLL output clock divider settings.

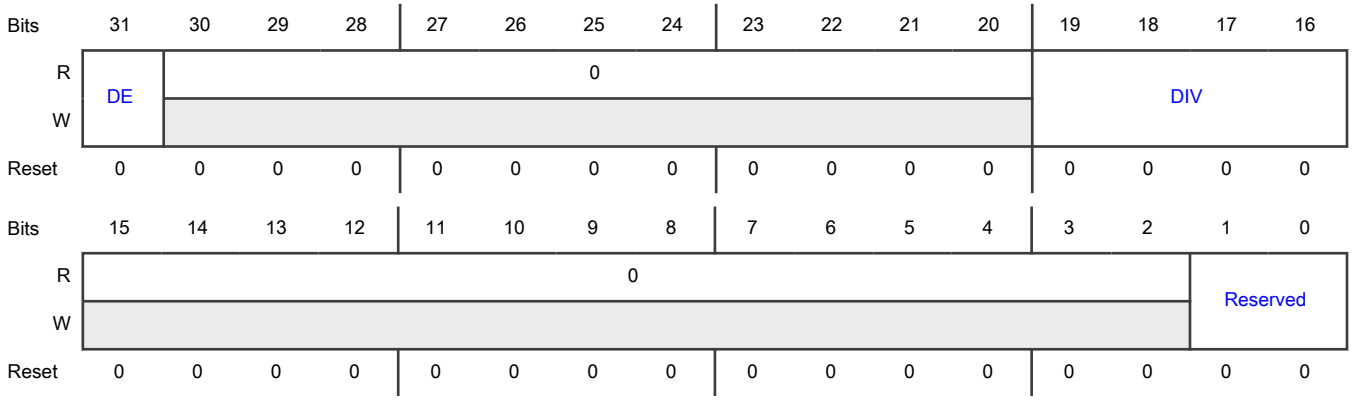
This divider has a 50% duty cycle.

**NOTE**

These registers support only word accesses. Other write accesses lead to the following:

- Unpredictable behavior
- No transfer error generated

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>DE | Divider Enable<br>Enables PLL output divider. Divider must be disabled before disabling PLL.<br>0b - Disabled<br>1b - Enabled |
| 30-20<br>— | Reserved |
| 19-16 | Division Value |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| DIV | Provides the division value for the output clock divider. The clock period of the clock after division is DIV + 1 times the time period of the divider input clock. |
| 15-2<br>— | Reserved |
| 1-0<br>— | Reserved<br>Do not write any value other than the reset value. |

# Chapter 30
# Reset Overview

## 30.1 Introduction

This chip's reset logic consists of a reset sequence that leads the chip to a fixed deterministic state after predefined reset events occur. These events can pertain to chip failure events, the chip's special operating conditions, or certain software-governed events to initiate a chip reset sequence. This chapter discusses the chip reset scheme and related topics such as:

- Types of reset reactions
- Reset event sources
- Chip reset sequences
  - POR
  - Destructive reset
  - Functional reset
- RAM retention across functional reset
- Reset pin (RESET_b) behavior
- Debug system reset
- Signal-level reset flow

## 30.2 Chip reset types and reactions

### 30.2.1 Chip reset blocks

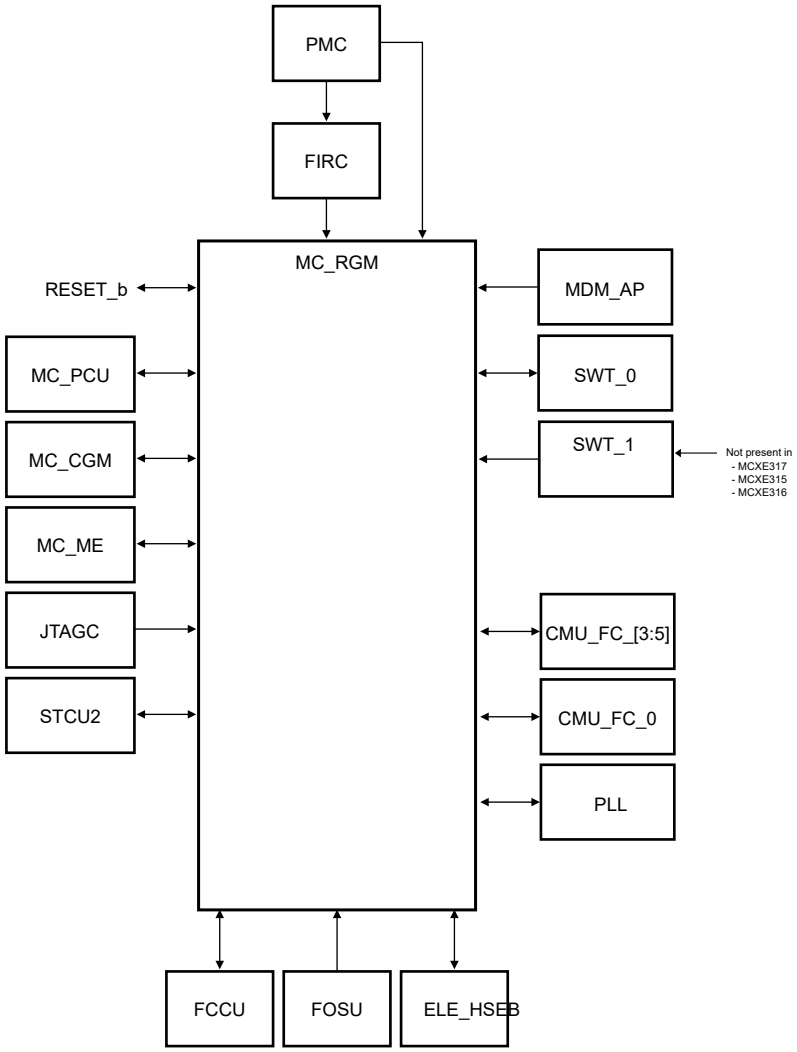



Figure 115. Chip reset blocks

### 30.2.2 Chip reset types

Table 150. Chip reset types

| Reset event type | Functional description |
|---|---|
| POR | Leads to a complete chip reset. |
| Destructive | Leads most parts of the chip, except a few modules, to reset. SRAM content is lost after this reset event. |
| Functional | Leads all the communication peripherals and cores to reset. The communication protocols' sanity is not guaranteed and they are assumed to be reinitialized after reset. The SRAM content, and the functionality of certain modules, is preserved across functional reset. |

### 30.2.3 High-level reset sequence overview

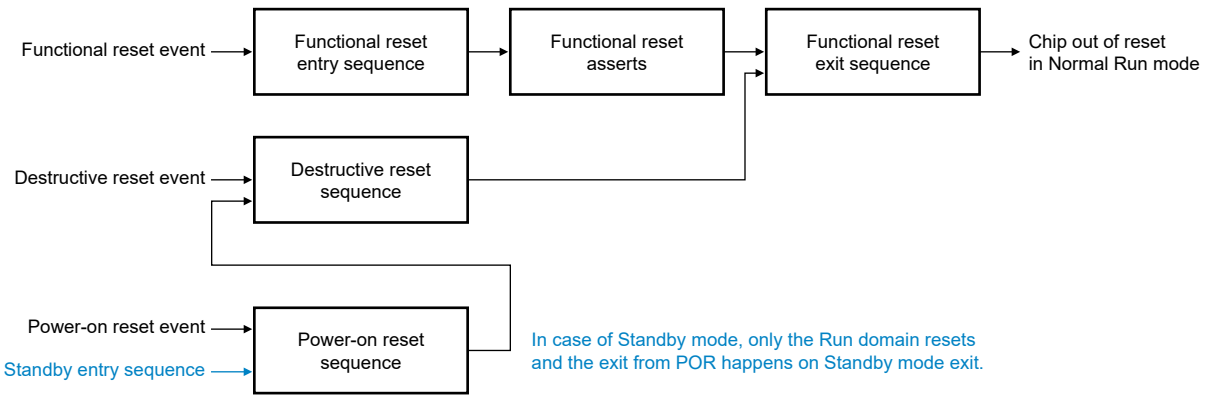



Figure 116. High-level reset sequence overview

#### 30.2.3.1 Reset event reactions

Table 151. Reset event reactions

| Reset event type | Triggered from | Reaction |
|---|---|---|
| POR | Anywhere | Moves to the beginning of the power-on sequence |
| Destructive reset | Power-on sequence | No reset sequence change |
| | Anywhere in the chip operation except in the power-on-sequence | Moves to the beginning of the destructive reset sequence |
| Functional reset | Out-of-reset | Moves to the beginning of the functional reset entry sequence |
| | | |
| | Anywhere within the functional reset sequence | No reset sequence change |

#### 30.2.3.2 Chip action after reset event

For each reset event, immediately after MC_RGM captures it, the chip performs these actions:

1. Writes 1 to the corresponding reset event status fields in MC_RGM.DES and MC_RGM.FES (see the MC_RGM chapter for more information).
2. Places its pins in their default states (see the IOMUX file attached to this document for more information).
3. Asserts the RESET_b pin.

**NOTE**

After self-test completes, you can configure RESET_b assertion using MC_RGM.FES[ST_DONE].

4. Enters the reset sequence as described in Reset event reactions, depending on the current state and reset event type.

## 30.3 Reset sources—POR, destructive, and functional

MC_RGM records reset events in MC_RGM.FES and MC_RGM.DES, indicating the source of functional reset events and destructive reset events, respectively. You must read these fields to identify the reset source on reset recovery.

### 30.3.1 POR sources

**Table 152. POR sources**

| Source module | Field in MC_RGM.DES | RESET_b assertion | Description |
|---|---|---|---|
| PMC | F_POR | Always | VDD_LV POR |
| | | | LVR on 1.1 V supply in Standby mode |
| | | | LVR on 1.1 V supply in Run mode |
| | | | LVR on 2.5 V supply in Standby mode |
| | | | LVR on 2.5 V supply in Run mode |
| | | | LVR on VDD_HV_A supply in Standby mode |
| | | | LVR on VDD_HV_A supply in Run mode |
| | | | LVR on VDD_HV_B supply in Standby mode [1] |
| | | | LVR on VDD_HV_B supply in Run mode [1] |
| POR_WDG | | | POR_WDG timeout (see the POR_WDG chapter for more information) |

1. LVR on VDD_HV_B run mode and LVR on VDD_HV_B standby mode are not present in MCXE315/MCXE316 and MCXE317.

**NOTE**

You cannot escalate or demote POR to an interrupt.

### 30.3.2 Stages of the POR sequence

**Table 153. Stages of the POR sequence**

| Stage | Process |
|---|---|
| PWRUP | 1. Starts after a POR event (for example, a POR source assert).<br>2. Waits for the power-up sequence to complete.<br>3. Exits when all the POR sources clear.<br>4. Transitions to the FIRC_STRT stage after the procedure completes. |
| FIRC_STRT | 1. Enters this stage after exiting the PWRUP stage.<br>**NOTE**<br>FIRC_CLK, if enabled, becomes available after it is stable. The duration depends on the clock startup time (see the chip datasheet for more information). The MC_RGM state machine proceeds further after FIRC_CLK is available.<br>2. Transitions to the destructive reset sequence after the procedure completes. |

### 30.3.3 Destructive reset sources

**Table 154. Destructive reset sources**

| Source module[1] | Field in MC_RGM.DES | Description |
|---|---|---|
| FOSU | FCCU_FTR | FCCU failure to react |
| STCU2 | STCU_URF | STCU2 unrecoverable fault |
| MC_RGM | MC_RGM_FRE | Functional reset escalation |
| CMU_FC_0 | FXOSC_FAIL | FXOSC failure |
| PLL | PLL_LOL | PLL loss of lock |
| CMU_FC_3 | CORE_CLK_FAIL | Core clock failure |
| CMU_FC_4 | AIPS_PLAT_CLK_FAIL | AIPS_PLAT_CLK failure |
| CMU_FC_5 | HSE_CLK_FAIL | HSE_CLK failure |
| MC_CGM | SYS_DIV_FAIL | System clock dividers alignment failure |
| ELE_HSEB | HSE_TMPR_R ST | ELE_HSEB tamper detect reset |
| ELE_HSEB | HSE_SNVS_RST | ELE_HSEB SNVS tamper detection |
| MC_ME | SW_DEST | Software destructive reset |
| MDM_AP | DEBUG_DEST | Debug destructive reset |
| RESET_b pin | EXT_RST | RESET_b pin assertion |

1. All destructive resets can be escalated, but only the PLL LOL destructive reset can be demoted to an interrupt (see Destructive reset event bypass for PLL LOL destructive reset bypass details).

**NOTE**

All reset sources in the table above assert the RESET_b pin.

### 30.3.4 Destructive sequence stage description

**Table 155. DEST0 description**

| Stage | Process |
|---|---|
| DEST0 | 1. Asserts reset to the entire chip, except logic running on POR.<br>2. Waits for all the destructive reset events to clear.<br>3. Waits for the minimum destructive reset assertion duration of eight FIRC_CLK cycles.<br>4. Deassets after stage completion. |

### 30.3.5 Functional reset sources

**Table 156. Functional reset sources**

| Source module | Field in MC_RGM.FES | RESET_b assertion | Demotable to IRQ[1] | Escalation[2] | Description |
|---|---|---|---|---|---|
| FCCU soft reaction[3] | FCCU_RST | Always | Yes[4] | Yes | FCCU reset reaction |

*Table continues on the next page...*

Table 156. Functional reset sources (continued)

| Source module | Field in MC_RGM.FES | RESET_b assertion | Demotable to IRQ[1] | Escalation[2] | Description |
|---|---|---|---|---|---|
| STCU2 | ST_DONE | Configurable | No | No | Self-test done |
| SWT_0 | SWT0_RST | Always | Yes[5] | Yes | SWT reset request |
| SWT_1[6] | SWT1_RST | Always | Yes[7] | Yes | SWT reset request |
| JTAGC | JTAG_RST | Always | Yes [8] | No | JTAG reset |
| ELE_HSEB | HSE_SWT_RST | Always | No | Yes | ELE_HSEB SWT timeout |
| ELE_HSEB | ELE_HSEBOOT_RST | Always | No | Yes | ELE_HSEB boot reset |
| MC_ME | SW_FUNC | Always | No | Yes | Software functional reset |
| MDM_AP | DEBUG_FUNC | Always | Yes [9] | Yes | Debug functional reset |

1. See Functional reset demotion to an interrupt for more information.
2. See Reset escalation for more information.
3. An FCCU soft functional reset is a chip functional reset (see the FCCU chapter for more information).
4. Controlled by MC_RGM.FERD[D_FCCU_RST].
5. Controlled by MC_RGM.FERD[D_SWT0_RST].
6. SWT_1 is not present in MCXE315/MCXE316 and MCXE317.
7. Controlled by MC_RGM.FERD[D_SWT1_RST].
8. Controlled by MC_RGM.FERD[D_JTAG_RST].
9. Controlled by MC_RGM.FERD[D_DEBUG_FUNC].

### 30.3.6 Functional reset sequence descriptions

Table 157. Functional reset sequence descriptions

| Stage | Series of events |
|---|---|
| **Functional reset entry sequences** | |
| FUNC0 | This stage starts after any functional reset event.<br>The FCCU fault monitoring and CMU_F*x_n* monitoring for FLL events is masked in this step to avoid any false fault or reset. |
| FUNC1 | In this stage, a halt sequence that includes daisy chaining of all the gaskets halts, disabling the crossbar.<br>The stage completes after the halt-handshake sequence completes. |
| FUNC2 | In this stage, MC_RGM triggers the MC_CGM hardware clock multiplexers to switch to FIRC_CLK.<br>• Software-based clock multiplexers do not support switching to FIRC_CLK on functional reset.<br>• If PCFS is enabled, the system clock switching can be done via PCFS.<br>This stage completes after MC_CGM switches the system clock to FIRC. |
| FUNC3 | In this stage, MC_RGM triggers all the MC_CGM hardware-based clock multiplexers with PCFS enabled or disabled to move their dividers to default values. |

*Table continues on the next page...*

**Table 157. Functional reset sequence descriptions (continued)**

| Stage | Series of events |
|---|---|
| | • Software-based clock multiplexers do not support this feature.<br>This stage completes after all the clock multiplexer dividers initialize to their corresponding default values. |
| FUNC4 | In this stage, PLLDIG turns off synchronously.<br>The stage completes after PLLDIG turns off. |
| FUNC5 | FXOSC_CLK switches off synchronously.<br>The FUNC4 and FUNC5 stages ensure that PLLDIG disables cleanly to ensure there are no glitches on the PLLDIG clock because of reset.<br>The stage completes after FXOSC switches off. |
| FUNC6 | In this stage, clocks of modules that are a part of LBIST and working on the destructive reset are enabled to meet their synchronous reset requirements, if any. For the self-test logic, in self-test, the destructive reset deasserts after this stage completes and after safe stating is removed.<br>**NOTE**<br>In the self-test sequence, the logic, which is a part of self-test (LBIST logic) resets when self-test completes. All the parts of logic in self-test (POR, destructive, and functional reset) reset after self-test completes whereas the rest of the chip undergoes a functional reset sequence because of self-test completion (MC_RGM.FES[ST_DONE] = 1). See the Safety Overview and STCU2 chapters for more information on the self-test operation. |
| FUNC7 | MC_RGM asserts the functional reset and triggers a counter running on FIRC_CLK (for up to 64 cycles) to enable clocks for the modules having synchronous reset requirements.<br>Flash memory comes out of reset after this stage completes.<br>**NOTE**<br>The flash memory resets after a functional reset event but comes out of reset, before the rest of the modules do, at the start of the functional reset exit sequence. The rest of the modules reset when the functional reset comes out of reset at the end of the functional reset exit sequence. Therefore, the reset to flash memory is an early functional reset that deasserts earlier than the functional reset, even if asserted at the same time. |
| **Functional reset exit sequences** | |
| FUNC8 | This stage consists of flash memory and MC_RGM handshaking.<br>Flash memory indicates the completion of its initialization to MC_RGM. |
| FUNC9 | DCM initiates the scanning of flash memory DCF records.<br>This state completes after the flash memory scanning completes. See the DCF clients file attached to this document for more information. |
| FUNC10 | After DCM scans the DCF records from the flash memory, DCM initiates the trim loading sequence for analog blocks.<br>The analog blocks are loaded with the configured trimmed values in this stage, which completes after a trim-loading sequence completes. |

*Table continues on the next page...*

Table 157. Functional reset sequence descriptions (continued)

| Stage | Series of events |
|---|---|
| FUNC11 | In this stage, MG_RGM stops driving RESET_b and checks that the signal does not assert externally.<br>If you enable low-power debug, MC_RGM waits for a debug acknowledge.<br>The completion of this stage indicates that MC_RGM completed the reset sequence, deasserting the functional reset to the system. |

## 30.4 Reset and boot sequence

The chip reset sequence consists of several reset stages based on the occurrence of a particular reset event. All reset events follow the same chip reset sequence; only the entry points vary depending on the type of reset event. MC_RGM triggers each stage after the previous stage completes. These stages execute in a specific order, which ensures a deterministic state of the chip when a reset event completes.

Figure 117 shows a high-level representation of the chip startup sequence.

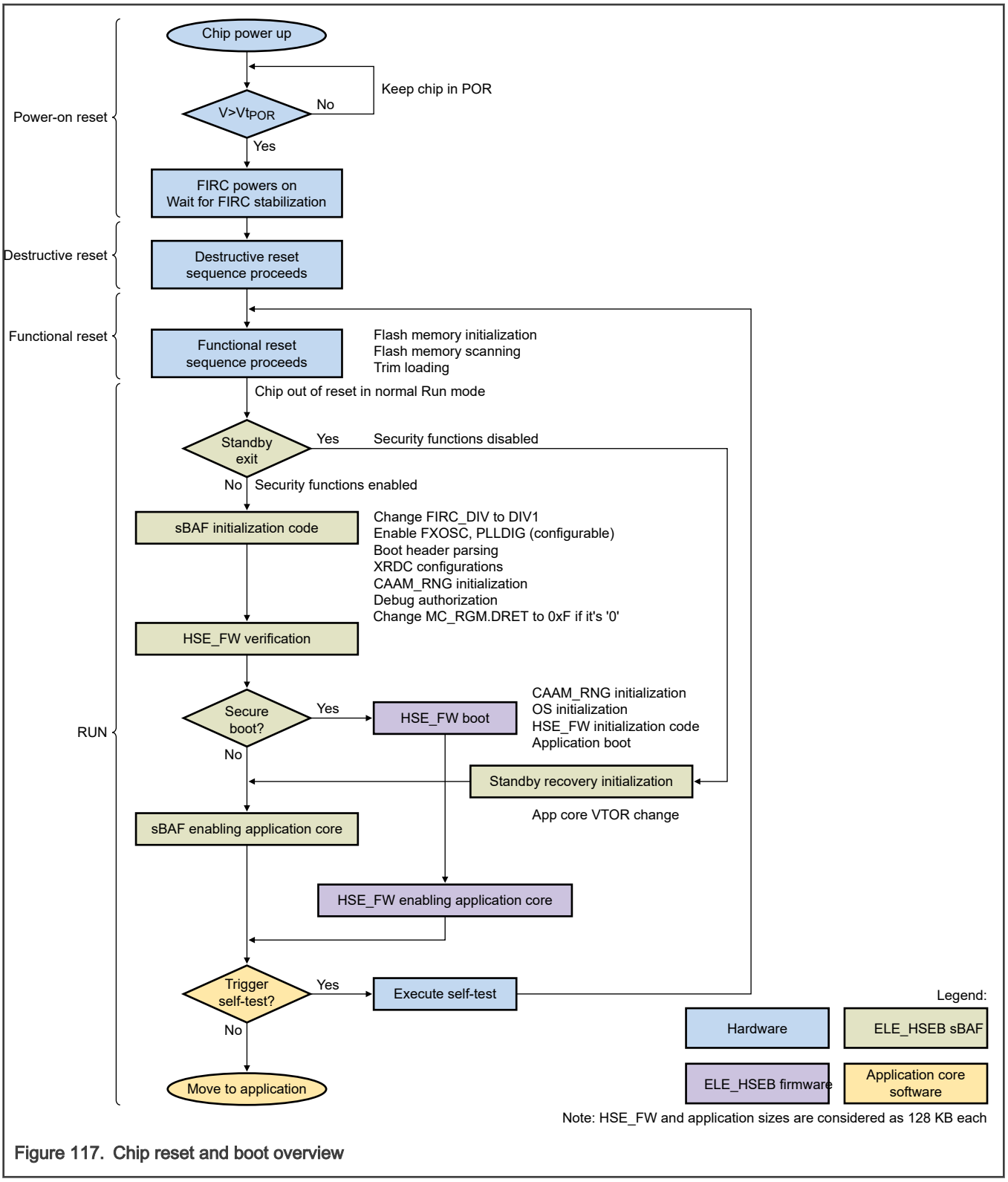



**Figure 117. Chip reset and boot overview**

## 30.4.1 POR

This stage starts when the POR event occurs, that is, when a POR source asserts. The logic within the Run power domain running on POR also resets in the chip Standby entry sequence.

**NOTE**

The logic in the Standby power domain does not reset in the chip Standby mode entry sequence.

The POR sequence consist of two stages:

1. Power-up (PWRUP)
2. FIRC oscillator start (FIRC_STRT)

See POR sequence for more information and Stages of the POR sequence for POR stages and their descriptions.

#### 30.4.1.1 POR sequence

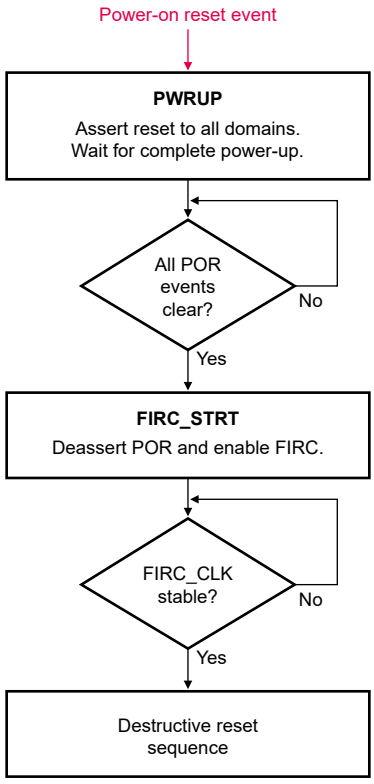



Figure 118. POR flow

### 30.4.2 Destructive reset

The chip enters a destructive reset sequence after the POR sequence or any destructive reset event completes. Destructive reset sequence illustrates the destructive reset sequence and Destructive sequence stage description discusses the stages of destructive sequence.

#### 30.4.2.1 Destructive reset sequence

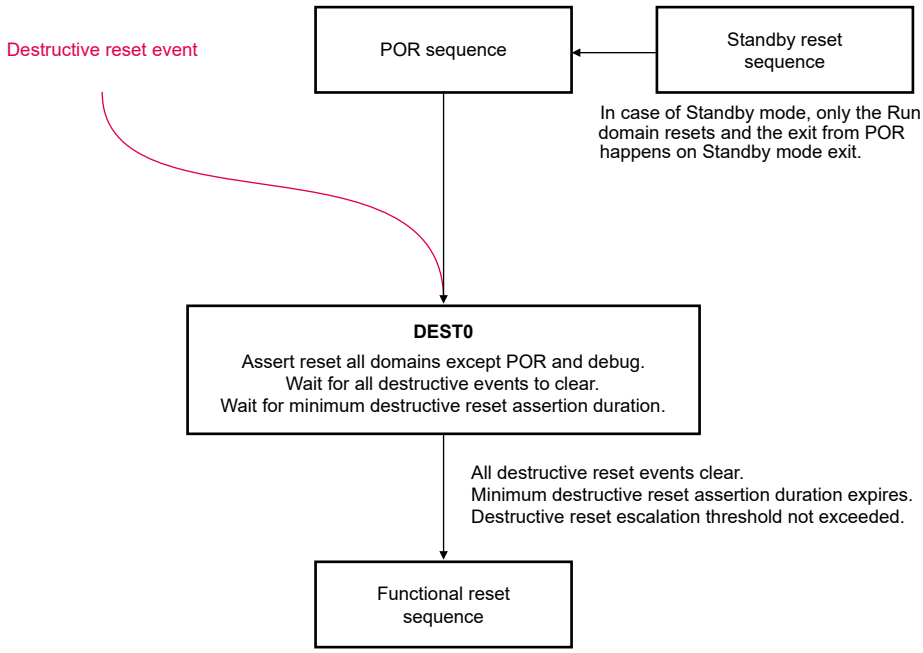



Figure 119. Destructive reset sequence

##### 30.4.2.1.1 Destructive sequence stage description

Table 158. DEST0 description

| Stage | Process |
|---|---|
| DEST0 | 1. Asserts reset to the entire chip, except logic running on POR.<br>2. Waits for all the destructive reset events to clear.<br>3. Waits for the minimum destructive reset assertion duration of eight FIRC_CLK cycles.<br>4. Deassets after stage completion. |

#### 30.4.2.2 Destructive reset event bypass

This chip supports a destructive reset event demotion mechanism that the application software configures. The destructive reset bypasses and an interrupt event occurs (demotion). Table 159 discusses details related to GPR configuration and corresponding interrupt identification.

A successful chip operation is not guranteed if a destructive reset event is bypassed.

Table 159. Destructive reset event bypass

| Destructive reset event | Destructive reset event description | DCM field to bypass reset event | NVIC interrupt ID |
|---|---|---|---|
| MC_RGM.DES[PLL_LOL] | PLL loss of lock | DCM.DCMRWP3[9] | 212 |

### 30.4.3 Functional reset

The chip enters the functional reset sequence when any of the following events occur:

- Functional reset
- POR or destructive reset (after the DEST0 stage completion)

On any functional reset event, the chip starts a functional reset entry sequence before the functional reset asserts and ensures the stability of logic running on a destructive reset and POR. On a destructive reset event or POR events the functional reset entry sequence does not execute.

The functional reset exit sequence consists of steps that ensure proper initialization of the chip after functional reset recovery.

#### 30.4.3.1 Functional reset sequence

Functional reset flow illustrates the functional reset flow and Functional reset sequence descriptions discusses the functional reset stages and their descriptions.

Stages FUNC0 to FUNC6 present the functional reset entry sequence. It occurs on any functional reset event before the functional reset. In other words, when a functional reset event occurs, MC_RGM holds the asserted reset and executes the functional reset entry sequence. After the sequence completes, MC_RGM resets the chip, which remains in Run mode during the functional reset entry sequence.

Stages FUNC7 to FUNC11 present the functional reset exit sequence, which occurs after a functional reset event before deasserting the chip reset. This includes handshaking with the flash memory and analog blocks, ensuring correct operation after reset exit.

#### 30.4.3.1.1 Functional reset flow

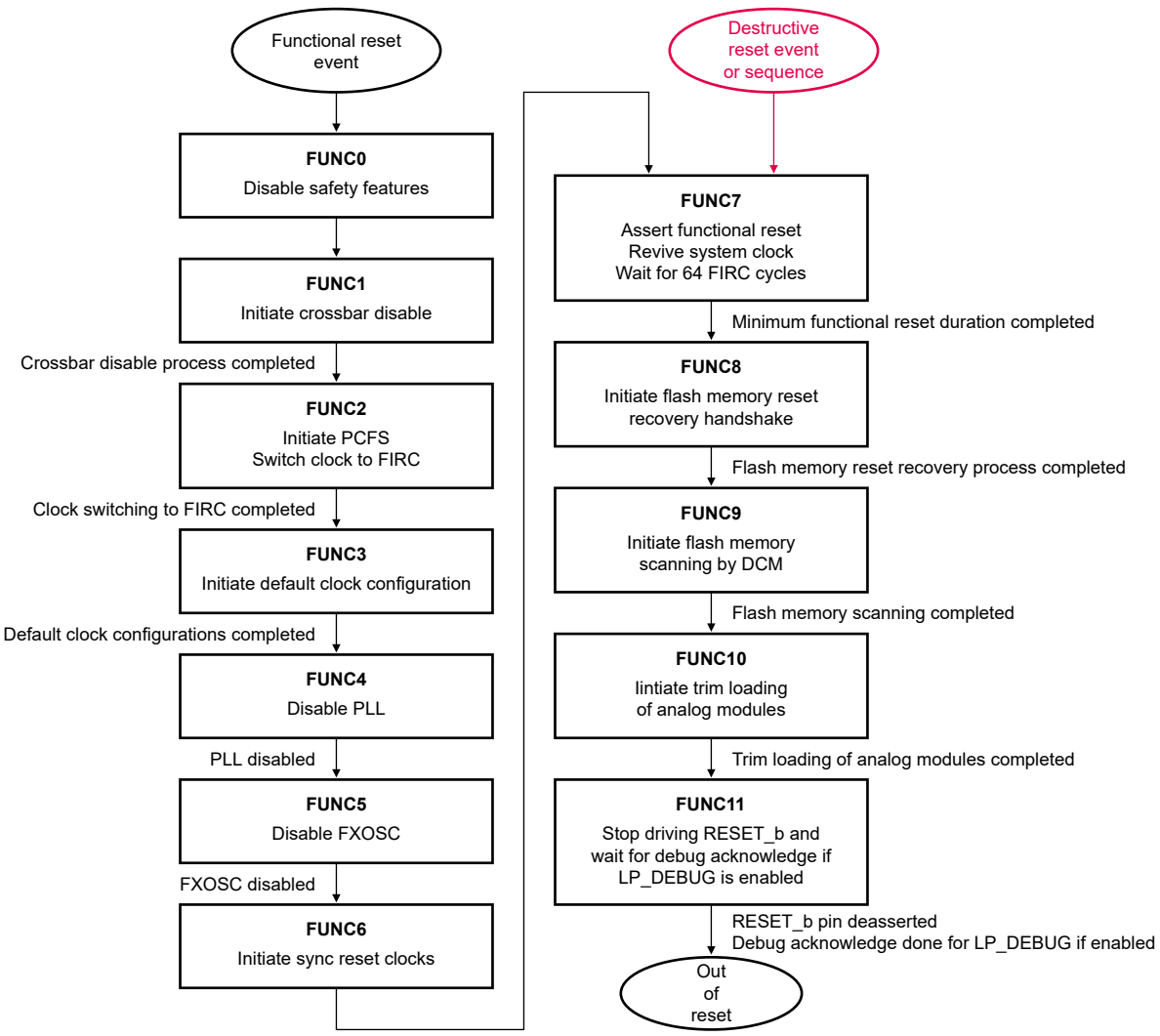



Figure 120. Functional reset flow

### 30.4.3.2 FUNC9 and FUNC10 stage bypass for faster Standby mode exit

The chip supports optional bypassing of the FUNC9 and FUNC10 (only FIRC and PMC phases) stages on Standby mode exit to considerably reduce the Standby mode exit duration. This feature is recommended only for Standby mode exit and must be configured on:

- Standby mode entry sequence
- SW3
- Disabled on Standby mode exit

See the "Faster Standby recovery" section in the "Power Management" chapter for more information.

### 30.4.3.3 Standby reset sequence

The logic within Run domain is reset additionally apart from the reset sequence in the chip standby entry sequence, wherein all the resets assert (POR, destructive, and functional) to the Run domain logic. The chip standby entry sequence does not have any impact on the logic in Standby power domain.

Wake-up from Standby mode removes the resets to the Run domain in the standby entry sequence.

See the "Peripheral reset status" section in the "Reset Overview" chapter for the logic present in Run domain.

#### 30.4.3.4 Reset function redirection

Resets may escalate or demote to an IRQ, depending on the chip configuration.

##### 30.4.3.4.1 Functional reset demotion to an interrupt

This chip supports the reset sequence demotion feature for functional resets. You can configure a functional reset to create an interrupt instead of a reset (see the MC_RGM chapter for details).

##### 30.4.3.4.2 Reset escalation

The chip supports the reset escalation feature. If multiple functional or destructive resets occur, the related reset can escalate to a higher priority reset sequence (see the "Functional reset escalation" and "Destructive reset escalation" sections in the MC_RGM chapter).

###### 30.4.3.4.2.1 Destructive reset escalation

You can enable destructive reset escalation by configuring a DCF client. You must also configure the destructive count threshold in MC_RGM.DRET (see "Destructive Reset Escalation Enable Register (DEST_RST_ESC):dcf_client_dest_rst_esc" in the DCF file attached to this document). The escalation event can individually be enabled or disabled for each reset source, and the fields in the dcf_client_dest_rst_esc register correspond with the fields in MC_RGM.DES register.

After being configured, MC_RGM immediately asserts a destructive reset escalation when the destructive event count reaches the threshold count in MC_RGM.DRET[DRET]. When the destructive and escalation reset assert, the reset sequence immediately enters the DEST0 state. The reset sequencing remains in DEST0 until a POR event occurs. If enabled, the destructive reset escalation counter increments with each destructive reset event. The application software clears the destructive reset escalation counter by writing any value to MC_RGM.DRET[DRET].

**NOTE**

You can configure GPR settings to allow demotion of destructive resets to interrupts instead of escalation (DCMRWP3[DEST_RST9_AS_IPI]). See Destructive reset event bypass for more information.

## 30.5 Reset timing diagram

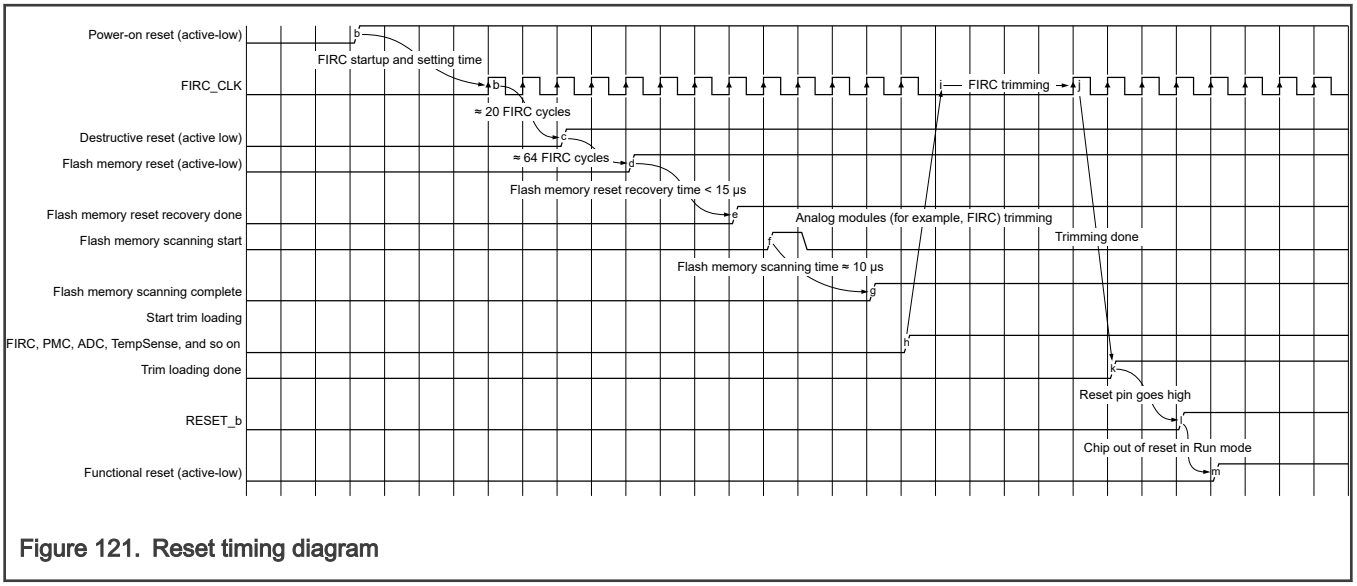



Figure 121. Reset timing diagram

## 30.6 Chip status after reset deassertion

**Table 160. Chip status after reset deassertion**

| Function or feature | After POR deassertion | After destructive reset deassertion | After functional reset deassertion |
|---|---|---|---|
| Clock sources | • FIRC_CLK and SIRC_CLK on<br>• Others off | • FIRC_CLK and SIRC_CLK on<br>• Others off | • FIRC_CLK and SIRC_CLK on<br>• SXOSC_CLK same as before functional reset<br>• FXOSC_CLK and PLL_PHIn_CLK off |
| Clock selection | FIRC_CLK | FIRC_CLK | FIRC_CLK |
| Clock dividers | Default configuration | Default configuration | Default configuration |
| Reset status flags | MC_RGM.DES[F_POR] equals 1, others equal 0 | MC_RGM.DES[F_DR_n] equals 1, others equal 0 | MC_RGM.FES[F_FR_n] equals 1, others equal 0 |
| MC_ME previous mode | Reset | Reset | Reset |
| FCCU fault information | Cleared | Cleared | Retained |
| ELE_HSEB RAM | Initialized | Initialized | Initialized |
| SRAM content | Invalid | Invalid | Retained |
| DCF configurations in DCM | Reset value | Existing loaded value (reset value after POR) | Reloaded from flash memory |
| Cores | All off | All off | ELE_HSEB core on, others off |
| Logic on POR[1] | Out of reset with default configuration | Out of reset with default configuration | Out of reset with default configuration |
| Logic on destructive reset [1] | Under reset with default configuration | Out of reset with default configuration | Out of reset with default configuration |
| Logic on functional reset [1] | Under reset with default configuration | Under reset with default configuration | Out of reset with default configuration |

1. For the list of peripherals affected by POR, destructive reset, and functional reset events, see Module reset status.

## 30.7 Module reset status

**Table 161. Module reset status**

| Module instances[1] | Destructive | Functional | Power domain[2] | Part of LBIST[3] |
|---|---|---|---|---|
| MC_RGM | Y | Y | Standby | No |
| PRAMC | Y | Y | Run | Yes |
| PFC | Y | Y | Run | Yes |
| SIUL_VIRTWRAPPER_PDAC0 | Y | Y | Run | No |

*Table continues on the next page...*

Table 161. Module reset status (continued)

| Module instances[1] | Destructive | Functional | Power domain[2] | Part of LBIST[3] |
|---|---|---|---|---|
| SIUL_VIRTWRAPPER_PDAC1 | Y | Y | Run | No |
| SIUL_VIRTWRAPPER_PDAC2 | Y | Y | Run | No |
| SIUL_VIRTWRAPPER_PDAC3 | Y | Y | Run | No |
| DCM | Y | Y | Run and Standby[4] | No |
| TRGMUX | Y | Y | Run | No |
| WKPU | Y | Y | Standby | No |
| CMU_F*x*_[0:5] | Y | Y | Run | CMU 0-3: No<br>CMU 1-2: Yes<br>CMU 4-5: Yes |
| FIRC | Y[5] | Y[5] | Standby | No |
| FXOSC | Y | Y | Standby | No |
| MC_CGM [6] | Y | N | Run | No |
| MC_ME | Y | Y | Run | No |
| PLL | Y | Y | Run | No |
| Configuration GPR | Y | Y | Run | No |
| eMIOS 0-2 | Y | Y | Run | No |
| PIT_0 | Y | Y | Standby[7] | No |
| PIT_[1:2] | Y | Y | Run | No |
| FlexCAN_[0:5] | Y | Y | Run | No |
| FlexIO | Y | Y | Run | No |
| LPUART_[0:15] | Y | Y | Run | No |
| LPI2C_[0:1] | Y | Y | Run | No |
| I3C | Y | Y | Run | No |
| LPSPI_[0:5] | Y | Y | Run | No |
| QuadSPI | Y | Y | Run | No |
| SAI_[0:1] | Y | Y | Run | No |
| ADC_[0:2] | Y | Y[10] | Run | No |
| LPCMP_[0:2] | Y | Y | Standby | No |
| TempSense | Y | Y[10] | Run | No |
| CRC | Y | Y | Run | Yes |
| FCCU (+FOSU) | Y | N | Run | Yes |
| STCU2 | Y | N | Run | No |
| ELE_HSEB MUA-MUB | Y | Y | Run | No |

*Table continues on the next page...*

Table 161. Module reset status (continued)

| Module instances[1] | Destructive | Functional | Power domain[2] | Part of LBIST[3] |
|---|---|---|---|---|
| JDC | Y[8] | Y[8] | Run | No |
| DMAMUX_[0:1] | Y | Y | Run | No |
| PMC | Y[9] | N | Standby | No |
| Flash memory | Y | Y[10] | Run | No |
| SIRC | Y[5] | Y[5] | Standby | No |
| SXOSC | Y[11] | N | Standby | No |
| BCTU | Y | Y | Run | No |
| LCU[0:1] | Y | Y | Run | No |
| RTC | Y | N[12] | Standby | No |
| EMAC | Y | Y | Run | No |
| ELE_HSEB | Y | Y | Run | No |
| SWT_0 | Y | Y | Standby | No |
| SWT_1 | Y | Y | Run | Yes |
| STM_0 | Y | Y | Run | No |
| STM_1 | Y | Y | Run | No |
| MSCM | Y | Y | Run | No |
| ERM | Y | Y | Run | Yes |
| EIM | Y | Y | Run | Yes |
| eDMA | Y | Y | Run | Yes |
| JTAGC | N | N | Run | No |
| MDM_AP | Y | N | Run | No |
| APB_AP | Y | N | Run | No |
| Cortex-M7_0 | Y | Y[13] | Run | No |
| Cortex-M7_0 AHB-AP | Y | N | Run | No |
| MC_PCU | Y | N | Standby | No |

| Legends: | |
|---|---|
| Y | The entire module resets on this particular reset. |
| Y | Only a portion of the module resets on this particular reset. |
| N | No portion of the module resets on this particular reset |

1. All the modules listed in this table are reset on a POR event. See the memory map file attached to this document for the availability of modules across various parts in the MCXE31 family.
2. The modules in the RUN domain get reset on standby exit. The modules in STANDBY domain are not impacted by standby exit and retain their contents. However in case of standby exit via functional reset or destructive reset event, the corresponding flops within the STANDBY domain modules will also get reset.
3. The modules in the LBIST logic get reset on selftest completion.
4. Flash memory scanning logic is available in the Run domain. GPRs and LC decode logic are available in Standby domain.

5. All memory-mapped registers are on functional reset. The trimming logic is on destructive reset. Rest of counter and other stuff is on POR.
6. During functional reset stages FUNC2 and FUNC3 (see the Functional reset sequence descriptions for functional reset stage descriptions), MC_CGM.MUX_*n*_CSC and MC_CGM.MUX_*n*_DIV_*m* are automatically set to their default values. The default value of the MC_CGM.MUX_*n*_CSC[SELCTL] selects FIRC_CLK as the source clock for all multiplexers. The default value of the MC_CGM.MUX_*n*_DIV_*m* is register instance specific (see the "MC_CGM register descriptions" section in the Clock Generation Module (MC_CGM) chapter).
7. Only PIT_0 supports the RTI feature, and exists in the Standby domain.
8. The system domain is reset on functional reset. A POR will reset it completely.
9. PMC registers are reset on a destructive reset except PMC.LVSC, which is reset only by own PMC.LVSC[PORF] flag. The LVR and POR logic is reset on an LVR or own PORF (see PMC.CONFIG and PMC.LVSC descriptions for details).
10. Reset on a functional reset; however, reset recovery occurs before the chip functional reset recovery, at the functional reset exit sequence start, for proper trim scanning and loading.
11. SXOSC is reset on a destructive reset so that the RTC operates properly across a functional reset.
12. RTC operates during a functional reset.
13. The functional reset maps to nSYSRESET to Arm Cortex-M7 and the destructive reset maps to nPORESET to Arm Cortex-M7. See the Cortex-M7 TRM for further description on part of logic on different domains within Cortex-M7.

## 30.8 System RAM retention across functional reset

System RAM retains content during functional reset through the crossbar halt handshake. The system crossbar halts during the functional reset entry sequence. Therefore, the accesses do not cause any content corruption (see Functional reset sequence for more information).

Follow this sequence for the crossbar halt handshake (see Halt handshake using the daisy-chaining method for gasket locations):

1. Send a halt request to the ELE_HSEB AXBS, DMA AXBS, and EMAC IAHB bridges in parallel.
2. Wait for a halt acknowledgement from HSE_AXBS and DMA AXBS.
3. Send a halt request to the DMA IAHB and ELE_HSEB IAHB gaskets.
4. Wait for a halt acknowledgement from all the gaskets listed in the aforementioned steps.
5. Send a halt request to the system AXBS.
6. Wait for a halt acknowledgement from the system AXBS.
7. Send a halt request to a peripheral AXBS.
8. Wait for a halt acknowledgement from a peripheral AXBS.
9. Send a halt request to the TCM IAHB and QSPI IAHB gaskets.
10. Wait for a halt acknowledgement from all the gaskets listed in the aforementioned steps.
11. Send a halt request to the AIPS0 IAHB and AIPS1 IAHB gaskets.
12. Wait for a acknowledgement from all of the gaskets listed in the aforementioned steps.
13. Send a halt request to TCM AXBS.
14. Wait for a halt acknowledgement from TCM AXBS.

After this halt sequence completes, the crossbar halt acknowledgement sequence also completes and the chip proceeds to the FUNC1 stage in the functional reset entry sequence.

**NOTE**

RAM retention is supported across the functional reset event for system RAMs only and not for ELE_HSEB or peripheral memories.

### 30.8.1 Halt handshake using the daisy-chaining method

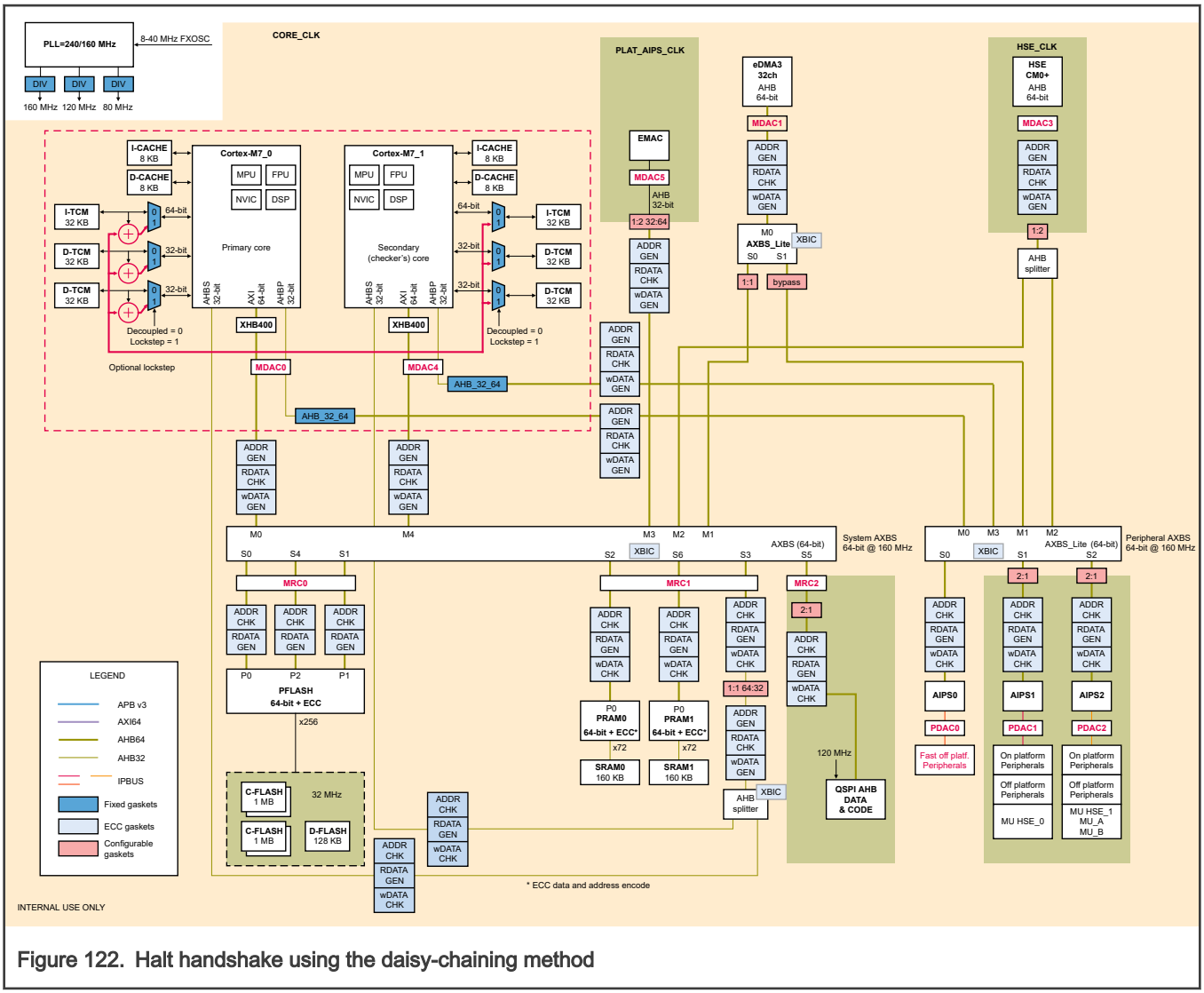



**Figure 122. Halt handshake using the daisy-chaining method**

## 30.9 Pad state during reset and after reset

SIUL2 controls the GPIO functionality. It sets to its default state on a functional reset, and ensures that every pad initializes to its default state (see the default configurations and reset states of the chip's GPIO in the IOMUX file attached to this document).

## 30.10 Reset pin

This chip contains a bidirectional reset pin (RESET_b) indicating the reset state. RESET_b multiplexes with the other functions on port PTA5 (see the IOMUX file attached to this document).

The DCF configuration controls the multiplexing capabilities of RESET_b. The default configuration of the RESET_b port is that of a dedicated reset signal (for example, not multiplexed. See the DCF clients file attached to this document for more information).

The RESET_b pin offers the following uses if you configure it for the reset functionality:

- Acts as an external destructive reset source
- Acts as an indicator for the chip reset sequence for both functional and destructive reset sequences

**NOTE**

MC_RGM.FES[F_EXR] captures an externally sourced RESET_b assertion for a destructive reset.

The RESET_b pin also indicates an internally asserted reset to external modules. It has a weak internal pullup. In normal Run mode, it keeps the chip out of reset.

### 30.10.1 Reset pin control during self-test

You can write 1 to MC_RGM.ERCTRL[ERASSERT] to assert RESET_b (writes only in Supervisor mode), before LBIST or MBIST executes. MC_RGM then asserts RESET_b and tristates the GPIO pins placing them in a safe state. Tristating GPIO ensures a safe state for the chip pins when LBIST or MBIST executes. Following BIST, the chip executes a reset sequence. The chip configures again for the application software, before executing the safety function. MC_RGM.ERCTRL[ERASSERT] clears on a functional reset. See Figure 8 for an illustration.

**NOTE**

Writing 1 to SIUL2.MSCR*n*[SMC] enables the GPIO pins and the chip continues its normal I/O functionality.

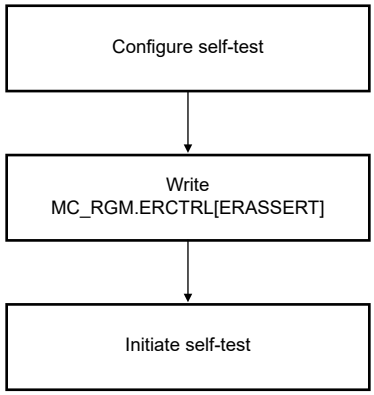



**Figure 123. Reset pin control before self-test**

Multiple chip configuration scenarios cause RESET_b to react differently after self-test completion:

- You can write 1 to MC_RGM.ERCTRL[ERASSERT] causing RESET_b to assert. This assertion does not impact the reset sequence, and the reset indicates that the chip is not available in Functional mode (although the chip is not in reset sequence).

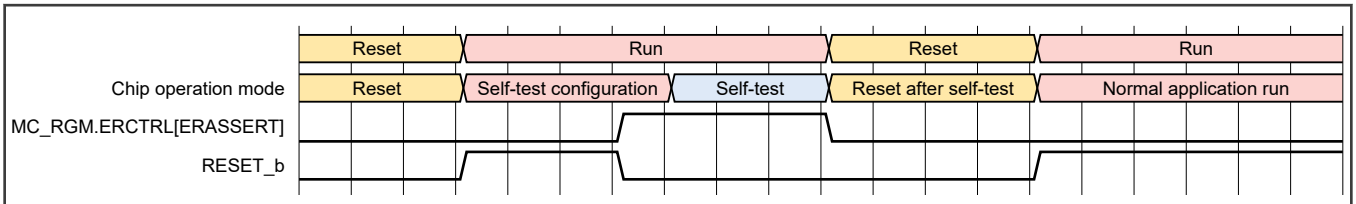



**Figure 124. Reaction when MC_RGM.ERCTRL[ERASSERT] = 1**

- If MC_RGM.ERCTRL[ERASSERT] = 0 (the default value), the RESET_b pin goes low after self-test completes. After self-test, the chip undergoes a functional reset in which the chip hardware writes 0 to MC_RGM.ERCTRL[ERASSERT]. The RESET_b pin goes high after the reset deasserts.

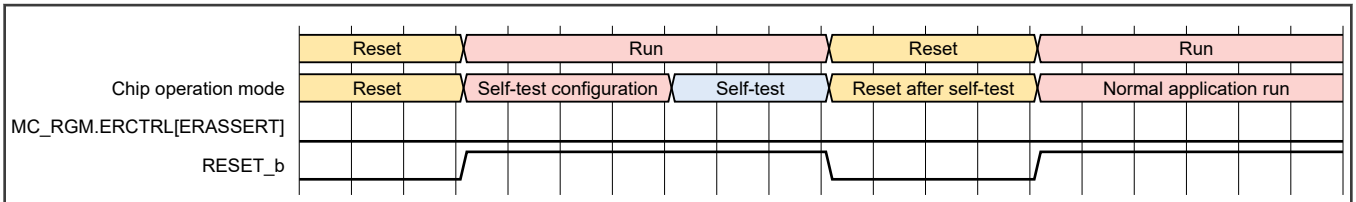



**Figure 125. Reaction when MC_RGM.ERCTRL[ERASSERT] = 0**

- If MC_RGM.ERCTRL[ERASSERT] = 0 (the default value), but you write 1 to MC_RGM.FBRE[ST_DONE], the RESET_b pin does not assert after self-test completes.

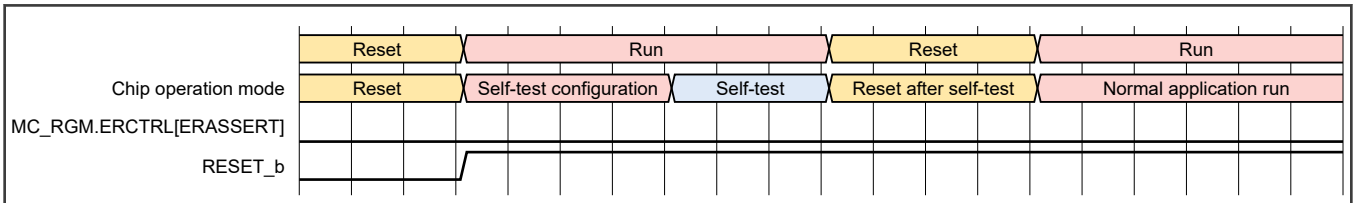



**Figure 126. Reaction when MC_RGM.ERCTRL[ERASSERT] = 0 and MC_RGM.FBRE[ST_DONE] = 1**

## 30.11 Application core debug from first instruction

The chip supports application core debug starting on the first instruction. This is supported on any functional reset and through a RESET_b pin destructive reset for debug from destructive reset.

### 30.11.1 Debug from first instruction on power-up or destructive reset onwards

1. When RESET_b is held low, the chip goes through a destructive reset sequence.
2. When you pull down the RESET_b pin, the chip goes through a destructive reset sequence, exits from the destructive reset, and remains in functional reset, until the reset pin is released.
3. When the RESET_b pin remains pulled down, the debugger configures SDA_AP.SDAAPRSTCTRL[RSTRELTLCM70] and SDA_AP.SDAAPRSTCTRL[RSTRELTLCM71] according to the requirements. To enable debug from first instruction, you write 0 to these fields. To disable, you write 1. By default, the debug disables from the first instruction.
4. On releasing the reset pin, the chip exits from the functional reset sequence and ELE_HSEB starts executing.
5. The PORESETn deassertion occurs for the application cores simultaneously (in case of lockstep, Cortex-M7_1 PORESETn deasserts after a two-cycle delay). The corresponding core clock enable gates the deassertion of SYSRESETn to cores. The SYSRESETn to the cores deasserts as follows.
6. On the basis of whether the debugger authentication is performed via challenge, response, or neither of them, you can consider these scenarios:
    a. Debugger not connected (DBG_PWRUP_REQ = 0):
        i. BAF or FW writes to the clock enable control field in MC_ME for application cores, that is, to MC_ME.PRTN0_CORE0_PCONF[CCE] for Cortex-M7_0, MC_ME.PRTN0_CORE1_PCONF[CCE] for Cortex-M7_1, or both the fields as required.
        ii. The clock enables in the subsequent clock cycle.
        iii. SYSRESETn deasserts after ten clock cycles (adding eight clock cycles of delay ensures that reset always deasserts when the clock is available; two cycle of reset deassertion delay). In case of lockstep, Cortex-M7_1 reset is delayed by two additional clock cycles.
    b. Debugger connected (DBG_PWRUP_REQ = 1):

i. BAF or FW writes to the clock enable control field in MC_ME for application cores, that is, to MC_ME.PRTN0_CORE0_PCONF[CCE] for Cortex-M7_0,MC_ME.PRTN0_CORE1_PCONF[CCE] for Cortex-M7_1 or to both the fields as required.

ii. The clock is enabled.

iii. The debugger waits for sometime (at least ten clock cycles) and then writes 1 to SDA_AP.SDAAPRSTCTRL[RSTRELTLCM70] for Cortex-M7_0 and to SDA_AP.SDAAPRSTCTRL[RSTRELTLCM71] for Cortex-M7_1 for releasing SYSRESETn of the corresponding core.

iv. SYSRESETn deasserts after ten clock cycles (adding eight clock cycles of delay ensures that the reset always deaserts when the clock is available; two cycle of RSL delay). In case of lockstep, the Cortex-M7_1 reset is delayed by two additional clock cycles.

**NOTE**

Cortex-M7_1 is not available in MCXE315/MCXE316/MCXE317/MCXE31B.

### 30.11.2 Debug from first instruction on functional reset onwards

If you enable debug, on functional reset, the debug connection is retained. That is because the complete debug infrastructure is on destructive reset domain.

### 30.11.3 Debug on standby exit by wake-up or functional reset

If you enable debug before standby entry, then the Standby domain stores the final status. The configuration for reset release is reset (SDA_AP.SDAAPRSTCTRL[RSTRELTLCM70] and SDA_AP.SDAAPRSTCTRL[RSTRELTLCM71] the reset value of both the fields is 1), which defaults to debug from the first instruction disabled.

You must configure the reset release on standby exit. It uses MC_RGM's low-power debug protocol before ungating the reset deassertion.

## 30.12 Glossary

| | |
|---|---|
| **DCF** | Device configuration format |
| **POR** | Power-on reset |

# Chapter 31
# Boot Overview

## 31.1 Overview

### 31.1.1 Introduction

This chapter describes the system boot sequence and provides details of boot options. The only CPU available when the hardware reset sequence completes is in the HSE subsystem hereafter referred to as HSE CPU.

The HSE CPU starts executing firmware code in HSE code flash from fixed location that contains the Secure BAF code . Secure BAF provides the boot sequence until control is passed as determined by the type of boot:

- Non-Secure boot - To customer software that executes on one or all the application cores.
- Secure boot - To the HSE FW running on HSE CPU.

## 31.2 Secure BAF features

The features of Secure BAF are as follows:

- Secure and Non-secure boot modes
- Application boot core selection
- Chip LC advancement
- Debug authorization
- XRDC configuration

## 31.3 Chip Configuration

This section describes the chip configuration details for MCXE31 variants when Secure BAF is programmed and HSE Firmware feature flag is disabled.

If HSE firmware feature flag is enabled see the HSE Firmware Reference manual.

### 31.3.1 Memory map

This section explains the memory sections used by the Secure BAF in various configurations.

#### 31.3.1.1 Memory map when HSE firmware usage feature flag is disabled

**Table 162. Memory map when HSE firmware usage feature flag is disabled**

| Memory Section | MCXE315/MCXE316 | MCXE317 | MCXE31B |
|---|---|---|---|
| Flash memory | 1 MB | 2 MB | 4 MB |
| IVT<br>Locations in Priority order | 0040_0000h,<br>0048_0000h,<br>1000_0000h | 0040_0000h,<br>0050_0000h,<br>1000_0000h | 0040_0000h,<br>0050_0000h,<br>0060_0000h,<br>0070_0000h<br>1000_0000h |

*Table continues on the next page...*

Table 162. Memory map when HSE firmware usage feature flag is disabled (continued)

| Memory Section | MCXE315/MCXE316 | MCXE317 | MCXE31B |
|---|---|---|---|
| Flash memory | 1 MB | 2 MB | 4 MB |
| Secure BAF Boot Location | 004F_FC00h | 005F_FC00h | 007F_FC00h |
| Secure BAF code region | 004F_4000h to 004F_FFFFh (48 KB) | 005F_4000h to 005F_FFFFh (48 KB) | 007F_4000h to 007F_FFFFh (48 KB) |
| Application Flash area | 00400_000h to 004F_3FFFh (976 KB) | 00400_000h to 005F_3FFFh (2000 KB) | 0040_0000h to 007F_3FFFh (4048 KB) |
| Application Data Flash | 1000_0000h to 1000_FFFFh (64 KB) | 1000_0000h to 1001_FFFFh (128 KB) | |
| Secure BAF UTEST flash | 1B00_0200h to 1B00_06FFh (1280 Bytes) | | |

#### 31.3.1.2 Memory map when HSE firmware usage feature flag is enabled

Following table explains the memory sections used by the Secure BAF in case HSE firmware feature flag is enabled in UTEST. Below tables are applicable for standard firmware in which HSE Firmware size is 128 KB. For custom firmware, the secure size depends on size of custom HSE Firmware. This table is provided for the information to the user that how much memory will be used by security firmware in case security feature is enabled.

Table 163. Memory map when full memory HSE firmware is installed

| Memory Section | MCXE315/MCXE316 | MCXE317 | MCXE31B |
|---|---|---|---|
| Flash memory | 1 MB | 2 MB | 4 MB |
| IVT Locations in Priority order | 0040_0000h, 0048_0000h, 1000_0000h | 0040_0000h, 0050_0000h, 1000_0000h | 0040_0000h, 0050_0000h, 0060_0000h, 0070_0000h 1000_0000h |
| Secure BAF Boot Location | 004F_FC00h | 005F_FC00h | 007F_FC00h |
| Secure BAF code region | 004F_4000h to 004F_FFFFh (48 KB) | 005F4_000h to 005F_FFFFh (48 KB) | 007F_4000h to 007F_FFFFh (48 KB) |

*Table continues on the next page...*

Table 163. Memory map when full memory HSE firmware is installed (continued)

| Memory Section | MCXE315/MCXE316 | MCXE317 | MCXE31B |
|---|---|---|---|
| Flash memory | 1 MB | 2 MB | 4 MB |
| Full memory HSE Firmware code area in last block | 004D_4000h to 004F_3FFFh (128 KB) | 005D_4000h to 005F_3FFFh (128 KB) | 007D_4000h to 007F_3FFFh (128 KB) |
| Application Flash area when Full memory HSE Firmware present | 0040_0000h To 004D_3FFFh (848 KB) | 0040_0000h to 005D_3FFFh (1824 KB) | 0040_0000h to 007D_3FFFh (3920 KB) |
| Application Data Flash | 1000_0000h to 1000_FFFFh (64 KB) | 1000_0000h to 1001_5FFFh (88 KB) | |
| Secure BAF UTEST Flash | 1B00_0200h to 1B00_06FFh (1280 Bytes) | | |

Table 164. Memory map in AB swap configuration

| Memory Section | MCXE315/MCXE316 | MCXE317 | MCXE31B |
|---|---|---|---|
| Flash memory | 1 MB | 2 MB | 4 MB |
| IVT locations in priority order | 0040_0000h, 1000_0000h | 0040_0000h, 1000_0000h | 0040_0000h, 0050_0000h, 1000_0000h |
| HSE FW boot location | 0047_FC00h | 004F_FC00h | 005F_FC00h |
| AB Swap HSE firmware code area in active block [1] | 0045_4000h to 0047_FFFFh (176 KB) | 004D_4000h to 004F_FFFFh (176 KB) | 005D_4000h to 005F_FFFFh (176 KB) |
| AB Swap HSE firmware code area in passive block[1] | 004D_4000h to 004F_FFFFh (176 KB) | 005D_4000h to 005F_FFFFh (176 KB) | 007D_4000h to 007F_FFFFh (176 KB) |
| Application Flash area in active block | 0040_0000h to 0045_3FFFh (336 KB) | 0040_0000h to 004D_3FFFh (848 KB) | 0040_0000h to 005D_3FFFh (1872 KB) |
| Application Flash area in passive block | 00480_000h to 004_D3FFFh (336 KB) | 0050_0000h to 005D_3FFFh (848 KB) | 0060_0000h to 007D_3FFFh (1872 KB) |
| Application Data Flash | 1000_0000h to | 1000_0000h to 1001_FFFFh (128 KB) | |

*Table continues on the next page...*

Table 164. Memory map in AB swap configuration (continued)

| Memory Section | MCXE315/MCXE316 | MCXE317 | MCXE31B |
|---|---|---|---|
| Flash memory | 1 MB | 2 MB | 4 MB |
| | 1000_FFFFh (64 KB) | | |
| Secure BAF UTEST Flash | 1B00_0200h to 1B00_06FFh (1280 Bytes) | | |

1. HSE firmware includes the Secure BAF in AB swap configuration

### 31.3.2 Hardware protection

Platform flash of MCXE31 has facilitated hardware protection exclusive to HSE core. BAF releases hardware protection of that area which is being used by Application as mentioned in previous section. If the hardware protection fails, for security purpose, Secure BAF issues a reset.

Below table explains default values of Hardware protection.

Table 165. Hardware protection by Secure BAF

| Register name | Register address | Secure memory | Configured value |
|---|---|---|---|
| CONFIG_RAMPR | 4039_C038h | SRAM memory | 4000_0000h |
| CONFIG_CFPRL | 4039_C03Ch | Code flash memory active block | 4000_0000h |
| CONFIG_CFPRH | 4039_C040h | Code flash memory passive block | 4000_C000h |
| CONFIG_DFPR | 4039_C044h | Data flash memory | 4002_0000h |

## 31.4 Device common configuration

### 31.4.1 Feature configuration in virgin device

Following table lists down the status of various features when device "LC = CUST_DEL". Many features are configurable directly or indirectly by application software.

| Feature | Configuration information | Status | Configurability |
|---|---|---|---|
| OTA functionality | Samples are always shipped in the OTA disabled configuration | Disabled | Yes<br>Can be enabled by application by requesting to HSE firmware or Secure BAF |
| HSE firmware usage feature flag | This flag indicates whether firmware installation allowed in the device or not. By default, this flag | Disabled | Yes |

*Table continues on the next page...*

*Table continued from the previous page...*

| Feature | Configuration information | Status | Configurability |
|---|---|---|---|
| | is unprogrammed and Secure BAF assumes that firmware installation is not allowed. | | Can be enabled by programming in the UTEST location. More details on “UTEST Flag description” section. |
| Secure BAF firmware | Samples are always shipped with Secure BAF programmed | Programmed | No |
| HSE Firmware | HSE firmware is always programmed by user/ Encrypted and signed firmware image is always<br>delivered to user. | Not programmed | Yes<br>Can be installed by the application software. |
| Image Vector Table | No IVT is programmed at any of the IVT locations. | Not programmed | Yes<br>Can be programmed by application software. |
| SWT0 | Application core watchdog SWT0 is disabled. | Disabled | Yes<br>It can be optionally enabled by BAF if SWT bit is configured in boot configuration word. |
| Boot Sequence | Boot sequence is non-secure boot<br>i.e. application are booted by Secure BAF without any authentication. | Non secure boot | Yes<br>It can be changed to secure boot by programming the BOOT_SEQ bit in IVT. |
| Life Cycle | Samples are always delivered in LC=CUST_DEL | Customer Delivery | Yes<br>It can be changed to OEM_PROD and IN_FIELD by application through Secure BAF and HSE firmware. |
| Application Core enablement status | All applications cores are in disable state. | Disabled. | Yes<br>Single or all application cores can be enabled at required address by programming the required fields in the IVT. |
| Clock Source | Fast Internal Reference clock | FIRC | Yes |

*Table continues on the next page...*

*Table continued from the previous page...*

| Feature | Configuration information | Status | Configurability |
|---|---|---|---|
| FIRC Frequency Value | 48 MHz | 48 MHz | Yes, application can configures after Secure BAF goes in WFI. |
| Application debug authorization mode | Debug Authorization mode of application cores is password based. It can be changed to challenge response mode by programming the configuration in UTEST. | Password based approach | When HSE Firmware Feature flag is disabled, debug authorization mode cannot be changed.<br>When HSE Firmware Feature flag is enabled, debug authorization mode can be changed to challenge response mode by requesting to HSE<br>firmware. |
| Application core debug status | Debug of application cores is enabled in customer delivery lifecycle. | Enabled | No |

### 31.4.2 Secure BAF memory map description

Following table explains the memory sections used by the Secure BAF.

**Table 166. Secure BAF memory**

| Memory Section | Remarks |
|---|---|
| IVT location | This is the base address of IVT. The size of IVT is always 256 byte starting from base address. IVT at one location must be programmed by the application. |
| Secure BAF boot location | Location where interrupt vector table of HSE core is present. Secure BAF on HSE CPU boots from this location. From this address 1024 bytes of interrupt vector table is present.<br>This location is protected by XRDC and it can be accessed only by HSE core. |
| Secure BAF code region | Secure BAF code area in flash. This area is protected by XRDC and it can be accessed only by HSE core. |
| UTEST flash location | 1280 Kbytes of UTEST area of flash. |

### 31.4.3 UTEST memory usage by Secure BAF

**Table 167. UTEST memory location usage by Secure BAF**

| Start address | End address | Size (bytes) | Description | Programmed by | Write protected |
|---|---|---|---|---|---|
| 1B00_0000h | 1B00_0007h | 8 | HSE Firmware Feature Usage | Application | No |

*Table continues on the next page...*

Table 167. UTEST memory location usage by Secure BAF (continued)

| Start address | End address | Size (bytes) | Description | Programmed by | Write protected |
|---|---|---|---|---|---|
| | | | flag. For more detail, see UTEST flag description | | |
| 1B00_0040h | 1B00_0047h | 8 | Unique Chip Identifier (UID) 0 | NXP | No |
| 1B00_0050h | 1B00_0057h | 8 | FXOSC enablement flag<br>For more detail, UTEST flag description | Application | No |
| 1B00_0080h | 1B00_009Fh | 32 | Debug password(CUST_DB_PSWD_A).<br>When HSE FW feature Flag is disabled, this location is used by Secure BAF to run the debug authorization feature. Secure BAF copies this value in application expected response register and this value is used to derive the HSE expected response register. Size of this register is 16B and 1B00_0090h – 1B00_009h is reserved.<br>When HSE Firmware feature flag is enabled, password is programmed by HSE firmware at different location.<br>Password is scanned by DCM during Reset only. | Application | No |

*Table continues on the next page...*

Table 167. UTEST memory location usage by Secure BAF (continued)

| Start address | End address | Size (bytes) | Description | Programmed by | Write protected |
|---|---|---|---|---|---|
| | | | Password to be retained in standby. | | |
| 1B00_0220h | 1B00_022Fh | 16 | Lifecycle slot 1: CUST_DEL | NXP | Yes |
| 1B00_0230h | 1B00_023Fh | 16 | Lifecycle slot 2: OEM_PROD | Secure BAF | Yes |
| 1B00_0240h | 1B00_024Fh | 16 | Lifecycle slot 3: IN_FEILD | Secure BAF | Yes |
| 1B00_0250h | 1B00_025Fh | 16 | Lifecycle slot 4: Pre-FA | Secure BAF | Yes |
| 1B00_0260h | 1B00_026Fh | 16 | Lifecycle slot 5: FA | Secure BAF | Yes |

Detail description of each field is out of scope of this document. For details, see MCXE31_DCF_clients file attached with this document.

## 31.4.4 UTEST flag description

### 31.4.4.1 HSE firmware usage feature flag

This flag is an indication to Secure BAF that application intend to use HSE firmware on the device. By default, this flag is unprogrammed and Secure BAF assumes that HSE firmware installation is not allowed in the secure samples. But, if application decides to allow installation of HSE firmware, it can program the HSE Firmware Usage Feature flag in the UTEST at location 0x1B000000.

Description of the flag as follows:

Table 168. HSE firmware usage feature flag description

| Field type | Description | Remarks |
|---|---|---|
| Size | 64 bits | |
| Default value | 0xFFFFFFFFFFFFFFFF | |
| Default status | Disabled | Secure BAF does not allows installation of HSE firmware. |
| UTEST location | 0x1B000000 | |
| Programmed by | Application software | |
| LC in which programming is allowed | All LC | |
| Value to program to to enable HSE FW installation | Any value other than default value | Secure BAF allows installation of HSE firmware. |

Implication when this flag is not programmed by application

- Secure BAF never try to install the HSE firmware.
- Secure BAF does not protect the HSE firmware code flash area.
- The logic of debug authorization protocol is different. See Debug authorization for more details.
- Secure BAF does not protect the data flash area reserved for HSE firmware.
- Debug authorization always be password based for application cores.
- Secure BAF always boot the application without any authentication. BOOT_SEQ bit in the boot configuration word is ignored.
- Secure BAF does not authenticate the IVT and XRDC configuration table in all LC.
- Application password area in UTEST area is not read and write protected by XRDC in LC = CUST_DEL. In other LC, it is accessible only to HSE core.
- Number of region descriptors in MRC0 and MRC1 increased which can be configured by the application.
- Default configuration of XRDC and valid address range changed. For more details see XRDC configuration.

#### 31.4.4.2 Crystal oscillator configuration flag

Description of the flag as follows:

| Field type | Description |
|---|---|
| Size | 64 bits |
| Default Value | FFFF_FFFF_FFFF_FFFFh |
| Default Status | Boot via FIRC |
| UTEST Location | 1B00_0050h |
| Programmed by | Application |
| LC in which programming is allowed | All LC |

**Table 169. Crystal oscillator configuration flag in UTEST**

| | 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FXOSC_ENABLE_MAGIC_NUMBER | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
| R | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | GMSEL | | | | EOCV | | | | | | | | | | | |

*Table continues on the next page...*

*Table continued from the previous page...*

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | | | | | | | | | | | | | | | | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R | CRYSTAL_OSCILLATOR_FREQUENCY | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |

**Table 170. Crystal oscillator configuration bit definition**

| Field | Description |
|---|---|
| 63-33 | FXOSC_ENABLE_MAGIC_NUMBER<br>AAAA_5555h – Enable<br>FFFF_FFFFh – Disable |
| 31 - 28 | GMSEL (Crystal overdrive protection):<br>For invalid values = 0x0 or 0xF, default value 0xC is used. |
| 27 - 20 | EOCV (End of count value):<br>For invalid value = 0x0, default value 9Dh is used. |
| 19 - 16 | Reserved |
| 15 - 0 | CRYSTAL_OSCILLATOR_FREQUENCY:<br>Crystal frequency in KHz. Valid Crystal frequency range is 8000 – 40000 KHz. BAF does not checks for invalid frequencies. This field is only necessary if HSE Firmware installation is required. |

## 31.5 Image Vector Table

The following section describes the various fields in Image Vector Table that is to be programmed by the application. IVT is scanned by Secure BAF after the device is out of reset. The structure is 256 bytes in size. This structure contains application start address, XRDC configuration etc. It must be programmed at least at one of the locations as mentioned in Chip Configuration

After reset, Secure BAF searches for the first valid IVT starting from the lowest address. If there are multiple valid IVT at IVT locations at the same time, the lowest address wins the priority.

**Table 171. Image Vector Table**

| Address offset | Size in bytes | Content | Comments |
|---|---|---|---|
| 00h | 4 | Image Vector Table Marker | It is a magic number which marks the starting of Image Vector Table location. Its value must be 5AA5_5AA5h |
| 04h | 4 | Boot Configuration Word | Configuration word that allows the user to select the various configuration in which device can be booted. Details are mentioned in the next section. |

*Table continues on the next page...*

Table 171. Image Vector Table (continued)

| Address offset | Size in bytes | Content | Comments |
|---|---|---|---|
| 08h | 4 | Reserved | |
| 0Ch | 4 | CM7_0<br>application<br>Start Address | Boot address of CM7_0 application in code flash area. It must honor core VTOR register alignment restrictions. This field is used by BAF when BOOT_SEQ bit is 0. |
| 10h | 4 | Reserved | |
| 14h | 4 | CM7_1<br>application<br>Start Address | Boot address of CM7_1 application in code flash area. It must honor core VTOR register alignment restrictions. This field is used by BAF only when BOOT_SEQ bit is 0.<br>This filed is ignored if Lockstep configuration is enabled.<br>Reserved for MCXE315/MCXE316/MCXE317/MCXE31B |
| 18h | 4 | Reserved | |
| 1Ch | 4 | CM7_2<br>application<br>Start Address | Boot address of CM7_2 application in code flash area. It must honor core VTOR register alignment restrictions. This field is used by BAF only when BOOT_SEQ bit is 0<br>Reserved for MCXE315/MCXE316/MCXE317/MCXE31B. |
| 20h | 4 | Reserved | |
| 24h | 4 | Address of<br>LC Configuration | Address of Configuration word that allows User to Advance LC. Details are mentioned below. |
| 28h | 216 | Reserved | |

### 31.5.1 Boot configuration word

This register informs Secure BAF to allow booting of selected applications.

Table 172. Boot configuration register

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

*Table continues on the next page...*

Table 172. Boot configuration register (continued)

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | | | | | | | | | APP_SWT_INIT | RESERVED | BOOT_SEQ | CM7_2_ENABLE | CM7_1_ENABLE | CM7_0_ENABLE |
| W | | | | | | | | | | | | | | | | |

Table 173. Boot configuration register bit definition

| Field | Description |
|---|---|
| 31 – 6 | Reserved |
| 5 | APP_SWT_INIT: Control SWT0 enablement before control is passed to application core(s).<br>0 Disable<br>1 Enable<br>BAF initializes the SWT0 before enabling the application cores. BAF scans this bit only when BOOT_SEQ bit is 0. |
| 4 | RESERVED |
| 3 | BOOT_SEQ<br>Control the boot flow of the application.<br>0 Non-Secure Boot- Application image without any authentication is started by Secure BAF in parallel to HSE firmware.<br>1 Secure Boot- Application image is executed by HSE firmware after the authentication. Secure BAF only starts the HSE firmware after successful authentication.<br>(Only valid for HSE Firmware feature flag enabled) |
| 2 | CM7_2_ENABLE<br>0 Cortex-M7_2 application core clock gated after boot<br>1 Cortex-M7_2 application core clock un-gated after boot<br>Reserved for MCXE315/MCXE316/MCXE317/MCXE31B. |
| 1 | CM7_1_ENABLE<br>0 Cortex-M7_1 application core clock gated after boot<br>1 Cortex-M7_1 application core clock un-gated after boot<br>This field is ignored if Lockstep configuration is enabled. |
| 0 | CM7_0_ENABLE<br>0 Cortex-M7_0 application core clock gated after boot<br>1 Cortex-M7_0 application core clock un-gated after boot |

### 31.5.2 Address LC configuration word

This field allows User to Advance LC. To advance lifecycle User must program a valid 32 bits wide value at an address which must be given in IVT at address offset 24h. This address must be 4 bytes aligned and should not lie in Secure BAF/HSE FW protected areas.

Valid values for advancement in next lifecycles are mentioned in below table. For all other values at given address lifecycle advancement is not allowed.

**Table 174. Valid values for lifecycle advancement**

| Lifecyle | Valid values for lifecycle advancement |
|---|---|
| OEM_PROD | DADA_DADAh |
| IN_FIELD | BABA_BABAh |

Application password on the location depending on HSE FW Feature flag must be program before lifecycle advancement otherwise lifecycle advancement doesn't attempted by Secure BAF.

The SoC provides a LC mechanism for an irreversible progression of restrictions to accessing security related content on the device. The device LC cannot be reversed, so it is only possible to mature the device. Secure BAF takes on the responsibility of advancing the device through the LC:

- CUST_DEL --> OEM_PROD
- CUST_DEL and OEM_PROD --> IN_FIELD

Advancing the LC through Secure BAF involves configuring the LC configuration word in the IVT. Upon successful advancement, Secure BAF issues a Destructive reset. LC advancement is ignored if the device is found in same LC as indicated by the IVT.

### 31.5.3 Structure definition of Image Vector Table

Following structure can be used by application to configure the IVT configuration.

**NOTE**

The fields which are not valid for the specific configuration must be kept reserved. For Secure-Boot, see HSE Firmware Reference Manual.

```
typedef const struct image_vector_table
{
uint32_t Header; /*Header of IVT Structure */
uint32_t BootConfig; /*Boot Configuration Word */
const uint32_t Reserved1; /* Reserved */
const uint32_t * CM7_0_StartAddress; /*Start Address of Application on CM7_0 Core */
const uint32_t Reserved2; /* Reserved */
const uint32_t * CM7_1_StartAddress; /*Start Address of Application on CM7_1 Core */
const uint32_t Reserved3; /* Reserved */
const uint32_t * CM7_2_StartAddress; /*Start Address of Application on CM7_1 Core */
const uint32_t * Reserved4; /* Reserved */
const uint32_t * LCConfig; /*Address of LC configuration Word */
uint8_t Reserved5[216];
}ivt_t;
```

### 31.5.4 Invalid IVT scenarios

This section lists down the scenarios in which IVT is considered as invalid by Secure BAF.

1. IVT marker is not present.
2. ECC error is present in any of the 256 bytes of IVT memory.

## 31.6 Boot flow

Below diagram explains boot sequence flow of Secure BAF.

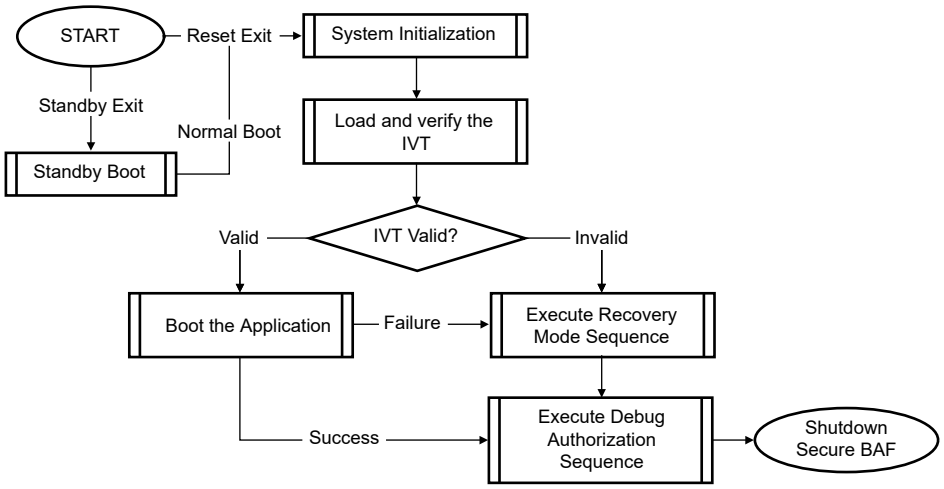



Figure 127. Boot flow

## 31.7 Standby boot

Secure BAF supports boot from Standby exit. SOC register DCMRWF5 (address 0x402AC610) provided to support boot on standby exit. This register gets clear on POR. For more details, see chapter Device Configuration Module (DCM) in MCXE31 Reference Manual.

Application is supposed to program this register before entering the standby mode.

There are two types of boot mode on exit from standby.

- Fast standby mode
- Normal boot on exit from standby

In Fast standby mode Secure BAF boots CM7_0 and halts the HSE core. The flow of standby boot is explained below:

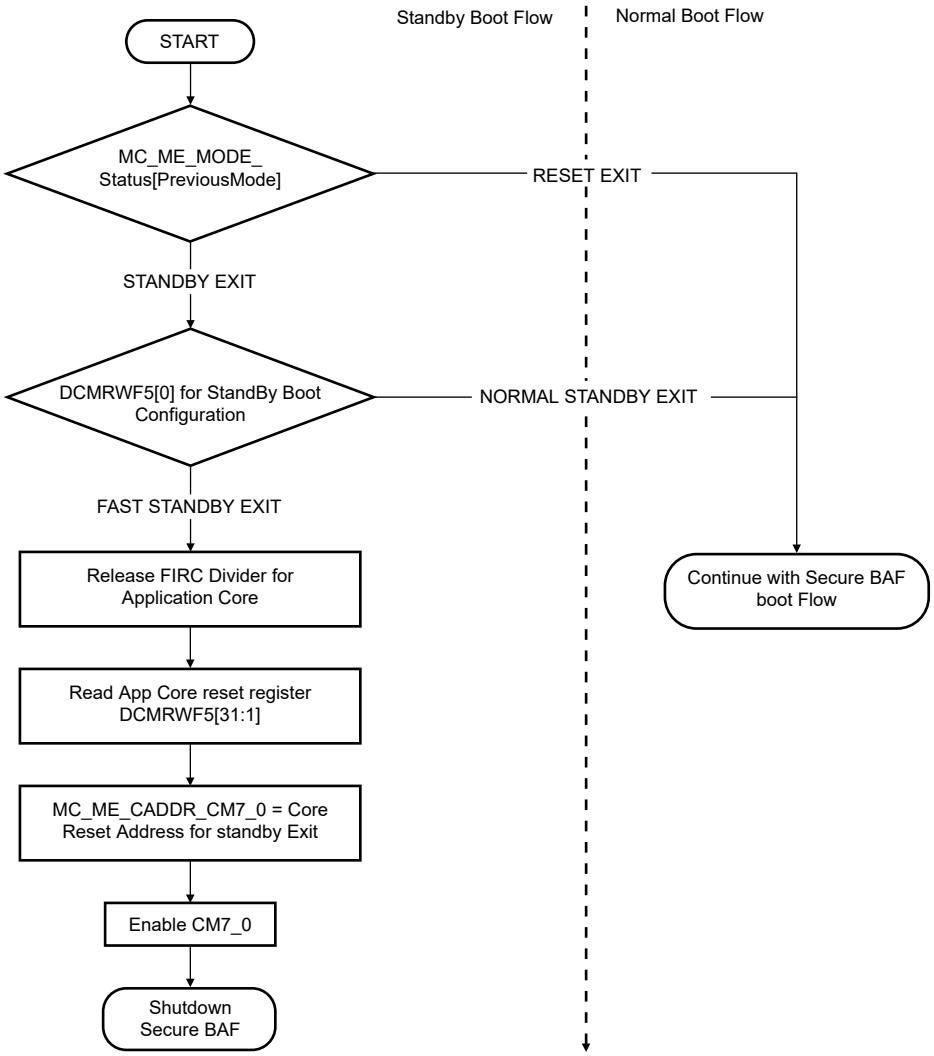



Figure 128. Standby boot flow

## 31.8 Reduced clock mode configuration

In case user wants to use clocking option B, application can enable this mode by setting “dcf_client_utest_misc” DCF record. For more details on DCF record, see the DCF sheet available with the MCXE31 Reference Manual.

After reset, Secure BAF checks for the HSE_CLK_MODE_OPTION field in DCMROF21 register in DCM. If this field is configured for clocking option B, Secure BAF configures the following dividers in MC CGM

- MUX_0_DC_1
- MUX_0_DC_2

Below figure explains steps for reduced clock mode configuration steps by Secure BAF.

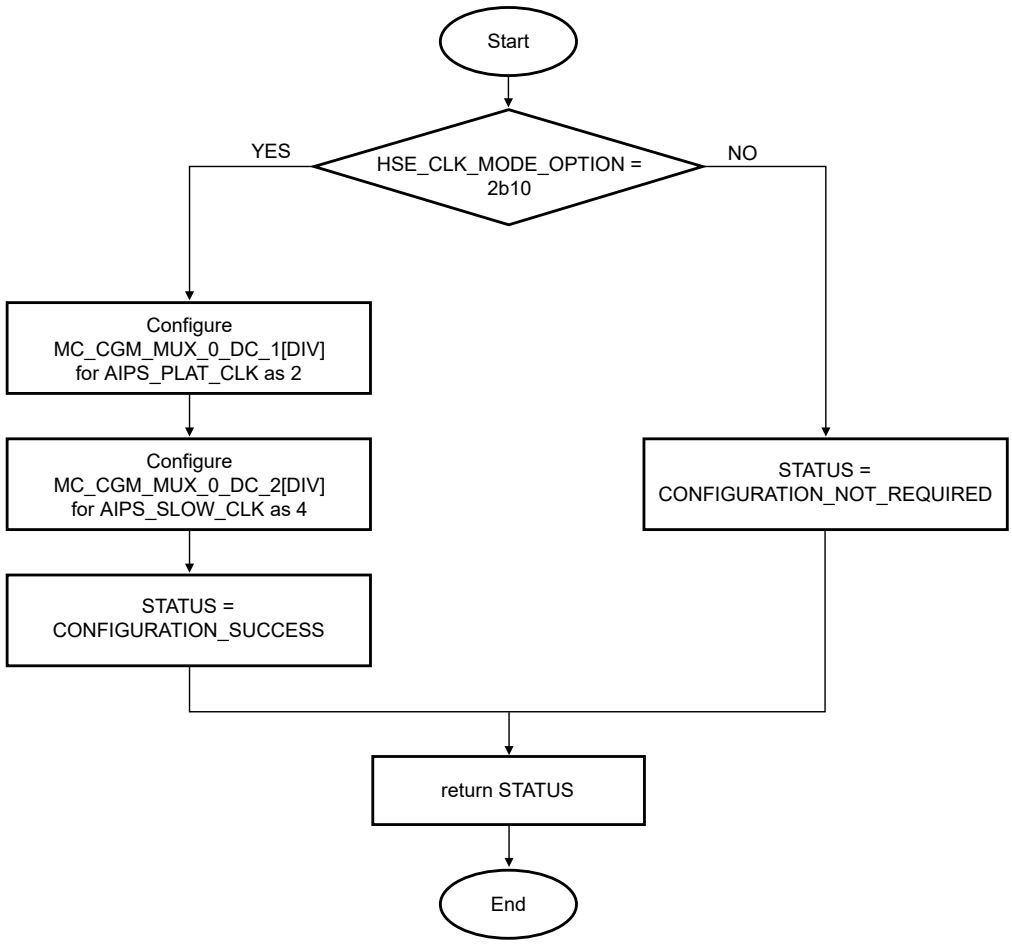



Figure 129. Reduced clock mode configuration

## 31.9 Debug authorization

If HSE firmware feature flag is disabled then application core debug is always password based. CUST_DB_PSWD_A must be programmed at location 1B00_0080h. See UTEST memory usage by Secure BAF more details.

## 31.10 FIRC divider register control

FIRC divider is controlled by Configuration HSE GPR Register (CONFIG_REG_GPR) address 0x4039C064. This HSE GPR register for FIRC divider is write protected by default and application can't modify it.

After Secure BAF executes WFI, it provides write access to FIRC divider field of this HSE GPR register and application can configure this register. Before access to this register, application should wait for Secure BAF to go in WFI by reading register PRTN0_CORE2_STAT.

## 31.11 Application boot

Secure-BAF is not responsible for booting application image in a secure way. The responsibility of booting application image in a secure way lies with HSE firmware. In case secure boot is not requested (by setting BOOT_SEQ bit in Boot configuration word in IVT structure to 0), the application image is always loaded by Secure-BAF without authentication in all LC of the device. By default, Secure-BAF releases the reset of every core which is configured to be enabled.

Following flow chart explains the steps of application boot performed by Secure-BAF when boot sequence field in boot configuration word is configured to 0.

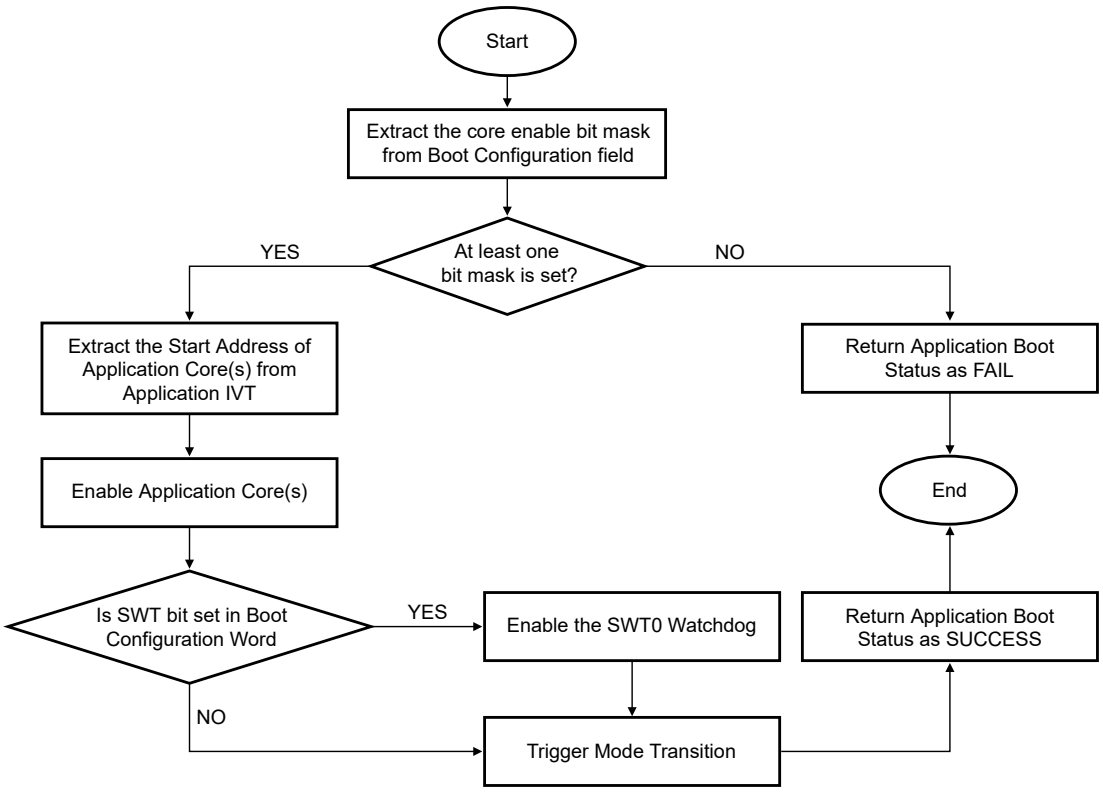



Figure 130. Application boot sequence

Before configuring HSE_CLK, application should wait for Secure BAF to go in WFI by reading register PRTN0_CORE2_STAT.

## 31.12 Recovery mode sequence

This feature is provided to allow the user to program the application image in LC = MCU_PROD and LC= CUST_DEL and debug the reason of corruption of IVT and re-program the IVT in other LC.

Below are the scenarios when Secure BAF executes recovery mode sequence.

- Valid IVT is not found.
- Application is not booted by Secure BAF.
- Application enablement bit is not programmed in Boot configuration word in IVT.
- Application issued more than eight functional reset and disable recovery mode on functional reset bit is not disabled in DCMRWP1 register.
- Application issued more than eight destructive reset and disable recovery mode on destructive reset bit is not disabled in DCMRWP1 register.

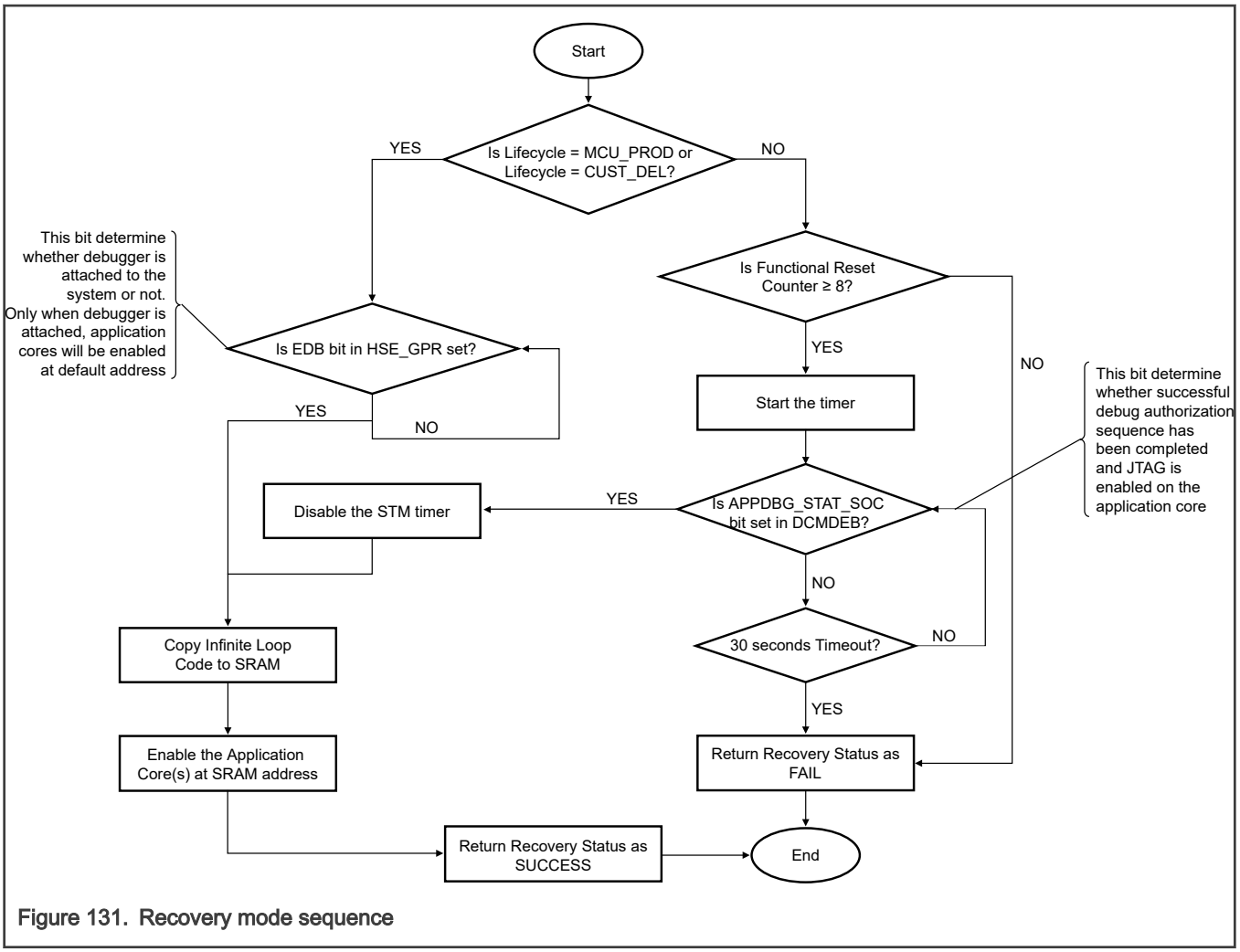



Figure 131. Recovery mode sequence

The infinite loop with WFI code is copied to SRAM1 i.e. 2040_0100h and size of code is 16B. To prevent prefetching errors in application cores, the Infinite loop instruction is placed at 2040_012Ch.

## 31.13 XRDC configuration

Secure BAF needs to ensure that is has access to its own resources and application don't have access to secure area. The following sections describe the XRDC default configuration values when HSE Firmware feature flag is disabled. In this scenario, application don't required to provide XRDC configuration via IVT and Secure BAF never enables the XRDC at any time.

When HSE Firmware Feature flag is enabled, Application can provide XRDC configuration via IVT. For this scenario, see ELE_HSEB Firmware reference manual for the details of XRDC configuration.

### 31.13.1 XRDC configuration of MDAC

Following table list down the default configuration of MDAC by Secure BAF.

Table 175. MDAC registers default value

| Device variant | MDAC Register HSE CPU | Value | Domain number assigned to HSE core |
|---|---|---|---|
| MCXE315/MCXE316/ MCXE317/MCXE31B | MDAC3 | C000_0001h | 1 |

All the other MDAC domain numbers are 0 and not locked by Secure-BAF.

### 31.13.2 XRDC configuration of MRC

Following table list down the default configuration of MRC by Secure BAF.

**Table 176. Default Configuration of MRC**

| MRC Number | Region Descriptor Number | Name | Remarks | Configuration done by Secure BAF |
|---|---|---|---|---|
| 0 | 14 | HSE UTEST area | UTEST area that is accessible by HSE core only.<br>See chip configuration chapter for Start address and End address. | W0- Start address W1= End Address W2={ SE=0,SNUM=0,<br>HSE domain =0x7[All permission] |
| 0 | 15 | Secure BAF Code area | Code area for Secure BAF. See chip configuration chapter for Start address and End address. | W0- Start address<br>W1= End Address<br>W2={ SE=0, SNUM=0,<br>HSE domain =0x7[All permission]<br>Other domains = 0x00 [No permission]<br>}<br>W3={VLD=1, LK2=0x3} |

## 31.14 BAF flash programming controls

Platform flash controller generates an exception of Read-while-Write if Read and Write are performed on the same block simultaneously. HSE core can also perform the Program and Erase for application, and the status is indicated at Configuration PE Lock Register (CONFIG_PE_LOCK) in the HSE space.

BAF locks the high addressing code flash block during its execution and it's cleared when HSE core enters into WFI. Address of high addressing code flash area should be referred to chip configuration chapter.

Before Programming/Erasing/Executing from this address space, application core is requested to poll for WFI bit of PRTN0_CORE2_STAT register to ensure HSE core is in WFI state.

For Boot Sequence 1 or flash synchronization with HSE Firmware, see HSE Firmware reference manual.

Below table explains PE lock bits setting in CONFIG_PE_LOCK register of HSE GPR during Secure BAF execution.

**Table 177. PE lock bits setting in CONFIG_PE_LOCK register of HSE GPR**

| Chip variants | UTEST Block | Data Flash Block | Code Flash Block 3 | Code Flash Block 2 | Code Flash Block 1 | Code Flash Block 0 |
|---|---|---|---|---|---|---|
| MCXE315/ MCXE316 | 1 | 0 | 0 | 0 | 0 | 1 |
| MCXE317 | 1 | 0 | 0 | 0 | 1 | 0 |
| MCXE31B | 1 | 0 | 1 | 0 | 0 | 0 |

## 31.15 Status registers for application usage

This section explain various status register configuration to provide status information to application.

### 31.15.1 Secure BAF version information

Secure BAF version is a 64-bit field. Secure BAF version can be read by application from HSE GPR register 4039_C020h. The version information is explained below.

**Table 178. Secure BAF version information**

| Bit number | Field Name | Description |
|---|---|---|
| 0 – 7 | RESERVED | Reserved, must be 0 |
| 8 - 15 | SOC_TYPE_ID | This field Identifies the SoC family<br>0 – not used<br>5 – ELE_HSEB MCXE31B<br>12 - ELE_HSEB MCXE315/MCXE316<br>13 - ELE_HSEB MCXE317 |
| 16 - 31 | FW_TYPE | This field identify the FW type:<br>0 – Standard generic FW targeting all customers<br>1-7 – Reserved<br>8 >= Custom1, Custom2 (For example: Custom1 = customer X’s project A, Custom2 = customer Y’s project B) |
| 32 - 39 | MAJOR_REV | Major version number<br>0 - EAR/BETA<br>1 - at first RTM and incremented by 1 for subsequent RTM |
| 40 - 47 | MINOR_REV | 4 – Pre-EAR<br>8 - EAR<br>9 - BETA<br>This field is reset for every new MAJOR_REV change. |
| 48 - 63 | PATCH_REV | bug fix releases (After BETA) or new feature for EAR/PRE-EAR releases. (least significant 8 bits are reserved for hotfix / frozen branch info) |

### 31.15.2 DCM Register DCMRWP1

Application can disable recovery mode entry by Secure BAF after programming bits 23 and 22 of DCM register DCMRWP1. This register gets clears on power on reset.

**Table 179. DCM register DCMRWP1 availalble at address 402A_C400h**

| Bits | Number of bits | R/W access by application | Description |
|---|---|---|---|
| 24-31 | 8 | | Reserved |

*Table continues on the next page...*

Table 179. DCM register DCMRWP1 availalble at address 402A_C400h (continued)

| Bits | Number of bits | R/W access by application | Description |
|---|---|---|---|
| 23 | 1 | R/W | Disable recovery mode on destructive reset.<br>This bit is reset by default and Secure BAF allows recovery mode sequence if Application issues >8 destructive reset. Application can set this bit to disable recovery mode when Application issues > 8<br>destructive reset. [1] |
| 22 | 1 | R/W | Disable recovery mode on functional reset.<br>This bit is reset by default and Secure BAF allows recovery mode sequence if Application issues >8 functional reset. Application can set this bit to disable recovery mode when Application issues > 8<br>functional reset. [1] |
| 21 | 1 | R | Reserved |
| 16-20 | 5 | R | Recovery mode Reset counter. For OEM_PROD and INFIELD lifecycle, to enable recovery mode functionality, Secure BAF increments this counter when Functional or destructive reset is issued. |
| 15 | 1 | R | Reserved |
| 11-14 | 4 | R | Destructive reset counter. Secure BAF increments this counter when Destructive reset is issued. |
| 0-10 | 11 | R | Reserved |

[1] Application core should not modify other bits of this register while updating these bits.

### 31.15.3 Status bits on HSE GPR register

HSE GPR register 4039_C028h is configured by Secure BAF to show various status information as explained in below table.

Table 180. Status bits on HSE GPR register available at 4039_C028h

| Bit | Description |
|---|---|
| 0 | HSE FW is present and booted by Secure BAF. |
| 1-4 | Reserved |
| 5 | Application cores booted in Recovery mode by SBAF. |
| 6 | Debug Authentication performed by SBAF. |
| 7-31 | Reserved. |

## 31.16 Interrupt and exception handling

### 31.16.1 Interrupt handling

No special interrupt handling routines are required during the boot process. Interrupts are disabled during Secure BAF execution.

### 31.16.2 Exception handling

Secure BAF enters the recovery mode after Enabling Debug Authorization.

After eight consecutive functional resets or destructive resets either from Application Firmware, the Device Enters the Recovery Mode.

### 31.16.3 Boot target watchdog

Secure BAF conditionally configures SWT0 watchdog with default timeout i.e. 25 ms, before enabling the application core(s). It is expected that the application services this watchdog before expiration.

## 31.17 Hardware modules used by Secure BAF

### 31.17.1 MC_ME

Secure BAF uses MC_ME to enable the Application cores, mode switch, and other operations during its execution. MC_ME operations supported by Secure BAF are discussed in below section.

### 31.17.2 FXOSC

Secure BAF configures FXOSC if FXOSC enablement flag is programmed in UTEST.

### 31.17.3 Clock Generation Module (MC_CGM)

Secure BAF configures MC_CGM in Reduced Clock Mode as well. See following section for the same.

### 31.17.4 DCM

Secure BAF uses the DCM to identify the Life-Cycle, Standby Boot Configuration, Lockstep, Reduced Clock Mode Configuration.

### 31.17.5 HSE_GPR

Secure BAF configures Hardware Protection, Program Erase Lock, FIRC Divider Control, Secure BAF Version number, HSE FW Handshaking via HSE-GPR.

### 31.17.6 XRDC

Secure BAF configures the XRDC Configuration. See XRDC Configuration for the same.

### 31.17.7 Flash module

Secure BAF always perform the write and erase operation on alternate interface.

See Chip Configuration for details of Flash memory usage by Secure BAF.

## 31.18 Hardware IP registers details modified by Secure BAF

Below table explains the registers details which are modified by Secure BAF when HSE Firmware Feature Flag is disabled.

Table 181. Hardware IP registers modified by Secure BAF

| IP | Register Name | Default value (hex) | Modified value (hex) |
|---|---|---|---|
| | Master Domain Assignment (MDA_W0_3_DFMT0) | 0000_0000h | C000_0002h<br>C000_0001h |
| | Memory Region Descriptor (MRGD_W0_14) | 0000_0001h | HSE Utest Start [1] |
| | Memory Region Descriptor (MRGD_W1_14) | 0000_001Fh | HSE Utest End [1] |
| | Memory Region Descriptor (MRGD_W2_14) | 0000_0000h | 0000_01C0h |
| | Memory Region Descriptor (MRGD_W3_14) | 0000_0000h | E000_0000h |
| | Memory Region Descriptor (MRGD_W0_15) | 0000_0001h | HSE Code Flash Start [1] |
| | Memory Region Descriptor (MRGD_W1_15) | 0000_001Fh | HSE code Flash End [1] |
| | Memory Region Descriptor (MRGD_W2_15) | 0000_0000h | 0000_01C0h |
| | Memory Region Descriptor (MRGD_W3_15) | 0000_0000h | E000_0000h |
| | Peripheral Domain Access Control (PDAC_W0_155) | 0000_0000h | 0000_01C0h |
| | Peripheral Domain Access Control (PDAC_W1_155) | 0000_0000h | E000_0000h |
| | Peripheral Domain Access Control (PDAC_W0_188) | 0000_0000h | 0000_01C0h |
| | Peripheral Domain Access Control (PDAC_W1_188) | 0000_0000h | E000_0000h |
| | Peripheral Domain Access Control (PDAC_W0_231) | 0000_0000h | 0000_01FFh |
| | Peripheral Domain Access Control (PDAC_W1_231) | 0000_0000h | E000_0000h |
| General purpose registers with write access by HSE core | Configuration Program and Erase Lock (CONFIG_PE_LOCK) | 0000_0000h | Protect only HSE Area [1] |
| | Configuration RAM Protected Region (CONFIG_RAMPR) | See chapter EdgeLock Secure Enclave (HSE_B) | Release Full memory [1] |
| | Configuration RAM Protected Region Alternate (CONFIG_RAMPR_ALT) | See chapter EdgeLock Secure Enclave (HSE_B) | Release Full memory [1] |
| | Configuration Code Flash Memory Active Block (CONFIG_CFPRL) | See chapter EdgeLock Secure Enclave (HSE_B) | Protect only HSE Code Flash Area [1] |
| | Configuration Code Flash Memory Active Block Alternate (CONFIG_CFPRL_ALT) | See chapter EdgeLock Secure Enclave (HSE_B) | Protect only HSE Code Flash Area [1] |

*Table continues on the next page...*

Table 181. Hardware IP registers modified by Secure BAF (continued)

| IP | Register Name | Default value (hex) | Modified value (hex) |
|---|---|---|---|
| | Configuration Code Flash Memory Passive Block (CONFIG_CFPRH) | See chapter EdgeLock Secure Enclave (HSE_B) | Protect only HSE Code Flash Area [1] |
| | Configuration Code Flash Memory Passive Block Alternate (CONFIG_CFPRH_ALT) | See chapter EdgeLock Secure Enclave (HSE_B) | Protect only HSE Code Flash Area [1] |
| | Configuration Data Flash Memory Protected Region (CONFIG_DFPR) | See chapter EdgeLock Secure Enclave (HSE_B) | Protect only HSE Data Flash Area [1] |
| | Configuration Data Flash Memory Protected Region Alternate (CONFIG_DFPR_ALT) | See chapter EdgeLock Secure Enclave (HSE_B) | Protect only HSE Data Flash Area [1] |
| | Configuration REG_GPR (CONFIG_REG_GPR) | A000_0003h | Refer chapter ”FIRC Divider Register Control“. |
| MC_ME | Partition 1 COFB Set 1 Clock Status Register (PRTN1_COFB1_STAT) | 1CFE_2FFCh | Clock of FXOSC and MC_CGM are enabled by default. But for 120 MHz clock request, Secure BAF ensures if MC_CGM clock is enabled. |
| | Partition 0 Core 0 Address Register (PRTN0_CORE0_ADDR) | 0040_0000h | Secure BAF update this address if App core 0 boot is requested during Normal boot sequence in IVT. Or in recovery mode. |
| | Partition 0 Core 0 Process Configuration Register (PRTN0_CORE0_PCONF) | 0000_0000h | 0000_0001h |
| | Partition 0 Core 0 Process Update Register (PRTN0_CORE0_PUPD) | 0000_0000h | 0000_0001h |
| | Partition 0 Core 1 Process Configuration Register (PRTN0_CORE1_PCONF) | 0000_0000h | 0000_0001h |
| | Partition 0 Core 1 Process Update Register (PRTN0_CORE1_PUPD) | 0000_0000h | 00000001h |
| | Partition 0 Core 1 Address Register (PRTN0_CORE1_ADDR) | 0041_0000h | Secure BAF update this address if App core 1 boot is requested during Normal boot sequence in IVT. Or in recovery mode. |

*Table continues on the next page...*

Table 181. Hardware IP registers modified by Secure BAF (continued)

| IP | Register Name | Default value (hex) | Modified value (hex) |
|---|---|---|---|
| | Partition 0 Core 2 Process Configuration Register (PRTN0_CORE2_PCONF) | 0000_0000h | 0000_0001h |
| | Partition 0 Core 2 Process Update Register (PRTN0_CORE2_PUPD) | 0000_0000h | 0000_0001h |
| | Partition 0 Core 2 Address Register (PRTN0_CORE2_ADDR) | See Mode Entry Module (MC_ME) chapter | Secure BAF update this address if App core 2 boot is requested during Normal boot sequence in IVT. Or in recovery mode. |
| | Control Key Register (CTL_KEY) | 0000_5AF0h | Secure BAF update this register if anyone of application core is booted by Secure BAF. 0000_5AF0h and then 0000_A50Fh |
| FXOSC | FXOSC Control Register (CTRL) | 019D_00C0h | See Crystal oscillator configuration flag |
| MC_RGM | Functional Reset Escalation Counter Register (FREC) | 0000_0000h | 0000_0000h |
| | Destructive Reset Escalation Threshold Register (DRET) | 0000_0000h | 0000_000Fh |
| SWT_0 | Control (CR) | FF00_010Ah | FF00_000Bh |
| | Interrupt (IR) | 0000_0000h | 0000_0001h |
| | Service (SR) | 0000_0000h | C520h and then D928h |

1. See Chip Configuration for details of HSE area addressing.

## 31.19 Appendix

Secure BAF takes into account for following scenarios to prevent bricking of device :

After Maximum Functional Resets are observed in the device, BAF enters into Recovery Mode, to recover application core's failing status.

BAF does not allows the LC to be advanced to OEM_PROD or IN_FIELD, if CUST_DBG_PSWD/ADKP is not programmed by the Application.

BAF boots the Application from System-RAM during recovery to avoid unpredictable behavior.

## 31.20 Glossary

**AES** Advanced Encryption Standard; a cipher primitive

**BAF** Boot Assist Flash

**CUST_DEL** Device Life Cycle customer delivery

| | |
|---|---|
| **IN_FIELD** | Device Life Cycle is IN_FIELD |
| **LC** | Device Life Cycle; used to limit by design the configuration and debug/test possibilities of the device for in-field usage |
| **MCU_PROD** | Device Life Cycle MCU Production |
| **OEM_PROD** | Device Life Cycle OEM Production |
| **OTA** | Over The Air |
| **Secure BAF** | Secure Boot Assist Flash |

# Chapter 32
# Reset Generation Module (MC_RGM)

## 32.1 Chip-specific MC_RGM information

### 32.1.1 MC_RGM configuration

The "Reset sources—POR, Destructive, and Functional" section of the "Reset Overview" chapter provides information about MC_RGM's reset sources. The chapter also provides details about the chip's reset architecture.

**NOTE**

The ERCTRL register configuration takes several cycles to be effective. Any further access to MC_RGM must happen after at least nine AIPS_SLOW_CLK cycles of writing to ERCTRL.

### 32.1.2 Functional reset entry timer implementation

The default timeout value of the functional reset entry timer is 2048 clocks of MC_RGM clock (FIRC). If the RGM entry sequence hangs, then POR_WDG would trigger and the status of this functional reset entry sequence timeout is indicated at chip-level status register. The status of timeout in such case, is indicated at DCM.DCMROPP1[28]. There is no impact to the device behavior if the functional reset entry sequence gets completed within the POR_WDG timeout window.

### 32.1.3 MCXE31 reset state machine

The reset sequence of the MCXE31 products is depicted in the figure below:

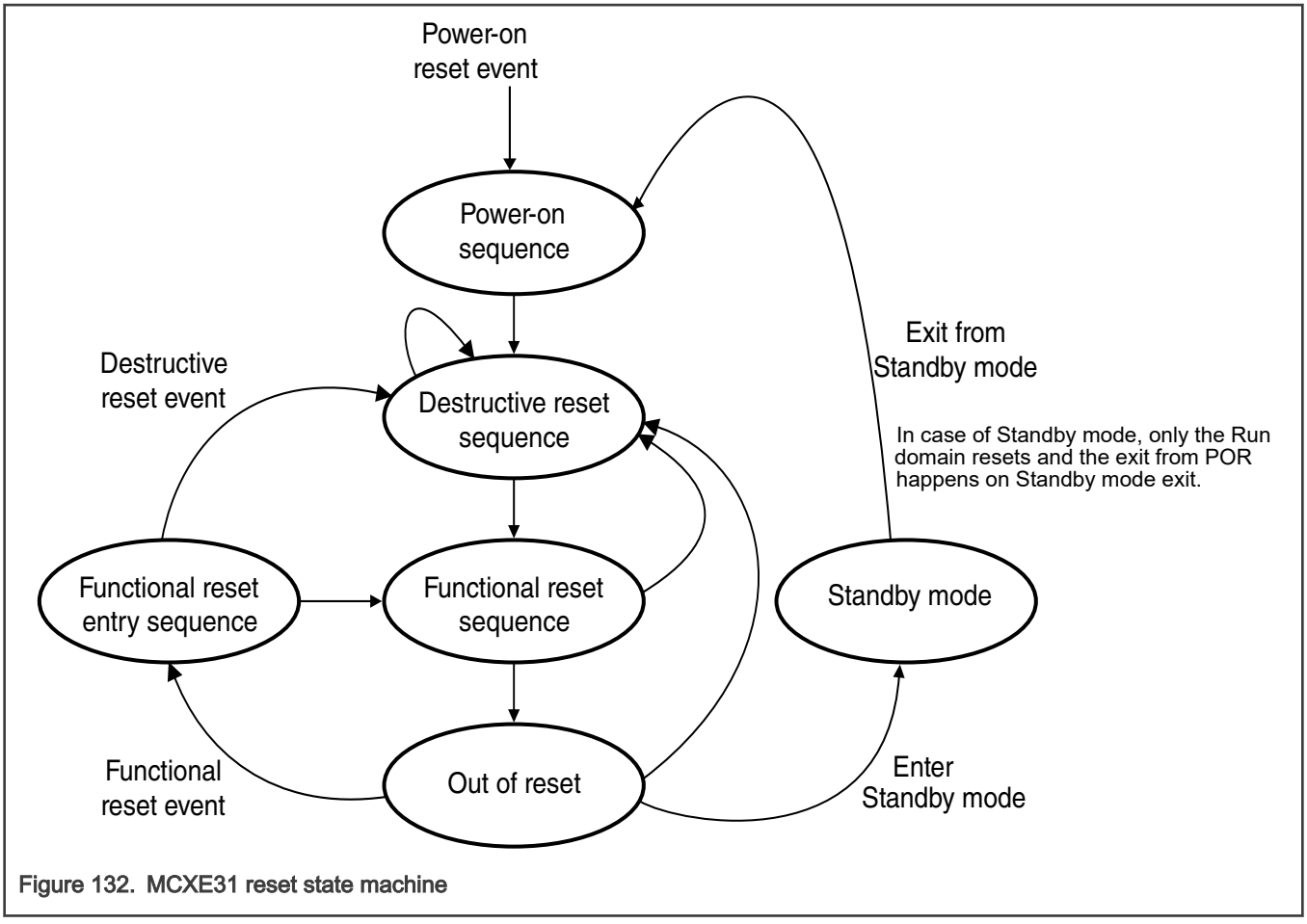



**Figure 132. MCXE31 reset state machine**

## 32.2 Introduction

The Reset Generation Module (MC_RGM) centralizes the different reset sources and manages the reset sequence of the chip. It provides a register interface and the reset sequencer. There are various registers available in this module to monitor and control the chip reset sequence.

The following figure shows the block diagram of MC_RGM.

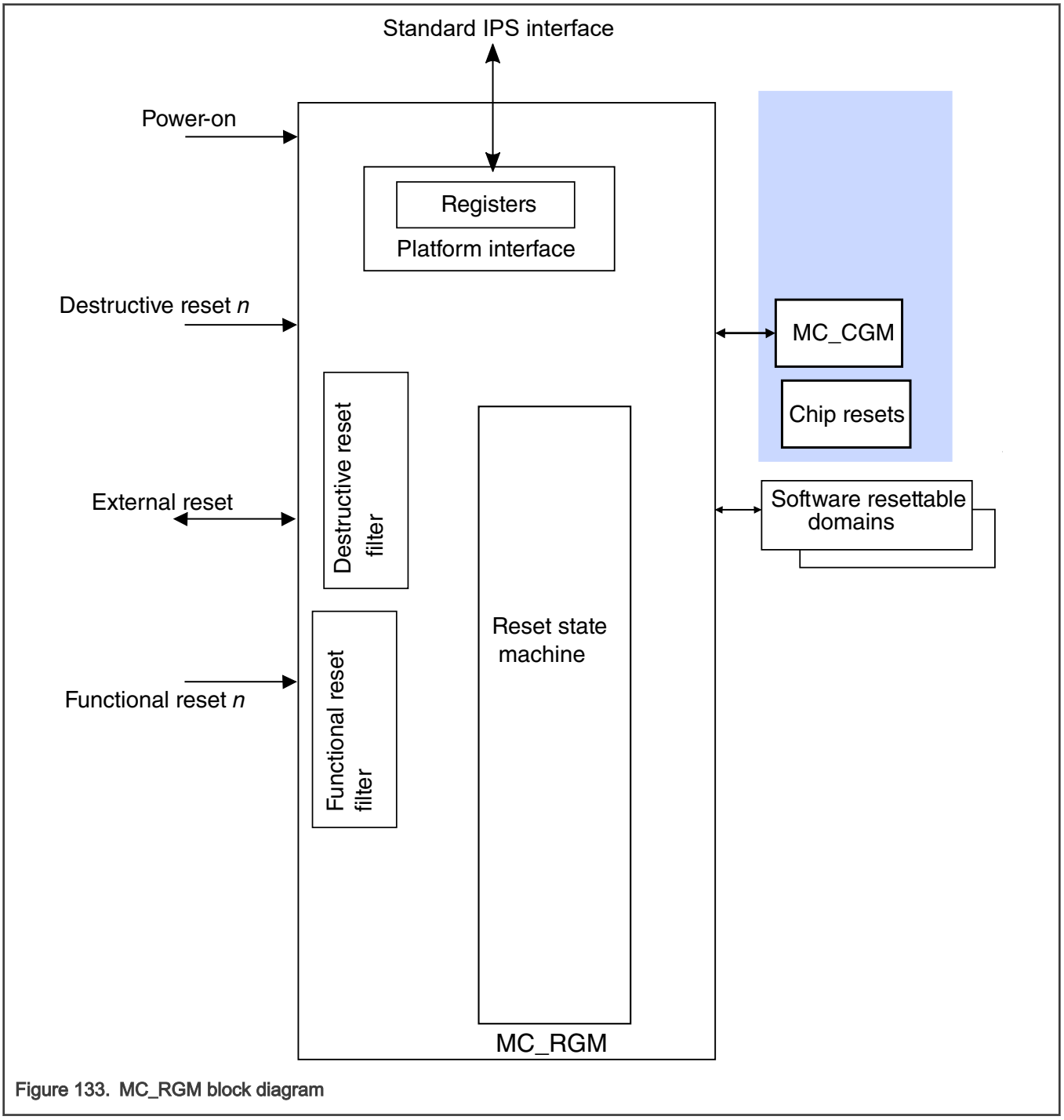



Figure 133. MC_RGM block diagram

## 32.3 Features

Here are the key features of MC_RGM:

- Destructive and functional reset management
- Capturing the reset sources for each reset sequence (reset status flags)
- Assertion the RESET_B pin to propagate the reset sequence out of chip
- Configurable escalation of recurring 'functional' resets to 'destructive' reset
- Configurable escalation of recurring 'destructive' resets to keep the chip in the reset state until the next power-on reset
- Software controllable reset assertion
- Pad safe state control generation

## 32.4 Reset sources

The reset sources are organized in three categories: power-on, destructive, and functional.

A power-on reset source is associated with an event typically related to power-up or low-voltage scenarios. When a power-on reset occurs, the full reset sequence is applied to the chip. This resets the full chip, including the MC_RGM, and the memory content must be considered to be invalid.

A destructive reset source is associated with an event related to a critical, usually hardware, error or dysfunction. When a destructive reset event occurs, the full reset sequence is applied to the chip. This resets the full chip ensuring a safe start-up state for both digital and analog modules, and the memory content must be considered to be invalid.

A functional reset source is associated with an event related to a less-critical, usually non-hardware, error or dysfunction. When a functional reset event occurs, a partial reset sequence is applied to the chip. In this case, most digital modules are reset normally, while the state of analog modules or specific digital modules as well as the system memory content is preserved.

## 32.5 External signal description

The MC_RGM interfaces with the RESET_B pin.

The following table describes the signals that are connected to the I/O pad ring.

**Table 182. MC_RGM external signals**

| Signal name | Reset value | Description |
|---|---|---|
| RESET_B | 0 | Active low external reset.<br>A bidirectional reset pin indicating the reset state. |

## 32.6 RESET_B pin assertion and pin safe state control

The MC_RGM asserts the RESET_B pin when the device is in a reset sequence, and it remains asserted until the end of the reset sequence. During this reset sequence, most of the chip's pins are safe/pad stated according to the values shown in the IOMUX table/spreadsheet. Note that the safe state values may vary according to the reset sequence type.

In addition, the MC_RGM has a feature to assert the RESET_B pin through software, without initiating a reset sequence. This is achieved by writing 1b to the ERCTRL[ERASSERT]. When this occurs, most of the chip's pins are safe-stated according to the values shown in the IOMUX table/spread sheet. The RESET_B assertion and pin safe-stating remain active until the end of the next reset sequence.

This features has to be used only with selftest of the main reset domain.

## 32.7 Functional description

### 32.7.1 Reset state machine

The main role of MC_RGM is the generation of the reset sequence that ensures that the correct parts of the chip are reset based on the reset source event.

For each reset event, immediately after it is captured by the MC_RGM, the following takes place:

1. The corresponding reset event status bit is set in the MC_RGM_DES and MC_RGM_FES registers.
2. The pins are put into their default states
3. The RESET_B pin is asserted.

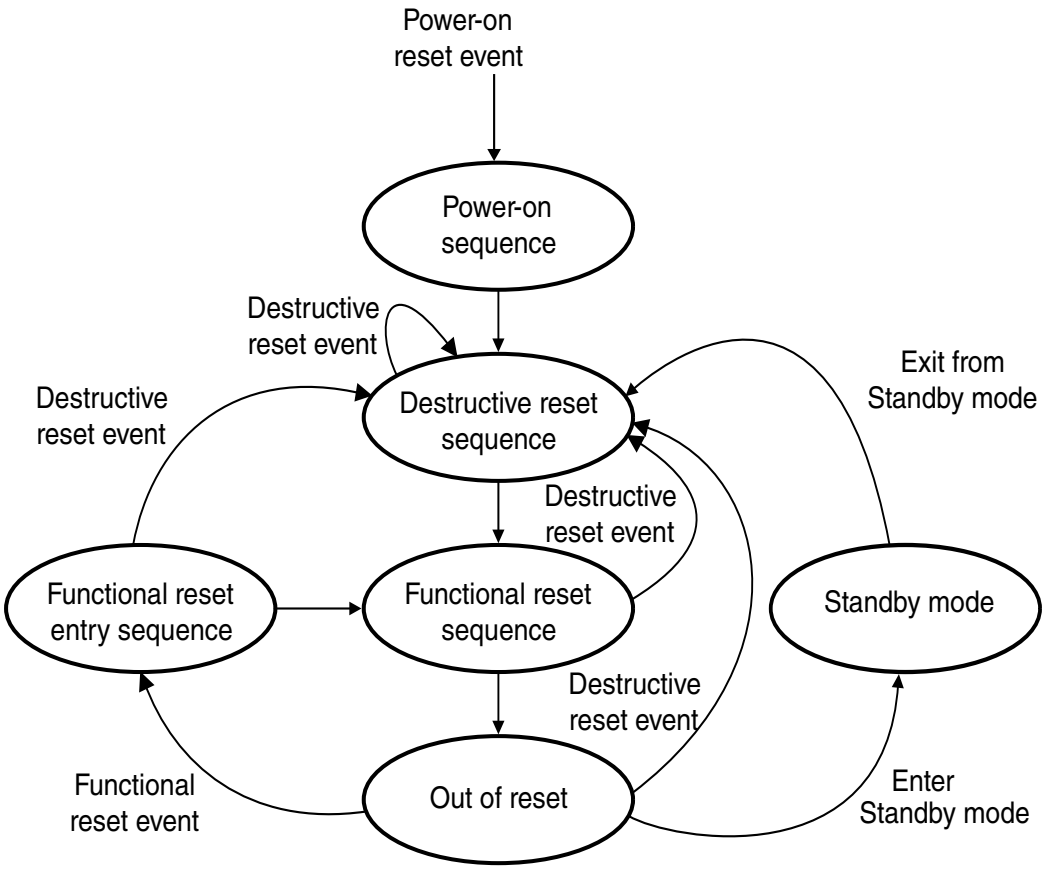



Figure 134. Reset sequence

**NOTE**

See chip-specific MC_RGM information for chip-specific reset sequence details.

#### 32.7.1.1 Power-on reset sequence

A reset is always generated when the power-on reset source is asserted, and it has priority over all other reset sources. Such a power-on reset forces the reset state machine to enter the power-on sequence resulting in the assertion of all reset signals. The reset state machine starts progressing when the following two conditions are verified:

1. All the power-on reset events are cleared
2. The MC_RGM's clock source (the FIRC) has started up and stabilized

If a power-on reset event has occurred, the DES[F_POR] bit is set.

The power-on reset cannot be demoted by the software.

#### 32.7.1.2 Destructive reset sequence

The 'Destructive reset sequence' is comprised of a number of phases, where DEST0 is the first phase and is followed by DEST1 and so forth.

This phase is entered immediately from any phase on a power-on, standby reset sequence, or enabled destructive reset event . A destructive reset counter starts immediately on entry in the DEST0 phase. The DEST0 state is exited to the DEST1 state on the rising edge of FIRC_CLK immediately after all of the following conditions have been established:

- The DEST0 duration time has expired
- All destructive reset inputs are cleared

The DEST0 state is immediately exited to the power-on state if a power-on reset event occurs.

The reset state machine exits the destructive reset sequence and enters the functional reset sequence when:

- All the destructive reset events are cleared.
- All the processes that take place during the destructive reset sequence have completed. For details, see Reset chapter.
- The 'destructive' reset escalator counter has not reached the value in DRET[DRET].

#### 32.7.1.3 Functional reset sequence

There are two functional reset sequence, the functional reset entry sequence and the functional reset exit sequence.

##### 32.7.1.3.1 Functional reset entry sequence

The functional reset entry sequence is only entered when a functional reset event occurs during the idle phase.

If a functional reset event occurs during an ongoing reset sequence, the corresponding event status flag is set, and the RESET_B pin is asserted per the reset event's configuration. However, the reset sequence is not influenced, and it continues to progress without interruption.

Functional reset is not asserted during functional reset entry sequence.

The functional reset entry sequence is exited to the DEST0 on the next rising edge of FIRC if a destructive reset event has occurred. The sequence immediately enters the power up sequence if a POR event occurs

In all other cases, the sequence exits to the first stage of functional reset exit sequence.

##### 32.7.1.3.2 Functional reset exit sequence

The functional reset exit sequence is entered either on exit from the destructive reset sequence or on completion of the functional reset entry sequence. The reset state machine exits this sequence and enters the idle phase on verification of the following:

- All the functional reset events are cleared.
- All the processes that take place during the functional reset sequence have completed. For details, see Reset chapter.

If a functional reset event occurs during an ongoing reset sequence, the corresponding event status flag is set, and the RESET_B pin is asserted per the reset event's configuration. However, the reset sequence is not influenced, and it continues to progress without interruption.

#### 32.7.1.4 Idle phase

This is the final phase and is entered on exit from the functional reset exit sequence . When this phase is reached, MC_RGM releases control of the system to the platform and waits for the new reset events that can trigger a reset sequence.

### 32.7.2 Destructive resets

A destructive reset indicates that an event has occurred after which critical register or memory content can no longer be guaranteed.

The status flag associated with a given destructive reset event (Destructive Event Status Register (DES)) is set when the destructive reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously and the software determines which reset source is the most critical for the application.

The low-voltage detector threshold ensures that when the reset corresponding to the core supply low-voltage detect is enabled, the supply is sufficient to have the destructive event correctly propagated through the digital logic. Therefore, if a given destructive reset is enabled, MC_RGM ensures that the associated reset event is correctly triggered to the full system.

An enabled destructive reset triggers a reset sequence starting from the beginning of DEST0.

### 32.7.3 External reset

MC_RGM manages the external reset coming from RESET_B. The detection of a falling edge on RESET_B starts the reset sequence from the beginning of the destructive reset entry sequence.

The status flag associated with the external reset falling edge event (the FES[F_EXR]) is set when the external reset is asserted and the power-on reset is not asserted.

### 32.7.4 Functional resets

A functional reset indicates that an event has occurred after which it can be guaranteed that critical register and memory content is still intact.

The status flag associated with a given functional reset event (Functional /External Reset Status Register (FES)) is set when the functional reset is asserted and the power-on reset is not asserted. It is possible for multiple status bits to be set simultaneously and the software determines which reset source is the most critical for the application.

An enabled functional reset triggers a reset sequence starting from the beginning of the functional reset entry sequence.

### 32.7.5 Functional reset entry timer

The purpose of Functional Reset Entry Timer feature is to indicate any scenario wherein functional reset entry sequence does not get completed in prerequisite time .

When FRENTC[FRET_EN] is enabled, the functional reset entry timer starts from functional reset event point and gets terminated if the functional reset entry sequence is successfully completed or when it reached to the timeout value . It gets cleared on any destructive or POR reset events.

The timeout value is controlled by FRENTC[FRET_TIMEOUT]. The FRENTC[FRET_TIMEOUT] value should be greater than zero. If configured as zero, the default timeout value 200 gets used.

### 32.7.6 Alternate event generation

MC_RGM provides alternative events to be generated on reset source assertion. When a reset source is asserted, MC_RGM normally enters the reset sequence. Alternatively, it is possible for some reset source events to be converted from a reset to an interrupt request issued to the core. Alternate event selection for a given reset source is made through the RGM_FERD register as shown in the following table.

**Table 183. Functional Reset Disable Register (RGM_FERD) field descriptions**

| RGM bit FERD value | Generated event |
|---|---|
| 0 | Reset |
| 1 | Interrupt request |

The alternate event is cleared by deasserting the source of the request (that is, at the reset source that caused the alternate request) and also clearing the appropriate RGM_FES status bit.

### 32.7.7 RESET_B assertion control

The software indicates to the MC_RGM that the RESET_B is to be asserted by writing to the **ERASSERT** bit in the RGM_ERCTRL register. When this bit is set by the software, RESET_B gets asserted. Setting of this field does not impact the reset sequence in any way.

An example where the ERCTRL[ERASSERT] field could be set by the software is when entering the self test sequence, during which RESET_B is to be asserted. This indicates the chip is not available in the functional mode although a reset sequence is not in progress. The deassertion of RESET_B is not controlled by the software. Instead, the RESET_B pin remains asserted until the next time the chip exits a reset sequence.

**ERASSERT** bit is also cleared during the reset sequence.

MC_RGM asserts the external reset if the reset sequence is triggered by one of the following:

- A power-on reset
- A destructive reset event
- A functional reset event

In this case, external reset is asserted until all conditions for the exiting reset sequence have been met, with the exception of the RESET_B assertion check

### 32.7.8 Functional reset escalation

Functional reset escalation can be used to generate a destructive reset if a number of functional resets is occurred between software writes to the RGM_FRET register. This function is enabled by writing a non-zero value to the FRET field of this register.

After the functional reset escalation is enabled, MC_RGM increases a counter on each functional reset that causes a reset sequence to be initiated (which means, entrance into FUNC0 from the IDLE phase). This counter is cleared on a write of any value to the RGM_FRET register and on any power-on or destructive reset. If the counter reaches the value in the FRET field of the RGM_FRET register, MC_RGM asserts a destructive reset.

The following figure shows the functional reset escalation counter.

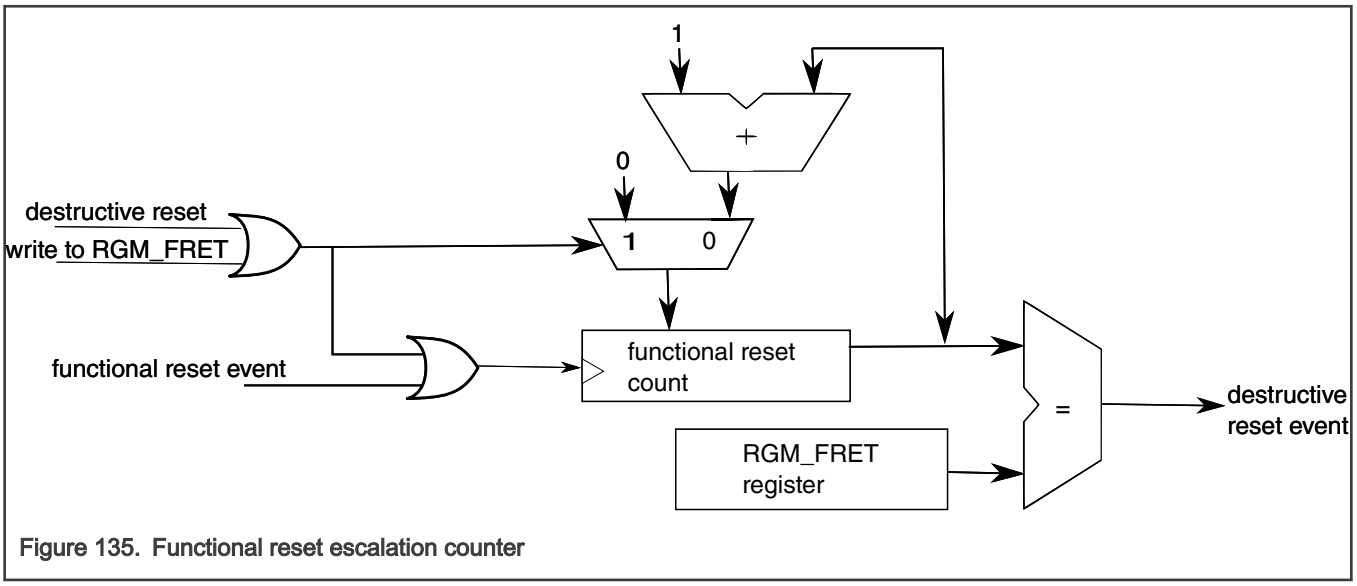



Figure 135. Functional reset escalation counter

**NOTE**

Functional counter increments for each reset source for which escalation is enabled. For details, see "Reset sources" table in the Reset chapter.

### 32.7.9 Destructive reset escalation

Destructive reset escalation can be used to keep the chip in the reset state until the power-on triggers a reset sequence if a number of destructive resets are occurred between software writes to the RGM_DRET register. This function is enabled by writing a non-zero value to the DRET field of this register.

After destructive reset escalation is enabled, MC_RGM increases a counter on each destructive reset that is enabled. . This causes a reset sequence to be initiated (that is, entrance into DEST0 from the idle phase or any other reset phase) or an ongoing reset sequence to restart (that is, entrance into DEST0 from any other reset phase). This counter is cleared on a write of any value to the RGM_DRET register and on any power-on reset. If the counter reaches the value in the DRET field of the RGM_DRET register, MC_RGM enters reset DEST0 and stays there until the next power-on reset occurs .

The following figure shows the destructive reset escalation counter.

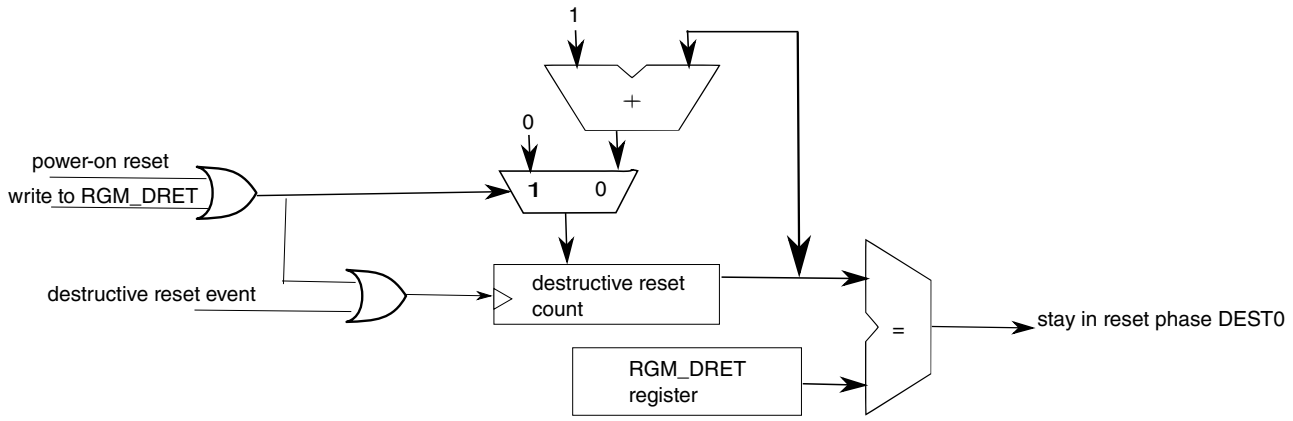



**Figure 136. Destructive reset escalation counter**

**NOTE**

Destructive counter increments for each reset source for which escalation is enabled. For details, see "Reset sources" table in the Reset chapter.

## 32.8 MC_RGM register descriptions

Access to the following locations do not generate transfer error:

- 2Ch

All registers can be accessed as read/write in supervisor mode.

### 32.8.1 MC RGM Register Map memory map

MC_RGM base address: 4028_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Destructive Event Status Register (DES) | 32 | RW | 0000_0001h |
| 8h | Functional /External Reset Status Register (FES) | 32 | RW | 0000_0000h |
| Ch | Functional Event Reset Disable Register (FERD) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 10h | Functional Bidirectional Reset Enable Register (FBRE) | 32 | RW | 0000_0000h |
| 14h | Functional Reset Escalation Counter Register (FREC) | 32 | RW | 0000_0000h |
| 18h | Functional Reset Escalation Threshold Register (FRET) | 32 | RW | 0000_000Fh |
| 1Ch | Destructive Reset Escalation Threshold Register (DRET) | 32 | RW | 0000_0000h |
| 20h | External Reset Control Register (ERCTRL) | 32 | RW | 0000_0000h |
| 24h | Reset During Standby Status Register (RDSS) | 32 | RW | 0000_0000h |
| 28h | Functional Reset Entry Timeout Control Register (FRENTC) | 32 | RW | 0000_0001h |

## 32.8.2 Destructive Event Status Register (DES)

### Offset

| Register | Offset |
|---|---|
| DES | 0h |

### Function

This register contains the status of the 'destructive' reset sources. This register can be accessed as read/write in supervisor mode and read-only in user mode. Register bits are cleared on write '1'. This register is reset only on power-on reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DEBUG_D... | SW_DEST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | HSE_SNV... | HSE_TMP... | 0 |
| W | | W1C | W1C | | | | | | | | | | | W1C | W1C | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SYS_DIV... | HSE_CLK... | 0 | AIPS_PL... | 0 | CORE_CL... | PLL_LOL | FXOSC_F... | 0 | MC_RGM_... | 0 | STCU_URF | FCCU_FTR | 0 | 0 | F_POR |
| W | W1C | W1C | | W1C | | W1C | W1C | W1C | | W1C | | W1C | W1C | | | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### Fields

| Field | Function |
|---|---|
| 31 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 30<br>DEBUG_DEST | Flag for 'Destructive' Reset DEBUG_DEST<br>0b - 'Destructive' reset event DEBUG_DEST has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Destructive' reset event DEBUG_DEST has occurred. |
| 29<br>SW_DEST | Flag for 'Destructive' Reset SW_DEST<br>0b - 'Destructive' reset event SW_DEST has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Destructive' reset event SW_DEST has occurred. |
| 28<br>— | Reserved |
| 27<br>— | Reserved |
| 26<br>— | Reserved |
| 25<br>— | Reserved |
| 24<br>— | Reserved |
| 23<br>— | Reserved |
| 22<br>— | Reserved |
| 21<br>— | Reserved |
| 20<br>— | Reserved |
| 19<br>— | Reserved |
| 18 | Flag for 'Destructive' Reset HSE_SNVS_RST |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| HSE_SNVS_RST | 0b - 'Destructive' reset event HSE_SNVS_RST has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Destructive' reset event HSE_SNVS_RST has occurred. |
| 17<br>HSE_TMPR_RST | Flag for 'Destructive' Reset HSE_TMPR_RST<br>0b - 'Destructive' reset event HSE_TMPR_RST has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Destructive' reset event HSE_TMPR_RST has occurred. |
| 16<br>— | Reserved |
| 15<br>SYS_DIV_FAIL | Flag for 'Destructive' Reset SYS_DIV_FAIL<br>0b - 'Destructive' reset event SYS_DIV_FAIL has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Destructive' reset event SYS_DIV_FAIL has occurred. |
| 14<br>HSE_CLK_FAIL | Flag for 'Destructive' Reset HSE_CLK_FAIL<br>0b - 'Destructive' reset event HSE_CLK_FAIL has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Destructive' reset event HSE_CLK_FAIL has occurred. |
| 13<br>— | Reserved |
| 12<br>AIPS_PLAT_CLK_FAIL | Flag for 'Destructive' Reset AIPS_PLAT_CLK_FAIL<br>0b - 'Destructive' reset event AIPS_PLAT_CLK_FAIL has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Destructive' reset event AIPS_PLAT_CLK_FAIL has occurred. |
| 11<br>— | Reserved |
| 10<br>CORE_CLK_FAIL | Flag for 'Destructive' Reset CORE_CLK_FAIL<br>0b - 'Destructive' reset event CORE_CLK_FAIL has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Destructive' reset event CORE_CLK_FAIL has occurred. |
| 9<br>PLL_LOL | Flag for 'Destructive' Reset PLL_LOL<br>0b - 'Destructive' reset event PLL_LOL has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Destructive' reset event PLL_LOL has occurred. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 8<br>FXOSC_FAIL | Flag for 'Destructive' Reset FXOSC_FAIL<br>0b - 'Destructive' reset event FXOSC_FAIL has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Destructive' reset event FXOSC_FAIL has occurred. |
| 7<br>— | Reserved |
| 6<br>MC_RGM_FRE | Flag for 'Destructive' Reset MC_RGM_FRE<br>0b - 'Destructive' reset event MC_RGM_FRE has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Destructive' reset event MC_RGM_FRE has occurred. |
| 5<br>— | Reserved |
| 4<br>STCU_URF | Flag for 'Destructive' Reset STCU_URF<br>0b - 'Destructive' reset event STCU_URF has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Destructive' reset event STCU_URF has occurred. |
| 3<br>FCCU_FTR | Flag for 'Destructive' Reset FCCU_FTR<br>0b - 'Destructive' reset event FCCU_FTR has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Destructive' reset event FCCU_FTR has occurred. |
| 2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>F_POR | Flag for power-on reset<br>**NOTE**<br>If this field is set, ignore all the fields of Destructive Event Status Register (DES) and Functional /External Reset Status Register (FES) registers at power up.<br>0b - No power-on event has occurred since the last clear.<br>1b - A power-on event has occurred. |

## 32.8.3 Functional /External Reset Status Register (FES)

### Offset

| Register | Offset |
|---|---|
| FES | 8h |

### Function

This register contains the status of the 'functional' and external reset sources. This register can be accessed as read/write in supervisor mode and read-only in user mode. Register fields are cleared on write '1' if the triggering event has already been cleared at the source. This register is reset only on power-on reset.

**NOTE**

- The startup self-test functional reset event to MC_RGM initiates a functional reset sequence after which the FES[ST_DONE] is set. This field is sticky in nature and gets cleared if software clears this field.
- If functional reset escalation to destructive reset is disabled, then the status of this register must be ignored if the fields of Destructive Event Status Register (DES) other than DES[F_POR] are set.
- If functional reset escalation to destructive reset is enabled and if the fields of Destructive Event Status Register (DES), other than DES[F_POR], are set, then based on these fields user should check if the cause of the destructive reset was due to functional reset escalation or if it was triggered directly by a destructive reset source, in which case FES needs to be ignored.
- See chip-specific MC_RGM information for applicability of the FES[ST_DONE] field.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DEBUG_F... | SW_FUNC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | HSE_BOO... | 0 | 0 | 0 | HSE_SWT... |
| W | | W1C | W1C | | | | | | | | | W1C | | | | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | JTAG_RST | 0 | 0 | SWT0_RST | 0 | ST_DONE | FCCU_RST | 0 | 0 | F_EXR |
| W | | | | | | | W1C | | | W1C | | W1C | W1C | | | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30 | Flag for 'Functional' Reset DEBUG_FUNC |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| DEBUG_FUNC | 0b - 'Functional' reset event DEBUG_FUNC has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Functional' reset event DEBUG_FUNC has occurred. |
| 29<br>SW_FUNC | Flag for 'Functional' Reset SW_FUNC<br>0b - 'Functional' reset event SW_FUNC has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Functional' reset event SW_FUNC has occurred. |
| 28<br>— | Reserved |
| 27<br>— | Reserved |
| 26<br>— | Reserved |
| 25<br>— | Reserved |
| 24<br>— | Reserved |
| 23<br>— | Reserved |
| 22<br>— | Reserved |
| 21<br>— | Reserved |
| 20<br>HSE_BOOT_RST | Flag for 'Functional' Reset HSE_BOOT_RST<br>0b - 'Functional' reset event HSE_BOOT_RST has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Functional' reset event HSE_BOOT_RST has occurred. |
| 19<br>— | Reserved |
| 18<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 17<br>— | Reserved |
| 16<br>HSE_SWT_RST | Flag for 'Functional' Reset HSE_SWT_RST<br>0b - 'Functional' reset event HSE_SWT_RST has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Functional' reset event HSE_SWT_RST has occurred. |
| 15<br>— | Reserved |
| 14<br>— | Reserved |
| 13<br>— | Reserved |
| 12<br>— | Reserved |
| 11<br>— | Reserved |
| 10<br>— | Reserved |
| 9<br>JTAG_RST | Flag for 'Functional' Reset JTAG_RST<br>0b - 'Functional' reset event JTAG_RST has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Functional' reset event JTAG_RST has occurred. |
| 8<br>— | Reserved |
| 7<br>— | Reserved |
| 6<br>SWT0_RST | Flag for 'Functional' Reset SWT0_RST<br>0b - 'Functional' reset event SWT0_RST has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Functional' reset event SWT0_RST has occurred. |
| 5 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 4<br>ST_DONE | Flag for 'Functional' Reset ST_DONE<br>0b - 'Functional' reset event ST_DONE has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Functional' reset event ST_DONE has occurred. |
| 3<br>FCCU_RST | Flag for 'Functional' Reset FCCU_RST<br>0b - 'Functional' reset event FCCU_RST has not occurred since either the last clear or the last power-on reset assertion.<br>1b - 'Functional' reset event FCCU_RST has occurred. |
| 2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>F_EXR | Flag for External Reset<br>External reset is a source of destructive reset.<br>0b - No external reset event has occurred since either the last clear or the last power-on reset assertion.<br>1b - An external reset event has occurred. |

### 32.8.4 Functional Event Reset Disable Register (FERD)

#### Offset

| Register | Offset |
|---|---|
| FERD | Ch |

#### Function

This register provides dedicated fields to disable functional reset sources. When any of these reset sources are disabled, the associated functional event is demoted to trigger an interrupt request. This register can be accessed as read/write in supervisor mode and read-only in user mode. Each byte can be written to only once after a destructive or power-on reset and this register is reset only on power-on and any destructive reset.

**NOTE**

It is important to clear the Functional /External Reset Status Register (FES) before writing 1 to any of the fields in this register. Otherwise a interrupt request may occur.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | D_JTAG_... | 0 | 0 | D_SWT0_... | 0 | 0 | D_FCCU_... | 0 | 0 | Reserved |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>D_DEBUG_FUNC | DEBUG_FUNC Disable Control<br>0b - Functional reset event DEBUG_FUNC triggers a reset sequence.<br>1b - Functional reset event DEBUG_FUNC generates an interrupt request. |
| 29<br>— | Reserved |
| 28<br>— | Reserved |
| 27<br>— | Reserved |
| 26<br>— | Reserved |
| 25<br>— | Reserved |
| 24<br>— | Reserved |
| 23<br>— | Reserved |
| 22 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 21<br>— | Reserved |
| 20<br>— | Reserved |
| 19<br>— | Reserved |
| 18<br>— | Reserved |
| 17<br>— | Reserved |
| 16<br>— | Reserved |
| 15<br>— | Reserved |
| 14<br>— | Reserved |
| 13<br>— | Reserved |
| 12<br>— | Reserved |
| 11<br>— | Reserved |
| 10<br>— | Reserved |
| 9<br>D_JTAG_RST | JTAG_RST Disable Control<br>0b - Functional reset event JTAG_RST triggers a reset sequence.<br>1b - Functional reset event JTAG_RST generates an interrupt request. |
| 8 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 7<br>— | Reserved |
| 6<br>D_SWT0_RST | SWT0_RST Disable Control<br>0b - Functional reset event SWT0_RST triggers a reset sequence.<br>1b - Functional reset event SWT0_RST generates an interrupt request. |
| 5<br>— | Reserved |
| 4<br>— | Reserved |
| 3<br>D_FCCU_RST | FCCU_RST Disable Control<br>0b - Functional reset event FCCU_RST triggers a reset sequence.<br>1b - Functional reset event FCCU_RST generates an interrupt request. |
| 2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>— | Reserved |

## 32.8.5 Functional Bidirectional Reset Enable Register (FBRE)

### Offset

| Register | Offset |
|---|---|
| FBRE | 10h |

### Function

This register enables the generation of an external reset on 'functional' reset.This register can be accessed as read/write in supervisor mode and read-only in user mode. This register is reset on power-on and any 'destructive' reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | BE_DEBU... | BE_SW_F... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BE_HSE_... | 0 | 0 | 0 | BE_HSE_... |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | BE_JTAG... | 0 | 0 | BE_SWT0... | 0 | BE_ST_D... | BE_FCCU... | 0 | 0 | Reserved |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>BE_DEBUG_FUNC | Bidirectional Reset Enables for 'Functional' Reset DEBUG_FUNC<br>0b - External reset pin is asserted on a 'Functional' reset DEBUG_FUNC event if the reset is enabled.<br>1b - External reset pin is not asserted on a 'functional' reset DEBUG_FUNC event. |
| 29<br>BE_SW_FUNC | Bidirectional Reset Enables for 'Functional' Reset SW_FUNC<br>0b - External reset pin is asserted on a 'Functional' reset SW_FUNC event if the reset is enabled.<br>1b - External reset pin is not asserted on a 'functional' reset SW_FUNC event. |
| 28<br>— | Reserved |
| 27<br>— | Reserved |
| 26<br>— | Reserved |
| 25<br>— | Reserved |
| 24<br>— | Reserved |
| 23 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 22<br>— | Reserved |
| 21<br>— | Reserved |
| 20<br>BE_HSE_BOOT_RST | Bidirectional Reset Enables for 'Functional' Reset HSE_BOOT_RST<br>0b - External reset pin is asserted on a 'Functional' reset HSE_BOOT_RST event if the reset is enabled.<br>1b - External reset pin is not asserted on a 'functional' reset HSE_BOOT_RST event. |
| 19<br>— | Reserved |
| 18<br>— | Reserved |
| 17<br>— | Reserved |
| 16<br>BE_HSE_SWT_RST | Bidirectional Reset Enables for 'Functional' Reset HSE_SWT_RST<br>0b - External reset pin is asserted on a 'Functional' reset HSE_SWT_RST event if the reset is enabled.<br>1b - External reset pin is not asserted on a 'functional' reset HSE_SWT_RST event. |
| 15<br>— | Reserved |
| 14<br>— | Reserved |
| 13<br>— | Reserved |
| 12<br>— | Reserved |
| 11<br>— | Reserved |
| 10 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 9<br>BE_JTAG_RST | Bidirectional Reset Enables for 'Functional' Reset JTAG_RST<br>0b - External reset pin is asserted on a 'Functional' reset JTAG_RST event if the reset is enabled.<br>1b - External reset pin is not asserted on a 'functional' reset JTAG_RST event. |
| 8<br>— | Reserved |
| 7<br>— | Reserved |
| 6<br>BE_SWT0_RST | Bidirectional Reset Enables for 'Functional' Reset SWT0_RST<br>0b - External reset pin is asserted on a 'Functional' reset SWT0_RST event if the reset is enabled.<br>1b - External reset pin is not asserted on a 'functional' reset SWT0_RST event. |
| 5<br>— | Reserved |
| 4<br>BE_ST_DONE | Bidirectional Reset Enables for 'Functional' Reset ST_DONE<br>0b - External reset pin is asserted on a 'Functional' reset ST_DONE event if the reset is enabled.<br>1b - External reset pin is not asserted on a 'functional' reset ST_DONE event. |
| 3<br>BE_FCCU_RST | Bidirectional Reset Enables for 'Functional' Reset FCCU_RST<br>0b - External reset pin is asserted on a 'Functional' reset FCCU_RST event if the reset is enabled.<br>1b - External reset pin is not asserted on a 'functional' reset FCCU_RST event. |
| 2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>— | Reserved |

### 32.8.6 Functional Reset Escalation Counter Register (FREC)

**Offset**

| Register | Offset |
|---|---|
| FREC | 14h |

### Function

This register provides the current value of functional reset escalation counter. It can be accessed in read/write, in supervisor mode. It can be accessed in read in the user mode. This register is reset by power-on reset, destructive reset, when you reconfigure the FREC field to Fh and when you write any value to the Functional Reset Escalation Threshold Register (FRET) register.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | FREC | | | |
| W | | | | | | | | | | | | | W1C | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>FREC | Functional' Reset Escalation Counter<br>This field provides the value of functional reset escalation counter. |

## 32.8.7 Functional Reset Escalation Threshold Register (FRET)

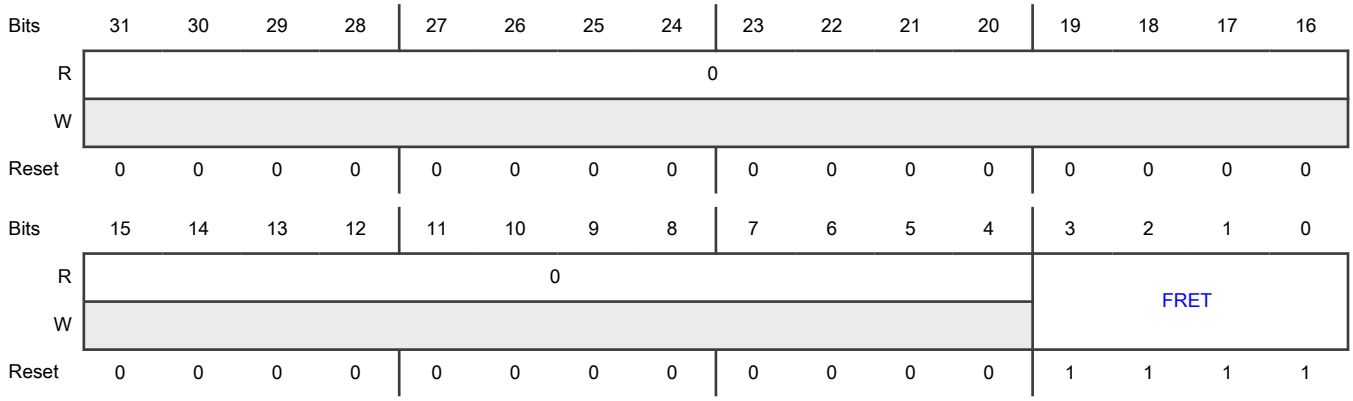
### Offset

| Register | Offset |
|---|---|
| FRET | 18h |

### Function

This register sets the threshold for 'functional' reset escalation to a 'destructive' reset. It can be accessed in read/write, either in supervisor mode. It can be accessed in read-only in the user mode. Writing a non-zero value to the FRET field enables the 'functional' reset escalation function. Writing any value to this register resets the 'functional' reset escalation counter. See Functional reset escalation for details on the 'functional' reset escalation function. This register is reset on power-on and any 'destructive' reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>FRET | 'Functional' Reset Escalation Threshold<br>If the value of this field is 0, the 'functional' reset escalation function is disabled. Any other value is the number of 'functional' resets that causes a 'destructive' reset. |

## 32.8.8 Destructive Reset Escalation Threshold Register (DRET)

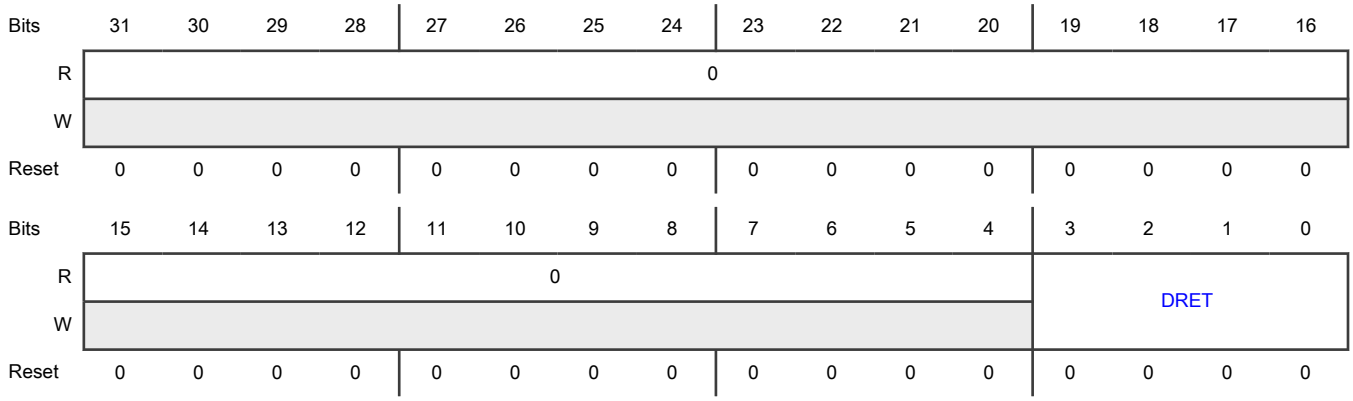
### Offset

| Register | Offset |
|---|---|
| DRET | 1Ch |

### Function

This register sets the threshold for 'destructive' reset escalation to keeping the chip in the reset state until the next power-on reset triggers a new reset sequence. It can be accessed in read/write, either in supervisor mode. It can be accessed in read-only in the user mode. Writing a non-zero value to the DRET field enables the 'destructive' reset escalation function. Writing any value to this register resets the 'destructive' reset counter. See Destructive reset escalation for details on the 'destructive' reset escalation function. This register is reset only on power-on reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>DRET | 'Destructive' Reset Escalation Threshold<br>If the value of this field is 0, the 'destructive' reset escalation function is disabled. Any other value is the number of 'destructive' resets which keeps the chip in the reset state until the next power-on reset triggers a new reset sequence. |

## 32.8.9 External Reset Control Register (ERCTRL)

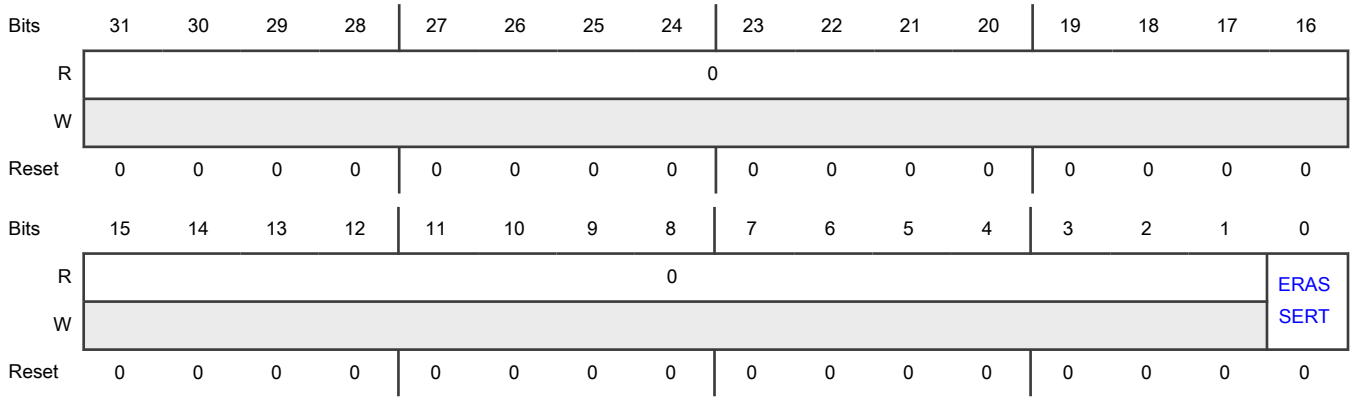
### Offset

| Register | Offset |
|---|---|
| ERCTRL | 20h |

### Function

This register allows software to control the assertion of External reset pin. It can be accessed in read/write, in supervisor mode. It can be accessed in read-only in the user mode. This register is reset on all resets.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>ERASSERT | ERASSERT<br>**NOTE**<br>Setting ERASSERT to 1b also safe/pad states most of the chip's pins. See the IOMUX table/spreadsheet for each pin's safe/pad state value. Software must use the ERASSERT field for this purpose only as part of the main reset domain self-test entry procedure. Using it at any other time may result in unpredictable system behavior.<br>0b - No change<br>1b - External reset is asserted |

## 32.8.10 Reset During Standby Status Register (RDSS)

### Offset

| Register | Offset |
|---|---|
| RDSS | 24h |

### Function

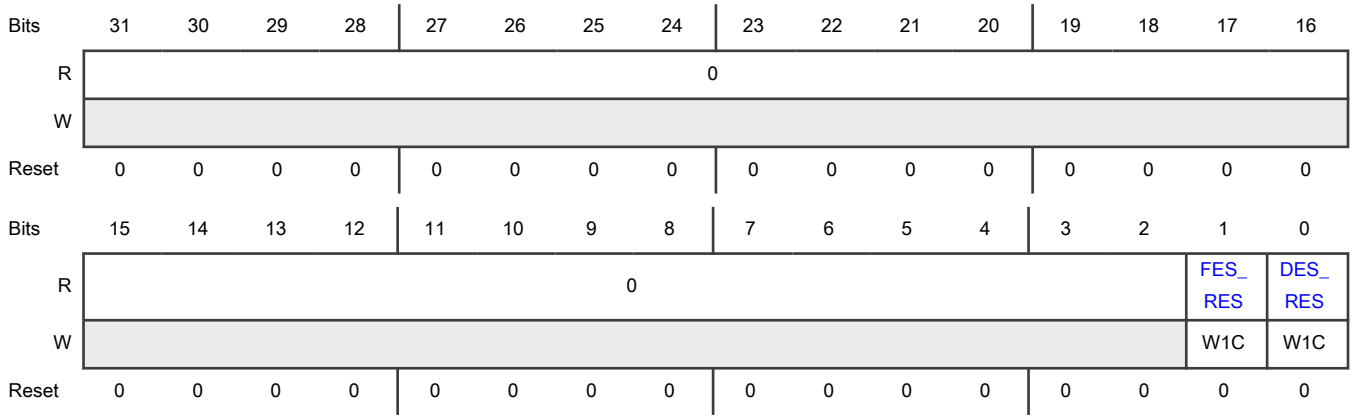
This register provides status of whether a reset event occurred during standby mode. Register bits are cleared on write '1'. This register is reset only on power-on reset.

**NOTE**

On exiting a reset sequence after standby exit, the software must perform a read operation on MC_ME[PREV_MODE] and RDSS register. If any field of the RDSS register is set, the software must ignore the status reported by MC_ME[PREV_MODE] register otherwise the status of MC_ME[PREV_MODE] register reports the device status.

If MC_ME indicates last mode as RESET, then perform a reset exit in software else a standby exit.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>FES_RES | FES_RES<br>0b - No functional reset event occurred during standby mode.<br>1b - Functional reset event occurred during standby mode. |
| 0<br>DES_RES | DES_RES<br>0b - No destructive reset event occurred during standby mode.<br>1b - Destructive reset event occurred during standby mode. |

## 32.8.11 Functional Reset Entry Timeout Control Register (FRENTC)

### Offset

| Register | Offset |
|---|---|
| FRENTC | 28h |

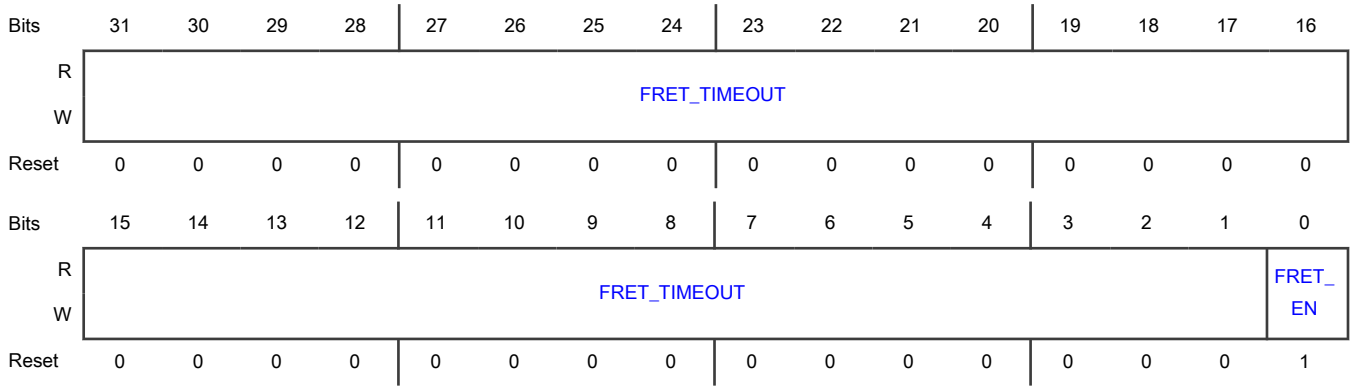
### Function

This register provides control for functional reset entry timeout counter. This register is reset on power-on and destructive reset events. This is a word accessible register. Byte accesses are not allowed on this register.

**NOTE**

This field should be set a value greater than one.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>FRET_TIMEOUT | Functional Reset Entry Timer Value<br>This field allows to override the timeout duration of the functional reset entry timer. The counter value is in terms of the Fast IRC clock cycles (typical clock frequency of 48 MHz). A non zero value of this field would indicate the timer value to be used during the functional reset entry sequence. |
| 0<br>FRET_EN | Functional Reset Entry Timer Enable/Disable<br>0b - Functional reset entry timer is disabled.<br>1b - Functional reset entry timer is enabled |

# Chapter 33
# Power-on Reset Watchdog (POR_WDG)

## 33.1 Introduction

POR_WDG monitors the chip for stuck or hang scenarios when in reset and standby sequences. It generates a chip power-on-reset event to recover the chip in case it remains stuck in reset or standby entry/exit duration for a pre-configured period of time.

POR_WDG consists of a watchdog counter with a configurable threshold. After the threshold is reached POR_WDG generates a chip power-on-reset event.

SIRC clocks POR_WDG.

## 33.2 Features

- Counter with four possible configurable threshold limits
- Associate register interfaces to capture the status of input signals
- Monitors reset, standby entry, and standby exit

## 33.3 Configurations

POR_WDG is enabled by default out of reset to actively monitor the chip reset sequence with the default timeout threshold duration. The POR_WDG configuration controls support provisions for enabling or disabling POR_WDG and the counter threshold, as described in the following sections.

### 33.3.1 Enable/disable configurations

By default POR_WDG is enabled for reset sequence monitoring, as well as standby entry/exit sequence monitoring for stuck scenarios. The following table describes the controls to enable or disable POR_WDG operation.

**Table 184. POR_WDG enable/disable control**

| POR_WDG monitoring | Default configuration | Enable/disable |
|---|---|---|
| Reset monitoring | Enabled | dcf_client_utest_misc[16]<br>Enables or disables POR_WDG for reset monitoring.<br>0: Disable<br>1: Enable<br>See the DCF clients file attached to this document for details. |
| Standby entry/exit monitoring | Disabled | DCM's DCMRWP1[8]<br>Enables or disables POR_WDG for standby entry/exit monitoring.<br>0: Enable<br>1: Disable<br>See the DCM chapter for details. |

### 33.3.2 Timeout configurations

POR_WDG supports four configurable counter threshold levels for its counter, as listed in the table below. The default POR_WDG timeout is 25 ms.

Table 185. POR_WDG timeout configuration

| DCM's DCMRWP1[10] | DCM's DCMRWP1[9] | POR_WDG timeout |
|---|---|---|
| 0 | 0 | 6.25 ms |
| 0 | 1 | 12.50 ms |
| 1 | 0 | 25.00 ms |
| 1 | 1 | 50.00 ms |

### 33.3.3 Event and status registers

POR_WDG provides indication of a POR_WDG event along with the status of the chip at the point when the POR_WDG counter overflows.

*POR_WDG event status register*: A POR_WDG event is captured in DCM status register DCMROPP4[0].

*POR_WDG status registers*: The status of the chip at the POR_WDG event is captured in DCM status registers DCMROPP1–DCMROPP3. See the DCM chapter for details.

DCM's DCMROPP*n* are reset on a POR caused by PMC and are unaffected by a POR_WDG reset.

## 33.4 Reset sequence monitoring

### 33.4.1 Introduction

At power-up POR_WDG is enabled for reset monitoring (DCF record in UTEST Misc can bypass POR_WDG, as described in Table 184).

On every reset event, POR_WDG starts monitoring the reset sequence (including the functional reset entry sequence, in case of functional reset event).

If the chip does not exit the reset sequence within the POR_WDG timeout threshold, then POR_WDG initiates a POR sequence to recover the chip from a stuck or hung scenario. You can view the POR_WDG status in DCM's DCMROPPn registers to obtain:

- Diagnostic status
- Chip details in the event of a POR_WDG reset

### 33.4.2 Inactive windows or bypass operation

POR_WDG monitoring of the reset sequence is inactive in the following conditions:

- POR_WDG is disabled for reset monitoring by chip configuration in UTEST via dcf_client_utest_misc[16]. See the DCF clients file attached to this document for details.
- MC_RGM destructive reset escalation (ensures that the chip stays in reset).
- Extend pin reset by keeping it pressed externally.
- During selftest, when MC_RGM.RGM_ERCTRL[ERASSERT] is configured as 1.

### 33.4.3 Windows of operation

POR_WDG monitors the reset sequence in the following scenarios:

- Reset pin is active (when chip is in reset, including the functional reset entry sequence).
- Standby domain functional reset is asserted (when reset pad is pulled up but functional reset is asserted).
- RUN domain destructive reset is asserted but flash memory is not in Low-Power mode (when RUN domain destructive reset fires unexpectedly in Run mode).

- RUN domain functional reset is asserted but flash memory is not in Low-Power mode (when RUN domain functional reset fires unexpectedly in RUN mode).

## 33.5 Standby entry/exit sequence monitoring

### 33.5.1 Introduction

On every low-power entry or exit event, POR_WDG starts monitoring the low-power sequence (standby entry and standby exit sequence).

If the chip does not exit the standby entry or exit sequence within the POR_WDG timeout threshold, POR_WDG initiates a POR sequence to recover the chip from the stuck or hung scenario. You can then check the POR_WDG status registers in DCM to check the chip status when POR_WDG raised the POR to the chip.

### 33.5.2 Inactive windows/bypass operation

You can configure DCM's DCMRWP1[8] to disable POR_WDG for standby entry and exit sequence monitoring.

### 33.5.3 Windows of operation

POR_WDG monitors the standby sequence (standby entry and standby exit) in these scenarios:

- Standby entry sequence monitoring: Monitoring from main core WFI execution until PMC acknowledges Low-Power mode entry.
- Standby exit sequence monitoring: Monitoring from wakeup event until the RUN domain reset recovery.

## 33.6 Glossary

| | |
|---|---|
| **WFI** | Wait for Interrupt |

# Chapter 34
# Security Overview

## 34.1 Introduction

This chip has a comprehensive set of customer-configurable security features designed to protect code and data from unauthorized access. The content of this section is for non-secure operation only. Contact your NXP representative for details if you need secure operation.

## 34.2 Security features

MCXE31:

- Exceeds the EVITA- full [5] functionality and offers high performance for edge nodes applications.
- Bases its chip censorship on the life cycle model. Access to chip code and data becomes progressively more restricted as the chip matures through a defined set of life cycle steps.
- Offers these memory security features:
  - NVM censorship support
  - Debug password protection
  - OTP flash memory areas
- Supports a unique ID—The chip has a unique ID stored in an OTP flash memory area. Any core on the chip can read this unique ID.
- Includes ELE_HSEB, which is a dedicated security system providing:
  - A processor core
  - Dedicated SRAM
  - Symmetric Hardware Accelerator
  - Asymmetric Hardware Accelerator
  - True Random Number Generator (TRNG)
  - Pseudo Random Number Generator (PRNG)
  - Exclusive access to secure areas of the chip's flash memory
  - OTA : In Field Secure Code/Data updates

  **NOTE**

  Refer Security Reference Manual and Firmware Reference Manual for the details on OTA.

  Also, ELE_HSEB runs NXP firmware independently from the main chip processor cores and can implement advanced security and monitoring functions.
- Supports a secure debugger interface.
- Provides boot modes:

[5] EVITA was a European research project existing from 2008 to 2011. There is no standard or a specification to test compliancy. However, ELE_HSEB meets the common expectations often described in the industry as EVITA Full and exceeds them in these cases:
- ELE_HSEB supports up to AES-256 instead of AES-128
- ELE_HSEB supports up to ECC-521 instead of ECC-256
- ELE_HSEB supports SHA-2/Miyaguchi-Preneel instead of Whirlpool

  - Trusted and secure boot support
  - Handshake that supports BAF
- Monitors operating conditions to maximize tampering resistance.
- Includes basic debugger restrictions (on and off via censorship mode).

When ELE_HSEB detects a security failure, it moves the chip to a secure state. The chip secure states comply with the chip safe states. You enter a safe state as reset when there are security errors leading to reset. Moreover, if the chip stays in Run mode and the lifecycle moves to the IN_FIELD phase in reaction to errors, the safety application can continue running without glitches.

## 34.3 Security information

Chip security feature details are published in the HSE firmware reference manual and HSE service interface manual. Contact your NXP sales representative for more information.

## 34.4 Glossary

| | |
|---|---|
| **AES** | Advanced encryption standard |
| **BAF** | Boot assist flash |
| **ECC** | Elliptic curve cryptography |
| **EVITA** | E-safety vehicle intrusion protected applications |
| **NVM** | Nonvolatile memory |
| **OTP** | One-time programmable |

# Chapter 35
# EdgeLock Secure Enclave (HSE_B)

## 35.1 HSE subsystem

HSE is a security subsystem. It runs security functions for applications having stringent confidentiality and authenticity requirements. HSE has the following objectives:

- Isolating security-sensitive information (for example, secret keys) from the application (the host)
- Transferring the cryptographic operations from application cores and processing them
- Accelerating cryptographic operations with dedicated coprocessors
- Enforcing security measures on the application, during runtime and system startup

The HSE subsystem is the only master that is unconditionally released from reset after POR. It then releases the CPU subsystems in the host from reset, with the opportunity to apply certain checks beforehand (secure system startup). Based on certain conditions, HSE can also trigger interrupts and reset signals to the host during runtime.

### 35.1.1 CPU subsystem

The HSE CPU subsystems process the security functions and control system resources to provide security services to the application domain (the host).

### 35.1.2 Cryptographic accelerators

The HSE subsystem supports the following cryptographic accelerators:

- An AES engine supporting all standard key sizes (128, 192, 256 bits) and various complex ciphering modes (ECB, CBC, CTR, OFB, CFB, CCM, CMAC)
- A hash engine that supports standard several primitives: MD5, SHA-1, SHA-224, SHA-256
- PKC which accelerates RSA and ECC operations (key generation, signature generation, signature verification, ciphering)
  - RSA (1024, 2048, 3072, 4096-bit)
  - ECC over prime numbers
    - BN P256, P638
    - ANSI X9P192 to X9P512
    - Brainpool P160 to P512
    - Sec P128 to P521
    - TU Darmstadt prime Curve 1 to 35
    - Ed25519

The HSE subsystem supports several other cryptographic primitives through the software. See HSE firmware for more information.

### 35.1.3 Random number generator (RNG)

The RNG in this chip consists of a TRNG and a DRBG. Both are designed to be compliant to the highest strength in security as specified in

- BSI AIS20/31
- NIST SP800-90a,b,c

The TRNG function provides a seed for the DRBG, while the DRBG is available to the host via dedicated random number generation services.

### 35.1.4 Timers

The HSE subsystem features:

- An independent dedicated system timer
- HSE_STM (apart from chip timer resources), that allows recurring autonomous functions such as runtime memory verification checks
- A watchdog timer to reset the HSE subsystem in case of unexpected runtime failure

### 35.1.5 Memory resources

See Configuration_GPR memory map for more information on configuration controls related to HSE functions.

#### 35.1.5.1 Secure RAM

Secure RAM refers to RAM area that the HSE subsystem access exclusively. The HSE firmware use it to operate as well as to store a copy of the cryptographic keys in service.

#### 35.1.5.2 Secure NVM

Secure NVM refers to nonvolatile memory that the HSE subsystem accesses exclusively. It stores the HSE firmware and the security assets (for example, keys and private data).

The size of the secure NVM is configurable by the HSE firmware.

## 35.2 HSE interface

### 35.2.1 Messaging unit (MU)

The HSE subsystem has two messaging units:

- HSE_MU0
- HSE_MU1

See the "Messaging Unit (MU)" chapter for more information.

#### 35.2.1.1 Overview

MU is the communication interface between the host and the HSE subsystem. The host uses MU to trigger service requests and to receive service responses. The HSE firmware uses MU to receive service requests, return service responses, and provide a general status of the HSE subsystem.

Each of the two MU instances available in the HSE subsystem has:

- Two sides:
  - MUA: Only the HSE subsystem accesses it.
  - MUB: The host accesses it.

    The registers on one side have corresponding registers on the other side.
- A set of 32-bit readable and writable transmit registers (MUB_TRn), which the host uses to transfer the address of the service descriptor to the HSE firmware. The HSE firmware retrieves the address of the service descriptor in the corresponding registers MUA_RRn.
- A set of 32-bit read-only receive registers (MUB_RRn), which the host uses to retrieve the response to the service requests. The HSE firmware provides the response to service requests in the corresponding registers MUA_TRn.

- Control and status registers to manage MU and the access to transmit and receive registers.

The advantages of using the MU to manage the HSE service requests and responses are:

- Hardware mechanisms are in place on the transmit registers to avoid service request overrun.
- Interrupt signals are available to allow asynchronous management of the requests (avoiding active waiting loops).
- Freedom from interference between different application cores. You can configure each MU instance with specific access restrictions that can be used, for example, to isolate the requests that different masters make (in different MU instances). You configure such access controls using XRDC.

### 35.2.2 HSE interface RAM

The HSE interface RAM (also known as host interface RAM) is a secure RAM area. The host accesses it with the restrictions that HSE set via XRDC. This RAM area is a part of chip RAM and the size is configurable by secure core using CONFIG_RAMPR[SECURE_SIZE].

## 35.3 Debug

### 35.3.1 HSE subsystem debug

The debugging of the HSE subsystem and associated firmware is restricted to NXP engineering teams.

### 35.3.2 Host debug

The host debug is either open or protected, depending on the device Lifecycle state. See the ‘Life cycle’ section in ‘Device Configuration Module’ chapter for details on device lifecycle advancement and decoding. The details of the device security features is published in a separate document, the Security Reference Manual. Please contact your NXP sales representative for details. See the Debug chapter for details on host/application core debug.

The debug protection consists of locking the debugger access through the JTAG interface until the HSE firmware authenticates the debugger. This authentication is based on ADK/P, a 16-byte region within UTEST used for application core debug. See UTEST memory map in the DCF clients file attached to this document.

The authentication method can be:

- Static: In this case, ADK/P is a password which the debugger provides in plain form.
- Dynamic: In this case, ADK/P is a cryptographic key which the debugger uses to calculate a cryptographic response to a random challenge.

## 35.4 HSE firmware

Factory supplied firmware that runs in the HSE subsystem controls HSE functionality. It essentially serves the host with a set of security services as described in Table 186.

**Table 186. Summary of firmware security services**

| Service | Summary |
|---|---|
| Administration | Install, configure, update, and test the HSE firmware |
| Key management | Available for the application to manage different sets of keys that the HSE firmware manages, for example, cryptographic services |
| Cryptographic | Provide the application with cryptographic primitives that high level security stacks use in the application |
| Random number | Generate random streams that can be used in various security protocols |

*Table continues on the next page...*

**Table 186. Summary of firmware security services (continued)**

| Service | Summary |
|---|---|
| Memory verification | Allow the application to verify different memory areas at startup (after reset) and during runtime |
| Monotonic counter | Provide the application with a set of monotonic counters that can be read and only incremented |
| Secure time | Allow the configuration of a secure tick to be signaled to the application |

Table 187 provides an overview of services and features that the HSE firmware supports.

**Table 187. HSE firmware features**

| Service | Category | Feature |
|---|---|---|
| Cryptography | Ciphers | AES: ECB, CBC, CFB, OFB, CTR, XTS[1]<br>RSAES: PKCS1-v1_5, OAEP |
| | Message Authentication Code (MAC) | AES: CMAC, XCBC-MAC[1], HMAC[1], and GMAC[1] |
| | Hashing | SHA1<br>SHA224, SHA256, SHA384, SHA512<br>SHA3_224[1], SHA3_256[1], SHA3_384[1], SHA3_512[1]<br>MD5<br>Miyaguchi-Preneel Compression |
| | Authenticated ciphers | AES: CCM, GCM[1] |
| | Digital signature generation and verification | RSASSA_PSS<br>RSASSA_PKCS1-v1_5<br>ECDSA – ECC over GF(P) with all prime standard curve supported<br>EdDSA - Ed25519 pre-hashed curve |
| Key management | Max key sizes | AES: 256 bits<br>RSA: 4096 bits<br>ECC: 521 bits<br>DH: 4096 bits |
| | Key generation | Permanent and ephemeral RSA and ECC key pair generation |
| | Key import or export | Plain or encrypted form, with optional authentication tag.<br>SHE key update protocol |
| | Key derivation | NIST 800-108, PBKDF2, and so on |
| | Key exchange | ECDH and Classic DH |
| | Certificate handling | Key Installation from X.509 and CVC certificates<br>Certificate installation for Root of Trust establishment. |

*Table continues on the next page...*

Table 187. HSE firmware features (continued)

| Service | Category | Feature |
|---|---|---|
| Boot and memory verification | Authentication schema | AES CMAC, XCBC-MAC<br>RSA & ECC signatures |
| | Verification flow | Before application startup (strict secure boot)<br>In parallel to the application startup<br>On the application demand |
| | Sanctions | No startup (strict secure boot)<br>Chip reset<br>Key usage restrictions |
| Monotonic counter | Counter management | Incrementing and reading volatile and non-volatile counters |
| Random number | Deterministic random bit generation | Based on a True Random Number<br>AIS31 Class P2 high and FIPS 140-2 compliant |
| Secure time | Secure tick | Application interrupts at configurable frequency |
| Administration services | HSE administration | Firmware installation and update<br>Subsystem configuration and testing |

1. Software implementation of cryptographic primitives.

See documents in Table 188 for more information about how to install, configure, and use the HSE firmware that NXP provides.

Table 188. References

| Document number | Document title | Description |
|---|---|---|
| HSEFWRM | HSE Firmware Reference Manual | Contains details about how to install, configure, and use the HSE subsystem. |
| HSESIRM | HSE Service Interface Reference Manual | Security firmware API reference for non-AUTOSAR users. |

## 35.5 General purpose registers with write access by HSE core

This is a general purpose register bank that only the HSE core can write. Secure master ID gating allows only the secure core to write in these registers, whereas any application core can read.

## 35.6 Configuration_GPR register descriptions

### 35.6.1 Configuration_GPR memory map

This section describes the chip configurations that only the HSE core manages. These constitute control of peripherals, secure size configurations for SRAM and flash memory, flash memory program or erase control, and so on.

**NOTE**

Write accesses to configuration registers apart from 32-bit accesses might result in unpredictable chip behavior, therefore must not be done.

Configuration base address: 4039_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1Ch | General Purpose Configuration 0 (CONFIG_REG0) | 32 | R | 0000_0000h |
| 34h | General Purpose Configuration 6 (CONFIG_REG6) | 32 | R | 0000_0035h |
| 38h | Configuration RAM Protected Region (CONFIG_RAMPR) | 32 | R | See section |
| 3Ch | Configuration Code Flash Memory Active Block (CONFIG_CFPRL) | 32 | R | See section |
| 40h | Configuration Code Flash Memory Passive Block (CONFIG_CFPRH) | 32 | R | See section |
| 44h | Configuration Data Flash Memory Protected Region (CONFIG_DFPR) | 32 | R | See section |
| 50h | Configuration Program and Erase Lock (CONFIG_PE_LOCK) | 32 | R | 0000_0000h |
| 54h | Configuration RAM Protected Region Alternate (CONFIG_RAMPR_ALT) | 32 | R | See section |
| 58h | Configuration Code Flash Memory Active Block Alternate (CONFIG_CFPRL_ALT) | 32 | R | See section |
| 5Ch | Configuration Code Flash Memory Passive Block Alternate (CONFIG_CFPRH_ALT) | 32 | R | See section |
| 60h | Configuration Data Flash Memory Protected Region Alternate (CONFIG_DFPR_ALT) | 32 | R | See section |
| 64h | Configuration REG_GPR (CONFIG_REG_GPR) | 32 | RW | A000_0003h |

## 35.6.2 General Purpose Configuration 0 (CONFIG_REG0)

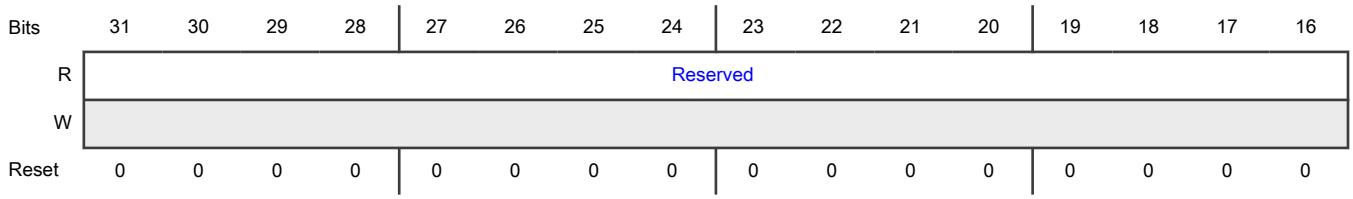
### Offset

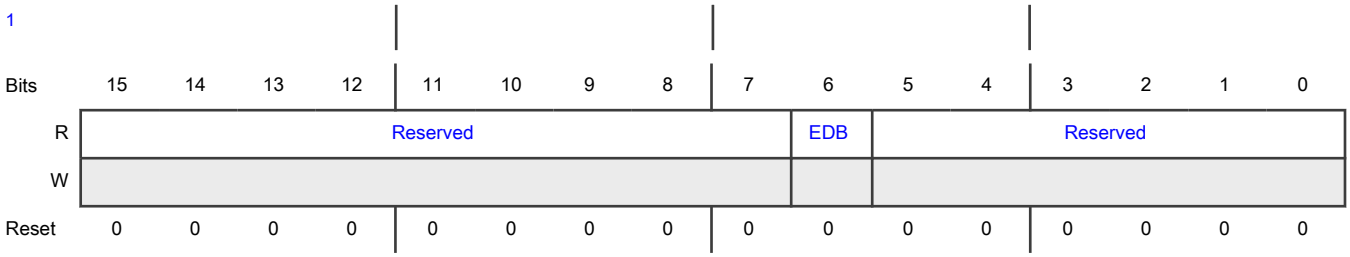
| Register | Offset |
|---|---|
| CONFIG_REG0 | 1Ch |

### Function

The EDB field resets on destructive or POR reset events and functional reset has no impact on it.

### Diagram





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

1. If export_control=1, the reset value of EDB is 1. If export_control=0, the reset value is 0.

**Fields**

| Field | Function |
|---|---|
| 31-7<br>— | Reserved |
| 6<br>EDB | Hardware Debugger Attached<br>This is a sticky field that clears on destructive reset.<br>0b - Debugger not connected<br>1b - Debugger connected |
| 5-0<br>— | Reserved |

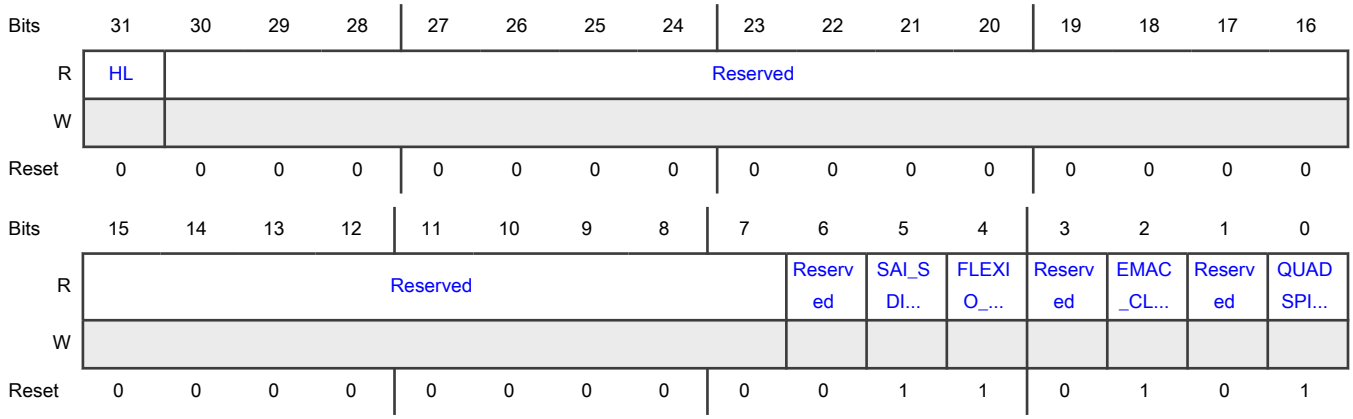
## 35.6.3 General Purpose Configuration 6 (CONFIG_REG6)

**Offset**

| Register | Offset |
|---|---|
| CONFIG_REG6 | 34h |

**Function**
Resets on functional reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>HL | Hard Lock<br>Only functional reset clears this field.<br>0b - You can write to this register<br>1b - Register is locked for any write |
| 30-7<br>— | Reserved |
| 6<br>— | Reserved |
| 5<br>SAI_SDID_PCTL | SAI0 and SAI1 clock gating<br>Clock to SAI peripheral is on or off.<br>0b - Clock is off (gated)<br>1b - Clock is on |
| 4<br>FLEXIO_CLOCK_GATE | FlexIO Clock Gating<br>Clock to FlexIO peripheral is on or off.<br>0b - Clock is off (gated)<br>1b - Clock is on |
| 3<br>— | Reserved |
| 2<br>EMAC_CLOCK_GATE | Ethernet Clock Gating<br>Clock to Ethernet peripheral is on or off. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Clock is off (gated)<br>1b - Clock is on |
| 1<br>— | Reserved |
| 0<br>QUADSPI_SDID_PCTL | QuadSPI Clock Gating<br>Clock to QuadSPI peripheral is on or off.<br>0b - Clock is off (gated)<br>1b - Clock is on |

## 35.6.4 Configuration RAM Protected Region (CONFIG_RAMPR)

### Offset

| Register | Offset |
|---|---|
| CONFIG_RAMPR | 38h |

### Function

Resets on functional reset.

**NOTE**

The secure size will go to default or reset value on assertion of any DCF violation from HSE.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | HARD_LO... | SOFT_LO... | Reserved | | | | | | | | | | SECURE_SIZE | | | |
| W | | | | | | | | | | | | | | | | |
| Reset [1] | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SECURE_SIZE | | | | | | | | | | | | Reserved | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

1. The default reset value of this register varies on the part basis, depending on NXP factory configurations.

### Fields

| Field | Function |
|---|---|
| 31<br>HARD_LOCK | Hard Lock<br>This is a sticky field. If you write 1 to it, it remains 1 until next reset.<br>0b - Write access is allowed<br>1b - Write access is not allowed until next functional reset |
| 30<br>SOFT_LOCK | Soft Lock<br>0b - Write access is allowed<br>1b - Write access is not allowed |
| 29-21<br>— | Reserved |
| 20-5<br>SECURE_SIZE | Secure Size<br>Secure size region (in bytes) for PRAM1. This is 32-byte-aligned to ensure alignment with cache lines. |
| 4-0<br>— | Reserved |

## 35.6.5 Configuration Code Flash Memory Active Block (CONFIG_CFPRL)

### Offset

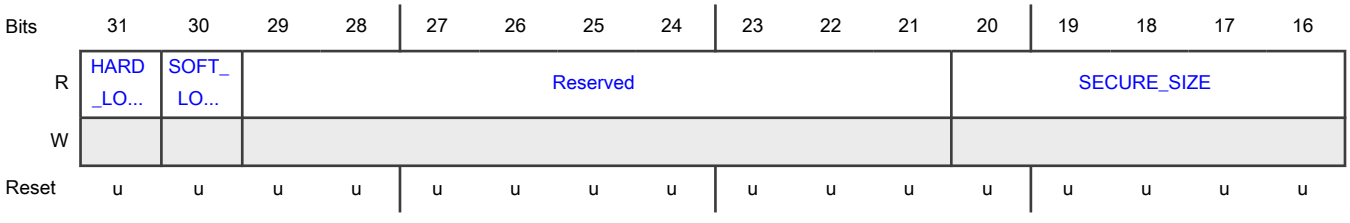
| Register | Offset |
|---|---|
| CONFIG_CFPRL | 3Ch |

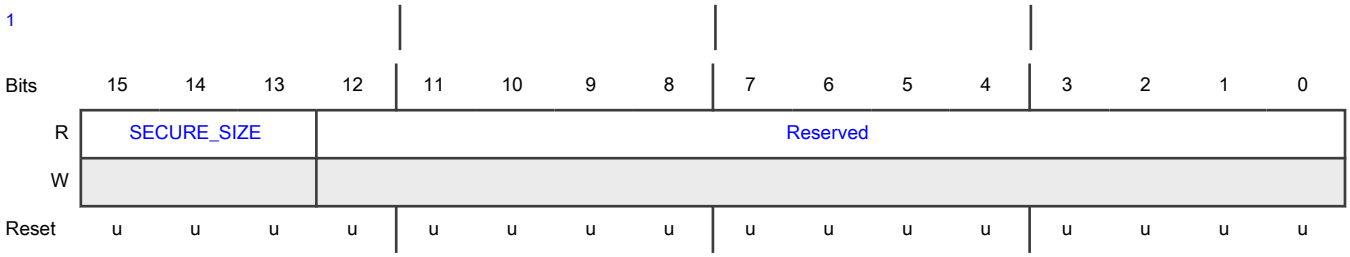
### Function
Resets on functional reset.

**NOTE**

The secure size will go to default or reset value on assertion of any DCF violation from HSE.

### Diagram






1. The default reset value of this register varies on the part basis, depending on NXP factory configurations.

**Fields**

| Field | Function |
|---|---|
| 31<br>HARD_LOCK | Hard Lock<br>After this field is configured it cannot be modified until a reset event.<br>0b - Write access is allowed<br>1b - Write access is not allowed until next functional reset |
| 30<br>SOFT_LOCK | Soft Lock<br>0b - Write access is allowed<br>1b - Write access is not allowed |
| 29-21<br>— | Reserved |
| 20-13<br>SECURE_SIZE | Secure Size<br>Flash memory active block secure size in bytes to align to 8 KB (sector) aligned write. |
| 12-0<br>— | Reserved |

## 35.6.6 Configuration Code Flash Memory Passive Block (CONFIG_CFPRH)

**Offset**

| Register | Offset |
|---|---|
| CONFIG_CFPRH | 40h |

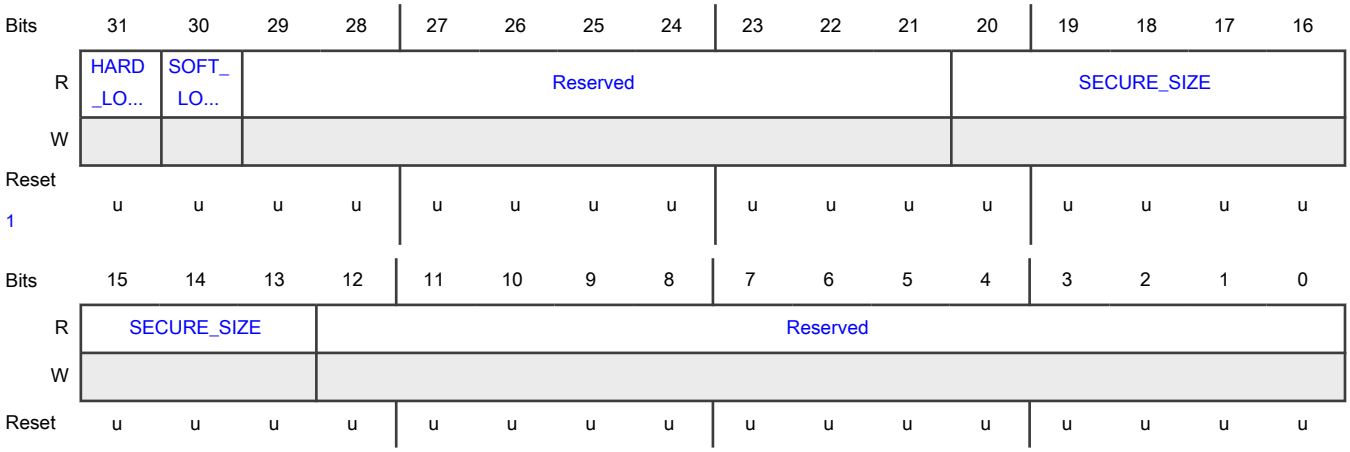
**Function**

Resets on functional reset.

**NOTE**

The secure size will go to default or reset value on assertion of any DCF violation from HSE.

### Diagram




1. The default reset value of this register varies on the part basis, depending on NXP factory configurations.

### Fields

| Field | Function |
|---|---|
| 31<br>HARD_LOCK | Hard Lock<br>After this field is configured, it cannot be modified until a reset event.<br>0b - Write access is allowed<br>1b - Write access is not allowed until next functional reset |
| 30<br>SOFT_LOCK | Soft Lock<br>0b - Write access is allowed<br>1b - Write access is not allowed |
| 29-21<br>— | Reserved |
| 20-13<br>SECURE_SIZE | Secure Size<br>Secure size region (in bytes) for flash memory passive block for alignment with 8 KB (sector) aligned writes. |
| 12-0<br>— | Reserved |

## 35.6.7 Configuration Data Flash Memory Protected Region (CONFIG_DFPR)

### Offset

| Register | Offset |
|---|---|
| CONFIG_DFPR | 44h |

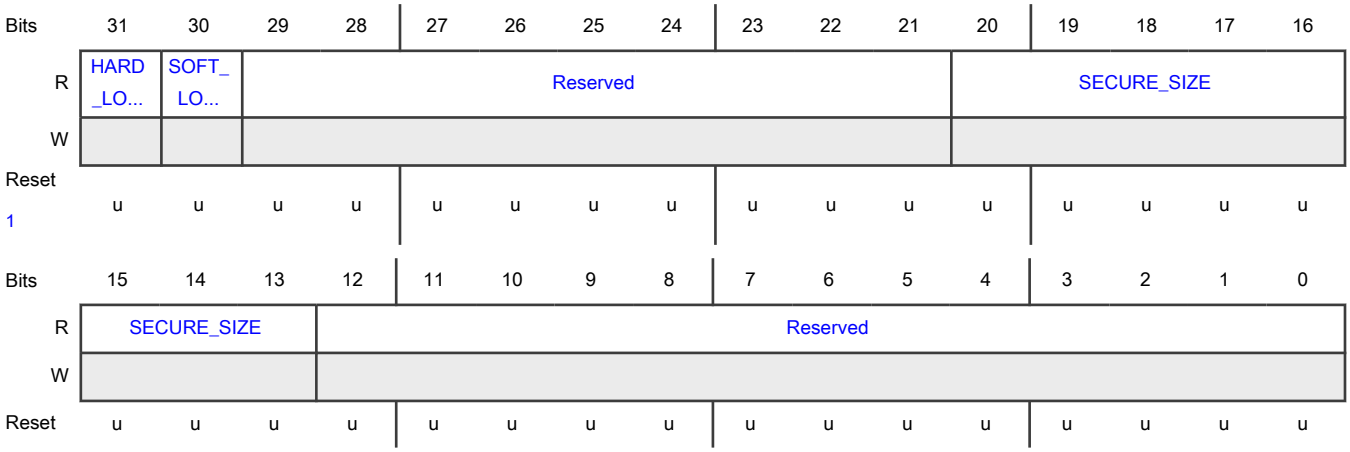
### Function

Resets on functional reset.

**NOTE**

The secure size will go to default or reset value on assertion of any DCF violation from HSE.

### Diagram




1. The default reset value of this register varies on the part basis, depending on NXP factory configurations.

### Fields

| Field | Function |
|---|---|
| 31<br>HARD_LOCK | Hard Lock<br>After this field is configured, it cannot be modified until a reset event.<br>0b - Write access is allowed<br>1b - Write access is not allowed until next functional reset |
| 30<br>SOFT_LOCK | Soft Lock<br>0b - Write access is allowed<br>1b - Write access is not allowed |
| 29-21<br>— | Reserved |
| 20-13<br>SECURE_SIZE | Secure Size<br>Secure size region (in bytes) for data flash memory aligned to 8KB (sector) aligned writes. |
| 12-0<br>— | Reserved |

## 35.6.8 Configuration Program and Erase Lock (CONFIG_PE_LOCK)

### Offset

| Register | Offset |
|---|---|
| CONFIG_PE_LOCK | 50h |

### Function

Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | | | | | | PE_LOCK... | Reserved | PE_LOCK... |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PE_LOCK... | PE_LOCK... | PE_LOCK... | PE_LOCK... | Reserved | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18<br>PE_LOCK_UTEST | Program/Erase Lock for UTEST<br>0b - UTEST lock request is clear<br>1b - UTEST lock requested |
| 17<br>— | Reserved |
| 16<br>PE_LOCK_BLOCK_4 | Program/Erase Lock for Block 4<br>0b - Block 4 lock request is clear<br>1b - Block 4 lock requested |
| 15<br>PE_LOCK_BLOCK_3 | Program/Erase Lock for Block 3<br>0b - Block 3 lock request is clear<br>1b - Block 3 lock requested |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 14<br>PE_LOCK_BLOCK_2 | Program/Erase Lock for Block 2<br>0b - Block 2 lock request is clear<br>1b - Block 2 lock requested |
| 13<br>PE_LOCK_BLOCK_1 | Program/Erase Lock for Block 1<br>0b - Block 1 lock request is clear<br>1b - Block 1 lock requested |
| 12<br>PE_LOCK_BLOCK_0 | Program/Erase Lock for Block 0<br>0b - Block 0 lock request is clear<br>1b - Block 0 lock requested |
| 11-0<br>— | Reserved |

## 35.6.9 Configuration RAM Protected Region Alternate (CONFIG_RAMPR_ALT)

### Offset

| Register | Offset |
|---|---|
| CONFIG_RAMPR_ALT | 54h |

### Function

Resets on functional reset.

**NOTE**

The secure size will go to default or reset value on assertion of any DCF violation from HSE.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | INVERT_VALUE_RAMPR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset [1] | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

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

1. The default reset value of this register varies on the part basis, depending on NXP factory configurations.

### Fields

| Field | Function |
|---|---|
| 31-0<br>INVERT_VALUE_RAMPR | Invert Value RAMPR<br>Write inverted value of register CONFIG_RAMPR to CONFIG_RAMPR_ALT. |

## 35.6.10 Configuration Code Flash Memory Active Block Alternate (CONFIG_CFPRL_ALT)

### Offset

| Register | Offset |
|---|---|
| CONFIG_CFPRL_ALT | 58h |

### Function
Resets on functional reset.

**NOTE**

The secure size will go to default or reset value on assertion of any DCF violation from HSE.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | INVERT_VALUE_CFPRAB | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset [1] | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

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

1. The default reset value of this register varies on the part basis, depending on NXP factory configurations.

### Fields

| Field | Function |
|---|---|
| 31-0<br>INVERT_VALUE_CFPRAB | Invert Value CFPRAB<br>Write inverted value of register CONFIG_CFPRL to CONFIG_CFPRL_ALT. |

## 35.6.11 Configuration Code Flash Memory Passive Block Alternate (CONFIG_CFPRH_ALT)

### Offset

| Register | Offset |
|---|---|
| CONFIG_CFPRH_ALT | 5Ch |

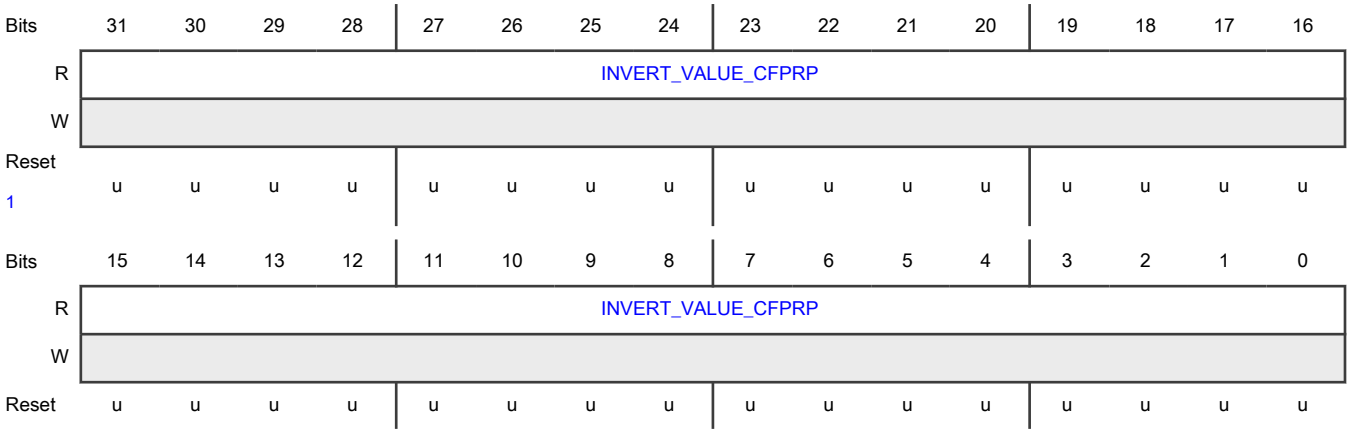
### Function

Resets on functional reset.

**NOTE**

The secure size will go to default or reset value on assertion of any DCF violation from HSE.

### Diagram




1. The default reset value of this register varies on the part basis, depending on NXP factory configurations.

### Fields

| Field | Function |
|---|---|
| 31-0<br>INVERT_VALUE_CFPRP | Invert Value CFPRP<br>Write inverted value of register CONFIG_CFPRH to CONFIG_CFPRH_ALT. |

## 35.6.12 Configuration Data Flash Memory Protected Region Alternate (CONFIG_DFPR_ALT)

### Offset

| Register | Offset |
|---|---|
| CONFIG_DFPR_ALT | 60h |

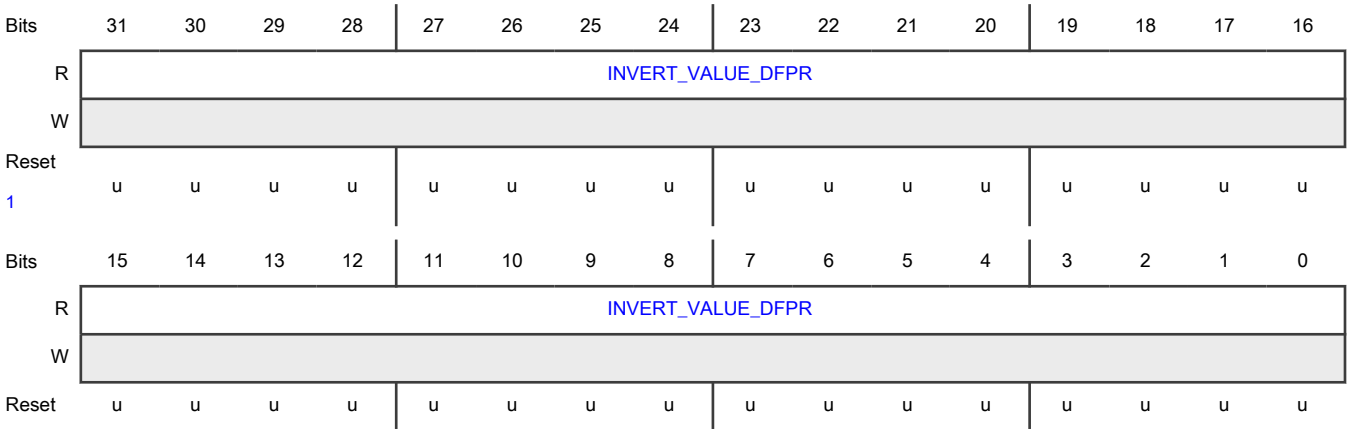
### Function

Resets on functional reset.

**NOTE**

The secure size will go to default or reset value on assertion of any DCF violation from HSE.

### Diagram




1. The default reset value of this register varies on the part basis, depending on NXP factory configurations.

### Fields

| Field | Function |
|---|---|
| 31-0<br>INVERT_VALUE_DFPR | Invert Value CFPRP<br>Write inverted value of register CONFIG_DFPR to CONFIG_DFPR_ALT. |

## 35.6.13 Configuration REG_GPR (CONFIG_REG_GPR)

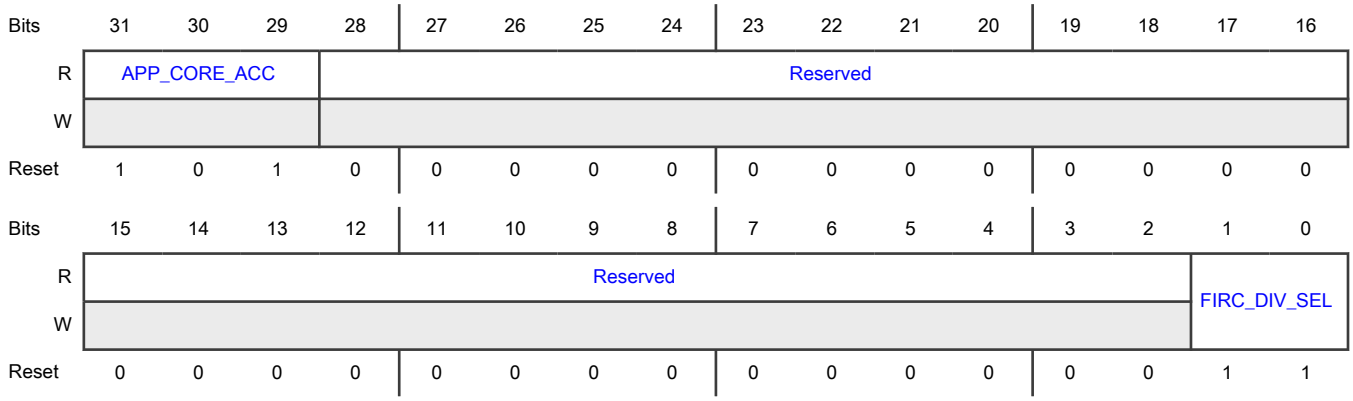
### Offset

| Register | Offset |
|---|---|
| CONFIG_REG_GPR | 64h |

### Function

Resets on functional reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-29<br>APP_CORE_ACC | FIRC Divider<br>**NOTE**<br>While writing to this register, APP_CORE_ACC is RO and should not be changed from 0b101.<br>101b - Application core can write this field [FIRC_DIV_SEL]<br>All other values - No access to application core |
| 28-2<br>— | Reserved |
| 1-0<br>FIRC_DIV_SEL | FIRC Divider<br>Indicates this chip's FIRC clock division factor.<br>00b - Divided by 2<br>01b - Divided by 2<br>10b - Divided by 16<br>11b - Undivided |

## 35.7 Glossary

| | |
|---|---|
| **ADK/P** | Application debug key/password |
| **AES** | Advanced encryption standard |
| **CBC** | Cipher block chaining |
| **CCM** | Counter with CBC MAC (Cipher block chaining message authentication code) |
| **CFB** | Cipher feedback mode |
| **CMAC** | Cipher-based message authentication code |
| **CTR** | Counter-based block cipher mode |

| | |
|---|---|
| **CVC** | Card verifiable certificates |
| **Classic DH** | Classical Diffie–Hellman, a key exchange method |
| **DRBG** | Deterministic random bit generator |
| **ECB** | Electronic code book |
| **ECC** | Elliptic curve cryptography |
| **ECDSA** | Elliptic curve digital signature algorithm |
| **ECDH** | Elliptic-curve Diffie–Hellman, a key exchange method |
| **EdDSA** | Edwards-curve digital signature algorithm |
| **FIPS** | Federal information processing standards |
| **MUA** | Messaging unit A interface |
| **MUB** | Messaging unit B interface |
| **NIST** | National institute of standards and technology |
| **OAEP** | Optimal asymmetric encryption padding |
| **OFB** | Output feedback based block cipher mode |
| **GCM** | Galois/counter mode |
| **GMAC** | Galois message authentication code |
| **HMAC** | Keyed-hash message authentication code |
| **PKC** | Public key cryptographic engine |
| **PKCS1** | Public-key cryptography standards. PKCS provides the basic definitions of, and recommendations for implementing the RSA algorithm. |
| **POR** | Power-on reset |
| **RNG** | Random number generator |
| **RSA** | Rivest–Shamir–Adleman (a public key cryptosystem) |
| **SHE** | Secure hardware extension |
| **TRNG** | True random number generator |
| **XCBC-MAC** | Extended ciphertext block chaining MAC |
| **XTS** | XEX (XOR encrypt XOR) based tweaked-codebook mode with ciphertext stealing |

# Chapter 36
# Device Configuration Format (DCF) records

## 36.1 Introduction

DCF configures certain registers of this chip during system boot while the reset signal asserts. An individual DCF record points to an internal register in the chip and the data to be written to that register.

UTEST DCF—The UTEST DCF clients are present within the UTEST region of the flash and programmed during production testing. You may write the other records and program them at the same time the application code is programmed in the flash memory. See the DCF client file attached to this document for the description on the UTEST DCF records.

System boot is a complex process that requires you to initialize the chip properly before releasing reset. Before using the chip, you must write the user application code and the DCF records into their respective flash memory.

After power is supplied to an appropriately configured chip, PMC controls the chip, and after the system power supplies reach predefined levels, PMC signals MC_RCM to start the boot sequence. During this sequence, MC_RGM enables DCM to read the chip configuration records and then write the configuration information to the specified registers.

## 36.2 DCF clients

These are 32-bit wide hardware registers inside a module that receive and store data from a DCF record. This stored data is used to initialize registers and configure features.

DCF clients:

- Are assigned a default value before any DCF records are written.
- May have special writing constraints, such as:
  - Write once.
  - Change from 1 to 0 only.
  - Change from 0 to 1 only.
- May not implement all 32 bits.

### 36.2.1 Safety features of DCF clients

Depending on the DCF client's role in the chip, the client may be equipped with a safety feature or a combination of these features.

#### 36.2.1.1 Parity

If a DCF client implements parity checking, the client receives a parity bit in addition to its data in the DCF record. During chip operation, the client continuously monitors whether the data it stores matches the parity. It also reports errors to DCM in case of discrepancies.

#### 36.2.1.2 Triple voting

DCF clients that use triple voting have three copies of the register. DCM writes to all the three registers in a single write cycle. During chip operation, the DCF client continuously monitors whether all these three copies match. In case of discrepancies, the client reports errors to DCM. The chip uses the majority result, so single errors do not affect the chip's operation.

#### 36.2.1.3 Spreaded triple voting

DCF clients that use spreaded triple voting have three copies of the register that are individually addressed and where the data is stored in three different formats. DCM fetches these records independently. During chip operation, the DCF client continuously monitors whether all the three copies can be resolved from the stored format to the value. In case of discrepancies, the client reports errors to DCM. The chip uses the majority result, so single errors do not affect the chip's operation.

#### 36.2.1.4 Second write check

This feature is used by DCF clients that receive their records twice—for example, once before and once after BIST for clients that retain their value during self-test. The DCF client monitors whether the second write matches the first and reports deviations to DCM.

### 36.2.2 DCF client modification rules

Depending on its role in the chip, a DCF client may implement one or a combination of modification rules. If a modification rule is in effect, the order in which DCF records are placed in the record list may be important.

#### 36.2.2.1 Write once

A DCF client using the write once rule can only be written with a single DCF record. The records that are appended later in the list are ignored and do not change the value of the client.

#### 36.2.2.2 Write 0 only

A field in a DCF client can only be changed from 1 to 0. Therefore, if the value of a field in the previous DCF record is 0, an attempt by a later record to write 1 to it is ignored.

#### 36.2.2.3 Write 1 only

A field in a DCF client can only be changed from 0 to 1. Therefore, if the value of a field in the previous DCF record is 1, an attempt by a later record to write 0 to the field is ignored.

## 36.3 DCF record structure

A DCF record is a double-word (64-bit) entry that consists of the following:

- Control word—This provides information to locate the corresponding DCF client internal to the chip (pointer to the location of a register internal to the chip).
- Data word—This contains the data to be written to the DCF client.

DCF records select the target DCF client using a 30-bit field in the DCF record that consists of a 15-bit chip select field and a 15-bit address field. All modules that include DCF clients are assigned a chip select during chip definition. The address field is only relevant for address decoding within that module and may not necessarily relate to the address of a register that is visible to you.

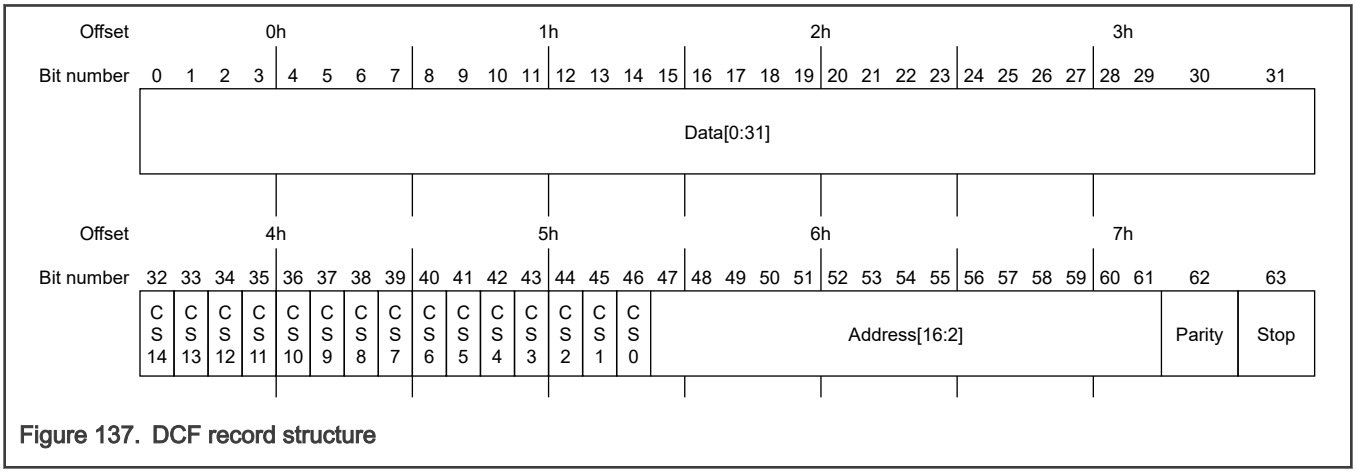



Figure 137. DCF record structure

Table 189. DCF record field descriptions

| Field | Name | Description |
|---|---|---|
| 0-31 | Data[0:31] | Provides the data that is to be written to the DCF client. |
| 32-46 | CS*n* | Indicates chip select *n*.<br>The value 1 is written to a chip select per DCF record to select the target module for the DCF client. The value 0 must be written to all other chip selects. |
| 47-61 | Address[16:2] | Contains the address of the DCF client within the selected module.<br>Address decoding for DCF clients may not match the standard software address map decoding. For details, see the DCF client addresses provided with each module. |
| 62 | Parity | Indicates parity for the DCF record. |
| 63 | Stop | Indicates the end of the list for DCF records.<br>0 – Not the end<br>1 – End<br>The erased state of flash memory is FFFF_FFFF_FFFF_FFFFh. Therefore, the list ends with the first unprogrammed double word.<br>This location can be programmed with a new record to extend the list. |

## 36.4 DCF records sequence

An individual DCF record contains information to locate the corresponding DCF client internal to the chip (control word) and the data to be written to that client (data word).

DCF records appear as contiguous series of entries programmed at a specific address within flash memory, and must present the following pattern:

- The first DCF record must be a start record. This record must be placed at the beginning of a DCF area in flash memory to indicate to the chip that the specified records must be processed.
- DCF records containing configuration data must immediately follow the start record with no blank records in between. An unprogrammed record is interpreted as a stop record and no DCF records following that are processed. This allows you to program the records in several sessions, appending new records at the end of the list each time.
- The presence of a stop record indicates the end of configuration records. It is not recommended to set valid UTEST DCF record programmed during production because that prevents appending additional UTEST records. The flash memory location following the last UTEST DCF record programmed at the factory is an unprogrammed location, which has FFFF_FFFFh as its content. Thus, the stop bit location in this unprogrammed flash memory location is logic 1, signifying that this is the last DCF record and it is not to be acted upon.

Table 190 shows the record that DCM recognizes as a start record.

Table 190. DCF start record

| 0h (0:31) | 4h (32:63) |
|---|---|
| 05AA_55AFh | 0000_0000h |

The factory sets the start bit at the beginning of the UTEST flash memory area.

Table 191 shows the record that DCM recognizes as a stop record.

**Table 191. DCF stop record**

| 0:31 | 32:62 | 63 |
|---|---|---|
| Ignored | Ignored | 1 |

The DCF records that you supply may be added in a contiguous manner immediately following the factory-written DCF records. You must never have an unprogrammed record in the series of DCF records because that is interpreted as a stop record.

Table 192 shows the series of DCF records when *n* data records are stored in the UTEST flash memory.

**Table 192. Series of DCF records in UTEST flash memory**

| Record type | Address offset | Data | | | |
|---|---|---|---|---|---|
| Start | 0h | 05AA_55AFh | | | |
| | 4h | 0000_0000h | | | STOP = 0 |
| Data | 8h | WDATA[31:0] | | | |
| | Ch | CS[14:0] | ADDR[16:2] | Parity | STOP = 0 |
| | 10h | WDATA[31:0] | | | |
| | 14h | CS[14:0] | ADDR[16:2] | Parity | STOP = 0 |
| | ... | ... | | | |
| Stop | 8(*n*-1) + 0h | Reserved | | | |
| | 8(*n*-1) + 4h | Reserved | | | 1 |
| | 8*n* + 0h | | | | |
| | 8*n* + 4h | | | | |

More than one DCF records can write to the same DCF client. In this case, the later record usually overrides a DCF client value defined by a previous record. However, not all DCF clients allow overwrites; this depends on individual implementation of DCF clients.

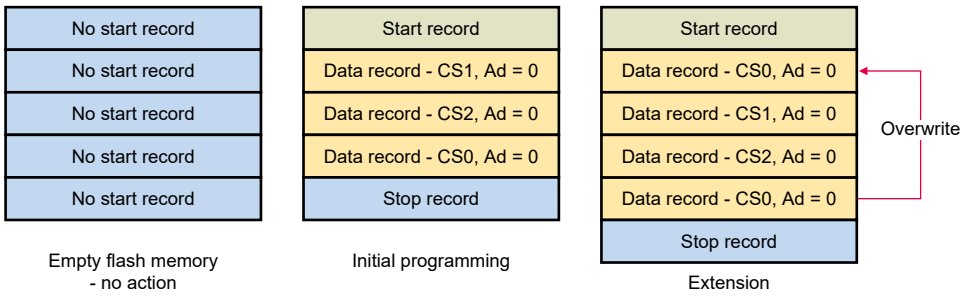



**Figure 138. Appending DCF records**

## 36.5 Chip configuration records

The DCF clients table contains information on DCF clients available in the chip. See the DCF clients file attached to this document.

The next table shows an example of how the information in this chapter is integrated with the attached DCF file.

**Table 193. Integration of DCF information**

| Type | Data(n) assuming Quad page program | Data | Comment |
|---|---|---|---|
| UTEST DCF start record | Data0 | 05AA_55AFh | — |
| | Data1 | 0000_0000h | — |
| Reset pad dedicated control DCF client | Data0 | 0000_0001h | Data to enable pad as dedicated reset pad |
| | Data1 | 00_10_0008h | Chip Select is 3 and address is 8, 0010_0000h+ 8h |

## 36.6 Glossary

**BIST** Built-in self-test

**OTP** One time programmable

**UTEST** User test. Refers to UTEST region of the flash.

# Chapter 37
# Device Configuration Module General-Purpose Registers (DCM_GPR)

## 37.1 DCM controlled features and availability in product family

Based on the chip features described in ‘Feature comparison’ section in ‘Introduction’ chapter, there are some features which are present only in specific parts in the MCXE31xx product family. The following table summarizes the corresponding DCM register fields along with the parts wherein the corresponding register fields are available. In rest of the parts within the product family, the corresponding fields are reserved.

**Table 194. DCM controlled features and availability in product family**

| Register field | Register field abbreviation | Register field description | Parts wherein this field is available |
|---|---|---|---|
| DCMROD3[5] | TCM_GSKT_ALARM | See Register section for details | MCXE31B, MCXE317, MCXE315/MCXE316 |
| DCMROD3[6] | DMA_SYS_GSKT_ALARM | See Register section for details | MCXE31B |
| DCMROD3[7] | DMA_PERIPH_GSKT_ALARM | See Register section for details | MCXE31B |
| DCMROD3[9] | DMA_AXBS_ALARM | See Register section for details | MCXE31B |
| DCMROD3[12] | QSPI_GSKT_ALARM | See Register section for details | MCXE31B |
| DCMROD3[14] | AIPS2_GSKT_ALARM | See Register section for details | MCXE31B |
| DCMROD3[17] | TCM_AXBS_ALARM | See Register section for details | MCXE31B |
| DCMROD3[18] | MAC_GSKT_ALARM | See Register section for details | MCXE31B |
| DCMROD3[19] | PERIPH_AXBS_ALARM | See Register section for details | MCXE31B |
| DCMROD3[24] | PRAM1_ECC_ERR | See Register section for details | MCXE31B |
| DCMROD4[10] | PRAM1_FCCU_ALARM | See Register section for details | MCXE31B |
| DCMROD5[14] | TCM_RDATA_EDC_ERR | Specifies whether an integrity error is reported on the TCM read data for safety.<br>Read this field to identify the reason for a fault in case of FCCU NCF 1. | MCXE31B |
| DCMROD5[15] | MAC_RDATA_EDC_ERR | See Register section for details | MCXE31B |
| DCMROF1[0] | MAC_MDC_CHID_0 | See Register section for details | MCXE31B |
| DCMROF1[1] | MAC_MDC_CHID_1 | See Register section for details | MCXE31B |
| DCMROF20[1] | LMAUTO_DIS | Specifies whether the PMC last-mile automatic crossover from the boot regulation feature is supported for the chip. | MCXE31B |
| DCMROF20[3] | DMA_AXBS_IAHB_BYP | See Register section for details | MCXE31B |

*Table continues on the next page...*

Table 194. DCM controlled features and availability in product family (continued)

| Register field | Register field abbreviation | Register field description | Parts wherein this field is available |
|---|---|---|---|
| DCMROF20[5] | QSPI_IAHB_BYP | See Register section for details | MCXE31B |
| DCMRWD3[5] | TCM_GSKT_ALARM_EN | See Register section for details | MCXE31B, MCXE317, MCXE315/MCXE316 |
| DCMRWD3[6] | DMA_SYS_GSKT_ALARM_EN | See Register section for details | MCXE31B |
| DCMRWD3[7] | DMA_PERIPH_GSKT_ALARM_EN | See Register section for details | MCXE31B |
| DCMRWD3[9] | DMA_AXBS_ALARM_EN | See Register section for details | MCXE31B |
| DCMRWD3[12] | QSPI_GSKT_ALARM_EN | See Register section for details | MCXE31B |
| DCMRWD3[14] | AIPS2_GSKT_ALARM_EN | See Register section for details | MCXE31B |
| DCMRWD3[17] | TCM_AXBS_ALARM_EN | See Register section for details | MCXE31B |
| DCMRWD3[18] | MAC_GSKT_ALARM_EN | See Register section for details | MCXE31B |
| DCMRWD3[19] | PERIPH_AXBS_ALARM_EN | See Register section for details | MCXE31B |
| DCMRWD3[24] | PRAM1_ECC_ERR_EN | See Register section for details | MCXE31B |
| DCMRWD4[10] | PRAM1_FCCU_ALARM_EN | See Register section for details | MCXE31B |
| DCMRWD5[14] | TCM_RDATA_EDC_ERR_EN | Specifies whether an integrity error is reported on the TCM read data.<br>The field enables fault monitoring at FCCU NCF 1, if there is an integrity error on the TCM read data, for safety. | MCXE31B |
| DCMRWD5[15] | MAC_RDATA_EDC_ERR_EN | See Register section for details | MCXE31B |
| DCMRWD6[6] | eMIOS2_DBG_DIS_CM7_0 | See Register section for details | MCXE31B |
| DCMRWD6[11] | STM1_DBG_DIS_CM7_0 | See Register section for details | MCXE31B |
| DCMRWD6[14] | PIT2_DBG_DIS_CM7_0 | See Register section for details | MCXE31B |
| DCMRWD6[19] | LPSPI4_DBG_DIS_CM7_0 | See Register section for details | MCXE31B |
| DCMRWD6[20] | LPSPI5_DBG_DIS_CM7_0 | See Register section for details | MCXE31B |
| DCMRWD6[27] | FLEXCAN3_DBG_DIS_CM7_0 | See Register section for details | MCXE31B, MCXE317 |
| DCMRWD6[28] | FLEXCAN4_DBG_DIS_CM7_0 | See Register section for details | MCXE31B, MCXE317 |
| DCMRWD6[29] | FLEXCAN5_DBG_DIS_CM7_0 | See Register section for details | MCXE31B, MCXE317 |
| DCMRWD6[30] | SAI0_DBG_DIS_CM7_0 | See Register section for details | MCXE31B |
| DCMRWD6[31] | SAI1_DBG_DIS_CM7_0 | See Register section for details | MCXE31B |
| DCMRWF1[0] | CAN_TIMESTAMP_SEL | See Register section for details | MCXE31B |
| DCMRWF1[1] | CAN_TIMESTAMP_EN | See Register section for details | MCXE31B |

*Table continues on the next page...*

Table 194. DCM controlled features and availability in product family (continued)

| Register field | Register field abbreviation | Register field description | Parts wherein this field is available |
|---|---|---|---|
| DCMRWF1[7] | MAC_CONF_SEL | See Register section for details | MCXE31B |
| DCMRWF1[15] | VDD_HV_B_IO_CTRL_LATCH | See Register section for details | MCXE31B |
| DCMRWF1[26] | VDD_HV_B_VLT_DVDR_EN | See Register section for details | MCXE31B |
| DCMRWF1[27] | VDD_1_5_VLT_DVDR_EN | See Register section for details | MCXE31B |
| DCMRWF4[0] | MUX_MODE_EN_ADC1_S18 | Controls the selection of GPIOs to drive ADC1 standard channel 18th. | MCXE315/MCXE316 |
| DCMRWF4[5] | MUX_MODE_EN_ADC1_S22 | See Register section for details | MCXE31B, MCXE317 |
| DCMRWF4[6] | MUX_MODE_EN_ADC1_S23 | See Register section for details | MCXE31B, MCXE317 |
| DCMRWF4[7] | MUX_MODE_EN_ADC0_S12 | Controls the selection of GPIOs to drive ADC_0 standard channel 12. | MCXE315/MCXE316 |
| DCMRWF4[8] | MUX_MODE_EN_ADC0_S13 | Controls the selection of GPIOs to drive ADC_0 standard channel 13. | MCXE315/MCXE316 |
| DCMRWF4[9] | MUX_MODE_EN_ADC2_S8 | See Register section for details | MCXE31B |
| DCMRWF4[10] | MUX_MODE_EN_ADC2_S9 | See Register section for details | MCXE31B |
| DCMRWF4[11] | MUX_MODE_EN_ADC0_S14 | Controls the selection of GPIOs to drive ADC_0 standard channel 14. | MCXE315/MCXE316 |
| DCMRWF4[12] | MUX_MODE_EN_ADC0_S17 | Controls the selection of GPIOs to drive ADC_0 standard channel 17. | MCXE315/MCXE316 |
| DCMRWF4[28] | MUX_MODE_EN_ADC0_P2 | Controls the selection of GPIOs to drive ADC0 precision channel 2nd. | MCXE315/MCXE316 |

## 37.2 DCM_GPR register descriptions

**NOTE**

Do not write to the reserved locations, registers, or reserved bits in the registers. Chip behavior is not guaranteed in case of such writes.

**NOTE**

The MC_RGM_ERCTRL register configuration takes several cycles to be effective. Any further access to DCM should be made after at least 9 AIPS_SLOW_CLK cycles after writing to MC_RGM_ERCTRL

**NOTE**

The DCM Read Write registers, DCMRW*X* are synchronized. Hence changes to these configurations might take upto 4 CORE_CLK cycles to be effective.

**NOTE**

The DCM read only registers are present in standby domain and retain values across standby mode. Hence reading the status registers after standby exit might indicate a previously latched value. To read back these status registers after any reset event or standby exit, first clear these and then read. This would clear the previous status or transient information due to reset event and indicate the correct status.

### 37.2.1 DCM_GPR memory map

DCM_GPR base address: 402A_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 200h | Read Only GPR On Destructive Reset Register (DCMROD1) | 32 | RW | 0000_0000h |
| 208h | Read Only GPR On Destructive Reset Register (DCMROD3) | 32 | RW | 0000_0000h |
| 20Ch | Read Only GPR On Destructive Reset Register (DCMROD4) | 32 | RW | 0000_0000h |
| 210h | Read Only GPR On Destructive Reset Register (DCMROD5) | 32 | RW | 0000_0000h |
| 300h | Read Only GPR On Functional Reset Register (DCMROF1) | 32 | RW | 0000_0000h |
| 304h | Read Only GPR On Functional Reset Register (DCMROF2) | 32 | RW | 0000_0000h |
| 308h | Read Only GPR On Functional Reset Register (DCMROF3) | 32 | RW | 0000_0000h |
| 30Ch | Read Only GPR On Functional Reset Register (DCMROF4) | 32 | RW | 0000_0000h |
| 310h | Read Only GPR On Functional Reset Register (DCMROF5) | 32 | RW | 0000_0000h |
| 314h | Read Only GPR On Functional Reset Register (DCMROF6) | 32 | RW | 0000_0000h |
| 318h | Read Only GPR On Functional Reset Register (DCMROF7) | 32 | RW | 0000_0000h |
| 31Ch | Read Only GPR On Functional Reset Register (DCMROF8) | 32 | RW | 0000_0000h |
| 320h | Read Only GPR On Functional Reset Register (DCMROF9) | 32 | RW | 0000_0000h |
| 324h | Read Only GPR On Functional Reset Register (DCMROF10) | 32 | RW | 0000_0000h |
| 328h | Read Only GPR On Functional Reset Register (DCMROF11) | 32 | RW | 0000_0000h |
| 32Ch | Read Only GPR On Functional Reset Register (DCMROF12) | 32 | RW | 0000_0000h |
| 330h | Read Only GPR On Functional Reset Register (DCMROF13) | 32 | RW | 0000_0000h |
| 334h | Read Only GPR On Functional Reset Register (DCMROF14) | 32 | RW | 0000_0000h |
| 338h | Read Only GPR On Functional Reset Register (DCMROF15) | 32 | RW | 0000_0000h |
| 33Ch | Read Only GPR On Functional Reset Register (DCMROF16) | 32 | RW | 0000_0000h |
| 340h | Read Only GPR On Functional Reset Register (DCMROF17) | 32 | RW | 0000_0000h |
| 348h | Read Only GPR On Functional Reset Register (DCMROF19) | 32 | R | 4000_0000h |
| 34Ch | Read Only GPR On Functional Reset Register (DCMROF20) | 32 | R | See section |
| 350h | Read Only GPR On Functional Reset Register (DCMROF21) | 32 | R | 0000_0000h |
| 400h | Read Write GPR On Power On Reset Register (DCMRWP1) | 32 | RW | 0000_0400h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 408h | Read Write GPR On Power On Reset Register (DCMRWP3) | 32 | RW | 0000_0000h |
| 504h | Read Write GPR On Destructive Reset Register (DCMRWD2) | 32 | RW | 0000_0000h |
| 508h | Read Write GPR On Destructive Reset Register (DCMRWD3) | 32 | RW | FF4F_FBFFh |
| 50Ch | Read Write GPR On Destructive Reset Register (DCMRWD4) | 32 | RW | EF7F_FFFFh |
| 510h | Read Write GPR On Destructive Reset Register (DCMRWD5) | 32 | RW | 007F_FFFFh |
| 514h | Read Write GPR On Destructive Reset Register (DCMRWD6) | 32 | RW | 0000_0000h |
| 518h | Read Write GPR On Destructive Reset Register (DCMRWD7) | 32 | R | 0000_0000h |
| 51Ch | Read Write GPR On Destructive Reset Register (DCMRWD8) | 32 | R | 0000_0000h |
| 520h | Read Write GPR On Destructive Reset Register (DCMRWD9) | 32 | R | 0000_0000h |
| 600h | Read Write GPR On Functional Reset Register (DCMRWF1) | 32 | RW | 0000_0000h |
| 604h | Read Write GPR On Functional Reset Register (DCMRWF2) | 32 | RW | 0000_0000h |
| 60Ch | Read Write GPR On Functional Reset Register (DCMRWF4) | 32 | RW | 0000_0000h |
| 610h | Read Write GPR On Functional Reset Register (DCMRWF5) | 32 | RW | See section |
| 700h | Read Only GPR On PMCPOR Reset (DCMROPP1) | 32 | RW | 0000_0000h |
| 704h | Read Only GPR On PMCPOR Reset (DCMROPP2) | 32 | RW | 0000_0000h |
| 708h | Read Only GPR On PMCPOR Reset (DCMROPP3) | 32 | RW | 0000_0000h |
| 70Ch | Read Only GPR On PMCPOR Reset (DCMROPP4) | 32 | RW | 0000_0000h |

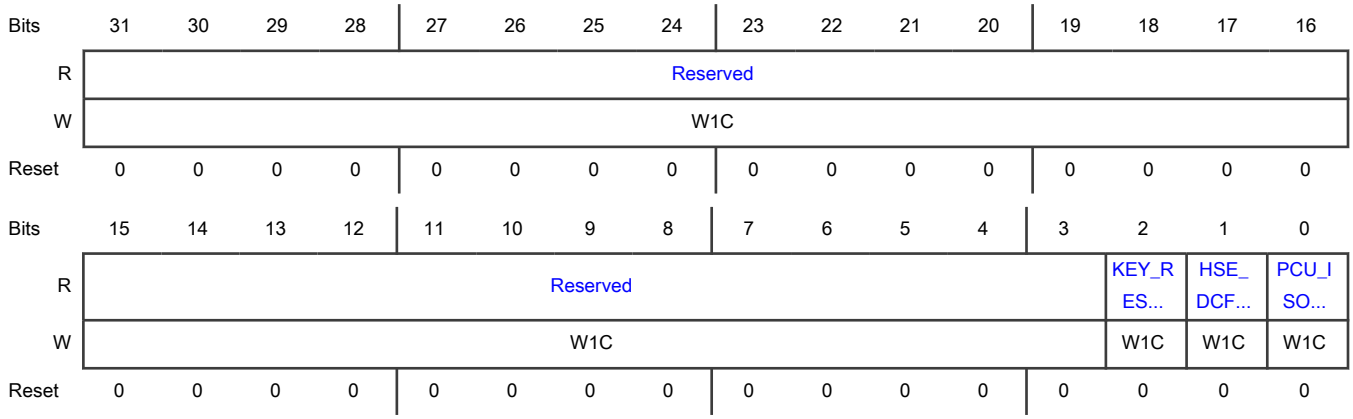
### 37.2.2 Read Only GPR On Destructive Reset Register (DCMROD1)

**Offset**

| Register | Offset |
|---|---|
| DCMROD1 | 200h |

**Function**
Resets on destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2<br>KEY_RESP_READY | Key Response Ready<br>SDA-AP key response status. Indicates that the for the SDA-AP all key values have been written and HSE can begin authentication.<br>0b - Key response is not ready.<br>1b - Key response is ready. |
| 1<br>HSE_DCF_VIO | DCF violation from HSE<br>0b - No violation from HSE DCF records while decoding.<br>1b - Violation from HSE DCF records while decoding. |
| 0<br>PCU_ISO_STATUS | PCU Input Isolation status on previous standby entry<br>0b - Input isolation did not get enabled in the previous standby entry.<br>1b - Input isolation enabled in the previous standby entry. |

## 37.2.3 Read Only GPR On Destructive Reset Register (DCMROD3)

### Offset

| Register | Offset |
|---|---|
| DCMROD3 | 208h |

### Function

Resets on destructive reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | CM7_0_I... | Reserved | CM7_0_D... | Reserved | CM7_0_D... | PRAM0_E... | PRAM1_E... | Reserved | LC_ERR | Reserved | | PERIPH_... | EMAC_GS... | TCM_AXB... | DATA_ED... |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ADDR_ED... | AIPS2_G... | AIPS1_G... | QSPI_GS... | HSE_GSK... | Reserved | DMA_AXB... | SYS_AXB... | DMA_PER... | DMA_SYS... | TCM_GSK... | Reserved | Reserved | HSE_LOC... | Reserved | CM7_0_L... |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>CM7_0_ICDATA_ECC_ERR | Multi bit ECC error from CM7_0 ICache data memory. Read this bit to identify the reason of fault in case of FCCU NCF 2.<br>0b - CM7_1 core not in lockup state.<br>1b - CM7_1 core in lockup state. |
| 29<br>— | Reserved |
| 28<br>CM7_0_DCTAG_ECC_ERR | Multi bit ECC error from CM7_0 DCache tag memory. Read this bit to identify the reason of fault in case of FCCU NCF 2.<br>0b - No multi-bit ECC error.<br>1b - Multi-bit ECC error. |
| 27<br>— | Reserved |
| 26<br>CM7_0_DCDATA_ECC_ERR | Multi bit ECC error from CM7_0 DCache data memory. Read this bit to identify the reason of fault in case of FCCU NCF 2.<br>0b - No multi-bit ECC error.<br>1b - Multi-bit ECC error. |
| 25<br>PRAM0_ECC_ERR | Multi bit ECC error from SRAM0. Read this bit to identify the reason of fault in case of FCCU NCF 2.<br>0b - No multi-bit ECC error.<br>1b - Multi-bit ECC error. |
| 24 | Multi bit ECC error from SRAM1. Read this bit to identify the reason of fault in case of FCCU NCF 2. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| PRAM1_ECC_ERR | 0b - No multi-bit ECC error.<br>1b - Multi-bit ECC error. |
| 23<br>— | Reserved |
| 22<br>LC_ERR | Error in Lifecycle Scanning. Read this bit to identify the reason of fault in case of FCCU NCF 3.<br>0b - No error while lifecycle scanning.<br>1b - Error while lifecycle scanning. |
| 21-20<br>— | Reserved |
| 19<br>PERIPH_AXBS_ALARM | PERIPH AXBS_Lite Safety Alarm Status. Read this bit to identify the reason of fault in case of FCCU NCF 1.<br>0b - No safety alarm indicated by Periph AXBS_Lite.<br>1b - Safety alarm indicated by Periph AXBS_Lite. |
| 18<br>EMAC_GSKT_ALARM | EMAC IAHB Gasket Alarm Status. Read this bit to identify the reason of fault in case of FCCU NCF 1.<br>0b - No alarm indicated by EMAC IAHB gasket.<br>1b - Alarm indicated by EMAC IAHB gasket. |
| 17<br>TCM_AXBS_ALARM | TCM AHB Splitter Safety Alarm Status. Read this bit to identify the reason of fault in case of FCCU NCF 1.<br>0b - No safety alarm indicated by TCM AHB splitter.<br>1b - Safety alarm indicated by TCM AHB splitter. |
| 16<br>DATA_EDC_ERR | Status of integrity error on data for safety. Read this bit to identify the reason of fault in case of FCCU NCF 1.<br>0b - No integrity error on data.<br>1b - Data integrity error. |
| 15<br>ADDR_EDC_ERR | Status of integrity error on address for safety. Read this bit to identify the reason of fault in case of FCCU NCF 1.<br>0b - No integrity error on address.<br>1b - Address integrity error. |
| 14<br>AIPS2_GSKT_ALARM | AIPS2 IAHB Gasket Alarm Status. Read this bit to identify the reason of fault in case of FCCU NCF 1.<br>0b - No alarm indicated by AIPS2 IAHB gasket.<br>1b - Alarm indicated by AIPS2 IAHB gasket. |
| 13 | AIPS1 IAHB Gasket Alarm Status. Read this bit to identify the reason of fault in case of FCCU NCF 1. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| AIPS1_GSKT_ALARM | 0b - No alarm indicated by AIPS1 IAHB gasket.<br>1b - Alarm indicated by AIPS1 IAHB gasket. |
| 12<br>QSPI_GSKT_ALARM | QSPI IAHB Gasket Alarm Status. Read this bit to identify the reason of fault in case of FCCU NCF 1.<br>0b - No alarm indicated by QSPI IAHB gasket.<br>1b - Alarm indicated by QSPI IAHB gasket. |
| 11<br>HSE_GSKT_ALARM | HSE IAHB Gasket Alarm Status. Read this bit to identify the reason of fault in case of FCCU NCF 1.<br>0b - No alarm indicated by HSE IAHB gasket.<br>1b - Alarm indicated by HSE IAHB gasket. |
| 10<br>— | Reserved |
| 9<br>DMA_AXBS_ALARM | DMA AXBS_Lite Safety Alarm Status. Read this bit to identify the reason of fault in case of FCCU NCF 1.<br>0b - No safety alarm indicated by DMA AXBS_Lite.<br>1b - Safety alarm indicated by DMA AXBS_Lite. |
| 8<br>SYS_AXBS_ALARM | System AXBS Safety Alarm Status. Read this bit to identify the reason of fault in case of FCCU NCF 1.<br>0b - No safety alarm indicated by system AXBS.<br>1b - Safety alarm indicated by system AXBS. |
| 7<br>DMA_PERIPH_GSKT_ALARM | Status of IAHB gasket safety alarm from DMA periph AXBS IAHB gasket. Read this bit to identify the reason of fault in case of FCCU NCF 1.<br>0b - No safety alarm reported from DMA-periph AXBS IAHB gasket.<br>1b - Safety alarm indicated by DMA-periph AXBS IAHB gasket. |
| 6<br>DMA_SYS_GSKT_ALARM | Status of IAHB gasket safety alarm from DMA system AXBS IHAB gasket. Read this bit to identify the reason of fault in case of FCCU NCF 1.<br>0b - No safety alarm reported from DMA-system AXBS IAHB gasket.<br>1b - Safety alarm indicated by DMA-system AXBS IAHB gasket. |
| 5<br>TCM_GSKT_ALARM | TCM IAHB Gasket Monitor Alarm Status. Read this bit to identify the reason of fault in case of FCCU NCF 1.<br>0b - No alarm reported from TCM IAHB gasket.<br>1b - Monitor alarm reported from TCM IAHB gasket. |
| 4<br>— | Reserved |
| 3<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 2<br>HSE_LOCKUP | HSE Core Lockup Status. Read this bit to identify the reason of fault in case of FCCU NCF 0.<br>0b - HSE core not in lockup state.<br>1b - HSE core in lockup state. |
| 1<br>— | Reserved |
| 0<br>CM7_0_LOCKUP | CM7_0 Core Lockup Status. Read this bit to identify the reason of fault in case of FCCU NCF 0.<br>0b - CM7_0 core not in lockup state.<br>1b - CM7_0 core in lockup state. |

### 37.2.4 Read Only GPR On Destructive Reset Register (DCMROD4)

#### Offset

| Register | Offset |
|---|---|
| DCMROD4 | 20Ch |

#### Function
Resets on destructive reset.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | TEST_AC... | TEST_AC... | Reserved | VDD2P5_... | VDD1P1_... | Reserved | FLASH_A... | Reserved | FLASH_S... | FLASH_R... | FLASH_R... | FLASH_A... | FLASH_E... | Reserved | Reserved |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PF1_DAT... | PF1_COD... | PF0_DAT... | PF0_COD... | HSE_RAM... | PRAM1_F... | PRAM0_F... | DMA_TCD... | Reserved | Reserved | Reserved | CM7_0_D... | CM7_0_D... | CM7_0_I... | Reserved | CM7_0_I... |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 30<br>TEST_ACTIVATION_1_ERR | Accidental Partial Test Activation. Read this bit to identify the reason of fault in case of FCCU NCF 5.<br>0b - No partial test activated accidentally.<br>1b - Partial test activated accidentally. |
| 29<br>TEST_ACTIVATION_0_ERR | Accidental Partial Test Activation. Read this bit to identify the reason of fault in case of FCCU NCF 5.<br>0b - No partial test activated accidentally.<br>1b - Partial test activated accidentally. |
| 28<br>— | Reserved |
| 27<br>VDD2P5_GNG_ERR | Go/no-go for VDD_HV_FLA (double bond) going to FXOSC and PLL. Read this bit to identify the reason of fault in case of FCCU NCF 4.<br>0b - Go indication referring to the supply being clean.<br>1b - No-go indication referring to the supply being unclean and a fault in double bond connection or its routing within the chip. |
| 26<br>VDD1P1_GNG_ERR | Go/no-go indicator for VDD1PD1 (double bond) supply going to PLL. Read this bit to identify the reason of fault in case of FCCU NCF 4.<br>0b - Go indication referring to the supply being clean.<br>1b - No-go indication referring to the supply being unclean and a fault in double bond connection or its routing within the chip. |
| 25<br>— | Reserved |
| 24<br>FLASH_ACCESS_ERR | Transaction Monitor Mismatch Error From Flash Controller<br>This alarm idicates that the flash controller detected a transaction monitor mismatch when compared to flash safety feedback output. There is an indication in the flash where reconstructed address is compared with the address which invoked the flash access. Read this bit to identify the reason of fault in case of FCCU NCF 3.<br>0b - No transaction monitor mismatch error from flash controller.<br>1b - Transaction monitor mismatch error from flash controller. |
| 23<br>— | Reserved |
| 22<br>FLASH_SCAN_ERR | Error while DCM flash scanning process due to invalid data. Read this bit to identify the reason of fault in case of FCCU NCF 3.<br>0b - No error while DCM flash scanning.<br>1b - Error while DCM flash scanning. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 21<br>FLASH_RST_ERR | Flash Reset Error Status<br>This error indication is set when flash encounters errors during its reset reads. Read this bit to identify the reason of fault in case of FCCU NCF 3.<br>0b - No flash reset error indicated.<br>1b - Flash reset error indicated. |
| 20<br>FLASH_REF_ERR | Flash reference current loss or read voltage error while prevoius read(s). Read this bit to identify the reason of fault in case of FCCU NCF 3.<br>0b - No reference current loss or read voltage error while previous reads(s).<br>1b - Reference current loss or read voltage error while previous read(s). |
| 19<br>FLASH_ADDR_ENC_ERR | Flash Address Encode Error<br>In address decoding, if multiple or no address line is selected, FMU reports address encode error. Read this bit to identify the reason of fault in case of FCCU NCF 3.<br>0b - No address encode error in flash.<br>1b - Address enocde error in flash. |
| 18<br>FLASH_EDC_ERR | Status of flash ECC correction error through EDC reported by FMU. Read this bit to identify the reason of fault in case of FCCU NCF 3.<br>0b - No EDC after ECC error reported in flash.<br>1b - EDC after ECC error reported in flash. |
| 17<br>— | Reserved |
| 16<br>— | Reserved |
| 15<br>PF1_DATA_ECC_ERR | Flash1 Data ECC Uncorrectable Error<br>The errors are reported from the FMU and are connected to FCCU NCFs. These are also connected to ERM. See ERM chapter for the memory errors and mapping onto ERM channels. Read this bit to identify the reason of fault in case of FCCU NCF 3.<br>0b - No uncorrectable error reported in flash controller port 1 data memory by FMU.<br>1b - Uncorrectable error reported in flash controller port 1 data memory by FMU. |
| 14<br>PF1_CODE_ECC_ERR | Flash1 Code ECC Uncorrectable Error<br>The errors are reported from the FMU and are connected to FCCU NCFs. These are also connected to ERM. See ERM chapter for the memory errors and mapping onto ERM channels. Read this bit to identify the reason of fault in case of FCCU NCF 3.<br>0b - No uncorrectable error reported in flash controller port 1 code memory by FMU.<br>1b - Uncorrectable error reported in flash controller port 1 code memory by FMU. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 13<br>PF0_DATA_ECC_ERR | Flash0 Data ECC Uncorrectable Error<br>The errors are reported from the FMU and are connected to FCCU NCFs. These are also connected to ERM. See ERM chapter for the memory errors and mapping onto ERM channels. Read this bit to identify the reason of fault in case of FCCU NCF 3.<br>0b - No uncorrectable error reported in flash controller port 0 data memory by FMU.<br>1b - Uncorrectable error reported in flash controller port 0 data memory by FMU. |
| 12<br>PF0_CODE_ECC_ERR | Flash0 Code ECC Uncorrectable Error<br>The errors are reported from the FMU and are connected to FCCU NCFs. These are also connected to ERM. See ERM chapter for the memory errors and mapping onto ERM channels. Read this bit to identify the reason of fault in case of FCCU NCF 3.<br>0b - No uncorrectable error reported in flash controller port 0 code memory by FMU.<br>1b - Uncorrectable error reported in flash controller port 0 code memory by FMU. |
| 11<br>HSE_RAM_ECC_ERR | HSE RAM Uncorrectable ECC Status. Read this bit to identify the reason of fault in case of FCCU NCF 2.<br>0b - No uncorrectable ECC reported by HSE RAM.<br>1b - Uncorrectable ECC reported by HSE RAM. |
| 10<br>PRAM1_FCCU_ALARM | Status of PRAM1 safety alarm. This alarm is set on faulty SRAM1 read or read modify error. Read this bit to identify the reason of fault in case of FCCU NCF 2.<br>0b - No safety alarm indicated by PRAM1.<br>1b - Safety alarm indicated by PRAM1. |
| 9<br>PRAM0_FCCU_ALARM | Status of PRAM0 safety alarm. This alarm is set on faulty SRAM0 read or read modify error. Read this bit to identify the reason of fault in case of FCCU NCF 2.<br>0b - No safety alarm indicated by PRAM0.<br>1b - Safety alarm indicated by PRAM0. |
| 8<br>DMA_TCD_RAM_ECC_ERR | Uncorrectable ECC error reported from DMA_TCD memory. This uncorrectable ECC error consists of multi-bit data ECC error and address ECC error. Read this bit to identify the reason of fault in case of FCCU NCF 2.<br>0b - No uncorrectable ECC error detected.<br>1b - Uncorrectable ECC error detected. |
| 7<br>— | Reserved |
| 6<br>— | Reserved |
| 5 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 4<br>CM7_0_DTCM1_ECC_ERR | Uncorrectable ECC error reported from CM7_0 Data TCM memory block 1. This uncorrectable ECC error consists of multi-bit data ECC error and address ECC error. The CM7_0 Data TCM physically consists of two blocks. Read this bit to identify the reason of fault in case of FCCU NCF 2.<br>**NOTE**<br>Address ECC error is not supported on CM7_0 Data TCM memory in MCXE31B.<br>0b - No uncorrectable ECC error detected.<br>1b - Uncorrectable ECC error detected. |
| 3<br>CM7_0_DTCM0_ECC_ERR | Uncorrectable ECC error reported from CM7_0 Data TCM memory block 0. This uncorrectable ECC error consists of multi-bit data ECC error and address ECC error. The CM7_0 Data TCM physically consists of two blocks. Read this bit to identify the reason of fault in case of FCCU NCF 2.<br>**NOTE**<br>Address ECC error is not supported on CM7_0 Data TCM memory in MCXE31B.<br>0b - No uncorrectable ECC error detected.<br>1b - Uncorrectable ECC error detected. |
| 2<br>CM7_0_ITCM_ECC_ERR | Uncorrectable ECC error reported from CM7_0 Instruction TCM memory. This uncorrectable ECC error consists of multi-bit data ECC error and address ECC error. Read this bit to identify the reason of fault in case of FCCU NCF 2.<br>**NOTE**<br>Address ECC error is not supported on CM7_0 Instruction TCM memory in MCXE31B.<br>0b - No uncorrectable ECC error detected.<br>1b - Uncorrectable ECC error detected. |
| 1<br>— | Reserved |
| 0<br>CM7_0_ICTAG_ECC_ERR | Multi bit ECC error from CM7_0 ICache tag memory. Read this bit to identify the reason of fault in case of FCCU NCF 2.<br>0b - No multi-bit ECC error.<br>1b - Multi-bit ECC error. |

### 37.2.5 Read Only GPR On Destructive Reset Register (DCMROD5)

**Offset**

| Register | Offset |
|---|---|
| DCMROD5 | 210h |

### Function

Resets on destructive reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | HSE_RDA... | CM7_0_A... | CM7_0_A... | Reserved | Reserved | DMA_RDA... | Reserved |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EMAC_RD... | TCM_RDA... | DEBUG_A... | MTR_BUS... | STCU_BI... | MBIST_A... | STCU_NCF | SW_NCF_3 | SW_NCF_2 | SW_NCF_1 | SW_NCF_0 | INTM_3_... | INTM_2_... | INTM_1_... | INTM_0_... | Reserved |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>— | Reserved |
| 29<br>— | Reserved |
| 28<br>— | Reserved |
| 27<br>— | Reserved |
| 26<br>— | Reserved |
| 25<br>— | Reserved |
| 24<br>— | Reserved |
| 23 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 22<br>HSE_RDATA_EDC_ERR | Integrity (EDC) error on HSE read data for safety. Read this bit to identify the reason of fault in case of FCCU NCF 1.<br>0b - No integrity error reported on HSE read data.<br>1b - Integrity error reported on HSE read data. |
| 21<br>CM7_0_AHBM_RDATA_EDC_ERR | Integrity error on CM7_0 main read data for safety. Read this bit to identify the reason of fault in case of FCCU NCF 1.<br>0b - No integrity error reported on CM7_0 main read data.<br>1b - Integrity error reported on CM7_0 main read data. |
| 20<br>CM7_0_AHBP_RDATA_EDC_ERR | Integrity error on CM7_0 peripheral read data for safety. Read this bit to identify the reason of fault in case of FCCU NCF 1.<br>0b - No integrity error reported on CM7_0 peripheral read data.<br>1b - Integrity error reported on CM7_0 peripheral read data. |
| 19<br>— | Reserved |
| 18<br>— | Reserved |
| 17<br>DMA_RDATA_EDC_ERR | Integrity(EDC) error on eDMA read data for safety. Read this bit to identify the reason of fault in case of FCCU NCF 1.<br>0b - No integrity error reported on eDMA read data.<br>1b - Integrity error reported on eDMA read data. |
| 16<br>— | Reserved |
| 15<br>EMAC_RDATA_EDC_ERR | Integrity(EDC) error on EMAC read data for safety. Read this bit to identify the reason of fault in case of FCCU NCF 1.<br>0b - No integrity error reported on EMAC read data.<br>1b - Integrity error reported on EMAC read data. |
| 14<br>TCM_RDATA_EDC_ERR | Integrity(EDC) error on TCM read data for safety. Read this bit to identify the reason of fault in case of FCCU NCF 1.<br>0b - No integrity error reported on TCM read data.<br>1b - Integrity error reported on TCM read data. |
| 13 | Monitoring of unintended debug activation. This is set when the core is in halted state with application debug not enabled or debugger request not enabled. Read this bit to identify the reason of fault in case of FCCU NCF 5. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| DEBUG_ACTIVATION_ERR | 0b - Unintended debug not activated.<br>1b - Unintended debug activated. |
| 12<br>MTR_BUS_ERR | Fault reported due to illegal access on MTR. This fault is reported via a transfer error indication to the system. Read this bit to identify the reason of fault in case of FCCU NCF 5.<br>0b - No transfer error indicated from MTR.<br>1b - Transfer error indicated from MTR. |
| 11<br>STCU_BIST_USER_CF | L/M BIST enabled accidentally. Fault condition in run mode (LBIST or MBIST control signals go to wrong condition). Read this bit to identify the reason of fault in case of FCCU NCF 5.<br>0b - No accidental L/M BIST enabled.<br>1b - Accidental L/M BIST enabled. |
| 10<br>MBIST_ACTIVATION_ERR | Indicates an accidental backdoor access on memories. This monitor needs to be disabled on FCCU when doing fault injection. Read this bit to identify the reason of fault in case of FCCU NCF 5.<br>0b - No accidental backdoor enabled on memories.<br>1b - Accidental backdoor enabled on memories. |
| 9<br>STCU_NCF | STCU non-critical fault, that is, BIST result error. Read this bit to identify the reason of fault in case of FCCU NCF 5.<br>0b - STCU non-critical fault not reported.<br>1b - STCU non-critical fault reported. |
| 8<br>SW_NCF_3 | Software NCF3 Status<br>Status of DCMRWF1[FCCU_SW_NCF3]. Read this bit to identify the reason of fault in case of FCCU NCF 7.<br>0b - Software NCF3, DCMRWF1[FCCU_SW_NCF3] not enabled by software.<br>1b - Software NCF3, DCMRWF1[FCCU_SW_NCF3] enabled by software. |
| 7<br>SW_NCF_2 | Software NCF2 Status<br>Status of DCMRWF1[FCCU_SW_NCF2]. Read this bit to identify the reason of fault in case of FCCU NCF 7.<br>0b - Software NCF2, DCMRWF1[FCCU_SW_NCF2] not enabled by software.<br>1b - Software NCF2, DCMRWF1[FCCU_SW_NCF2] enabled by software. |
| 6<br>SW_NCF_1 | Software NCF1 Status<br>Status of DCMRWF1[FCCU_SW_NCF1]. Read this bit to identify the reason of fault in case of FCCU NCF 7.<br>0b - Software NCF1, DCMRWF1[FCCU_SW_NCF1] not enabled by software.<br>1b - Software NCF1, DCMRWF1[FCCU_SW_NCF1] enabled by software. |
| 5 | Software NCF0 Status |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| SW_NCF_0 | Status of DCMRWF1[FCCU_SW_NCF0]. Read this bit to identify the reason of fault in case of FCCU NCF 7.<br>0b - Software NCF0, DCMRWF1[FCCU_SW_NCF0] not enabled by software.<br>1b - Software NCF0, DCMRWF1[FCCU_SW_NCF0] enabled by software. |
| 4<br>INTM_3_ERR | Interrupt monitor3 error reported by INTM. Also this error is reflected in INTM.INTM_STATUS3. See 'Functional Description' section in the INTM chapter for details. Read this bit to identify the reason of fault in case of FCCU NCF 6.<br>0b - No error reported by interrupt monitor 3.<br>1b - Error reported by interrupt monitor 3. |
| 3<br>INTM_2_ERR | Interrupt monitor2 error reported by INTM. Also this error is reflected in INTM.INTM_STATUS2. See 'Functional Description' section in the INTM chapter for details. Read this bit to identify the reason of fault in case of FCCU NCF 6.<br>0b - No error reported by interrupt monitor 2.<br>1b - Error reported by interrupt monitor 2. |
| 2<br>INTM_1_ERR | Interrupt monitor1 error reported by INTM. Also this error is reflected in INTM.INTM_STATUS1. See 'Functional Description' section in the INTM chapter for details. Read this bit to identify the reason of fault in case of FCCU NCF 6.<br>0b - No error reported by interrupt monitor 1.<br>1b - Error reported by interrupt monitor 1. |
| 1<br>INTM_0_ERR | Interrupt monitor0 error reported by INTM. Also this error is reflected in INTM.INTM_STATUS0. See 'Functional Description' section in the INTM chapter for details. Read this bit to identify the reason of fault in case of FCCU NCF 6.<br>0b - No error reported by interrupt monitor 0.<br>1b - Error reported by interrupt monitor 0. |
| 0<br>— | Reserved |

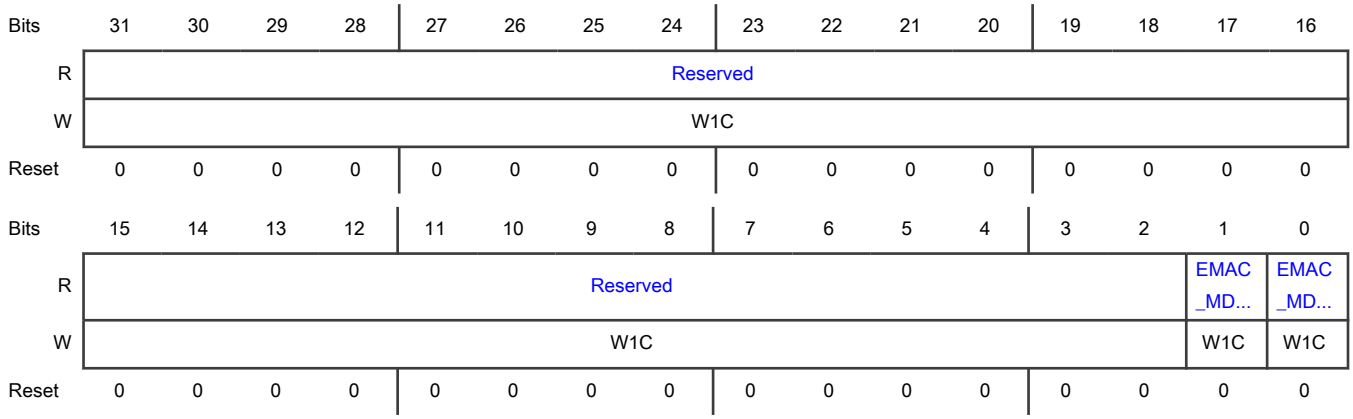
### 37.2.6 Read Only GPR On Functional Reset Register (DCMROF1)

**Offset**

| Register | Offset |
|---|---|
| DCMROF1 | 300h |

**Function**
Resets on functional reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>EMAC_MDC_CHID_1 | EMAC DMA Channel ID1 Status<br>0b - Channel ID1 is not the current transfer channel ID.<br>1b - Channel ID1 is the current transfer channel ID. |
| 0<br>EMAC_MDC_CHID_0 | EMAC DMA Channel ID0 Status<br>0b - Channel ID0 is not the current transfer channel ID.<br>1b - Channel ID0 is the current transfer channel ID. |

## 37.2.7 Read Only GPR On Functional Reset Register (DCMROF2)

### Offset

| Register | Offset |
|---|---|
| DCMROF2 | 304h |

### Function

Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID0 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID0 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>DCF_SDID0 | Configuration bits of DCF client SDID 0.<br>This register indicates the SDID0 contents scanned from flash by DCM<br>**NOTE**<br>See the DCF clients file attached to this document for more information. These are DCF clients in Utest sector which contains Device Configurable ID information and are being captured as status on Read Only GPR on Functional Reset. |

## 37.2.8 Read Only GPR On Functional Reset Register (DCMROF3)

### Offset

| Register | Offset |
|---|---|
| DCMROF3 | 308h |

### Function
Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID1 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID1 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>DCF_SDID1 | Configuration bits of DCF client SDID 1.<br>This register indicates the SDID1 contents scanned from flash by DCM.<br>**NOTE**<br>See the DCF clients file attached to this document for more information. These are DCF clients in Utest sector which contains Device Configurable ID information and are being captured as status on Read Only GPR on Functional Reset. |

## 37.2.9 Read Only GPR On Functional Reset Register (DCMROF4)

### Offset

| Register | Offset |
|---|---|
| DCMROF4 | 30Ch |

### Function
Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID2 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID2 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>DCF_SDID2 | Configuration bits of DCF client SDID 2.<br>This register indicates the SDID2 contents scanned from flash by DCM.<br>**NOTE**<br>See the DCF clients file attached to this document for more information. These are DCF clients in Utest sector which contains Device Configurable ID information and are being captured as status on Read Only GPR on Functional Reset. |

## 37.2.10 Read Only GPR On Functional Reset Register (DCMROF5)

### Offset

| Register | Offset |
|---|---|
| DCMROF5 | 310h |

### Function
Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID3 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID3 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>DCF_SDID3 | Configuration bits of DCF client SDID 3.<br>This register indicates the SDID3 contents scanned from flash by DCM.<br>**NOTE**<br>See the DCF clients file attached to this document for more information. These are DCF clients in Utest sector which contains Device Configurable ID information and are being captured as status on Read Only GPR on Functional Reset. |

## 37.2.11 Read Only GPR On Functional Reset Register (DCMROF6)

### Offset

| Register | Offset |
|---|---|
| DCMROF6 | 314h |

### Function
Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID4 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID4 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>DCF_SDID4 | Configuration bits of DCF client SDID 4.<br>This register indicates the SDID4 contents scanned from flash by DCM.<br>**NOTE**<br>See the DCF clients file attached to this document for more information. These are DCF clients in Utest sector which contains Device Configurable ID information and are being captured as status on Read Only GPR on Functional Reset. |

## 37.2.12 Read Only GPR On Functional Reset Register (DCMROF7)

### Offset

| Register | Offset |
|---|---|
| DCMROF7 | 318h |

### Function
Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID5 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID5 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>DCF_SDID5 | Configuration bits of DCF client SDID 5.<br>This register indicates the SDID5 contents scanned from flash by DCM.<br>**NOTE**<br>See the DCF clients file attached to this document for more information. These are DCF clients in Utest sector which contains Device Configurable ID information and are being captured as status on Read Only GPR on Functional Reset. |

## 37.2.13 Read Only GPR On Functional Reset Register (DCMROF8)

### Offset

| Register | Offset |
|---|---|
| DCMROF8 | 31Ch |

### Function
Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID6 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID6 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>DCF_SDID6 | Configuration bits of DCF client SDID 6.<br>This register indicates the SDID6 contents scanned from flash by DCM.<br>**NOTE**<br>See the DCF clients file attached to this document for more information. These are DCF clients in Utest sector which contains Device Configurable ID information and are being captured as status on Read Only GPR on Functional Reset. |

## 37.2.14 Read Only GPR On Functional Reset Register (DCMROF9)

### Offset

| Register | Offset |
|---|---|
| DCMROF9 | 320h |

### Function
Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID7 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID7 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>DCF_SDID7 | Configuration bits of DCF client SDID 7.<br>This register indicates the SDID7 contents scanned from flash by DCM.<br>**NOTE**<br>See the DCF clients file attached to this document for more information. These are DCF clients in Utest sector which contains Device Configurable ID information and are being captured as status on Read Only GPR on Functional Reset. |

## 37.2.15 Read Only GPR On Functional Reset Register (DCMROF10)

### Offset

| Register | Offset |
|---|---|
| DCMROF10 | 324h |

### Function
Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID8 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID8 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>DCF_SDID8 | Configuration bits of DCF client SDID 8.<br>This register indicates the SDID8 contents scanned from flash by DCM.<br>**NOTE**<br>See the DCF clients file attached to this document for more information. These are DCF clients in Utest sector which contains Device Configurable ID information and are being captured as status on Read Only GPR on Functional Reset. |

## 37.2.16 Read Only GPR On Functional Reset Register (DCMROF11)

### Offset

| Register | Offset |
|---|---|
| DCMROF11 | 328h |

### Function
Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID9 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID9 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>DCF_SDID9 | Configuration bits of DCF client SDID 9.<br>This register indicates the SDID9 contents scanned from flash by DCM.<br>**NOTE**<br>See the DCF clients file attached to this document for more information. These are DCF clients in Utest sector which contains Device Configurable ID information and are being captured as status on Read Only GPR on Functional Reset. |

## 37.2.17 Read Only GPR On Functional Reset Register (DCMROF12)

### Offset

| Register | Offset |
|---|---|
| DCMROF12 | 32Ch |

### Function
Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID10 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID10 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>DCF_SDID10 | Configuration bits of DCF client SDID 10.<br>This register indicates the SDID10 contents scanned from flash by DCM.<br>**NOTE**<br>See the DCF clients file attached to this document for more information. These are DCF clients in Utest sector which contains Device Configurable ID information and are being captured as status on Read Only GPR on Functional Reset. |

## 37.2.18 Read Only GPR On Functional Reset Register (DCMROF13)

### Offset

| Register | Offset |
|---|---|
| DCMROF13 | 330h |

### Function
Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID11 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID11 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>DCF_SDID11 | Configuration bits of DCF client SDID 11.<br>This register indicates the SDID11 contents scanned from flash by DCM.<br>**NOTE**<br>See the DCF clients file attached to this document for more information. These are DCF clients in Utest sector which contains Device Configurable ID information and are being captured as status on Read Only GPR on Functional Reset. |

## 37.2.19 Read Only GPR On Functional Reset Register (DCMROF14)

### Offset

| Register | Offset |
|---|---|
| DCMROF14 | 334h |

### Function
Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID12 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID12 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>DCF_SDID12 | Configuration bits of DCF client SDID 12.<br>This register indicates the SDID12 contents scanned from flash by DCM.<br>**NOTE**<br>See the DCF clients file attached to this document for more information. These are DCF clients in Utest sector which contains Device Configurable ID information and are being captured as status on Read Only GPR on Functional Reset. |

## 37.2.20 Read Only GPR On Functional Reset Register (DCMROF15)

### Offset

| Register | Offset |
|---|---|
| DCMROF15 | 338h |

### Function
Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID13 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID13 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>DCF_SDID13 | Configuration bits of DCF client SDID 13.<br>This register indicates the SDID13 contents scanned from flash by DCM.<br>**NOTE**<br>See the DCF clients file attached to this document for more information. These are DCF clients in Utest sector which contains Device Configurable ID information and are being captured as status on Read Only GPR on Functional Reset. |

## 37.2.21 Read Only GPR On Functional Reset Register (DCMROF16)

### Offset

| Register | Offset |
|---|---|
| DCMROF16 | 33Ch |

### Function
Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID14 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID14 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>DCF_SDID14 | Configuration bits of DCF client SDID 14.<br>This register indicates the SDID14 contents scanned from flash by DCM.<br>**NOTE**<br>See the DCF clients file attached to this document for more information. These are DCF clients in Utest sector which contains Device Configurable ID information and are being captured as status on Read Only GPR on Functional Reset. |

## 37.2.22 Read Only GPR On Functional Reset Register (DCMROF17)

### Offset

| Register | Offset |
|---|---|
| DCMROF17 | 340h |

### Function
Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID15 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_SDID15 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>DCF_SDID15 | Configuration bits of DCF client SDID 15.<br>This register indicates the SDID15 contents scanned from flash by DCM.<br>**NOTE**<br>See the DCF clients file attached to this document for more information. These are DCF clients in Utest sector which contains Device Configurable ID information and are being captured as status on Read Only GPR on Functional Reset. |

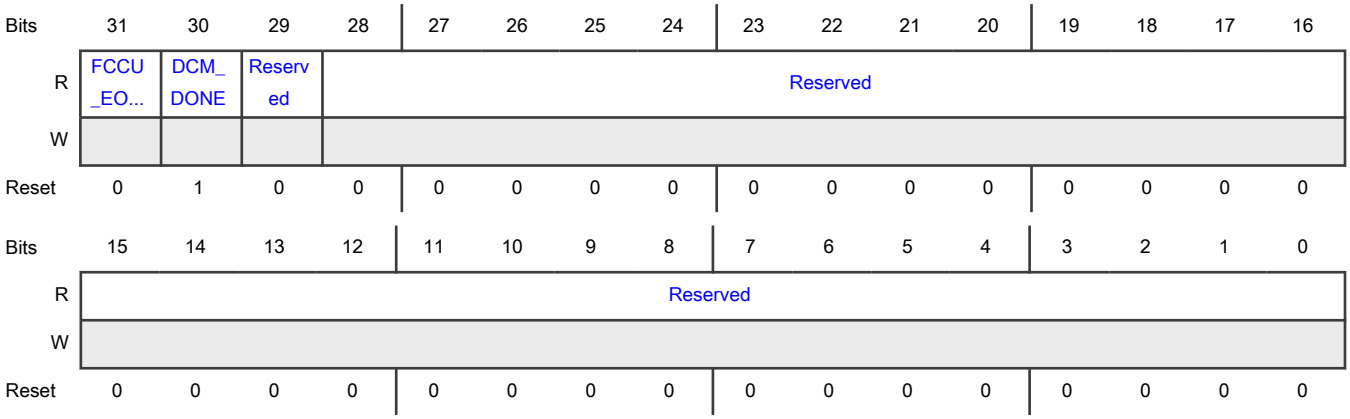
## 37.2.23 Read Only GPR On Functional Reset Register (DCMROF19)

### Offset

| Register | Offset |
|---|---|
| DCMROF19 | 348h |

### Function
Resets on functional reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>FCCU_EOUT_DEDICATED | FCCU EOUT Dedicated<br>Indicates the status of FCCU_EOUT functions on GPIO2 and GPIO3 as configured in DCF record, UTEST_MISC[FCCU_EOUT_DEDICATED].<br>0b - FCCU_EOUT pins on GPIO2 and GPIO3 acts as general purpose pins supporting all functions.<br>1b - FCCU_EOUT pins on GPIO2 and GPIO3 acts as dedicated EOUT pins. |
| 30<br>DCM_DONE | DCM Done<br>Status of flash scanning by DCM.<br>0b - Flash scanning by DCM not complete.<br>1b - Flash scanning completed by DCM. |
| 29<br>— | Reserved |
| 28-0<br>— | Reserved |

## 37.2.24 Read Only GPR On Functional Reset Register (DCMROF20)

### Offset

| Register | Offset |
|---|---|
| DCMROF20 | 34Ch |

### Function

Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCF_DEST_RST_ESC | | | | | | | | | | | | | | Reserved | Reserved |
| W | | | | | | | | | | | | | | | | |
| Reset | u[1] | u | u | u | u | u | u | u | u | u | u | u | u | u | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | Reserved | | | | | AIPS_IA... | QSPI_IA... | Reserved | DMA_AXB... | Reserved | LMAUTO_... | POR_WDG... |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | u[1] | u[1] | 0 | u[1] | 0 | u[1] | u[1] |

1. The reset value of this register is dependent on DCF client default value.

### Fields

| Field | Function |
|---|---|
| 31-18<br>DCF_DEST_RST_ESC | This register indicates the information of chip destructive reset escalation support for destructive reset sources as configured in DCF record, DEST_RST_ESC[13:0]. See the DCF clients file attached to this document for the mapping of corresponding destructive reset event. A 1 (or 0) in the corresponding field indicates that the destructive reset escalation feature is enabled (or disabled) for that corresponding destructive reset event.<br>00_0000_0000_0000b - Destructive Reset Escalation disabled.<br>00_0000_0000_0001b - Destructive Reset Escalation enabled. |
| 17<br>— | Reserved |
| 16<br>— | Reserved |
| 15-12<br>— | Reserved |
| 11-7<br>— | Reserved |
| 6<br>AIPS_IAHB_BYP | Status of AIPS1/2 IAHB gasket as configured in DCF record, UTEST_MISC[AIPS_IAHB_BYP].<br>0b - Register wall enabled.<br>1b - Register wall bypassed. |
| 5 | Status of QSPI IAHB gasket as configured in DCF record, UTEST_MISC[QSPI_IAHB_BYP].<br>0b - Register wall enabled. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| QSPI_IAHB_BYP | 1b - Register wall bypassed. |
| 4<br>— | Reserved |
| 3<br>DMA_AXBS_IAHB_BYP | Status of DMA AXBS IAHB gasket as configured in DCF record, UTEST_MISC[DMA_AXBS_IAHB_BYP].<br>0b - Register wall enabled.<br>1b - Register wall bypassed. |
| 2<br>— | Reserved |
| 1<br>LMAUTO_DIS | PMC last mile automatic croosover from boot regulationfeature support.<br>0b - Last mile auto crossover feautre is supported for the chip.<br>1b - Last mile auto crossover feautre is not supported for the chip. |
| 0<br>POR_WDG_EN | Indicates the status of POR_WDG as configured in DCF record, UTEST_MISC[POR_WDG_EN].<br>0b - POR Watchdog Disabled.<br>1b - POR Watchdog Enabled (default). |

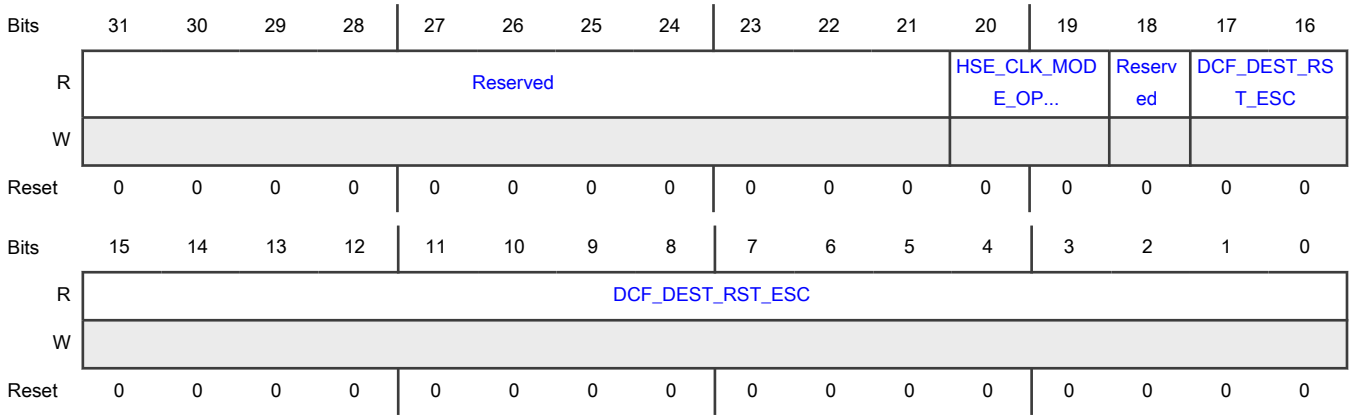
### 37.2.25 Read Only GPR On Functional Reset Register (DCMROF21)

**Offset**

| Register | Offset |
|---|---|
| DCMROF21 | 350h |

**Function**
Resets on functional reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-21<br>— | Reserved |
| 20-19<br>HSE_CLK_MODE_OPTION | HSE Clock Mode Option<br>00b - Applicable for clocking option A. Ratio of 1:2 in between HSE IPS interface clock (AIPS_SLOW_CLK) and HSE module clock (HSE_CLK), HSE_IAHB gasket enabled.<br>01b - Applicable for clocking option C, D, E, E2, and F. Ratio of 1:2 in between HSE IPS interface clock (AIPS_SLOW_CLK) and HSE module clock (HSE_CLK), HSE_IAHB gasket bypass.<br>10b - 10b and 11b both are applicable for clocking option B in same way. Ratio of 1:4 in between HSE IPS interface clock (AIPS_SLOW_CLK) and HSE module clock (HSE_CLK), HSE_IAHB gasket enabled.<br>11b - 10b and 11b both are applicable for clocking option B in same way. Ratio of 1:4 in between HSE IPS interface clock (AIPS_SLOW_CLK) and HSE module clock (HSE_CLK), HSE_IAHB gasket enabled. |
| 18<br>— | Reserved |
| 17-0<br>DCF_DEST_RST_ESC | This register indicates the information of chip destructive reset escalation support for destructive reset sources as configured in DCF record, DEST_RST_ESC[31:14]. See the DCF clients file attached to this document for the mapping of corresponding destructive reset event. A 1 (or 0) in the corresponding field indicates that the destructive reset escalation feature is enabled (or disabled) for that corresponding destructive reset event.<br>00_0000_0000_0000_0000b - Destructive Reset Escalation disabled.<br>00_0000_0000_0000_0001b - Destructive Reset Escalation enabled. |

## 37.2.26 Read Write GPR On Power On Reset Register (DCMRWP1)

### Offset

| Register | Offset |
|---|---|
| DCMRWP1 | 400h |

### Function
Resets on Power On reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | POR_WDOG_TRIM | | STANBDY... | Reserved | Reserved | Reserved | | CLKOUT_... | Reserved | | Reserved |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-11<br>— | Reserved |
| 10-9<br>POR_WDOG_TRIM | Trims for POR WDG timeout value.<br>POR_WDOG TRIM BITS:<br>00= POR WDOG Timeout= 06.25ms<br>01= POR WDOG Timeout = 12.50ms<br>10= POR WDOG Timeout = 25.00ms<br>11= POR WDOG Timeout = 50.00ms |
| 8<br>STANBDY_PWDOG_DIS | Disables the standby entry and exit monitoring window of the POR WDOG.<br>0b - Enabled<br>1b - Disabled |
| 7<br>— | Reserved |
| 6 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 5-4<br>— | Reserved |
| 3<br>CLKOUT_STANDBY | Clockout standby expose over functional and destructive reset.<br>This field controls the CLKOUT_STANDBY function availability while functional or destructive reset on PTA12.<br>0b - The CLKOUT_STANDBY function is not available while functional or destructive reset on PTA12.<br>1b - The CLKOUT_STANDBY function is available while functional or destructive reset on PTA12. |
| 2-1<br>— | Reserved |
| 0<br>— | Reserved |

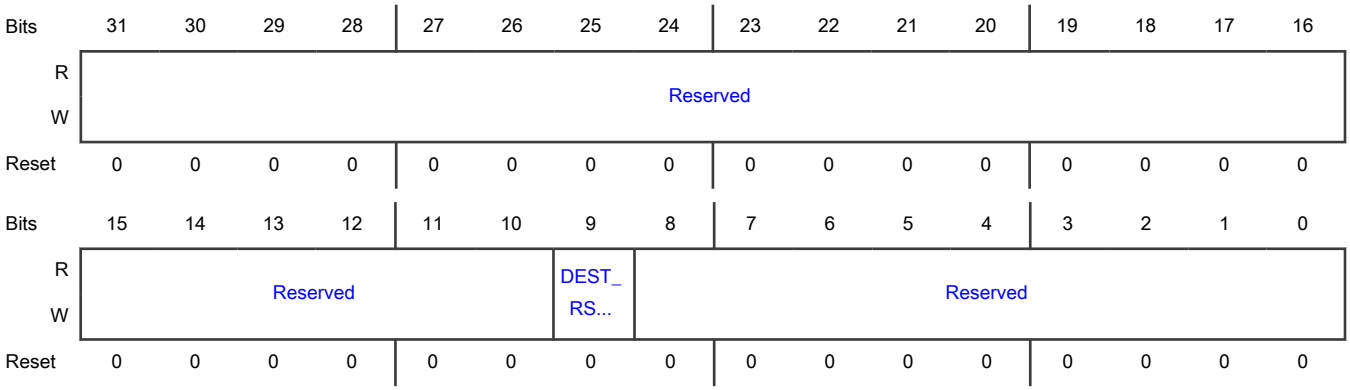
### 37.2.27 Read Write GPR On Power On Reset Register (DCMRWP3)

#### Offset

| Register | Offset |
|---|---|
| DCMRWP3 | 408h |

#### Function
Resets on Power On reset.

#### Diagram



### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>DEST_RST9_AS_IPI | Configures a destructive reset to interrupt.<br>0b - Destructive reset.<br>1b - PLL LOL interrupt. |
| 8-0<br>— | Reserved |

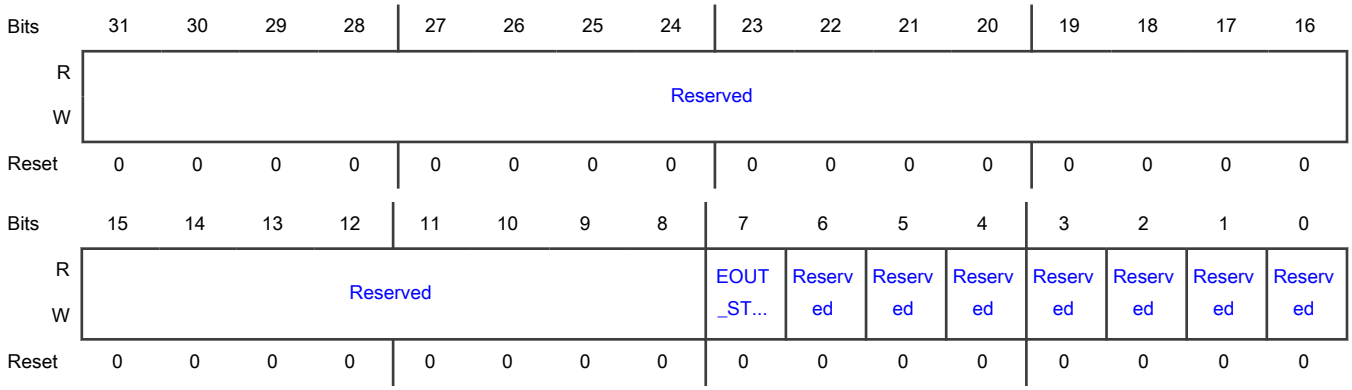
## 37.2.28 Read Write GPR On Destructive Reset Register (DCMRWD2)

### Offset

| Register | Offset |
|---|---|
| DCMRWD2 | 504h |

### Function
Resets on destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7 | Controls the EOUT state during selftest. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| EOUT_STAT_DUR_STEST | 0b - EOUT state will change to high Z post selftest when device is under reset.<br>1b - EOUT state will remain in FAULT state until this bit is cleared. |
| 6<br>— | Reserved |
| 5<br>— | Reserved |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>— | Reserved |

### 37.2.29 Read Write GPR On Destructive Reset Register (DCMRWD3)

**Offset**

| Register | Offset |
|---|---|
| DCMRWD3 | 508h |

**Function**
Fault disable fields. Resets on destructive reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | CM7_0_I... | Reserved | CM7_0_D... | Reserved | CM7_0_D... | PRAM0_E... | PRAM1_E... | Reserved | LC_ERR_... | Reserved | | PERIPH_... | EMAC_GS... | TCM_AXB... | DATA_ED... |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ADDR_ED... | AIPS2_G... | AIPS1_G... | QSPI_GS... | HSE_GSK... | Reserved | DMA_AXB... | SYS_AXB... | DMA_PER... | DMA_SYS... | TCM_GSK... | Reserved | Reserved | Reserved | Reserved | CM7_0_L... |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>CM7_0_ICDATA_ECC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 2 for the fault: Multi bit ECC error from CM7_0 ICache data memory.<br>0b - No multi-bit ECC error.<br>1b - Multi-bit ECC error. |
| 29<br>— | Reserved |
| 28<br>CM7_0_DCTAG_ECC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 2 for the fault: Multi bit ECC error from CM7_0 DCache tag memory.<br>0b - No multi-bit ECC error.<br>1b - Multi-bit ECC error. |
| 27<br>— | Reserved |
| 26<br>CM7_0_DCDATA_ECC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 2 for the fault: Multi bit ECC error from CM7_0 DCache data memory.<br>0b - No multi-bit ECC error.<br>1b - Multi-bit ECC error. |
| 25<br>PRAM0_ECC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 2 for the fault: Multi bit ECC error from SRAM0.<br>0b - No multi-bit ECC error.<br>1b - Multi-bit ECC error. |
| 24 | Enable bit for enabling the fault monitoring at FCCU NCF 2 for the fault: Multi bit ECC error from SRAM1. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| PRAM1_ECC_ERR_EN | 0b - No multi-bit ECC error.<br>1b - Multi-bit ECC error. |
| 23<br>— | Reserved |
| 22<br>LC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 3 for the fault: Error in lifecycle scanning.<br>**NOTE**<br>On any POR or destructive reset event, since this bit gets cleared, it has no effect and lifecycle error (in case if present) will not get disabled.<br>0b - No error while lifecycle scanning.<br>1b - Error while lifecycle scanning. |
| 21-20<br>— | Reserved |
| 19<br>PERIPH_AXBS_ALARM_EN | Enable bit for enabling the fault monitoring at FCCU NCF 1 for the fault: PERIPH AXBS_Lite safety alarm.<br>0b - No safety alarm indicated by Periph AXBS_Lite.<br>1b - Safety alarm indicated by Periph AXBS_Lite. |
| 18<br>EMAC_GSKT_ALARM_EN | Enable bit for enabling the fault monitoring at FCCU NCF 1 for the fault: EMAC IAHB gasket alarm.<br>0b - No alarm indicated by EMAC IAHB gasket.<br>1b - Alarm indicated by EMAC IAHB gasket. |
| 17<br>TCM_AXBS_ALARM_EN | Enable bit for enabling the fault monitoring at FCCU NCF 1 for the fault: TCM AHB splitter safety alarm.<br>0b - No safety alarm indicated by TCM AHB splitter.<br>1b - Safety alarm indicated by TCM AHB splitter. |
| 16<br>DATA_EDC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 1 for the fault: integrity error on data for safety.<br>0b - No integrity error on data.<br>1b - Data integrity error.. |
| 15<br>ADDR_EDC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 1 for the fault: integrity error on address for safety.<br>0b - No integrity error on address.<br>1b - Address integrity error. |
| 14<br>AIPS2_GSKT_ALARM_EN | Enable bit for enabling the fault monitoring at FCCU NCF 1 for the fault: AIPS2 IAHB gasket alarm.<br>0b - No alarm indicated by AIPS2 IAHB gasket.<br>1b - Alarm indicated by AIPS2 IAHB gasket. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 13<br>AIPS1_GSKT_ALARM_EN | Enable bit for enabling the fault monitoring at FCCU NCF 1 for the fault: AIPS1 IAHB gasket alarm.<br>0b - No alarm indicated by AIPS1 IAHB gasket.<br>1b - Alarm indicated by AIPS1 IAHB gasket. |
| 12<br>QSPI_GSKT_ALARM_EN | Enable bit for enabling the fault monitoring at FCCU NCF 1 for the fault: QSPI IAHB gasket alarm.<br>0b - No alarm indicated by QSPI IAHB gasket.<br>1b - Alarm indicated by QSPI IAHB gasket. |
| 11<br>HSE_GSKT_ALARM_EN | Enable bit for enabling the fault monitoring at FCCU NCF 1 for the fault: HSE IAHB gasket alarm.<br>0b - No alarm indicated by HSE IAHB gasket.<br>1b - Alarm indicated by HSE IAHB gasket. |
| 10<br>— | Reserved |
| 9<br>DMA_AXBS_ALARM_EN | Enable bit for enabling the fault monitoring at FCCU NCF 1 for the fault: DMA AXBS_Lite safety alarm.<br>0b - No safety alarm indicated by DMA AXBS_Lite.<br>1b - Safety alarm indicated by DMA AXBS_Lite. |
| 8<br>SYS_AXBS_ALARM_EN | Enable bit for enabling the fault monitoring at FCCU NCF 1 for the fault: system AXBS safety alarm.<br>0b - No safety alarm indicated by system AXBS.<br>1b - Safety alarm indicated by system AXBS. |
| 7<br>DMA_PERIPH_GSKT_ALARM_EN | Enable bit for enabling the fault monitoring at FCCU NCF 1 for the fault: IAHB gasket safety alarm from DMA periph AXBS IAHB gasket.<br>0b - No safety alarm reported from DMA-periph AXBS IAHB gasket.<br>1b - Safety alarm indicated by DMA-periph AXBS IAHB gasket. |
| 6<br>DMA_SYS_GSKT_ALARM_EN | Enable bit for enabling the fault monitoring at FCCU NCF 1 for the fault: IAHB gasket safety alarm from DMA system AXBS IHAB gasket.<br>0b - No safety alarm reported from DMA-system AXBS IAHB gasket.<br>1b - Safety alarm indicated by DMA-system AXBS IAHB gasket. |
| 5<br>TCM_GSKT_ALARM_EN | Enable bit for enabling the fault monitoring at FCCU NCF 1 for the fault: TCM IAHB Gasket monitor alarm.<br>0b - No alarm reported from TCM IAHB gasket.<br>1b - Monitor alarm reported from TCM IAHB gasket. |
| 4<br>— | Reserved |
| 3 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>CM7_0_LOCKUP_EN | Enable bit for enabling the fault monitoring at FCCU NCF 0 for the fault: CM7_0 core lockup.<br>0b - CM7_0 core not in lockup state.<br>1b - CM7_0 core in lockup state. |

### 37.2.30 Read Write GPR On Destructive Reset Register (DCMRWD4)

#### Offset

| Register | Offset |
|---|---|
| DCMRWD4 | 50Ch |

#### Function
Resets on destructive reset.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | TEST_AC... | TEST_AC... | Reserved | VDD2P5_... | VDD1P1_... | Reserved | FLASH_A... | Reserved | FLASH_S... | FLASH_R... | FLASH_R... | FLASH_A... | FLASH_E... | Reserved | Reserved |
| Reset | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PF1_DAT... | PF1_COD... | PF0_DAT... | PF0_COD... | HSE_RAM... | PRAM1_F... | PRAM0_F... | DMA_TCD... | Reserved | Reserved | Reserved | CM7_0_D... | CM7_0_D... | CM7_0_I... | Reserved | CM7_0_I... |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 30<br>TEST_ACTIVATION_1_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 5 for the fault: Accidental partial test activation.<br>0b - No partial test activated accidentally.<br>1b - Partial test activated accidentally. |
| 29<br>TEST_ACTIVATION_0_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 5 for the fault: Accidental partial test activation.<br>0b - No partial test activated accidentally.<br>1b - Partial test activated accidentally. |
| 28<br>— | Reserved |
| 27<br>VDD2P5_GNG_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 4 for the fault: Go/Nogo for VDD_HV_FLA (double bond) going to FXOSC and PLL.<br>0b - Go indication referring to the supply being clean.<br>1b - No go indication referring to the supply being unclean and a fault in double bond connection or its routing within the chip. |
| 26<br>VDD1P1_GNG_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 4 for the fault: Go/Nogo indicator for VDD1PD1 (double bond) supply going to PLL.<br>0b - Go indication referring to the supply being clean.<br>1b - No go indication referring to the supply being unclean and a fault in double bond connection or its routing within the chip. |
| 25<br>— | Reserved |
| 24<br>FLASH_ACCESS_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 3 for the fault: Transaction monitor mismatch error from flash controller. This alarm idicates that the flash controller detected a transaction monitor mismatch when compared to flash safety feedback output. There is an indication in the flash where reconstructed address is compared with the address which invoked the flash access.<br>0b - No transaction monitor mismatch error from flash controller.<br>1b - Transaction monitor mismatch error from flash controller. |
| 23<br>— | Reserved |
| 22<br>FLASH_SCAN_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 3 for the fault: Error while DCM flash scanning process due to invalid data.<br>**NOTE**<br>On any POR or destructive reset event, since this bit gets cleared, it has no effect and DCM flash scanning error (in case if present) will not get disabled. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No error while DCM flash scanning.<br>1b - Error while DCM flash scanning. |
| 21<br>FLASH_RST_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 3 for the fault: Flash reset error . This error indication is set when flash encounters errors during its reset reads.<br>0b - No flash reset error indicated.<br>1b - Flash reset error indicated. |
| 20<br>FLASH_REF_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 3 for the fault: Flash reference current loss or read voltage error while prevoius read(s).<br>0b - No reference current loss or read voltage error while previous reads(s).<br>1b - Reference current loss or read voltage error while previous read(s). |
| 19<br>FLASH_ADDR_ENC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 3 for the fault: Flash address encode error. In address decoding, if multiple or no address line is selected, FMU reports address encode error.<br>0b - No address encode error in flash.<br>1b - Address encode error in flash. |
| 18<br>FLASH_EDC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 3 for the fault: Flash ECC correction error through EDC reported by FMU.<br>0b - No EDC after ECC error reported in flash.<br>1b - EDC after ECC error reported in flash. |
| 17<br>— | Reserved |
| 16<br>— | Reserved |
| 15<br>PF1_DATA_ECC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 3 for the fault: Flash1 data ECC uncorrectable error.<br>0b - No uncorrectable error reported in flash controller port 1 data memory by FMU.<br>1b - Uncorrectable error reported in flash controller port 1 data memory by FMU. |
| 14<br>PF1_CODE_ECC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 3 for the fault: Flash1 code ECC uncorrectable error.<br>0b - No uncorrectable error reported in flash controller port 1 code memory by FMU.<br>1b - Uncorrectable error reported in flash controller port 1 code memory by FMU. |
| 13<br>PF0_DATA_ECC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 3 for the fault: Flash0 data ECC uncorrectable error.<br>0b - No uncorrectable error reported in flash controller port 0 data memory by FMU.<br>1b - Uncorrectable error reported in flash controller port 0 data memory by FMU. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 12<br>PF0_CODE_ECC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 3 for the fault: Flash0 code ECC uncorrectable error.<br>0b - No uncorrectable error reported in flash controller port 0 code memory by FMU.<br>1b - Uncorrectable error reported in flash controller port 0 code memory by FMU. |
| 11<br>HSE_RAM_ECC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 2 for the fault: HSE RAM Uncorrectable ECC.<br>0b - No uncorrectable ECC reported by HSE RAM.<br>1b - Uncorrectable ECC reported by HSE RAM |
| 10<br>PRAM1_FCCU_ALARM_EN | Enable bit for enabling the fault monitoring at FCCU NCF 2 for the fault: PRAM1 safety alarm. This alarm is set on faulty SRAM1 read or read modify error.<br>0b - No safety alarm indicated by PRAM1.<br>1b - Safety alarm indicated by PRAM1. |
| 9<br>PRAM0_FCCU_ALARM_EN | Enable bit for enabling the fault monitoring at FCCU NCF 2 for the fault: PRAM0 safety alarm. This alarm is set on faulty SRAM0 read or read modify error.<br>0b - No safety alarm indicated by PRAM0.<br>1b - Safety alarm indicated by PRAM0. |
| 8<br>DMA_TCD_RAM_ECC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 2 for the fault: Uncorrectable ECC error reported from DMA_TCD memory. This uncorrectable ECC error consists of multi-bit data ECC error and address ECC error.<br>0b - Uncorrectable ECC error detection at FCCU not enabled.<br>1b - Uncorrectable ECC error detection enabled at FCCU. |
| 7<br>— | Reserved |
| 6<br>— | Reserved |
| 5<br>— | Reserved |
| 4<br>CM7_0_DTCM1_ECC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 2 for the fault: Uncorrectable ECC error from CM7_0 Data TCM memory block 1. This uncorrectable ECC error consists of multi-bit data ECC error and address ECC error. The CM7_0 Data TCM physically consists of two blocks.<br>**NOTE**<br>Address ECC error is not supported on CM7_0 Data TCM memory in MCXE31B.<br>0b - Uncorrectable ECC error detection at FCCU not enabled.<br>1b - Uncorrectable ECC error detection enabled at FCCU. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 3<br>CM7_0_DTCM0_ECC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 2 for the fault: Uncorrectable ECC error from CM7_0 Data TCM memory block 0. This uncorrectable ECC error consists of multi-bit data ECC error and address ECC error. The CM7_0 Data TCM physically consists of two blocks.<br>**NOTE**<br>Address ECC error is not supported on CM7_0 Data TCM memory in MCXE314 and MCXE31B.<br>0b - Uncorrectable ECC error detection at FCCU not enabled.<br>1b - Uncorrectable ECC error detection enabled at FCCU. |
| 2<br>CM7_0_ITCM_ECC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 2 for the fault: Uncorrectable ECC error from CM7_0 Instruction TCM memory. This uncorrectable ECC error consists of multi-bit data ECC error and address ECC error.<br>**NOTE**<br>Address ECC error is not supported on CM7_0 Instruction TCM memory in MCXE314 and MCXE31B.<br>0b - Uncorrectable ECC error detection at FCCU not enabled.<br>1b - Uncorrectable ECC error detection enabled at FCCU. |
| 1<br>— | Reserved |
| 0<br>CM7_0_ICTAG_ECC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 2 for the fault: Multi bit ECC error from CM7_0 ICache tag memory.<br>0b - No multi-bit ECC error.<br>1b - Multi-bit ECC error |

## 37.2.31 Read Write GPR On Destructive Reset Register (DCMRWD5)

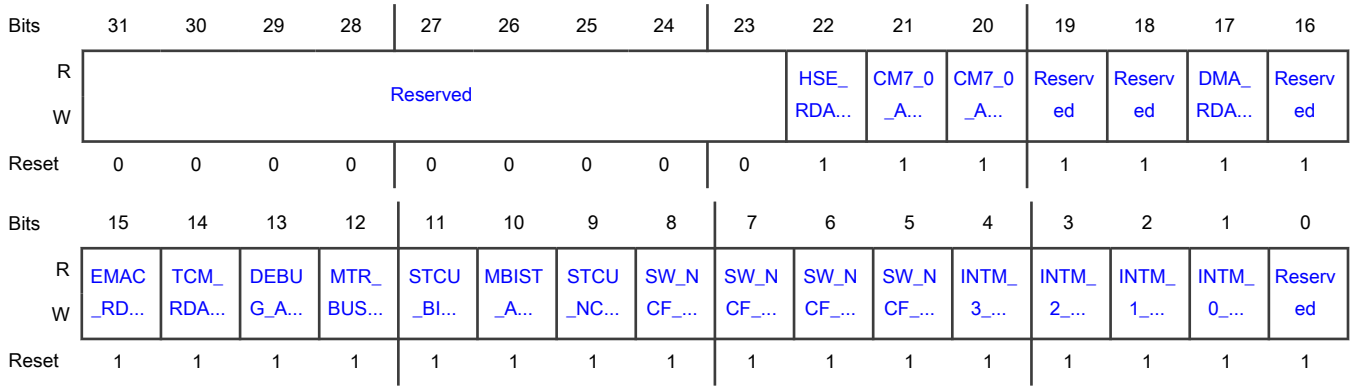
### Offset

| Register | Offset |
|---|---|
| DCMRWD5 | 510h |

### Function

Resets on destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-23<br>— | Reserved |
| 22<br>HSE_RDATA_EDC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 1 for the fault: Integrity (EDC) error on HSE read data for safety.<br>0b - No integrity error reported on HSE read data.<br>1b - Integrity error reported on HSE read data. |
| 21<br>CM7_0_AHBM_RDATA_EDC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 1 for the fault: Integrity error on CM7_0 main read data for safety.<br>0b - No integrity error reported on CM7_0 main read data.<br>1b - Integrity error reported on CM7_0 main read data. |
| 20<br>CM7_0_AHBP_RDATA_EDC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 1 for the fault: Integrity error on CM7_0 peripheral read data for safety.<br>0b - No integrity error reported on CM7_0 peripheral read data.<br>1b - Integrity error reported on CM7_0 peripheral read data. |
| 19<br>— | Reserved |
| 18<br>— | Reserved |
| 17<br>DMA_RDATA_EDC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 1 for the fault: Integrity(EDC) error on eDMA read data for safety.<br>0b - No integrity error reported on eDMA read data.<br>1b - Integrity error reported on eDMA read data. |
| 16 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 15<br>EMAC_RDATA_EDC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 1 for the fault: Integrity(EDC) error on EMAC read data for safety.<br>0b - No integrity error reported on EMAC read data.<br>1b - Integrity error reported on EMAC read data. |
| 14<br>TCM_RDATA_EDC_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 1 for the fault: Integrity(EDC) error on TCM read data for safety.<br>0b - No integrity error reported on TCM read data.<br>1b - Integrity error reported on TCM read data |
| 13<br>DEBUG_ACTIVATION_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 5 for the fault: Monitoring of unintended debug activation. This is set when the core is in halted state with application debug not enabled or debugger request not enabled.<br>**NOTE**<br>While debugger is connected, DCM.DCMRWD5[DEBUG_ACTIVATION_ERR_EN] should be configured as 0, to disable debug activation error monitoring, since debugger is intentionally connected to the chip.<br>0b - Unintended debug not activated.<br>1b - Unintended debug activated. |
| 12<br>MTR_BUS_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 5 for the fault: Fault reported due to illegal access on MTR. This fault is reported via a transfer error indication to the system.<br>0b - No transfer error indicated from MTR.<br>1b - Transfer error indicated from MTR. |
| 11<br>STCU_BIST_USER_CF_EN | Enable bit for enabling the fault monitoring at FCCU NCF 5 for the fault: L/M BIST enabled accidentally. Fault condition in run mode (LBIST or MBIST control signals go to wrong condition).<br>0b - No accidental L/M BIST enabled.<br>1b - Accidental L/M BIST enabled. |
| 10<br>MBIST_ACTIVATION_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 5 for the fault: Indicates an accidental backdoor access on memories. This monitor needs to be disabled on FCCU when doing fault injection.<br>0b - No accidental backdoor enabled on memories.<br>1b - Accidental backdoor enabled on memories. |
| 9<br>STCU_NCF_EN | Enable bit for enabling the fault monitoring at FCCU NCF 5 for the fault: STCU non-critical fault, that is, BIST result error.<br>0b - STCU non-critical fault not reported.<br>1b - STCU non-critical fault reported. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 8<br>SW_NCF_3_EN | Enable bit for enabling the fault monitoring at FCCU NCF 7 for the fault: Software NCF3 . DCMRWF1[FCCU_SW_NCF3].<br>0b - Software NCF3, DCMRWF1[FCCU_SW_NCF3] not enabled by software.<br>1b - Software NCF3, DCMRWF1[FCCU_SW_NCF3] enabled by software. |
| 7<br>SW_NCF_2_EN | Enable bit for enabling the fault monitoring at FCCU NCF 7 for the fault: Software NCF2 . DCMRWF1[FCCU_SW_NCF2]<br>0b - Software NCF2, DCMRWF1[FCCU_SW_NCF2] not enabled by software.<br>1b - Software NCF2, DCMRWF1[FCCU_SW_NCF2] enabled by software. |
| 6<br>SW_NCF_1_EN | Enable bit for enabling the fault monitoring at FCCU NCF 7 for the fault: Software NCF1 . DCMRWF1[FCCU_SW_NCF1].<br>0b - Software NCF1, DCMRWF1[FCCU_SW_NCF1] not enabled by software.<br>1b - Software NCF1, DCMRWF1[FCCU_SW_NCF1] enabled by software. |
| 5<br>SW_NCF_0_EN | Enable bit for enabling the fault monitoring at FCCU NCF 7 for the fault: Software NCF0 . DCMRWF1[FCCU_SW_NCF0].<br>0b - Software NCF0, DCMRWF1[FCCU_SW_NCF0] not enabled by software.<br>1b - Software NCF0, DCMRWF1[FCCU_SW_NCF0] enabled by software. |
| 4<br>INTM_3_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 6 for the fault: Interrupt monitor3 error reported by error reported by INTM. Also this error is reflected in INTM.INTM_STATUS3. See 'Functional Description' section in the INTM chapter for details.<br>0b - No error reported by interrupt monitor 3.<br>1b - Error reported by interrupt monitor 3. |
| 3<br>INTM_2_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 6 for the fault: Interrupt monitor2 error reported by error reported by INTM. Also this error is reflected in INTM.INTM_STATUS2. See 'Functional Description' section in the INTM chapter for details.<br>0b - No error reported by interrupt monitor 2.<br>1b - Error reported by interrupt monitor 2. |
| 2<br>INTM_1_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 6 for the fault: Interrupt monitor1 error reported by error reported by INTM. Also this error is reflected in INTM.INTM_STATUS1. See 'Functional Description' section in the INTM chapter for details.<br>0b - No error reported by interrupt monitor 1.<br>1b - Error reported by interrupt monitor 1. |
| 1<br>INTM_0_ERR_EN | Enable bit for enabling the fault monitoring at FCCU NCF 6 for the fault: Interrupt monitor0 error reported by INTM. Also this error is reflected in INTM.INTM_STATUS0. See 'Functional Description' section in the INTM chapter for details.<br>0b - No error reported by interrupt monitor 0.<br>1b - Error reported by interrupt monitor 0. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 0<br>— | Reserved |

## 37.2.32 Read Write GPR On Destructive Reset Register (DCMRWD6)

### Offset

| Register | Offset |
|---|---|
| DCMRWD6 | 514h |

### Function
Resets on destructive reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SAI1_DB... | SAI0_DB... | FLEXCAN... | FLEXCAN... | FLEXCAN... | FLEXCAN... | FLEXCAN... | FLEXCAN... | FLEXIO_... | LPI2C1_... | LPI2C0_... | LPSPI5_... | LPSPI4_... | LPSPI3_... | LPSPI2_... | LPSPI1_... |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LPSPI0_... | PIT2_DB... | PIT1_DB... | PIT0_DB... | STM1_DB... | STM0_DB... | SWT1_DB... | SWT0_DB... | RTC_DBG... | EMIOS2_... | EMIOS1_... | EMIOS0_... | LCU1_DB... | LCU0_DB... | FCCU_DB... | EDMA_DB... |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>SAI1_DBG_DIS_CM7_0 | SAI1 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - SAI1 enters debug mode when CM7_0 enters debug mode.<br>1b - SAI1 remains functional and is not impacted when CM7_0 enters debug mode. |
| 30<br>SAI0_DBG_DIS_CM7_0 | SAI0 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - SAI0 enters debug mode when CM7_0 enters debug mode.<br>1b - SAI0 remains functional and is not impacted when CM7_0 enters debug mode. |
| 29<br>FLEXCAN5_DBG_DIS_CM7_0 | FLEXCAN5 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - FLEXCAN5 enters debug mode when CM7_0 enters debug mode.<br>1b - FLEXCAN5 remains functional and is not impacted when CM7_0 enters debug mode. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 28<br>FLEXCAN4_DBG_DIS_CM7_0 | FLEXCAN4 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - FLEXCAN4 enters debug mode when CM7_0 enters debug mode.<br>1b - FLEXCAN4 remains functional and is not impacted when CM7_0 enters debug mode. |
| 27<br>FLEXCAN3_DBG_DIS_CM7_0 | FLEXCAN3 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - FLEXCAN3 enters debug mode when CM7_0 enters debug mode.<br>1b - FLEXCAN3 remains functional and is not impacted when CM7_0 enters debug mode. |
| 26<br>FLEXCAN2_DBG_DIS_CM7_0 | FLEXCAN2 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - FLEXCAN2 enters debug mode when CM7_0 enters debug mode.<br>1b - FLEXCAN2 remains functional and is not impacted when CM7_0 enters debug mode. |
| 25<br>FLEXCAN1_DBG_DIS_CM7_0 | FLEXCAN1 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - FLEXCAN1 enters debug mode when CM7_0 enters debug mode.<br>1b - FLEXCAN1 remains functional and is not impacted when CM7_0 enters debug mode. |
| 24<br>FLEXCAN0_DBG_DIS_CM7_0 | FLEXCAN0 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - FLEXCAN0 enters debug mode when CM7_0 enters debug mode.<br>1b - FLEXCAN0 remains functional and is not impacted when CM7_0 enters debug mode. |
| 23<br>FLEXIO_DBG_DIS_CM7_0 | FLEXIO debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - FLEXIO enters debug mode when CM7_0 enters debug mode.<br>1b - FLEXIO remains functional and is not impacted when CM7_0 enters debug mode. |
| 22<br>LPI2C1_DBG_DIS_CM7_0 | LPI2C1 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - LPI2C1 enters debug mode when CM7_0 enters debug mode.<br>1b - LPI2C1 remains functional and is not impacted when CM7_0 enters debug mode. |
| 21<br>LPI2C0_DBG_DIS_CM7_0 | LPI2C0 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - LPI2C0 enters debug mode when CM7_0 enters debug mode.<br>1b - LPI2C0 remains functional and is not impacted when CM7_0 enters debug mode. |
| 20<br>LPSPI5_DBG_DIS_CM7_0 | LPSPI5 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - LPSPI5 enters debug mode when CM7_0 enters debug mode.<br>1b - LPSPI5 remains functional and is not impacted when CM7_0 enters debug mode. |
| 19<br>LPSPI4_DBG_DIS_CM7_0 | LPSPI4 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - LPSPI4 enters debug mode when CM7_0 enters debug mode.<br>1b - LPSPI4 remains functional and is not impacted when CM7_0 enters debug mode. |
| 18 | LPSPI3 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| LPSPI3_DBG_DIS_CM7_0 | 0b - LPSPI3 enters debug mode when CM7_0 enters debug mode.<br>1b - LPSPI3 remains functional and is not impacted when CM7_0 enters debug mode. |
| 17<br>LPSPI2_DBG_DIS_CM7_0 | LPSPI2 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - LPSPI2 enters debug mode when CM7_0 enters debug mode.<br>1b - LPSPI2 remains functional and is not impacted when CM7_0 enters debug mode. |
| 16<br>LPSPI1_DBG_DIS_CM7_0 | LPSPI1 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - LPSPI1 enters debug mode when CM7_0 enters debug mode.<br>1b - LPSPI1 remains functional and is not impacted when CM7_0 enters debug mode. |
| 15<br>LPSPI0_DBG_DIS_CM7_0 | LPSPI0 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - LPSPI0 enters debug mode when CM7_0 enters debug mode.<br>1b - LPSPI0 remains functional and is not impacted when CM7_0 enters debug mode. |
| 14<br>PIT2_DBG_DIS_CM7_0 | PIT2 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - PIT2 enters debug mode when CM7_0 enters debug mode.<br>1b - PIT2 remains functional and is not impacted when CM7_0 enters debug mode. |
| 13<br>PIT1_DBG_DIS_CM7_0 | PIT1 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - PIT1 enters debug mode when CM7_0 enters debug mode.<br>1b - PIT1 remains functional and is not impacted when CM7_0 enters debug mode. |
| 12<br>PIT0_DBG_DIS_CM7_0 | PIT0 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - PIT0 enters debug mode when CM7_0 enters debug mode.<br>1b - PIT0 remains functional and is not impacted when CM7_0 enters debug mode. |
| 11<br>STM1_DBG_DIS_CM7_0 | STM1 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - STM1 enters debug mode when CM7_0 enters debug mode.<br>1b - STM1 remains functional and is not impacted when CM7_0 enters debug mode. |
| 10<br>STM0_DBG_DIS_CM7_0 | STM0 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - STM0 enters debug mode when CM7_0 enters debug mode.<br>1b - STM0 remains functional and is not impacted when CM7_0 enters debug mode. |
| 9<br>SWT1_DBG_DIS_CM7_0 | SWT1 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - SWT0 enters debug mode when CM7_0 enters debug mode.<br>1b - SWT0 remains functional and is not impacted when CM7_0 enters debug mode. |
| 8<br>SWT0_DBG_DIS_CM7_0 | SWT0 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - SWT0 enters debug mode when CM7_0 enters debug mode.<br>1b - SWT0 remains functional and is not impacted when CM7_0 enters debug mode. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 7<br>RTC_DBG_DIS_CM7_0 | RTC debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - RTC enters debug mode when CM7_0 enters debug mode.<br>1b - RTC remains functional and is not impacted when CM7_0 enters debug mode. |
| 6<br>EMIOS2_DBG_DIS_CM7_0 | EMIOS2 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - eMIOS2 enters debug mode when CM7_0 enters debug mode.<br>1b - eMIOS2 remains functional and is not impacted when CM7_0 enters debug mode. |
| 5<br>EMIOS1_DBG_DIS_CM7_0 | EMIOS1 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - eMIOS1 enters debug mode when CM7_0 enters debug mode.<br>1b - eMIOS1 remains functional and is not impacted when CM7_0 enters debug mode. |
| 4<br>EMIOS0_DBG_DIS_CM7_0 | EMIOS0 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - eMIOS0 enters debug mode when CM7_0 enters debug mode.<br>1b - eMIOS0 remains functional and is not impacted when CM7_0 enters debug mode. |
| 3<br>LCU1_DBG_DIS_CM7_0 | LCU1 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - LCU1 enters debug mode when CM7_0 enters debug mode.<br>1b - LCU1 remains functional and is not impacted when CM7_0 enters debug mode. |
| 2<br>LCU0_DBG_DIS_CM7_0 | LCU0 debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - LCU0 enters debug mode when CM7_0 enters debug mode.<br>1b - LCU0 remains functional and is not impacted when CM7_0 enters debug mode. |
| 1<br>FCCU_DBG_DIS_CM7_0 | FCCU debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - FCCU enters debug mode when CM7_0 enters debug mode.<br>1b - FCCU remains functional and is not impacted when CM7_0 enters debug mode. |
| 0<br>EDMA_DBG_DIS_CM7_0 | EDMA debug disable bit for CM7_0. Set this bit 1 to disable the debug of IP.<br>0b - EDMA enters debug mode when CM7_0 enters debug mode.<br>1b - EDMA remains functional and is not impacted when CM7_0 enters debug mode. |

### 37.2.33 Read Write GPR On Destructive Reset Register (DCMRWD7)

**Offset**

| Register | Offset |
|---|---|
| DCMRWD7 | 518h |

**Function**
Resets on destructive reset.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-21<br>— | Reserved |
| 20<br>— | Reserved |
| 19<br>— | Reserved |
| 18<br>— | Reserved |
| 17<br>— | Reserved |
| 16<br>— | Reserved |
| 15<br>— | Reserved |
| 14<br>— | Reserved |
| 13<br>— | Reserved |
| 12<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 11<br>— | Reserved |
| 10<br>— | Reserved |
| 9<br>— | Reserved |
| 8<br>— | Reserved |
| 7<br>— | Reserved |
| 6<br>— | Reserved |
| 5<br>— | Reserved |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>— | Reserved |

### 37.2.34 Read Write GPR On Destructive Reset Register (DCMRWD8)

**Offset**

| Register | Offset |
|---|---|
| DCMRWD8 | 51Ch |

### Function

Resets on destructive reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>— | Reserved |
| 29<br>— | Reserved |
| 28<br>— | Reserved |
| 27<br>— | Reserved |
| 26<br>— | Reserved |
| 25<br>— | Reserved |
| 24<br>— | Reserved |
| 23<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 22<br>— | Reserved |
| 21<br>— | Reserved |
| 20<br>— | Reserved |
| 19<br>— | Reserved |
| 18<br>— | Reserved |
| 17<br>— | Reserved |
| 16<br>— | Reserved |
| 15<br>— | Reserved |
| 14<br>— | Reserved |
| 13<br>— | Reserved |
| 12<br>— | Reserved |
| 11<br>— | Reserved |
| 10<br>— | Reserved |
| 9<br>— | Reserved |
| 8 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 7<br>— | Reserved |
| 6<br>— | Reserved |
| 5<br>— | Reserved |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>— | Reserved |

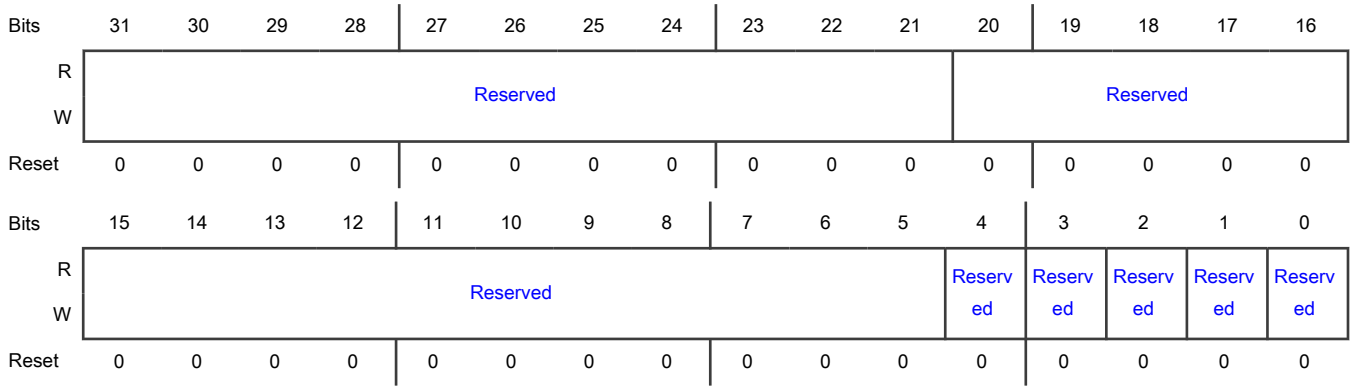
### 37.2.35 Read Write GPR On Destructive Reset Register (DCMRWD9)

**Offset**

| Register | Offset |
|---|---|
| DCMRWD9 | 520h |

**Function**
Resets on destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-21<br>— | Reserved |
| 20-5<br>— | Reserved |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>— | Reserved |

## 37.2.36 Read Write GPR On Functional Reset Register (DCMRWF1)

### Offset

| Register | Offset |
|---|---|
| DCMRWF1 | 600h |

### Function

Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | Reserved | | | VDD_1_5... | VDD_HV_... | VDD_HV_... | VSS_LV_... | SUPPLY_MON_SEL | | | SUPPLY_... | Reserved | | Reserved | STANDBY... |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | VDD_HV_... | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | RMII_MI... | Reserved | FCCU_SW... | FCCU_SW... | FCCU_SW... | FCCU_SW... | CAN_TIM... | CAN_TIM... |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-28<br>— | Reserved |
| 27<br>VDD_1_5_VLT_DVDR_EN | Enable for VDD1P5 2:1 divider for voltage measurement using supply voltage monitoring by ADC.<br>0b - Disabled<br>1b - Enabled |
| 26<br>VDD_HV_B_VLT_DVDR_EN | Enable for VDD_HV_B 2:1 divider for voltage measurement using supply voltage monitoring by ADC.<br>0b - Disabled<br>1b - Enabled |
| 25<br>VDD_HV_A_VLT_DVDR_EN | Enable for VDD_HV_A 2:1 divider for voltage measurement using supply voltage monitoring by ADC.<br>0b - Disabled<br>1b - Enabled |
| 24<br>VSS_LV_ANMUX_EN | Enable VSS_LV monitoring.<br>**NOTE**<br>This field must be set alongwith SUPPLY_MON_EN configured as 1b0 for VSS_LV monitoring by ADC0.<br>0b - Disabled<br>1b - Enabled |
| 23-21<br>SUPPLY_MON_SEL | Selects the source of voltage used by ADC for supply monitoring. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>The SUPPLY_MON_SEL configurations are effective only when SUPPLY_MON_EN is configured as 1b1.<br>When SUPPLY_MON_EN is configured as 0 and VSS_LV_ANMUX_EN is configured as 1 then VSS_LV is monitored.<br>000b - VDD_HV_A_DIV<br>001b - VDD_HV_B_DIV<br>010b - VDD_1.5_DIV<br>011b - VDD_2.5_OSC<br>100b - VDD1.1_PD1_HOT_POINT<br>101b - VDD1.1_PD1_COLD_POINT<br>110b - VDD1.1_PLL<br>111b - VDD1.1_PD0 |
| 20<br>SUPPLY_MON_EN | Enable the supply voltage monitoring by ADC.<br>0b - Disabled<br>1b - Enabled |
| 19-18<br>— | Reserved |
| 17<br>— | Reserved |
| 16<br>STANDBY_IO_CONFIG | Controls the IO state in the standby mode.<br>0b - STANDBY IO configuration standby entry: Must be written as 0 before IO configurations are done in standby entry sequence.<br>1b - STANDBY IO configuration standby exit: Must be written as 1 after IO configurations are done on standby exit. |
| 15<br>VDD_HV_B_IO_CTRL_LATCH | Controls the IO controls latching in low frequency RUN mode to reduce power consumption on VDD_HV_B domain pins.<br>**NOTE**<br>This bit should remain 0, except in FIRC 3MHz and FIRC 187.5kHz operation mode.<br>0b - VDD_HV_B domain pins function as normal.<br>1b - The IO controls of VDD_HV_B domain pins are latched. |
| 14<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 13<br>— | Reserved |
| 12<br>— | Reserved |
| 11<br>— | Reserved |
| 10<br>— | Reserved |
| 9<br>— | Reserved |
| 8<br>— | Reserved |
| 7<br>RMII_MII_SEL | Selects between MII and RMII mode of ethernet.<br>0b - MII mode<br>1b - RMII mode |
| 6<br>— | Reserved |
| 5<br>FCCU_SW_NCF3 | Control to initiate Software NCF to FCCU.<br>0b - NCF to FCCU will not be generated.<br>1b - NCF to FCCU will be generated. For exact slot of FCCU, please refer FCCU chapter. |
| 4<br>FCCU_SW_NCF2 | Control to initiate Software NCF to FCCU.<br>0b - NCF to FCCU will not be generated.<br>1b - NCF to FCCU will be generated. For exact slot of FCCU, please refer FCCU chapter. |
| 3<br>FCCU_SW_NCF1 | Control to initiate Software NCF to FCCU.<br>0b - NCF to FCCU will not be generated.<br>1b - NCF to FCCU will be generated. For exact slot of FCCU, please refer FCCU chapter. |
| 2<br>FCCU_SW_NCF0 | Control to initiate Software NCF to FCCU.<br>0b - NCF to FCCU will not be generated.<br>1b - NCF to FCCU will be generated. For exact slot of FCCU, please refer FCCU chapter. |
| 1 | Enables CAN timestamping feature for all flexcans. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| CAN_TIMESTAMP_EN | 0b - CAN timestamping is disabled.<br>1b - CAN timestamping is enabled. |
| 0<br>CAN_TIMESTAMP_SEL | Select between EMAC and STM for CAN timestamping<br>0b - EMAC selected for CAN timestamping.<br>1b - STM0 selected for CAN timestamping. |

## 37.2.37 Read Write GPR On Functional Reset Register (DCMRWF2)

### Offset

| Register | Offset |
|---|---|
| DCMRWF2 | 604h |

### Function
Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | | | | | | | HSE_GSK... |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | SIRC_TR... | PMC_TRI... | FIRC_TR... | DCM_SCA... | Reserved | | Reserved |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>— | Reserved |
| 29 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 28<br>— | Reserved |
| 27<br>— | Reserved |
| 26<br>— | Reserved |
| 25<br>— | Reserved |
| 24<br>— | Reserved |
| 23-17<br>— | Reserved |
| 16<br>HSE_GSKT_BYPASS | Control to enable the HSE IAHB gasket behavior out of standby mode.<br>**NOTE**<br>If configured as 1 to bypass HSE IAHB gasket out of standby, the system should continue to run on FIRC. If intended to be run on PLL, a functional reset should be asserted.<br>0b - When configured as 0, HSE IAHB gasket bypass configuration is controlled by DCF client.<br>1b - When configured as 1, HSE IAHB gasket is bypassed out of standby. |
| 15-7<br>— | Reserved |
| 6<br>SIRC_TRIM_BYP_STDBY_EXT | Control to bypass the SIRC trimming on standby exit.<br>0b - Not bypassed<br>1b - Bypassed |
| 5<br>PMC_TRIM_RGM_DCF__BYP_STDBY_EXT | Control to bypass the PMC trimming and RGM DCF loading on standby exit.<br>0b - Not bypassed<br>1b - Bypassed |
| 4<br>FIRC_TRIM_BYP_STDBY_EXT | Control to bypass the FIRC trimming on standby exit.<br>0b - Not bypassed<br>1b - Bypassed |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 3<br>DCM_SCAN_BYP_STDBY_EXT | Control to bypass the DCM scanning on standby exit.<br>0b - Not bypassed<br>1b - Bypassed |
| 2-1<br>— | Reserved |
| 0<br>— | Reserved |

## 37.2.38 Read Write GPR On Functional Reset Register (DCMRWF4)

### Offset

| Register | Offset |
|---|---|
| DCMRWF4 | 60Ch |

### Function
Resets on functional reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | Reserved | Reserved | CM7_0_C... | GLITCH_... |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GLITCH_... | GLITCH_... | GLITCH_... | Reserved | | MUX_MOD... | MUX_MOD... | Reserved | | MUX_MOD... | MUX_MOD... | MUX_MOD... | MUX_MOD... | MUX_MOD... | MUX_MOD... | Reserved |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 18<br>— | Reserved |
| 17<br>CM7_0_CPUWAIT | Configuration to put CM7_0 core into wait mode.<br>0b - CPUWAIT for CM7_0 disabled.<br>1b - CM7_0 put into wait mode using CPUWAIT. |
| 16<br>GLITCH_FIL_TRG_IN3_BYP | Selects whether to BYPASS or filter out the pulse.<br>0b - Glitch filter on TRGMUX input60 is enabled.<br>1b - Glitch filter on TRGMUX input60 is bypassed. |
| 15<br>GLITCH_FIL_TRG_IN2_BYP | Selects whether to BYPASS or filter out the pulse.<br>0b - Glitch filter on TRGMUX input61 is enabled.<br>1b - Glitch filter on TRGMUX input61 is bypassed. |
| 14<br>GLITCH_FIL_TRG_IN1_BYP | Selects whether to BYPASS or filter out the pulse.<br>0b - Glitch filter on TRGMUX input62 is enabled.<br>1b - Glitch filter on TRGMUX input62 is bypassed. |
| 13<br>GLITCH_FIL_TRG_IN0_BYP | Selects whether to BYPASS or filter out the pulse.<br>0b - Glitch filter on TRGMUX input63 is enabled.<br>1b - Glitch filter on TRGMUX input63 is bypassed. |
| 12-11<br>— | Reserved |
| 10<br>MUX_MODE_EN_ADC2_S9 | Selects the GPIO46 to drive adc2 standard channel 9th.<br>0b - Selects the default PAD 1.<br>1b - Selects the PAD mentioned in description. |
| 9<br>MUX_MODE_EN_ADC2_S8 | Selects the GPIO45 to drive adc2 standard channel 8th.<br>0b - Selects the default PAD 1.<br>1b - Selects the PAD mentioned in description. |
| 8-7<br>— | Reserved |
| 6<br>MUX_MODE_EN_ADC1_S23 | Selects the GPIO115 to drive ADC1 standard channel 23rd.<br>0b - Selects the default PAD 1.<br>1b - Selects the PAD mentioned in description. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 5<br>MUX_MODE_EN_ADC1_S22 | Selects the GPIO114 to drive ADC1 standard channel 22nd.<br>0b - Selects the default PAD 1.<br>1b - Selects the PAD mentioned in description. |
| 4<br>MUX_MODE_EN_ADC1_S15 | Selects the GPIO33 to drive ADC1 standard channel 15th.<br>0b - Selects the default PAD 1.<br>1b - Selects the PAD mentioned in description. |
| 3<br>MUX_MODE_EN_ADC1_S14 | Selects the GPIO32 to drive ADC1 standard channel 14th.<br>0b - Selects the default PAD 1.<br>1b - Selects the PAD mentioned in description. |
| 2<br>MUX_MODE_EN_ADC0_S9 | Selects the GPIO46 to drive ADC0 standard channel 9th.<br>0b - Selects the default PAD 1.<br>1b - Selects the PAD mentioned in description. |
| 1<br>MUX_MODE_EN_ADC0_S8 | Selects the GPIO45 to drive ADC0 standard channel 8th.<br>0b - Selects the default PAD 1.<br>1b - Selects the PAD mentioned in description. |
| 0<br>— | Reserved |

## 37.2.39 Read Write GPR On Functional Reset Register (DCMRWF5)

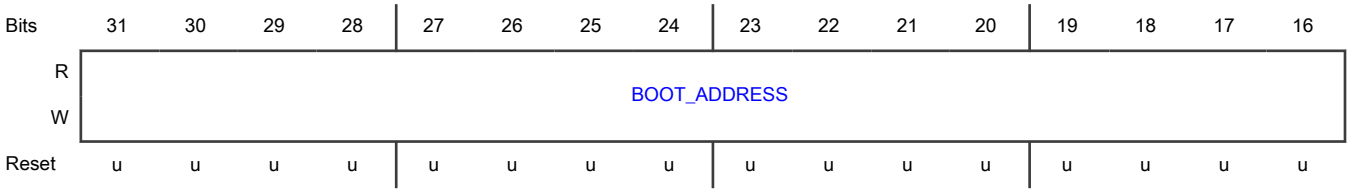
### Offset

| Register | Offset |
|---|---|
| DCMRWF5 | 610h |

### Function

Resets on functional reset.

### Diagram



| | Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | R<br>W | BOOT_ADDRESS | | | | | | | | | | | | | | | BOOT_MO... |
| | Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

1. The reset value is undefined on reset and gets loaded from the flash contents at the end of reset sequence.

### Fields

| Field | Function |
|---|---|
| 31-1<br>BOOT_ADDRESS | Cortex-M7_0 base address of vector table to be used after exiting Standby mode (only to be considered in Fast Standby mode. |
| 0<br>BOOT_MODE | Selects the boot mode after exiting Standby mode.<br>0b - Normal<br>1b - Fast Standby |

## 37.2.40 Read Only GPR On PMCPOR Reset (DCMROPP1)

### Offset

| Register | Offset |
|---|---|
| DCMROPP1 | 700h |

### Function
Resets on PMCPOR reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | POR_WDG... | POR_WDG... | POR_WDG... | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | POR_WDG... | Reserved | Reserved | POR_WDG... | Reserved |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | POR_WDG... | Reserved | Reserved | POR_WDG... | POR_WDG... | Reserved | Reserved | Reserved | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>POR_WDG_STAT31 | This bit captures MC_RGM reset event (if occured) while the device is in STANDBY mode.<br>**NOTE**<br>This bit is applicable only for standby sequence monitoring.<br>0b - MC_RGM reset event not detected or no reset event arrived in STANDBY mode when POR_WDG overflows.<br>1b - MC_RGM reset event is detected in STANDBY mode when POR_WDG overflows. |
| 30<br>POR_WDG_STAT30 | This bit captures the status of standby exit acknowledgement by MC_PCU when POR_WDG overflows.<br>**NOTE**<br>This bit is applicable only for standby sequence monitoring.<br>0b - MC_PCU has not acknowledged standby exit when POR_WDG overflows.<br>1b - MC_PCU has acknowledged standby exit when POR_WDG overflows. |
| 29<br>POR_WDG_STAT29 | This bit captures the status of standby entry request initiated by MC_ME when POR_WDG overflows.<br>**NOTE**<br>This bit is applicable only for standby sequence monitoring.<br>0b - MC_ME standby request is active when POR_WDG overflows.<br>1b - MC_ME standby request is not active when POR_WDG overflows. |
| 28<br>— | Reserved |
| 27<br>— | Reserved |
| 26<br>— | Reserved |
| 25<br>— | Reserved |
| 24<br>— | Reserved |
| 23<br>— | Reserved |
| 22<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 21<br>— | Reserved |
| 20<br>POR_WDG_STAT20 | This bit captures the status of functional reset sequence process DEST0 when POR_WDG overflows.<br>0b - DEST0 is inactive when POR_WDG overflows.<br>1b - DEST0 is active when POR_WDG overflows. |
| 19<br>— | Reserved |
| 18<br>— | Reserved |
| 17<br>POR_WDG_STAT17 | This bit captures the status of functional reset sequence process FUNC10 when POR_WDG overflows.<br>0b - FUNC10 is inactive when POR_WDG overflows.<br>1b - FUNC10 is active when POR_WDG overflows. |
| 16<br>— | Reserved |
| 15<br>— | Reserved |
| 14<br>POR_WDG_STAT14 | This bit captures the status of functional reset sequence process FUNC9 when POR_WDG overflows.<br>0b - FUNC9 is inactive when POR_WDG overflows.<br>1b - FUNC9 is active when POR_WDG overflows. |
| 13<br>— | Reserved |
| 12<br>— | Reserved |
| 11<br>POR_WDG_STAT11 | This bit captures the status of functional reset sequence process FUNC7 when POR_WDG overflows.<br>0b - FUNC8 is inactive when POR_WDG overflows.<br>1b - FUNC8 is active when POR_WDG overflows. |
| 10<br>POR_WDG_STAT10 | This bit captures the status of functional reset sequence process FUNC7 when POR_WDG overflows.<br>0b - FUNC7 is inactive when POR_WDG overflows.<br>1b - FUNC7 is active when POR_WDG overflows. |
| 9 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 8<br>— | Reserved |
| 7<br>— | Reserved |
| 6<br>POR_WDG_STAT6 | This bit captures the status of functional reset sequence process FUNC6 when POR_WDG overflows.<br>0b - FUNC6 is inactive when POR_WDG overflows.<br>1b - FUNC6 is active when POR_WDG overflows. |
| 5<br>POR_WDG_STAT5 | This bit captures the status of functional reset sequence process FUNC5 when POR_WDG overflows.<br>0b - FUNC5 is inactive when POR_WDG overflows.<br>1b - FUNC5 is active when POR_WDG overflows. |
| 4<br>POR_WDG_STAT4 | This bit captures the status of functional reset sequence process FUNC4 when POR_WDG overflows.<br>0b - FUNC4 is inactive when POR_WDG overflows.<br>1b - FUNC4 is active when POR_WDG overflows. |
| 3<br>POR_WDG_STAT3 | This bit captures the status of functional reset sequence process FUNC3 when POR_WDG overflows.<br>0b - FUNC3 is inactive when POR_WDG overflows.<br>1b - FUNC3 is active when POR_WDG overflows. |
| 2<br>POR_WDG_STAT2 | This bit captures the status of functional reset sequence process FUNC2 when POR_WDG overflows.<br>0b - FUNC2 is inactive when POR_WDG overflows.<br>1b - FUNC2 is active when POR_WDG overflows. |
| 1<br>POR_WDG_STAT1 | This bit captures the status of functional reset sequence process FUNC1 when POR_WDG overflows.<br>0b - FUNC1 is inactive when POR_WDG overflows.<br>1b - FUNC1 is active when POR_WDG overflows. |
| 0<br>POR_WDG_STAT0 | This bit captures the status of functional reset sequence process FUNC0 when POR_WDG overflows.<br>0b - FUNC0 is inactive when POR_WDG overflows.<br>1b - FUNC0 is active when POR_WDG overflows. |

### 37.2.41 Read Only GPR On PMCPOR Reset (DCMROPP2)

**Offset**

| Register | Offset |
|---|---|
| DCMROPP2 | 704h |

### Function

Resets on PMCPOR reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | POR_WDG... | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | POR_WDG... | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. | POR_WDG.. |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>POR_WDG_STAT63 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[31] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[31] is 1 when POR_WDG overflows. |
| 30<br>POR_WDG_STAT62 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[30] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[30] is 1 when POR_WDG overflows. |
| 29<br>POR_WDG_STAT61 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[29] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[29] is 1 when POR_WDG overflows. |
| 28<br>POR_WDG_STAT60 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[28] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[28] is 1 when POR_WDG overflows. |
| 27<br>POR_WDG_STAT59 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[27] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[27] is 1 when POR_WDG overflows. |
| 26<br>POR_WDG_STAT58 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[26] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[26] is 1 when POR_WDG overflows. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 25<br>POR_WDG_STAT57 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[25] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[25] is 1 when POR_WDG overflows. |
| 24<br>POR_WDG_STAT56 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[24] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[24] is 1 when POR_WDG overflows. |
| 23<br>POR_WDG_STAT55 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[23] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[23] is 1 when POR_WDG overflows. |
| 22<br>POR_WDG_STAT54 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[22] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[22] is 1 when POR_WDG overflows. |
| 21<br>POR_WDG_STAT53 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[21] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[21] is 1 when POR_WDG overflows. |
| 20<br>POR_WDG_STAT52 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[20] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[20] is 1 when POR_WDG overflows. |
| 19<br>POR_WDG_STAT51 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[19] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[19] is 1 when POR_WDG overflows. |
| 18<br>POR_WDG_STAT50 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[18] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[18] is 1 when POR_WDG overflows. |
| 17<br>POR_WDG_STAT49 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[17] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[17] is 1 when POR_WDG overflows. |
| 16<br>POR_WDG_STAT48 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[16] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[16] is 1 when POR_WDG overflows. |
| 15 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| POR_WDG_STAT47 | 0b - MC_RGM.RGM_FES[15] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[15] is 1 when POR_WDG overflows. |
| 14<br>POR_WDG_STAT46 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[14] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[14] is 1 when POR_WDG overflows. |
| 13<br>POR_WDG_STAT45 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[13] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[13] is 1 when POR_WDG overflows. |
| 12<br>POR_WDG_STAT44 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[12] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[12] is 1 when POR_WDG overflows. |
| 11<br>POR_WDG_STAT43 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[11] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[11] is 1 when POR_WDG overflows. |
| 10<br>POR_WDG_STAT42 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[10] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[10] is 1 when POR_WDG overflows. |
| 9<br>POR_WDG_STAT41 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[9] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[9] is 1 when POR_WDG overflows. |
| 8<br>POR_WDG_STAT40 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[8] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[8] is 1 when POR_WDG overflows. |
| 7<br>POR_WDG_STAT39 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[7] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[7] is 1 when POR_WDG overflows. |
| 6<br>POR_WDG_STAT38 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[6] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[6] is 1 when POR_WDG overflows. |
| 5<br>POR_WDG_STAT37 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[5] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[5] is 1 when POR_WDG overflows. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 4<br>POR_WDG_STAT36 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[4] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[4] is 1 when POR_WDG overflows. |
| 3<br>POR_WDG_STAT35 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[3] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[3] is 1 when POR_WDG overflows. |
| 2<br>POR_WDG_STAT34 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[2] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[2] is 1 when POR_WDG overflows. |
| 1<br>POR_WDG_STAT33 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[1] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[1] is 1 when POR_WDG overflows. |
| 0<br>POR_WDG_STAT32 | This bit captures MC_RGM functional/external event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_FES[0] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_FES[0] is 1 when POR_WDG overflows. |

### 37.2.42 Read Only GPR On PMCPOR Reset (DCMROPP3)

**Offset**

| Register | Offset |
|---|---|
| DCMROPP3 | 708h |

**Function**
Resets on PMCPOR reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... | POR_WDG... |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>POR_WDG_STAT95 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[31] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[31] is 1 when POR_WDG overflows. |
| 30<br>POR_WDG_STAT94 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[30] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[30] is 1 when POR_WDG overflows. |
| 29<br>POR_WDG_STAT93 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[29] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[29] is 1 when POR_WDG overflows. |
| 28<br>POR_WDG_STAT92 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[28] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[28] is 1 when POR_WDG overflows. |
| 27<br>POR_WDG_STAT91 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[27] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[27] is 1 when POR_WDG overflows. |
| 26<br>POR_WDG_STAT90 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[26] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[26] is 1 when POR_WDG overflows. |
| 25 | This bit captures MC_RGM destructive event status register when POR_WDG overflows. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| POR_WDG_STAT89 | 0b - MC_RGM.RGM_DES[25] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[25] is 1 when POR_WDG overflows. |
| 24<br>POR_WDG_STAT88 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[24] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[24] is 1 when POR_WDG overflows. |
| 23<br>POR_WDG_STAT87 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[23] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[23] is 1 when POR_WDG overflows. |
| 22<br>POR_WDG_STAT86 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[22] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[22] is 1 when POR_WDG overflows. |
| 21<br>POR_WDG_STAT85 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[21] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[21] is 1 when POR_WDG overflows. |
| 20<br>POR_WDG_STAT84 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[20] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[20] is 1 when POR_WDG overflows. |
| 19<br>POR_WDG_STAT83 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[19] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[19] is 1 when POR_WDG overflows. |
| 18<br>POR_WDG_STAT82 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[18] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[18] is 1 when POR_WDG overflows. |
| 17<br>POR_WDG_STAT81 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[17] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[17] is 1 when POR_WDG overflows. |
| 16<br>POR_WDG_STAT80 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[16] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[16] is 1 when POR_WDG overflows. |
| 15<br>POR_WDG_STAT79 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[15] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[15] is 1 when POR_WDG overflows. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 14<br>POR_WDG_STAT78 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[14] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[14] is 1 when POR_WDG overflows. |
| 13<br>POR_WDG_STAT77 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[13] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[13] is 1 when POR_WDG overflows. |
| 12<br>POR_WDG_STAT76 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[12] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[12] is 1 when POR_WDG overflows. |
| 11<br>POR_WDG_STAT75 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[11] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[11] is 1 when POR_WDG overflows. |
| 10<br>POR_WDG_STAT74 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[10] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[10] is 1 when POR_WDG overflows. |
| 9<br>POR_WDG_STAT73 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[9] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[9] is 1 when POR_WDG overflows. |
| 8<br>POR_WDG_STAT72 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[8] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[8] is 1 when POR_WDG overflows. |
| 7<br>POR_WDG_STAT71 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[7] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[7] is 1 when POR_WDG overflows. |
| 6<br>POR_WDG_STAT70 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[6] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[6] is 1 when POR_WDG overflows. |
| 5<br>POR_WDG_STAT69 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[5] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[5] is 1 when POR_WDG overflows. |
| 4 | This bit captures MC_RGM destructive event status register when POR_WDG overflows. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| POR_WDG_STAT68 | 0b - MC_RGM.RGM_DES[4] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[4] is 1 when POR_WDG overflows. |
| 3<br>POR_WDG_STAT67 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[3] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[3] is 1 when POR_WDG overflows. |
| 2<br>POR_WDG_STAT66 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[2] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[2] is 1 when POR_WDG overflows. |
| 1<br>POR_WDG_STAT65 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[1] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[1] is 1 when POR_WDG overflows. |
| 0<br>POR_WDG_STAT64 | This bit captures MC_RGM destructive event status register when POR_WDG overflows.<br>0b - MC_RGM.RGM_DES[0] is 0 when POR_WDG overflows.<br>1b - MC_RGM.RGM_DES[0] is 1 when POR_WDG overflows. |

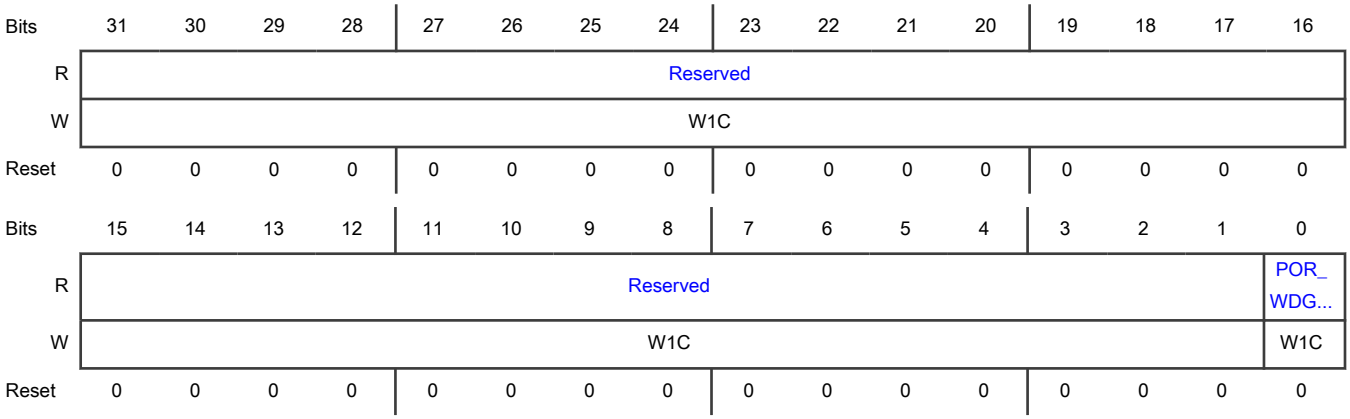
## 37.2.43 Read Only GPR On PMCPOR Reset (DCMROPP4)

### Offset

| Register | Offset |
|---|---|
| DCMROPP4 | 70Ch |

### Function
Resets on PMCPOR reset.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>POR_WDG_STAT96 | This bit captures POR_WDG reset event if POR_WDG initiates a POR sequence.<br>0b - POR_WDG is inactive.<br>1b - POR_WDG has detected a stuck scenario and raised device POR event. Refer POR_WDG_STAT[95:0] for device status when POR_WDG overflows |

# Chapter 38
# Device Configuration Module (DCM)

## 38.1 Overview

DCM controls:

- LC
- DCF client registers
- Debug authorization (Export Control mode)

The module also establishes a RoT for the chip by parsing the master root key and other security records.

### 38.1.1 Block diagram

This figure is a block representation of DCM.

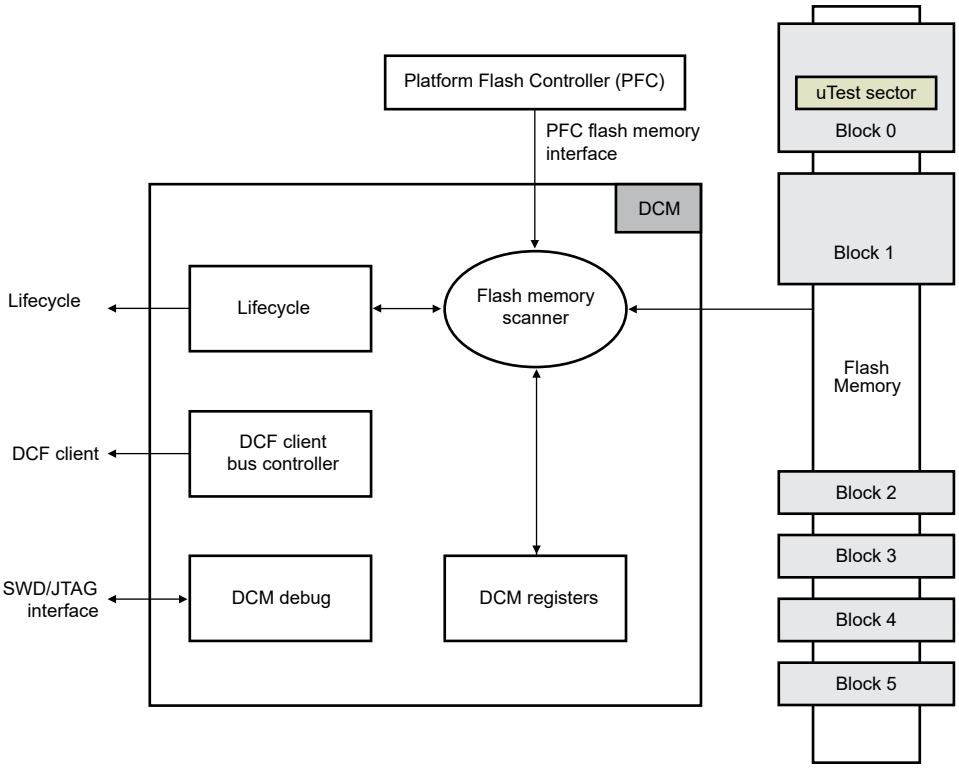



Figure 139. DCM block diagram

### 38.1.2 Features

- Scans flash memory and configures the system for following information:
  - LC detection
  - DCF records configuration using flash memory
- Allows debug features for flash memory content and the chip
- Provides debug enable control for HSE and application

- Provides a valid boot address detection
- Enables the DCF client to be writable via the IPS bus
- Parametrizes control to mask a set of DCF client chip select
- Supports temporary advancement of the LC by programming DCMLCC[DCMFLC]
- Manages debug features for flash memory content and the chip like debug enable control and debug authorization.

## 38.2 Functional description

DCM provides information about the current state and configuration of the system that you could use to:

- Configure the application software.
- Debug the system.

### 38.2.1 Modes of operation

DCM operates identically in all system modes of operation.

### 38.2.2 DCF mechanism

The DCF mechanism handles chip parameter settings via the OTP flash memory.

You can store a series of DCF records in UTEST flash memory, and each record is 64 bits in length. The chip processes these records during the system reset sequence before the CPU leaves reset.

### 38.2.3 DCF error recording

DCF errors are recorded in DCM for both types of clients—spread spectrum safe and normal. See DCF client error mechanism for details on spread spectrum safe clients.

### 38.2.4 DCF client error mechanism

The DCM consists of DCMMISC[DCMCERS] for detecting faulty DCF records. The DCMSRRn registers capture the details up to 16 faulty records. The DCMCERS bit can be cleared by writing 1 to it.

While scanning the flash, in case if a faulty record is encountered, the DCMMISC[DCMCERS] gets set indicating that there is atleast one faulty record. In such a case, the user can identify the details about the faulty record in the DCMSRRn registers and should update the flash memory with a new record, provided the record is not write-once.

For example, in the image below, if the faulty DCF record is DCF2, then the correct DCF record must also be DCF2. Even in case if the faulty record is updated, the DCM stores the faulty record via the DCMMISC[DCMCERS] and DCMSRRn registers.

This figure shows that a faulty DCF record is loaded between the original and final DCF record sets.

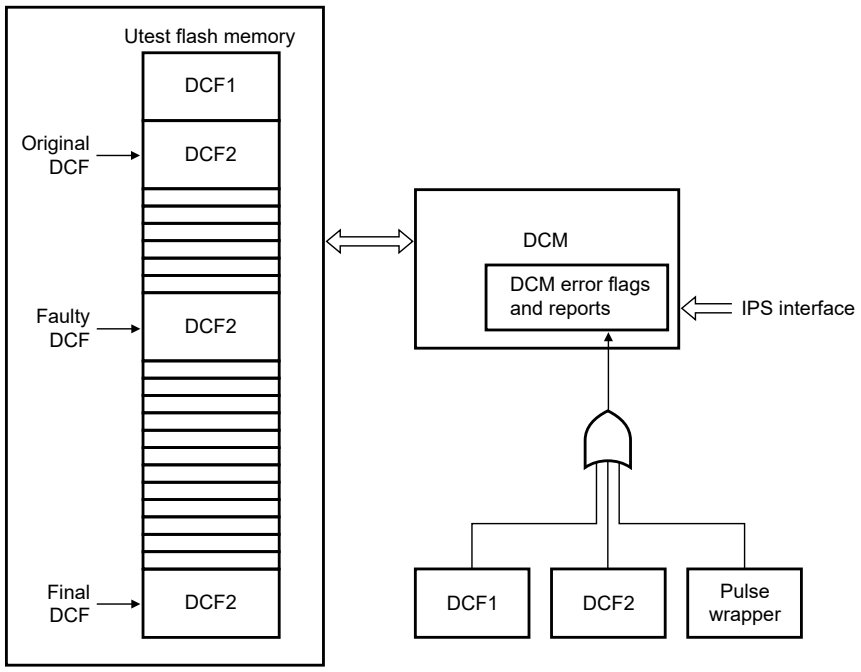



Figure 140. DCF error handling

### 38.2.5 DCM error detection mechanism

This figure shows the flow diagram of errors logged into DCM from DCF clients and Flash memory scanner. It also shows that DCM provides software control for lifecycle and export control mode.

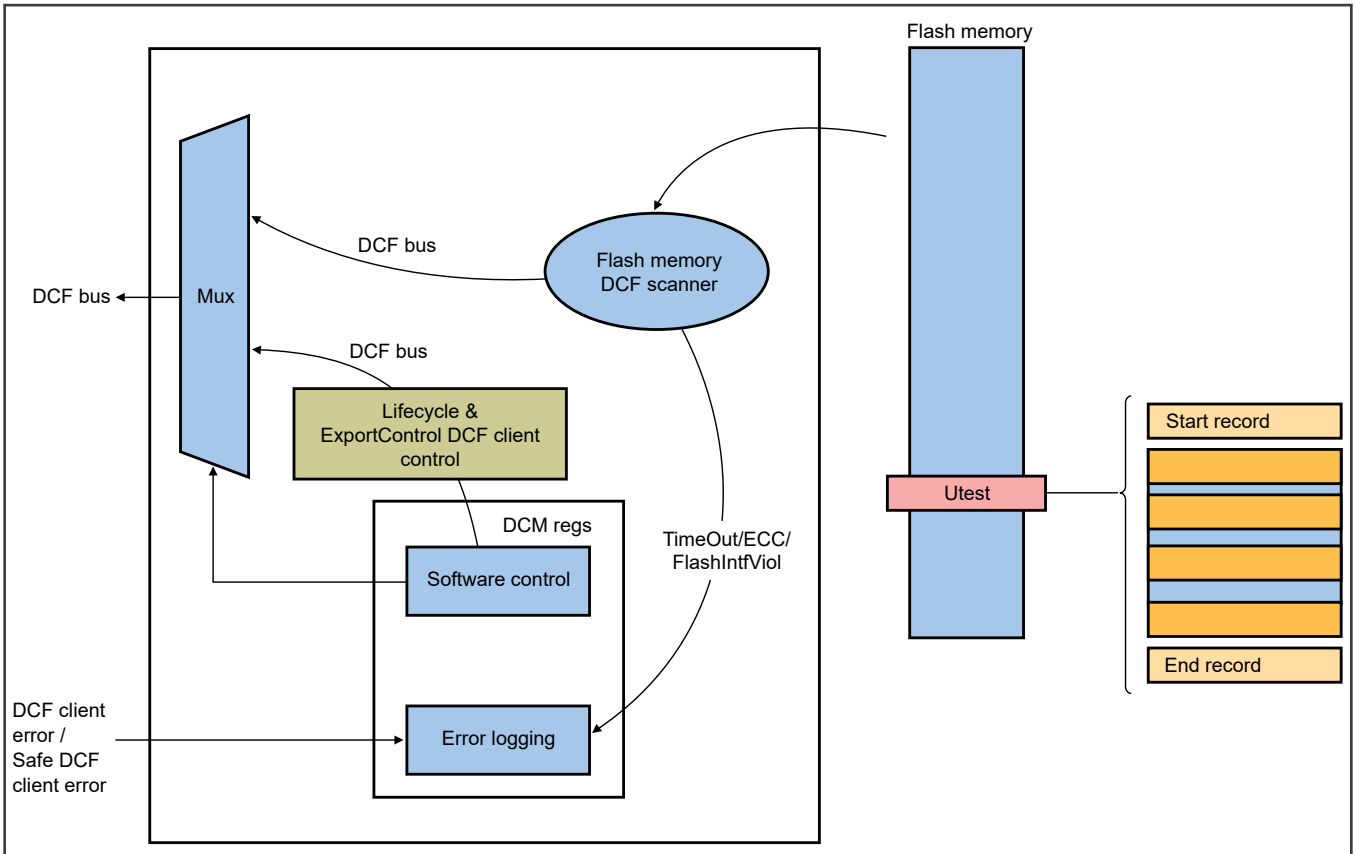



Figure 141. DCM error detection mechanism

## 38.2.6 LC

DCM determines the LC of the chip by reading the LC slots from the Utest flash memory. This read operation is performed during the reset phase, with normal timings. The operating monitors and an ECC check protect the operation. Additionally, a set of sanity checks executed over the LC read data guarantee the integrity of the final LC value.

At the end of the reset phase, the LC contains one of the following values:

- OEM production (OEM_PROD)
- In field (IN_FIELD)
- Pre-FA
- FA

The DCM LC progresses in the direction shown in this figure:

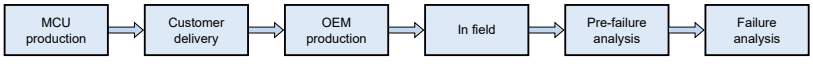



Figure 142. LC sequence

The LC is written into six slots, 128 bits each, and at fixed locations in the Utest flash memory block. Each LC slot is read in a single atomic operation and is organized in two types of fields:

- Valid (lower 64b of LC slot)
- Invalid (higher 64b of LC slot)

Depending on the possible combinations of data programmed into these fields, each LC slot indicates one of the four possible statuses as shown in Table 195. To know more about LC slots, see Table 196.

**Table 195. LC slot status**

| LC slots | | LC slot value |
|---|---|---|
| Valid field (64 bits) | Invalid field (64 bits) | |
| Erased | Erased | Erased |
| Marked | Erased | Active |
| Marked | Marked | Inactive |
| Other values | | Illegal |

In this case, "Marked" refers to a value that is configured based on the bit pattern 55AA_50AF_55AA_50AFh, and "Erased" is detected using the bit pattern FFFF_FFFF_FFFF_FFFFh.

**Table 196. LC slots**

| LC slot 2<br>OEM_PROD | LC slot 3<br>IN_FIELD | LC slot 4<br>Pre-FA | LC slot 5<br>FA | Resulting LC |
|---|---|---|---|---|
| Active | Erased | Erased | Erased | OEM_PROD |
| Inactive | Active | Erased | Erased | IN_FIELD |
| Inactive | Inactive | Active | Erased | Pre-FA |
| Inactive | Inactive | Inactive | Active | FA |

**NOTE**

When triggering DCM for rescanning, the software must ensure that the flash memory program and erase fields are not set. Otherwise, DCM does not configure the chip and ignores all data returned from the UTEST flash memory. This results in LC becoming IN_FIELD. All DCF clients indicate their default values in this case.

## 38.3 DCM register descriptions

### 38.3.1 DCM memory map

DCM base address: 402A_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | DCM Status (DCMSTAT) | 32 | R | 0000_0371h |
| 4h | LC and LC Control (DCMLCC) | 32 | R | See section |
| 8h | LC Scan Status (DCMLCS) | 32 | RW | 0000_0000h |
| 1Ch | DCM Miscellaneous (DCMMISC) | 32 | RW | 0000_0001h |
| 20h | Debug Status and Configuration (DCMDEB) | 32 | R | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 2Ch | DCF Error Count (DCMEC) | 32 | R | 0000_0000h |
| 30h | DCF Scan Report (DCMSRR1) | 32 | RW | 0000_0000h |
| 34h | DCF Scan Report (DCMSRR2) | 32 | RW | 0000_0000h |
| 38h | DCF Scan Report (DCMSRR3) | 32 | RW | 0000_0000h |
| 3Ch | DCF Scan Report (DCMSRR4) | 32 | RW | 0000_0000h |
| 40h | DCF Scan Report (DCMSRR5) | 32 | RW | 0000_0000h |
| 44h | DCF Scan Report (DCMSRR6) | 32 | RW | 0000_0000h |
| 48h | DCF Scan Report (DCMSRR7) | 32 | RW | 0000_0000h |
| 4Ch | DCF Scan Report (DCMSRR8) | 32 | RW | 0000_0000h |
| 50h | DCF Scan Report (DCMSRR9) | 32 | RW | 0000_0000h |
| 54h | DCF Scan Report (DCMSRR10) | 32 | RW | 0000_0000h |
| 58h | DCF Scan Report (DCMSRR11) | 32 | RW | 0000_0000h |
| 5Ch | DCF Scan Report (DCMSRR12) | 32 | RW | 0000_0000h |
| 60h | DCF Scan Report (DCMSRR13) | 32 | RW | 0000_0000h |
| 64h | DCF Scan Report (DCMSRR14) | 32 | RW | 0000_0000h |
| 68h | DCF Scan Report (DCMSRR15) | 32 | RW | 0000_0000h |
| 6Ch | DCF Scan Report (DCMSRR16) | 32 | RW | 0000_0000h |
| 80h | LC Scan Status 2 (DCMLCS_2) | 32 | RW | 0000_0000h |

## 38.3.2 DCM Status (DCMSTAT)

### Offset

| Register | Offset |
|---|---|
| DCMSTAT | 0h |

### Function

Indicates the status of DCM at different stages.

**NOTE**

This register resets on functional reset.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | 0 | DCMDBGPS | 0 | DCMUTS | 0 | | | DCMLCST | 0 | | DCMERR | DCMDONE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |

### Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18-16<br>— | Reserved |
| 15-12<br>— | Reserved |
| 11<br>— | Reserved |
| 10<br>DCMDBGPS | Debug Password Scanning Status<br>Indicates the DCM debug password scanning status.<br>0b - Completed with errors<br>1b - Completed successfully |
| 9<br>— | Reserved |
| 8<br>DCMUTS | DCM Utest DCF Scanning Status (valid only if DCMDONE bit is set)<br>**NOTE**<br>This bit always returns 0 in In Field LC.<br>0b - Completed with errors.<br>1b - Completed successfully. |
| 7-5 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 4<br>DCMLCST | LC Scanning Status (valid only if DCMDONE bit is set)<br>Indicates the DCM LC scanning status.<br>**NOTE**<br>This bit always returns 0 in INFIELD lifecycle.<br>0b - Completed with errors<br>1b - Completed successfully |
| 3-2<br>— | Reserved |
| 1<br>DCMERR | DCM Completion of Flash Scan with Error Status (valid only if DCMDONE bit is set)<br>0b - Completed with success.<br>1b - Completed with error. |
| 0<br>DCMDONE | DCM Flash Scanning Status<br>Indicates whether the DCM scanning is in progress or complete.<br>0b - Running<br>1b - Completed |

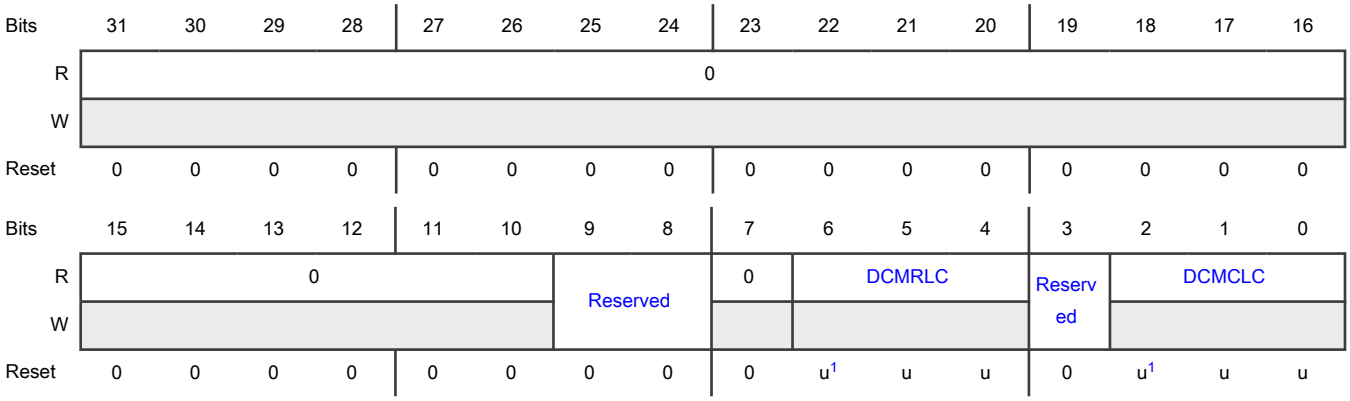
### 38.3.3 LC and LC Control (DCMLCC)

**Offset**

| Register | Offset |
|---|---|
| DCMLCC | 4h |

**Function**
Resets on the functional reset.

### Diagram




1. Post reset, the reset value of this register is 0000_0077h, and after scanning, it changes according to the programmed value.

### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9-8<br>— | Reserved |
| 7<br>— | Reserved |
| 6-4<br>DCMRLC | Real LC<br>Projects the real LC of the chip.<br>The LC can move in this sequence: (010 : OEM_PROD) > (111 : IN_FIELD) > (001 : Pre-FA) > (000 : FA)<br>000b - FA<br>001b - Pre-FA<br>010b - OEM_PROD<br>100b - Reserved<br>101b - Reserved<br>111b - IN_FIELD |
| 3<br>— | Reserved |
| 2-0<br>DCMCLC | Current LC<br>Projects the current LC of the chip.<br>The LC can move in this sequence: (010 : OEM_PROD) > (111 : IN_FIELD) > (001 : Pre-FA) > (000 : FA) |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 000b - FA<br>001b - Pre-FA<br>010b - OEM_PROD<br>100b - Reserved<br>101b - Reserved<br>111b - IN_FIELD |

## 38.3.4 LC Scan Status (DCMLCS)

### Offset

| Register | Offset |
|---|---|
| DCMLCS | 8h |

### Function

Stores the status of LC scanning. By default, the status of each LC is "not yet scanned."

This register:

- Resets on destructive reset.
- Always returns 0 in a valid IN_FIELD LC (in LC without an error).

This register captures the errors related to LC scanning on each of these resets: POR, destructive, and functional. If an error is captured, its status in this register is cleared by writing 1 to the corresponding field or to any of the destructive or POR events.

All LC slot errors are captured and cleared independently.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | DCMLCFE5 | DCMLCE5 | DCMLCC5 | | | DCMLCSS5 | DCMLCFE4 | DCMLCE4 | DCMLCC4 | | | DCMLCSS4 | DCMLCFE3 | DCMLCE3 |
| W | | | W1C | W1C | W1C | | | W1C | W1C | W1C | W1C | | | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCMLCC3 | | | DCMLCSS3 | Reserved | | | | | | | | | | | |
| W | W1C | | | W1C | W1C | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>DCMLCFE5 | Pre-FA Flash Memory Error Check<br>Indicates the status of the Pre-FA flash memory error check.<br>0b - No errors<br>1b - Errors exist |
| 28<br>DCMLCE5 | Pre-FA ECC Errors<br>Indicates if Pre-FA is successful or has ECC errors.<br>0b - No errors<br>1b - Errors exist |
| 27-25<br>DCMLCC5 | Pre-FA Marking Status<br>Indicates the Pre-FA marking status.<br>These errors may cause this field to indicate the "not scanned yet" status:<br>• If the reading completes too early and DCM has not yet scanned the LC.<br>• If there is an error in the flash memory after completion of the reading.<br>000b - Not scanned yet<br>001b - Marked as active<br>010b - Marked as inactive<br>011b - Region is erased/virgin<br>101b - Marked as inactive by an unknown pattern<br>110b - Scanning timed out |
| 24<br>DCMLCSS5 | Pre-FA Scan Status<br>Indicates the status of the Pre-FA scan.<br>0b - No errors<br>1b - Errors exist |
| 23<br>DCMLCFE4 | IN_FIELD Flash Memory Error Check<br>Indicates the status of IN_FIELD flash memory error check.<br>0b - No errors<br>1b - Errors exist |
| 22<br>DCMLCE4 | IN_FIELD ECC Errors<br>Indicates if IN_FIELD has ECC errors. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No errors<br>1b - Errors exist |
| 21-19<br>DCMLCC4 | IN_FIELD Marking Status<br>Indicates the IN_FIELD marking status.<br>These errors may cause this field to indicate the "not scanned yet" status:<br>• If the reading completes too early and DCM has not yet scanned the LC.<br>• If there is an error in the flash memory after completion of the reading.<br>000b - Not scanned yet<br>001b - Marked as active<br>010b - Marked as inactive<br>011b - Region is erased/virgin<br>101b - Marked as inactive by an unknown pattern<br>110b - Scanning timed out |
| 18<br>DCMLCSS4 | IN_FIELD Scan Status<br>Indicates the status of the IN_FIELD scan.<br>0b - No errors<br>1b - Errors exist |
| 17<br>DCMLCFE3 | OEM_PROD Flash Memory Error Check<br>Indicates the status of the OEM_PROD flash memory error check.<br>0b - No errors<br>1b - Errors exist |
| 16<br>DCMLCE3 | OEM_PROD ECC Errors<br>Indicates if OEM_PROD has ECC errors.<br>0b - No errors<br>1b - Errors exist |
| 15-13<br>DCMLCC3 | OEM_PROD Marking<br>Indicates the OEM_PROD marking status.<br>These errors may cause this field to indicate the "not scanned yet" status:<br>• If the reading completes too early and DCM has not yet scanned the LC.<br>• If there is an error in the flash memory after completion of the reading.<br>000b - Not scanned yet |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 001b - Marked as active<br>010b - Marked as inactive<br>011b - Region is erased/virgin<br>101b - Marked as inactive by an unknown pattern<br>110b - Scanning timed out |
| 12<br>DCMLCSS3 | OEM_PROD Scan Status<br>Indicates the status of OEM_PROD scan.<br>0b - No errors<br>1b - Errors exist |
| 11-0<br>— | Reserved |

## 38.3.5 DCM Miscellaneous (DCMMISC)

### Offset

| Register | Offset |
|---|---|
| DCMMISC | 1Ch |

### Function

Resets on destructive reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | DCMCERS | 0 | | | | 0 | | | | | | | |
| W | | | | W1C | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | 0 | | DCMDBGE | DCMDBGT | 0 | | | | | | | | | |
| W | | | | | W1C | W1C | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>— | Reserved |
| 29<br>— | Reserved |
| 28<br>DCMCERS | DCF Client Errors<br>Records the status of errors from DCF clients.<br>0b - No errors on any of the DCF clients<br>1b - Atleast one safety DCF client has an error |
| 27-24<br>— | Reserved |
| 23-14<br>— | Reserved |
| 13-12<br>— | Reserved |
| 11<br>DCMDBGE | DCM ECC error on DBG sections<br>This bit is set if there is any ECC error during scanning of CUST_PWD, or UID.<br>0b - No ECC error<br>1b - ECC error |
| 10<br>DCMDBGT | DBG Section Timeout Error<br>Indicates if there is a DCM UTEST flash memory timeout error in DBG sections. The value of this field is 1 in case a timeout error occurs when scanning CUST_PWD, or UID.<br>0b - No error<br>1b - Error exists |
| 9-0<br>— | Reserved |

## 38.3.6 Debug Status and Configuration (DCMDEB)

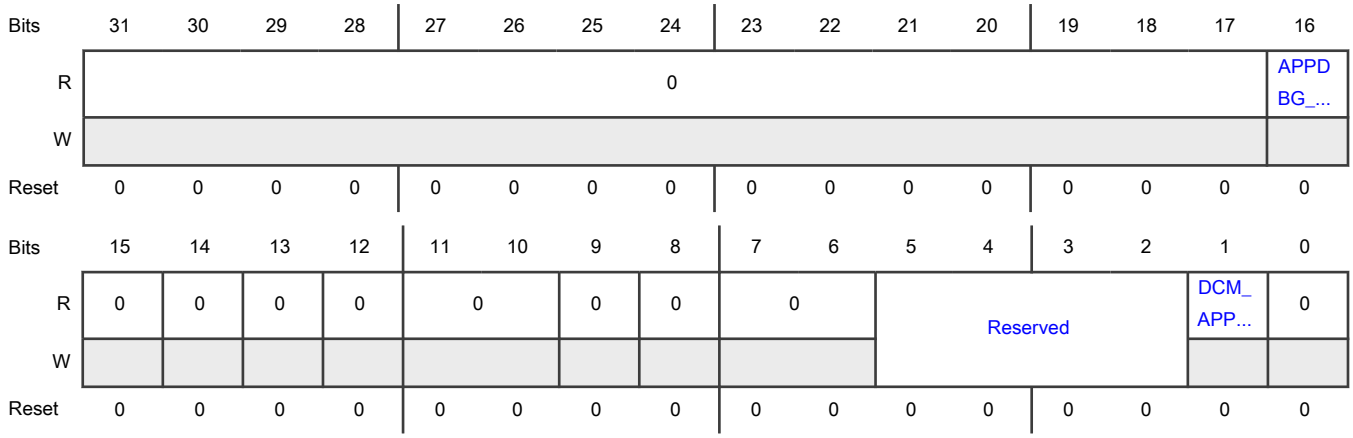
### Offset

| Register | Offset |
|---|---|
| DCMDEB | 20h |

### Function

Resets on destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-17<br>— | Reserved |
| 16<br>APPDBG_STAT_SOC | Application Debug Status<br>Indicates the application debug status of the chip.<br>0b - Disabled<br>1b - Enabled |
| 15<br>— | Reserved |
| 14<br>— | Reserved |
| 13<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 12<br>— | Reserved |
| 11-10<br>— | Reserved |
| 9<br>— | Reserved |
| 8<br>— | Reserved |
| 7-6<br>— | Reserved |
| 5-2<br>— | Reserved |
| 1<br>DCM_APPDBG_STAT | DCM Authentication Engine Status for Application Core<br>**NOTE**<br>This bit will read 0 in non-export control mode. |
| 0<br>— | Reserved |

## 38.3.7 DCF Error Count (DCMEC)

### Offset

| Register | Offset |
|---|---|
| DCMEC | 2Ch |

### Function

Indicates the number of faulty DCF records.

This register resets on destructive reset.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>DCMECT | DCF Error Count<br>Indicates the number of faulty DCF records. |

## 38.3.8 DCF Scan Report (DCMSRR1)

### Offset

| Register | Offset |
|---|---|
| DCMSRR1 | 30h |

### Function

Resets on destructive reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | DCMDCFT1 | DCMESD1 | DCMESF1 | DCMDCFF1 | | | 0 | | | | DCMDCFE1 | | | |
| W | | | W1C | W1C | W1C | W1C | | | | | | | W1C | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCMDCFE1 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Fields**

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>DCMDCFT1 | Scanning Timeout On Flash Memory<br>Indicates if scanning timeout exists on flash memory address.<br>0b - Does not exist<br>1b - Exists |
| 28<br>DCMESD1 | Chip Side Error<br>Indicates if an error exists on chip side. These errors could be parity errors or the ones reported by the DCF client, such as write-once error.<br>0b - No errors<br>1b - Errors exist |
| 27<br>DCMESF1 | Flash Memory Error<br>Indicates if a UTEST flash memory ECC error exists.<br>0b - No errors<br>1b - Errors exist |
| 26-24<br>DCMDCFF1 | DCF Record Location<br>Indicates the DCF record location.<br>010b - Utest flash memory<br>101b - Others: Reserved |
| 23-21<br>— | Reserved |
| 20-0<br>DCMDCFE1 | Flash Memory Address<br>Indicates the flash memory address of the DCF client having an error. |

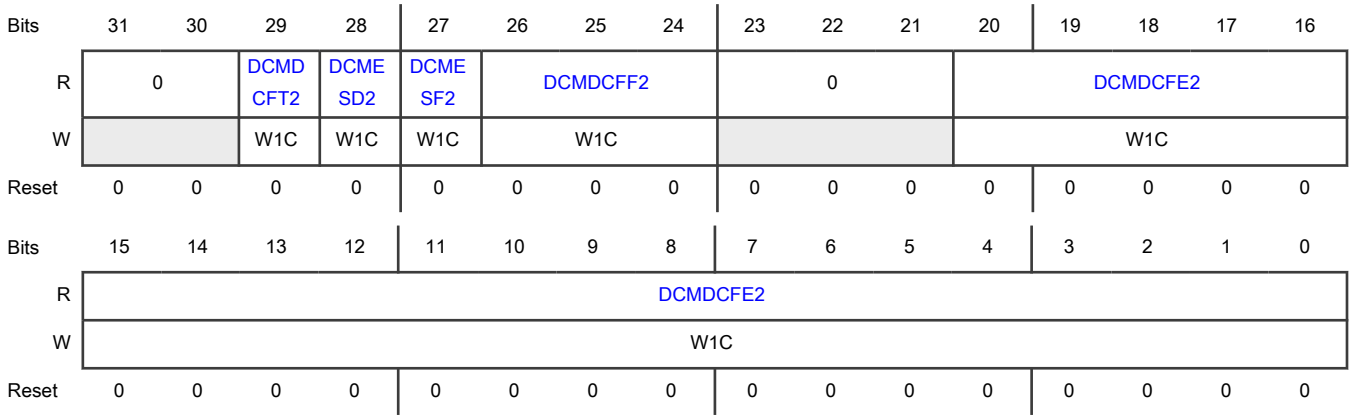
### 38.3.9 DCF Scan Report (DCMSRR2)

**Offset**

| Register | Offset |
|---|---|
| DCMSRR2 | 34h |

**Function**
Resets on destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>DCMDCFT2 | Scanning Timeout On Flash Memory<br>Indicates if scanning timeout exists on flash memory address.<br>0b - Does not exist<br>1b - Exists |
| 28<br>DCMESD2 | Chip Side Error<br>Indicates if an error exists on chip side. These errors could be parity errors or the ones reported by the DCF client, such as write-once error.<br>0b - No errors<br>1b - Errors exist |
| 27<br>DCMESF2 | Flash Memory Error<br>Indicates if a UTEST flash memory ECC error exists.<br>0b - No errors<br>1b - Errors exist |
| 26-24<br>DCMDCFF2 | DCF Record Location<br>Indicates the DCF record location.<br>010b - Utest flash memory<br>101b - Others: Reserved |
| 23-21<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 20-0<br>DCMDCFE2 | Flash Memory Address<br>Indicates the flash memory address of the DCF client having an error. |

## 38.3.10 DCF Scan Report (DCMSRR3)

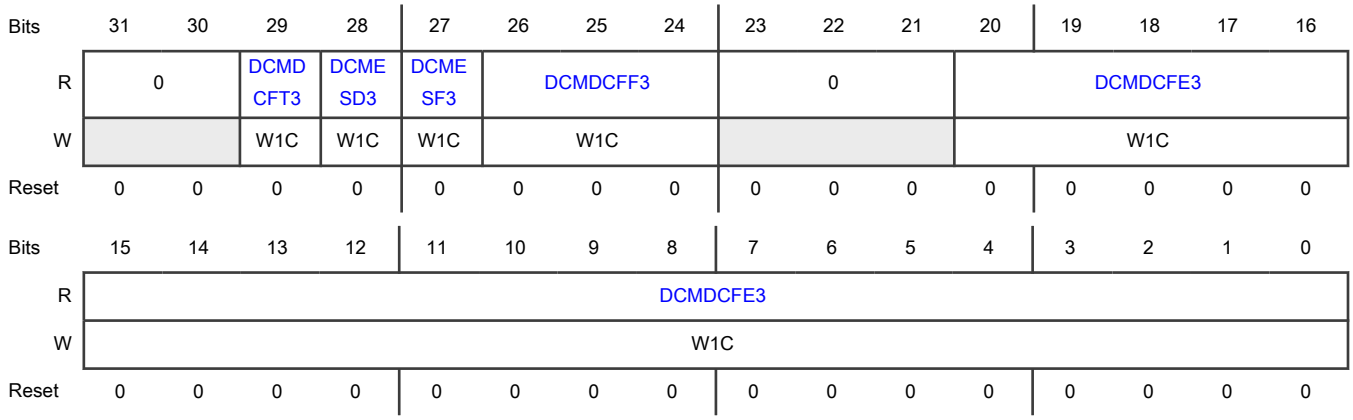
### Offset

| Register | Offset |
|---|---|
| DCMSRR3 | 38h |

### Function

Resets on destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>DCMDCFT3 | Scanning Timeout On Flash Memory<br>Indicates if scanning timeout exists on flash memory address.<br>0b - Does not exist<br>1b - Exists |
| 28<br>DCMESD3 | Chip Side Error |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Indicates if an error exists on chip side. These errors could be parity errors or the ones reported by the DCF client, such as write-once error.<br>0b - No errors<br>1b - Errors exist |
| 27<br>DCMESF3 | Flash Memory Error<br>Indicates if a UTEST flash memory ECC error exists.<br>0b - No errors<br>1b - Errors exist |
| 26-24<br>DCMDCFF3 | DCF Record Location<br>Indicates the DCF record location.<br>010b - Utest flash memory<br>101b - Others: Reserved |
| 23-21<br>— | Reserved |
| 20-0<br>DCMDCFE3 | Flash Memory Address<br>Indicates the flash memory address of the DCF client having an error. |

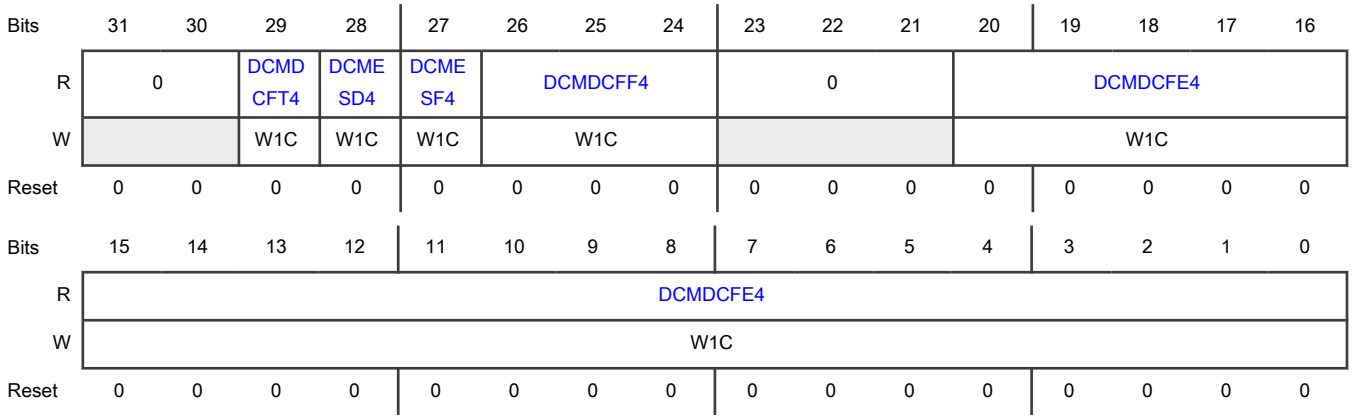
### 38.3.11 DCF Scan Report (DCMSRR4)

**Offset**

| Register | Offset |
|---|---|
| DCMSRR4 | 3Ch |

**Function**
Resets on destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>DCMDCFT4 | Scanning Timeout On Flash Memory<br>Indicates if scanning timeout exists on flash memory address.<br>0b - Does not exist<br>1b - Exists |
| 28<br>DCMESD4 | Chip Side Error<br>Indicates if an error exists on chip side. These errors could be parity errors or the ones reported by the DCF client, such as write-once error.<br>0b - No errors<br>1b - Errors exist |
| 27<br>DCMESF4 | Flash Memory Error<br>Indicates if a UTEST flash memory ECC error exists.<br>0b - No errors<br>1b - Errors exist |
| 26-24<br>DCMDCFF4 | DCF Record Location<br>Indicates the DCF record location.<br>010b - Utest flash memory<br>101b - Others: Reserved |
| 23-21<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 20-0<br>DCMDCFE4 | Flash Memory Address<br>Indicates the flash memory address of the DCF client having an error. |

## 38.3.12 DCF Scan Report (DCMSRR5)

### Offset

| Register | Offset |
|---|---|
| DCMSRR5 | 40h |

### Function

Resets on destructive reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | DCMDCFT5 | DCMESD5 | DCMESF5 | DCMDCFF5 | | | 0 | | | | DCMDCFE5 | | | |
| W | | | W1C | W1C | W1C | W1C | | | | | | | W1C | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCMDCFE5 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>DCMDCFT5 | Scanning Timeout On Flash Memory<br>Indicates if scanning timeout exists on flash memory address.<br>0b - Does not exist<br>1b - Exists |
| 28<br>DCMESD5 | Chip Side Error |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Indicates if an error exists on chip side. These errors could be parity errors or the ones reported by the DCF client, such as write-once error.<br>0b - No errors<br>1b - Errors exist |
| 27<br>DCMESF5 | Flash Memory Error<br>Indicates if a UTEST flash memory ECC error exists.<br>0b - No errors<br>1b - Errors exist |
| 26-24<br>DCMDCFF5 | DCF Record Location<br>Indicates the DCF record location.<br>010b - Utest flash memory<br>101b - Others: Reserved |
| 23-21<br>— | Reserved |
| 20-0<br>DCMDCFE5 | Flash Memory Address<br>Indicates the flash memory address of the DCF client having an error. |

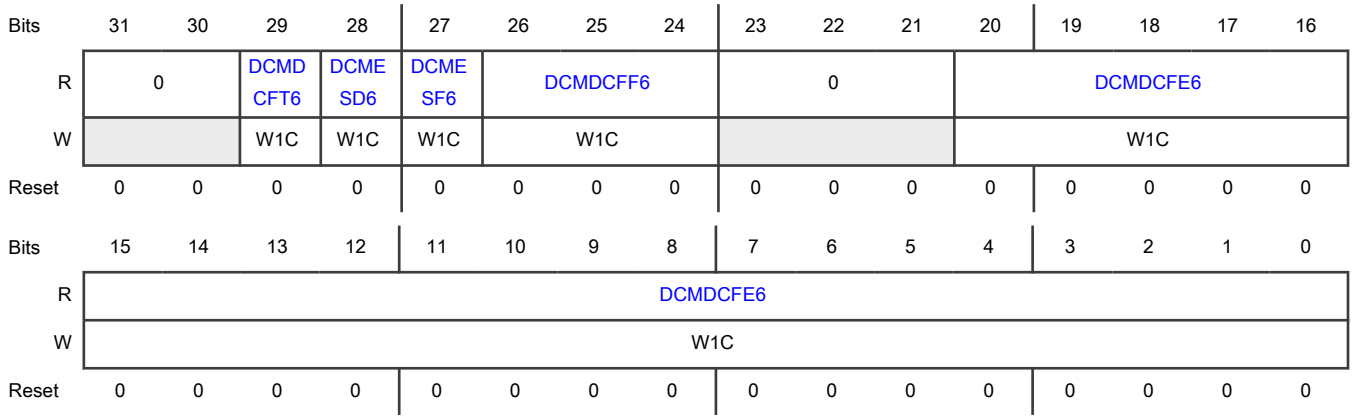
### 38.3.13 DCF Scan Report (DCMSRR6)

**Offset**

| Register | Offset |
|---|---|
| DCMSRR6 | 44h |

**Function**
Resets on destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>DCMDCFT6 | Scanning Timeout On Flash Memory<br>Indicates if scanning timeout exists on flash memory address.<br>0b - Does not exist<br>1b - Exists |
| 28<br>DCMESD6 | Chip Side Error<br>Indicates if an error exists on chip side. These errors could be parity errors or the ones reported by the DCF client, such as write-once error.<br>0b - No errors<br>1b - Errors exist |
| 27<br>DCMESF6 | Flash Memory Error<br>Indicates if a UTEST flash memory ECC error exists.<br>0b - No errors<br>1b - Errors exist |
| 26-24<br>DCMDCFF6 | DCF Record Location<br>Indicates the DCF record location.<br>010b - Utest flash memory<br>101b - Others: Reserved |
| 23-21<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 20-0<br>DCMDCFE6 | Flash Memory Address<br>Indicates the flash memory address of the DCF client having an error. |

## 38.3.14 DCF Scan Report (DCMSRR7)

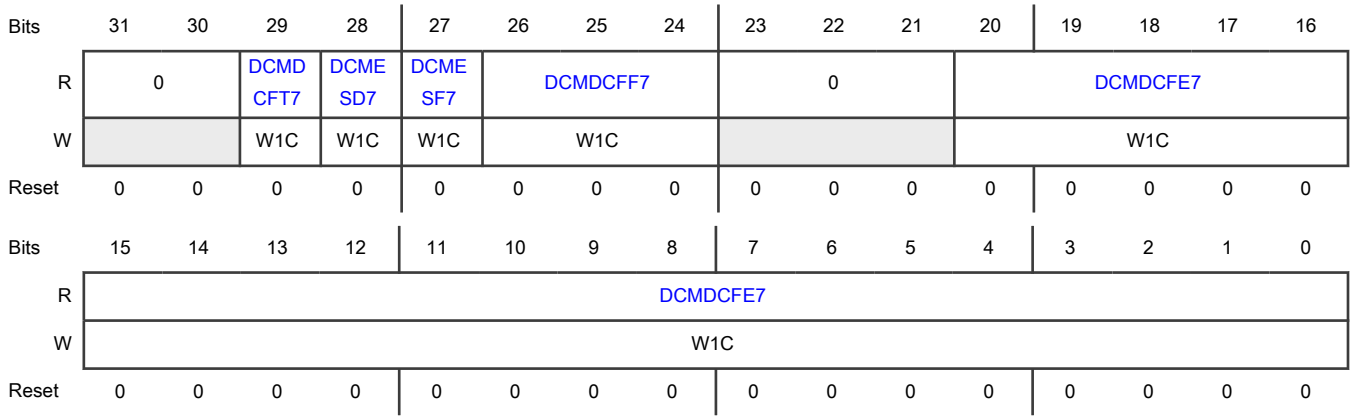
### Offset

| Register | Offset |
|---|---|
| DCMSRR7 | 48h |

### Function

Resets on destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>DCMDCFT7 | Scanning Timeout On Flash Memory<br>Indicates if scanning timeout exists on flash memory address.<br>0b - Does not exist<br>1b - Exists |
| 28<br>DCMESD7 | Chip Side Error |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Indicates if an error exists on chip side. These errors could be parity errors or the ones reported by the DCF client, such as write-once error.<br>0b - No errors<br>1b - Errors exist |
| 27<br>DCMESF7 | Flash Memory Error<br>Indicates if a UTEST flash memory ECC error exists.<br>0b - No errors<br>1b - Errors exist |
| 26-24<br>DCMDCFF7 | DCF Record Location<br>Indicates the DCF record location.<br>010b - Utest flash memory<br>101b - Others: Reserved |
| 23-21<br>— | Reserved |
| 20-0<br>DCMDCFE7 | Flash Memory Address<br>Indicates the flash memory address of the DCF client having an error. |

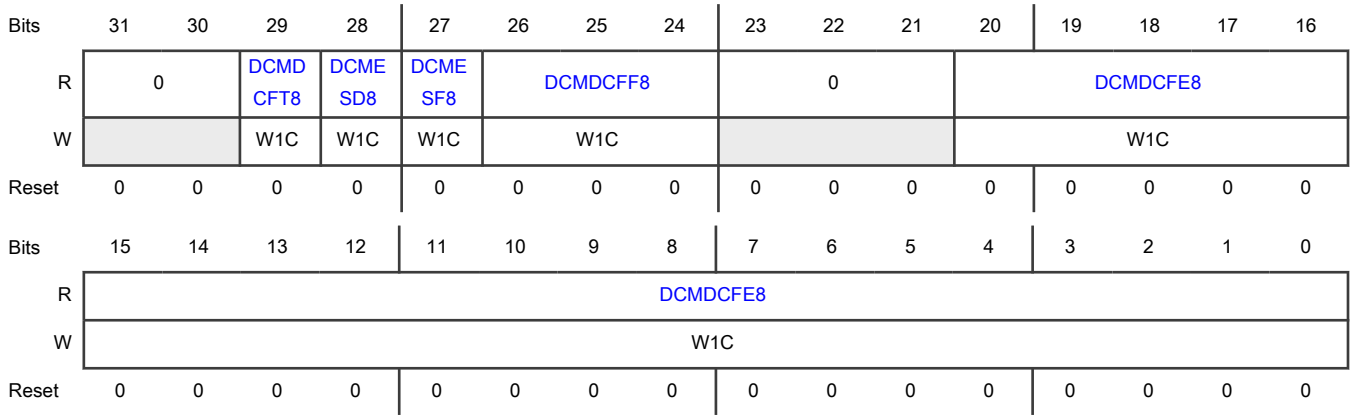
### 38.3.15 DCF Scan Report (DCMSRR8)

**Offset**

| Register | Offset |
|---|---|
| DCMSRR8 | 4Ch |

**Function**
Resets on destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>DCMDCFT8 | Scanning Timeout On Flash Memory<br>Indicates if scanning timeout exists on flash memory address.<br>0b - Does not exist<br>1b - Exists |
| 28<br>DCMESD8 | Chip Side Error<br>Indicates if an error exists on chip side. These errors could be parity errors or the ones reported by the DCF client, such as write-once error.<br>0b - No errors<br>1b - Errors exist |
| 27<br>DCMESF8 | Flash Memory Error<br>Indicates if a UTEST flash memory ECC error exists.<br>0b - No errors<br>1b - Errors exist |
| 26-24<br>DCMDCFF8 | DCF Record Location<br>Indicates the DCF record location.<br>010b - Utest flash memory<br>101b - Others: Reserved |
| 23-21<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 20-0<br>DCMDCFE8 | Flash Memory Address<br>Indicates the flash memory address of the DCF client having an error. |

## 38.3.16 DCF Scan Report (DCMSRR9)

### Offset

| Register | Offset |
|---|---|
| DCMSRR9 | 50h |

### Function

Resets on destructive reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | DCMDCFT9 | DCMESD9 | DCMESF9 | DCMDCFF9 | | | 0 | | | | DCMDCFE9 | | | |
| W | | | W1C | W1C | W1C | W1C | | | | | | | W1C | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCMDCFE9 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>DCMDCFT9 | Scanning Timeout On Flash Memory<br>Indicates if scanning timeout exists on flash memory address.<br>0b - Does not exist<br>1b - Exists |
| 28<br>DCMESD9 | Chip Side Error |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Indicates if an error exists on chip side. These errors could be parity errors or the ones reported by the DCF client, such as write-once error.<br>0b - No errors<br>1b - Errors exist |
| 27<br>DCMESF9 | Flash Memory Error<br>Indicates if a UTEST flash memory ECC error exists.<br>0b - No errors<br>1b - Errors exist |
| 26-24<br>DCMDCFF9 | DCF Record Location<br>Indicates the DCF record location.<br>010b - Utest flash memory<br>101b - Others: Reserved |
| 23-21<br>— | Reserved |
| 20-0<br>DCMDCFE9 | Flash Memory Address<br>Indicates the flash memory address of the DCF client having an error. |

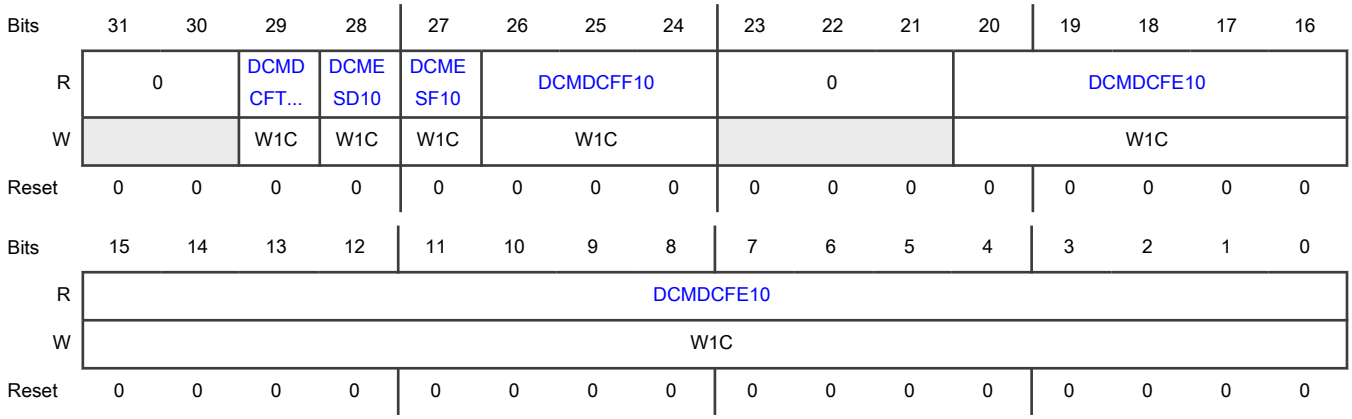
## 38.3.17 DCF Scan Report (DCMSRR10)

**Offset**

| Register | Offset |
|---|---|
| DCMSRR10 | 54h |

**Function**
Resets on destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>DCMDCFT10 | Scanning Timeout On Flash Memory<br>Indicates if scanning timeout exists on flash memory address.<br>0b - Does not exist<br>1b - Exists |
| 28<br>DCMESD10 | Chip Side Error<br>Indicates if an error exists on chip side. These errors could be parity errors or the ones reported by the DCF client, such as write-once error.<br>0b - No errors<br>1b - Errors exist |
| 27<br>DCMESF10 | Flash Memory Error<br>Indicates if a UTEST flash memory ECC error exists.<br>0b - No errors<br>1b - Errors exist |
| 26-24<br>DCMDCFF10 | DCF Record Location<br>Indicates the DCF record location.<br>010b - Utest flash memory<br>101b - Others: Reserved |
| 23-21<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 20-0<br>DCMDCFE10 | Flash Memory Address<br>Indicates the flash memory address of the DCF client having an error. |

## 38.3.18 DCF Scan Report (DCMSRR11)

### Offset

| Register | Offset |
|---|---|
| DCMSRR11 | 58h |

### Function

Resets on destructive reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | DCMDCFT... | DCMESD11 | DCMESF11 | DCMDCFF11 | | | 0 | | | | DCMDCFE11 | | | |
| W | | | W1C | W1C | W1C | W1C | | | | | | | W1C | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCMDCFE11 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>DCMDCFT11 | Scanning Timeout On Flash Memory<br>Indicates if scanning timeout exists on flash memory address.<br>0b - Does not exist<br>1b - Exists |
| 28<br>DCMESD11 | Chip Side Error |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Indicates if an error exists on chip side. These errors could be parity errors or the ones reported by the DCF client, such as write-once error.<br>0b - No errors<br>1b - Errors exist |
| 27<br>DCMESF11 | Flash Memory Error<br>Indicates if a UTEST flash memory ECC error exists.<br>0b - No errors<br>1b - Errors exist |
| 26-24<br>DCMDCFF11 | DCF Record Location<br>Indicates the DCF record location.<br>010b - Utest flash memory<br>101b - Others: Reserved |
| 23-21<br>— | Reserved |
| 20-0<br>DCMDCFE11 | Flash Memory Address<br>Indicates the flash memory address of the DCF client having an error. |

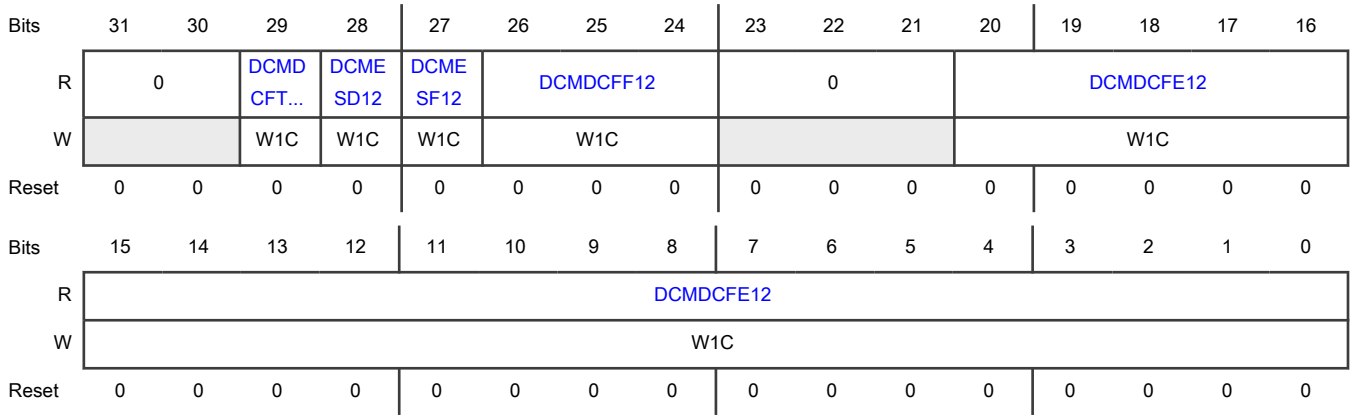
### 38.3.19 DCF Scan Report (DCMSRR12)

**Offset**

| Register | Offset |
|---|---|
| DCMSRR12 | 5Ch |

**Function**
Resets on destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>DCMDCFT12 | Scanning Timeout On Flash Memory<br>Indicates if scanning timeout exists on flash memory address.<br>0b - Does not exist<br>1b - Exists |
| 28<br>DCMESD12 | Chip Side Error<br>Indicates if an error exists on chip side. These errors could be parity errors or the ones reported by the DCF client, such as write-once error.<br>0b - No errors<br>1b - Errors exist |
| 27<br>DCMESF12 | Flash Memory Error<br>Indicates if a UTEST flash memory ECC error exists.<br>0b - No errors<br>1b - Errors exist |
| 26-24<br>DCMDCFF12 | DCF Record Location<br>Indicates the DCF record location.<br>010b - Utest flash memory<br>101b - Others: Reserved |
| 23-21<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 20-0<br>DCMDCFE12 | Flash Memory Address<br>Indicates the flash memory address of the DCF client having an error. |

## 38.3.20 DCF Scan Report (DCMSRR13)

### Offset

| Register | Offset |
|---|---|
| DCMSRR13 | 60h |

### Function

Resets on destructive reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | DCMDCFT... | DCMESD13 | DCMESF13 | DCMDCFF13 | | | 0 | | | | DCMDCFE13 | | | |
| W | | | W1C | W1C | W1C | W1C | | | | | | | W1C | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DCMDCFE13 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>DCMDCFT13 | Scanning Timeout On Flash Memory<br>Indicates if scanning timeout exists on flash memory address.<br>0b - Does not exist<br>1b - Exists |
| 28<br>DCMESD13 | Chip Side Error |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Indicates if an error exists on chip side. These errors could be parity errors or the ones reported by the DCF client, such as write-once error.<br>0b - No errors<br>1b - Errors exist |
| 27<br>DCMESF13 | Flash Memory Error<br>Indicates if a UTEST flash memory ECC error exists.<br>0b - No errors<br>1b - Errors exist |
| 26-24<br>DCMDCFF13 | DCF Record Location<br>Indicates the DCF record location.<br>010b - Utest flash memory<br>101b - Others: Reserved |
| 23-21<br>— | Reserved |
| 20-0<br>DCMDCFE13 | Flash Memory Address<br>Indicates the flash memory address of the DCF client having an error. |

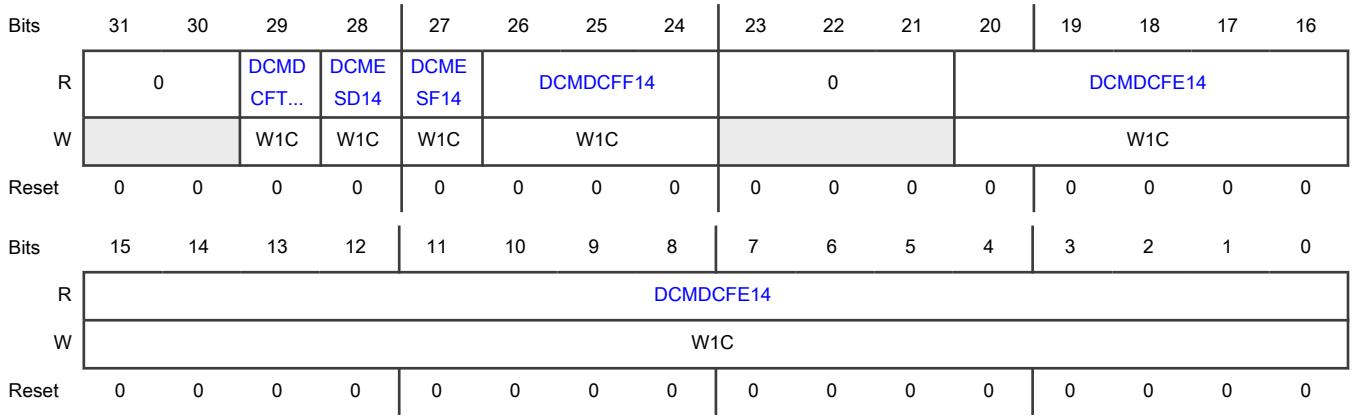
### 38.3.21 DCF Scan Report (DCMSRR14)

**Offset**

| Register | Offset |
|---|---|
| DCMSRR14 | 64h |

**Function**
Resets on destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>DCMDCFT14 | Scanning Timeout On Flash Memory<br>Indicates if scanning timeout exists on flash memory address.<br>0b - Does not exist<br>1b - Exists |
| 28<br>DCMESD14 | Chip Side Error<br>Indicates if an error exists on chip side. These errors could be parity errors or the ones reported by the DCF client, such as write-once error.<br>0b - No errors<br>1b - Errors exist |
| 27<br>DCMESF14 | Flash Memory Error<br>Indicates if a UTEST flash memory ECC error exists.<br>0b - No errors<br>1b - Errors exist |
| 26-24<br>DCMDCFF14 | DCF Record Location<br>Indicates the DCF record location.<br>010b - Utest flash memory<br>101b - Others: Reserved |
| 23-21<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 20-0<br>DCMDCFE14 | Flash Memory Address<br>Indicates the flash memory address of the DCF client having an error. |

## 38.3.22 DCF Scan Report (DCMSRR15)

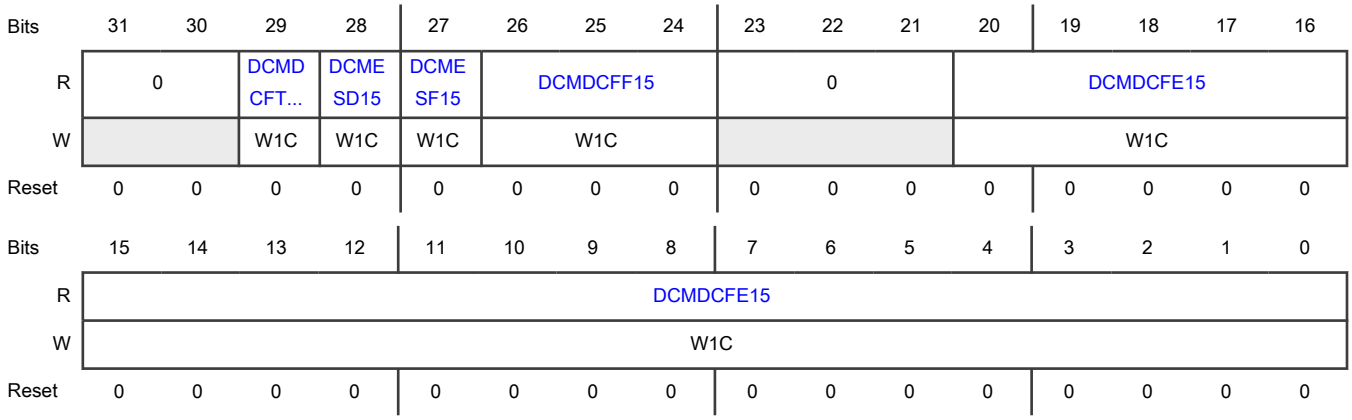
### Offset

| Register | Offset |
|---|---|
| DCMSRR15 | 68h |

### Function

Resets on destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>DCMDCFT15 | Scanning Timeout On Flash Memory<br>Indicates if scanning timeout exists on flash memory address.<br>0b - Does not exist<br>1b - Exists |
| 28<br>DCMESD15 | Chip Side Error |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Indicates if an error exists on chip side. These errors could be parity errors or the ones reported by the DCF client, such as write-once error.<br>0b - No errors<br>1b - Errors exist |
| 27<br>DCMESF15 | Flash Memory Error<br>Indicates if a UTEST flash memory ECC error exists.<br>0b - No errors<br>1b - Errors exist |
| 26-24<br>DCMDCFF15 | DCF Record Location<br>Indicates the DCF record location.<br>010b - Utest flash memory<br>101b - Others: Reserved |
| 23-21<br>— | Reserved |
| 20-0<br>DCMDCFE15 | Flash Memory Address<br>Indicates the flash memory address of the DCF client having an error. |

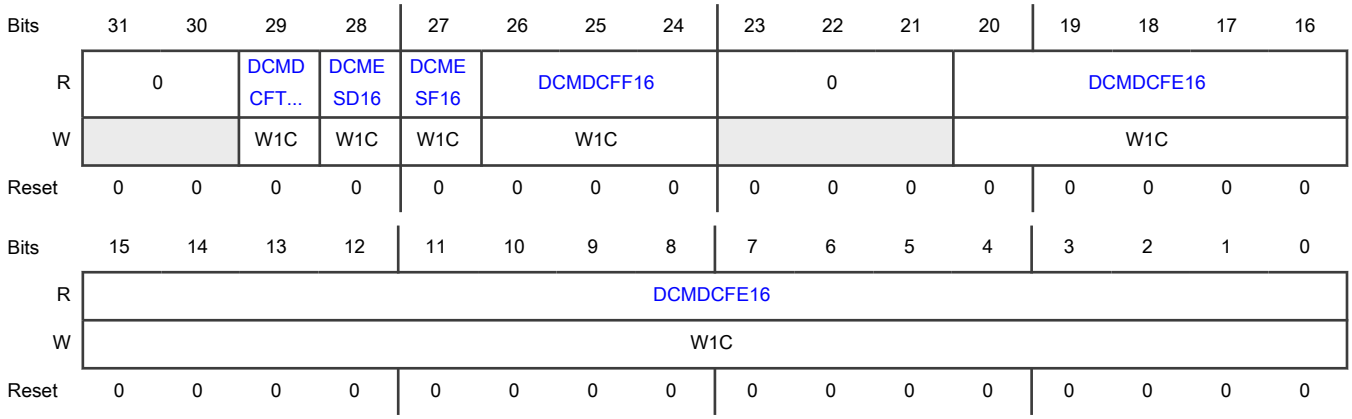
### 38.3.23 DCF Scan Report (DCMSRR16)

**Offset**

| Register | Offset |
|---|---|
| DCMSRR16 | 6Ch |

**Function**
Resets on destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>DCMDCFT16 | Scanning Timeout On Flash Memory<br>Indicates if scanning timeout exists on flash memory address.<br>0b - Does not exist<br>1b - Exists |
| 28<br>DCMESD16 | Chip Side Error<br>Indicates if an error exists on chip side. These errors could be parity errors or the ones reported by the DCF client, such as write-once error.<br>0b - No errors<br>1b - Errors exist |
| 27<br>DCMESF16 | Flash Memory Error<br>Indicates if a UTEST flash memory ECC error exists.<br>0b - No errors<br>1b - Errors exist |
| 26-24<br>DCMDCFF16 | DCF Record Location<br>Indicates the DCF record location.<br>010b - Utest flash memory<br>101b - Others: Reserved |
| 23-21<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 20-0<br>DCMDCFE16 | Flash Memory Address<br>Indicates the flash memory address of the DCF client having an error. |

## 38.3.24 LC Scan Status 2 (DCMLCS_2)

### Offset

| Register | Offset |
|---|---|
| DCMLCS_2 | 80h |

### Function

Stores the status of LC scanning. By default, the status of each LC is "not yet scanned."

This register:

- Resets on destructive reset.
- Always returns 0 in a valid IN_FIELD LC (in LC without an error).

This register captures the errors related to LC scanning on each of these resets: POR, destructive, and functional. If an error is captured, its status in this register is cleared by writing 1 to the corresponding field or to any of the destructive or POR events.

All LC slot errors are captured and cleared independently.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | DCMLCFE6 | DCMLCE6 | DCMLCC6 | | | DCMLCSS6 |
| W | | | | | | | | | | | W1C | W1C | W1C | | | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 5<br>DCMLCFE6 | Flash Memory Error Check<br>Indicates the status of flash memory check.<br>0b - No errors<br>1b - Errors exist |
| 4<br>DCMLCE6 | FA ECC Errors<br>Indicates if ECC errors exist in FA.<br>0b - No errors<br>1b - Errors exist |
| 3-1<br>DCMLCC6 | FA Marking<br>Indicates the FA marking status.<br>These errors may cause this field to indicate the "not scanned yet" status:<br>• If the reading completes too early and DCM has not yet scanned the LC.<br>• If there is an error in the flash memory after completion of the reading.<br>000b - Not scanned yet<br>001b - Marked as active<br>010b - Marked as inactive<br>011b - Region is erased/virgin<br>101b - Marked as inactive by an unknown pattern<br>110b - Scanning timed out |
| 0<br>DCMLCSS6 | FA Scan Status<br>Indicates if errors exist in the FA scan.<br>0b - No errors<br>1b - Errors exist |

## 38.4 Glossary

| | |
|---|---|
| **FA** | Failure analysis |
| **LC** | Life cycle |
| **Pre-FA** | Pre-failure analysis |
| **RoT** | Root of trust |
| **OTP** | One Time Programmable |
| **DCF** | Device Configuration Format |
| **PFC** | Platform Flash Controller |

# Chapter 39
# Messaging Unit (MU)

## 39.1 Chip-specific MU information

### 39.1.1 MU instances and configuration

This chip includes MUs for communication across the different cores. Each MU includes two interfaces, MUA and MUB. Two different processors control them for communication.

**NOTE**

The ELE_HSEB core controls the MUA interface of MU_0 and MU_1. Therefore, the application core cannot control the interface.

The base address of MU_1's MUB interface is different across the MCXE31 product series because AIPS_2 is unavailable in MCXE317 and MCXE315/MCXE316. Table 197 summarizes this difference.

**Table 197. Base-address difference in MU_1's MUB interface across MCXE 31 product series**

| | MCXE31B | MCXE317 | MCXE315/MCXE316 |
|---|---|---|---|
| Base address | 404E_C000h | 4039_0000h | |

## 39.2 Overview

MU enables two processors on a chip to communicate and coordinate by passing messages (for example, data, status, and control) through the MU interface. MU also allows one processor to signal the other processor using interrupts.

MU must synchronize the accesses from one side to the other because MU can manage messaging between processors using different clocks. MU accomplishes synchronization using two sets of matching registers: processor A-facing and processor B-facing.

### 39.2.1 Block diagram

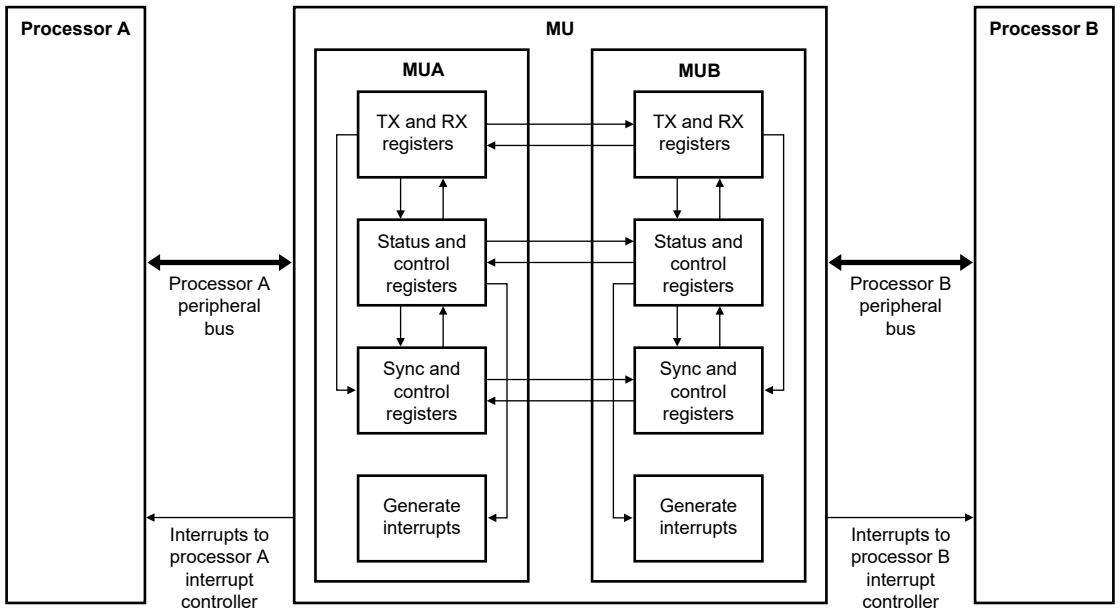



Figure 143. Block diagram

### 39.2.2 Features

- Memory-mapped registers (MU is connected as a peripheral under the peripheral bus on the processor A-side and processor B-side)
- Synchronized message transfers between cores
  - To send data or messages from one side to the other, MUA provides 4 transmit registers and 4 receive registers. MUB provides 4 transmit registers and 4 receive registers.
  - Transmit empty flags (TSR[TEn]) and receive full flags (RSR[RFn]) facilitate the transfer of data or messages between cores on both sides of MU.
  - A synchronization mechanism updates the transmit and the receive flags. There is an inherent latency between updating the flag on one side and reflecting its status on the other side. See Event update timing.
  - MU has a 3-bit flag data register, which you can use to send flag data between the two MU sides.
- Interprocessor interrupts
  - MU has 9 interrupt sources on each side (processor A-side, processor B-side) for signaling the other processor. You can use the interrupts for notification of receive and transmit events and for general-purpose signaling between processors. There are 1 general-purpose interrupt requests available and 8 receive and transmit interrupt sources.
- Reset (Each processor can issue a reset to MU via CR[MUR], which is a self-clearing field).

## 39.3 Functional description

MU enables two cores (processor A and processor B) to communicate with each other:

- By sharing messages and data.
- By enabling one core to wake the other core by using interrupts.

The messaging, control, and status registers of the two cores are mapped to processor A memory and processor B memory as a regular peripheral. The peripheral data bus is 32 bits wide inside MU.

Messaging logic is used with shared memory. Various messaging methods can implement a messaging protocol. For example, a message could mean one of the following:

- A message of *n* words has been written starting at offset *x* in the memory.
- The previous data block that was sent has been read.

The ability to keep messaging logic independent of the shared memory is not restricted to a predefined hardware protocol. The software required to manage the messaging is short and straightforward.

Most of the messaging mechanisms are symmetric. They are duplicated and are available on both the processor A side and processor B side. The messaging mechanisms are:

- 4 32-bit transmit registers, which are each reflected in 4 read-only receive registers on the side of the other processor. These registers can transfer 32-bit word messages or the frame information of the messages written to the shared memory. For example, they can transfer the number of words, initial address, and message type code.
- Writing to a transmitter-side transmit register (TR*n*) clears the Transmitter Empty flag in the transmitter-side Status register (TS*n*[TE*n*]), and sets the Receiver Full flag in the receiver-side Status register (RS*n*[RF*n*]). Setting the flag on the receiver side can trigger an interrupt on the receiver side (a maskable receive interrupt).
- Reading a receiver-side receive register (RR*n*) clears the Receiver Full flag in the receiver-side Status register (RS*n*[RF*n*]), and sets the Transmitter Empty flag in the transmitter-side Status register (TS*n*[TE*n*]). Setting the Transmitter Empty flag can trigger an interrupt on the transmitter side (a maskable transmit interrupt).
- 1 general-purpose interrupt request flags are reflected in General-purpose Status (GSR) on the receiver side.
- 3-bit flag data is transmitted from Flag Control (FCR) to Flag Status (FSR) on the side receiving the flag data. SR[FUP] sets when the flag data is transmitted and clears when the receiving side acknowledges the flag data (the flag is updated).

Writing to a transmit register signals to the receiver side that data is ready for retrieval.

Do not write to the transmit register again without verifying that the data is retrieved. The transmitter side cannot determine the exact time that the receiver attempts to retrieve the data. Before attempting to write to the transmit register again, the transmitter side must wait for the Transmitter Empty interrupt or it must poll TSR[TE*n*].

Reading a receive register signals to the transmitter side that data can be written to that register.

Do not read the receive register again without verifying that the data is written. The receiver side cannot determine the exact time that the transmitter attempts to write the data. Before attempting to read the receive register again, the receiver side must wait for the Receiver Full interrupt or it must poll RSR[RF*n*].

## 39.3.1 Submodules

### 39.3.1.1 MUA side

MUA receives its register configuration via the processor A peripheral bus. It sends or receives messages to or from MUB. Processor A can receive messages by reading MUA registers, and MUA can send interrupts to processor A when interrupts are enabled.

**NOTE**

Processor B should not access MUA registers. Processor C, if exist, should also not access MUA registers. TRDC might be needed to prevent such illegal access of Processor B or C to MUA registers.

### 39.3.1.2 MUB side

MUB receives its register configuration via the processor B peripheral bus. It sends or receives messages to or from MUA. Processor B can receive messages by reading MUB registers, and MUB can send interrupts to processor B when interrupts are enabled.

**NOTE**

Processor A should not access MUB registers. Processor C, if exist, should also not access MUB registers. TRDC might be needed to prevent such illegal access of Processor A or C to MUB registers.

### 39.3.2 Event update timing

The messaging side of each processor has a hardware mechanism to send event update requests to the other processor. An event occurs when the status register of the receiving processor must reflect any information change. The event update latency is the delay between the event being ready at one processor and the resulting update at the status register of the other processor:

- The minimum event latency is (1 clock cycle of the sending side) + (2.5 clock cycles of the receiving side). This minimum case happens when no event is pending when the new event occurs.
- The maximum event latency is (6 clock cycles of the sending side) + (6.5 clock cycles of the receiving side). The maximum case happens when the event occurs just after a previous event is sent to the other side.

The event update latency varies depending on the time at which the subsequent event is triggered.

### 39.3.3 Clocking

**Table 198. MU clocks**

| Clock name | Description |
|---|---|
| Bus clock MUA | Used only for bus accesses to MUA control and configuration registers |
| Bus clock MUB | Used only for bus accesses to MUB control and configuration registers |

### 39.3.4 Resets

The following sections list the reset sources included in MU. Each reset performs a different function for the MU module compared to the function it performs for the system.

#### 39.3.4.1 Asynchronous system reset

When the asynchronous system reset on either side of MU is asserted, SR[MURS] becomes 1 until the asynchronous system reset sequence on both the MUA and MUB sides ends. Verify that SR[MURS] becomes 0 before accessing MU.

The asynchronous system reset on one side of MU resets the other side of MU. The reset forces all control and status registers to return to their default values and clears all internal states. The following table shows the exceptions to this behavior.

**Table 199. Exceptions to asynchronous system reset**

| MUA-side exceptions | MUB-side exceptions |
|---|---|
| MUB_CR[MURIE] | MUA_CR[MURIE] |
| MUB_SR[MURIP] | MUA_SR[MURIP] |
| MUB_SR[MURS] | MUA_SR[MURS] |

#### 39.3.4.2 MU software reset

Writing 1 to CR[MUR] causes most control and status registers to return to their default values and clears all internal states.

The instruction immediately following the assertion of CR[MUR] must not write to the MU registers. The reset sequence may overwrite such a write, with the register retaining the reset value. To know the end of the reset sequence for both processors, monitor the value of SR[MURS]. After the reset sequence on both processors has ended, a write to the MU registers can be attempted.

**NOTE**

The process of CR[MUR] becoming 1 is delicate because it asynchronously affects the registers on the other side. CR[MUR] becoming 1 may cause unpredictable behavior if, for example, the other processor is concurrently testing an MU register field (TSR[TE*n*] in the other processor). Before writing 1 to CR[MUR], verify that the other processor is not engaged in an MU signaling activity.

### 39.3.5 Interrupts

MU controls interrupt requests that one processor makes to the other processor. This section describes all the interrupts that the module generates.

MU can generate these interrupt sources individually to send to the processors:

- 4 receive interrupts (asserted when the Receive Full flags are set in Receive Status (RSR) and enabled in Receive Control (RCR)) for each receive register. These interrupts are Ored into a single interrupt for each MUA or MUB, please consult the interrupt map file attached to this document.
- 4 transmit interrupts (asserted when the Transmit Empty flags are set in Transmit Status (TSR) and enabled in Transmit Control (TCR)) for each transmit register. These interrupts are Ored into a single interrupt for each MUA or MUB, please consult the interrupt map file attached to this document.
- 1 general-purpose interrupt (asserted when the GIP flag is set in General-Purpose Interrupt Enable (GIER) and enabled in General-Purpose Interrupt Enable (GIER))
- Core Reset Assertion interrupts (asserted when the processor enters the Reset state and enabled by RAIE)

All interrupts are maskable in the processor control registers: TCR, RCR, GIER, and CR. MU does not assume any internal priority of these interrupts. Multiple interrupts (for example, receive 0 and receive 1, or any transmit or general-purpose interrupt) can be asserted simultaneously. The interrupt controller must resolve the priority of these interrupts at the chip level.

Triggering any enabled interrupt wakes the processor from below mode before servicing the interrupt

- Wait mode
- Stop mode
- Power-down mode

The software (as part of the interrupt handler) must clear the general-purpose interrupt pending flag (GSR[GIP0]) to deassert the request to the interrupt controller.

When a processor writes to the general-purpose interrupt flag (GCR[GIR0]), MU synchronizes the write event to the other processor to set the general-interrupt request pending flag (GSR[GIP0]). When GSR[GIP0] is set, if the general-purpose interrupt is enabled on the receiving processor side (GIER[GIE0] is 1), the transmitting-side general-purpose interrupt is issued to the receiving processor, which clears this interrupt by writing 1 to GSR[GIP0]. The interrupt is deasserted as soon as the write to GSR[GIP0] occurs. MU synchronizes the write event of GSR[GIP0] to the other processor. The synchronized signal clears the GIR0 flag.

Before writing 1 to GCR[GIR0], verify that this field is 0, which means that a general interrupt is not pending. Generally, MU ignores writing 1 to this field while the field is already 1, but in some cases it may issue a second interrupt.

## 39.4 External signals

This module has no external signals.

## 39.5 Initialization

MU does not require initialization.

## 39.6 Application information

MU facilitates messages between processors. For example, MU passes:

- Short messages. Transmit registers can pass short messages from 1 to 4 words in length for processor A and from 1 to 4 words for processor B. For example, for a four-word message, only one register must have its corresponding interrupt enabled on the receiving side. The first three words of the message are written to the registers with masked interrupts. The fourth word is written to the last register, triggering an interrupt on the receiving side.
- Frame information. Transmit registers can pass frame information for long messages written to the shared system memory. Such frame information normally includes a start address and a number of words. It can include a message type code.
- Event notices and requests. MU can signal events and requests that do not include data words between processors using general-purpose interrupts. For example, one such event is acknowledging that a long message is read from the shared system memory.
- Fixed-length data. Formatted data with a fixed length can be written to predetermined locations in the shared memory. A processor can use general-purpose interrupts to signal to the other processor that the data is ready.
- Announcements. A processor can use the 3 flags to announce its current program state or other billboard messages to the other processor.

### 39.6.1 Messaging protocols using interrupts

The example below describes a four-word messaging sequence sent between the processor A and processor B.

The transmitting processor writes to the transmit registers sequentially. When *n* = 0, 1, and 2, the interrupts are disabled, so no interrupt goes to the processor (although interrupt conditions occur). For *n* = 3, the interrupt is enabled, and MU generates the last receive interrupt request.

1. Write sequence:
    a. The transmitting processor writes the message information sequentially to its Transmit registers 0, 1, 2.
    b. When the write to Transmit register 3 occurs, RSR[RF3] is set after synchronization. It immediately triggers the receive full 3 interrupt on the receiving processor.
2. Read sequence:
    a. The receiving processor receives the receive full 3 interrupt and starts reading the message transferred from the receive registers.
    b. After the receiving processor reads Receive register 3, MU clears RSR[RF3].

Table 200 and Figure 144 describe the messaging model, using transmit and receive registers and the interrupt messaging protocol.

**Table 200. Interrupt messaging protocol (generalized)**

| Step | Action | Description |
|---|---|---|
| 1 | Processor A writes data. | RR*n* on processor B's side reflects a data write to TR*n* by processor A. |
| 2 | Clear the transmitter empty flag and set the receiver full flag. | The data write to TR*n*:<br>• Clears TSR[TE*n*] on the processor A side.<br>• Sets RSR[RF*n*] on the processor B side. |
| 3 | Generate the receive interrupt request. | Setting RSR[RF*n*] generates a receive interrupt request to processor B. |
| 4 | Processor B reads the data. | After receiving the receive interrupt request, processor B performs a data read of RR*n*. |

*Table continues on the next page...*

Table 200. Interrupt messaging protocol (generalized) (continued)

| Step | Action | Description |
|---|---|---|
| 5 | Clear the receiver full flag and set the transmitter empty flag. | Reading the data from the RR*n* register:<br>• Clears RSR[RF*n*] on the processor B side.<br>• Sets TSR[TE*n*] on the processor A side. |
| 6 | Generate the transmit interrupt request. | Setting TSR[TE*n*] generates a transmit interrupt request to processor A. |

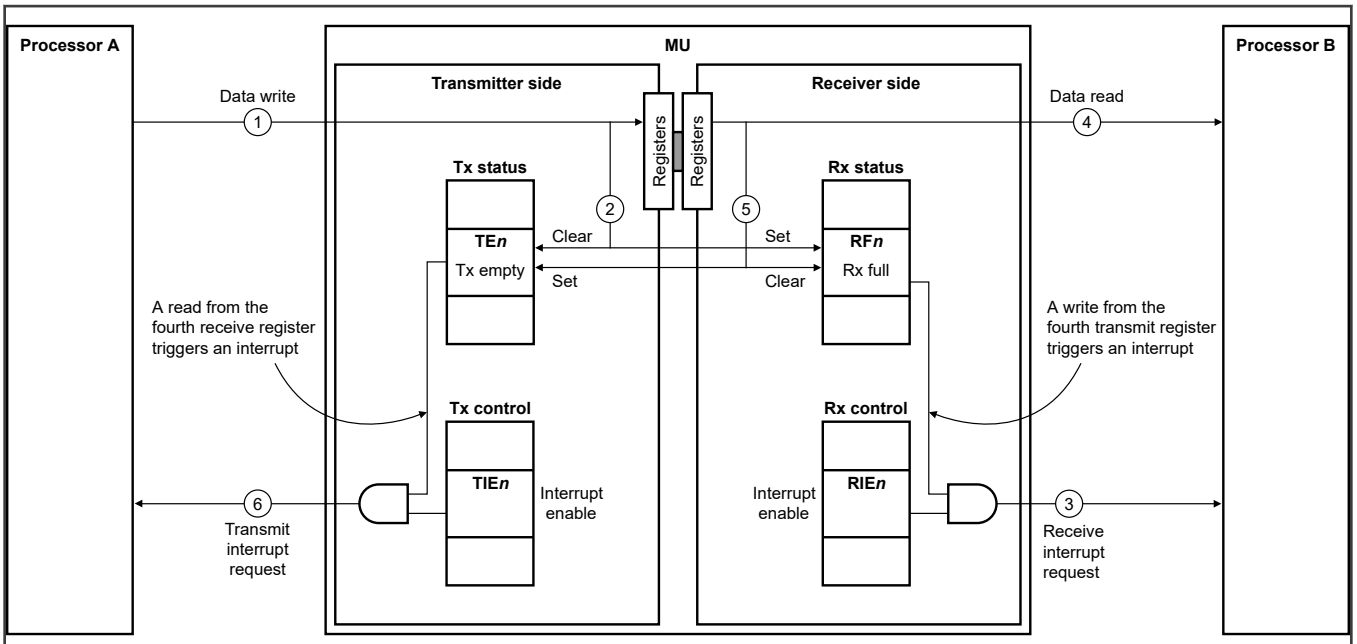



Figure 144. Messaging model using transmit and receive registers

You can use the messaging hardware to implement messaging protocols for an array of message types. MU provides full support for interrupt and polling management schemes.

## 39.6.2 Messaging protocols using event interrupts

MU can signal events and requests that do not include data words between processors using general-purpose interrupts.

Formatted data with a fixed length can be written to predetermined locations in the shared memory. A processor can use a general-purpose interrupt to signal to the other processor that the data is ready.

A processor can use the 3 flags to announce its current program state (or similar messages) to the other processor.

Table 201 and Figure 145 describe the event steps when the processor triggers a general-purpose interrupt.

Table 201. General-purpose interrupt messaging protocol (generalized)

| Step | Action | Description |
|---|---|---|
| 1 | Processor A sets its associated general-purpose interrupt request flag. | Processor A sets GCR[GIR0]. |

*Table continues on the next page...*

Table 201. General-purpose interrupt messaging protocol (generalized) (continued)

| Step | Action | Description |
|---|---|---|
| 2 | The general-purpose interrupt request pending status flag is set. | GSR[GIP0] is set. |
| 3 | MU generates the general-purpose interrupt request to processor B. | Setting GSR[GIP0] generates the general-purpose interrupt request to processor B. GIER[GIE0] must be set for processor B. |
| 4 | Processor B reads the status register. | Processor B reads GSR[GIP0]. |
| 5 | Processor B services the interrupt. | — |
| 6 | Processor B sets GSR[GIP0] to clear the interrupt. | Processor B writes 1 to the corresponding GSR[GIP0] flag to clear the interrupt. |
| 7 | GCR[GIR0] is cleared. | Setting GSR[GIP0] clears GCR[GIR0] on the processor A side. |

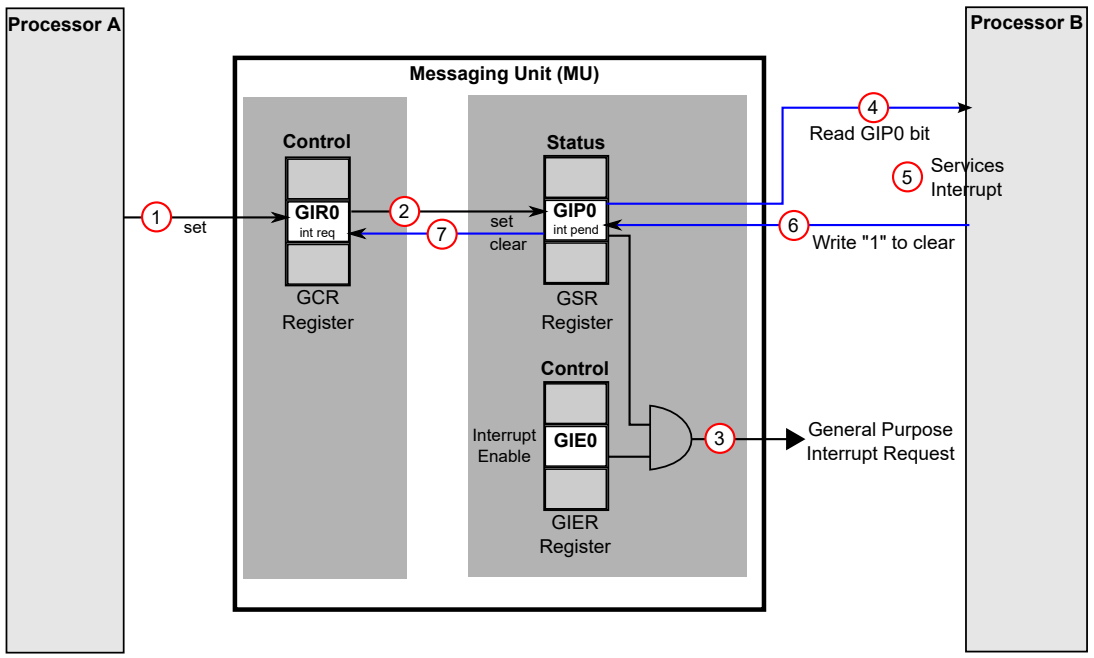



Figure 145. Messaging model using a general-purpose interrupt

## 39.6.3 Exclusive access to shared memory

MU can signal one processor about its current access to shared memory. This signaling prevents the other processor from overwriting data during the exclusive memory access period.

The following tables describe the signaling protocol that processor A uses to inform processor B about its current access (write) to shared memory:

- Table 202 shows processor A performing an exclusive access to shared memory.
- Table 203 shows processor B scanning for transaction information.

- Table 204 shows processor B accepting exclusive access from processor A.
- Table 205 shows processor B rejecting exclusive access from processor A.

According to the examples shown in the following tables, GCR[GIR0], RR*n*, and TR*n* are reserved to support exclusive access to the shared memory protocol.

**Table 202. Processor A performs an exclusive access to shared memory**

| Step | Action | Description |
|---|---|---|
| 1 | Processor A sends the GIR0 request to processor B using the processor A control register. | Before processor A performs an exclusive access to the shared memory, it sends a GIR0 request to processor B. |
| 2 | Processor A sends an exclusive-access request using a transmit data register (TR*n*). | Processor A sends an exclusive-access request (command, location, and length of target access) to processor B using a selected transmit data register (TR0). |
| 3 | Processor A waits for a dedicated interrupt from processor B. | Processor A waits for a dedicated interrupt (as an acknowledgment) triggered by processor B before proceeding. |
| 4 | Processor A accesses the shared memory. | After receiving a dedicated interrupt from processor B, processor A proceeds. |

**Table 203. Processor B scans for transaction information**

| Step | Action | Description |
|---|---|---|
| 1 | Processor B receives an interrupt from a receive data register (RR*n*). | — |
| 2 | Processor B reads the receive data register (RR*n*). | — |
| 3 | Processor B scans the receive data register contents. | Processor B scans for transaction information (whether processor A has requested exclusive access). |

**Table 204. Processor B accepts exclusive access from processor A**

| Step | Action | Description |
|---|---|---|
| 1 | Processor B triggers a dedicated interrupt. | Processor B acknowledges the request from processor A by triggering a dedicated interrupt (ack) to processor A. |
| 2 | Processor B sends a code message to processor A. | With the acknowledge interrupt, processor B sends a code message to processor A through the selected transmit register (TR*n*). The message informs processor A that it can exclusively access the shared memory. |

**Table 205. Processor B rejects exclusive access from processor A**

| Step | Action | Description |
|---|---|---|
| 1 | Processor B ignores the request from processor A for exclusive access. | If processor B does not provide permission to processor A, processor B ignores the exclusive-access request. |

### 39.6.4 Packet data transfers

The following table describes an example packet transfer sequence between processor B and processor A subsystems.

**Table 206. Packet data transfer sequence**

| Step | Action | Description |
|---|---|---|
| 1 | Processor B requests DMA. | Processor B sends a DMA request to initiate the packet data transfer. |
| 2 | DMA data transfer | DMA acknowledges. |
| 3 | | DMA starts transferring data from the specified processor B memory location to the specified shared memory. |
| 4 | | DMA interrupts processor B to signal that the packet transfer has finished. |
| 5 | Processor B informs processor A that data is in shared memory. | Using a B-side transmit register, processor B sends a packet information message to processor A about the arrival of new packet data stored in shared memory. The message contains the command, location, and length of packet data. |
| 6 | Processor A receives an interrupt. | Processor A receives an interrupt (assuming its corresponding processor A MU-side receive interrupt is enabled). The pending processing task becomes active and processes packet data from memory. |
| 7 | Processor A reads data, then writes data. | Processor A reads or processes packet data from shared memory. |
| 8 | | Processor A writes the result from packet processing to a separate buffer. |
| 9 | Processor A informs processor B that the transfer is finished. | After the processing of the packet data finishes, processor A informs processor B (using MU processor A-side transmit register, MUA_TR*n*). |
| 10 | Processor A sends an interrupt to processor B (a request for more data). | Processor B receives the next interrupt from processor A, in which processor A requests more packet data. |

### 39.6.5 Freeing a processor from deadlock

During normal operation, one processor may determine that the other processor is not working or is deadlocked. Using Status (SR), the processor can use the methods in the following table to identify and correct the problem.

**Table 207. Freeing a processor from deadlock**

| Processor mode | Technique | Description |
|---|---|---|
| — | Processor issues an interrupt. | The other processor can interrupt the processor by issuing one of the 9 (general purpose, receive, or transmit) interrupts. |
| — | Processor issues a hardware reset. | If the above methods are not helpful, the processor can issue a hardware reset strobe to the other processor. |

## 39.7 Register definition

MU provides transmit and receive data registers (*x*TR0–*n*, *x*RR0–*n*) for communication between processor A and processor B. It also provides control registers (*x*CR) for operations such as interrupts and resets, and status registers (*x*SR) for checking the status of the other MU-side. Figure 146 shows the schematic for MU registers.

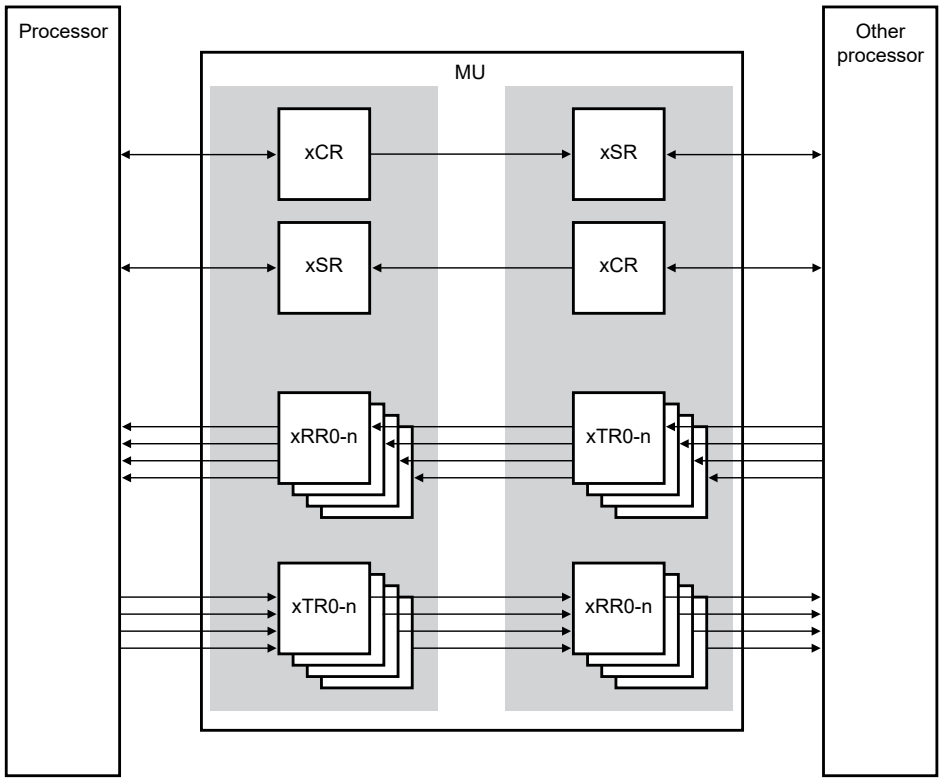



Figure 146. MU registers

### 39.7.1 MU register descriptions

This section contains the detailed register descriptions for MUA registers.

**NOTE**

A module transfer error to processor A or processor B is generated when:

- A read or write access is made to an invalid location.
- A write operation is performed on a read-only register on the processor A side or processor B side of MU.

#### 39.7.1.1 MU memory map

MU_1.MUA base address: 404E_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID (VER) | 32 | R | 0300_000Fh |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 4h | Parameter (PAR) | 32 | R | 0301_0404h |
| 8h | Control (CR) | 32 | RW | 0000_0000h |
| Ch | Status (SR) | 32 | RW | See section |
| 10h | Core Control 0 (CCR0) | 32 | R | 0000_0000h |
| 100h | Flag Control (FCR) | 32 | RW | 0000_0000h |
| 104h | Flag Status (FSR) | 32 | R | 0000_0000h |
| 110h | General-Purpose Interrupt Enable (GIER) | 32 | RW | 0000_0000h |
| 114h | General-Purpose Control (GCR) | 32 | RW | 0000_0000h |
| 118h | General-purpose Status (GSR) | 32 | RW | 0000_0000h |
| 120h | Transmit Control (TCR) | 32 | RW | 0000_0000h |
| 124h | Transmit Status (TSR) | 32 | R | 0000_000Fh |
| 128h | Receive Control (RCR) | 32 | RW | 0000_0000h |
| 12Ch | Receive Status (RSR) | 32 | R | 0000_0000h |
| 200h - 20Ch | Transmit (TR0 - TR3) | 32 | RW | 0000_0000h |
| 280h - 28Ch | Receive (RR0 - RR3) | 32 | R | 0000_0000h |

#### 39.7.1.2 Version ID (VER)

**Offset**

| Register | Offset |
|---|---|
| VER | 0h |

**Function**

Determines the version ID and feature set number of MUA.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number |
| 23-16<br>MINOR | Minor Version Number |
| 15-0<br>FEATURE | Feature Set Number<br>Indicates the feature set number.<br>MU implements:<br>• Standard features<br>• RAIP and RAIE fields<br>• Expanded number of TR*n*/RR*n* registers |

## 39.7.1.3 Parameter (PAR)

### Offset

| Register | Offset |
|---|---|
| PAR | 4h |

### Function

Defines the number of flags, transmit registers, receive registers, and general-purpose interrupt requests available for MU.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | FLAG_WIDTH | | | | | | | | GIR_NUM | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

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

### Fields

| Field | Function |
|---|---|
| 31-24<br>FLAG_WIDTH | Flag Width<br>Specifies the number of flag bits (3) in the Flag Control (FCR) and Flag Status (FSR) registers. |
| 23-16<br>GIR_NUM | General-Purpose Interrupt Request Number<br>Specifies the number of general-purpose interrupt requests available (1). |
| 15-8<br>RR_NUM | Receive Register Number<br>Specifies the number of receive registers (4). |
| 7-0<br>TR_NUM | Transmit Register Number<br>Specifies the number of transmit registers (4). |

## 39.7.1.4 Control (CR)

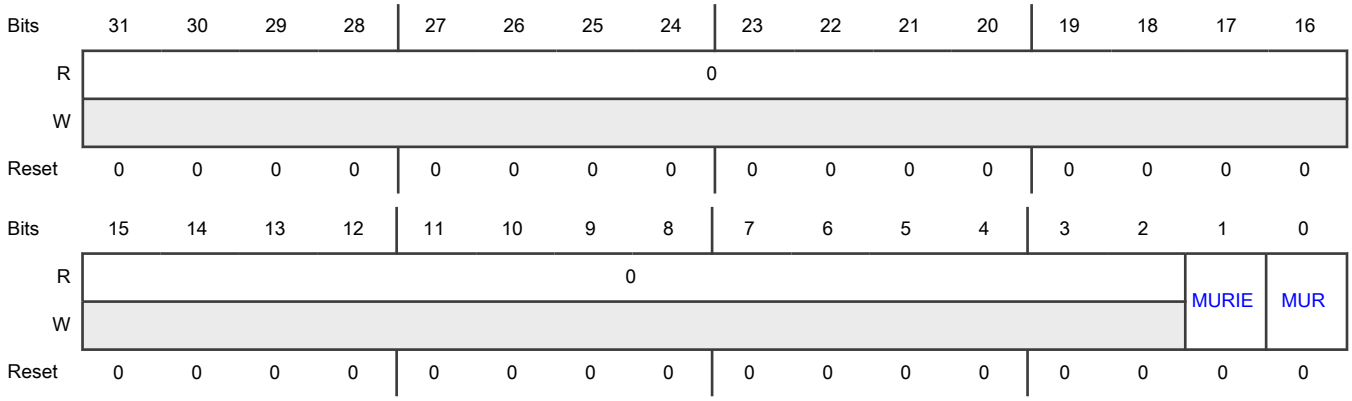
### Offset

| Register | Offset |
|---|---|
| CR | 8h |

### Function

Controls MU reset and reset interrupt enable.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>MURIE | MUA Reset Interrupt Enable<br>Enables the processor A-side MU reset interrupt request due to MU reset issued by MUB.<br>If the value of this field is 1, an MU reset interrupt request is issued to processor A when MUA_SR[MURIP] = 1.<br>If the value of this field is 0, MU ignores the value of MURIP and issues no MU reset interrupt request.<br>Only a system reset can reset this field. CR[MUR] cannot reset this field.<br>0b - Disable<br>1b - Enable |
| 0<br>MUR | MU Reset<br>Resets MU. Writing 1 to this field resets the MUA and MUB sides. All internal states are cleared. It forces all control and status registers to return to their default values (except in MUA/B_CCR0 registers; MURIE in MUA/B_CR registers; MURIP and MURS in MUA/B_SR registers).<br>Before writing 1 to this field, interrupt processor B because writing 1 to this field may affect the ongoing processor B program.<br>After writing 1 to this field, monitor the value of MUA_SR[MURS] to know when the reset sequence on the processor B-side has ended.<br>This field always reads 0, and it becomes 0 during the MU reset sequence.<br>0b - Idle<br>1b - Reset |

### 39.7.1.5 Status (SR)

#### Offset

| Register | Offset |
|---|---|
| SR | Ch |

#### Function

Shows the status of MU resets and the status of pending events and requests.

#### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | 0 | RFP | TEP | GIRP | FUP | EP | MURIP | MURS |
| W | | | | | | | | | | | | | | | W1C | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | u |

#### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7<br>— | Reserved |
| 6<br>RFP | MUA Receive Full Pending<br>Indicates whether a receive full message is pending.<br>This field becomes 1 when MUB writes to a TR*n* register to send data to MUA. After this field becomes 1, MU checks RSR[RF*n*] to determine whether the data in the Receive register is ready for MUA to read it.<br>This field becomes 0 when all MUA RR*n* registers are read, or when MU is reset.<br>0b - Not pending; MUB is not writing to a Transmit register<br>1b - Pending; MUB is writing to a Transmit register |
| 5<br>TEP | MUA Transmit Empty Pending<br>Indicates whether a transmit empty message is pending. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | This field becomes 1 when any TCR[TIE*n*] field is 1 and TSR[TE*n*] flag is set. After this field becomes 1, MU checks TSR[TE*n*] to determine whether the data in the Transmit (TR*n*) register is ready for MUA to write to it.<br>This field becomes 0 when write operations to all MUA Transmit (TR*n*) registers where TCR[TIE*n*] = 1 (transfer interrupt enabled) are completed, or when MU is reset.<br>0b - No MUA transmit empty event pending<br>1b - Pending; any TCR[TIEn] field is 1 and TSR[TEn] flag is set |
| 4<br>GIRP | MUA General-Purpose Interrupt Pending<br>Indicates that MUB has sent a general-purpose interrupt request.<br>This field becomes 1 when the MUB side sends a general-purpose interrupt request to the MUA side. GSR[GIP0] identifies which general-purpose interrupt request is received.<br>This field becomes 0 when MUA_GSR[GIP0] is cleared, or when MU is reset.<br>0b - No request sent<br>1b - Request sent |
| 3<br>FUP | MUA Flag Update Pending<br>Indicates whether a flag update request is pending. MU generates this request when there is a change to the F*n*[2:0] bits of MUA_FCR.<br>This field becomes 1 when the MUA side sends a flag update request to the MUB side.<br>This field becomes 0 when MU acknowledges this flag update request internally (the flag is updated) from the MUB side, or during MU reset.<br>No flag update changes are allowed when this field is 1. When FUP = 1, a write to the F*n*[2:0] bits of MUA_FCR does not generate a flag update event. The F*n*[2:0] bits do not change.<br>If SR[EP] = 1 (event pending), writing to MUA_FCR does not immediately cause this field to become 1.<br>0b - No pending update flags (initiated by MUA)<br>1b - Pending update flags (initiated by MUA) |
| 2<br>EP | MUA Side Event Pending<br>Indicates a pending side event when the MUA side sends an event update request to the MUB side. An event is any hardware message that the Status register on the MUB side reflects. For example, an event occurs when Transmit register 0 is the target of a write operation. During normal operations, the update mechanism for this field operates automatically.<br>MU clears this field automatically when the event update acknowledgment is received, or when MU resets.<br>To ensure that events are posted to MUB, verify that this field is 0. If it is 1, wait and continue to poll this field until it becomes 0.<br>0b - Not pending<br>1b - Pending |
| 1 | MU Reset Interrupt Pending Flag |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| MURIP | Indicates whether processor B has issued an MU reset.<br>This flag is set after processor B initiates an MU reset by setting MUB_CR[MUR]. If CR[MURIE] = 1, the processor A MU reset interrupt request is issued when processor B writes 1 to MUB_CR[MUR].<br>Clearing this flag also clears the MU reset interrupt request.<br>Only a system reset can reset this flag. MU reset cannot reset this flag.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Reset not issued<br>1b - Reset issued<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 0<br>MURS | MUA and MUB Reset State<br>Indicates the reset state of MUA and MUB.<br>This field becomes 1 during any system reset or MU reset from the MUA or MUB side.<br>This field becomes 0 when the reset sequence on both MUA and MUB sides ends. After issuing any of the aforementioned reset events, verify that this field is 0 before starting any access.<br>0b - Out of reset<br>1b - In reset |

### 39.7.1.6 Core Control 0 (CCR0)

#### Offset

| Register | Offset |
|---|---|
| CCR0 | 10h |

#### Function

Allows MUA to control the processor on the MUB side.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-7<br>— | Reserved |
| 6-5<br>— | Reserved |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>— | Reserved |

## 39.7.1.7 Flag Control (FCR)

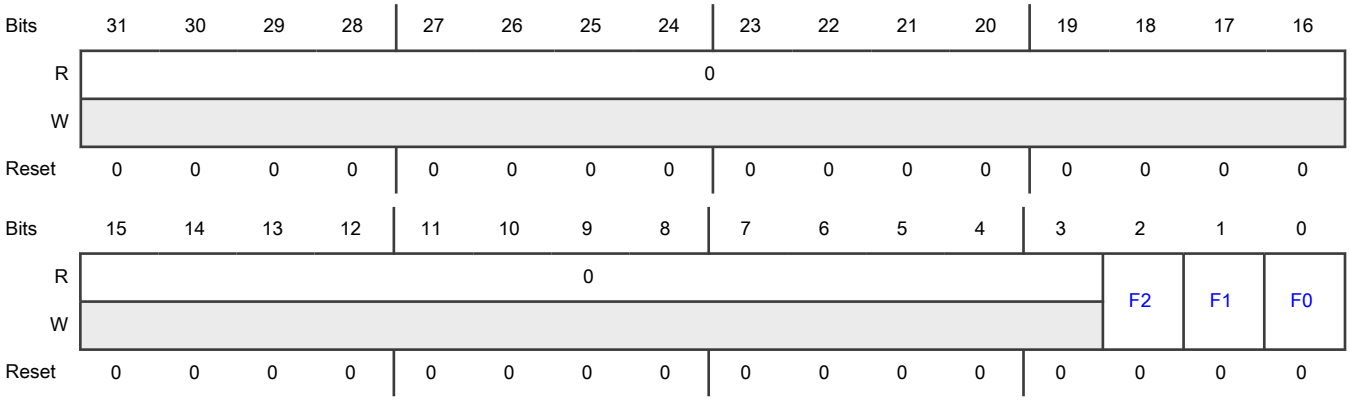
### Offset

| Register | Offset |
|---|---|
| FCR | 100h |

### Function

Configures MUB_FSR[F*n*] flags.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2-0<br>Fn | MUA to MUB Flag<br>Configures MUB_FSR[F*n*] flags, where *n* = 0 to 2.<br>F*n* configures the corresponding MUB_FSR[F*n*] flag.<br>F*n* becomes 0 when MU resets.<br>0b - Clear MUB_FSR[Fn]<br>1b - Set MUB_FSR[Fn] |

## 39.7.1.8 Flag Status (FSR)

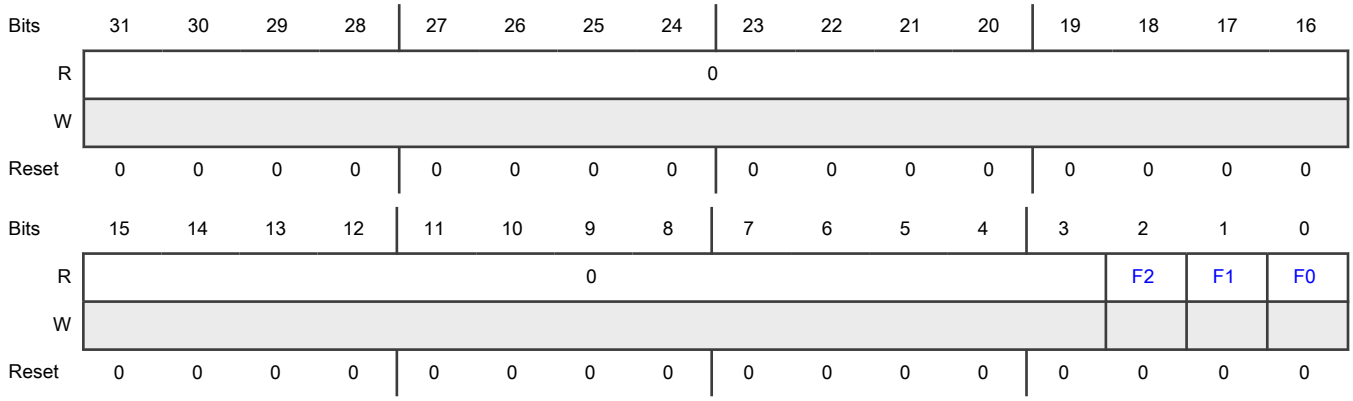
### Offset

| Register | Offset |
|---|---|
| FSR | 104h |

### Function

Contains flags configured by the values written to MUB_FCR[F*n*].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2-0<br>Fn | MUB to MUA-Side Flag<br>Contains flags configured by the values written to MUB_FCR[F*n*], where *n* = 0 to 2.<br>F*n* is the MUA-side flag configured by the values written to MUB_FCR[F*n*].<br>When MUB_FCR[F*n*] is written to, the write event updates MUA_FSR[F*n*], after the event update latency.<br>0b - MUB_FCR[Fn] = 0<br>1b - MUB_FCR[Fn] = 1 |

## 39.7.1.9 General-Purpose Interrupt Enable (GIER)

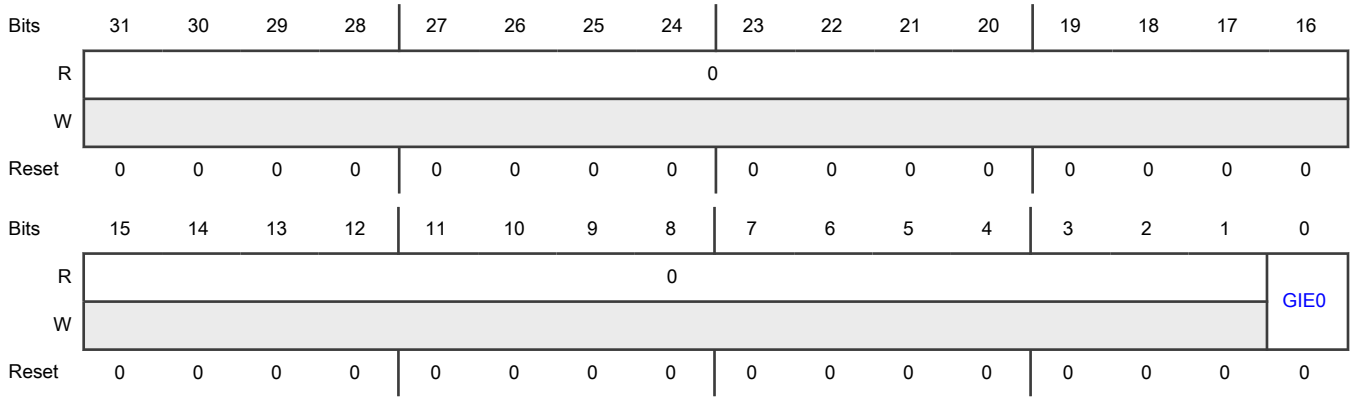
### Offset

| Register | Offset |
|---|---|
| GIER | 110h |

### Function

Contains the MUA general-purpose interrupt enable fields.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>GIEn | MUA General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There is one general-purpose interrupt (*n* = 0).<br>When GIE0 = 1, a general-purpose interrupt *n* request is issued to processor A when MUA GSR[GIP0] = 1.<br>If GIE0 = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIE0 becomes 0 when MU resets.<br>0b - Disable<br>1b - Enable |

## 39.7.1.10 General-Purpose Control (GCR)

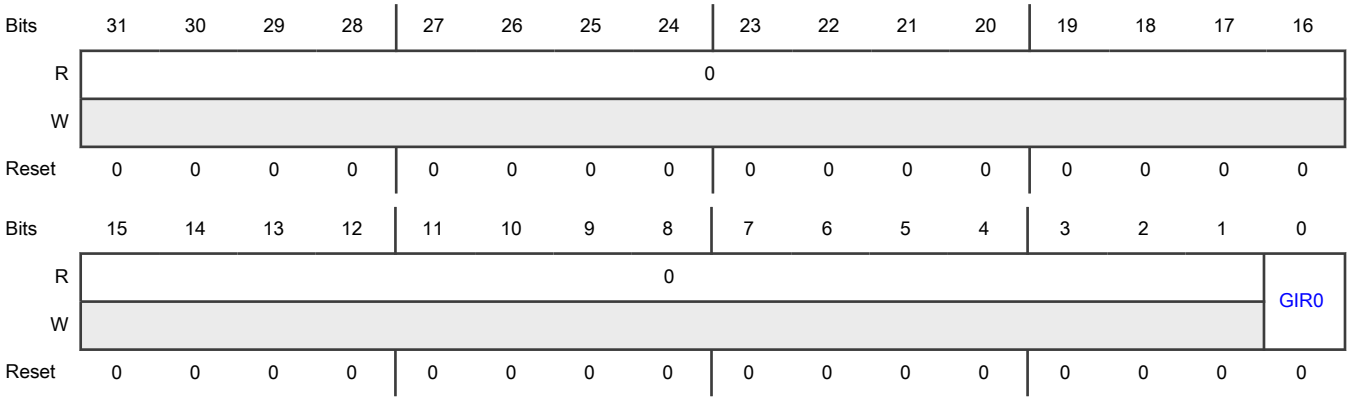
### Offset

| Register | Offset |
|---|---|
| GCR | 114h |

### Function

Contains the MUA general-purpose interrupt request fields.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>GIRn | MUA General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUB. There is one general-purpose interrupt (*n* = 0).<br>Writing 1 to GIR0 sets MUB_GSR[GIP0]. If MUB_GIER[GIE0] = 1, a general-purpose interrupt request is triggered on processor B.<br>This field becomes 0 when MUB_GSR[GIP0] is cleared. This clearing informs MUA that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIR0 is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>0b - Not requested<br>1b - Requested |

## 39.7.1.11 General-purpose Status (GSR)

### Offset

| Register | Offset |
|---|---|
| GSR | 118h |

### Function

Contains the status of the MUA general-purpose interrupt pending requests.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | GIP0 |
| W | | | | | | | | | | | | | | | | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>GIPn | MUA General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There is one general-purpose interrupt (*n* = 0).<br>GIP0 informs MUA that MUB_GCR[GIR0] changed from 0 to 1. If MUA_GIER[GIE0] = 1, a general-purpose interrupt request is issued to processor A.<br>GIP0 is cleared when MU resets.<br>After GIP0 is cleared, if MUA_GIER[GIE0] = 1, the general-purpose interrupt request is cleared on the MUA side.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag |

## 39.7.1.12 Transmit Control (TCR)

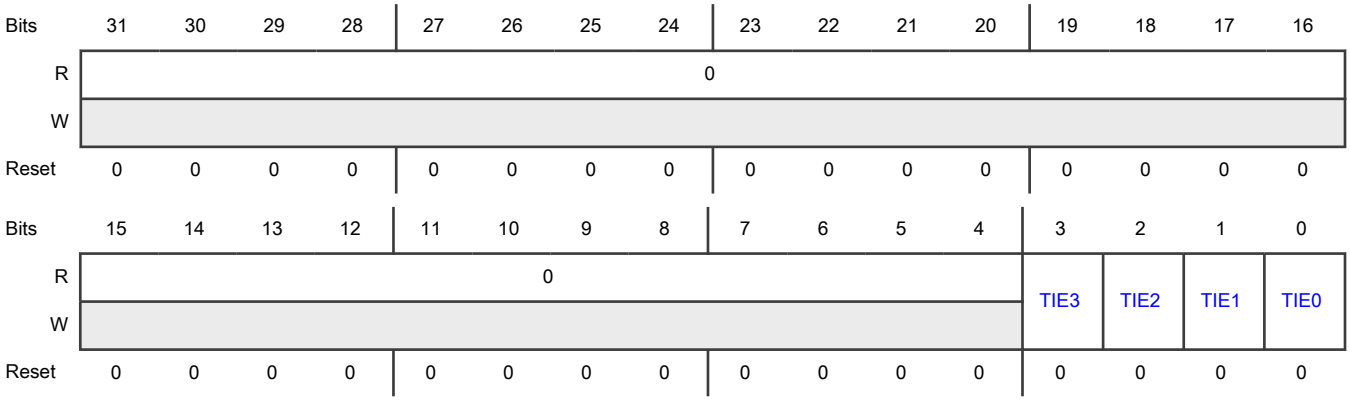
### Offset

| Register | Offset |
|---|---|
| TCR | 120h |

### Function

Contains the MUA transmit interrupt enable fields.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>TIEn | MUA Transmit Interrupt Enable<br>Enables MUA transmit interrupt *n*, where *n* = 0 to 3.<br>If this field is 1, an MUA transmit interrupt *n* request is issued when MUA_TSR[TE*n*] is set.<br>If this field is 0, MU ignores the value of MUA_TSR[TE*n*], and no MUA transmit interrupt *n* request is issued.<br>0b - Disable<br>1b - Enable |

## 39.7.1.13 Transmit Status (TSR)

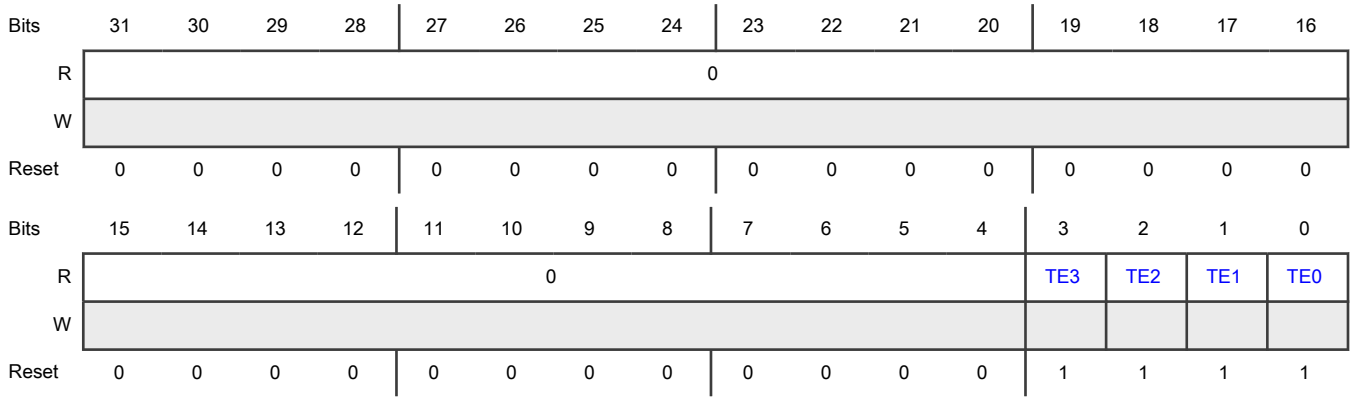
### Offset

| Register | Offset |
|---|---|
| TSR | 124h |

### Function

Indicates whether the MUA transmit registers are empty.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>TEn | MUA Transmit Empty<br>Indicates whether MUA Transmit (TR*n*) register is empty, where *n* = 0 to 3.<br>This field becomes 1 after the MUB_RR*n* register is read on the MUB side. When TE*n* = 1, it indicates to the MUA side that the MUA_TR*n* register is ready to be written on the MUA side. If MUA_TCR[TIE*n*] = 1, a transmit *n* interrupt is issued on the MUA side.<br>This field becomes 0 after the MUA_TR*n* register is written to on the MUA side. After this field becomes 0, if MUA_TCR[TIE*n*] = 1, the transmit *n* interrupt request is cleared on the MUA side.<br>This field becomes 1 when MU resets.<br>0b - Not empty<br>1b - Empty |

## 39.7.1.14 Receive Control (RCR)

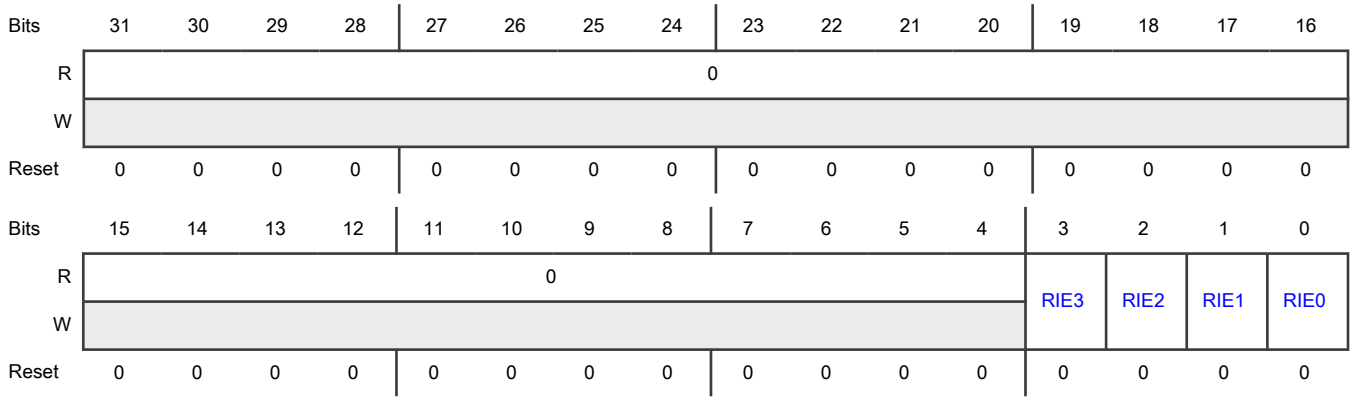
### Offset

| Register | Offset |
|---|---|
| RCR | 128h |

### Function

Contains the MUA receive interrupt enables.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>RIEn | MUA Receive Interrupt Enable<br>Enables MUA receive interrupt *n*, where *n* = 0 to 3.<br>If this field is 1, an MUA receive interrupt *n* request is issued when MUA_RSR[RF*n*] is set.<br>If this field is 0, MU ignores the value of MUA_RSR[RF*n*], and no MUA receive interrupt request is issued.<br>0b - Disable<br>1b - Enable |

## 39.7.1.15 Receive Status (RSR)

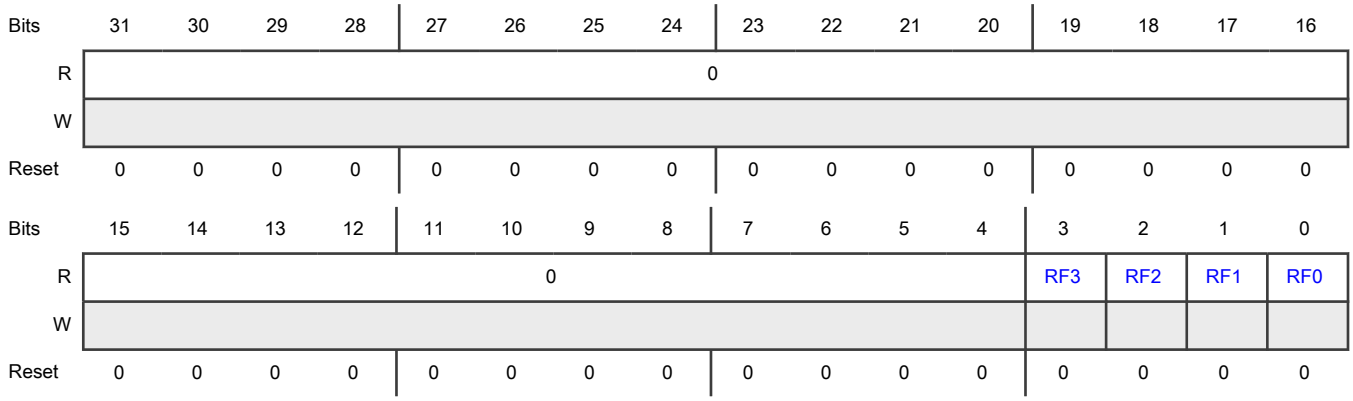
### Offset

| Register | Offset |
|---|---|
| RSR | 12Ch |

### Function

Indicates whether the MUA receive registers are full.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>RFn | MUA Receive Register Full<br>Indicates whether MUA Receive register (RR*n*) is full, where *n* = 0 to 3.<br>RF*n* becomes 1 when the MUB_TR*n* register is written to on the MUB side.<br>When RF*n* is 1, it indicates to the MUA side that new data in the MUA_RR*n* register is ready for MUA to read it. If MUA_RCR[RIE*n*] = 1, a receive *n* interrupt is issued on the MUA side.<br>RF*n* becomes 0 when the MUA_RR*n* register is read, or when MU is reset.<br>After RF*n* becomes 0, if MUA_RCR[RIE*n*] = 1, the receive *n* interrupt request is cleared on the MUA side.<br>0b - Not full<br>1b - Full |

## 39.7.1.16 Transmit (TR0 - TR3)

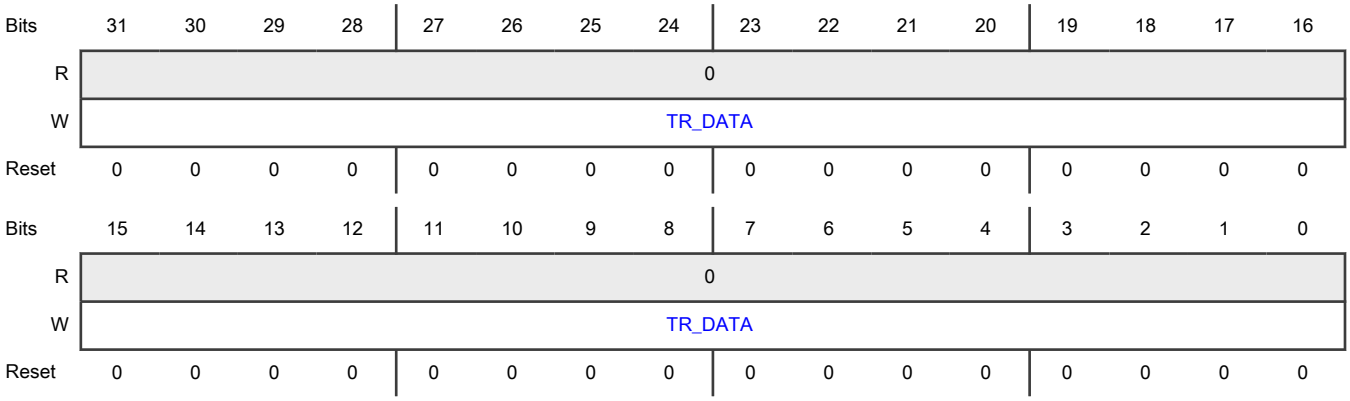
### Offset

| Register | Offset |
|---|---|
| TR0 | 200h |
| TR1 | 204h |
| TR2 | 208h |
| TR3 | 20Ch |

### Function

Contains MUA transmit data.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>TR_DATA | MUA Transmit Data<br>Contains MUA transmit data. MUB_RR*n* reflects the data written to this register.<br>The TR*n* and RR*n* registers are not double-buffered. Writing to MUA_TR*n* overrides the data readable in the MUA_RR*n* register.<br>A write to the Transmit register clears MUA_TSR[TE*n*] on the transmitter side, and sets MUB_RSR[RF*n*] on the receiver side.<br>You can write to this register only when MUA_TSR[TE*n*] = 1.<br>Reading this register returns all zeroes. |

## 39.7.1.17 Receive (RR0 - RR3)

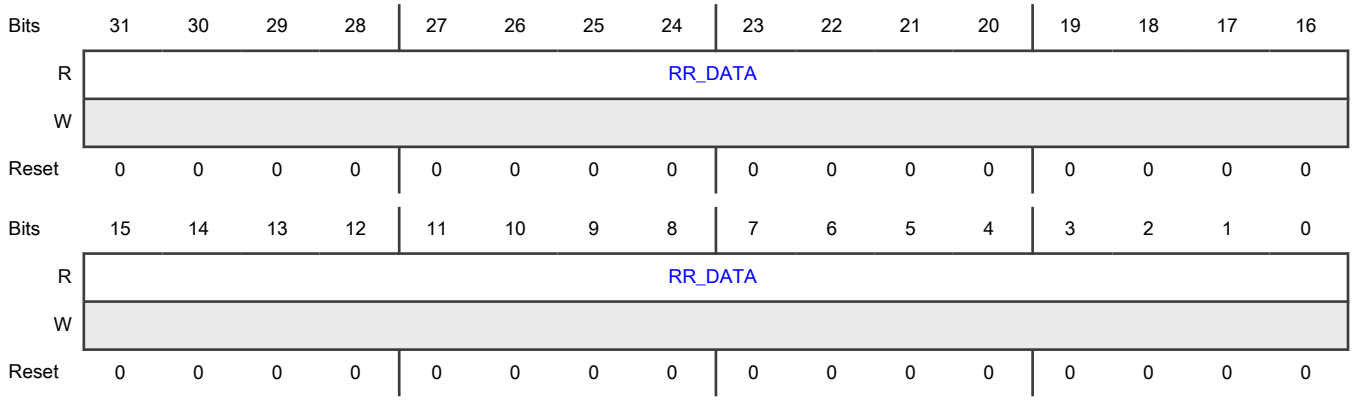
### Offset

| Register | Offset |
|---|---|
| RR0 | 280h |
| RR1 | 284h |
| RR2 | 288h |
| RR3 | 28Ch |

### Function

Contains MUA receive data.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>RR_DATA | MUA Receive Data<br>Reflects the data written to MUB TR*n*.<br>Reading this register clears MUA_RSR[RF*n*] on the receiver side, and sets MUB_TSR[TE*n*] on the transmitter side.<br>You can read this register only when MUA_RSR[RF*n*] = 1. Reading it before MUA_RSR[RF*n*] becomes 1 may result in reading incorrect data. Poll MUA_RSR[RF*n*] to confirm it is set before reading RR*n*.<br>Writing to this register generates an error response to MUA. |

## 39.7.2 MU register descriptions

This section contains the detailed register descriptions for MUB registers.

**NOTE**

A module transfer error to processor A or processor B is generated when:

- A read or write access is made to an invalid location.
- A write operation is performed on a read-only register on the processor A side or processor B side of MU.

### 39.7.2.1 MU memory map

MU_0.MUB base address: 404E_C000h

MU_1.MUB base address: 4038_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID (VER) | 32 | R | 0300_000Fh |
| 4h | Parameter (PAR) | 32 | R | See section |
| 8h | Control (CR) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| Ch | Status (SR) | 32 | RW | See section |
| 10h | Core Control 0 (CCR0) | 32 | R | 0000_0000h |
| 100h | Flag Control (FCR) | 32 | RW | 0000_0000h |
| 104h | Flag Status (FSR) | 32 | R | 0000_0000h |
| 110h | General-Purpose Interrupt Enable (GIER) | 32 | RW | 0000_0000h |
| 114h | General-Purpose Control (GCR) | 32 | RW | 0000_0000h |
| 118h | General-purpose Status (GSR) | 32 | RW | 0000_0000h |
| 120h | Transmit Control (TCR) | 32 | RW | 0000_0000h |
| 124h | Transmit Status (TSR) | 32 | R | 0000_000Fh |
| 128h | Receive Control (RCR) | 32 | RW | 0000_0000h |
| 12Ch | Receive Status (RSR) | 32 | R | 0000_0000h |
| 200h - 20Ch | Transmit (TR0 - TR3) | 32 | RW | 0000_0000h |
| 280h - 28Ch | Receive (RR0 - RR3) | 32 | R | 0000_0000h |

### 39.7.2.2 Version ID (VER)

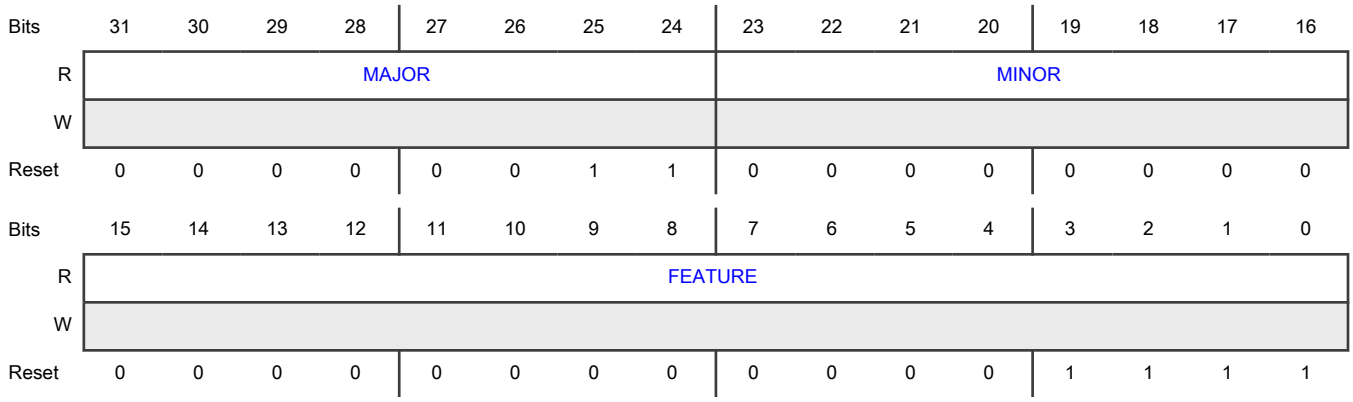
#### Offset

| Register | Offset |
|---|---|
| VER | 0h |

#### Function

Determines the version ID and feature set number of MUB.

#### Diagram



### Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number |
| 23-16<br>MINOR | Minor Version Number |
| 15-0<br>FEATURE | Feature Set Number<br>Indicates the feature set number.<br>MU implements:<br>• Standard features<br>• RAIP and RAIE fields<br>• Expanded number of TR*n*/RR*n* registers |

## 39.7.2.3 Parameter (PAR)

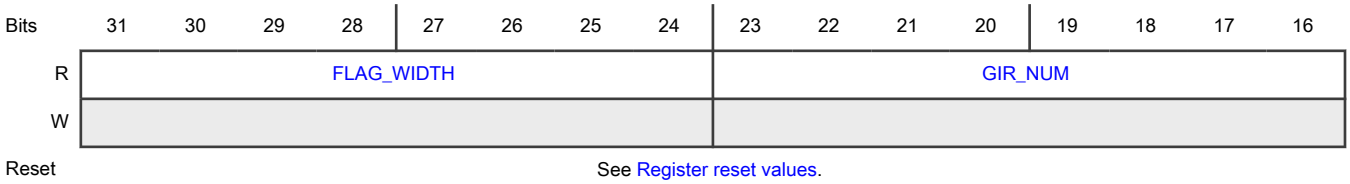
### Offset

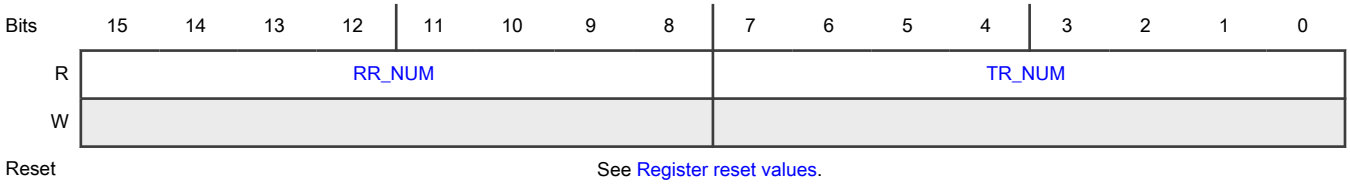
| Register | Offset |
|---|---|
| PAR | 4h |

### Function

Defines the number of flags, transmit registers, receive registers, and general-purpose interrupt requests available for MU.

### Diagram



### Register reset values

| Register | Reset value |
|---|---|
| PAR | MU_0.MUB: 2020_0404h<br>MU_1.MUB: 0301_0404h |

### Fields

| Field | Function |
|---|---|
| 31-24<br>FLAG_WIDTH | Flag Width<br>Specifies the number of flag bits (32) in the Flag Control (FCR) and Flag Status (FSR) registers. |
| 23-16<br>GIR_NUM | General-Purpose Interrupt Request Number<br>Specifies the number of general-purpose interrupt requests available (32). |
| 15-8<br>RR_NUM | Receive Register Number<br>Specifies the number of receive registers (4). |
| 7-0<br>TR_NUM | Transmit Register Number<br>Specifies the number of transmit registers (4). |

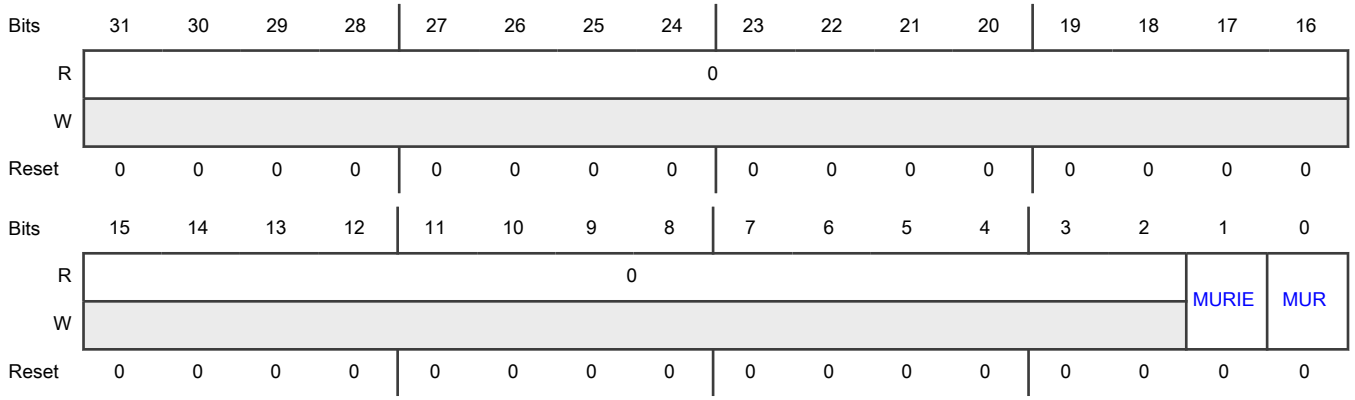
## 39.7.2.4 Control (CR)

### Offset

| Register | Offset |
|---|---|
| CR | 8h |

### Function
Controls MU reset and reset interrupt enable.

### Diagram



**Fields**

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>MURIE | MUB Reset Interrupt Enable<br>Enables the processor B-side MU reset interrupt request due to MU reset issued by MUA.<br>If the value of this field is 1, an MU reset interrupt request is issued to processor B when MUB_SR[MURIP] = 1.<br>If the value of this field is 0, MU ignores the value of MURIP and issues no MU reset interrupt request.<br>Only a system reset can reset this field. CR[MUR] cannot reset this field.<br>0b - Disable<br>1b - Enable |
| 0<br>MUR | MU Reset<br>Resets MU. Writing 1 to this field resets the MUB and MUA sides. All internal states are cleared. It forces all control and status registers to return to their default values (except in MUB/A_CCR0 registers; MURIE in MUB/A_CR registers; MURIP and MURS in MUB/A_SR registers).<br>Before writing 1 to this field, interrupt processor A because writing 1 to this field may affect the ongoing processor A program.<br>After writing 1 to this field, monitor the value of MUB_SR[MURS] to know when the reset sequence on the processor A-side has ended.<br>This field always reads 0, and it becomes 0 during the MU reset sequence.<br>0b - Idle<br>1b - Reset |

### 39.7.2.5 Status (SR)

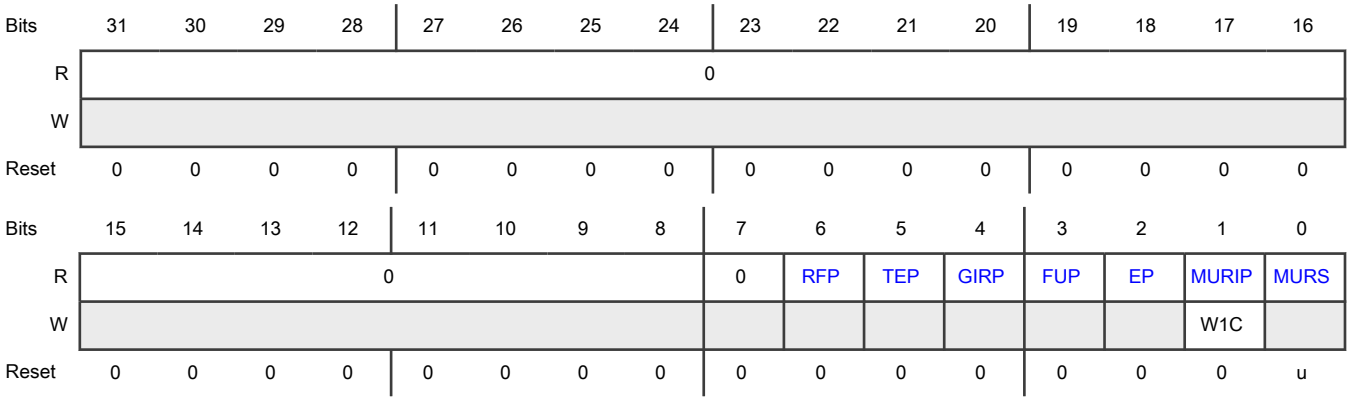
**Offset**

| Register | Offset |
|---|---|
| SR | Ch |

**Function**

Shows the status of MU resets and the status of pending events and requests.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7<br>— | Reserved |
| 6<br>RFP | MUB Receive Full Pending<br>Indicates whether a receive full message is pending.<br>This field becomes 1 when MUA writes to a TR*n* register to send data to MUB. After this field becomes 1, MU checks RSR[RF*n*] to determine whether the data in the Receive register is ready for MUB to read it.<br>This field becomes 0 when all MUB RR*n* registers are read, or when MU is reset.<br>0b - Not pending; MUA is not writing to a Transmit register<br>1b - Pending; MUA is writing to a Transmit register |
| 5<br>TEP | MUB Transmit Empty Pending<br>Indicates whether a transmit empty message is pending.<br>This field becomes 1 when any TCR[TIE*n*] field is 1 and TSR[TE*n*] flag is set. After this field becomes 1, MU checks TSR[TE*n*] to determine whether the data in the Transmit (TR*n*) register is ready for MUB to write to it.<br>This field becomes 0 when write operations to all MUB Transmit (TR*n*) registers where TCR[TIE*n*] = 1 (transfer interrupt enabled) are completed, or when MU is reset.<br>0b - No MUB transmit empty event pending<br>1b - Pending; any TCR[TIEn] field is 1 and TSR[TEn] flag is set |
| 4<br>GIRP | MUB General-Purpose Interrupt Pending<br>Indicates that MUA has sent a general-purpose interrupt request.<br>This field becomes 1 when the MUA side sends a general-purpose interrupt request to the MUB side. GSR[GIP*n*] identifies which general-purpose interrupt request is received. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | This field becomes 0 when all MUB_GSR[GIP*n*] fields are cleared, or when MU is reset.<br>0b - No request sent<br>1b - Request sent |
| 3<br>FUP | MUB Flag Update Pending<br>Indicates whether a flag update request is pending. MU generates this request when there is a change to the F*n*[31:0] bits of MUB_FCR.<br>This field becomes 1 when the MUB side sends a flag update request to the MUA side.<br>This field becomes 0 when MU acknowledges this flag update request internally (the flag is updated) from the MUA side, or during MU reset.<br>No flag update changes are allowed when this field is 1. When FUP = 1, a write to the F*n*[31:0] bits of MUB_FCR does not generate a flag update event. The F*n*[31:0] bits do not change.<br>If SR[EP] = 1 (event pending), writing to MUB_FCR does not immediately cause this field to become 1.<br>0b - No pending update flags (initiated by MUB)<br>1b - Pending update flags (initiated by MUB) |
| 2<br>EP | MUB Side Event Pending<br>Indicates a pending side event when the MUB side sends an event update request to the MUA side. An event is any hardware message that the Status register on the MUA side reflects. For example, an event occurs when Transmit register 0 is the target of a write operation. During normal operations, the update mechanism for this field operates automatically.<br>MU clears this field automatically when the event update acknowledgment is received, or when MU resets.<br>To ensure that events are posted to MUA, verify that this field is 0. If it is 1, wait and continue to poll this field until it becomes 0.<br>0b - Not pending<br>1b - Pending |
| 1<br>MURIP | MU Reset Interrupt Pending Flag<br>Indicates whether processor A has issued an MU reset.<br>This flag is set after processor A initiates an MU reset by setting MUB_CR[MUR]. If CR[MURIE] = 1, the processor B MU reset interrupt request is issued when processor A writes 1 to MUA_CR[MUR].<br>Clearing this flag also clears the MU reset interrupt request.<br>Only a system reset can reset this flag. MU reset cannot reset this flag.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Reset not issued |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Reset issued<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 0<br>MURS | MUA and MUB Reset State<br>Indicates the reset state of MUA and MUB.<br>This field becomes 1 during any system reset or MU reset from the MUA or MUB side.<br>This field becomes 0 when the reset sequence on both MUA and MUB sides ends. After issuing any of the aforementioned reset events, verify that this field is 0 before starting any access.<br>0b - Out of reset<br>1b - In reset |

### 39.7.2.6 Core Control 0 (CCR0)

#### Offset

| Register | Offset |
|---|---|
| CCR0 | 10h |

#### Function

Allows MUB to control the processor on the MUA side.

#### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-7<br>— | Reserved |
| 6-5<br>— | Reserved |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>— | Reserved |

## 39.7.2.7 Flag Control (FCR)

### Offset

| Register | Offset |
|---|---|
| FCR | 100h |

### Function

Configures MUA_FSR[F*n*] flags.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | F31 | F30 | F29 | F28 | F27 | F26 | F25 | F24 | F23 | F22 | F21 | F20 | F19 | F18 | F17 | F16 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | F15 | F14 | F13 | F12 | F11 | F10 | F9 | F8 | F7 | F6 | F5 | F4 | F3 | F2 | F1 | F0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>31<br>F31</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr></table><br>0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]</td></tr>
<tr><td>30<br>F30</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr></table><br>0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]</td></tr>
<tr><td>29<br>F29</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.</td></tr>
</table>

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>28<br>F28</td><td>
MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>27<br>F27</td><td>
MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>26<br>F26</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>25<br>F25</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>24<br>F24</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>23<br>F23</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>22</td><td>MUB to MUA Flag</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>F22</td><td>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr></table><br>0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]</td></tr>
<tr><td>21<br>F21</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr></table><br>0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]</td></tr>
<tr><td>20<br>F20</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>19<br>F19</td><td>
MUB to MUA Flag<br>
Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>
Fn configures the corresponding MUA_FSR[Fn] flag.<br>
Fn becomes 0 when MU resets.
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>18<br>F18</td><td>
MUB to MUA Flag<br>
Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>
Fn configures the corresponding MUA_FSR[Fn] flag.<br>
Fn becomes 0 when MU resets.
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>17<br>F17</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>16<br>F16</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>15<br>F15</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>14<br>F14</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>13</td><td>MUB to MUA Flag</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>F13</td><td>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr></table><br>0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]</td></tr>
<tr><td>12<br>F12</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr></table><br>0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]</td></tr>
<tr><td>11<br>F11</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>10<br>F10</td><td>
MUB to MUA Flag<br>
Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>
Fn configures the corresponding MUA_FSR[Fn] flag.<br>
Fn becomes 0 when MU resets.
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>9<br>F9</td><td>
MUB to MUA Flag<br>
Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>
Fn configures the corresponding MUA_FSR[Fn] flag.<br>
Fn becomes 0 when MU resets.
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>8<br>F8</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>7<br>F7</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>6<br>F6</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>5<br>F5</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr>
</table>
0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]
</td></tr>
<tr><td>4</td><td>MUB to MUA Flag</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>F4</td><td>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr></table><br>0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]</td></tr>
<tr><td>3<br>F3</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>FCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>FCR</td></tr></table><br>0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]</td></tr>
<tr><td>2<br>F2</td><td>MUB to MUA Flag<br>Configures MUA_FSR[Fn] flags, where n = 0 to 31.<br>Fn configures the corresponding MUA_FSR[Fn] flag.<br>Fn becomes 0 when MU resets.<br>0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn]</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 1<br>F1 | MUB to MUA Flag<br>Configures MUA_FSR[F*n*] flags, where *n* = 0 to 31.<br>F*n* configures the corresponding MUA_FSR[F*n*] flag.<br>F*n* becomes 0 when MU resets.<br>0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn] |
| 0<br>F0 | MUB to MUA Flag<br>Configures MUA_FSR[F*n*] flags, where *n* = 0 to 31.<br>F*n* configures the corresponding MUA_FSR[F*n*] flag.<br>F*n* becomes 0 when MU resets.<br>0b - Clear MUA_FSR[Fn]<br>1b - Set MUA_FSR[Fn] |

### 39.7.2.8 Flag Status (FSR)

#### Offset

| Register | Offset |
|---|---|
| FSR | 104h |

#### Function

Contains flags configured by the values written to MUA_FCR[F*n*].

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | F31 | F30 | F29 | F28 | F27 | F26 | F25 | F24 | F23 | F22 | F21 | F20 | F19 | F18 | F17 | F16 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | F15 | F14 | F13 | F12 | F11 | F10 | F9 | F8 | F7 | F6 | F5 | F4 | F3 | F2 | F1 | F0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>31<br>F31</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr></table><br>0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1</td></tr>
<tr><td>30<br>F30</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr></table><br>0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1</td></tr>
<tr><td>29<br>F29</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.</td></tr>
</table>

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1
</td></tr>
<tr><td>28<br>F28</td><td>
MUA to MUB-Side Flag<br>
Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>
Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>
When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1
</td></tr>
<tr><td>27<br>F27</td><td>
MUA to MUB-Side Flag<br>
Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>
Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>
When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1
</td></tr>
<tr><td>26<br>F26</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1
</td></tr>
<tr><td>25<br>F25</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1
</td></tr>
<tr><td>24<br>F24</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1</td></tr>
<tr><td>23<br>F23</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1</td></tr>
<tr><td>22</td><td>MUA to MUB-Side Flag</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>F22</td><td>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr></table><br>0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1</td></tr>
<tr><td>21<br>F21</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr></table><br>0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1</td></tr>
<tr><td>20<br>F20</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1
</td></tr>
<tr><td>19<br>F19</td><td>
MUA to MUB-Side Flag<br>
Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>
Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>
When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1
</td></tr>
<tr><td>18<br>F18</td><td>
MUA to MUB-Side Flag<br>
Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>
Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>
When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1
</td></tr>
<tr><td>17<br>F17</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1
</td></tr>
<tr><td>16<br>F16</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1
</td></tr>
<tr><td>15<br>F15</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1</td></tr>
<tr><td>14<br>F14</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1</td></tr>
<tr><td>13</td><td>MUA to MUB-Side Flag</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>F13</td><td>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr></table><br>0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1</td></tr>
<tr><td>12<br>F12</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr></table><br>0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1</td></tr>
<tr><td>11<br>F11</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1
</td></tr>
<tr><td>10<br>F10</td><td>
MUA to MUB-Side Flag<br>
Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>
Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>
When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1
</td></tr>
<tr><td>9<br>F9</td><td>
MUA to MUB-Side Flag<br>
Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>
Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>
When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1
</td></tr>
<tr><td>8<br>F8</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1
</td></tr>
<tr><td>7<br>F7</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1
</td></tr>
<tr><td>6<br>F6</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1</td></tr>
<tr><td>5<br>F5</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr>
</table>
0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1</td></tr>
<tr><td>4</td><td>MUA to MUB-Side Flag</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>F4</td><td>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr></table><br>0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1</td></tr>
<tr><td>3<br>F3</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>FSR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>FSR</td></tr></table><br>0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1</td></tr>
<tr><td>2<br>F2</td><td>MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[Fn], where n = 0 to 31.<br>Fn is the MUB-side flag configured by the values written to MUA_FCR[Fn].<br>When MUA_FCR[Fn] is written to, the write event updates MUB_FSR[Fn], after the event update latency.<br>0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 1<br>F1 | MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[F*n*], where *n* = 0 to 31.<br>F*n* is the MUB-side flag configured by the values written to MUA_FCR[F*n*].<br>When MUA_FCR[F*n*] is written to, the write event updates MUB_FSR[F*n*], after the event update latency.<br>0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1 |
| 0<br>F0 | MUA to MUB-Side Flag<br>Contains flags configured by the values written to MUA_FCR[F*n*], where *n* = 0 to 31.<br>F*n* is the MUB-side flag configured by the values written to MUA_FCR[F*n*].<br>When MUA_FCR[F*n*] is written to, the write event updates MUB_FSR[F*n*], after the event update latency.<br>0b - MUA_FCR[Fn] = 0<br>1b - MUA_FCR[Fn] = 1 |

### 39.7.2.9 General-Purpose Interrupt Enable (GIER)

#### Offset

| Register | Offset |
|---|---|
| GIER | 110h |

#### Function

Contains the MUB general-purpose interrupt enable fields.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GIE31 | GIE30 | GIE29 | GIE28 | GIE27 | GIE26 | GIE25 | GIE24 | GIE23 | GIE22 | GIE21 | GIE20 | GIE19 | GIE18 | GIE17 | GIE16 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GIE15 | GIE14 | GIE13 | GIE12 | GIE11 | GIE10 | GIE9 | GIE8 | GIE7 | GIE6 | GIE5 | GIE4 | GIE3 | GIE2 | GIE1 | GIE0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>31<br>GIE31</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr></table><br>0b - Disable<br>1b - Enable</td></tr>
<tr><td>30<br>GIE30</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr></table><br>0b - Disable<br>1b - Enable</td></tr>
</table>

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>29<br>GIE29</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr></table><br>0b - Disable<br>1b - Enable</td></tr>
<tr><td>28<br>GIE28</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr></table><br>0b - Disable<br>1b - Enable</td></tr>
<tr><td>27<br>GIE27</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr></table><br>0b - Disable<br>1b - Enable</td></tr>
<tr><td>26<br>GIE26</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr></table><br>0b - Disable<br>1b - Enable</td></tr>
<tr><td>25<br>GIE25</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr>
</table>
0b - Disable<br>1b - Enable
</td></tr>
<tr><td>24<br>GIE24</td><td>
MUB General-purpose Interrupt Enable<br>
Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>
When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>
If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>
GIEn becomes 0 when MU resets.<br>
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr>
</table>
0b - Disable<br>1b - Enable
</td></tr>
<tr><td>23<br>GIE23</td><td>
MUB General-purpose Interrupt Enable<br>
Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>
When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>
If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>
GIEn becomes 0 when MU resets.<br>
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr>
</table>
0b - Disable<br>1b - Enable
</td></tr>
<tr><td>22<br>GIE22</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr>
</table>
0b - Disable<br>1b - Enable
</td></tr>
<tr><td>21<br>GIE21</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr>
</table>
0b - Disable<br>1b - Enable
</td></tr>
<tr><td>20<br>GIE20</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr>
</table>
0b - Disable<br>1b - Enable
</td></tr>
<tr><td>19<br>GIE19</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr>
</table>
0b - Disable<br>1b - Enable
</td></tr>
<tr><td>18<br>GIE18</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr>
</table>
0b - Disable<br>1b - Enable
</td></tr>
<tr><td>17<br>GIE17</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | <table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr></table><br>0b - Disable<br>1b - Enable |
| 16<br>GIE16 | MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (*n* = 0 to 31).<br>When GIE*n* = 1, a general-purpose interrupt *n* request is issued to processor B when MUB GSR[GIP*n*] = 1.<br>If GIE*n* = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIE*n* becomes 0 when MU resets.<br>**NOTE**<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr></table><br>0b - Disable<br>1b - Enable |
| 15<br>GIE15 | MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (*n* = 0 to 31).<br>When GIE*n* = 1, a general-purpose interrupt *n* request is issued to processor B when MUB GSR[GIP*n*] = 1.<br>If GIE*n* = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIE*n* becomes 0 when MU resets.<br>**NOTE**<br>This field is not supported in every instance. The following table includes only supported registers. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | <table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr></table><br>0b - Disable<br>1b - Enable |
| 14<br>GIE14 | MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (*n* = 0 to 31).<br>When GIE*n* = 1, a general-purpose interrupt *n* request is issued to processor B when MUB GSR[GIP*n*] = 1.<br>If GIE*n* = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIE*n* becomes 0 when MU resets.<br>**NOTE**<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr></table><br>0b - Disable<br>1b - Enable |
| 13<br>GIE13 | MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (*n* = 0 to 31).<br>When GIE*n* = 1, a general-purpose interrupt *n* request is issued to processor B when MUB GSR[GIP*n*] = 1.<br>If GIE*n* = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIE*n* becomes 0 when MU resets.<br>**NOTE**<br>This field is not supported in every instance. The following table includes only supported registers. |

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr>
</table>
0b - Disable<br>1b - Enable
</td></tr>
<tr><td>12<br>GIE12</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr>
</table>
0b - Disable<br>1b - Enable
</td></tr>
<tr><td>11<br>GIE11</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr>
</table>
0b - Disable<br>1b - Enable
</td></tr>
<tr><td>10<br>GIE10</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr>
</table>
0b - Disable<br>1b - Enable
</td></tr>
<tr><td>9<br>GIE9</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr>
</table>
0b - Disable<br>1b - Enable</td></tr>
<tr><td>8<br>GIE8</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr>
</table>
0b - Disable<br>1b - Enable</td></tr>
<tr><td>7<br>GIE7</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr>
</table>
0b - Disable<br>1b - Enable</td></tr>
<tr><td>6<br>GIE6</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr>
</table>
0b - Disable<br>1b - Enable</td></tr>
<tr><td>5<br>GIE5</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr>
</table>
0b - Disable<br>1b - Enable</td></tr>
<tr><td>4<br>GIE4</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr>
</table>
0b - Disable<br>1b - Enable</td></tr>
<tr><td>3<br>GIE3</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr>
</table>
0b - Disable<br>1b - Enable
</td></tr>
<tr><td>2<br>GIE2</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr>
</table>
0b - Disable<br>1b - Enable
</td></tr>
<tr><td>1<br>GIE1</td><td>MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (n = 0 to 31).<br>When GIEn = 1, a general-purpose interrupt n request is issued to processor B when MUB GSR[GIPn] = 1.<br>If GIEn = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIEn becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | <table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GIER</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GIER</td></tr></table><br>0b - Disable<br>1b - Enable |
| 0<br>GIE0 | MUB General-purpose Interrupt Enable<br>Enables general-purpose interrupt. There are 32 general-purpose interrupts (*n* = 0 to 31).<br>When GIE*n* = 1, a general-purpose interrupt *n* request is issued to processor B when MUB GSR[GIP*n*] = 1.<br>If GIE*n* = 0, the general-purpose interrupt request pending does not trigger the general-purpose interrupt.<br>GIE*n* becomes 0 when MU resets.<br>0b - Disable<br>1b - Enable |

### 39.7.2.10 General-Purpose Control (GCR)

#### Offset

| Register | Offset |
|---|---|
| GCR | 114h |

#### Function

Contains the MUB general-purpose interrupt request fields.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GIR31 | GIR30 | GIR29 | GIR28 | GIR27 | GIR26 | GIR25 | GIR24 | GIR23 | GIR22 | GIR21 | GIR20 | GIR19 | GIR18 | GIR17 | GIR16 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | GIR15 | GIR14 | GIR13 | GIR12 | GIR11 | GIR10 | GIR9 | GIR8 | GIR7 | GIR6 | GIR5 | GIR4 | GIR3 | GIR2 | GIR1 | GIR0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>31<br>GIR31</td><td>MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr></table><br>0b - Not requested<br>1b - Requested</td></tr>
<tr><td>30<br>GIR30</td><td>MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr></table></td></tr>
</table>

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
</table>
0b - Not requested<br>1b - Requested
</td></tr>
<tr><td>29<br>GIR29</td><td>MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr>
</table>
0b - Not requested<br>1b - Requested
</td></tr>
<tr><td>28<br>GIR28</td><td>MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr>
</table>
0b - Not requested<br>1b - Requested
</td></tr>
<tr><td>27<br>GIR27</td><td>MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr>
</table>
0b - Not requested<br>1b - Requested
</td></tr>
<tr><td>26<br>GIR26</td><td>MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr>
</table>
0b - Not requested<br>1b - Requested
</td></tr>
<tr><td>25<br>GIR25</td><td>
MUB General-Purpose Interrupt Request<br>
Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>
Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>
This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>
To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>
This field becomes 0 when MU resets.
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr>
</table>
0b - Not requested<br>1b - Requested
</td></tr>
<tr><td>24<br>GIR24</td><td>
MUB General-Purpose Interrupt Request<br>
Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>
Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr></table><br>0b - Not requested<br>1b - Requested</td></tr>
<tr><td>23<br>GIR23</td><td>MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr></table><br>0b - Not requested<br>1b - Requested</td></tr>
<tr><td>22</td><td>MUB General-Purpose Interrupt Request</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>GIR22</td><td>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br><br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br><br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br><br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br><br>This field becomes 0 when MU resets.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><br>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr>
</table>
<br>0b - Not requested<br><br>1b - Requested</td></tr>
<tr><td>21<br><br>GIR21</td><td>MUB General-Purpose Interrupt Request<br><br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br><br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br><br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br><br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br><br>This field becomes 0 when MU resets.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><br>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr>
</table>
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>0b - Not requested<br>1b - Requested</td></tr>
<tr><td>20<br>GIR20</td><td>MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr></table><br>0b - Not requested<br>1b - Requested</td></tr>
<tr><td>19<br>GIR19</td><td>MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr>
</table>
0b - Not requested<br>1b - Requested
</td></tr>
<tr><td>18<br>GIR18</td><td>MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr>
</table>
0b - Not requested<br>1b - Requested
</td></tr>
<tr><td>17<br>GIR17</td><td>MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr>
</table>
0b - Not requested<br>1b - Requested
</td></tr>
<tr><td>16<br>GIR16</td><td>
MUB General-Purpose Interrupt Request<br>
Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>
Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>
This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>
To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>
This field becomes 0 when MU resets.
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr>
</table>
0b - Not requested<br>1b - Requested
</td></tr>
<tr><td>15<br>GIR15</td><td>
MUB General-Purpose Interrupt Request<br>
Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>
Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr></table><br>0b - Not requested<br>1b - Requested</td></tr>
<tr><td>14<br>GIR14</td><td>MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr></table><br>0b - Not requested<br>1b - Requested</td></tr>
<tr><td>13</td><td>MUB General-Purpose Interrupt Request</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>GIR13</td><td>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br><br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br><br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br><br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br><br>This field becomes 0 when MU resets.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr></table><br>0b - Not requested<br><br>1b - Requested</td></tr>
<tr><td>12<br><br>GIR12</td><td>MUB General-Purpose Interrupt Request<br><br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br><br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br><br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br><br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br><br>This field becomes 0 when MU resets.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr></table></td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>0b - Not requested<br>1b - Requested</td></tr>
<tr><td>11<br>GIR11</td><td>MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr></table><br>0b - Not requested<br>1b - Requested</td></tr>
<tr><td>10<br>GIR10</td><td>MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr>
</table>
0b - Not requested<br>1b - Requested
</td></tr>
<tr><td>9<br>GIR9</td><td>MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr>
</table>
0b - Not requested<br>1b - Requested
</td></tr>
<tr><td>8<br>GIR8</td><td>MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr>
</table>
0b - Not requested<br>1b - Requested
</td></tr>
<tr><td>7<br>GIR7</td><td>
MUB General-Purpose Interrupt Request<br>
Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>
Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>
This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>
To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>
This field becomes 0 when MU resets.
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr>
</table>
0b - Not requested<br>1b - Requested
</td></tr>
<tr><td>6<br>GIR6</td><td>
MUB General-Purpose Interrupt Request<br>
Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>
Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr></table><br>0b - Not requested<br>1b - Requested</td></tr>
<tr><td>5<br>GIR5</td><td>MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr></table><br>0b - Not requested<br>1b - Requested</td></tr>
<tr><td>4</td><td>MUB General-Purpose Interrupt Request</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>GIR4</td><td>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr></table><br>0b - Not requested<br>1b - Requested</td></tr>
<tr><td>3<br>GIR3</td><td>MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr></table></td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>0b - Not requested<br>1b - Requested</td></tr>
<tr><td>2<br>GIR2</td><td>MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr></table><br>0b - Not requested<br>1b - Requested</td></tr>
<tr><td>1<br>GIR1</td><td>MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (n = 0 to 31).<br>Writing 1 to GIRn sets MUA_GSR[GIPn]. If MUA_GIER[GIEn] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIPn] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIRn is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | <table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GCR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GCR</td></tr></table><br>0b - Not requested<br>1b - Requested |
| 0<br>GIR0 | MUB General-Purpose Interrupt Request<br>Specifies whether general-purpose interrupts are requested to MUA. There are 32 general-purpose interrupts (*n* = 0 to 31).<br>Writing 1 to GIR*n* sets MUA_GSR[GIP*n*]. If MUA_GIER[GIE*n*] = 1, a general-purpose interrupt request is triggered on processor A.<br>This field becomes 0 when MUA_GSR[GIP*n*] is cleared. This clearing informs MUB that the interrupt is accepted (cleared by software).<br>To ensure proper operations, verify that GIR*n* is 0 (no pending interrupt) before writing 1 to it.<br>This field becomes 0 when MU resets.<br>0b - Not requested<br>1b - Requested |

### 39.7.2.11 General-purpose Status (GSR)

**Offset**

| Register | Offset |
|---|---|
| GSR | 118h |

**Function**

Contains the status of the MUB general-purpose interrupt pending requests.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | GIP31 | GIP30 | GIP29 | GIP28 | GIP27 | GIP26 | GIP25 | GIP24 | GIP23 | GIP22 | GIP21 | GIP20 | GIP19 | GIP18 | GIP17 | GIP16 |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | GIP15 | GIP14 | GIP13 | GIP12 | GIP11 | GIP10 | GIP9 | GIP8 | GIP7 | GIP6 | GIP5 | GIP4 | GIP3 | GIP2 | GIP1 | GIP0 |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>31<br>GIP31</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.<br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
NOTE<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>30</td><td>MUB General-Purpose Interrupt Request Pending</td></tr>
</table>

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>GIP30</td><td>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br><br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br><br>GIPn is cleared when MU resets.<br><br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><br>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
<br>NOTE<br>This field behaves differently for register reads and writes.<br><br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>29<br>GIP29</td><td>MUB General-Purpose Interrupt Request Pending<br><br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br><br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br><br>GIPn is cleared when MU resets.<br><br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
NOTE<br>This field behaves differently for register reads and writes.<br><br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>28<br>GIP28</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.<br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
NOTE<br>This field behaves differently for register reads and writes.<br><br>When reading<br>0b - Not pending</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>27<br>GIP27</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.<br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr></table><br>NOTE<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>26<br>GIP26</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><br>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
<br>NOTE<br>This field behaves differently for register reads and writes.<br><br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>25<br>GIP25</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.<br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><br>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>NOTE<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>24<br>GIP24</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.<br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr></table><br>NOTE<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>23<br>GIP23</td><td>MUB General-Purpose Interrupt Request Pending</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br><br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br><br>GIPn is cleared when MU resets.<br><br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><br>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
<br>NOTE<br>This field behaves differently for register reads and writes.<br><br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>22<br>GIP22</td><td>MUB General-Purpose Interrupt Request Pending<br><br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br><br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br><br>GIPn is cleared when MU resets.<br><br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
NOTE<br>This field behaves differently for register reads and writes.<br><br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>21<br>GIP21</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.<br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
NOTE<br>This field behaves differently for register reads and writes.<br><br>When reading<br>0b - Not pending</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>20<br>GIP20</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.<br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr></table><br>NOTE<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>19<br>GIP19</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><br>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
<br>NOTE<br>This field behaves differently for register reads and writes.<br><br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>18<br>GIP18</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.<br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><br>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>NOTE<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>17<br>GIP17</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.<br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr></table><br>NOTE<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>16<br>GIP16</td><td>MUB General-Purpose Interrupt Request Pending</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br><br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br><br>GIPn is cleared when MU resets.<br><br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><br>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
<br>NOTE<br>This field behaves differently for register reads and writes.<br><br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>15<br>GIP15</td><td>MUB General-Purpose Interrupt Request Pending<br><br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br><br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br><br>GIPn is cleared when MU resets.<br><br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
NOTE<br>This field behaves differently for register reads and writes.<br><br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>14<br>GIP14</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.<br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
NOTE<br>This field behaves differently for register reads and writes.<br><br>When reading<br>0b - Not pending</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>13<br>GIP13</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.<br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr></table><br>NOTE<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>12<br>GIP12</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><br>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
<br>NOTE<br>This field behaves differently for register reads and writes.<br><br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>11<br>GIP11</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.<br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><br>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>NOTE<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>10<br>GIP10</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.<br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr></table><br>NOTE<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>9<br>GIP9</td><td>MUB General-Purpose Interrupt Request Pending</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br><br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br><br>GIPn is cleared when MU resets.<br><br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><br>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
<br>NOTE<br>This field behaves differently for register reads and writes.<br><br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>8<br>GIP8</td><td>MUB General-Purpose Interrupt Request Pending<br><br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br><br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br><br>GIPn is cleared when MU resets.<br><br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
NOTE<br>This field behaves differently for register reads and writes.<br><br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>7<br>GIP7</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.<br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
NOTE<br>This field behaves differently for register reads and writes.<br><br>When reading<br>0b - Not pending</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>6<br>GIP6</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.<br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr><tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr></table><br>NOTE<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>5<br>GIP5</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><br>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
<br>NOTE<br>This field behaves differently for register reads and writes.<br><br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>4<br>GIP4</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.<br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><br>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>NOTE<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>3<br>GIP3</td><td>MUB General-Purpose Interrupt Request Pending<br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br>GIPn is cleared when MU resets.<br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
NOTE<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>2<br>GIP2</td><td>MUB General-Purpose Interrupt Request Pending</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br><br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br><br>GIPn is cleared when MU resets.<br><br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><br>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
<br>NOTE<br>This field behaves differently for register reads and writes.<br><br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>1<br>GIP1</td><td>MUB General-Purpose Interrupt Request Pending<br><br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br><br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br><br>GIPn is cleared when MU resets.<br><br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br><br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>MU_0.MUB</td><td>GSR</td><td>—</td></tr>
<tr><td>MU_1.MUB</td><td>—</td><td>GSR</td></tr>
</table>
NOTE<br>This field behaves differently for register reads and writes.<br><br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
<tr><td>0<br>GIP0</td><td>MUB General-Purpose Interrupt Request Pending<br><br>Indicates whether a general-purpose interrupt request is pending. There are 32 general-purpose interrupts (n = 0 to 31).<br><br>GIPn informs MUB that MUA_GCR[GIRn] changed from 0 to 1. If MUB_GIER[GIEn] = 1, a general-purpose interrupt request is issued to processor B.<br><br>GIPn is cleared when MU resets.<br><br>After GIPn is cleared, if MUB_GIER[GIEn] = 1, the general-purpose interrupt request is cleared on the MUB side.<br><br>NOTE<br>This field behaves differently for register reads and writes.<br><br>When reading<br>0b - Not pending<br>1b - Pending<br>When writing<br>0b - No effect<br>1b - Clear the flag</td></tr>
</table>

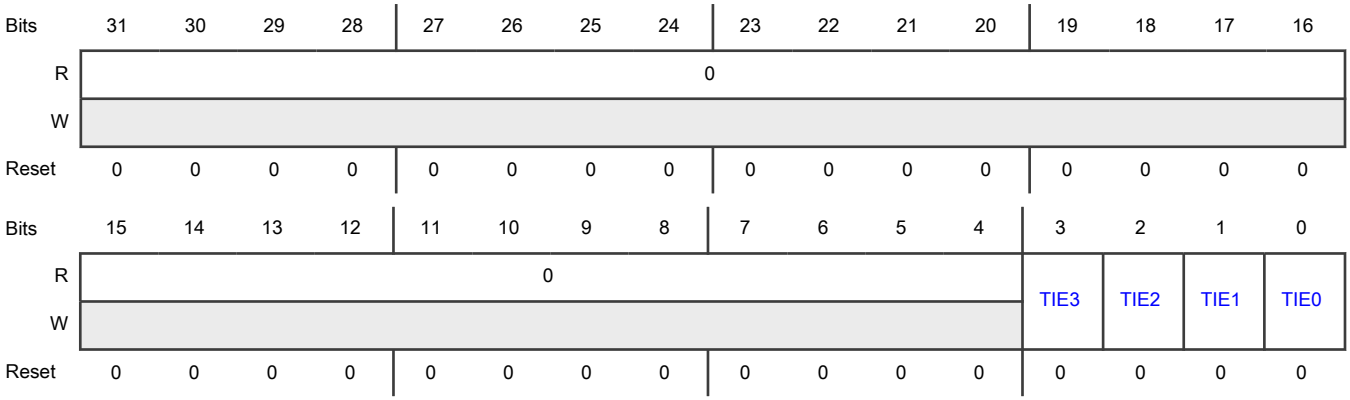
### 39.7.2.12 Transmit Control (TCR)

#### Offset

| Register | Offset |
|---|---|
| TCR | 120h |

#### Function
Contains the MUB transmit interrupt enable fields.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>TIEn | MUB Transmit Interrupt Enable<br>Enables MUB transmit interrupt *n*, where *n* = 0 to 3.<br>If this field is 1, an MUB transmit interrupt *n* request is issued when MUB_TSR[TE*n*] is set.<br>If this field is 0, MU ignores the value of MUB_TSR[TE*n*], and no MUB transmit interrupt *n* request is issued.<br>0b - Disable<br>1b - Enable |

### 39.7.2.13 Transmit Status (TSR)

#### Offset

| Register | Offset |
|---|---|
| TSR | 124h |

### Function

Indicates whether the MUB transmit registers are empty.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | TE3 | TE2 | TE1 | TE0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>TEn | MUB Transmit Empty<br>Indicates whether MUB Transmit (TR*n*) register is empty, where *n* = 0 to 3.<br>This field becomes 1 after the MUA_RR*n* register is read on the MUA side. When TE*n* = 1, it indicates to the MUB side that the MUB_TR*n* register is ready to be written on the MUB side. If MUB_TCR[TIE*n*] = 1, a transmit *n* interrupt is issued on the MUB side.<br>This field becomes 0 after the MUB_TR*n* register is written to on the MUB side. After this field becomes 0, if MUB_TCR[TIE*n*] = 1, the transmit *n* interrupt request is cleared on the MUB side.<br>This field becomes 1 when MU resets.<br>0b - Not empty<br>1b - Empty |

## 39.7.2.14 Receive Control (RCR)

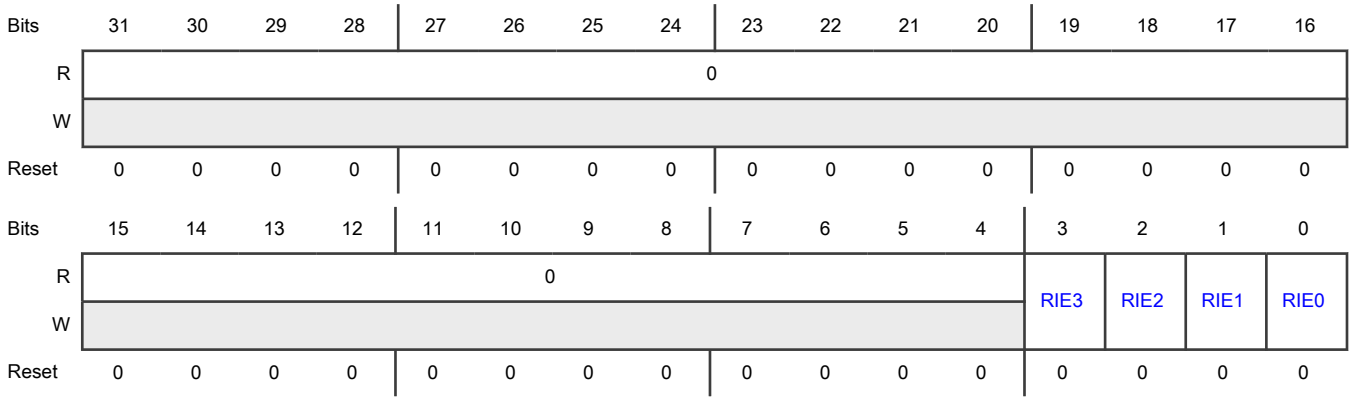
### Offset

| Register | Offset |
|---|---|
| RCR | 128h |

### Function

Contains the MUB receive interrupt enables.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>RIEn | MUB Receive Interrupt Enable<br>Enables MUB receive interrupt *n*, where *n* = 0 to 3.<br>If this field is 1, an MUB receive interrupt *n* request is issued when MUB_RSR[RF*n*] is set.<br>If this field is 0, MU ignores the value of MUB_RSR[RF*n*], and no MUB receive interrupt request is issued.<br>0b - Disable<br>1b - Enable |

## 39.7.2.15 Receive Status (RSR)

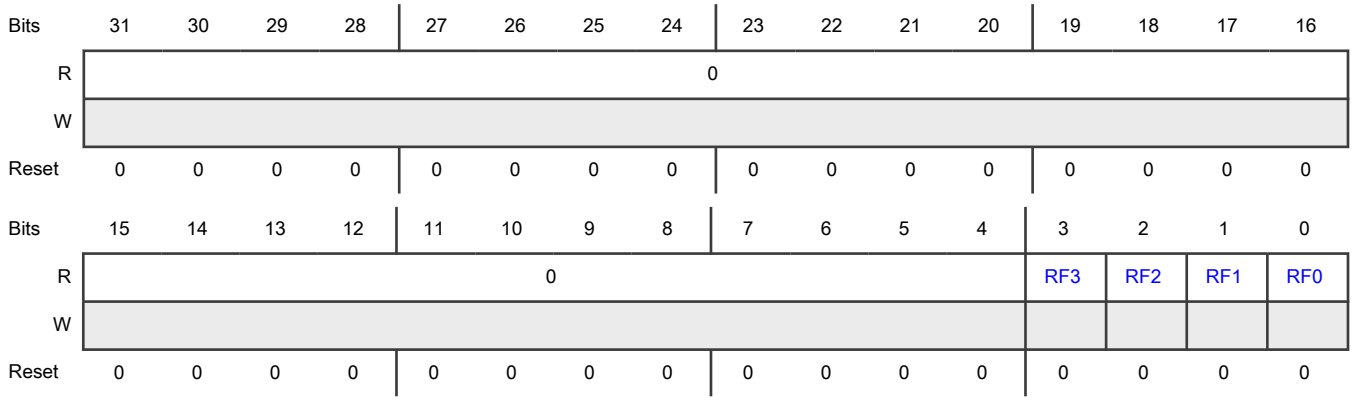
### Offset

| Register | Offset |
|---|---|
| RSR | 12Ch |

### Function

Indicates whether the MUB receive registers are full.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>RFn | MUB Receive Register Full<br>Indicates whether MUB Receive register (RR*n*) is full, where *n* = 0 to 3.<br>RF*n* becomes 1 when the MUA_TR*n* register is written to on the MUA side.<br>When RF*n* is 1, it indicates to the MUB side that new data in the MUB_RR*n* register is ready for MUB to read it. If MUB_RCR[RIE*n*] = 1, a receive *n* interrupt is issued on the MUB side.<br>RF*n* becomes 0 when the MUB_RR*n* register is read, or when MU is reset.<br>After RF*n* becomes 0, if MUB_RCR[RIE*n*] = 1, the receive *n* interrupt request is cleared on the MUB side.<br>0b - Not full<br>1b - Full |

## 39.7.2.16 Transmit (TR0 - TR3)

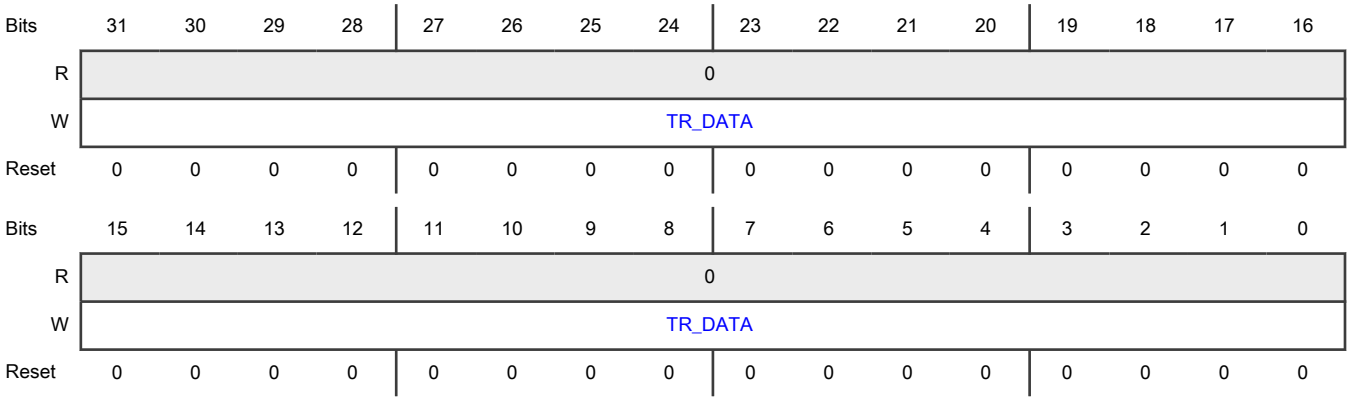
### Offset

| Register | Offset |
|---|---|
| TR0 | 200h |
| TR1 | 204h |
| TR2 | 208h |
| TR3 | 20Ch |

### Function

Contains MUB transmit data.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>TR_DATA | MUB Transmit Data<br>Contains MUB transmit data. MUA_RR*n* reflects the data written to this register.<br>The TR*n* and RR*n* registers are not double-buffered. Writing to MUB_TR*n* overrides the data readable in the MUA_RR*n* register.<br>A write to the Transmit register clears MUB_TSR[TE*n*] on the transmitter side, and sets MUA_RSR[RF*n*] on the receiver side.<br>You can write to this register only when MUB_TSR[TE*n*] = 1.<br>Reading this register returns all zeroes. |

## 39.7.2.17 Receive (RR0 - RR3)

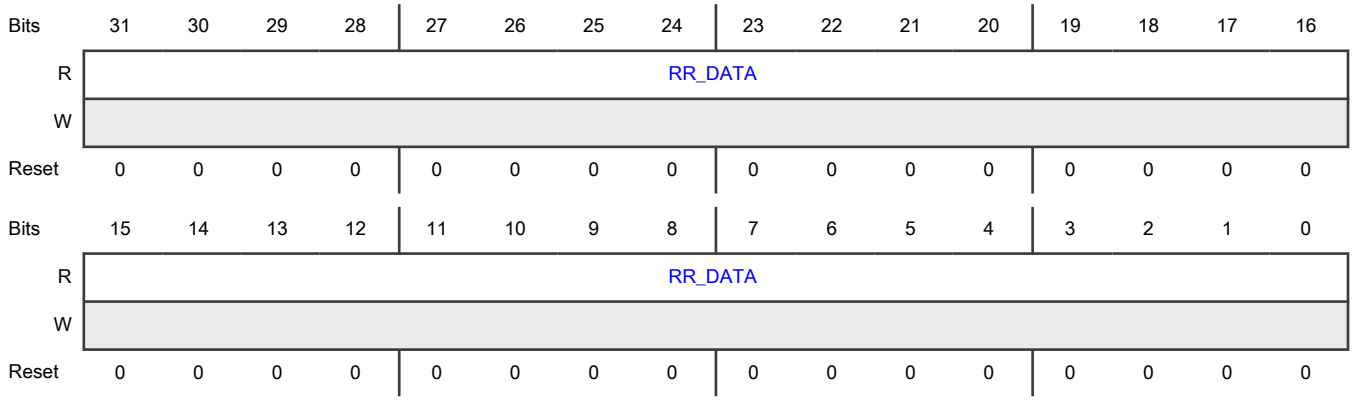
### Offset

| Register | Offset |
|---|---|
| RR0 | 280h |
| RR1 | 284h |
| RR2 | 288h |
| RR3 | 28Ch |

### Function

Contains MUB receive data.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>RR_DATA | MUB Receive Data<br>Reflects the data written to MUA TR*n*.<br>Reading this register clears MUB_RSR[RF*n*] on the receiver side, and sets MUA_TSR[TE*n*] on the transmitter side.<br>You can read this register only when MUB_RSR[RF*n*] = 1. Reading it before MUB_RSR[RF*n*] becomes 1 may result in reading incorrect data. Poll MUB_RSR[RF*n*] to confirm it is set before reading RR*n*.<br>Writing to this register generates an error response to MUB. |

## 39.8 Glossary

| | |
|---|---|
| **EP** | Event Pending |
| **GIR** | General-purpose Interrupt Request |
| **GIP** | General-purpose Interrupt Pending |
| **MUR** | Messaging Unit Reset |
| **RF** | Receiver Full |
| **RFP** | Receive Full Pending |
| **TE** | Transmitter Empty |
| **TEP** | Transmit Empty Pending |
| **MUA** | Messaging Unit A |
| **MUB** | Messaging Unit B |

# Chapter 40
# Power Management

## 40.1 Introduction

The power management system generates, monitors, and controls power supplies and related resets. This chapter describes the system's interaction with other peripherals.

The power management system includes the following modules:

**Table 208. Power management system modules and their functions**

| Part | Function |
|---|---|
| MC_ME | Initiates entry into Low-Power mode |
| MC_PCU | Controls entry into and exit from Low-Power mode |
| PMC | Regulates and monitors power supplies of the Run and the Standby domains |
| MC_RGM | Ensures a clean state and Run domain sanity by controlling the reset sequence |
| FIRC, WKPU, and other chip peripherals | Controls Standby mode wake-up and operations in Run and Standby modes |

Power management system for MCXE315/MCXE316 and MCXE317 represent block diagrams of the power management system for the MCXE31 family.

### 40.1.1 Power management system (for MCXE31B)

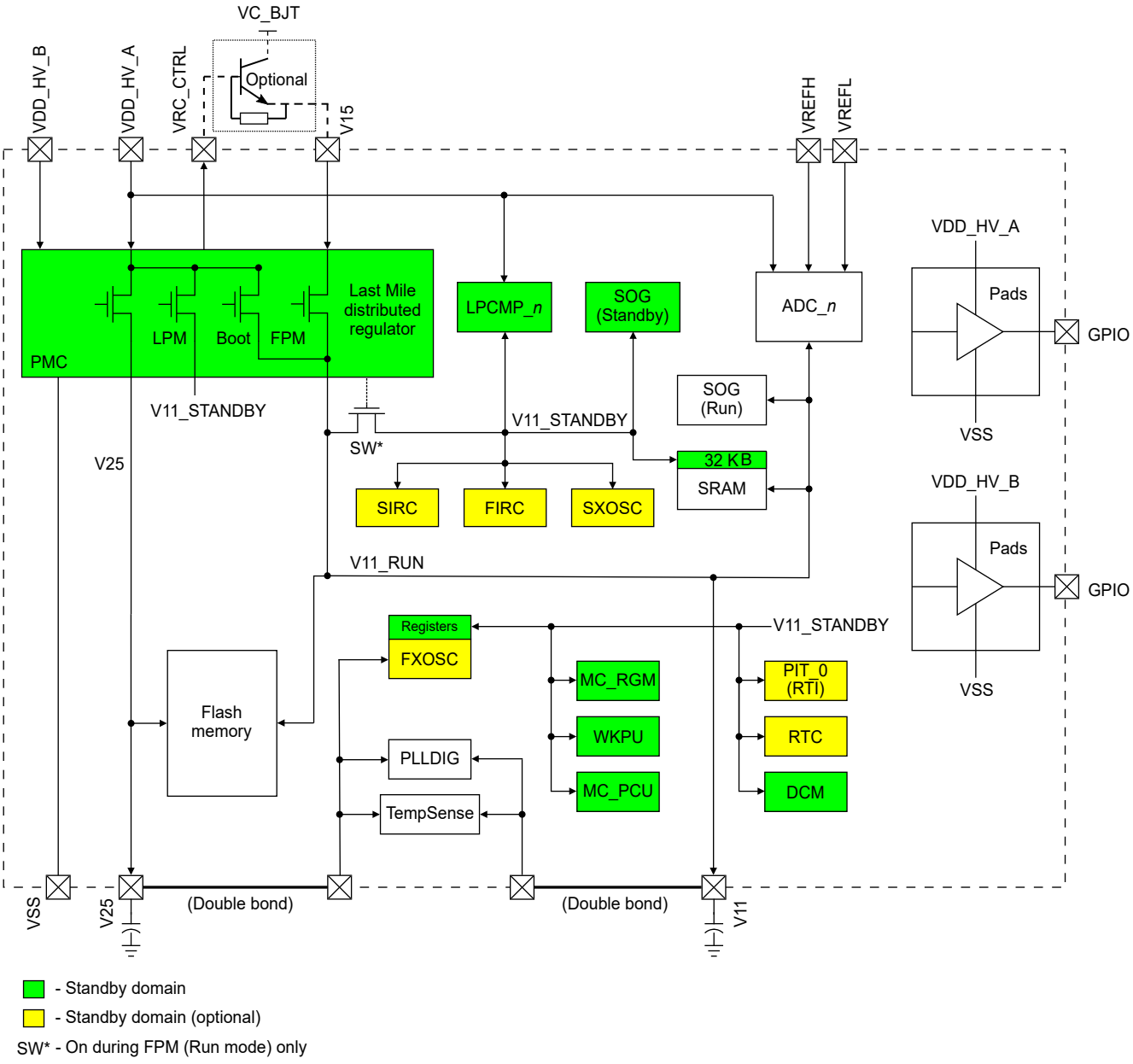



Figure 147. Power management system for MCXE31B

### 40.1.2 Power management system for MCXE315/MCXE316 and MCXE317

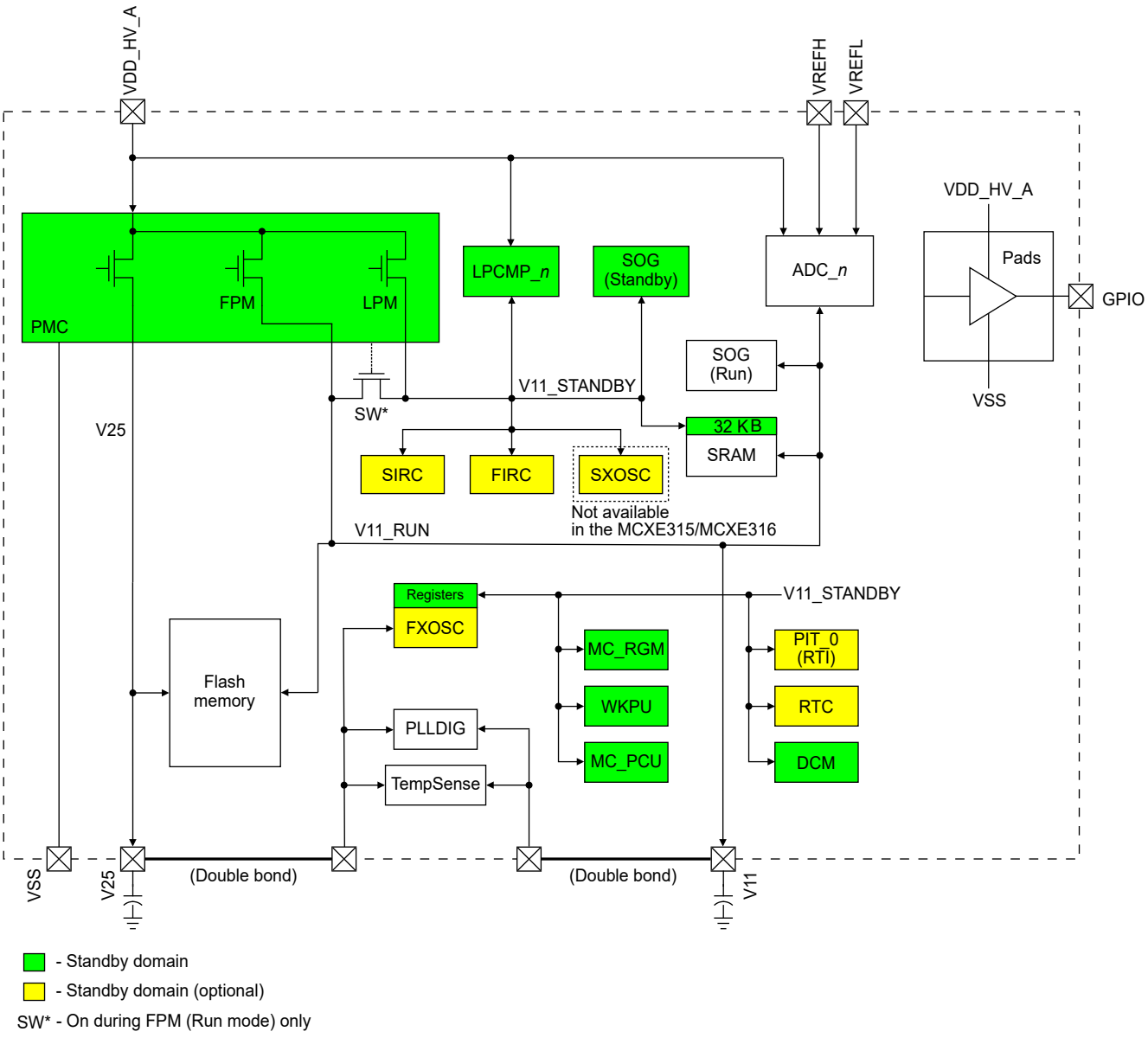



**Figure 148. Power management system for MCXE315/MCXE316 and MCXE317**

### 40.1.3 Features

- For MCXE315, MCXE316, MCXE317, PMC uses VDD_HV_A to generate 1.1 V (nominal) supply for the core logic (V11_RUN)
- For MCXE31B, PMC uses a 1.5 V supply to generate 1.1 V (nominal) supply for the core logic (V11_RUN)
- MCXE31B includes a linear regulator to use an optional external ballast (BJT) and VRC_CTRL output for 1.5 V generation.
- Supports a low-power regulator (LPM) supplying core logic during Standby mode (V11_STANDBY)
- MCXE31B includes 1.1 V FPM regulator options for driving the logic in RUN mode (V11_RUN) from a 1.5V supply:

  - Boot LDO
  - Internal Last-mile
- Supports power switches to isolate voltage islands and configure the chip in Standby mode
- Includes a linear regulator that generates a 2.5 V supply (V25) from VDD_HV_A
- Supports voltage monitors ensuring transitions to a safe state (POR) when a supply is out of a valid range
- Controls power-mode transitions through an interaction with the digital interface
- Offers a padkeeping feature that retains PAD state during standby mode till software boots up
- Separates ADC reference supplies (VREFH and VREFL)

### 40.1.4 Operational power modes

This chip uses two modes:

- Run mode (FPM): Main operation mode having full-chip performance and a higher current consumption as compared to Standby mode.
- Standby mode (LPM): Low-performance mode of the chip in which the Run domain is turned off. Most of the cores and peripherals turn off in this mode.

The boot regulator manages the chip during the booting process except for MCXE315, MCXE316, and MCXE317.

The last-mile regulator is the full-performance regulator, which you enable for running applications. The upcoming sections discuss the sequence to enable or disable the regulator.

The LPM regulator manages the chip in Standby mode.

### 40.1.5 External 1.5 V source (for MCXE31B)

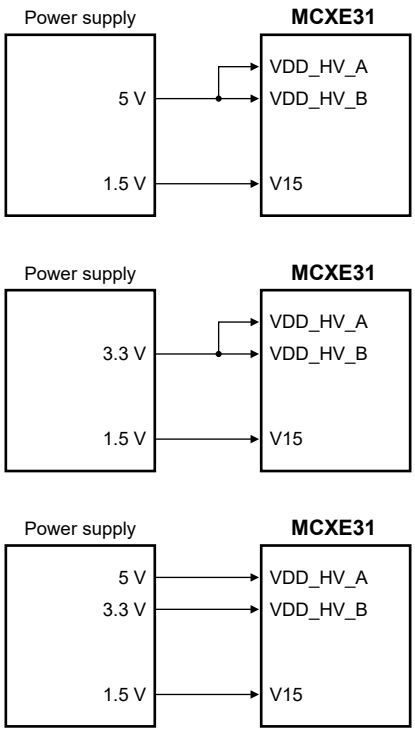



Figure 149. Core supply from external 1.5 V source for MCXE31B

### 40.1.6 Using a BJT for 1.5 V generation (for MCXE31B)

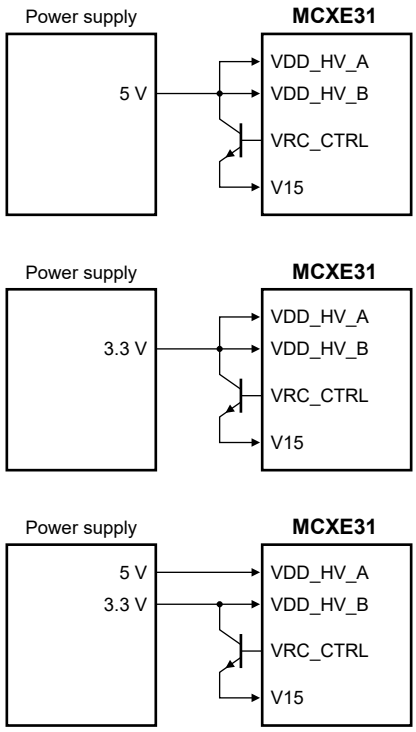



**Figure 150. Using a BJT for 1.5 V generation for MCXE31B**

## 40.2 Power-up sequence

Figure 151 shows the reset sequence for a power-up or Standby mode event. This sequence starts from the POR phase. For the Run domain logic, Standby mode exit operation is same as power-up.

See the Reset Overview chapter for more information about the chip reset sequence.

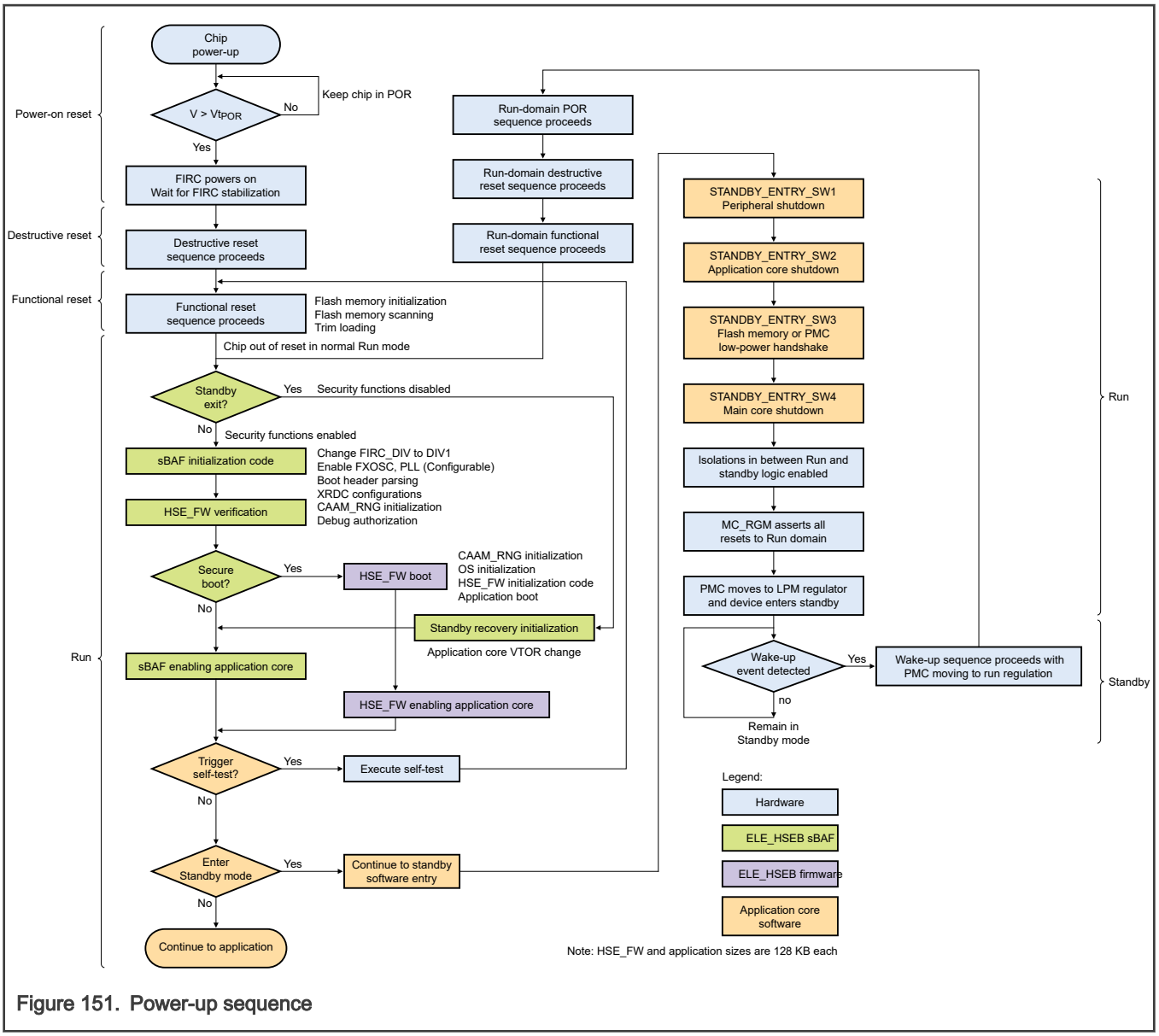



**Figure 151. Power-up sequence**

## 40.3 PMC last-mile regulator auto-enable feature (for MCXE31B)

PMC includes an automatic last-mile auto-enable feature. After starting on boot regulator, this feature allows automatic switch over to last-mile regulator if 1.5 V is present during:

- Chip startup
- Standby mode recovery

You can control the PMC last-mile regulator by appropriately configuring:

- PMC's CONFIG[LMAUTOEN] field
- PMC's CONFIG[LM_EN] field

  For more information, see the descriptions of these fields in PMC's CONFIG register.

**NOTE**

You must write 1 to PMC's CONFIG[LM_EN] field before transtioning to faster clock frequencies irrespective of the setting of PMC's CONFIG[LMAUTOEN] field. This is because of the reduced clock speed when using the boot regulator.

### 40.3.1 Last-mile regulator with 1.5 V from an external source

**Table 209. Last-mile regulator with 1.5 V from an external source**

| Operating condition | Last-mile regulator operation |
|---|---|
| After POR | Boots on boot regulator and then automatically switches to the last-mile regulator |
| After destructive reset | Remains on last-mile regulator<br>**NOTE**<br>If the destructive reset source is low voltage reset on 1.1 V then switchback happens from last-mile to boot regulator for boot. |
| After functional reset | Remains on last-mile regulator |
| After PMC's LVSC[LVD15S] field becomes 1 | Switches, automatically, to the boot regulator to configure clocks for slow speed |

### 40.3.2 Last-mile regulator using a BJT

**Table 210. Last-mile regulator using a BJT**

| Operating condition | Last-mile regulator operation |
|---|---|
| After POR | Boots on boot regulator; switches to the last-mile regulator post reset when the Cortex-M7 core configures the software (post-secure boot) on FIRC (100 ms) |
| After destructive reset | Switches to boot regulator to check reset propagation delay |
| After functional reset | Remains on last-mile regulator |
| After PMC's LVSC[LVD15S] field becomes 1 | Switches, automatically, to boot regulator to configure clocks for slow speed |

## 40.4 Standby mode entry sequence

The Standby mode entry sequence includes three phases of operation:

- Standby mode entry configuration phase or software Standby mode entry sequence
- Standby mode entry handshake phase or hardware Standby mode entry sequence
- Standby mode entry or PMC Standby mode entry

**NOTE**

The Standby mode entry sequence described in this section is the only supported sequence. Contact NXP support if you require an alternate Standby mode entry sequence.

### 40.4.1 Software Standby mode entry sequence

The Standby mode entry sequence discusses chip configuration described in the sections that follow.

Before entering the software Standby mode sequence, the system clock source must be changed to FIRC at 48 MHz because PLLDIG is not available in Standby mode. In this mode, all clock sources can be optionally disabled (including FIRC, which results

in a no-clock, low-power consumption mode). You could use FXOSC, if enabled, when the 2.5 V supply is available by appropriate configuration of PMC's CONFIG[LPM25EN].

The software Standby mode entry sequence consists of four steps:

1. SW1: Module shutdown process
2. SW2: Application core shutdown process
3. SW3: Flash memory low-power handshake and PMC last-mile regulator control
4. SW4: Main core shutdown process

These processes are described in detail in the sections that follow. See Figure 151 that shows the relationship between these four steps in a flow diagram.

#### 40.4.1.1 SW1: Module shutdown process

I/O and module configuration for Standby mode discusses the procedure to configure I/O and the chip modules for Standby mode.

The entry sequence for this mode includes module clock disabling steps (see the "Clocking" chapter for module clock turn on and turn off processes). You must use MC_ME's PRTN*n*_COFB*m*_CLKEN[REQ*p*] fields to enable or disable module clocks.

##### 40.4.1.1.1 Disabling modules

Disable modules by configuring the appropriate fields in their registers for Standby mode operation. See specific module chapters for more information.

The Standby mode entry sequence includes the module clock disabling step, with which you can disable the modules that you do not need for Standby mode operation.

The sequence of disabling modules is shown in I/O and module configuration for Standby mode.

**NOTE**

While enabling or disabling the modules, you must verify that the module is disabled when you read MC_ME's PRTNn_COFBm_STAT register and the module disable field, if applicable. In case of a discrepancy, you must perform proper diagnostic steps.

You must clear the I/O controls for the pads that you do not require in Standby mode (OBE, IBE, and so on). This avoids any unwanted pad keeping settings. See Pad keeping for more information on the chip pad keeping process.

For any standby wake source, if an interrupt occurs it must be disabled before entering into standby and only the wake up event of that source must be enabled. This is to avoid any SW conflict in the interrupt handling for multi application core cases.

The MC_ME's PRTN*n*_COFB*m*_STAT register indicates the status of peripheral clock enable or disable. It may take up to three clock cycles for MC_ME's PRTN*n*_COFB*m*_STAT register to update after MC_ME's PRTN*n*_COFB*m*_CLKEN register is updated.

Once modules are disabled by following above steps:

- Switch to FIRC as the system clock
- Disable FXOSC and wait for clock status
- Configure standby entry

#### 40.4.1.1.2 I/O and module configuration for Standby mode

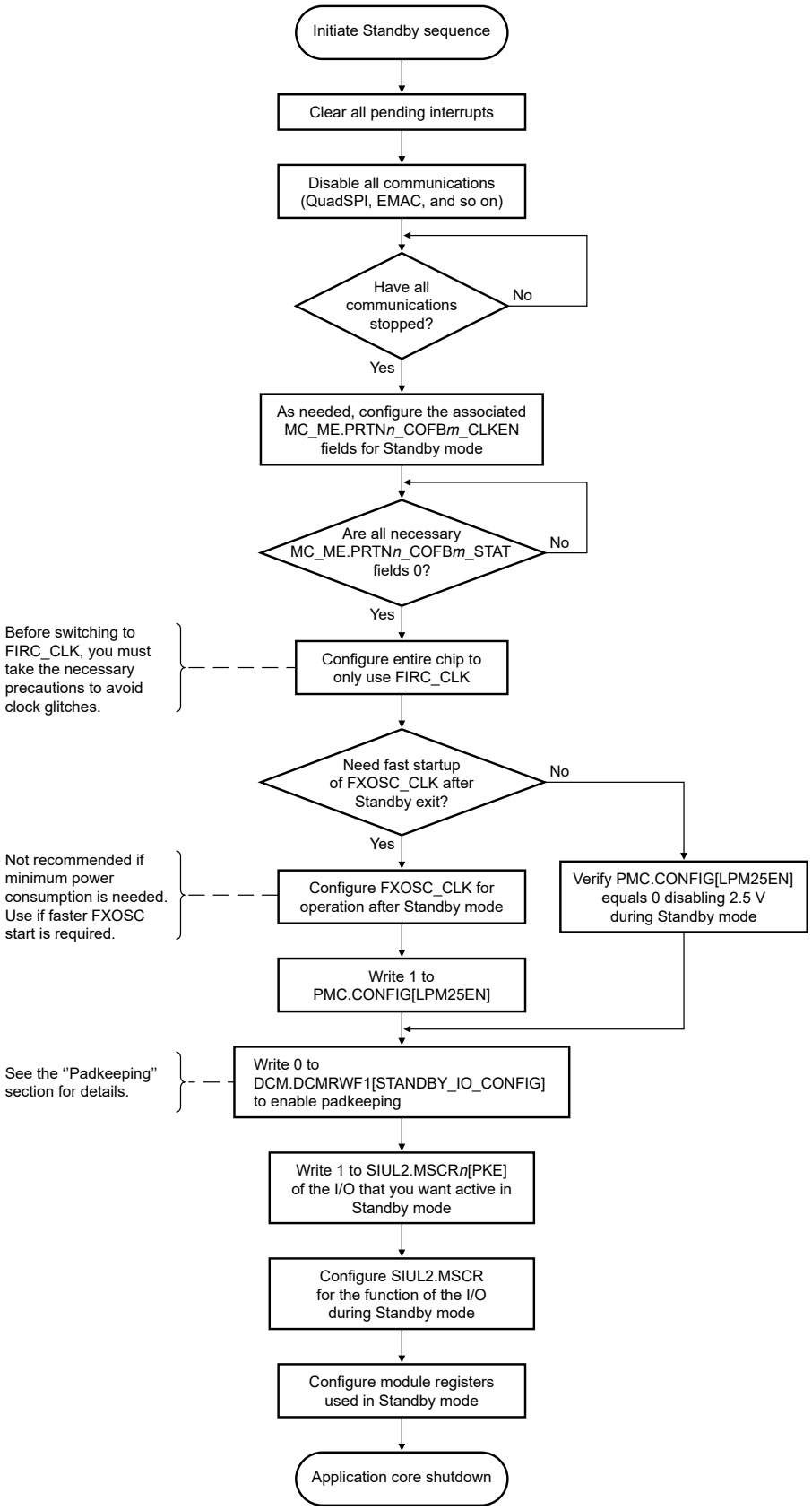



**Figure 152. I/O and module configuration for Standby mode**

#### 40.4.1.2 SW2: Application core shutdown process

The application cores(s) execute a WFI (as opposed to the main core running the Standby mode entry sequence). See the section "Application core shutdown" in the MC_ME chapter for more information.

The main core configuration (programming valid core ID and enabling standby entry process) and wake-up source configuration must also be set in SW2, so that SW4 contains only the main core WFI execution.

#### 40.4.1.3 SW3: Flash memory low-power handshake and PMC last-mile regulator

In this process, you execute a flash memory low-power handshake and disable the PMC last-mile regulator by executing the procedures indicated in Figure 153 and Figure 154 .

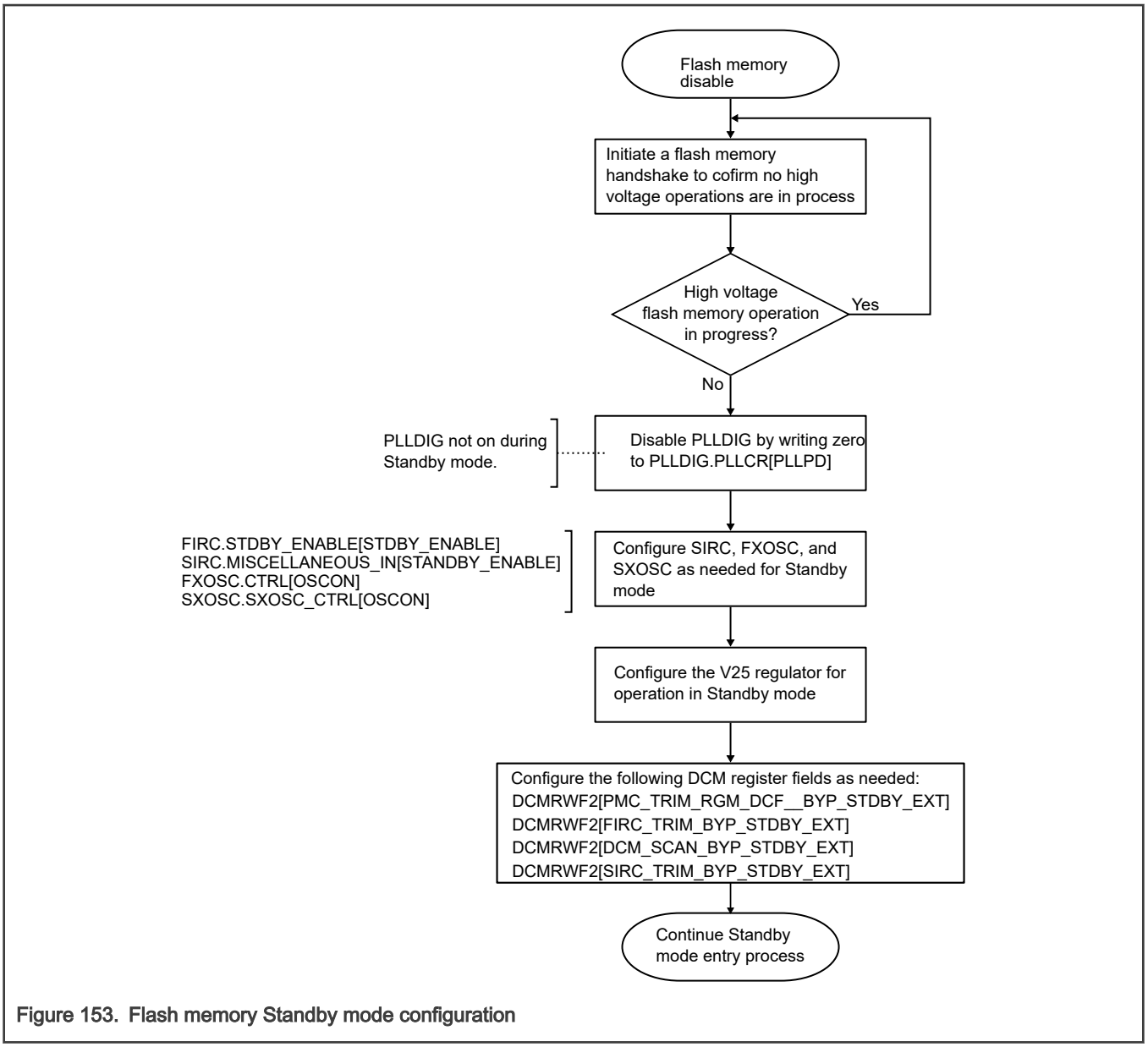



**Figure 153. Flash memory Standby mode configuration**

**NOTE**

Disable the last-mile regulator according to the Last-mile regulator disable sequence (not applicable for MCXE315, MCXE316, and MCXE317).

#### 40.4.1.3.1 Last-mile regulator disable sequence (not applicable for MCXE315, MCXE316, and MCXE317)

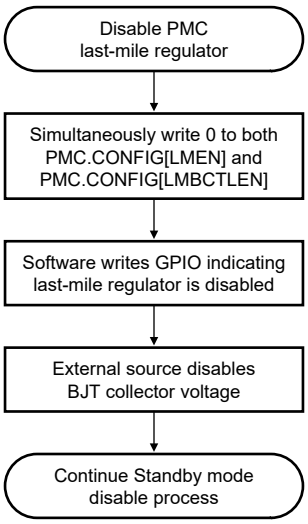



**Figure 154. Last-mile regulator disable sequence**

When exiting Standby mode, the 1.1 V capacitor is charging. You can charge the 1.5 V capacitor during or after the 1.1 V capacitor is charged. If you charge the 1.5 V capacitor sequentially after the 1.1 V capacitor, you will need additional time to complete the overall charging process.

Leaving PMCs CTRL[LMBCTLEN] = 1 saves time when exiting Standby mode and does not draw any additional current during this mode.

### 40.4.1.4 SW4: Main core shutdown process

For information on this process, see these in the "Mode Entry Module (MC_ME)" chapter:

- Figure "Standby entry sequence along with main core shutdown"
- Section "Main core shutdown and Standby mode entry"

You must configure WKPU before disabling interrupts to avoid missing any events as shown in the Standby entry sequence along with main core shutdown flowchart in the MC_ME chapter. You must program WKPU, disable interrupts, and configure MC_ME's Standby mode entry in SW1 (see SW1: Module shutdown process for more information). In SW4, you must perform only the main-core WFI execution.

## 40.4.2 Hardware Standby mode entry sequence

The hardware Standby mode entry sequence consists of handshaking between MC_PCU and MC_ME occuring after Software Standby mode entry sequence completes. The FSM in MC_PCU performs these steps automatically and does not require your intervention (see Power sequence FSM in the Power Control Unit (MC_PCU) chapter for more information).

During the MC_ME and PCU phase, MC_ME and PCU:

- Enable FIRC
- Deassert isolation
- Deassert reset to PD1

## 40.4.3 PMC Standby mode entry

The PMC Standby mode entry sequence starts after Hardware Standby mode entry sequence completes, and consists of these phases:

1. Standby mode entry acknowledgment and initiation of an internal low-power process on receiving Standby mode entry request
2. Disabling of the boot regulator within the PMC low-power process (for MCXE31B)
3. Disabling of the V25 regulator and the FPM LVR monitors
4. Disabling of the V25 regulator (oscillator and flash memory supply) and the LPM monitor, unless it is enabled during Standby mode (see PMC's CONFIG[LPM25EN] field in the "Power management controller (PMC)" chapter for more information)
5. Disabling of the VDD_HV_B LPM monitors (not applicable for MCXE315, MCXE316, and MCXE317)

During PMC phase, after receiving an LPM request, PMC:

- Deasserts FPM ready signal
- Starts the courtesy timer
- When the courtesy timer expires, PMC:
  - Disables the V25 regulator
  - Disables the HP reference blocks
  - Disables the external NMOS
  - Opens the PD0 switch
  - Enables a core LPM request
  - If selected, enables a 2.5 V LPM request
  - Disengages FPM monitors
  - Disables the VDD_HV_B LPM monitor, if present and deselected
- Waits for LPM signal deassertion

## 40.5 Chip status at the end of Standby mode entry sequence

After PMC Standby mode entry completes, the chip completes Standby mode entry as follows:

1. Configures Standby domain peripherals according to SW1 (see SW1: Module shutdown process).
2. Enables pad keeping on pins as described in SW1 (see SW1: Module shutdown process). See Pad keeping for other details.
3. Powers down all memory types except Standby RAM (in SRAM0).
   - The V25 regulator can remain on during Standby mode. However, for maximum power savings, it must remain off in this mode.
4. Isolates Standby and Run domains from each other:
   - Standby domain is functional.
   - Run domain is held in reset.
5. Configures the system clock (FIRC_CLK or PLL_PHI*n*_CLK, depending on their configuration) and other clock sources according to SW3 (see Figure 153 for the procedure details).

   The cores are off and in the Standby domain.
6. Turns off the boot (not applicable for MCXE315, MCXE316, and MCXE317) and FPM regulators
7. Waits for a wake-up event to initiate recovery from Standby mode.

## 40.6 Chip operation in Standby mode

This chip supports the following functionalities in Standby mode:

- STANDBY_RAM content retention during Standby mode.
- Wake-up from up to 60 digital inputs (for details, see the signals WKUP[*n*] functions of WKPU module in the IOMUX file attached to this document). The section WKPU configuration on the chip in the "Wakeup Unit (WKPU)" chapter shows mapping of the wake-up sources.
- Wake-up from up to 16 analog inputs through the Trigger mode functionality (see the signals CMP*n*_IN*m* functions of CMP*n* modules in the IOMUX file attached to this document).
- Wake-up from on-chip timers:
  - RTI (function of PIT[0])
  - SWT0
  - RTC
- Ability to configure the chip clocking modules to optionally enable or disable in Standby mode (FIRC, SXOSC, FXOSC, and SIRC).

## 40.7 Standby mode exit

This chip supports Standby mode exit from a wake-up, functional reset, or destructive reset event. The sources that cause chip Standby mode exit are:

- MC_RGM functional reset event
- MC_RGM destructive reset event
- WKPU wake-up events, WKPU[0]–WKPU[63]. See the WKPU chapter for more information.

After Standby mode exit, the following events occur (for more information, see "Power sequence FSM" in the MC_PCU chapter for MC_PCU FSM transitions during entry into and exit from Standby mode):

1. A wake-up event arrives.
2. FIRC starts powering up (if disabled in Standby mode).
3. PMC starts the transition process to FPM (for example, enables the last-mile regulator and provides V11_RUN supply to the chip).
4. MC_PCU removes the isolation between Run and Standby domains.
5. Run domain reset deasserts (asserts on Standby mode entry) and the chip undergoes a functional reset exit sequence for this domain.
6. The chip enters Run mode of operation.

**NOTE**

A reset event during standby results in pad controls to get reset. Thereby, resulting in unpredictable toggling at GPIO.

### 40.7.1 Faster Standby mode exit

The chip supports an optional configuration for faster recovery from Standby mode on the expense of a higher capacitor recharging current. See the CONFIG[FASTREC] field description in the PMC chapter for more information on faster PMC recovery from Standby mode.

This chip supports an optional feature that bypasses:

- FIRC trimming

- PMC trimming
- DCM scanning

To use the bypass operation, write 1 to DCM.DCMRWF2[5], DCM.DCMRWF2[4], and DCM.DCMRWF2[3] respectively before Standby mode entry. This results in a considerable reduction in Standby mode exit duration. The trim values are retained across Standby mode and bypassing these values does not cause any impact. Even if the FIRC trimming is bypassed, the FIRC must be at 48 MHz before entering Standby mode.

Configuration to achieve the tMODE_STDBYEXIT_FAST as specified in Datasheet

- DCMRWF2[PMC_TRIM_RGM_DCF__BYP_STDBY_EXT] = 1
- DCMRWF2[FIRC_TRIM_BYP_STDBY_EXT] = 1
- DCMRWF2[DCM_SCAN_BYP_STDBY_EXT] = 1

### 40.7.2 Last-mile regulator enable sequence (not applicable for MCXE315, MCXE316, and MCXE317)

After Standby mode exit, you must re-enable the PMC last-mile regulator before transitioning to faster clock frequencies. Figure 155 shows the last-mile regulator enable sequence part of the Standby mode exit process.

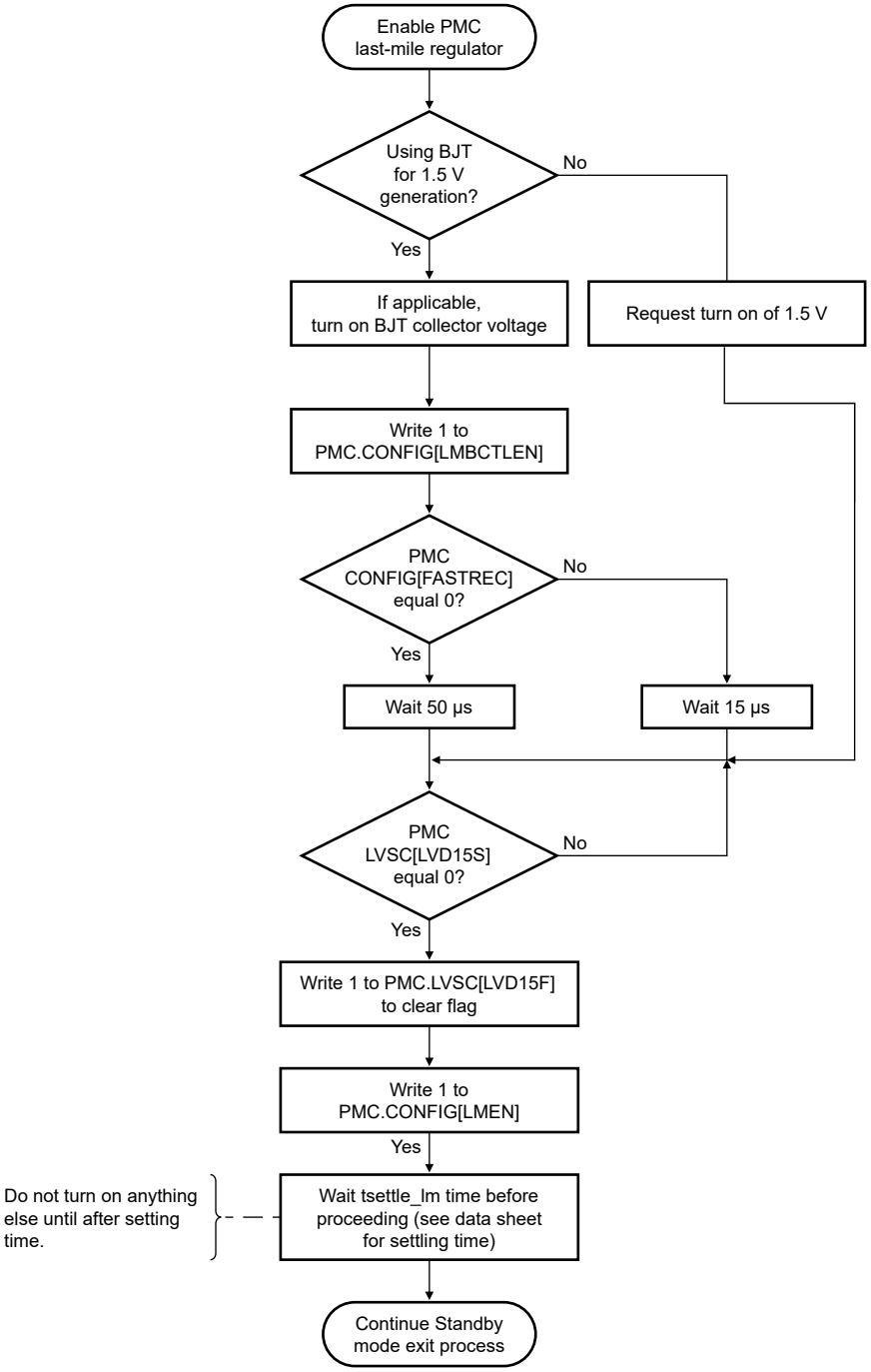



**Figure 155. Last-mile regulator enable sequence**

## 40.8 Chip power domain partitioning

These are the modules in the Standby domain. For more information, see "Peripheral reset status" in the "Reset Overview" chapter.

- FIRC
- FXOSC
- MC_RGM

- DCM
- WKPU
- PIT_0 (RTI)
- LPCMP_[0:2]
- PMC
- SIRC
- SXOSC
- RTC
- SWT_0
- DCF clients

### 40.8.1 Module power status

**Table 211. Module power status**

| PD0 contents | MCXE31B | MCXE317 | MCXE315/MCXE316 |
|---|---|---|---|
| External Pin wakeups | Minimum 60 | Minimum 60 | 33 on 100MQFP & 12 on 48LQFP |
| RTC_API | Yes | Yes | Yes |
| LPCMP0 | Yes | Yes | Yes |
| LPCMP1 | Yes | Yes | No |
| LPCMP2 | Yes | No | No |
| SWT0 | Yes | Yes | Yes |
| WKPU | Yes | Yes | Yes |
| PMC | Yes | Yes | Yes |
| RGM | Yes | Yes | Yes |
| SXOSC(only used for RTC) | Yes | Yes | No |
| FXOSC | Yes | Yes | Yes |
| FIRC | Yes | Yes | Yes |
| SIRC | Yes | Yes | Yes |
| CLK OUT | Yes | Yes | Yes |
| SRAM | 32K | 32K | 32K |
| PIT_RTI_0 | Yes | Yes | Yes |
| ATX | Yes | Yes | Yes |
| DCM and DCF records | Yes | Yes | Yes |
| DCM Flash Interface | Yes | Yes | Yes |

SWT0 (Cortex-M7_0) resides in the Standby domain and supports a configurable hardware-based timer operation during Standby mode depending on the configuration of SWTs.

All clock sources (excpet PLLDIG) are available in Standby mode. SIRC is present in the Standby domain for low-power wake-up, but can be enabled.

**NOTE**

This chip does not support Stop and Wait modes. The only low-power mode it supports is Standby.

## 40.9 Pad keeping

Pad keeping allows you to return to I/O Run mode configuration settings after using I/O during Standby mode. It eliminates I/O reconfiguration in Run mode.

### 40.9.1 Pad keeping on Run domain pins

The process of entering Standby mode ensures that the chip maintains the Run domain configuration settings. Software Standby mode entry sequence specifies that you must write 0 to DCM's DCMRWF1[STANDBY_IO_CONFIG] before configuring I/O pad keeping. You could consider DCM's DCMRWF1[STANDBY_IO_CONFIG] field as a global enable for all pad keeping purposes during Standby mode. If you are unable to write to this field as described, pad keeping works as explained in this chapter, during Standby mode.

After the chip exits Standby mode, you must write 1 to DCM's DCMRWF1[STANDBY_IO_CONFIG] field (see "Chip specific register descriptions" in the DCM chapter). Writing 1 to this field disables pad keeping and places chip I/O according to its Run mode configuration.

### 40.9.2 Pad keeping on Standby domain pins

The Standby domain pads can have pad keeping enabled or disabled based on pad availability in Standby mode. Writing to SIUL2's MSCR*n*[PKE] field configures Standby mode pad keeping on a selected I/O.

Ensure that DCM's DCMRWF1[STANDBY_IO_CONFIG] = 0 before Standby mode entry for pad keeping on Standby domain pads. Write 0 to the field in case its value is not 0 already.

If a pad is not required during Standby mode, the corresponding MSCR*n*[PKE] field in SIUL2 must be 0.

### 40.9.3 Pad keeping configuration procedure

This procedure specifies how to enable an I/O for pad keeping:

1. Configure SIUL2's MSCR*n* register to control the pad state prior to Standby mode entry (for example, writing 0 to SIUL2's MSCR*n*[OBE], MSCR*n*[IBE], and MSCR[PUE] fields tristates the corresponding I/O).
2. Configure SIUL2's MSCR*n*[PKE] field as needed for an I/O pad keeping state during Standby mode.
3. Write 0 to DCM's DCMRWF1[STANDBY_IO_CONFIG].

   The application core executes WFI, and Standby mode sequence starts.
4. Write 1 to DCM's DCMRWF1[STANDBY_IO_CONFIG] field on Standby mode exit to disable pad keeping. This configures SIUL2 according to its previous settings.

If you write 1 to DCM's DCMRWF1[STANDBY_IO_CONFIG] field before entering Standby mode, the isolation removal hardware removes pad keeping on Standby mode exit. This (writing 1 to DCM's DCMRWF1[STANDBY_IO_CONFIG] field before Standby mode entry) is useful in case of low-power debug because enabling pad keeping does not allow low-power debug protocol to work properly (because the TDO pad is padkept). For low-power debug, you must write 1 to DCM's DCMRWF1[STANDBY_IO_CONFIG] field prior to Standby mode entry.

In case of Standby mode exit by reset, the pad keeping of the reset pin is removed when the chip is reset on Standby mode wake up. See the "GPIO padkeeping enable" signal in Figure 157. The signal corresponds to wake-up via reset event case.

### 40.9.4 Pad keeping waveforms

### 40.9.4.1 Pad keeping when the chip wakes up from Standby mode via an interrupt event

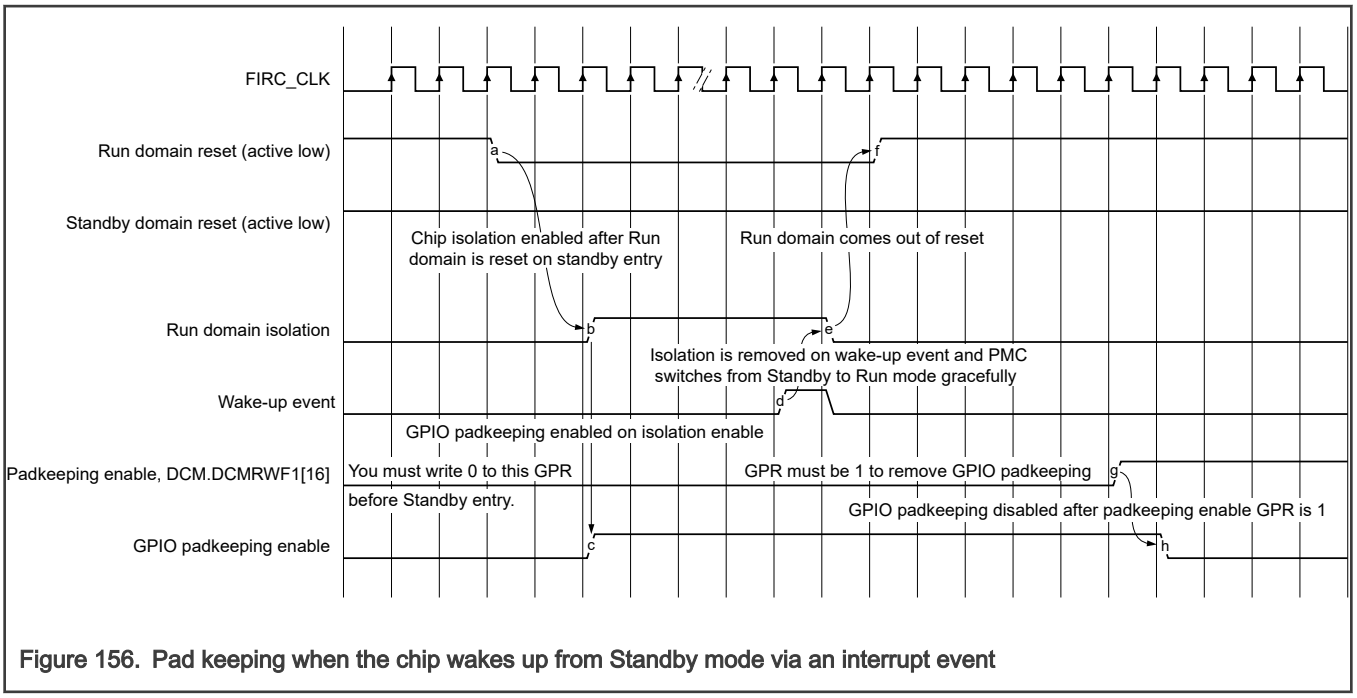



**Figure 156. Pad keeping when the chip wakes up from Standby mode via an interrupt event**

### 40.9.4.2 Pad keeping when the chip wakes up from Standby mode via reset

If the chip exits Standby mode via reset, the reset pad keeping is removed when the chip resets after Standby-mode wake-up. See the figure's "GPIO padkeeping enable" waveform corresponding to wake-up via a reset event.

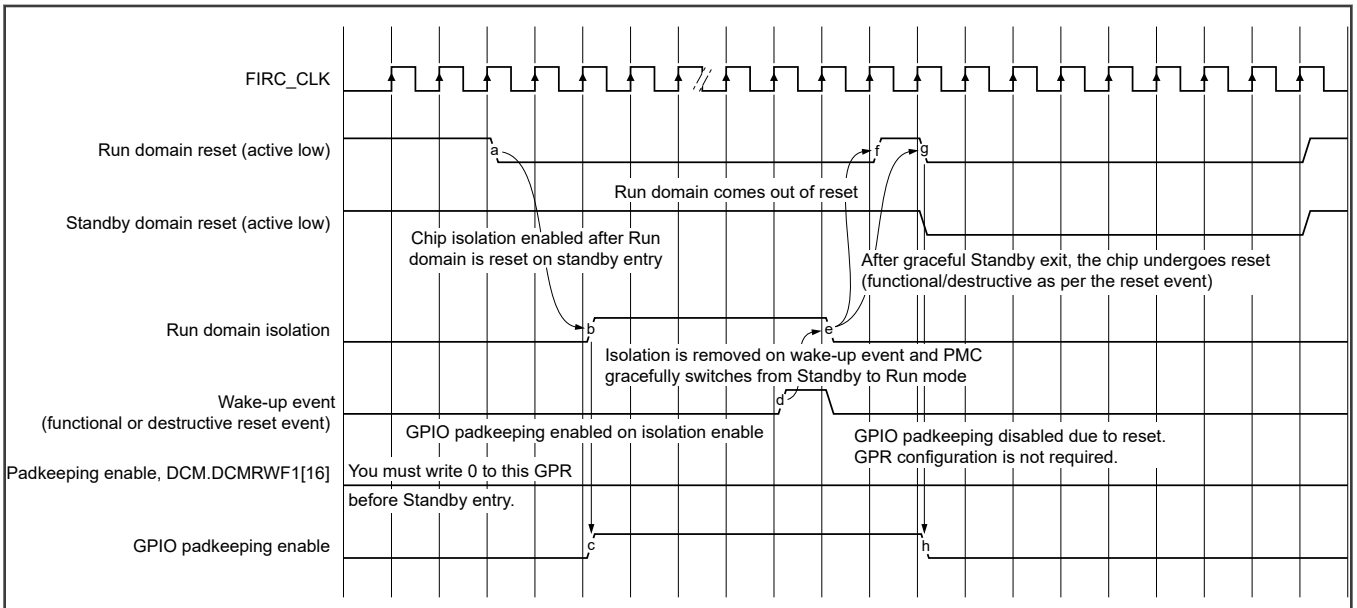



**Figure 157. Pad keeping when the chip wakes up from Standby mode via reset**

## 40.9.5 SIUL2's fields for actively-driven pins in Standby mode

If you use a WKPU pin as a wake-up input, perform these operations on SIUL2's fields:

1. Write 1 to MSCR*x*[IBE].

2. Program MSCR*x*[PUE] and MSCR*x*[PUS] according to the use case.
3. Write 0 to MSCR*x*[PKE].

For the GPIO pins that are driven to high impedance during Standby mode:

1. Write 0 to DCM's DCMRWF1[STANDBY_IO_CONFIG] field.
2. Write 0 to SIUL2's MSCR*x*[SSS] field (GPIO mode).
3. Write 0 to SIUL2's MSCR*x*[IBE] and MSCR*x*[OBE] fields.

Some of the pins are, by default, actively driven in Standby mode. If these pins retain a static value throughout the mode, program the corresponding MSCR*x*[PKE] bits individually.

This table shows the list of pins available in Standby mode along with the functions they perform.

**Table 212. Active pins in Standby mode**

| Pin[1] | Function |
|---|---|
| GPIO_4 | CMP0_OUT |
| GPIO_5 | RESET_b |
| GPIO_9 | CMP2_OUT |
| GPIO_11 | CMP0_RRT |
| GPIO_12 | CLKOUT_STANDBY; CMP1_OUT |
| GPIO_69 | CMP2_RRT |
| GPIO_110 | CMP0_RRT |
| GPIO_131 | CMP0_OUT |
| GPIO_138 | CLKOUT_STANDBY |
| GPIO_143 | CMP1_RRT |
| GPIO_158 | CMP1_OUT |
| GPIO_159 | CMP1_RRT |
| GPIO_174 | CMP0_OUT |
| GPIO_175 | CMP0_RRT |
| GPIO_196 | CMP2_OUT |
| GPIO_197 | CMP2_RRT |

1. See the IOMUX sheet attached to this document for details on pins configuration in different chips.

## 40.10 Glossary

**Application core** Core apart from the main core

**Boot regulator** The on chip 1.1 V regulator that is active during startup and entry and exit from Standby mode

**FSM** Finite state machine

**FPM** Full-power mode (Run mode)—uses the last-mile regulator and 1.5 V source to generate the 1.1 V core logic supply during a full-power operation (V11_RUN)

**LPM** Low-power mode (Standby mode)—uses the low-power regulator to generate the 1.1 V core logic supply during low-power operation (V11_STANDBY)

| | |
|---|---|
| **LVR** | Low voltage reset |
| **Main core** | Core initiating the chip Standby mode request (for example, the core corresponding to the "core index" in MC_ME's MAIN_COREID register) |
| **Pad keeping** | Maintains I/O pad configuration during Standby mode, if enabled |
| **V11_STANDBY** | Core logic and clock sources, low-voltage supply to Standby domain |
| **V11_RUN** | Low-voltage supply to Run domain |
| **V15** | High-current input for core or logic supply from either an external BJT or from direct 1.5 V external supply |
| **V25** | Flash memory, FXOSC, and PLLDIG high-voltage supply |
| **VDDA_ADC** | ADC supply voltage |
| **VREFH** | ADC high-voltage reference supply |
| **VREFL** | ADC low-voltage reference supply |
| **VSS** | Core logic ground supply |
| **VDD_HV_A** | Main I/O voltage supply (5 V or 3.3 V) |
| **VDD_HV_B** | Other I/O domain voltage supply (5 V or 3.3 V) |
| **VRC_CTRL** | PMC voltage regulator control output using the BJT option to generate a 1.5 V supply |
| **WFI** | Wait for interrupt software instruction |

# Chapter 41
# Power Management Controller (PMC for MCXE31B)

## 41.1 Introduction

PMC is the power management controller for the MCXE31x family of microcontrollers. It provides multiple power options to allow you to optimize power consumption for the level of functionality needed. It includes internal voltage regulators, POR, and the integrated low/high voltage detect system with reset (brown-out) capability. The voltage regulator requires a 3.3 V or 5 V input to generate all the required secondary supplies.

## 41.2 Features

PMC includes the following features:

- Combination of internal and external voltage regulator options, offering RUN and Standby modes
- Active POR providing brown-out detect
- LVR for all system-relevant power domains
- LVD and HVD as indication for software

## 41.3 Modes of operation

PMC provides two basic modes of operation for the voltage regulators and monitors:

- FPM, which is used on chip-level in RUN modes: For high-current consumption
- LPM, which is used on chip-level for Standby modes: For low-current consumption

## 41.4 Block diagram

The following figure shows the block diagram for this module.

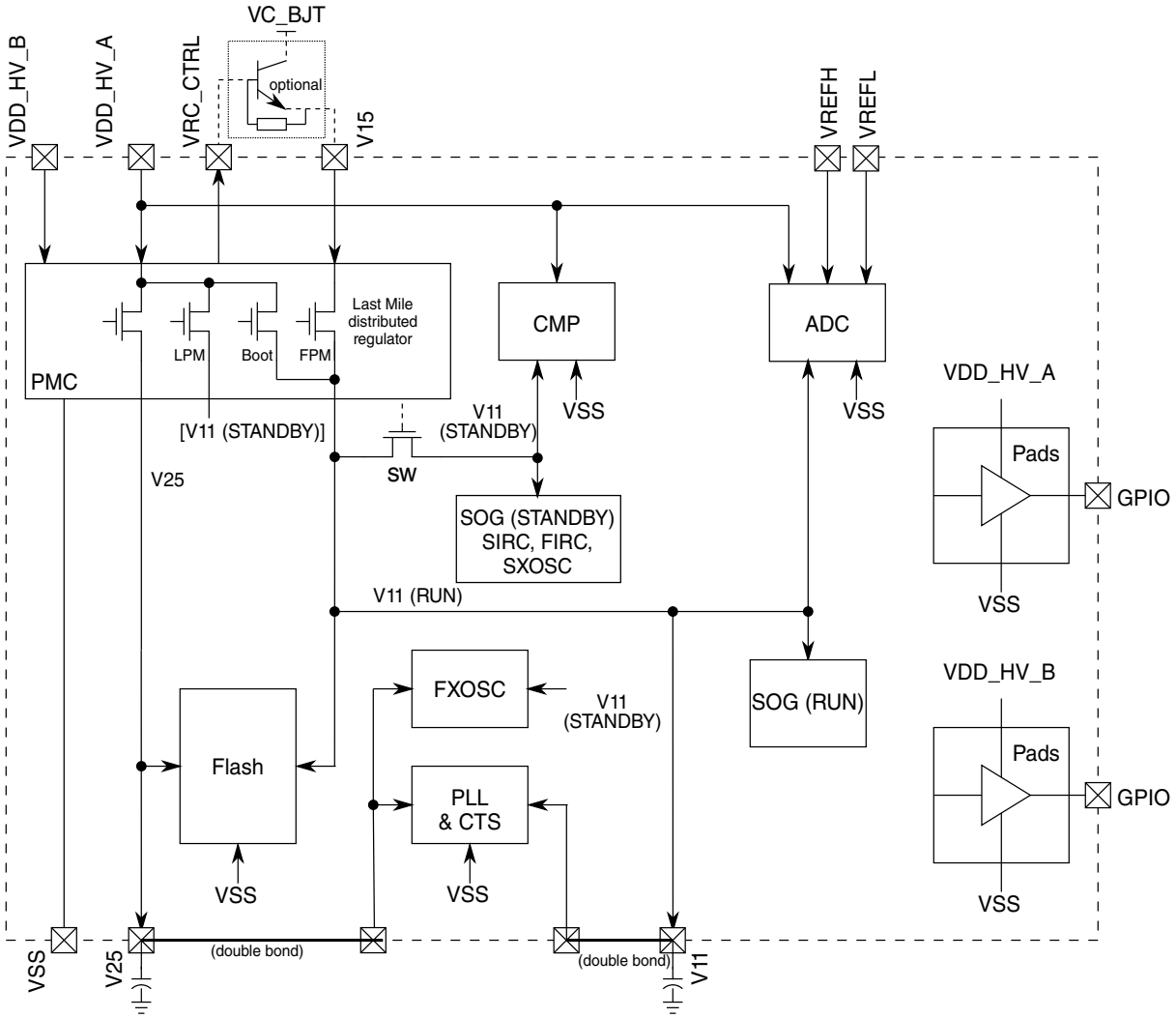



Figure 158. PMC block diagram

## 41.5 Signals

This table describes the PMC module signals.

Table 213. Signal Description

| Signal | I/O | Description |
|---|---|---|
| VDD_HV_A | Supply input | This is the primary high-voltage supply input to PMC. VDD_HV_A is used for the PMC internal precision references. After the VDD_HV_A domain is powered up, it must be kept powered at all times of operation (FPM and LPM). |
| VDD_HV_B[1] | Supply input | This is the secondary high-voltage supply input supervised by the PMC. After the VDD_HV_B domain is powered up, it must be kept powered at all times of operation (FPM and LPM). |

*Table continues on the next page...*

Table 213. Signal Description (continued)

| Signal | I/O | Description |
|---|---|---|
| V25 | Supply output | V25 power supply domain is driven by a fully integrated low-dropout linear voltage regulator. It supplies the Flash memory and (via a double bond) the clock modules. |
| V15[1] | Supply input | This is the high-current input for core/logic supply that can be fed by an external BJT or another source. |
| VRC_CTRL | Output | VRC_CTRL connects to the base of external BJT, if this option is used to generate V15. |
| V11 | Supply output | V11 is the core/logic supply. It is driven by a fully integrated low-dropout linear voltage regulator. |
| VSS | Ground | VSS must be grounded. All VSS Pins need to be externally connected to the same ground node. |

1. VDD_HV_B and V15 not present in MCXE315/MCXE316 and MCXE317.

## 41.6 Functional description

The following sections describe functional details of the PMC.

### 41.6.1 Reset

The POR and all LVRs are combined into one single MCU POR.

After an MCU POR event, it can be determined which power domain caused it, by reading in the PMC_LVSC register, the POR flag, and LVR flags.

After an initial power ramp up of the MCU in the PMC_LVSC register, the POR flag and the LVR flags are all set to 1. The Go/Nogo flags have an arbitrary value.

**NOTE**

After an initial power ramp up, all flags in the LVSC register must be cleared (by writing 0xFFFFFFFF to the LVSC register).

Because the flags are sticky bits, it is required to clear them before usage. So, in case of an unexpected MCU POR, the source of the problem can be tracked and debugged by reading the flags in the LVSC register.

### 41.6.2 Interrupts

PMC includes two interrupt sources:

- HVD interrupt: It combines all HVD monitors into one interrupt source. Interrupt enable is the HVDIE field in the CONFIG register. See the PMC Configuration Register (CONFIG) and Low Voltage Status and Control Register (LVSC) registers for details.
- LVD interrupt: It combines the LVD15 and LVD5A monitors into one interrupt source. Interrupt enable is the LVDIE field in the CONFIG register. See the CONFIG and LVSC registers for details.

## 41.7 PMC register descriptions

### 41.7.1 PMC memory map

This section includes the PMC module memory map and detailed descriptions for all the registers.

PMC base address: 402E_8000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Low Voltage Status and Control Register (LVSC) | 32 | RW | See section |
| 4h | PMC Configuration Register (CONFIG) | 32 | RW | See section |
| Ch | Version ID register (VERID) | 32 | R | 0200_0001h |

## 41.7.2 Low Voltage Status and Control Register (LVSC)

### Offset

| Register | Offset |
|---|---|
| LVSC | 0h |

### Function

This register contains status and control bits to support the low-voltage reset and low- or high-voltage detect function. When the PMC is in LPM, the low- or high-voltage detect systems are disabled.

**NOTE**

For all flags that are not affected by reset (POR flag, all LVR flags, all GNG flags), in case a reset occurs at the same time while trying to clear the flags (by writing 1), the flag value is not defined appropriately. In this case, you need to clear the flag again after exit from reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PORF | 0 | | | | | GNG11OS... | GNG25OS... | LVR11LPF | LVR11F | LVR25LPF | LVR25F | LVRBLPF | LVRBF | LVRALPF | LVRAF |
| W | W1C | | | | | | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | u | 0 | 0 | 0 | 0 | 0 | u | u | u | u | u | u | u | u | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | LVD15S | LVD5AS | HVD11S | HVD25S | HVDBS | HVDAS | 0 | | LVD15F | LVD5AF | HVD11F | HVD25F | HVDBF | HVDAF |
| W | | | | | | | | | | | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>PORF | POR flag<br>Indicates that a power-on reset event has occurred. Writing 1 to this field clears it, and other reset sources have no effect. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No power-on reset event has occurred<br>1b - Power-on reset event has occurred |
| 30-26<br>— | Reserved |
| 25<br>GNG11OSCF | Go/NoGo detect flag on Osc part of V11 domain<br>Indicates that the Go/NoGo sensor has detected a low voltage on the V11 domain in FPM. This applies to only that part of the domain which supplies the 1.1V clocking modules (for example, PLL). Writing 1 to the field clears it, and other reset sources have no effect.<br>0b - No event has occurred<br>1b - NoGo event has occurred |
| 24<br>GNG25OSCF | GO/NoGo detect flag on Osc part of V25 domain<br>Indicates that the Go/NoGo sensor has detected a low voltage on the V25 domain in FPM. This applies to only that part of the domain which supplies the 2.5V clocking modules (for example, XOSC and IRC). Writing 1 to the field clears it, and other reset sources have no effect.<br>0b - No event has occurred<br>1b - NoGo event has occurred |
| 23<br>LVR11LPF | LVR11LP flag on V11 domain<br>Indicates that a low-voltage reset event has occurred on the 1.1V V11 power domain (FPM or LPM). Writing 1 to the field clears it, and other reset sources have no effect.<br>0b - No low-voltage reset event has occurred<br>1b - Low-voltage reset event has occurred |
| 22<br>LVR11F | LVR11 flag on V11 domain in FPM<br>Indicates that a low-voltage reset event has occurred on the 1.1V V11 power domain in the FPM. Writing 1 to this field clears it, and other reset sources have no effect.<br>0b - No low-voltage reset event has occurred<br>1b - Low-voltage reset event has occurred |
| 21<br>LVR25LPF | LVR25LP flag on V25 domain<br>Indicates that a low-voltage reset event has occurred on the V25 power domain (FPM or LPM). Writing 1 to this field clears it, and other reset sources have no effect.<br>0b - No low-voltage reset event has occurred<br>1b - Low-voltage reset event has occurred |
| 20<br>LVR25F | LVR25 flag on V25 domain in FPM<br>Indicates that a low-voltage reset event has occurred on the V25 power domain in FPM. Writing 1 to this field clears it, and other reset sources have no effect. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No low-voltage reset event has occurred<br>1b - Low-voltage reset event has occurred |
| 19<br>LVRBLPF | LVRBLP flag on VDD_HV_B domain<br>Indicates that a low-voltage reset event has occurred on the VDD_HV_B power domain (FPM or LPM). Writing 1 to this field clears it, and other reset sources have no effect.<br>0b - No low-voltage reset event has occurred<br>1b - Low-voltage reset event has occurred |
| 18<br>LVRBF | LVRB flag on VDD_HV_B domain in FPM<br>Indicates that a low-voltage reset event has occurred on the VDD_HV_B power domain in FPM. Writing 1 to this field clears it, and other reset sources have no effect.<br>0b - No low-voltage reset event has occurred<br>1b - Low-voltage reset event has occurred |
| 17<br>LVRALPF | LVRALP flag on VDD_HV_A domain<br>Indicates that a low-voltage reset event has occurred on the VDD_HV_A power domain (FPM or LPM). Writing 1 to this field clears it, and other reset sources have no effect.<br>0b - No low-voltage reset event has occurred<br>1b - Low-voltage reset event has occurred |
| 16<br>LVRAF | LVRA flag on VDD_HV_A domain in FPM<br>Indicates that a low-voltage reset event has occurred on the VDD_HV_A power domain in FPM. Writing 1 to this field clears it, and other reset sources have no effect.<br>0b - No low-voltage reset event has occurred<br>1b - Low-voltage reset event has occurred |
| 15-14<br>— | Reserved |
| 13<br>LVD15S | LVD15 status on V15 domain in FPM<br>Shows the status of the 1.5V low-voltage detect, LVD15, on the V15 power domain. This monitor indicates when the V15 voltage level generated from external is on target. This feature is available only in FPM and disabled in LPM. After a reset or wakeup from LPM, the software should clear the LVD15F flag and check the status bit LVD15S to determine voltage level on V15 supply.<br>0b - Voltage on V15 is above low-voltage detect threshold or LPM.<br>1b - Voltage on V15 is below low-voltage detect threshold and FPM. |
| 12<br>LVD5AS | LVD5A status on VDD_HV_A domain in FPM<br>Shows the status of the 5V low-voltage detect, LVD5A, on the VDD_HV_A power domain. This monitor indicates if the voltage is below a certain threshold, which is set slightly below 4.5V (see Datasheet for |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | exact value). The feature is only available in FPM and disabled in LPM. After a reset or wakeup from LPM, the software should clear the LVD5AF flag and check the status bit LVD5AS to determine voltage level on VDD_HV_A supply.<br>0b - Voltage on VDD_HV_A is above low-voltage detect threshold<br>1b - Voltage on VDD_HV_A is below low-voltage detect threshold |
| 11<br>HVD11S | HVD11 status on V11 domain in FPM<br>Shows the status of the high-voltage detect, HVD11, on the V11 power domain. This feature is only available in FPM and disabled in LPM.<br>0b - Voltage on V11 is below high-voltage detect threshold or LPM.<br>1b - Voltage on V11 is above high-voltage detect threshold and FPM. |
| 10<br>HVD25S | HVD25 status on V25 domain in FPM<br>Shows the status of the high-voltage detect, HVD25, on the V25 power domain. The feature is only available in FPM and disabled in LPM.<br>0b - Voltage on V25 is below high-voltage detect threshold or LPM.<br>1b - Voltage on V25 is above high-voltage detect threshold and FPM. |
| 9<br>HVDBS | HVDB status on VDD_HV_B domain in FPM<br>Shows the status of the high-voltage detect, HVDB, on the VDD_HV_B power domain. The feature is only available in FPM and disabled in LPM.<br>0b - Voltage on VDD_HV_B is below high-voltage detect threshold or LPM.<br>1b - Voltage on VDD_HV_B is above high-voltage detect threshold and FPM. |
| 8<br>HVDAS | HVDA status on VDD_HV_A domain in FPM<br>Shows the status of the high-voltage detect HVDA on the VDD_HV_A power domain. The feature is only available in FPM and disabled in LPM.<br>0b - Voltage on VDD_HV_A is below high-voltage detect threshold or LPM.<br>1b - Voltage on VDD_HV_A is above high-voltage detect threshold and FPM. |
| 7-6<br>— | Reserved |
| 5<br>LVD15F | LVD15 flag on V15 domain in FPM<br>PMC writes 1 to the LVD15F field when the LVD15S status field changes. To clear LVD15F, write 1 to it. If enabled, LVD15F causes an interrupt request.<br>0b - LVD15S has not changed.<br>1b - LVD15S has changed. |
| 4 | LVD5A flag on VDD_HV_A domain in FPM |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| LVD5AF | PMC writes 1 to this field when LVD5AS status field changes. To clear LVD5AF, write 1 to it. If enabled, LVD5AF causes an interrupt request.<br>0b - LVD5AS has not changed.<br>1b - LVD5AS has changed. |
| 3<br>HVD11F | HVD11 flag on V11 domain in FPM<br>PMC writes 1 to this field when the HVD11S status field changes. To clear HVD11F, write 1 to it. If enabled, HVD11F causes an interrupt request.<br>0b - HVD11S has not changed.<br>1b - HVD11S has changed. |
| 2<br>HVD25F | HVD25 flag on V25 domain in FPM<br>PMC writes 1 to the HVD25F field when HVD25S status field changes. To clear HVD25F, write 1 to it. If enabled, HVD25F causes an interrupt request.<br>0b - HVD25S has not changed.<br>1b - HVD25S has changed. |
| 1<br>HVDBF | HVDB flag on VDD_HV_B domain in FPM<br>PMC writes 1 to the HVDBF field when HVDBS status field changes. To clear HVDBF, write 1 to it. If enabled, HVDBF causes an interrupt request.<br>0b - HVDBS has not changed.<br>1b - HVDBS has changed. |
| 0<br>HVDAF | HVDA flag on VDD_HV_A domain in FPM<br>PMC writes 1 to the HVDAF field when HVDAS status field changes. To clear HVDAF, write 1 to it. If enabled, HVDAF causes an interrupt request.<br>0b - HVDAS has not changed.<br>1b - HVDAS has changed. |

### 41.7.3 PMC Configuration Register (CONFIG)

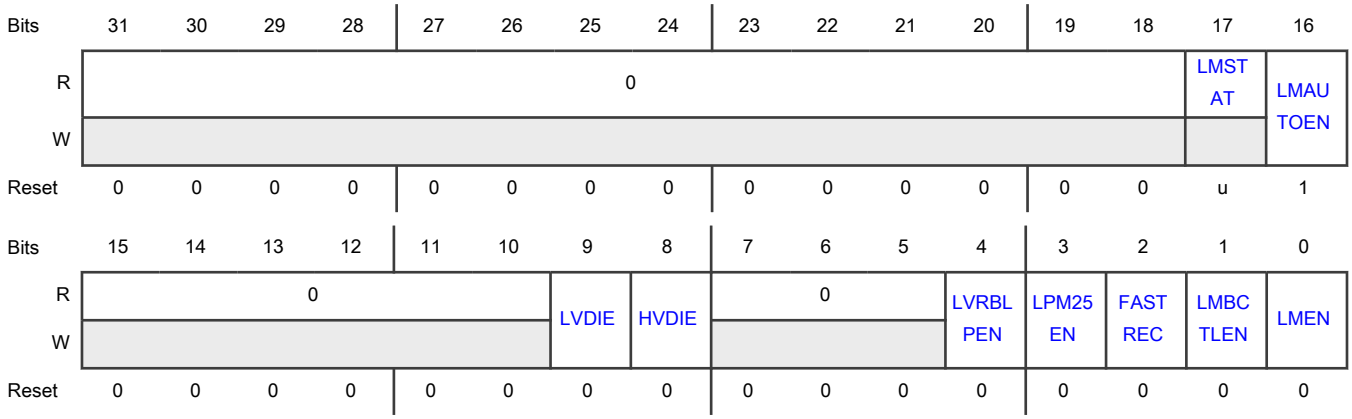
**Offset**

| Register | Offset |
|---|---|
| CONFIG | 4h |

**Function**

If the Last Mile Regulator option is not available on your device ( Version ID register (VERID) register: LMFEAT=0), then the bits LMEN, LMBCTLEN, LMAUTOEN and LMSTAT fields can not be written and read always 0.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-18<br>— | Reserved |
| 17<br>LMSTAT | Last Mile regulator status bit<br>This bits reflects the current status of the Last Mile regulator. This information is required as when auto turn over feature is enabled (LMAUTOEN=1) the PMC will switch automatically between Boot regulator and Last Mile regulator depending on the V15 status (LVD15S).<br>0b - Last Mile Regulator is off<br>1b - Last Mile Regulator is on |
| 16<br>LMAUTOEN | Last Mile regulator auto turn over bit<br>Enables to turn over automatically from Boot Regulator Mode to Last Mile regulator mode and vice versa depending on the V15 voltage status (LVD15S). As long as LMEN=0 software must make sure that the system clock is on FIRC clock or slower. To use higher clock speed software must set LMEN=1.<br>0b - Auto turnover disabled<br>1b - Auto turnover enabled |
| 15-10<br>— | Reserved |
| 9<br>LVDIE | Low voltage detect interrupt enable<br>Enables hardware interrupt requests if any of the following flags is set: LVD5AF, LVD15F. LVD interrupt must be disabled before going into LPM.<br>0b - LVD hardware interrupt is disabled (use polling).<br>1b - Request an LVD hardware interrupt when LVDA5F=1 or LVD15F=1. |
| 8<br>HVDIE | High voltage detect interrupt enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Enables hardware interrupt requests if any of the following flags is set: HVDAF, HVDBF, HVD25F, HVD11F.<br>0b - HVD hardware interrupt is disabled (use polling).<br>1b - Request an HVD hardware interrupt when HVDAF=1, HVDBF=1, HVD25F=1, or HVD11F=1. |
| 7-5<br>— | Reserved |
| 4<br>LVRBLPEN | LVRBLP enable bit during LPM<br>Controls whether the low-voltage reset detection (LVRBLP) on the VDD_HV_B power domain is active or inactive in LPM<br>0b - Low-voltage reset detection is disabled in LPM.<br>1b - Low-voltage reset detection is enabled in LPM. |
| 3<br>LPM25EN | V25 domain enable bit during LPM<br>Controls whether the V25 regulator and low-voltage reset detection (LVR25LP) are active or inactive in LPM<br>0b - V25 regulator and LVR25LP are disabled in LPM.<br>1b - V25 regulator and LVR25LP are enabled in LPM. |
| 2<br>FASTREC | Fast recovery from LPM enable bit<br>Controls the recovery time from LPM to FPM. At recovery from LPM, all the tank capacitors from the secondary supplies have to be recharged. This causes a high-current demand, which might not be met by the supply driving the VDD_HV_A primary domain. When selecting the fast recovery time, the current for recharging is approximately three times higher than that for FASTREC=0. The application must determine from the drive capability of the external VDD_HV_A regulator, the size of tank caps on the secondary supply pins and the selected recovery time if this is sufficient to start up from LPM in time.<br>0b - Normal recovery time from LPM<br>1b - Fast recovery time from LPM |
| 1<br>LMBCTLEN | Last Mile regulator base control enable bit<br>This field must be set to 1 if external BJT between VDD_HV_A and V15 is used on the PCB. The base of this BJT must be connected to the VRC_CTRL pin and is controlled by the PMC to regulate a voltage of 1.5V on V15 pin. After setting LMBCTLEN=1 the software has to wait for 15us (FASTREC=1) respectively 50us (FASTREC=0) before polling LVD15S=0 and then setting LMEN=1. This respects the softstart time of the V15 regulator. If LMAUTOEN=1 then LMBCTLEN can be left enabled when going into LPM, the hardware will turn the regulator off and back on automatically after recovery from LPM.<br>0b - External BCTL regulator for V15 disabled<br>1b - External BCTL regulator for V15 enabled |
| 0<br>LMEN | Last Mile regulator enable bit |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Enables the Last Mile regulator, which regulates an external 1.5V voltage on V15 down to the core and logic supply (V11 power domain), which is typically 1.1V. Setting LMEN=1 hands over the V11 voltage generation from the Boot regulator to the Last Mile regulator. The software must ensure that before enabling the Last Mile regulator, the voltage on V15 is sufficiently high as indicated by the LVD15S status field (LVD15S=0). To use external BJT between VDD_HV_A and V15, the LMBCTLEN field must be set before the LMEN field, and the software must wait until 1.5V is up (LVD15S=0). If LMAUTOEN=0 then to disable the Last Mile regulator, LMEN and LMBCTLEN must be cleared simultaneously (single register write). The software must disable (LMEN=0) the Last Mile regulator before going into LPM. After setting LMEN=1, software must wait a minimum time of 1.5us before changing clock rate.<br>0b - Last Mile regulator is disabled.<br>1b - Last Mile regulator is enabled. |

## 41.7.4 Version ID register (VERID)

### Offset

| Register | Offset |
|---|---|
| VERID | Ch |

### Function

This register returns the major and minor version numbers of hardware implementation.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MAJOR | | | | | | | | MINOR | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major version number<br>Returns the version number for the specification |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 23-16<br>MINOR | Minor version number<br>Returns the version number for the hardware implementation |
| 15-1<br>— | Reserved |
| 0<br>LMFEAT | Last Mile Regulator Feature<br>This read-only field shows if the Last Mile regulator feature is available.<br>0b - No Last Mile regulator<br>1b - Last Mile regulator (1.5V to 1.1V) is available |

## 41.8 Glossary

| | |
|---|---|
| **FPM** | Full Performance mode |
| **HVD** | High voltage detect |
| **IRC** | Internal RC oscillator |
| **LM** | Last mile regulator |
| **LPM** | Low Performance mode |
| **LVD** | Low voltage detect |
| **LVR** | Low voltage reset |
| **NVM** | Nonvolatile memory |
| **POR** | Power on reset |
| **XOSC** | External crystal oscillator |

# Chapter 42
# Power Management Controller (PMC for MCXE315, MCXE316, and MCXE317)

## 42.1 Overview

PMC provides multiple power options to allow you to optimize power consumption for the appropriate level of functionality. It includes:

- Internal voltage regulators
- POR
- Integrated low and high voltage detection system with reset (brownout) capability

The voltage regulator requires a 3.3 V or 5 V input to generate all the required secondary supplies.

### 42.1.1 Block diagram

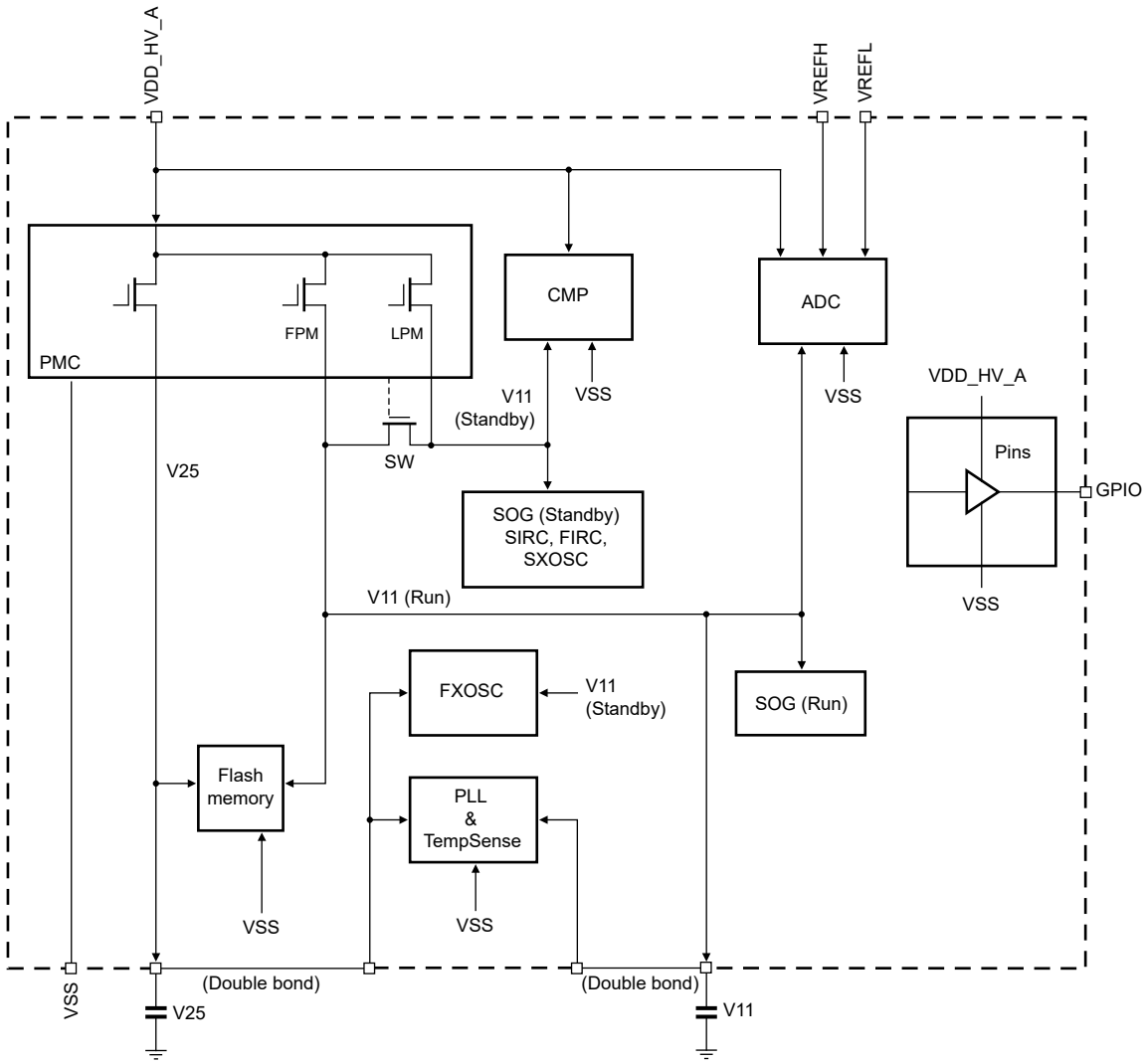



Figure 159. Block diagram

### 42.1.2 Features

- A combination of internal and external voltage regulator options, offering Run and Standby modes
- Active POR, providing brownout detect
- LVR for all system-relevant power domains
- Software-readable Low Voltage Status and Control register that contains flags that indicate low- or high-voltage conditions

## 42.2 Functional description

### 42.2.1 Modes of operation

PMC provides two basic modes of operation for voltage regulators and monitors:

- FPM supports chip run modes that have high current consumption.

- LPM supports chip standby modes that have low current consumption.

### 42.2.2 Reset

The POR and all LVRs combine into one single chip POR.

After a chip POR event, you can determine which power domain caused it by reading the PORF and LVR*x* flags in Low Voltage Status And Control (LVSC).

After an initial power ramp-up of the chip in LVSC, PMC sets the POR and LVR*x* flags. The go/no go flags have an arbitrary value.

**NOTE**

After an initial power ramp-up, you must clear all flags in LVSC by writing FFFF_FFFFh to that register.

Because the flags are sticky bits, you must clear them before using them. That way, if an unexpected chip POR occurs you can track and debug the source of the problem by reading the flags in LVSC.

### 42.2.3 Interrupts

PMC includes two interrupt sources:

- HVD interrupt: Combines all HVD monitors into one interrupt source. CONFIG[HVDIE] enables this interrupt. See PMC Configuration (CONFIG) and Low Voltage Status And Control (LVSC) for details.
- LVD interrupt: The LVD5A monitor is the only interrupt source. CONFIG[LVDIE] enables this interrupt. See PMC Configuration (CONFIG) and Low Voltage Status And Control (LVSC) for details.

## 42.3 Signals

**Table 214. Signals**

| Signal | Type | Description |
|---|---|---|
| VDD_HV_A | Supply input | The primary high-voltage supply input to PMC. VDD_HV_A is used for the PMC internal precision references. After the VDD_HV_A power domain is powered up, it must always be kept powered for both FPM and LPM. |
| V25 | Supply output | A fully integrated low-dropout linear voltage regulator drives the V25 power supply domain. V25 supplies the flash memory and (via a double bond) the clock modules. |
| V11 | Supply output | V11 is the core and logic supply. A fully integrated low-dropout linear voltage regulator drives V11. |
| VSS | Ground | VSS must be grounded. You must connect all VSS pins externally to the same ground node. |

## 42.4 PMC register descriptions

### 42.4.1 PMC memory map

DEFAULT_NICKNAME base address: 0h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Low Voltage Status And Control (LVSC) | 32 | RW | See section |
| 4h | PMC Configuration (CONFIG) | 32 | RW | 0000_0000h |
| Ch | Version ID (VERID) | 32 | R | 0300_0000h |

## 42.4.2 Low Voltage Status And Control (LVSC)

### Offset

| Register | Offset |
|---|---|
| LVSC | 0h |

### Function

Contains status and control fields that support the low-voltage reset and low- or high-voltage detect functions. When PMC is in LPM, the low- or high-voltage detect systems are disabled.

**NOTE**

For all flags that are not affected by reset (POR flag, all LVR flags, all GNG flags), if a reset occurs while trying to clear the flags (by writing 1), the flag value is not defined appropriately. In this case, you must clear the flag again after exiting from reset.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PORF | 0 | | | | | GNG11OS... | GNG25OS... | LVR11LPF | LVR11F | LVR25LPF | LVR25F | 0 | | LVRALPF | LVRAF |
| W | W1C | | | | | | W1C | W1C | W1C | W1C | W1C | W1C | | | W1C | W1C |
| Reset | u | 0 | 0 | 0 | 0 | 0 | u | u | u | u | u | u | 0 | 0 | u | u |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | LVD5AS | HVD11S | HVD25S | 0 | HVDAS | 0 | | | LVD5AF | HVD11F | HVD25F | 0 | HVDAF |
| W | | | | | | | | | | | | W1C | W1C | W1C | | W1C |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>PORF | POR Flag<br>Indicates that a power-on reset event has occurred. Other reset sources have no effect. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Event did not occur<br>1b - Event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 30-26<br>— | Reserved |
| 25<br>GNG11OSCF | Go/No Go Detect Flag On OSC Part Of V11 Power Domain<br>Indicates that the go/no go sensor has detected a low voltage in the V11 power domain in FPM. This applies only to the part of the power domain that supplies the 1.1 V clocking modules (for example, PLL). Other reset sources have no effect.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Event did not occur<br>1b - Event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 24<br>GNG25OSCF | Go/No Go Detect Flag On OSC Part of V25 Power Domain<br>Indicates that the go/no go sensor has detected a low voltage in the V25 power domain in FPM. This applies only to the part of the power domain that supplies the 2.5 V clocking modules (for example, XOSC and IRC). Other reset sources have no effect.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Event did not occur<br>1b - Event occurred<br>When writing<br>0b - No effect |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Clear the flag |
| 23<br>LVR11LPF | LVR11LP Flag On V11 Power Domain<br>Indicates that a low-voltage reset event has occurred in the 1.1 V V11 power domain (FPM or LPM).Other reset sources have no effect.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Event did not occur<br>1b - Event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 22<br>LVR11F | LVR11 Flag On V11 Power Domain In FPM<br>Indicates that a low-voltage reset event has occurred in the 1.1 V V11 power domain in FPM. Other reset sources have no effect.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Event did not occur<br>1b - Event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 21<br>LVR25LPF | LVR25LP Flag On V25 Power Domain<br>Indicates that a low-voltage reset event has occurred in the V25 power domain (FPM or LPM). Other reset sources have no effect.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Event did not occur<br>1b - Event occurred<br>When writing |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No effect<br>1b - Clear the flag |
| 20<br>LVR25F | LVR25 Flag On V25 Power Domain In FPM<br>Indicates that a low-voltage reset event has occurred in the V25 power domain in FPM. Other reset sources have no effect.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Event did not occur<br>1b - Event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 19-18<br>— | Reserved |
| 17<br>LVRALPF | LVRALP Flag On VDD_HV_A Power Domain<br>Indicates that a low-voltage reset event has occurred in the VDD_HV_A power domain (FPM or LPM). Other reset sources have no effect.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Event did not occur<br>1b - Event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 16<br>LVRAF | LVRA Flag On VDD_HV_A Power Domain In FPM<br>Indicates that a low-voltage reset event has occurred in the VDD_HV_A power domain in FPM. Other reset sources have no effect.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Event did not occur<br>1b - Event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 15-13<br>— | Reserved |
| 12<br>LVD5AS | LVD5A Status On VDD_HV_A Power Domain In FPM<br>Indicates whether the voltage on VDD_HV_A is above or below the low-voltage detect threshold. This monitor reflects the status of the 5 V low-voltage detect, LVD5A, and indicates if the voltage is below a certain threshold, which is set slightly below 4.5 V (see the chip data sheet for the exact value). The feature is only available in FPM and is disabled in LPM. After a reset or wake-up from LPM, you must clear the LVD5AF flag and check LVD5AS to determine the voltage level on the VDD_HV_A supply.<br>0b - Above<br>1b - Below |
| 11<br>HVD11S | HVD11 Status On V11 Power Domain In FPM<br>Indicates whether the voltage on V11 is above or below the high-voltage detect threshold. This field reflects the status of the high-voltage detect, HVD11, on the V11 power domain. This feature is only available in FPM and is disabled in LPM.<br>0b - Voltage is below threshold or chip is in LPM<br>1b - Voltage is above threshold and chip is in FPM |
| 10<br>HVD25S | HVD25 Status On V25 Power Domain In FPM<br>Indicates whether the voltage on V25 is above or below the high-voltage detect threshold. This field reflects the status of the high-voltage detect, HVD25, on the V25 power domain. The feature is only available in FPM and is disabled in LPM.<br>0b - Voltage is below threshold or chip is in LPM<br>1b - Voltage is above threshold and chip is in FPM |
| 9<br>— | Reserved |
| 8<br>HVDAS | HVDA Status On VDD_HV_A Power Domain In FPM<br>Indicates whether the voltage on VDD_HV_A is above or below the high-voltage detect threshold. This field reflects the status of the high-voltage detect, HVDA, on the V25 power domain. The feature is only available in FPM and is disabled in LPM.<br>0b - Voltage is below threshold or chip is in LPM<br>1b - Voltage is above threshold and chip is in FPM |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 7-5<br>— | Reserved |
| 4<br>LVD5AF | LVD5A Flag On VDD_HV_A Power Domain In FPM<br>Indicates whether LVD5AS has changed. When LVD5AS changes, PMC changes LVD5AF to 1.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Did not change<br>1b - Changed<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 3<br>HVD11F | HVD11 Flag On V11 Power Domain In FPM<br>Indicates whether HVD11S has changed. When HVD11S changes, PMC changes HVD11F to 1.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Did not change<br>1b - Changed<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 2<br>HVD25F | HVD25 Flag On V25 Power Domain In FPM<br>Indicates whether HVD25S has changed. When HVD25S changes, PMC changes HVD25F to 1.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Did not change<br>1b - Changed<br>When writing<br>0b - No effect<br>1b - Clear the flag |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 1<br>— | Reserved |
| 0<br>HVDAF | HVDA Flag On VDD_HV_A Power Domain In FPM<br>Indicates whether HVDAS has changed. When HVDAS changes, PMC changes HVDAF to 1.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Did not change<br>1b - Changed<br>When writing<br>0b - No effect<br>1b - Clear the flag |

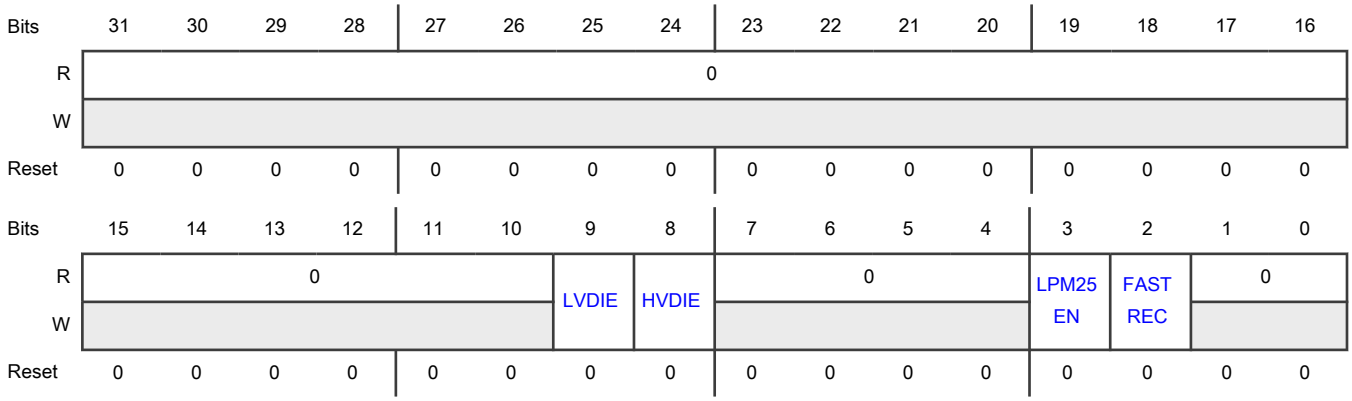
### 42.4.3 PMC Configuration (CONFIG)

#### Offset

| Register | Offset |
|---|---|
| CONFIG | 4h |

#### Function
Configures PMC.

#### Diagram



### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>LVDIE | Low Voltage Detect Interrupt Enable<br>Enables hardware interrupt requests if LVSC[LVD5AF] = 1. You must disable the LVD interrupt before entering LPM.<br>0b - LVD hardware interrupt is disabled (use polling)<br>1b - Request an LVD hardware interrupt when LVDA5F = 1 |
| 8<br>HVDIE | High Voltage Detect Interrupt Enable<br>Enables hardware interrupt requests if any of the following flags in Low Voltage Status And Control (LVSC) are set:<br>• HVDAF<br>• HVDBF<br>• HVD25F<br>• HVD11F<br>0b - HVD hardware interrupt is disabled (use polling)<br>1b - Request an HVD hardware interrupt when HVDAF=1, HVDBF=1, HVD25F=1, or HVD11F=1 |
| 7-4<br>— | Reserved |
| 3<br>LPM25EN | V25 Power Domain Enable During LPM<br>Controls whether the V25 regulator and low-voltage reset detection (LVR25LP) are enabled or disabled in LPM<br>0b - Disabled<br>1b - Enabled |
| 2<br>FASTREC | Fast Recovery From LPM Enable<br>Controls the recovery time from LPM to FPM.<br>At recovery from LPM, all the tank capacitors from the secondary supplies must be recharged. This causes a high-current demand, which the supply driving the VDD_HV_A primary power domain might not meet. When you select the fast recovery time, the current for recharging is approximately three times higher than that for normal recovery time. You must determine whether this current is sufficient to start up from LPM in time, using these criteria:<br>• Drive capability of the external VDD_HV_A regulator<br>• Size of the tank caps on the secondary supply pin<br>• Selected recovery time |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Normal<br>1b - Fast |
| 1-0<br>— | Reserved |

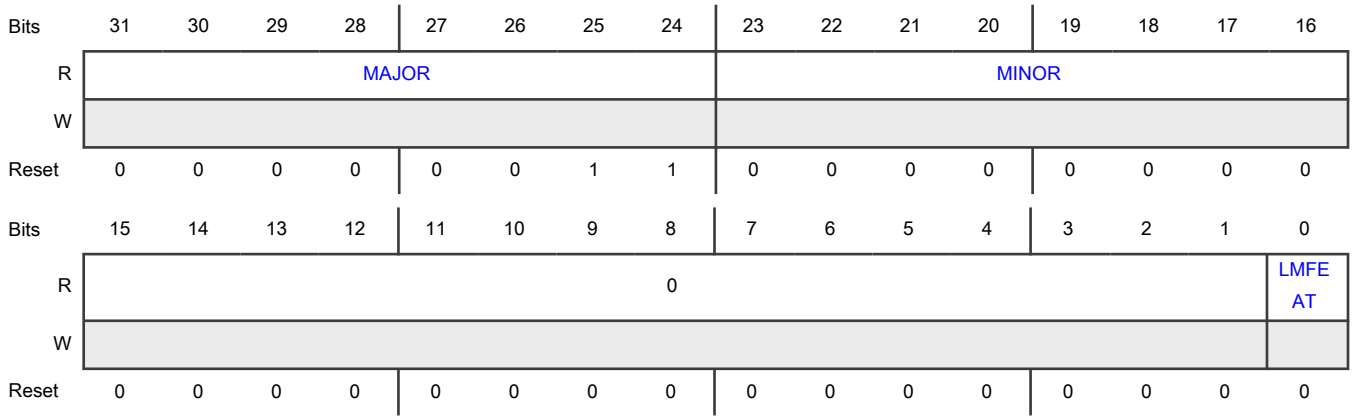
## 42.4.4 Version ID (VERID)

### Offset

| Register | Offset |
|---|---|
| VERID | Ch |

### Function
Records the specific PMC version in the chip.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>Indicates the major version number of the PMC design. |
| 23-16<br>MINOR | Minor Version Number<br>Indicates the minor version number of the PMC design. |
| 15-1 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 0<br>LMFEAT | Last-Mile Regulator Feature<br>Indicates whether the last-mile regulator (1.5 V to 1.1 V) is available.<br>0b - Not available<br>1b - Available |

## 42.5 Glossary

| | |
|---|---|
| **FPM** | Full Performance mode |
| **HVD** | High voltage detect |
| **LPM** | Low Power mode |
| **LVD** | Low voltage detect |
| **LVR** | Low voltage reset |
| **POR** | Power-on reset |

# Chapter 43
# Mode Entry Module (MC_ME)

## 43.1 Chip-specific MC_ME information

### 43.1.1 MC_ME modes

This chip implements these modes:

- Reset
- Run
- Standby

The chip always enters Run mode after exiting Reset mode wherein you can configure the chip to perform its computational and communication functions. In Run mode:

- The chip remains fully powered. The boot core is the only core enabled on Run entry.
- You can enable and disable application cores and peripherals as needed, based on functional and power requirements.
- You can configure the pins and self-test as needed. See "Self-Test programming sequence" section in the STCU2 chapter for the self-test programming sequence to be followed before initiating self-test, in which:
  - Pins are safe-stated.
  - No computational or communication activities are possible.
- The chip automatically enters Reset mode after self-test (BIST) is complete.

### 43.1.2 MC_ME partition mapping of cores and peripherals

MC_ME provides registers and interface signals to support multiple partitions. These MC_ME partitions are different from the chip's LBIST partitions described in the "Safety Overview" chapter. This chip has three MC_ME partitions:

- Partition 0: Contains application cores and the on/off-platform slots on AIPS_0 bridge
- Partition 1: Contains the on/off-platform slots on AIPS_1 bridge
- Partition 2: Contains the on/off-platform slots on AIPS_2 bridge

Table 215 and Table 216 specify the core and peripheral mapping on MC_ME partitions and their associated clock gating possiblities.

MC_ME also provides provisions to control the booting address for application cores, which can be configured to start from a nondefault address location by appropriately configuring PRTN*x*_CORE*n*_ADDR[ADDR].

### 43.1.3 Core clock gating

MC_ME has individual core-clock enable fields that gate the application core clocks, which can also be clock gated by executing Waiting for Interrupt (WFI).

You can enable the application cores by configuring the respective CCE fields. See Application core initialization process and Application core shutdown process for proper initialization and shutdown of application cores. There is no clock control for the ELE_HSEB core. It needs to be only put into WFI if required to be shutdown (for example, in Standby mode).

**Table 215. MC_ME partition core mapping**

| Core | MC_ME partition | MC_ME clock enable register field |
|---|---|---|
| Cortex-M7_0 | 0 | MC_ME.PRTN0_CORE0_PCONF[CCE] |

### 43.1.4 Peripheral clock gating

See Peripheral initialization process and Peripheral shutdown process for proper initialization and shutdown of peripherals. See Table 216 for peripheral clock gating possibilities.

The application core can program the reserved configurations.

**NOTE**

Before accessing the registers of a peripheral to start using it, its clock must be turned on, otherwise, a Hard-Fault event will occur.

**Table 216. MC_ME partition peripheral mapping and clock control**

| AIPS peripheral | Peripheral description[1] | MC_ME's PRTN*n*_COFB*n*_CLKEN register fields | MC_ME peripheral control register | MC_ME peripheral slot number in partition | MC_ME COFB present | MC_ME CLKEN present | MC_ME default CLKIN | On platform | IPS slot number |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ0] | 0 | 0 | 0 | 0 | 0 | Yes | 0 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ1] | 1 | 1 | 0 | 0 | 0 | Yes | 1 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ2] | 2 | 2 | 0 | 0 | 0 | Yes | 2 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ3] | 3 | 3 | 0 | 0 | 0 | Yes | 3 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ4] | 4 | 4 | 0 | 0 | 0 | Yes | 4 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ5] | 5 | 5 | 0 | 0 | 0 | Yes | 5 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ6] | 6 | 6 | 0 | 0 | 0 | Yes | 6 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ7] | 7 | 7 | 0 | 0 | 0 | Yes | 7 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ8] | 8 | 8 | 0 | 0 | 0 | Yes | 8 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ9] | 9 | 9 | 0 | 0 | 0 | Yes | 9 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ10] | 10 | 10 | 0 | 0 | 0 | Yes | 10 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ11] | 11 | 11 | 0 | 0 | 0 | Yes | 11 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ12] | 12 | 12 | 0 | 0 | 0 | Yes | 12 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ13] | 13 | 13 | 0 | 0 | 0 | Yes | 13 |

*Table continues on the next page...*

Table 216. MC_ME partition peripheral mapping and clock control (continued)

| AIPS peripheral | Peripheral description[1] | MC_ME's PRTN*n*_COFB*n*_CLKEN register fields | MC_ME peripheral control register | MC_ME peripheral slot number in partition | MC_ME COFB present | MC_ME CLKEN present | MC_ME default CLKIN | On platform | IPS slot number |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ14] | 14 | 14 | 0 | 0 | 0 | Yes | 14 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ15] | 15 | 15 | 0 | 0 | 0 | Yes | 15 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ16] | 16 | 16 | 0 | 0 | 0 | Yes | 16 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ17] | 17 | 17 | 0 | 0 | 0 | Yes | 17 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ18] | 18 | 18 | 0 | 0 | 0 | Yes | 18 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ19] | 19 | 19 | 0 | 0 | 0 | Yes | 19 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ20] | 20 | 20 | 0 | 0 | 0 | Yes | 20 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ21] | 21 | 21 | 0 | 0 | 0 | Yes | 21 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ22] | 22 | 22 | 0 | 0 | 0 | Yes | 22 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ23] | 23 | 23 | 0 | 0 | 0 | Yes | 23 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ24] | 24 | 24 | 0 | 0 | 0 | Yes | 24 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ25] | 25 | 25 | 0 | 0 | 0 | Yes | 25 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ26] | 26 | 26 | 0 | 0 | 0 | Yes | 26 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ27] | 27 | 27 | 0 | 0 | 0 | Yes | 27 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ28] | 28 | 28 | 0 | 0 | 0 | Yes | 28 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ29] | 29 | 29 | 0 | 0 | 0 | Yes | 29 |
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ30] | 30 | 30 | 0 | 0 | 0 | Yes | 30 |

*Table continues on the next page...*

Table 216. MC_ME partition peripheral mapping and clock control (continued)

| AIPS peripheral | Peripheral description[1] | MC_ME's PRTN*n*_COFB*n*_CLKEN register fields | MC_ME peripheral control register | MC_ME peripheral slot number in partition | MC_ME COFB present | MC_ME CLKEN present | MC_ME default CLKIN | On platform | IPS slot number |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Reserved | PRTN0_COFB0_CLKEN[REQ31] | 31 | 31 | 0 | 0 | 0 | Yes | 31 |
| 0 | TRGMUX | PRTN0_COFB1_CLKEN[REQ32] | 32 | 32 | 1 | 1 | 0 | No | 0 |
| 0 | BCTU | PRTN0_COFB1_CLKEN[REQ33] | 33 | 33 | 1 | 1 | 0 | No | 1 |
| 0 | eMIOS_0 | PRTN0_COFB1_CLKEN[REQ34] | 34 | 34 | 1 | 1 | 0 | No | 2 |
| 0 | eMIOS_1 | PRTN0_COFB1_CLKEN[REQ35] | 35 | 35 | 1 | 1 | 0 | No | 3 |
| 0 | eMIOS_2 | PRTN0_COFB1_CLKEN[REQ36] | 36 | 36 | 1 | 1 | 0 | No | 4 |
| 0 | Reserved | PRTN0_COFB1_CLKEN[REQ37] | 37 | 37 | 0 | 0 | 0 | No | 5 |
| 0 | LCU_0 | PRTN0_COFB1_CLKEN[REQ38] | 38 | 38 | 1 | 1 | 0 | No | 6 |
| 0 | LCU_1 | PRTN0_COFB1_CLKEN[REQ39] | 39 | 39 | 1 | 1 | 0 | No | 7 |
| 0 | ADC_0 | PRTN0_COFB1_CLKEN[REQ40] | 40 | 40 | 1 | 1 | 0 | No | 8 |
| 0 | ADC_1 | PRTN0_COFB1_CLKEN[REQ41] | 41 | 41 | 1 | 1 | 0 | No | 9 |
| 0 | ADC_2 | PRTN0_COFB1_CLKEN[REQ42] | 42 | 42 | 1 | 1 | 0 | No | 10 |
| 0 | Reserved | PRTN0_COFB1_CLKEN[REQ43] | 43 | 43 | 0 | 0 | 0 | No | 11 |
| 0 | PIT_0 | PRTN0_COFB1_CLKEN[REQ44] | 44 | 44 | 1 | 1 | 1 | No | 12 |
| 0 | PIT_1 | PRTN0_COFB1_CLKEN[REQ45] | 45 | 45 | 1 | 1 | 0 | No | 13 |
| 0 | MU_A | PRTN0_COFB1_CLKEN[REQ46] | 46 | 46 | 1 | 1 | 0 | No | 14 |
| 0 | MU_B | PRTN0_COFB1_CLKEN[REQ47] | 47 | 47 | 1 | 1 | 0 | No | 15 |

*Table continues on the next page...*

Table 216. MC_ME partition peripheral mapping and clock control (continued)

| AIPS peripheral | Peripheral description[1] | MC_ME's PRTN*n*_COFB*n*_CLKEN register fields | MC_ME peripheral control register | MC_ME peripheral slot number in partition | MC_ME COFB present | MC_ME CLKEN present | MC_ME default CLKIN | On platform | IPS slot number |
|---|---|---|---|---|---|---|---|---|---|
| 1 | AXBS switch | PRTN1_COFB0_CLKEN[REQ0] | 128 | 0 | 1 | 0 | 1 | Yes | 0 |
| 1 | AXBS_0 | PRTN1_COFB0_CLKEN[REQ1] | 129 | 1 | 1 | 0 | 1 | Yes | 1 |
| 1 | AXBS_1 | PRTN1_COFB0_CLKEN[REQ2] | 130 | 2 | 1 | 0 | 1 | Yes | 2 |
| 1 | eDMA control and status (MP_CSR; MP_ES; MP_HRS) | PRTN1_COFB0_CLKEN[REQ3] | 131 | 3 | 1 | 1 | 0 | Yes | 3 |
| 1 | eDMA transfer control descriptor 0 | PRTN1_COFB0_CLKEN[REQ4] | 132 | 4 | 1 | 1 | 0 | Yes | 4 |
| 1 | eDMA transfer control descriptor 1 | PRTN1_COFB0_CLKEN[REQ5] | 133 | 5 | 1 | 1 | 0 | Yes | 5 |
| 1 | eDMA transfer control descriptor 2 | PRTN1_COFB0_CLKEN[REQ6] | 134 | 6 | 1 | 1 | 0 | Yes | 6 |
| 1 | eDMA transfer control descriptor 3 | PRTN1_COFB0_CLKEN[REQ7] | 135 | 7 | 1 | 1 | 0 | Yes | 7 |
| 1 | eDMA transfer control descriptor 4 | PRTN1_COFB0_CLKEN[REQ8] | 136 | 8 | 1 | 1 | 0 | Yes | 8 |
| 1 | eDMA transfer control descriptor 5 | PRTN1_COFB0_CLKEN[REQ9] | 137 | 9 | 1 | 1 | 0 | Yes | 9 |
| 1 | eDMA transfer control descriptor 6 | PRTN1_COFB0_CLKEN[REQ10] | 138 | 10 | 1 | 1 | 0 | Yes | 10 |
| 1 | eDMA transfer control descriptor 7 | PRTN1_COFB0_CLKEN[REQ11] | 139 | 11 | 1 | 1 | 0 | Yes | 11 |
| 1 | eDMA transfer control descriptor 8 | PRTN1_COFB0_CLKEN[REQ12] | 140 | 12 | 1 | 1 | 0 | Yes | 12 |
| 1 | eDMA transfer control descriptor 9 | PRTN1_COFB0_CLKEN[REQ13] | 141 | 13 | 1 | 1 | 0 | Yes | 13 |
| 1 | eDMA transfer control descriptor 10 | PRTN1_COFB0_CLKEN[REQ14] | 142 | 14 | 1 | 1 | 0 | Yes | 14 |
| 1 | eDMA transfer control descriptor 11 | PRTN1_COFB0_CLKEN[REQ15] | 143 | 15 | 1 | 1 | 0 | Yes | 15 |

*Table continues on the next page...*

Table 216. MC_ME partition peripheral mapping and clock control (continued)

| AIPS peripheral | Peripheral description[1] | MC_ME's PRTN*n*_COFB*n*_CLKEN register fields | MC_ME peripheral control register | MC_ME peripheral slot number in partition | MC_ME COFB present | MC_ME CLKEN present | MC_ME default CLKIN | On platform | IPS slot number |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Debug APB page0 | PRTN1_COFB0_CLKEN[REQ16] | 144 | 16 | 1 | 0 | 1 | Yes | 16 |
| 1 | Debug APB page1 | PRTN1_COFB0_CLKEN[REQ17] | 145 | 17 | 1 | 0 | 1 | Yes | 17 |
| 1 | Debug APB page2 | PRTN1_COFB0_CLKEN[REQ18] | 146 | 18 | 1 | 0 | 1 | Yes | 18 |
| 1 | Debug APB page3 | PRTN1_COFB0_CLKEN[REQ19] | 147 | 19 | 1 | 0 | 1 | Yes | 19 |
| 1 | Debug APB paged area | PRTN1_COFB0_CLKEN[REQ20] | 148 | 20 | 1 | 0 | 1 | Yes | 20 |
| 1 | SDA_AP | PRTN1_COFB0_CLKEN[REQ21] | 149 | 21 | 1 | 1 | 1 | Yes | 21 |
| 1 | EIM | PRTN1_COFB0_CLKEN[REQ22] | 150 | 22 | 1 | 1 | 0 | Yes | 22 |
| 1 | ERM | PRTN1_COFB0_CLKEN[REQ23] | 151 | 23 | 1 | 1 | 0 | Yes | 23 |
| 1 | MSCM | PRTN1_COFB0_CLKEN[REQ24] | 152 | 24 | 1 | 1 | 0 | Yes | 24 |
| 1 | PRAMC_0 | PRTN1_COFB0_CLKEN[REQ25] | 153 | 25 | 1 | 0 | 1 | Yes | 25 |
| 1 | Flash memory controller | PRTN1_COFB0_CLKEN[REQ26] | 154 | 26 | 1 | 0 | 1 | Yes | 26 |
| 1 | Flash memory controller alternate | PRTN1_COFB0_CLKEN[REQ27] | 155 | 27 | 1 | 0 | 1 | Yes | 27 |
| 1 | SWT_0 | PRTN1_COFB0_CLKEN[REQ28] | 156 | 28 | 1 | 1 | 1 | Yes | 28 |
| 1 | STM_0 | PRTN1_COFB0_CLKEN[REQ29] | 157 | 29 | 1 | 1 | 0 | Yes | 29 |
| 1 | XRDC | PRTN1_COFB0_CLKEN[REQ30] | 158 | 30 | 1 | 0 | 1 | Yes | 30 |
| 1 | INTM | PRTN1_COFB0_CLKEN[REQ31] | 159 | 31 | 1 | 1 | 0 | Yes | 31 |
| 1 | DMAMUX_0 | PRTN1_COFB1_CLKEN[REQ32] | 160 | 32 | 1 | 1 | 0 | No | 0 |

*Table continues on the next page...*

Table 216. MC_ME partition peripheral mapping and clock control (continued)

| AIPS peripheral | Peripheral description[1] | MC_ME's PRTN*n*_COFB*n*_CLKEN register fields | MC_ME peripheral control register | MC_ME peripheral slot number in partition | MC_ME COFB present | MC_ME CLKEN present | MC_ME default CLKIN | On platform | IPS slot number |
|---|---|---|---|---|---|---|---|---|---|
| 1 | DMAMUX_1 | PRTN1_COFB1_CLKEN[REQ33] | 161 | 33 | 1 | 1 | 0 | No | 1 |
| 1 | RTC | PRTN1_COFB1_CLKEN[REQ34] | 162 | 34 | 1 | 1 | 1 | No | 2 |
| 1 | MC_RGM | PRTN1_COFB1_CLKEN[REQ35] | 163 | 35 | 1 | 0 | 1 | No | 3 |
| 1 | SIUL2_VIRTWRAPPER_PDAC0 | PRTN1_COFB1_CLKEN[REQ36] | 164 | 36 | 1 | 0 | 1 | No | 4 |
| 1 | SIUL2_VIRTWRAPPER_PDAC0 | PRTN1_COFB1_CLKEN[REQ37] | 165 | 37 | 1 | 0 | 1 | No | 5 |
| 1 | SIUL2_VIRTWRAPPER_PDAC1 | PRTN1_COFB1_CLKEN[REQ38] | 166 | 38 | 1 | 0 | 1 | No | 6 |
| 1 | SIUL2_VIRTWRAPPER_PDAC1 | PRTN1_COFB1_CLKEN[REQ39] | 167 | 39 | 1 | 0 | 1 | No | 7 |
| 1 | SIUL2_VIRTWRAPPER_PDAC2 | PRTN1_COFB1_CLKEN[REQ40] | 168 | 40 | 1 | 0 | 1 | No | 8 |
| 1 | SIUL2_VIRTWRAPPER_PDAC2 | PRTN1_COFB1_CLKEN[REQ41] | 169 | 41 | 1 | 0 | 1 | No | 9 |
| 1 | SIUL2_VIRTWRAPPER | PRTN1_COFB1_CLKEN[REQ42] | 170 | 42 | 1 | 1 | 1 | No | 10 |
| 1 | SSCM | PRTN1_COFB1_CLKEN[REQ43] | 171 | 43 | 1 | 0 | 1 | No | 11 |
| 1 | Reserved | PRTN1_COFB1_CLKEN[REQ44] | 172 | 44 | 0 | 0 | 0 | No | 12 |
| 1 | WKPU | PRTN1_COFB1_CLKEN[REQ45] | 173 | 45 | 1 | 1 | 1 | No | 13 |
| 1 | Reserved | PRTN1_COFB1_CLKEN[REQ46] | 174 | 46 | 0 | 0 | 0 | No | 14 |
| 1 | CMU 0-5 | PRTN1_COFB1_CLKEN[REQ47] | 175 | 47 | 1 | 1 | 0 | No | 15 |
| 1 | Reserved | PRTN1_COFB1_CLKEN[REQ48] | 176 | 48 | 0 | 0 | 0 | No | 16 |
| 1 | TSPC | PRTN1_COFB1_CLKEN[REQ49] | 177 | 49 | 1 | 1 | 1 | No | 17 |

*Table continues on the next page...*

Table 216. MC_ME partition peripheral mapping and clock control (continued)

| AIPS peripheral | Peripheral description[1] | MC_ME's PRTN*n*_COFB*n*_CLKEN register fields | MC_ME peripheral control register | MC_ME peripheral slot number in partition | MC_ME COFB present | MC_ME CLKEN present | MC_ME default CLKIN | On platform | IPS slot number |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 32 kHz SIRC | PRTN1_COFB1_CLKEN[REQ50] | 178 | 50 | 1 | 0 | 1 | No | 18 |
| 1 | 32 kHz SXOSC | PRTN1_COFB1_CLKEN[REQ51] | 179 | 51 | 1 | 1 | 1 | No | 19 |
| 1 | 48 MHz FIRC | PRTN1_COFB1_CLKEN[REQ52] | 180 | 52 | 1 | 0 | 1 | No | 20 |
| 1 | 8-40 MHz FXOSC | PRTN1_COFB1_CLKEN[REQ53] | 181 | 53 | 1 | 1 | 1 | No | 21 |
| 1 | MC_CGM | PRTN1_COFB1_CLKEN[REQ54] | 182 | 54 | 1 | 0 | 1 | No | 22 |
| 1 | MC_ME | PRTN1_COFB1_CLKEN[REQ55] | 183 | 55 | 1 | 0 | 1 | No | 23 |
| 1 | Frequency-modulated PLL | PRTN1_COFB1_CLKEN[REQ56] | 184 | 56 | 1 | 1 | 0 | No | 24 |
| 1 | Reserved | PRTN1_COFB1_CLKEN[REQ57] | 185 | 57 | 0 | 0 | 0 | No | 25 |
| 1 | PMC | PRTN1_COFB1_CLKEN[REQ58] | 186 | 58 | 1 | 0 | 1 | No | 26 |
| 1 | Flash memory controller | PRTN1_COFB1_CLKEN[REQ59] | 187 | 59 | 1 | 0 | 1 | No | 27 |
| 1 | Flash memory controller alternate | PRTN1_COFB1_CLKEN[REQ60] | 188 | 60 | 1 | 0 | 1 | No | 28 |
| 1 | Reserved | PRTN1_COFB1_CLKEN[REQ61] | 189 | 61 | 0 | 0 | 0 | No | 29 |
| 1 | Reserved | PRTN1_COFB1_CLKEN[REQ62] | 190 | 62 | 0 | 0 | 0 | No | 30 |
| 1 | PIT_2 | PRTN1_COFB1_CLKEN[REQ63] | 191 | 63 | 1 | 1 | 0 | No | 31 |
| 1 | Reserved | PRTN1_COFB2_CLKEN[REQ64] | 192 | 64 | 0 | 0 | 0 | No | 32 |
| 1 | FlexCAN_0 | PRTN1_COFB2_CLKEN[REQ65] | 193 | 65 | 1 | 1 | 0 | No | 33 |
| 1 | FlexCAN_1 | PRTN1_COFB2_CLKEN[REQ66] | 194 | 66 | 1 | 1 | 0 | No | 34 |

*Table continues on the next page...*

Table 216. MC_ME partition peripheral mapping and clock control (continued)

| AIPS peripheral | Peripheral description[1] | MC_ME's PRTN*n*_COFB*n*_CLKEN register fields | MC_ME peripheral control register | MC_ME peripheral slot number in partition | MC_ME COFB present | MC_ME CLKEN present | MC_ME default CLKIN | On platform | IPS slot number |
|---|---|---|---|---|---|---|---|---|---|
| 1 | FlexCAN_2 | PRTN1_COFB2_CLKEN[REQ67] | 195 | 67 | 1 | 1 | 0 | No | 35 |
| 1 | FlexCAN_3 | PRTN1_COFB2_CLKEN[REQ68] | 196 | 68 | 1 | 1 | 0 | No | 36 |
| 1 | FlexCAN_4 | PRTN1_COFB2_CLKEN[REQ69] | 197 | 69 | 1 | 1 | 0 | No | 37 |
| 1 | FlexCAN_5 | PRTN1_COFB2_CLKEN[REQ70] | 198 | 70 | 1 | 1 | 0 | No | 38 |
| 1 | Reserved | PRTN1_COFB2_CLKEN[REQ71] | 199 | 71 | 0 | 0 | 0 | No | 39 |
| 1 | Reserved | PRTN1_COFB2_CLKEN[REQ72] | 200 | 72 | 0 | 0 | 0 | No | 40 |
| 1 | FlexIO | PRTN1_COFB2_CLKEN[REQ73] | 201 | 73 | 1 | 1 | 0 | No | 41 |
| 1 | LPUART_0 | PRTN1_COFB2_CLKEN[REQ74] | 202 | 74 | 1 | 1 | 0 | No | 42 |
| 1 | LPUART_1 | PRTN1_COFB2_CLKEN[REQ75] | 203 | 75 | 1 | 1 | 0 | No | 43 |
| 1 | LPUART_2 | PRTN1_COFB2_CLKEN[REQ76] | 204 | 76 | 1 | 1 | 0 | No | 44 |
| 1 | LPUART_3 | PRTN1_COFB2_CLKEN[REQ77] | 205 | 77 | 1 | 1 | 0 | No | 45 |
| 1 | LPUART_4 | PRTN1_COFB2_CLKEN[REQ78] | 206 | 78 | 1 | 1 | 0 | No | 46 |
| 1 | LPUART_5 | PRTN1_COFB2_CLKEN[REQ79] | 207 | 79 | 1 | 1 | 0 | No | 47 |
| 1 | LPUART_6 | PRTN1_COFB2_CLKEN[REQ80] | 208 | 80 | 1 | 1 | 0 | No | 48 |
| 1 | LPUART_7 | PRTN1_COFB2_CLKEN[REQ81] | 209 | 81 | 1 | 1 | 0 | No | 49 |
| 1 | Reserved | PRTN1_COFB2_CLKEN[REQ82] | 210 | 82 | 0 | 0 | 0 | No | 50 |
| 1 | Reserved | PRTN1_COFB2_CLKEN[REQ83] | 211 | 83 | 0 | 0 | 0 | No | 51 |

*Table continues on the next page...*

Table 216. MC_ME partition peripheral mapping and clock control (continued)

| AIPS peripheral | Peripheral description[1] | MC_ME's PRTN*n*_COFB*n*_CLKEN register fields | MC_ME peripheral control register | MC_ME peripheral slot number in partition | MC_ME COFB present | MC_ME CLKEN present | MC_ME default CLKIN | On platform | IPS slot number |
|---|---|---|---|---|---|---|---|---|---|
| 1 | LPI2C_0 | PRTN1_COFB2_CLKEN[REQ84] | 212 | 84 | 1 | 1 | 0 | No | 52 |
| 1 | LPI2C_1 | PRTN1_COFB2_CLKEN[REQ85] | 213 | 85 | 1 | 1 | 0 | No | 53 |
| 1 | LPSPI_0 | PRTN1_COFB2_CLKEN[REQ86] | 214 | 86 | 1 | 1 | 0 | No | 54 |
| 1 | LPSPI_1 | PRTN1_COFB2_CLKEN[REQ87] | 215 | 87 | 1 | 1 | 0 | No | 55 |
| 1 | LPSPI_2 | PRTN1_COFB2_CLKEN[REQ88] | 216 | 88 | 1 | 1 | 0 | No | 56 |
| 1 | LPSPI_3 | PRTN1_COFB2_CLKEN[REQ89] | 217 | 89 | 1 | 1 | 0 | No | 57 |
| 1 | Reserved | PRTN1_COFB2_CLKEN[REQ90] | 218 | 90 | 0 | 0 | 0 | No | 58 |
| 1 | SAI_0 | PRTN1_COFB2_CLKEN[REQ91] | 219 | 91 | 1 | 1 | 0 | No | 59 |
| 1 | LPCMP_0 | PRTN1_COFB2_CLKEN[REQ92] | 220 | 92 | 1 | 1 | 1 | No | 60 |
| 1 | LPCMP_1 | PRTN1_COFB2_CLKEN[REQ93] | 221 | 93 | 1 | 1 | 1 | No | 61 |
| 1 | Reserved | PRTN1_COFB2_CLKEN[REQ94] | 222 | 94 | 0 | 0 | 0 | No | 62 |
| 1 | TempSense | PRTN1_COFB2_CLKEN[REQ95] | 223 | 95 | 1 | 1 | 0 | No | 63 |
| 1 | CRC | PRTN1_COFB3_CLKEN[REQ96] | 224 | 96 | 1 | 1 | 0 | No | 64 |
| 1 | FCCU (+FOSU) | PRTN1_COFB3_CLKEN[REQ97] | 225 | 97 | 1 | 0 | 1 | No | 65 |
| 1 | MTR | PRTN1_COFB3_CLKEN[REQ98] | 226 | 98 | 1 | 0 | 1 | No | 66 |
| 1 | HSE_MU0_B | PRTN1_COFB3_CLKEN[REQ99] | 227 | 99 | 1 | 0 | 1 | No | 67 |
| 1 | Reserved | PRTN1_COFB3_CLKEN[REQ100] | 228 | 100 | 0 | 0 | 0 | No | 68 |

*Table continues on the next page...*

Table 216. MC_ME partition peripheral mapping and clock control (continued)

| AIPS peripheral | Peripheral description[1] | MC_ME's PRTN*n*_COFB*n*_CLKEN register fields | MC_ME peripheral control register | MC_ME peripheral slot number in partition | MC_ME COFB present | MC_ME CLKEN present | MC_ME default CLKIN | On platform | IPS slot number |
|---|---|---|---|---|---|---|---|---|---|
| 1 | JDC | PRTN1_COFB3_CLKEN[REQ101] | 229 | 101 | 1 | 0 | 1 | No | 69 |
| 1 | Reserved | PRTN1_COFB3_CLKEN[REQ102] | 230 | 102 | 1 | 1 | 1 | No | 70 |
| 1 | Configuration GPR | PRTN1_COFB3_CLKEN[REQ103] | 231 | 103 | 1 | 0 | 1 | No | 71 |
| 1 | STCU2 | PRTN1_COFB3_CLKEN[REQ104] | 232 | 104 | 1 | 1 | 1 | No | 72 |
| 1 | Reserved | PRTN1_COFB3_CLKEN[REQ105] | 233 | 105 | 1 | 0 | 1 | No | 73 |
| 1 | Reserved | PRTN1_COFB3_CLKEN[REQ106] | 234 | 106 | 1 | 0 | 1 | No | 74 |
| 1 | Reserved | PRTN1_COFB3_CLKEN[REQ107] | 235 | 107 | 1 | 0 | 1 | No | 75 |
| 1 | SELFTEST_GPR[2] | PRTN1_COFB3_CLKEN[REQ108] | 236 | 108 | 1 | 0 | 1 | No | 76 |
| 1 | Reserved | PRTN1_COFB3_CLKEN[REQ109] | 237 | 109 | 0 | 0 | 0 | No | 77 |
| 1 | Reserved | PRTN1_COFB3_CLKEN[REQ110] | 238 | 110 | 1 | 0 | 1 | No | 78 |
| 2 | XBIC (TCM backdoor AHB_splitter) | PRTN2_COFB0_CLKEN[REQ0] | 256 | 0 | 1 | 0 | 1 | Yes | 0 |
| 2 | XBIC (eDMA AXBS-Lite) | PRTN2_COFB0_CLKEN[REQ1] | 257 | 1 | 1 | 0 | 1 | Yes | 1 |
| 2 | Reserved | PRTN2_COFB0_CLKEN[REQ2] | 258 | 2 | 0 | 0 | 0 | Yes | 2 |
| 2 | Reserved | PRTN2_COFB0_CLKEN[REQ3] | 259 | 3 | 0 | 0 | 0 | Yes | 3 |
| 2 | eDMA transfer control descriptor 12 | PRTN2_COFB0_CLKEN[REQ4] | 260 | 4 | 1 | 1 | 0 | Yes | 4 |
| 2 | eDMA transfer control descriptor 13 | PRTN2_COFB0_CLKEN[REQ5] | 261 | 5 | 1 | 1 | 0 | Yes | 5 |

*Table continues on the next page...*

Table 216. MC_ME partition peripheral mapping and clock control (continued)

| AIPS peripheral | Peripheral description[1] | MC_ME's PRTN*n*_COFB*n*_CLKEN register fields | MC_ME peripheral control register | MC_ME peripheral slot number in partition | MC_ME COFB present | MC_ME CLKEN present | MC_ME default CLKIN | On platform | IPS slot number |
|---|---|---|---|---|---|---|---|---|---|
| 2 | eDMA transfer control descriptor 14 | PRTN2_COFB0_CLKEN[REQ6] | 262 | 6 | 1 | 1 | 0 | Yes | 6 |
| 2 | eDMA transfer control descriptor 15 | PRTN2_COFB0_CLKEN[REQ7] | 263 | 7 | 1 | 1 | 0 | Yes | 7 |
| 2 | eDMA transfer control descriptor 16 | PRTN2_COFB0_CLKEN[REQ8] | 264 | 8 | 1 | 1 | 0 | Yes | 8 |
| 2 | eDMA transfer control descriptor 17 | PRTN2_COFB0_CLKEN[REQ9] | 265 | 9 | 1 | 1 | 0 | Yes | 9 |
| 2 | eDMA transfer control descriptor 18 | PRTN2_COFB0_CLKEN[REQ10] | 266 | 10 | 1 | 1 | 0 | Yes | 10 |
| 2 | eDMA transfer control descriptor 19 | PRTN2_COFB0_CLKEN[REQ11] | 267 | 11 | 1 | 1 | 0 | Yes | 11 |
| 2 | eDMA transfer control descriptor 20 | PRTN2_COFB0_CLKEN[REQ12] | 268 | 12 | 1 | 1 | 0 | Yes | 12 |
| 2 | eDMA transfer control descriptor 21 | PRTN2_COFB0_CLKEN[REQ13] | 269 | 13 | 1 | 1 | 0 | Yes | 13 |
| 2 | eDMA transfer control descriptor 22 | PRTN2_COFB0_CLKEN[REQ14] | 270 | 14 | 1 | 1 | 0 | Yes | 14 |
| 2 | eDMA transfer control descriptor 23 | PRTN2_COFB0_CLKEN[REQ15] | 271 | 15 | 1 | 1 | 0 | Yes | 15 |
| 2 | eDMA transfer control descriptor 24 | PRTN2_COFB0_CLKEN[REQ16] | 272 | 16 | 1 | 1 | 0 | Yes | 16 |
| 2 | eDMA transfer control descriptor 25 | PRTN2_COFB0_CLKEN[REQ17] | 273 | 17 | 1 | 1 | 0 | Yes | 17 |

*Table continues on the next page...*

Table 216. MC_ME partition peripheral mapping and clock control (continued)

| AIPS peripheral | Peripheral description[1] | MC_ME's PRTN*n*_COFB*n*_CLKEN register fields | MC_ME peripheral control register | MC_ME peripheral slot number in partition | MC_ME COFB present | MC_ME CLKEN present | MC_ME default CLKIN | On platform | IPS slot number |
|---|---|---|---|---|---|---|---|---|---|
| 2 | eDMA transfer control descriptor 26 | PRTN2_COFB0_CLKEN[REQ18] | 274 | 18 | 1 | 1 | 0 | Yes | 18 |
| 2 | eDMA transfer control descriptor 27 | PRTN2_COFB0_CLKEN[REQ19] | 275 | 19 | 1 | 1 | 0 | Yes | 19 |
| 2 | eDMA transfer control descriptor 28 | PRTN2_COFB0_CLKEN[REQ20] | 276 | 20 | 1 | 1 | 0 | Yes | 20 |
| 2 | eDMA transfer control descriptor 29 | PRTN2_COFB0_CLKEN[REQ21] | 277 | 21 | 1 | 1 | 0 | Yes | 21 |
| 2 | eDMA transfer control descriptor 30 | PRTN2_COFB0_CLKEN[REQ22] | 278 | 22 | 1 | 1 | 0 | Yes | 22 |
| 2 | eDMA transfer control descriptor 31 | PRTN2_COFB0_CLKEN[REQ23] | 279 | 23 | 1 | 1 | 0 | Yes | 23 |
| 2 | SEMA42 | PRTN2_COFB0_CLKEN[REQ24] | 280 | 24 | 1 | 1 | 0 | Yes | 24 |
| 2 | PRAMC_1 | PRTN2_COFB0_CLKEN[REQ25] | 281 | 25 | 1 | 0 | 1 | Yes | 25 |
| 2 | Reserved | PRTN2_COFB0_CLKEN[REQ26] | 282 | 26 | 0 | 0 | 0 | Yes | 26 |
| 2 | SWT_1 | PRTN2_COFB0_CLKEN[REQ27] | 283 | 27 | 1 | 1 | 0 | Yes | 27 |
| 2 | Reserved | PRTN2_COFB0_CLKEN[REQ28] | 284 | 28 | 0 | 0 | 0 | Yes | 28 |
| 2 | STM_1 | PRTN2_COFB0_CLKEN[REQ29] | 285 | 29 | 1 | 1 | 0 | Yes | 29 |
| 2 | Reserved | PRTN2_COFB0_CLKEN[REQ30] | 286 | 30 | 0 | 0 | 0 | Yes | 30 |
| 2 | Reserved | PRTN2_COFB0_CLKEN[REQ31] | 287 | 31 | 0 | 0 | 0 | Yes | 31 |
| 2 | EMAC | PRTN2_COFB1_CLKEN[REQ32] | 288 | 32 | 1 | 1 | 0 | No | 0 |

*Table continues on the next page...*

Table 216. MC_ME partition peripheral mapping and clock control (continued)

| AIPS peripheral | Peripheral description[1] | MC_ME's PRTN*n*_COFB*n*_CLKEN register fields | MC_ME peripheral control register | MC_ME peripheral slot number in partition | MC_ME COFB present | MC_ME CLKEN present | MC_ME default CLKIN | On platform | IPS slot number |
|---|---|---|---|---|---|---|---|---|---|
| 2 | Reserved | PRTN2_COFB1_CLKEN[REQ33] | 289 | 33 | 0 | 0 | 0 | No | 1 |
| 2 | Reserved | PRTN2_COFB1_CLKEN[REQ34] | 290 | 34 | 0 | 0 | 0 | No | 2 |
| 2 | LPUART_8 | PRTN2_COFB1_CLKEN[REQ35] | 291 | 35 | 1 | 1 | 0 | No | 3 |
| 2 | LPUART_9 | PRTN2_COFB1_CLKEN[REQ36] | 292 | 36 | 1 | 1 | 0 | No | 4 |
| 2 | LPUART_10 | PRTN2_COFB1_CLKEN[REQ37] | 293 | 37 | 1 | 1 | 0 | No | 5 |
| 2 | LPUART_11 | PRTN2_COFB1_CLKEN[REQ38] | 294 | 38 | 1 | 1 | 0 | No | 6 |
| 2 | LPUART_12 | PRTN2_COFB1_CLKEN[REQ39] | 295 | 39 | 1 | 1 | 0 | No | 7 |
| 2 | LPUART_13 | PRTN2_COFB1_CLKEN[REQ40] | 296 | 40 | 1 | 1 | 0 | No | 8 |
| 2 | LPUART_14 | PRTN2_COFB1_CLKEN[REQ41] | 297 | 41 | 1 | 1 | 0 | No | 9 |
| 2 | LPUART_15 | PRTN2_COFB1_CLKEN[REQ42] | 298 | 42 | 1 | 1 | 0 | No | 10 |
| 2 | Reserved | PRTN2_COFB1_CLKEN[REQ43] | 299 | 43 | 0 | 0 | 0 | No | 11 |
| 2 | Reserved | PRTN2_COFB1_CLKEN[REQ44] | 300 | 44 | 0 | 0 | 0 | No | 12 |
| 2 | Reserved | PRTN2_COFB1_CLKEN[REQ45] | 301 | 45 | 0 | 0 | 0 | No | 13 |
| 2 | Reserved | PRTN2_COFB1_CLKEN[REQ46] | 302 | 46 | 0 | 0 | 0 | No | 14 |
| 2 | LPSPI_4 | PRTN2_COFB1_CLKEN[REQ47] | 303 | 47 | 1 | 1 | 0 | No | 15 |
| 2 | LPSPI_5 | PRTN2_COFB1_CLKEN[REQ48] | 304 | 48 | 1 | 1 | 0 | No | 16 |
| 2 | Reserved | PRTN2_COFB1_CLKEN[REQ49] | 305 | 49 | 0 | 0 | 0 | No | 17 |

*Table continues on the next page...*

Table 216. MC_ME partition peripheral mapping and clock control (continued)

| AIPS peripheral | Peripheral description[1] | MC_ME's PRTN*n*_COFB*n*_CLKEN register fields | MC_ME peripheral control register | MC_ME peripheral slot number in partition | MC_ME COFB present | MC_ME CLKEN present | MC_ME default CLKIN | On platform | IPS slot number |
|---|---|---|---|---|---|---|---|---|---|
| 2 | Reserved | PRTN2_COFB1_CLKEN[REQ50] | 306 | 50 | 0 | 0 | 0 | No | 18 |
| 2 | QuadSPI | PRTN2_COFB1_CLKEN[REQ51] | 307 | 51 | 1 | 1 | 0 | No | 19 |
| 2 | Reserved | PRTN2_COFB1_CLKEN[REQ52] | 308 | 52 | 0 | 0 | 0 | No | 20 |
| 2 | Reserved | PRTN2_COFB1_CLKEN[REQ53] | 309 | 53 | 0 | 0 | 0 | No | 21 |
| 2 | Reserved | PRTN2_COFB1_CLKEN[REQ54] | 310 | 54 | 0 | 0 | 0 | No | 22 |
| 2 | SAI_1 | PRTN2_COFB1_CLKEN[REQ55] | 311 | 55 | 1 | 1 | 0 | No | 23 |
| 2 | Reserved | PRTN2_COFB1_CLKEN[REQ56] | 312 | 56 | 0 | 0 | 0 | No | 24 |
| 2 | Reserved | PRTN2_COFB1_CLKEN[REQ57] | 313 | 57 | 0 | 0 | 0 | No | 25 |
| 2 | LPCMP_2 | PRTN2_COFB1_CLKEN[REQ58] | 314 | 58 | 1 | 1 | 1 | No | 26 |
| 2 | HSE_MU1_B | PRTN2_COFB1_CLKEN[REQ59] | 315 | 59 | 1 | 0 | 1 | No | 27 |
| 2 | CM7_0_TCM_CLKEN | PRTN2_COFB1_CLKEN[REQ62] | 318 | 62 | 1 | 1 | 1 | No | 30 |
| 2 | CM7_1_TCM_CLKEN | PRTN2_COFB1_CLKEN[REQ63] | 319 | 63 | 1 | 1 | 1 | No | 31 |

1. See the memory-map sheet attached to this document for details on available peripherals in different chips.
2. While accessing SELFTEST_GPR space, software must ensure that the STCU2 block has its clock enabled. i.e., MC_ME.PRTN1_COFB3_CLKEN[REQ104] configured as 1'b1. Not ensuring this might result in unpredictable device behaviour.

### 43.1.5 Application core initialization process

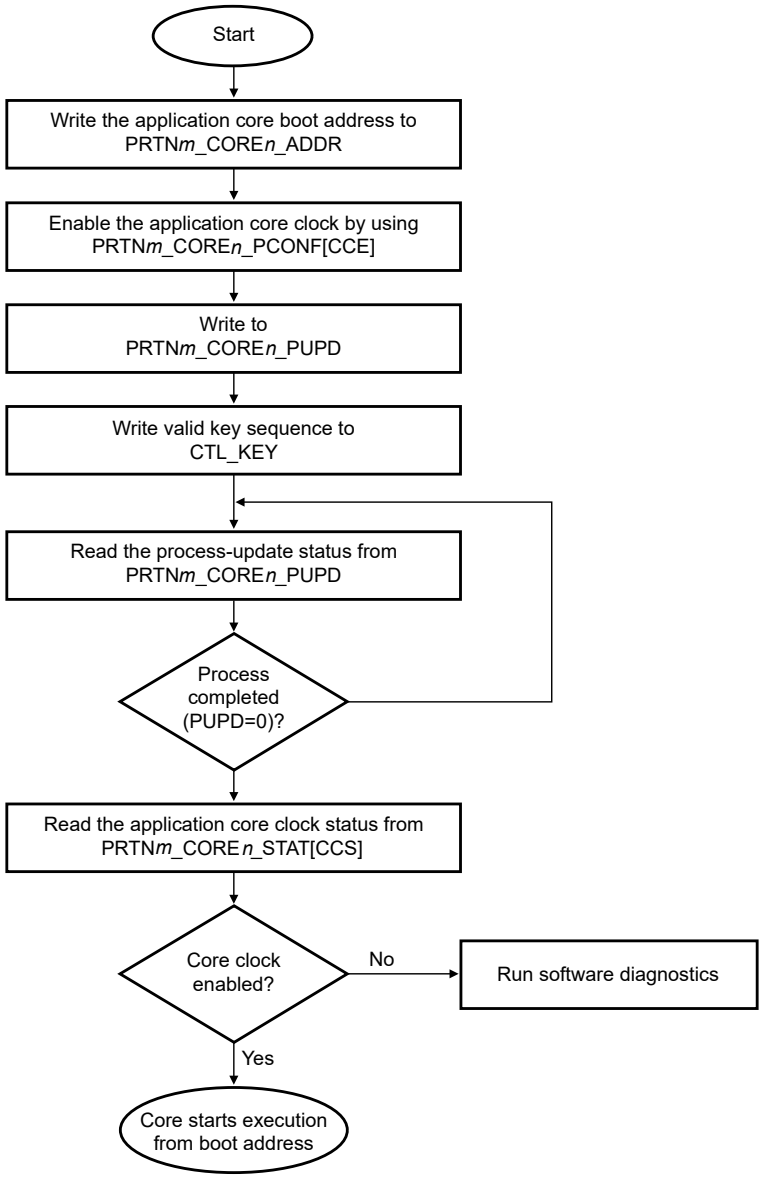



Figure 160. Application core initialization process

### 43.1.6 Application core shutdown process

If a debugger is attached to the chip and application debug is enabled, the application core continues running if you write 0 to MDM_AP.MDMAPCT[CM7_*n*_CORE_ACCESS].

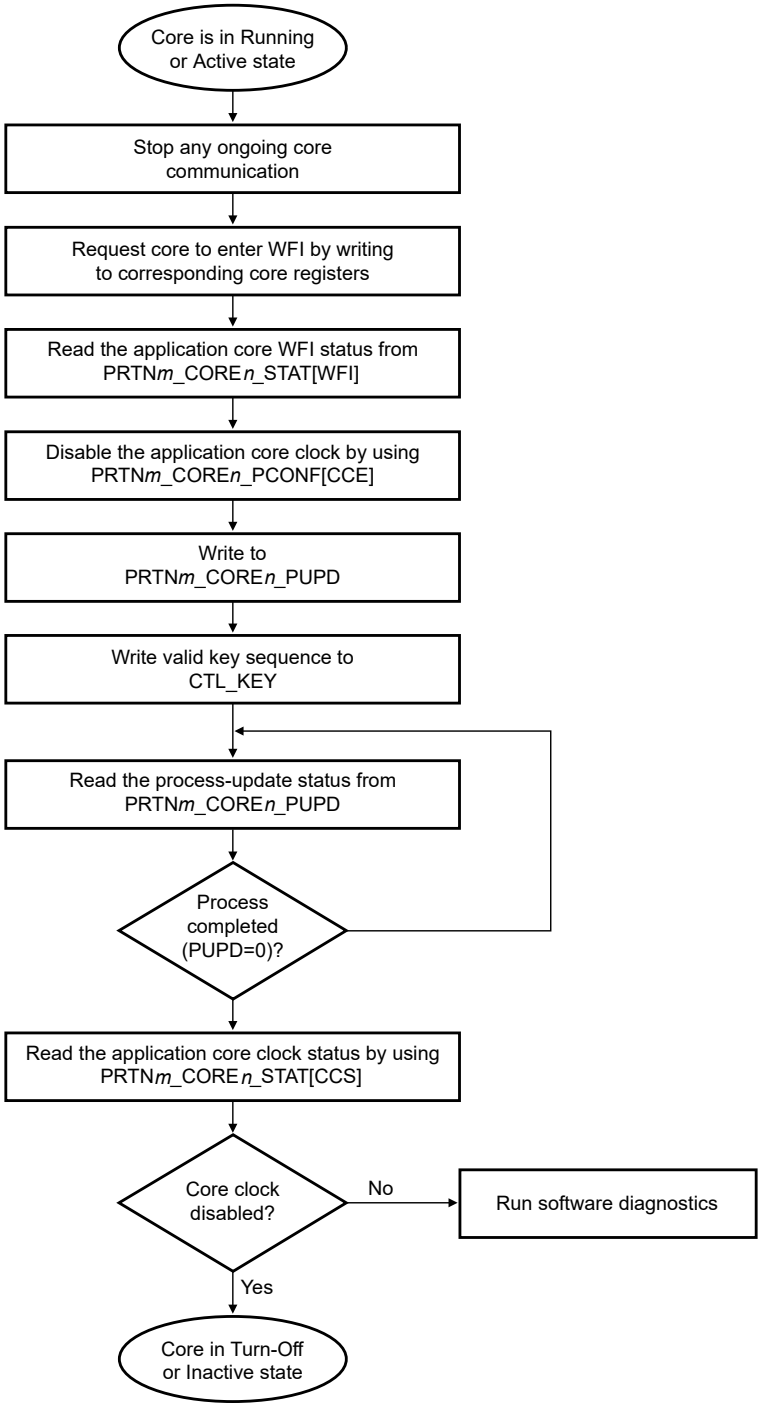



Figure 161. Application core shutdown process

## 43.1.7 Peripheral initialization process

You cannot control all the peripherals. For example, you cannot turn on and turn off the peripherals required for chip functionality across reset or power-up. They always remain on.

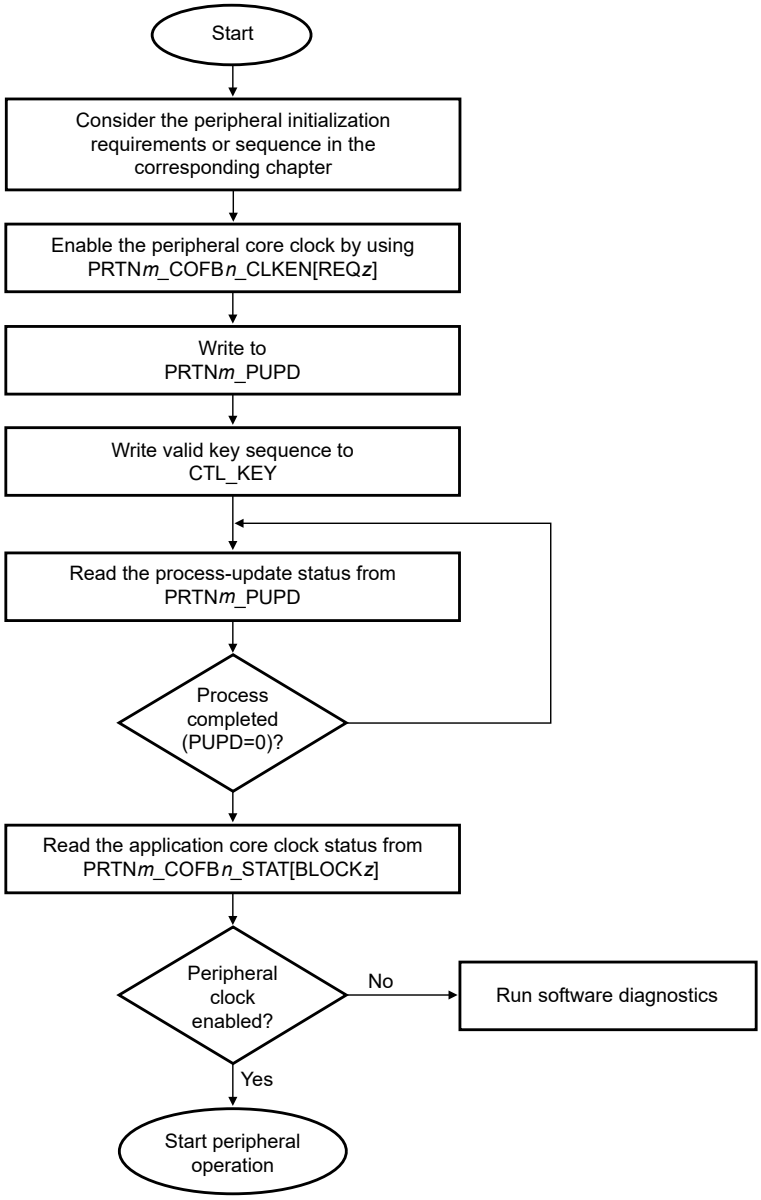



Figure 162. Peripheral initialization process

### 43.1.8 Peripheral shutdown process

You cannot control all the peripherals. For example, you cannot turn on and turn off the peripherals required for chip functionality across reset or power-up. They always remain on.

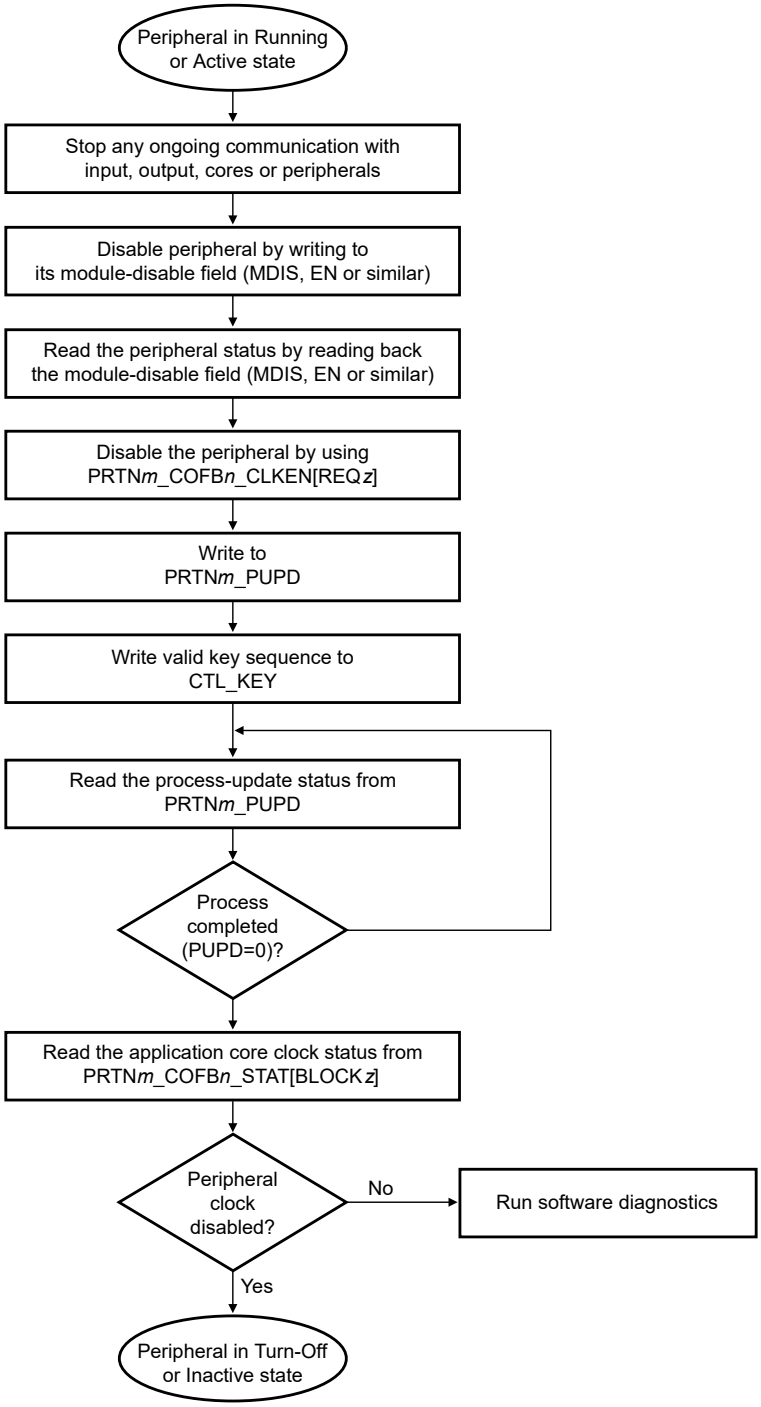



**Figure 163. Peripheral shutdown process**

## 43.2 Introduction

The MC_ME module generates control signals for a set of modules of the SoC. The set of signals are defined in corresponding 'Partition Configuration Registers'. It also implements a software-based mechanism for initiating a functional and destructive reset sequence and standby entry handshake with power management of SoC. See Figure 164 for the MC_ME block diagram.

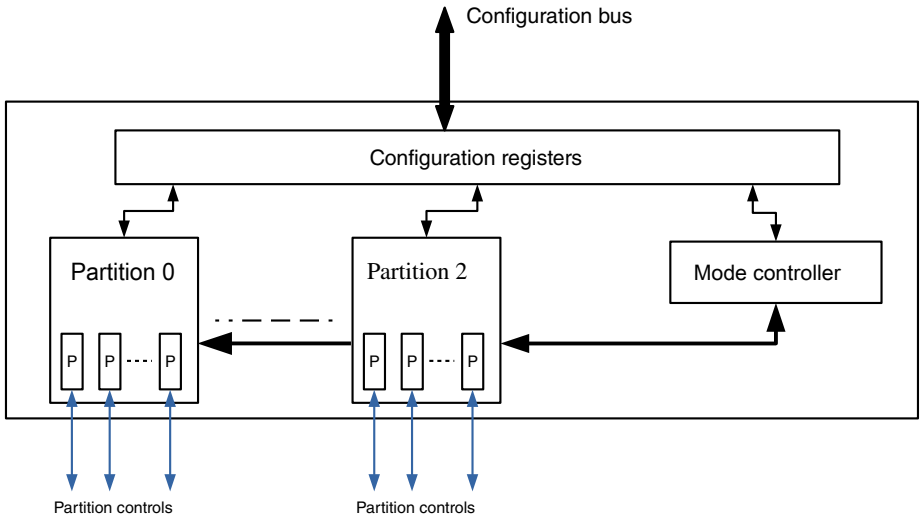



Figure 164. MC_ME block diagram

## 43.3 Features

MC_ME includes the following features:

- 3 logic partitions implementation and their controls
- Core clock controls
- Partition clock control
- Control mechanism for initiating a destructive or functional reset sequence to MC_RGM
- Control mechanism for initiating standby mode entry for SoC

The logic partition inside MC_ME refers to a certain group of on-chip resources (or IP blocks) that are clubbed together to represent a single 'Partition' inside MC_ME. The MC_ME partition can be the same or different than an LBIST partition. Each of the MC_ME partitions implements a certain number of hardware processes. These hardware processes provide a mechanism to regulate various control signals provided to or received from the IP blocks. The corresponding status signals can also be monitored from MC_ME register(s). Each of the hardware processes is bound to finish in 512 cycles of the MC_ME register configuration clocks. Therefore, the hardware processes are non blocking in nature. Mismatch in the expected versus actual status of any hardware process is controlled by a pre-defined software.

## 43.4 Partition processes

Each of the processes inside the partition controls register space and corresponds to a control signal provided to that partition. A partition can include a core, or COFBs, or both. The MC_ME hardware processes provide control and status via signals provided to partitions. Each partition can be assigned a signal for control and a signal for status. Each of the control signals implements functionality for the partition. For example, clock gating and peripheral control.

The hardware process can be triggered and monitored using a set of three registers:

- Configuration register; for example, Partition n Process Configuration register
- Update register; for example, Partition n Process Update register
- Status register; for example, Partition n Status register

Similar registers exist for cores inside the partition.

The process setup and triggering procedure is shown in Figure 165. Each of the processes is independent of others and can be triggered or re-triggered in parallel or sequential to other processes. The triggering or re-triggering mechanism remains the same.

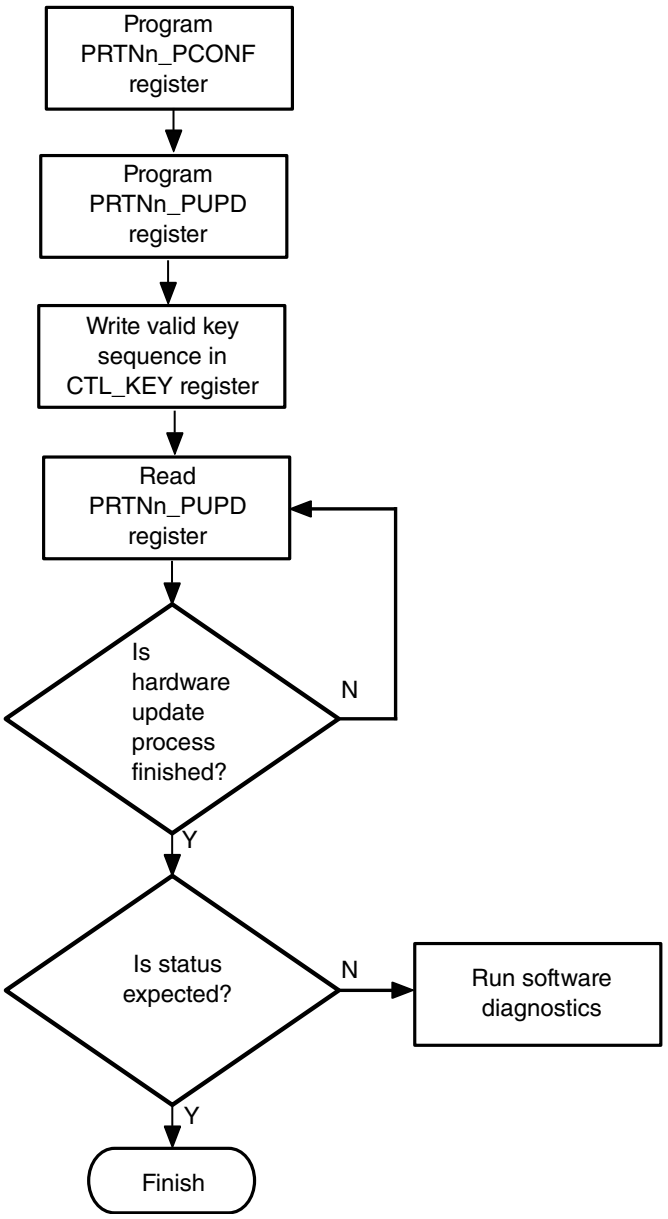



**Figure 165. Partition process setup procedure**

All the hardware processes are bound to finish in 512 cycles of the MC_ME configuration clock. If the actual and the expected status for a process does not match, then the diagnostics is left as a software responsibility. The software diagnostic can include further wait cycles for the status to match.

## 43.5 Mode transition

MC_ME implements a mode transition mechanism, whereby the mode of operation for SoC can be changed. Then module implements a mechanism that can lead to:

- Destructive reset
- Functional reset
- Standby mode entry

Destructive reset and functional reset requests from MC_ME are non-retractable transitions. After it is initiated, the other MC_ME functionality is rendered unusable and bus errors are provided for upcoming access to the MC_ME register until a reset sequence is executed by MC_RGM. Hence, it is vital that MC_RGM should never ignore or gate the reset requests from MC_ME.

For transition into the standby mode, the software should ensure that required IP blocks such as clock sources and I/O communication are in their respective inactive states before initiating a standby mode transition to MC_ME. After MC_ME initiates a power down sequence request, it cannot be retracted. The SoC enters a standby power down sequence and then reenter power-up sequence even for cases where the standby wakeup happens right at the time of initiating a power-down sequence request.

Steps for initiating the MC_ME mode transition:

1. Setup the MODE_CONF register with the corresponding target mode bit set to logic-1.
2. Perform the same update as done in the MODE_CONF register on the CONF_UPD register.
3. Write the valid control key (0x5AF0) on the CTL_KEY register.
4. Write the valid invert control key (0xA50F) on the CTL_KEY register.

Mode transition to MC_ME is initiated, after the sequence mentioned above is completed.

In step 1, if both FUNC_RST and DEST_RST in Mode Configuration Register (MODE_CONF) are 1:

- After step 4 is complete, MC_ME initiates a mode transition to a destructive (not functional) reset.
- After the chip exits reset, MC_RGM records that both MC_ME's destructive reset and MC_ME's functional reset were the reset source.

**NOTE**

Any hardware partition processes setup, along with mode transition, is executed in parallel to the mode transition of MC_ME.

## 43.6 Standby entry

MC_ME provides hardware processes that implement shutdown sequencing of on-chip resources, such as cores and COFBs. The standby entry sequencing can be achieved or implemented using these hardware processes. The order of the hardware process is determined by the software and MC_ME. It requires no restriction in sequencing of the operation. Following is an example sequence for initiating a power-down sequence for entering the standby mode for SoC. The standby entry sequence should include (but not limited to) the following steps:

1. Setting up wakeup lines
2. Shutting down cores and COFBs
3. Switching all MC_CGM muxes to FIRC with PCFS
4. Powering down all clock sources except FIRC
5. Setting up MC_ME using the main core and initiating a standby mode transition
6. Executing WFI instruction on the main core (per Arm specification)

### 43.6.1 Application core shutdown

This section describes a mechanism for shutting down an application core. The sequence proposed here is extendible with the housekeeping tasks required for other IPs. Each of the tasks mentioned in the following sequence, can be further integrated with an SoC-specific task.

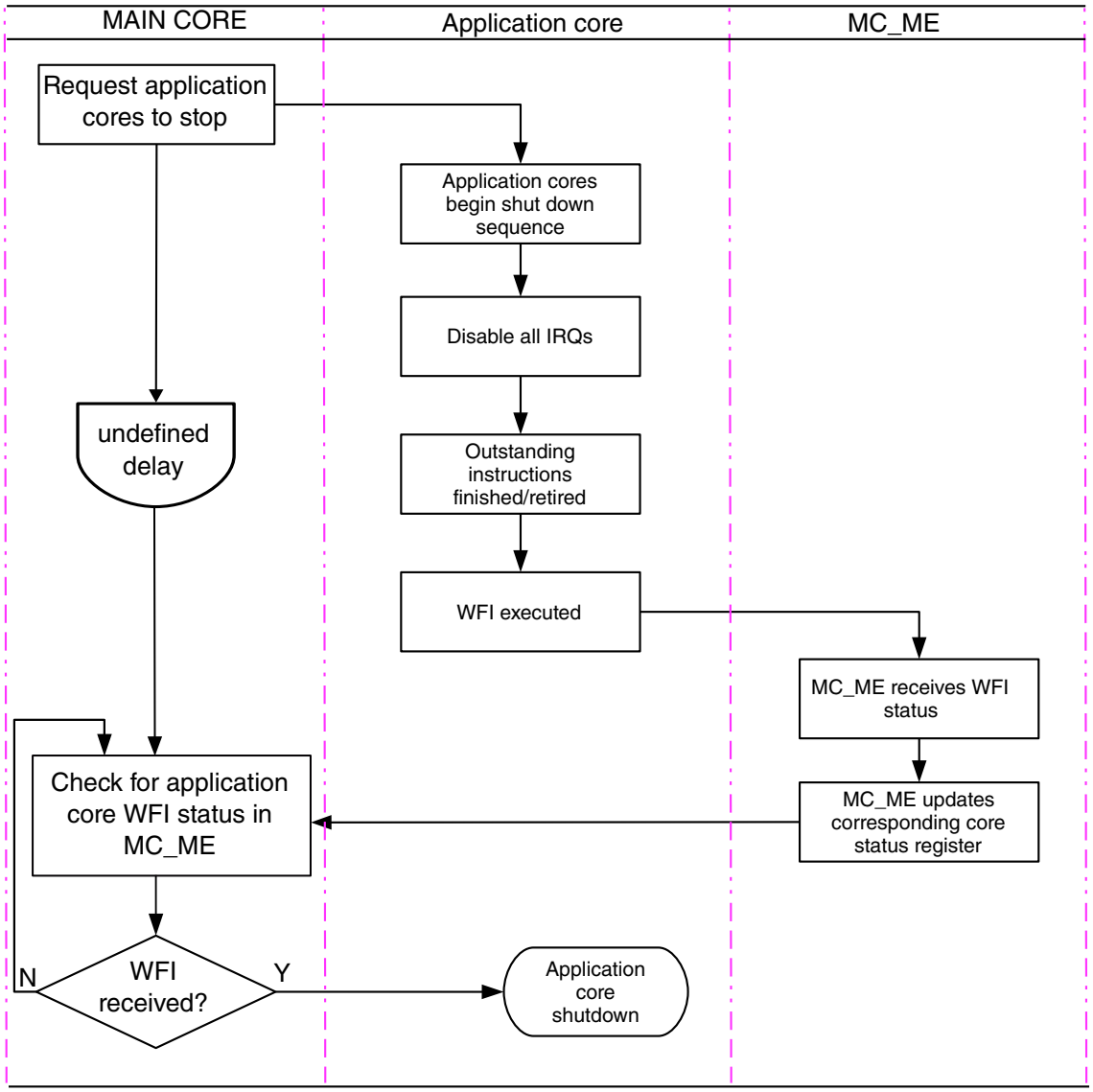



Figure 166. Application core shutdown

After the application core is shutdown, the main core can optionally decide to gate the respective core clock using the corresponding core clock hardware process.

### 43.6.2 Main core shutdown and standby entry

This section describes standby entry sequence along with the main core shutdown. This sequence should only be initiated after SoC is ready for entering standby and has completed all the housekeeping activities. It is necessary that the main core has completed all the operations pertaining to other (application cores) and is the last active core before initiating the standby entry sequence. See Figure 167.

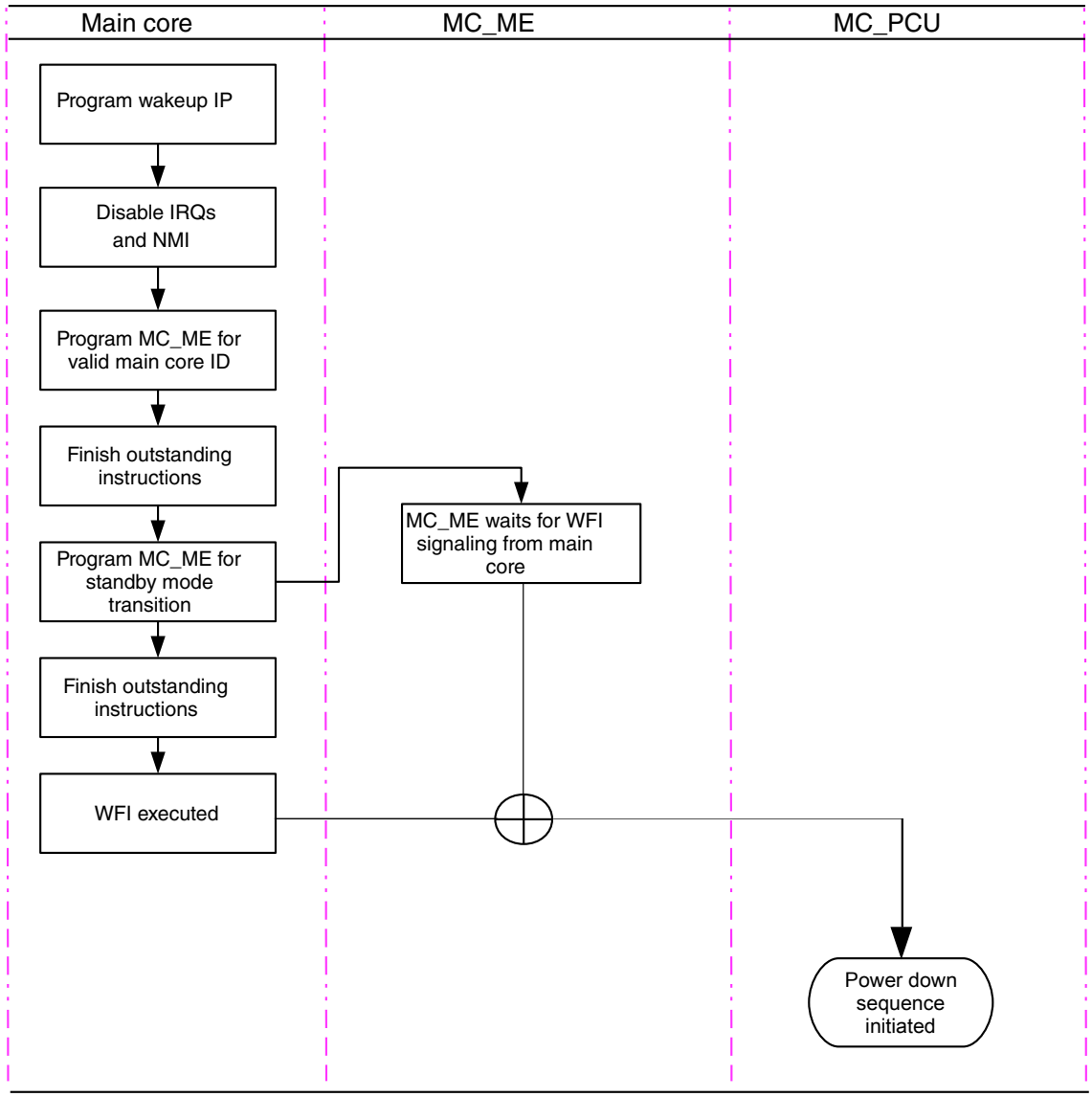



Figure 167. Standby entry sequence along with main core shutdown

**NOTE**

- MC_ME initiates the power sequence to MC_PCU. This enables the main core to remain inactive (WFI state) until it is reset and power-up again at standby exit.

## 43.7 MC_ME register descriptions

MC_ME implements set hardware processes that can be used by the software for changing the mode of operation for a partition. Following are the features of MC_ME registers:

- All registers are 32-bit wide.
- Only 32-bit read and write accesses are supported.
- Read/write accesses of less than 32 bits terminate with an error.
- Writes to read-only register fields in writable registers are ignored and do not provide an error message.

- Writes to read-only registers are aborted with an error message.

### 43.7.1 MC_ME memory map

MC_ME base address: 402D_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Control Key Register (CTL_KEY) | 32 | RW | 0000_5AF0h |
| 4h | Mode Configuration Register (MODE_CONF) | 32 | RW | 0000_0000h |
| 8h | Mode Update Register (MODE_UPD) | 32 | RW | 0000_0000h |
| Ch | Mode Status Register (MODE_STAT) | 32 | R | 0000_0000h |
| 10h | Main Core ID Register (MAIN_COREID) | 32 | RW | 0000_0000h |
| 100h | Partition 0 Process Configuration Register (PRTN0_PCONF) | 32 | RW | 0000_0001h |
| 104h | Partition 0 Process Update Register (PRTN0_PUPD) | 32 | RW | 0000_0000h |
| 108h | Partition 0 Status Register (PRTN0_STAT) | 32 | R | 0000_0001h |
| 114h | Partition 0 COFB Set 1 Clock Status Register (PRTN0_COFB1_STAT) | 32 | R | 0000_1000h |
| 134h | Partition 0 COFB Set 1 Clock Enable Register (PRTN0_COFB1_CLKEN) | 32 | RW | 0000_1000h |
| 140h | Partition 0 Core 0 Process Configuration Register (PRTN0_CORE0_PCONF) | 32 | RW | 0000_0000h |
| 144h | Partition 0 Core 0 Process Update Register (PRTN0_CORE0_PUPD) | 32 | RW | 0000_0000h |
| 148h | Partition 0 Core 0 Status Register (PRTN0_CORE0_STAT) | 32 | R | 0000_0000h |
| 14Ch | Partition 0 Core 0 Address Register (PRTN0_CORE0_ADDR) | 32 | RW | 0040_0000h |
| 188h | Partition 0 Core 2 Status Register (PRTN0_CORE2_STAT) | 32 | R | 0000_0001h |
| 18Ch | Partition 0 Core 2 Address Register (PRTN0_CORE2_ADDR) | 32 | R | 007F_FC00h |
| 300h | Partition 1 Process Configuration Register (PRTN1_PCONF) | 32 | RW | 0000_0001h |
| 304h | Partition 1 Process Update Register (PRTN1_PUPD) | 32 | RW | 0000_0000h |
| 308h | Partition 1 Status Register (PRTN1_STAT) | 32 | R | 0000_0001h |
| 310h | Partition 1 COFB Set 0 Clock Status Register (PRTN1_COFB0_STAT) | 32 | R | 5E3F_0007h |
| 314h | Partition 1 COFB Set 1 Clock Status Register (PRTN1_COFB1_STAT) | 32 | R | 1CFE_2FFCh |
| 318h | Partition 1 COFB Set 2 Clock Status Register (PRTN1_COFB2_STAT) | 32 | R | 3000_0000h |
| 31Ch | Partition 1 COFB Set 3 Clock Status Register (PRTN1_COFB3_STAT) | 32 | R | 0000_5FEEh |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 330h | Partition 1 COFB Set 0 Clock Enable Register (PRTN1_COFB0_CLKEN) | 32 | RW | 5E3F_0007h |
| 334h | Partition 1 COFB Set 1 Clock Enable Register (PRTN1_COFB1_CLKEN) | 32 | RW | 1CFE_2FFCh |
| 338h | Partition 1 COFB Set 2 Clock Enable Register (PRTN1_COFB2_CLKEN) | 32 | RW | 3000_0000h |
| 33Ch | Partition 1 COFB Set 3 Clock Enable Register (PRTN1_COFB3_CLKEN) | 32 | RW | 0000_5FEEh |
| 500h | Partition 2 Process Configuration Register (PRTN2_PCONF) | 32 | RW | 0000_0001h |
| 504h | Partition 2 Process Update Register (PRTN2_PUPD) | 32 | RW | 0000_0000h |
| 508h | Partition 2 Status Register (PRTN2_STAT) | 32 | R | 0000_0001h |
| 510h | Partition 2 COFB Set 0 Clock Status Register (PRTN2_COFB0_STAT) | 32 | R | 0200_0003h |
| 514h | Partition 2 COFB Set 1 Clock Status Register (PRTN2_COFB1_STAT) | 32 | R | CC00_0000h |
| 530h | Partition 2 COFB Set 0 Clock Enable Register (PRTN2_COFB0_CLKEN) | 32 | RW | 0200_0003h |
| 534h | Partition 2 COFB Set 1 Clock Enable Register (PRTN2_COFB1_CLKEN) | 32 | RW | CC00_0000h |

### 43.7.2 Control Key Register (CTL_KEY)

#### Offset

| Register | Offset |
|---|---|
| CTL_KEY | 0h |

#### Function

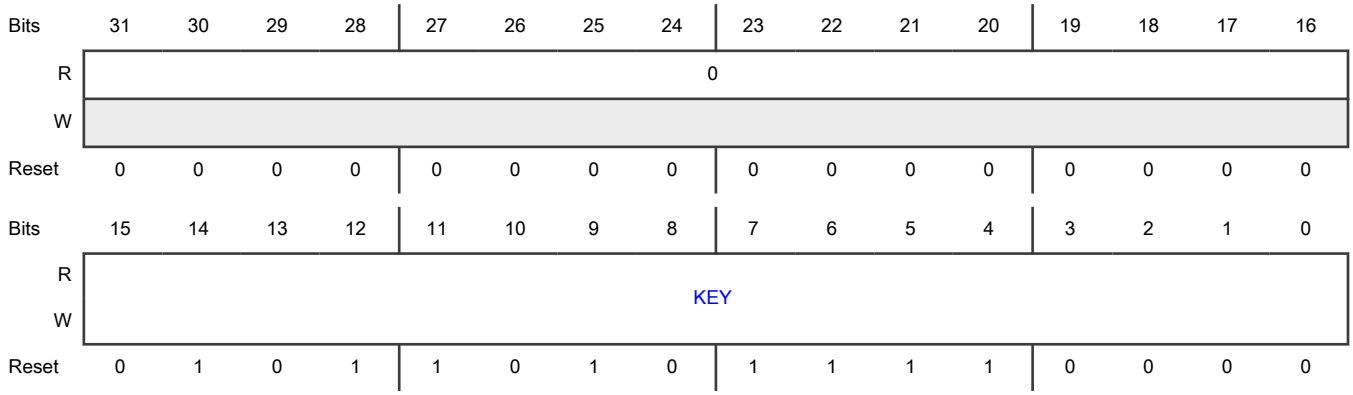
This register provides the mechanism to MC_ME for starting the hardware processes for the partition(s) and standby entry sequence. The hardware processes for partitions are triggered through the corresponding PRTNn_PCONF register. The mechanism to trigger the hardware processes of the respective partitions require two write operations: first time with key and second time with inverted key. The hexadecimal value of key is 0x5AF0 whereas for inverted key is 0xA50F.

For initiating a standby entry sequence, the MODE_CONF register is used for providing a standby entry request along with a valid key combination.

**NOTE**

Reads from this register return a valid key value to be written next.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 15-0<br>KEY | Control key<br>Key for starting the hardware processes. Writes with a value other than key or inverted key are ignored. Reads return bit inverted value corresponding to last write. |

## 43.7.3 Mode Configuration Register (MODE_CONF)

### Offset

| Register | Offset |
|---|---|
| MODE_CONF | 4h |

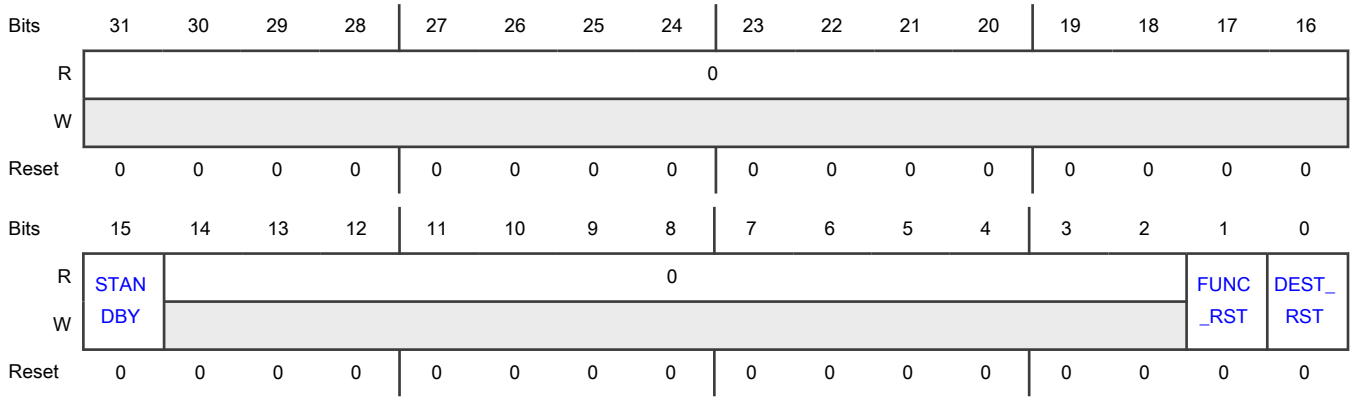
### Function

This register is used for initiating a standby request or a reset event (destructive or functional) for the chip. The functional or destructive events are signaled to MC_RGM for further handling.

**NOTE**

Software must not enable mode entry if the value of multiple fields is 1 in the MODE_CONF register.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 15<br>STANDBY | Standby request<br>Writing a logic-1 to this bit along with the MODE_UPD register configuration and followed with a valid key combination makes a standby entry sequence request to MC_ME. |
| 14-2<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 1<br>FUNC_RST | Functional reset request<br>Writing a logic-1 to this bit along with the MODE_UPD register configuration and followed with a valid key combination makes a functional reset event signaling to MC_RGM. |
| 0<br>DEST_RST | Destructive reset request<br>Writing a logic-1 to this bit along with the MODE_UPD register configuration and followed with a valid key combination makes a destructive reset event signaling to MC_RGM. |

## 43.7.4 Mode Update Register (MODE_UPD)

### Offset

| Register | Offset |
|---|---|
| MODE_UPD | 8h |

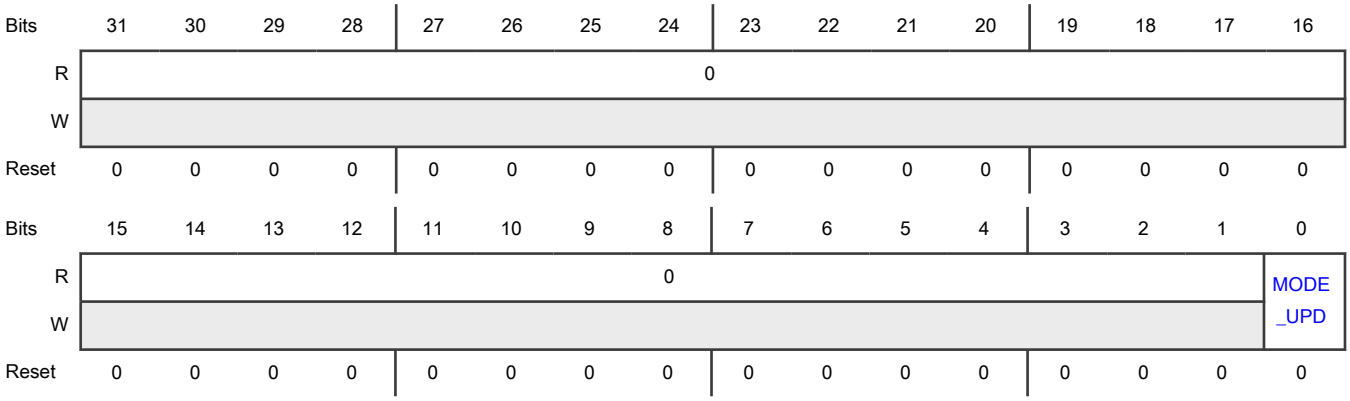
### Function

This register is used for initiating a mode change. Mode change refers to initiating a standby request, or generating a destructive or functional reset event to MC_RGM. Setting mode update field to logic-1, along with programming MODE_CONF registers and then followed by a valid key combination will generate a mode transition request.

**NOTE**

The MODE_UPD register is implemented to make mode transition programming model the same as partition programming model. This is for future expansion inside MC_ME.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 0<br>MODE_UPD | Mode update<br>Writing a logic-1 to this bit, followed by a valid key combination initiates a mode change as per the MODE_CONF register. |

## 43.7.5 Mode Status Register (MODE_STAT)

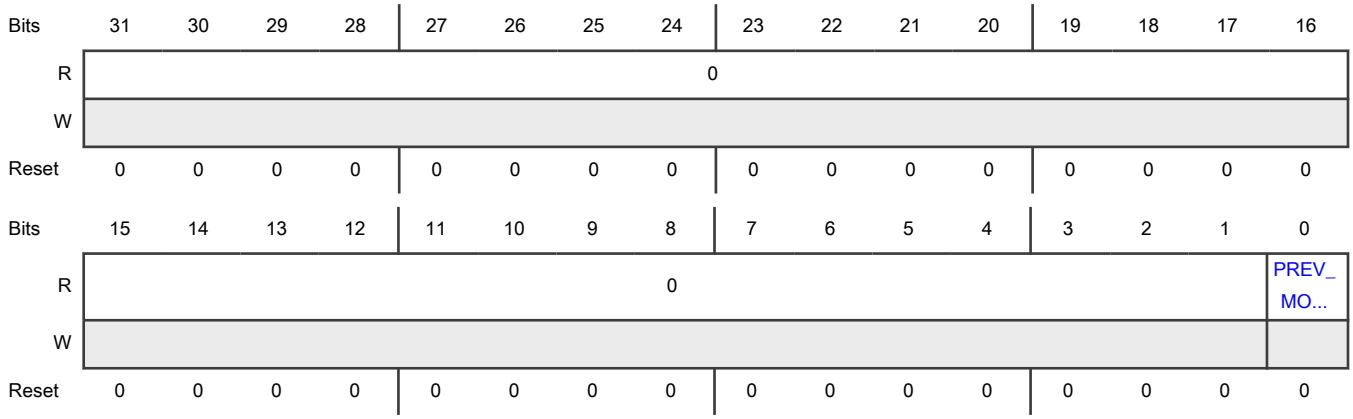
### Offset

| Register | Offset |
|---|---|
| MODE_STAT | Ch |

### Function

This register provides the status of the previous mode. In case of standby exit, if the reset event status register of MC_RGM are set, then contents of this register should be ignored.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 0<br>PREV_MODE | Previous mode<br>This bit shows the status of the previous mode.<br>0b - The previous mode was reset (any reset).<br>1b - The previous mode was standby. |

## 43.7.6 Main Core ID Register (MAIN_COREID)

### Offset

| Register | Offset |
|---|---|
| MAIN_COREID | 10h |

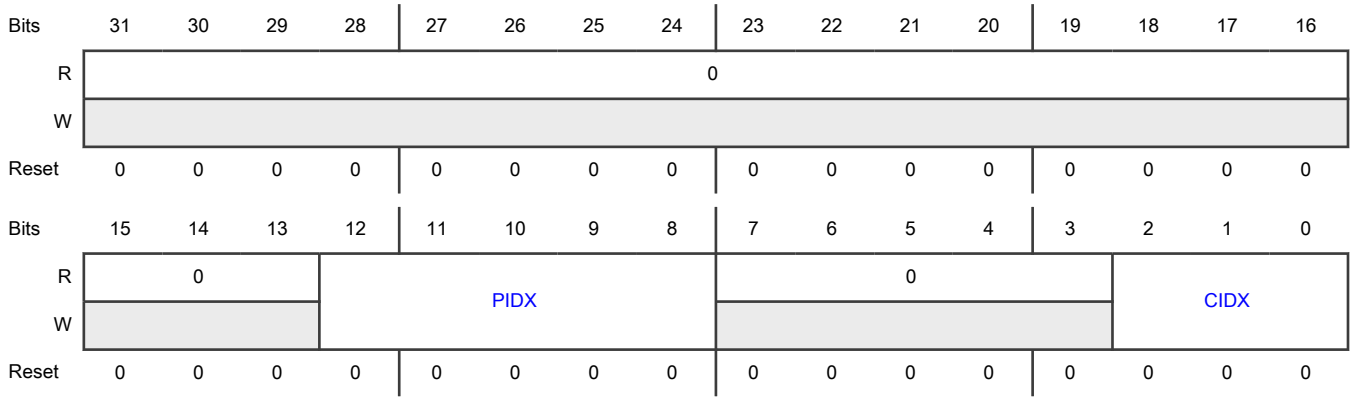
### Function

This register provides the ID of the main core sequencing the operation for the standby sequence. Core ID is required for entering in the standby mode, and using this MC_ME locates the WFI instruction execution of the main core. The core ID in this register is specified by the partition index along with the core index.

**NOTE**

Before initiating a standby entry sequence, the contents of this register should point to the correct main core. Providing non-existing or incorrect core ID leads to unpredictable hardware behavior.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-13<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 12-8<br>PIDX | Partition index<br>Provides the partition index of the main core. Only values 0 - 2 can be written. |
| 7-3<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 2-0<br>CIDX | Core index<br>Provides the core index of the main core inside the partition. |

## 43.7.7 Partition 0 Process Configuration Register (PRTN0_PCONF)

### Offset

| Register | Offset |
|---|---|
| PRTN0_PCONF | 100h |

### Function

This register provides a configuration for the hardware processes corresponding to partition 0. Each of the configuration bit corresponds to the 'nature' of the processes; for example, enabling/disabling and the trigger is controlled by the corresponding field in the PRTN0_PUPD register. When valid KEY combinations are written onto the CTL_KEY register, the PRTN0_PCONF and PRTN0_PUPD registers are used to determine the hardware processes to be executed. These are triggered in parallel and independent of each other. All dependent processes should be requested one after another from the software.

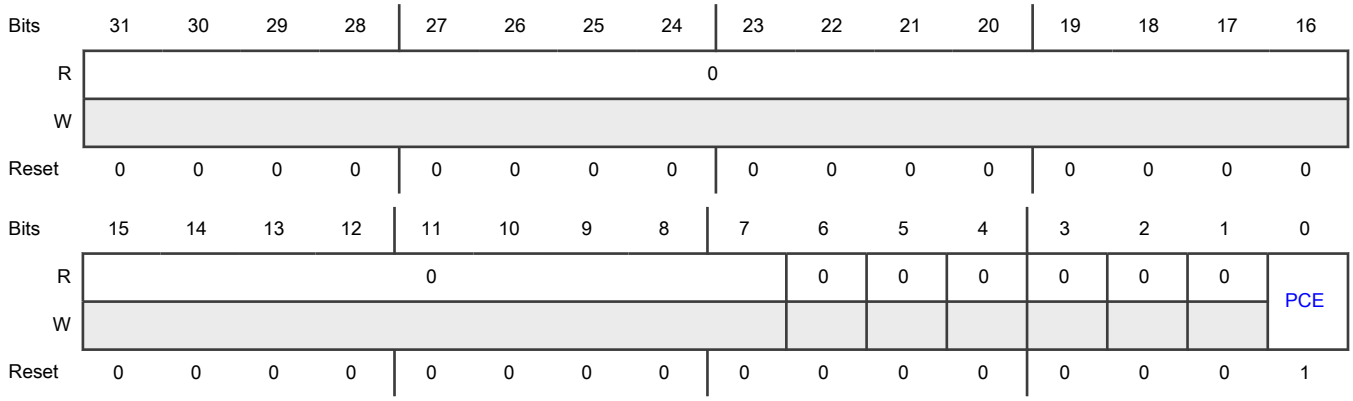
**NOTE**

The partition clock enable/disable are not standalone and must be done coherently in a fixed sequence. For details, see Software Reset Partition Turn-On Flow Chart and Software reset partition turn-off flowchart in Reset chapter.

**NOTE**

See chip-specific MC_ME information to check if this register is implemented on chip.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-7<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 6<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 5<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 4<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 3<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 2<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 1<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 0<br>PCE | Partition clock enable<br>This bit controls whether the clock to IPs (other than core(s)) in the partition should be enabled or disabled.<br>0b - Disable the clock to IPs<br>1b - Enable the clock to IPs |

## 43.7.8 Partition 0 Process Update Register (PRTN0_PUPD)

### Offset

| Register | Offset |
|---|---|
| PRTN0_PUPD | 104h |

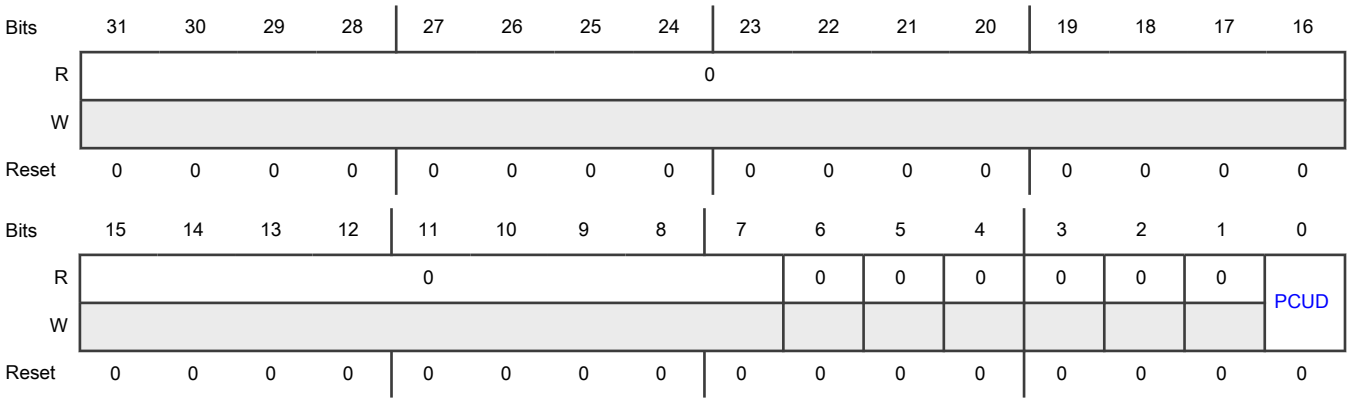
### Function

This register provides trigger signaling for the hardware processes corresponding to partition 0. Each of the control bit acts as a trigger for the corresponding hardware processes. When valid KEY combinations are written onto the CTL_KEY register, the hardware checks the bit fields that are programmed as logic-1 in this register, and then triggers the hardware process per the value in the corresponding bit field in the PRTN0_PCONF register. When the hardware process is finished the corresponding bit in this register is auto-cleared to logic-0.

**NOTE**

See chip-specific MC_ME information to check if this register is implemented on chip.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-7<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 6<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 5<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 4<br>— | Reserved<br>This field is reserved and read returns zeros. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 3<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 2<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 1<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 0<br>PCUD | Partition clock update<br>This bit controls whether the hardware processes for enabling/disabling the clock to IPs (other than core(s)) in the partition should be triggered or not.<br>0b - Do not trigger the hardware process<br>1b - Trigger the hardware process |

## 43.7.9 Partition 0 Status Register (PRTN0_STAT)

### Offset

| Register | Offset |
|---|---|
| PRTN0_STAT | 108h |

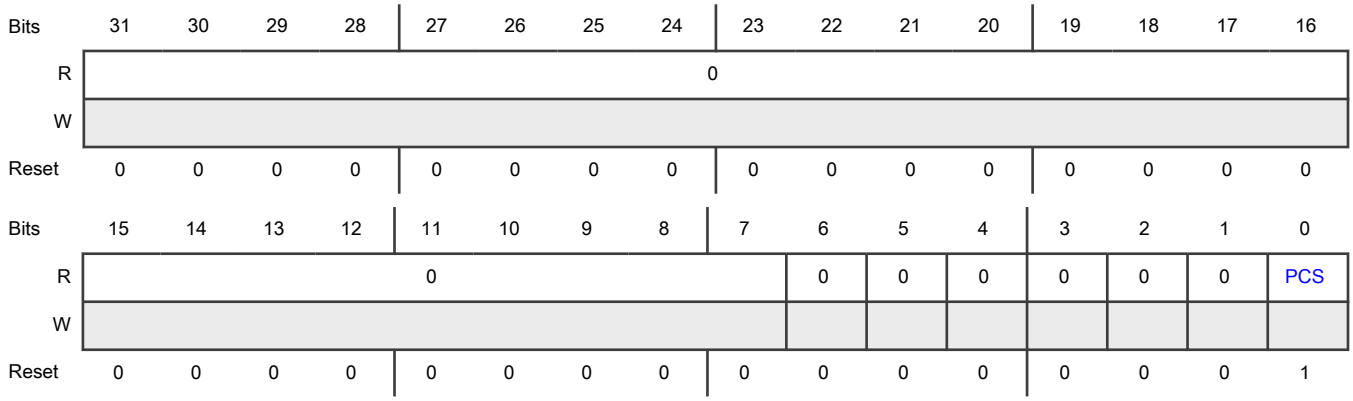
### Function

This register provides the current status of the control signals from the partition 0.

**NOTE**

See chip-specific MC_ME information to check if this register is implemented on chip.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-7<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 6<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 5<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 4<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 3<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 2<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 1<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 0<br>PCS | Partition clock status<br>This bit provides the status of the clock to partition.<br>0b - Clock is inactive<br>1b - Clock is active |

## 43.7.10 Partition 0 COFB Set 1 Clock Status Register (PRTN0_COFB1_STAT)

### Offset

| Register | Offset |
|---|---|
| PRTN0_COFB1_STAT | 114h |

### Function

This register provides the status of set 1 of COFBs inside partition 0.

**NOTE**

The reset value of this register can vary depending on the availability of active clock pulses inside partition 0.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BLOCK47 | BLOCK46 | BLOCK45 | BLOCK44 | 0 | BLOCK42 | BLOCK41 | BLOCK40 | BLOCK39 | BLOCK38 | 0 | BLOCK36 | BLOCK35 | BLOCK34 | BLOCK33 | BLOCK32 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 30<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 29<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 28<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 27<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 26<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 25<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 24<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 23<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 22<br>— | Reserved<br>This field is reserved and read returns zeros. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 21<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 20<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 19<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 18<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 17<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 16<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 15<br>BLOCK47 | IP block status<br>This bit provides the clock status of block 47 in partition 0.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 14<br>BLOCK46 | IP block status<br>This bit provides the clock status of block 46 in partition 0.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 13<br>BLOCK45 | IP block status<br>This bit provides the clock status of PIT_1 in partition 0.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 12<br>BLOCK44 | IP block status<br>This bit provides the clock status of PIT_0 in partition 0.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 11<br>— | Reserved<br>This field is reserved and read returns zeros. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 10<br>BLOCK42 | IP block status<br>This bit provides the clock status of ADC_2 in partition 0.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 9<br>BLOCK41 | IP block status<br>This bit provides the clock status of ADC_1 in partition 0.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 8<br>BLOCK40 | IP block status<br>This bit provides the clock status of ADC_0 in partition 0.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 7<br>BLOCK39 | IP block status<br>This bit provides the clock status of LCU_1 in partition 0.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 6<br>BLOCK38 | IP block status<br>This bit provides the clock status of LCU_0 in partition 0.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 5<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 4<br>BLOCK36 | IP block status<br>This bit provides the clock status of eMIOS_2 in partition 0.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 3<br>BLOCK35 | IP block status<br>This bit provides the clock status of eMIOS_1 in partition 0.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 2 | IP block status |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| BLOCK34 | This bit provides the clock status of eMIOS_0 in partition 0.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 1<br>BLOCK33 | IP block status<br>This bit provides the clock status of BCTU in partition 0.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 0<br>BLOCK32 | IP block status<br>This bit provides the clock status of TRGMUX in partition 0.<br>0b - Clock is not running.<br>1b - Clock is running. |

### 43.7.11 Partition 0 COFB Set 1 Clock Enable Register (PRTN0_COFB1_CLKEN)

#### Offset

| Register | Offset |
|---|---|
| PRTN0_COFB1_CLKEN | 134h |

#### Function

This register provides clock control signaling to the individual COFBs in set 1 inside partition 0. Whenever a partition clock enable (non-core) hardware process is initiated, the value of logic-1 in the corresponding bit locations of this register enables the clock to the corresponding block in the partition.

**NOTE**

The reset value of this register is not defined and is as per the availability of the clock source. See Chip-specific MC_ME information for clock source availability.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REQ47 | REQ46 | REQ45 | REQ44 | 0 | REQ42 | REQ41 | REQ40 | REQ39 | REQ38 | 0 | REQ36 | REQ35 | REQ34 | REQ33 | REQ32 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 30<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 29<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 28<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 27<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 26<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 25<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 24<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 23<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 22<br>— | Reserved<br>This field is reserved and read returns zeros. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 21<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 20<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 19<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 18<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 17<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 16<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 15<br>REQ47 | Clock enable<br>This bit provides the clock enable control for block 47 in partition 0.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 14<br>REQ46 | Clock enable<br>This bit provides the clock enable control for block 46 in partition 0.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 13<br>REQ45 | Clock enable<br>This bit provides the clock enable control for PIT_1 in partition 0.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 12<br>REQ44 | Clock enable<br>This bit provides the clock enable control for PIT_0 in partition 0.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 11<br>— | Reserved<br>This field is reserved and read returns zeros. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 10<br>REQ42 | Clock enable<br>This bit provides the clock enable control for ADC_2 in partition 0.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 9<br>REQ41 | Clock enable<br>This bit provides the clock enable control for ADC_1 in partition 0.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 8<br>REQ40 | Clock enable<br>This bit provides the clock enable control for ADC_0 in partition 0.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 7<br>REQ39 | Clock enable<br>This bit provides the clock enable control for LCU_1 in partition 0.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 6<br>REQ38 | Clock enable<br>This bit provides the clock enable control for LCU_0 in partition 0.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 5<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 4<br>REQ36 | Clock enable<br>This bit provides the clock enable control for eMIOS_2 in partition 0.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 3<br>REQ35 | Clock enable<br>This bit provides the clock enable control for eMIOS_1 in partition 0.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 2 | Clock enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| REQ34 | This bit provides the clock enable control for eMIOS_0 in partition 0.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 1<br>REQ33 | Clock enable<br>This bit provides the clock enable control for BCTU in partition 0.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 0<br>REQ32 | Clock enable<br>This bit provides the clock enable control for TRGMUX in partition 0.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |

## 43.7.12 Partition 0 Core 0 Process Configuration Register (PRTN0_CORE0_PCONF)

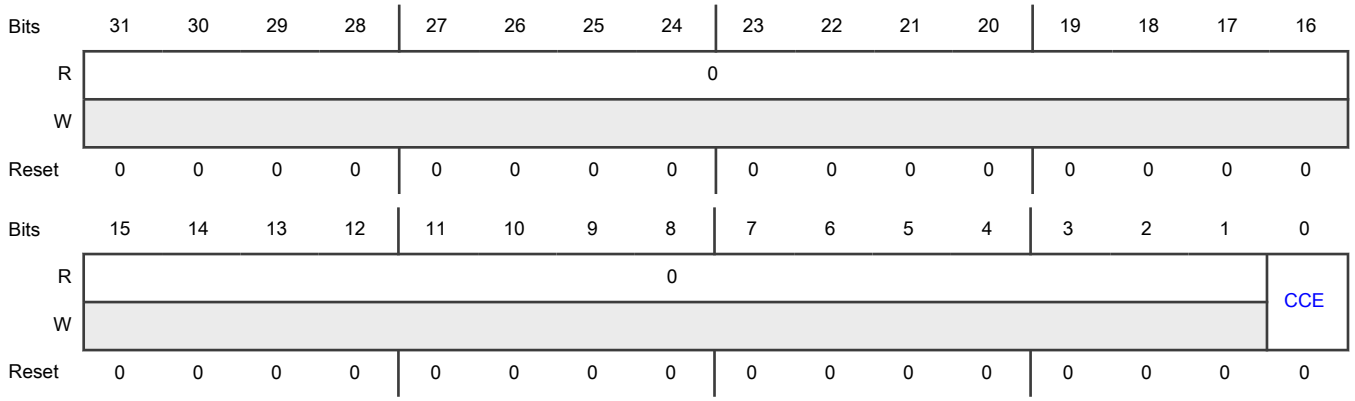
### Offset

| Register | Offset |
|---|---|
| PRTN0_CORE0_PCONF | 140h |

### Function

This register provides configurations for the Core 0 hardware processes corresponding to partition 0. Each of the configuration bit corresponds to the 'nature' of the processes; for example, enabling/disabling and the trigger is controlled by the corresponding field in the PRTN0_CORE0_PUPD register. When valid KEY combinations are written onto the CTL_KEY register, the PRTN0_CORE0_PUPD and PRTN0_CORE0_PCONF registers are used to determine the hardware processes to be executed. These processes are triggered in parallel and are independent of each other. All dependent processes should be requested one after another from the software.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 0<br>CCE | Core 0 clock enable<br>This bit controls whether the clock to Core 0 in partition 0 should be enabled or disabled.<br>0b - Disable the core clock<br>1b - Enable the core clock |

## 43.7.13 Partition 0 Core 0 Process Update Register (PRTN0_CORE0_PUPD)

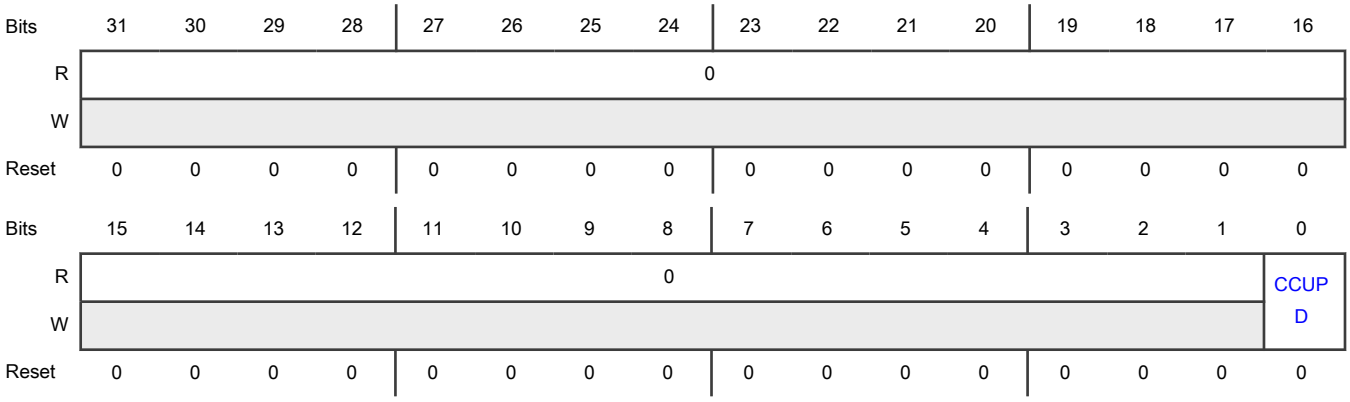
### Offset

| Register | Offset |
|---|---|
| PRTN0_CORE0_PUPD | 144h |

### Function

This register provides trigger signaling for the core hardware processes corresponding to partition 0. Each of the control bit acts as a trigger for the corresponding hardware processes. When valid KEY combinations are written onto the CTL_KEY register, the hardware checks the bit fields that are programmed as logic-1 in this register, and then triggers the hardware process per the value in the corresponding bit field in the PRTN0_CORE0_PCONF register. When the hardware process is finished, the corresponding bit in this register is auto-cleared to logic-0.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | This field is reserved and read returns zeros. |
| 0<br>CCUPD | Core 0 clock update<br>This bit controls whether the hardware processes for enabling/disabling the clock to Core 0 in the partition 0 should be triggered or not.<br>0b - Do not trigger the hardware process<br>1b - Trigger the hardware process |

## 43.7.14 Partition 0 Core 0 Status Register (PRTN0_CORE0_STAT)

### Offset

| Register | Offset |
|---|---|
| PRTN0_CORE0_STAT | 148h |

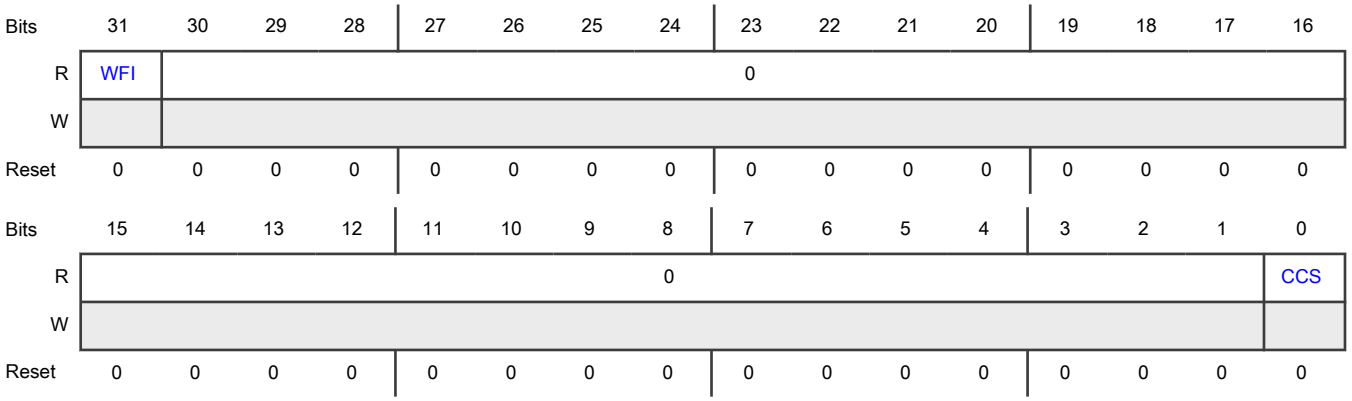
### Function

This register provides the status corresponding to Core 0 in partition 0. The status signal corresponds to clock states and the WFI signal included from Core 0.

**NOTE**

The value held in WFI field of this STATUS register is "current" value of the WFISTANDBY signal from the core. Hence out-of-reset, the reset value of this field will depend on the status of the core (core is running or in low power mode). So, simple reset read sweep will always return current value (different than other register reads such as on control registers).

### Diagram



### Fields

| Field | Function |
|---|---|
| 31<br>WFI | Wait for interrupt status<br>This bit provides the WFI status approaching from Core 0 in partition 0.<br>0b - No WFI executed<br>1b - WFI executed |
| 30-1<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 0<br>CCS | Core 0 clock process status<br>This bit provides the status of the clock corresponding to core clock enablement/disablement.<br>0b - Clock is inactive.<br>1b - Clock is active. |

## 43.7.15 Partition 0 Core 0 Address Register (PRTN0_CORE0_ADDR)

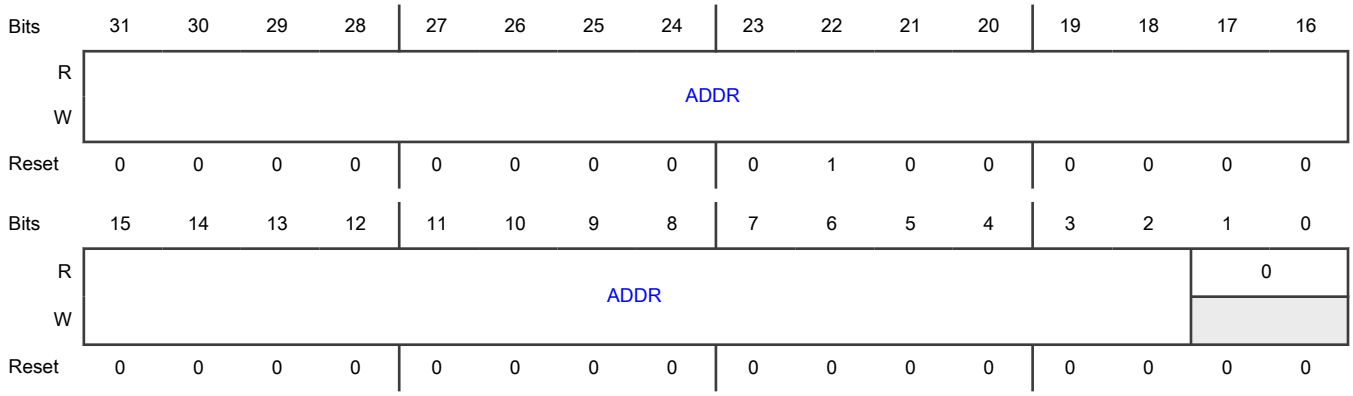
### Offset

| Register | Offset |
|---|---|
| PRTN0_CORE0_ADDR | 14Ch |

### Function

This register contains the boot address for Core 0 in partition 0.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-2<br>ADDR | Address<br>Core 0 boot address |
| 1-0<br>— | Reserved<br>This field is reserved and read returns zeros. |

## 43.7.16 Partition 0 Core 2 Status Register (PRTN0_CORE2_STAT)

### Offset

| Register | Offset |
|---|---|
| PRTN0_CORE2_STAT | 188h |

### Function

This register provides the status corresponding to Core 2 in partition 0. The status signal corresponds to clock states and the WFI signal included from Core 2.

**NOTE**

The value held in WFI field of this STATUS register is "current" value of the WFISTANDBY signal from the core. Hence out-of-reset, the reset value of this field will depend on the status of the core (core is running or in low power mode). So, simple reset read sweep will always return current value (different than other register reads such as on control registers).

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31 | Wait for interrupt status |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| WFI | This bit provides the WFI status approaching from Core 2 in partition 0.<br>0b - No WFI executed<br>1b - WFI executed |
| 30-1<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 0<br>CCS | Core 2 clock process status<br>This bit provides the status of the clock corresponding to core clock enablement/disablement.<br>1b - Clock is active. |

## 43.7.17 Partition 0 Core 2 Address Register (PRTN0_CORE2_ADDR)

### Offset

| Register | Offset |
|---|---|
| PRTN0_CORE2_ADDR | 18Ch |

### Function

This register contains the boot address for Core 2 in partition 0.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ADDR | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

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

### Fields

| Field | Function |
|---|---|
| 31-2<br>ADDR | Address<br>Core 2 boot address |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 1-0<br>— | Reserved<br>This field is reserved and read returns zeros. |

## 43.7.18 Partition 1 Process Configuration Register (PRTN1_PCONF)

### Offset

| Register | Offset |
|---|---|
| PRTN1_PCONF | 300h |

### Function

This register provides a configuration for the hardware processes corresponding to partition 1. Each of the configuration bit corresponds to the 'nature' of the processes; for example, enabling/disabling and the trigger is controlled by the corresponding field in the PRTN1_PUPD register. When valid KEY combinations are written onto the CTL_KEY register, the PRTN1_PCONF and PRTN1_PUPD registers are used to determine the hardware processes to be executed. These are triggered in parallel and independent of each other. All dependent processes should be requested one after another from the software.

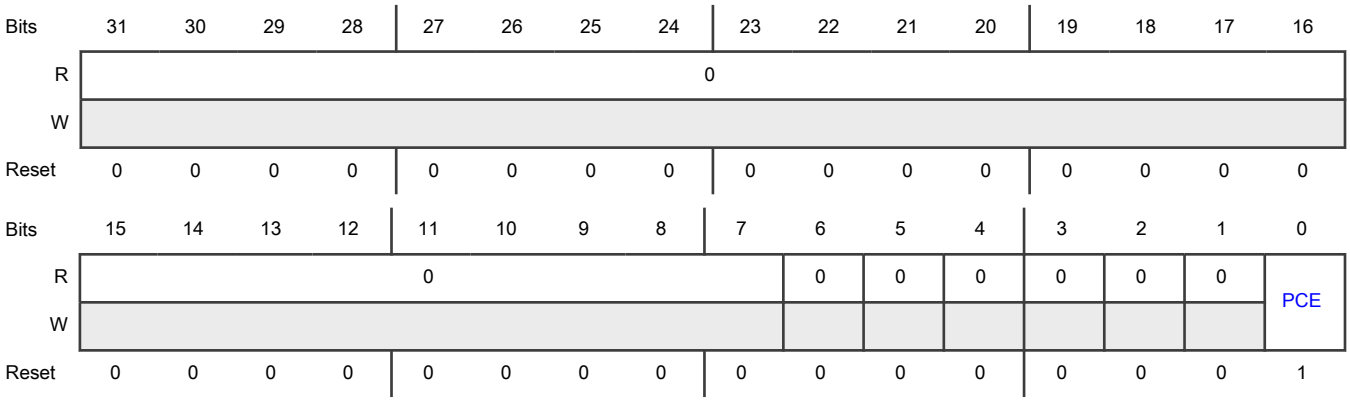
**NOTE**

The partition clock enable/disable are not standalone and must be done coherently in a fixed sequence. For details, see Software Reset Partition Turn-On Flow Chart and Software reset partition turn-off flowchart in Reset chapter.

**NOTE**

See chip-specific MC_ME information to check if this register is implemented on chip.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-7 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | This field is reserved and read returns zeros. |
| 6<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 5<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 4<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 3<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 2<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 1<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 0<br>PCE | Partition clock enable<br>This bit controls whether the clock to IPs (other than core(s)) in the partition should be enabled or disabled.<br>0b - Disable the clock to IPs<br>1b - Enable the clock to IPs |

### 43.7.19 Partition 1 Process Update Register (PRTN1_PUPD)

#### Offset

| Register | Offset |
|---|---|
| PRTN1_PUPD | 304h |

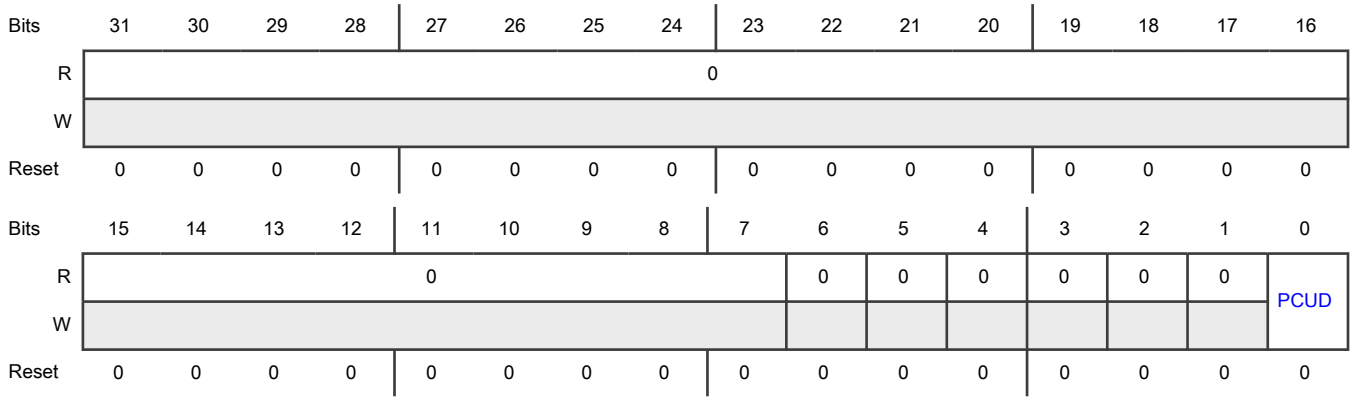
#### Function

This register provides trigger signaling for the hardware processes corresponding to partition 1. Each of the control bit acts as a trigger for the corresponding hardware processes. When valid KEY combinations are written onto the CTL_KEY register, the hardware checks the bit fields that are programmed as logic-1 in this register, and then triggers the hardware process per the value in the corresponding bit field in the PRTN1_PCONF register. When the hardware process is finished the corresponding bit in this register is auto-cleared to logic-0.

**NOTE**

See chip-specific MC_ME information to check if this register is implemented on chip.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-7<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 6<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 5<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 4<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 3<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 2<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 1<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 0<br>PCUD | Partition clock update<br>This bit controls whether the hardware processes for enabling/disabling the clock to IPs (other than core(s)) in the partition should be triggered or not.<br>0b - Do not trigger the hardware process<br>1b - Trigger the hardware process |

## 43.7.20 Partition 1 Status Register (PRTN1_STAT)

### Offset

| Register | Offset |
|---|---|
| PRTN1_STAT | 308h |

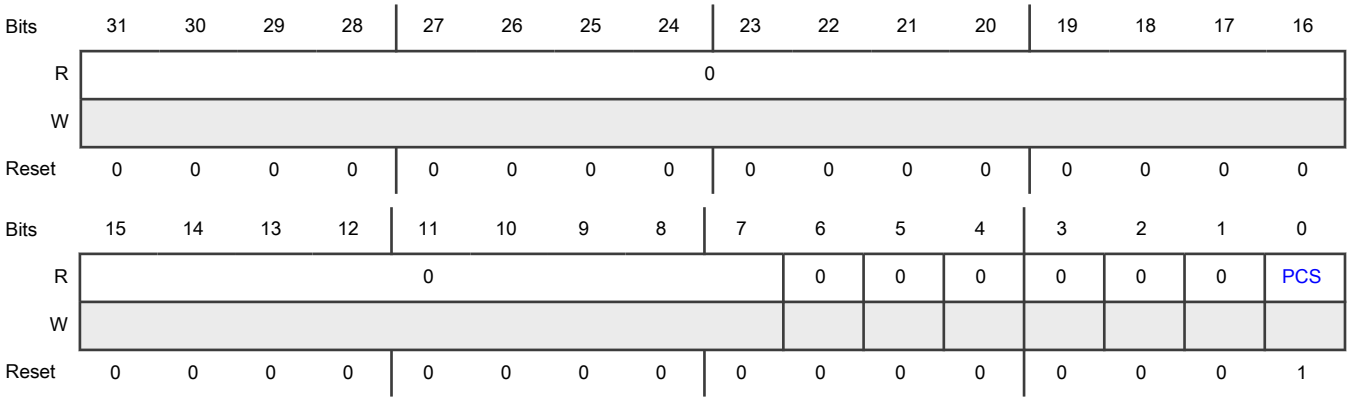
### Function

This register provides the current status of the control signals from the partition 1.

**NOTE**

See chip-specific MC_ME information to check if this register is implemented on chip.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-7<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 6<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 5<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 4<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 3<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 2 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | This field is reserved and read returns zeros. |
| 1<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 0<br>PCS | Partition clock status<br>This bit provides the status of the clock to partition.<br>0b - Clock is inactive<br>1b - Clock is active |

## 43.7.21 Partition 1 COFB Set 0 Clock Status Register (PRTN1_COFB0_STAT)

### Offset

| Register | Offset |
|---|---|
| PRTN1_COFB0_STAT | 310h |

### Function

This register provides the status of set 0 of COFBs inside partition 1.

**NOTE**

The reset value of this register can vary depending on the availability of active clock pulses inside partition 1.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BLOCK31 | BLOCK30 | BLOCK29 | BLOCK28 | BLOCK27 | BLOCK26 | BLOCK25 | BLOCK24 | BLOCK23 | BLOCK22 | BLOCK21 | BLOCK20 | BLOCK19 | BLOCK18 | BLOCK17 | BLOCK16 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BLOCK15 | BLOCK14 | BLOCK13 | BLOCK12 | BLOCK11 | BLOCK10 | BLOCK9 | BLOCK8 | BLOCK7 | BLOCK6 | BLOCK5 | BLOCK4 | BLOCK3 | BLOCK2 | BLOCK1 | BLOCK0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

### Fields

| Field | Function |
|---|---|
| 31<br>BLOCK31 | IP block status<br>This bit provides the clock status of INTM in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 30<br>BLOCK30 | IP block status<br>This bit provides the clock status of XRDC in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 29<br>BLOCK29 | IP block status<br>This bit provides the clock status of STM_0 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 28<br>BLOCK28 | IP block status<br>This bit provides the clock status of SWT_0 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 27<br>BLOCK27 | IP block status<br>This bit provides the clock status of PFC_alt in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 26<br>BLOCK26 | IP block status<br>This bit provides the clock status of PFC in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 25<br>BLOCK25 | IP block status<br>This bit provides the clock status of PRAM_0 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 24<br>BLOCK24 | IP block status<br>This bit provides the clock status of MSCM in partition 1.<br>0b - Clock is not running. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Clock is running. |
| 23<br>BLOCK23 | IP block status<br>This bit provides the clock status of ERM in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 22<br>BLOCK22 | IP block status<br>This bit provides the clock status of EIM in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 21<br>BLOCK21 | IP block status<br>This bit provides the clock status of SDA-AP in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 20<br>BLOCK20 | IP block status<br>This bit provides the clock status of Debug_APB in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 19<br>BLOCK19 | IP block status<br>This bit provides the clock status of Debug_APB in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 18<br>BLOCK18 | IP block status<br>This bit provides the clock status of Debug_APB in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 17<br>BLOCK17 | IP block status<br>This bit provides the clock status of Debug_APB in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 16<br>BLOCK16 | IP block status<br>This bit provides the clock status of Debug_APB in partition 1. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Clock is not running.<br>1b - Clock is running. |
| 15<br>BLOCK15 | IP block status<br>This bit provides the clock status of eDMA in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 14<br>BLOCK14 | IP block status<br>This bit provides the clock status of eDMA in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 13<br>BLOCK13 | IP block status<br>This bit provides the clock status of eDMA in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 12<br>BLOCK12 | IP block status<br>This bit provides the clock status of eDMA in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 11<br>BLOCK11 | IP block status<br>This bit provides the clock status of eDMA in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 10<br>BLOCK10 | IP block status<br>This bit provides the clock status of eDMA in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 9<br>BLOCK9 | IP block status<br>This bit provides the clock status of eDMA in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 8 | IP block status |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| BLOCK8 | This bit provides the clock status of eDMA in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 7<br>BLOCK7 | IP block status<br>This bit provides the clock status of eDMA in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 6<br>BLOCK6 | IP block status<br>This bit provides the clock status of eDMA in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 5<br>BLOCK5 | IP block status<br>This bit provides the clock status of eDMA in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 4<br>BLOCK4 | IP block status<br>This bit provides the clock status of eDMA in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 3<br>BLOCK3 | IP block status<br>This bit provides the clock status of eDMA in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 2<br>BLOCK2 | IP block status<br>This bit provides the clock status of XBIC in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 1<br>BLOCK1 | IP block status<br>This bit provides the clock status of XBIC in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 0<br>BLOCK0 | IP block status<br>This bit provides the clock status of AXBS in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |

## 43.7.22 Partition 1 COFB Set 1 Clock Status Register (PRTN1_COFB1_STAT)

### Offset

| Register | Offset |
|---|---|
| PRTN1_COFB1_STAT | 314h |

### Function

This register provides the status of set 1 of COFBs inside partition 1.

**NOTE**

The reset value of this register can vary depending on the availability of active clock pulses inside partition 1.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BLOCK63 | 0 | 0 | BLOCK60 | BLOCK59 | BLOCK58 | 0 | BLOCK56 | BLOCK55 | BLOCK54 | BLOCK53 | BLOCK52 | BLOCK51 | BLOCK50 | BLOCK49 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BLOCK47 | 0 | BLOCK45 | 0 | BLOCK43 | BLOCK42 | BLOCK41 | BLOCK40 | BLOCK39 | BLOCK38 | BLOCK37 | BLOCK36 | BLOCK35 | BLOCK34 | BLOCK33 | BLOCK32 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>BLOCK63 | IP block status<br>This bit provides the clock status of PIT_2 in partition 1.<br>0b - Clock is not running. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Clock is running. |
| 30<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 29<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 28<br>BLOCK60 | IP block status<br>This bit provides the clock status of FMU_alt in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 27<br>BLOCK59 | IP block status<br>This bit provides the clock status of FMU in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 26<br>BLOCK58 | IP block status<br>This bit provides the clock status of PMC in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 25<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 24<br>BLOCK56 | IP block status<br>This bit provides the clock status of PLL in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 23<br>BLOCK55 | IP block status<br>This bit provides the clock status of MC_ME in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 22<br>BLOCK54 | IP block status<br>This bit provides the clock status of MC_CGM in partition 1.<br>0b - Clock is not running. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Clock is running. |
| 21<br>BLOCK53 | IP block status<br>This bit provides the clock status of FXOSC in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 20<br>BLOCK52 | IP block status<br>This bit provides the clock status of FIRC in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 19<br>BLOCK51 | IP block status<br>This bit provides the clock status of SXOSC in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 18<br>BLOCK50 | IP block status<br>This bit provides the clock status of SIRC in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 17<br>BLOCK49 | IP block status<br>This bit provides the clock status of TSPC in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 16<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 15<br>BLOCK47 | IP block status<br>This bit provides the clock status of CMU_0-5 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 14<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 13 | IP block status<br>This bit provides the clock status of WKPU in partition 1. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| BLOCK45 | 0b - Clock is not running.<br>1b - Clock is running. |
| 12<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 11<br>BLOCK43 | IP block status<br>This bit provides the clock status of DCM in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 10<br>BLOCK42 | IP block status<br>This bit provides the clock status of SIUL_VIRTWRAPPER_PDAC3 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 9<br>BLOCK41 | IP block status<br>This bit provides the clock status of SIUL_VIRTWRAPPER_PDAC2_M7_1 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 8<br>BLOCK40 | IP block status<br>This bit provides the clock status of SIUL_VIRTWRAPPER_PDAC2_M7_1 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 7<br>BLOCK39 | IP block status<br>This bit provides the clock status of SIUL_VIRTWRAPPER_PDAC1_M7_0 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 6<br>BLOCK38 | IP block status<br>This bit provides the clock status of SIUL_VIRTWRAPPER_PDAC1_M7_0 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 5<br>BLOCK37 | IP block status<br>This bit provides the clock status of SIUL_VIRTWRAPPER_PDAC0_HSE in partition 1. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Clock is not running.<br>1b - Clock is running. |
| 4<br>BLOCK36 | IP block status<br>This bit provides the clock status of SIUL_VIRTWRAPPER_PDAC0_HSE in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 3<br>BLOCK35 | IP block status<br>This bit provides the clock status of MC_RGM in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 2<br>BLOCK34 | IP block status<br>This bit provides the clock status of RTC in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 1<br>BLOCK33 | IP block status<br>This bit provides the clock status of DMAMUX_1 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 0<br>BLOCK32 | IP block status<br>This bit provides the clock status of DMAMUX_0 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |

### 43.7.23 Partition 1 COFB Set 2 Clock Status Register (PRTN1_COFB2_STAT)

#### Offset

| Register | Offset |
|---|---|
| PRTN1_COFB2_STAT | 318h |

#### Function

This register provides the status of set 2 of COFBs inside partition 1.

**NOTE**

The reset value of this register can vary depending on the availability of active clock pulses inside partition 1.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BLOCK95 | 0 | BLOCK93 | BLOCK92 | BLOCK91 | 0 | BLOCK89 | BLOCK88 | BLOCK87 | BLOCK86 | BLOCK85 | BLOCK84 | 0 | 0 | BLOCK81 | BLOCK80 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BLOCK79 | BLOCK78 | BLOCK77 | BLOCK76 | BLOCK75 | BLOCK74 | BLOCK73 | 0 | 0 | BLOCK70 | BLOCK69 | BLOCK68 | BLOCK67 | BLOCK66 | BLOCK65 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>BLOCK95 | IP block status<br>This bit provides the clock status of TempSense in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 30<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 29<br>BLOCK93 | IP block status<br>This bit provides the clock status of LPCMP_1 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 28<br>BLOCK92 | IP block status<br>This bit provides the clock status of LPCMP_0 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 27<br>BLOCK91 | IP block status<br>This bit provides the clock status of SAI_0 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 26<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 25 | IP block status |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| BLOCK89 | This bit provides the clock status of LPSPI_3 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 24<br>BLOCK88 | IP block status<br>This bit provides the clock status of LPSPI_2 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 23<br>BLOCK87 | IP block status<br>This bit provides the clock status of LPSPI_1 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 22<br>BLOCK86 | IP block status<br>This bit provides the clock status of LPSPI_0 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 21<br>BLOCK85 | IP block status<br>This bit provides the clock status of LPI2C_1 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 20<br>BLOCK84 | IP block status<br>This bit provides the clock status of LPI2C_0 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 19<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 18<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 17<br>BLOCK81 | IP block status<br>This bit provides the clock status of LPUART_7 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 16<br>BLOCK80 | IP block status<br>This bit provides the clock status of LPUART_6 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 15<br>BLOCK79 | IP block status<br>This bit provides the clock status of LPUART_5 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 14<br>BLOCK78 | IP block status<br>This bit provides the clock status of LPUART_4 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 13<br>BLOCK77 | IP block status<br>This bit provides the clock status of LPUART_3 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 12<br>BLOCK76 | IP block status<br>This bit provides the clock status of LPUART_2 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 11<br>BLOCK75 | IP block status<br>This bit provides the clock status of LPUART_1 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 10<br>BLOCK74 | IP block status<br>This bit provides the clock status of LPUART_0 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 9<br>BLOCK73 | IP block status<br>This bit provides the clock status of FlexIO in partition 1.<br>0b - Clock is not running. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Clock is running. |
| 8<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 7<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 6<br>BLOCK70 | IP block status<br>This bit provides the clock status of FlexCAN_5 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 5<br>BLOCK69 | IP block status<br>This bit provides the clock status of FlexCAN_4 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 4<br>BLOCK68 | IP block status<br>This bit provides the clock status of FlexCAN_3 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 3<br>BLOCK67 | IP block status<br>This bit provides the clock status of FlexCAN_2 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 2<br>BLOCK66 | IP block status<br>This bit provides the clock status of FlexCAN_1 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 1<br>BLOCK65 | IP block status<br>This bit provides the clock status of FlexCAN_0 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 0 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | This field is reserved and read returns zeros. |

## 43.7.24 Partition 1 COFB Set 3 Clock Status Register (PRTN1_COFB3_STAT)

### Offset

| Register | Offset |
|---|---|
| PRTN1_COFB3_STAT | 31Ch |

### Function

This register provides the status of set 3 of COFBs inside partition 1.

**NOTE**

The reset value of this register can vary depending on the availability of active clock pulses inside partition 1.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | BLOCK110 | 0 | BLOCK108 | BLOCK107 | BLOCK106 | BLOCK105 | BLOCK104 | BLOCK103 | BLOCK102 | BLOCK101 | 0 | BLOCK99 | BLOCK98 | BLOCK97 | BLOCK96 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 30<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 29<br>— | Reserved<br>This field is reserved and read returns zeros. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 28<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 27<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 26<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 25<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 24<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 23<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 22<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 21<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 20<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 19<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 18<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 17<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 16<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 15<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 14 | IP block status |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| BLOCK110 | This bit provides the clock status of Reserved block in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 13<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 12<br>BLOCK108 | IP block status<br>This bit provides the clock status of SELF-TEST GPR in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 11<br>BLOCK107 | IP block status<br>This bit provides the clock status of Reserved block in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 10<br>BLOCK106 | IP block status<br>This bit provides the clock status of Reserved block in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 9<br>BLOCK105 | IP block status<br>This bit provides the clock status of Reserved block in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 8<br>BLOCK104 | IP block status<br>This bit provides the clock status of STCU2 in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 7<br>BLOCK103 | IP block status<br>This bit provides the clock status of Reserved block in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 6<br>BLOCK102 | IP block status<br>This bit provides the clock status of Reserved block in partition 1. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Clock is not running.<br>1b - Clock is running. |
| 5<br>BLOCK101 | IP block status<br>This bit provides the clock status of JDC in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 4<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 3<br>BLOCK99 | IP block status<br>This bit provides the clock status of HSE in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 2<br>BLOCK98 | IP block status<br>This bit provides the clock status of MTR in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 1<br>BLOCK97 | IP block status<br>This bit provides the clock status of FCCU in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 0<br>BLOCK96 | IP block status<br>This bit provides the clock status of CRC in partition 1.<br>0b - Clock is not running.<br>1b - Clock is running. |

### 43.7.25 Partition 1 COFB Set 0 Clock Enable Register (PRTN1_COFB0_CLKEN)

**Offset**

| Register | Offset |
|---|---|
| PRTN1_COFB0_CLKEN | 330h |

### Function

This register provides clock control signaling to the individual COFBs in set 0 inside partition 1. Whenever a partition clock enable (non-core) hardware process is initiated, the value of logic-1 in the corresponding bit locations of this register enables the clock to the corresponding block in the partition.

**NOTE**

The reset value of this register is not defined and is as per the availability of the clock source. See Chip-specific MC_ME information for clock source availability.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REQ31 | 0 | REQ29 | REQ28 | 0 | 0 | 0 | REQ24 | REQ23 | REQ22 | REQ21 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REQ15 | REQ14 | REQ13 | REQ12 | REQ11 | REQ10 | REQ9 | REQ8 | REQ7 | REQ6 | REQ5 | REQ4 | REQ3 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

### Fields

| Field | Function |
|---|---|
| 31<br>REQ31 | Clock enable<br>This bit provides the clock enable control for INTM in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 30<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 29<br>REQ29 | Clock enable<br>This bit provides the clock enable control for STM_0 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 28<br>REQ28 | Clock enable<br>This bit provides the clock enable control for SWT_0 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 27 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | This field is reserved and read returns zeros. |
| 26<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 25<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 24<br>REQ24 | Clock enable<br>This bit provides the clock enable control for MSCM in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 23<br>REQ23 | Clock enable<br>This bit provides the clock enable control for ERM in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 22<br>REQ22 | Clock enable<br>This bit provides the clock enable control for EIM in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 21<br>REQ21 | Clock enable<br>This bit provides the clock enable control for SDA-AP in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 20<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 19<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 18<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 17<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 16 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | This field is reserved and read returns zeros. |
| 15<br>REQ15 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 14<br>REQ14 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 13<br>REQ13 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 12<br>REQ12 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 11<br>REQ11 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 10<br>REQ10 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 9<br>REQ9 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 8<br>REQ8 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 1. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Clock is turned off.<br>1b - Clock is turned on. |
| 7<br>REQ7 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 6<br>REQ6 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 5<br>REQ5 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 4<br>REQ4 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 3<br>REQ3 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 2<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 1<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 0<br>— | Reserved<br>This field is reserved and read returns zeros. |

## 43.7.26 Partition 1 COFB Set 1 Clock Enable Register (PRTN1_COFB1_CLKEN)

### Offset

| Register | Offset |
|---|---|
| PRTN1_COFB1_CLKEN | 334h |

### Function

This register provides clock control signaling to the individual COFBs in set 1 inside partition 1. Whenever a partition clock enable (non-core) hardware process is initiated, the value of logic-1 in the corresponding bit locations of this register enables the clock to the corresponding block in the partition.

**NOTE**

The reset value of this register is not defined and is as per the availability of the clock source. See Chip-specific MC_ME information for clock source availability.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REQ63 | 0 | 0 | 0 | 0 | 0 | 0 | REQ56 | 0 | 0 | REQ53 | 0 | REQ51 | 0 | REQ49 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REQ47 | 0 | REQ45 | 0 | 0 | REQ42 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | REQ34 | REQ33 | REQ32 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>REQ63 | Clock enable<br>This bit provides the clock enable control for PIT_2 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 30<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 29<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 28 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | This field is reserved and read returns zeros. |
| 27<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 26<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 25<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 24<br>REQ56 | Clock enable<br>This bit provides the clock enable control for PLL in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 23<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 22<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 21<br>REQ53 | Clock enable<br>This bit provides the clock enable control for FXOSC in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 20<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 19<br>REQ51 | Clock enable<br>This bit provides the clock enable control for SXOSC in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 18<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 17<br>REQ49 | Clock enable<br>This bit provides the clock enable control for TSPC in partition 1. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Clock is turned off.<br>1b - Clock is turned on. |
| 16<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 15<br>REQ47 | Clock enable<br>This bit provides the clock enable control for CMU_0-5 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 14<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 13<br>REQ45 | Clock enable<br>This bit provides the clock enable control for WKPU in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 12<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 11<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 10<br>REQ42 | Clock enable<br>This bit provides the clock enable control for SIUL_VIRTWRAPPER_PDAC3 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 9<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 8<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 7<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 6<br>— | Reserved<br>This field is reserved and read returns zeros. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 5<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 4<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 3<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 2<br>REQ34 | Clock enable<br>This bit provides the clock enable control for RTC in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 1<br>REQ33 | Clock enable<br>This bit provides the clock enable control for DMAMUX_1 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 0<br>REQ32 | Clock enable<br>This bit provides the clock enable control for DMAMUX_0 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |

## 43.7.27 Partition 1 COFB Set 2 Clock Enable Register (PRTN1_COFB2_CLKEN)

### Offset

| Register | Offset |
|---|---|
| PRTN1_COFB2_CLKEN | 338h |

### Function

This register provides clock control signaling to the individual COFBs in set 2 inside partition 1. Whenever a partition clock enable (non-core) hardware process is initiated, the value of logic-1 in the corresponding bit locations of this register enables the clock to the corresponding block in the partition.

**NOTE**

The reset value of this register is not defined and is as per the availability of the clock source. See Chip-specific MC_ME information for clock source availability.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REQ95 | 0 | REQ93 | REQ92 | REQ91 | 0 | REQ89 | REQ88 | REQ87 | REQ86 | REQ85 | REQ84 | 0 | 0 | REQ81 | REQ80 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REQ79 | REQ78 | REQ77 | REQ76 | REQ75 | REQ74 | REQ73 | 0 | 0 | REQ70 | REQ69 | REQ68 | REQ67 | REQ66 | REQ65 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>REQ95 | Clock enable<br>This bit provides the clock enable control for TempSense in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 30<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 29<br>REQ93 | Clock enable<br>This bit provides the clock enable control for LPCMP_1 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 28<br>REQ92 | Clock enable<br>This bit provides the clock enable control for LPCMP_0 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 27<br>REQ91 | Clock enable<br>This bit provides the clock enable control for SAI_0 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 26<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 25 | Clock enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| REQ89 | This bit provides the clock enable control for LPSPI_3 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 24<br>REQ88 | Clock enable<br>This bit provides the clock enable control for LPSPI_2 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 23<br>REQ87 | Clock enable<br>This bit provides the clock enable control for LPSPI_1 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 22<br>REQ86 | Clock enable<br>This bit provides the clock enable control for LPSPI_0 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 21<br>REQ85 | Clock enable<br>This bit provides the clock enable control for LPI2C_1 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 20<br>REQ84 | Clock enable<br>This bit provides the clock enable control for LPI2C_0 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 19<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 18<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 17<br>REQ81 | Clock enable<br>This bit provides the clock enable control for LPUART_7 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 16<br>REQ80 | Clock enable<br>This bit provides the clock enable control for LPUART_6 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 15<br>REQ79 | Clock enable<br>This bit provides the clock enable control for LPUART_5 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 14<br>REQ78 | Clock enable<br>This bit provides the clock enable control for LPUART_4 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 13<br>REQ77 | Clock enable<br>This bit provides the clock enable control for LPUART_3 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 12<br>REQ76 | Clock enable<br>This bit provides the clock enable control for LPUART_2 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 11<br>REQ75 | Clock enable<br>This bit provides the clock enable control for LPUART_1 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 10<br>REQ74 | Clock enable<br>This bit provides the clock enable control for LPUART_0 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 9<br>REQ73 | Clock enable<br>This bit provides the clock enable control for FlexIO in partition 1.<br>0b - Clock is turned off. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Clock is turned on. |
| 8<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 7<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 6<br>REQ70 | Clock enable<br>This bit provides the clock enable control for FlexCAN_5 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 5<br>REQ69 | Clock enable<br>This bit provides the clock enable control for FlexCAN_4 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 4<br>REQ68 | Clock enable<br>This bit provides the clock enable control for FlexCAN_3 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 3<br>REQ67 | Clock enable<br>This bit provides the clock enable control for FlexCAN_2 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 2<br>REQ66 | Clock enable<br>This bit provides the clock enable control for FlexCAN_1 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 1<br>REQ65 | Clock enable<br>This bit provides the clock enable control for FlexCAN_0 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 0 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | This field is reserved and read returns zeros. |

## 43.7.28 Partition 1 COFB Set 3 Clock Enable Register (PRTN1_COFB3_CLKEN)

### Offset

| Register | Offset |
|---|---|
| PRTN1_COFB3_CLKEN | 33Ch |

### Function

This register provides clock control signaling to the individual COFBs in set 3 inside partition 1. Whenever a partition clock enable (non-core) hardware process is initiated, the value of logic-1 in the corresponding bit locations of this register enables the clock to the corresponding block in the partition.

**NOTE**

The reset value of this register is not defined and is as per the availability of the clock source. See Chip-specific MC_ME information for clock source availability.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | REQ104 | 0 | REQ102 | 0 | 0 | 0 | 0 | 0 | REQ96 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 30<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 29 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | This field is reserved and read returns zeros. |
| 28<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 27<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 26<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 25<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 24<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 23<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 22<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 21<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 20<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 19<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 18<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 17<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 16<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 15<br>— | Reserved<br>This field is reserved and read returns zeros. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 14<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 13<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 12<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 11<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 10<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 9<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 8<br>REQ104 | Clock enable<br>This bit provides the clock enable control for STCU2 in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 7<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 6<br>REQ102 | Clock enable<br>This bit provides the clock enable control for Reserved block in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 5<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 4<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 3<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 2 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | This field is reserved and read returns zeros. |
| 1<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 0<br>REQ96 | Clock enable<br>This bit provides the clock enable control for CRC in partition 1.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |

## 43.7.29 Partition 2 Process Configuration Register (PRTN2_PCONF)

### Offset

| Register | Offset |
|---|---|
| PRTN2_PCONF | 500h |

### Function

This register provides a configuration for the hardware processes corresponding to partition 2. Each of the configuration bit corresponds to the 'nature' of the processes; for example, enabling/disabling and the trigger is controlled by the corresponding field in the PRTN2_PUPD register. When valid KEY combinations are written onto the CTL_KEY register, the PRTN2_PCONF and PRTN2_PUPD registers are used to determine the hardware processes to be executed. These are triggered in parallel and independent of each other. All dependent processes should be requested one after another from the software.

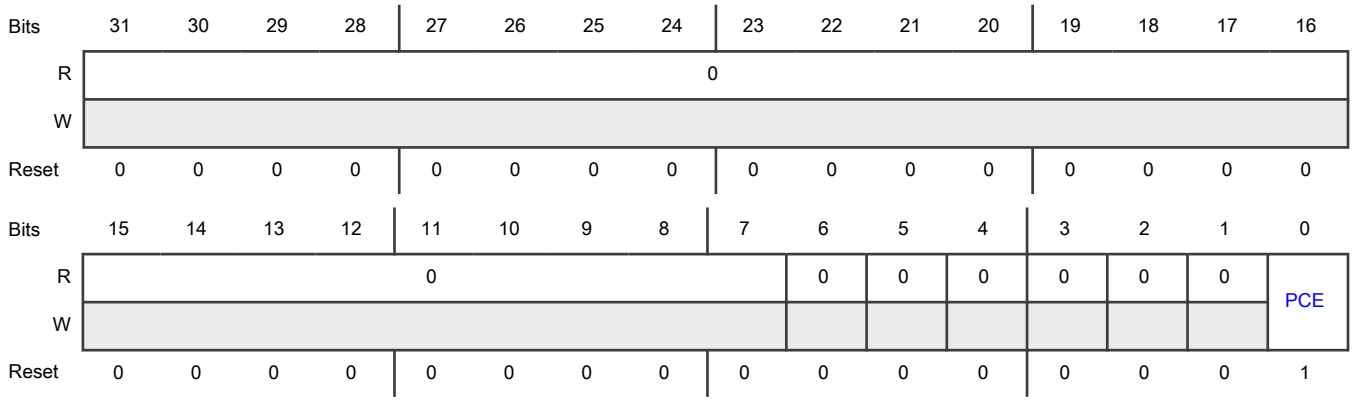
**NOTE**

The partition clock enable/disable are not standalone and must be done coherently in a fixed sequence. For details, see Software Reset Partition Turn-On Flow Chart and Software reset partition turn-off flowchart in Reset chapter.

**NOTE**

See chip-specific MC_ME information to check if this register is implemented on chip.

### Diagram



**Fields**

| Field | Function |
|---|---|
| 31-7<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 6<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 5<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 4<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 3<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 2<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 1<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 0<br>PCE | Partition clock enable<br>This bit controls whether the clock to IPs (other than core(s)) in the partition should be enabled or disabled.<br>0b - Disable the clock to IPs<br>1b - Enable the clock to IPs |

## 43.7.30 Partition 2 Process Update Register (PRTN2_PUPD)

**Offset**

| Register | Offset |
|---|---|
| PRTN2_PUPD | 504h |

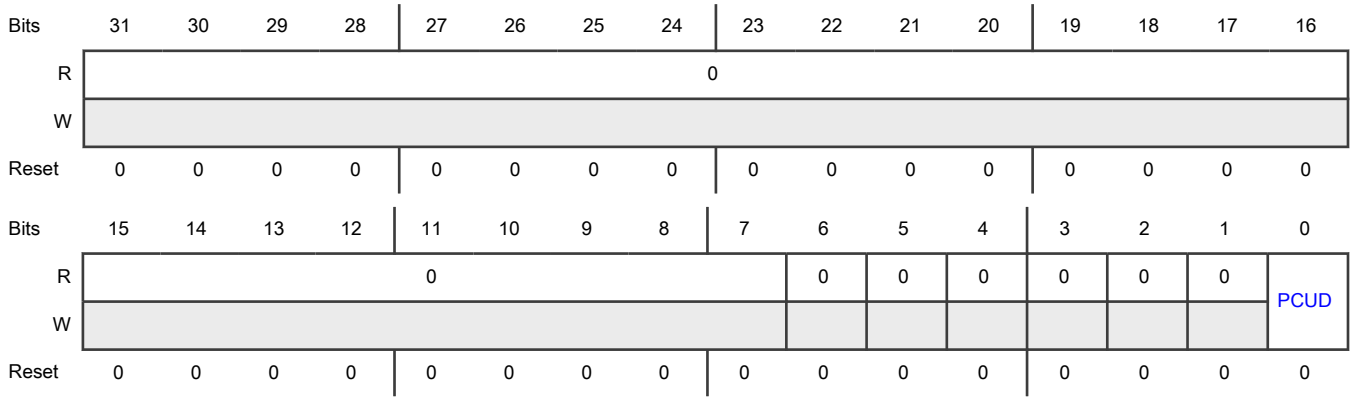
**Function**

This register provides trigger signaling for the hardware processes corresponding to partition 2. Each of the control bit acts as a trigger for the corresponding hardware processes. When valid KEY combinations are written onto the CTL_KEY register, the hardware checks the bit fields that are programmed as logic-1 in this register, and then triggers the hardware process per the value in the corresponding bit field in the PRTN2_PCONF register. When the hardware process is finished the corresponding bit in this register is auto-cleared to logic-0.

**NOTE**

See chip-specific MC_ME information to check if this register is implemented on chip.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-7<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 6<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 5<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 4<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 3<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 2<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 1<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 0<br>PCUD | Partition clock update<br>This bit controls whether the hardware processes for enabling/disabling the clock to IPs (other than core(s)) in the partition should be triggered or not.<br>0b - Do not trigger the hardware process<br>1b - Trigger the hardware process |

## 43.7.31 Partition 2 Status Register (PRTN2_STAT)

### Offset

| Register | Offset |
|---|---|
| PRTN2_STAT | 508h |

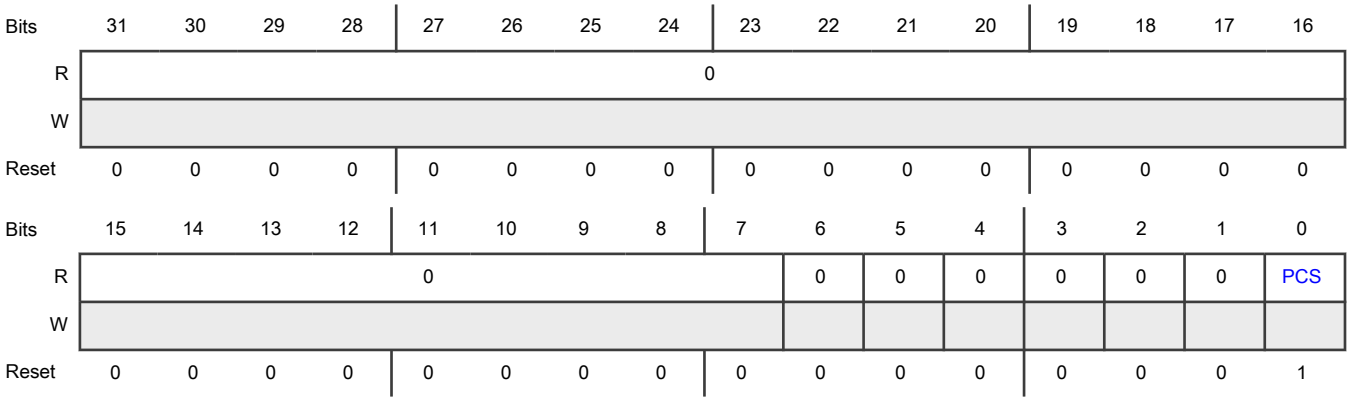
### Function

This register provides the current status of the control signals from the partition 2.

**NOTE**

See chip-specific MC_ME information to check if this register is implemented on chip.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-7<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 6<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 5<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 4<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 3<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 2 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | This field is reserved and read returns zeros. |
| 1<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 0<br>PCS | Partition clock status<br>This bit provides the status of the clock to partition.<br>0b - Clock is inactive<br>1b - Clock is active |

## 43.7.32 Partition 2 COFB Set 0 Clock Status Register (PRTN2_COFB0_STAT)

### Offset

| Register | Offset |
|---|---|
| PRTN2_COFB0_STAT | 510h |

### Function

This register provides the status of set 0 of COFBs inside partition 2.

**NOTE**

The reset value of this register can vary depending on the availability of active clock pulses inside partition 2.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | BLOCK29 | 0 | 0 | 0 | BLOCK25 | BLOCK24 | BLOCK23 | BLOCK22 | BLOCK21 | BLOCK20 | BLOCK19 | BLOCK18 | BLOCK17 | BLOCK16 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BLOCK15 | BLOCK14 | BLOCK13 | BLOCK12 | BLOCK11 | BLOCK10 | BLOCK9 | BLOCK8 | BLOCK7 | BLOCK6 | BLOCK5 | BLOCK4 | 0 | 0 | BLOCK1 | BLOCK0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 30<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 29<br>BLOCK29 | IP block status<br>This bit provides the clock status of STM_1 in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 28<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 27<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 26<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 25<br>BLOCK25 | IP block status<br>This bit provides the clock status of PRAM_1 in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 24<br>BLOCK24 | IP block status<br>This bit provides the clock status of SEMA42 in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 23<br>BLOCK23 | IP block status<br>This bit provides the clock status of eDMA in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 22<br>BLOCK22 | IP block status<br>This bit provides the clock status of eDMA in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 21<br>BLOCK21 | IP block status<br>This bit provides the clock status of eDMA in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 20<br>BLOCK20 | IP block status<br>This bit provides the clock status of eDMA in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 19<br>BLOCK19 | IP block status<br>This bit provides the clock status of eDMA in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 18<br>BLOCK18 | IP block status<br>This bit provides the clock status of eDMA in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 17<br>BLOCK17 | IP block status<br>This bit provides the clock status of eDMA in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 16<br>BLOCK16 | IP block status<br>This bit provides the clock status of eDMA in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 15<br>BLOCK15 | IP block status<br>This bit provides the clock status of eDMA in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 14<br>BLOCK14 | IP block status<br>This bit provides the clock status of eDMA in partition 2.<br>0b - Clock is not running. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Clock is running. |
| 13<br>BLOCK13 | IP block status<br>This bit provides the clock status of eDMA in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 12<br>BLOCK12 | IP block status<br>This bit provides the clock status of eDMA in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 11<br>BLOCK11 | IP block status<br>This bit provides the clock status of eDMA in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 10<br>BLOCK10 | IP block status<br>This bit provides the clock status of eDMA in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 9<br>BLOCK9 | IP block status<br>This bit provides the clock status of eDMA in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 8<br>BLOCK8 | IP block status<br>This bit provides the clock status of eDMA in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 7<br>BLOCK7 | IP block status<br>This bit provides the clock status of eDMA in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 6<br>BLOCK6 | IP block status<br>This bit provides the clock status of eDMA in partition 2. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Clock is not running.<br>1b - Clock is running. |
| 5<br>BLOCK5 | IP block status<br>This bit provides the clock status of eDMA in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 4<br>BLOCK4 | IP block status<br>This bit provides the clock status of eDMA in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 3<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 2<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 1<br>BLOCK1 | IP block status<br>This bit provides the clock status of XBIC in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 0<br>BLOCK0 | IP block status<br>This bit provides the clock status of XBIC in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |

### 43.7.33 Partition 2 COFB Set 1 Clock Status Register (PRTN2_COFB1_STAT)

#### Offset

| Register | Offset |
|---|---|
| PRTN2_COFB1_STAT | 514h |

#### Function

This register provides the status of set 1 of COFBs inside partition 2.

**NOTE**

The reset value of this register can vary depending on the availability of active clock pulses inside partition 2.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BLOCK63 | BLOCK62 | 0 | 0 | BLOCK59 | BLOCK58 | 0 | 0 | BLOCK55 | 0 | 0 | 0 | BLOCK51 | 0 | 0 | BLOCK48 |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BLOCK47 | 0 | 0 | 0 | 0 | BLOCK42 | BLOCK41 | BLOCK40 | BLOCK39 | BLOCK38 | BLOCK37 | BLOCK36 | BLOCK35 | 0 | 0 | BLOCK32 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>BLOCK63 | IP block status<br>This bit provides the clock status of CM7_1_TCM in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 30<br>BLOCK62 | IP block status<br>This bit provides the clock status of CM7_0_TCM in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 29<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 28<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 27<br>BLOCK59 | IP block status<br>This bit provides the clock status of HSE in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 26<br>BLOCK58 | IP block status<br>This bit provides the clock status of LPCMP_2 in partition 2. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Clock is not running.<br>1b - Clock is running. |
| 25<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 24<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 23<br>BLOCK55 | IP block status<br>This bit provides the clock status of SAI_1 in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 22<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 21<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 20<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 19<br>BLOCK51 | IP block status<br>This bit provides the clock status of QuadSPI in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 18<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 17<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 16<br>BLOCK48 | IP block status<br>This bit provides the clock status of LPSPI_5 in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 15<br>BLOCK47 | IP block status<br>This bit provides the clock status of LPSPI_4 in partition 2. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Clock is not running.<br>1b - Clock is running. |
| 14<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 13<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 12<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 11<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 10<br>BLOCK42 | IP block status<br>This bit provides the clock status of LPUART_15 in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 9<br>BLOCK41 | IP block status<br>This bit provides the clock status of LPUART_14 in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 8<br>BLOCK40 | IP block status<br>This bit provides the clock status of LPUART_13 in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 7<br>BLOCK39 | IP block status<br>This bit provides the clock status of LPUART_12 in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 6<br>BLOCK38 | IP block status<br>This bit provides the clock status of LPUART_11 in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 5<br>BLOCK37 | IP block status<br>This bit provides the clock status of LPUART_10 in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 4<br>BLOCK36 | IP block status<br>This bit provides the clock status of LPUART_9 in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 3<br>BLOCK35 | IP block status<br>This bit provides the clock status of LPUART_8 in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |
| 2<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 1<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 0<br>BLOCK32 | IP block status<br>This bit provides the clock status of EMAC in partition 2.<br>0b - Clock is not running.<br>1b - Clock is running. |

## 43.7.34 Partition 2 COFB Set 0 Clock Enable Register (PRTN2_COFB0_CLKEN)

### Offset

| Register | Offset |
|---|---|
| PRTN2_COFB0_CLKEN | 530h |

### Function

This register provides clock control signaling to the individual COFBs in set 0 inside partition 2. Whenever a partition clock enable (non-core) hardware process is initiated, the value of logic-1 in the corresponding bit locations of this register enables the clock to the corresponding block in the partition.

**NOTE**

The reset value of this register is not defined and is as per the availability of the clock source. See Chip-specific MC_ME information for clock source availability.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | REQ29 | 0 | 0 | 0 | 0 | REQ24 | REQ23 | REQ22 | REQ21 | REQ20 | REQ19 | REQ18 | REQ17 | REQ16 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REQ15 | REQ14 | REQ13 | REQ12 | REQ11 | REQ10 | REQ9 | REQ8 | REQ7 | REQ6 | REQ5 | REQ4 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 30<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 29<br>REQ29 | Clock enable<br>This bit provides the clock enable control for STM_1 in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 28<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 27<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 26<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 25<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 24<br>REQ24 | Clock enable<br>This bit provides the clock enable control for SEMA42 in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 23 | Clock enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| REQ23 | This bit provides the clock enable control for eDMA in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 22<br>REQ22 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 21<br>REQ21 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 20<br>REQ20 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 19<br>REQ19 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 18<br>REQ18 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 17<br>REQ17 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 16<br>REQ16 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 15<br>REQ15 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 14<br>REQ14 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 13<br>REQ13 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 12<br>REQ12 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 11<br>REQ11 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 10<br>REQ10 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 9<br>REQ9 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 8<br>REQ8 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 2.<br>0b - Clock is turned off. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Clock is turned on. |
| 7<br>REQ7 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 6<br>REQ6 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 5<br>REQ5 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 4<br>REQ4 | Clock enable<br>This bit provides the clock enable control for eDMA in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 3<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 2<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 1<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 0<br>— | Reserved<br>This field is reserved and read returns zeros. |

## 43.7.35 Partition 2 COFB Set 1 Clock Enable Register (PRTN2_COFB1_CLKEN)

**Offset**

| Register | Offset |
|---|---|
| PRTN2_COFB1_CLKEN | 534h |

### Function

This register provides clock control signaling to the individual COFBs in set 1 inside partition 2. Whenever a partition clock enable (non-core) hardware process is initiated, the value of logic-1 in the corresponding bit locations of this register enables the clock to the corresponding block in the partition.

**NOTE**

The reset value of this register is not defined and is as per the availability of the clock source. See Chip-specific MC_ME information for clock source availability.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REQ63 | REQ62 | 0 | 0 | 0 | REQ58 | 0 | 0 | REQ55 | 0 | 0 | 0 | REQ51 | 0 | 0 | REQ48 |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | REQ47 | 0 | 0 | 0 | 0 | REQ42 | REQ41 | REQ40 | REQ39 | REQ38 | REQ37 | REQ36 | REQ35 | 0 | 0 | REQ32 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>REQ63 | Clock enable<br>This bit provides the clock enable control for CM7_1_TCM in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 30<br>REQ62 | Clock enable<br>This bit provides the clock enable control for CM7_0_TCM in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 29<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 28<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 27<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 26 | Clock enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| REQ58 | This bit provides the clock enable control for LPCMP_2 in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 25<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 24<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 23<br>REQ55 | Clock enable<br>This bit provides the clock enable control for SAI_1 in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 22<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 21<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 20<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 19<br>REQ51 | Clock enable<br>This bit provides the clock enable control for QuadSPI in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 18<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 17<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 16<br>REQ48 | Clock enable<br>This bit provides the clock enable control for LPSPI_5 in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 15 | Clock enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| REQ47 | This bit provides the clock enable control for LPSPI_4 in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 14<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 13<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 12<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 11<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 10<br>REQ42 | Clock enable<br>This bit provides the clock enable control for LPUART_15 in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 9<br>REQ41 | Clock enable<br>This bit provides the clock enable control for LPUART_14 in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 8<br>REQ40 | Clock enable<br>This bit provides the clock enable control for LPUART_13 in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 7<br>REQ39 | Clock enable<br>This bit provides the clock enable control for LPUART_12 in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 6<br>REQ38 | Clock enable<br>This bit provides the clock enable control for LPUART_11 in partition 2.<br>0b - Clock is turned off. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Clock is turned on. |
| 5<br>REQ37 | Clock enable<br>This bit provides the clock enable control for LPUART_10 in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 4<br>REQ36 | Clock enable<br>This bit provides the clock enable control for LPUART_9 in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 3<br>REQ35 | Clock enable<br>This bit provides the clock enable control for LPUART_8 in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |
| 2<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 1<br>— | Reserved<br>This field is reserved and read returns zeros. |
| 0<br>REQ32 | Clock enable<br>This bit provides the clock enable control for EMAC in partition 2.<br>0b - Clock is turned off.<br>1b - Clock is turned on. |

## 43.8 Glossary

**WFI** Wait for interrupt

**COFB** Collection of functional blocks also referred as number of peripherals

# Chapter 44
# Power Control Unit (MC_PCU)

## 44.1 Introduction

The power control unit (MC_PCU) is used for initiating a Standby mode entry that reduces the overall chip power consumption. Power can be saved by disconnecting parts of the chip from the power supply. The blocks inside the chip are grouped into multiple parts having this capability, which are called "power domains".

When a power domain is disconnected from the supply, the power consumption is reduced to zero and the configuration of the every core and module that belongs to such power domain is completely lost. When you reconnect a power domain to the supply voltage, the domain draws an increased current until the power domain reaches its operational voltage. Maximum power saving is achieved by entering the Standby mode.

After the MC_ME asserts a standby entry request, MC_PCU initiates the power sequence, which is non-retractable and includes the handshake with the chip power management controller. The power-up/down sequences are handled by FSMs to ensure a smooth and safe transition into and out of the Standby mode. Exiting the Standby mode can only be done through a system wakeup event, power-on reset, destructive reset, or a functional reset.

## 44.2 Power sequence FSM

MC_PCU implements an FSM to initiate the power sequencing of the Standby mode entry/exit sequence for the chip.

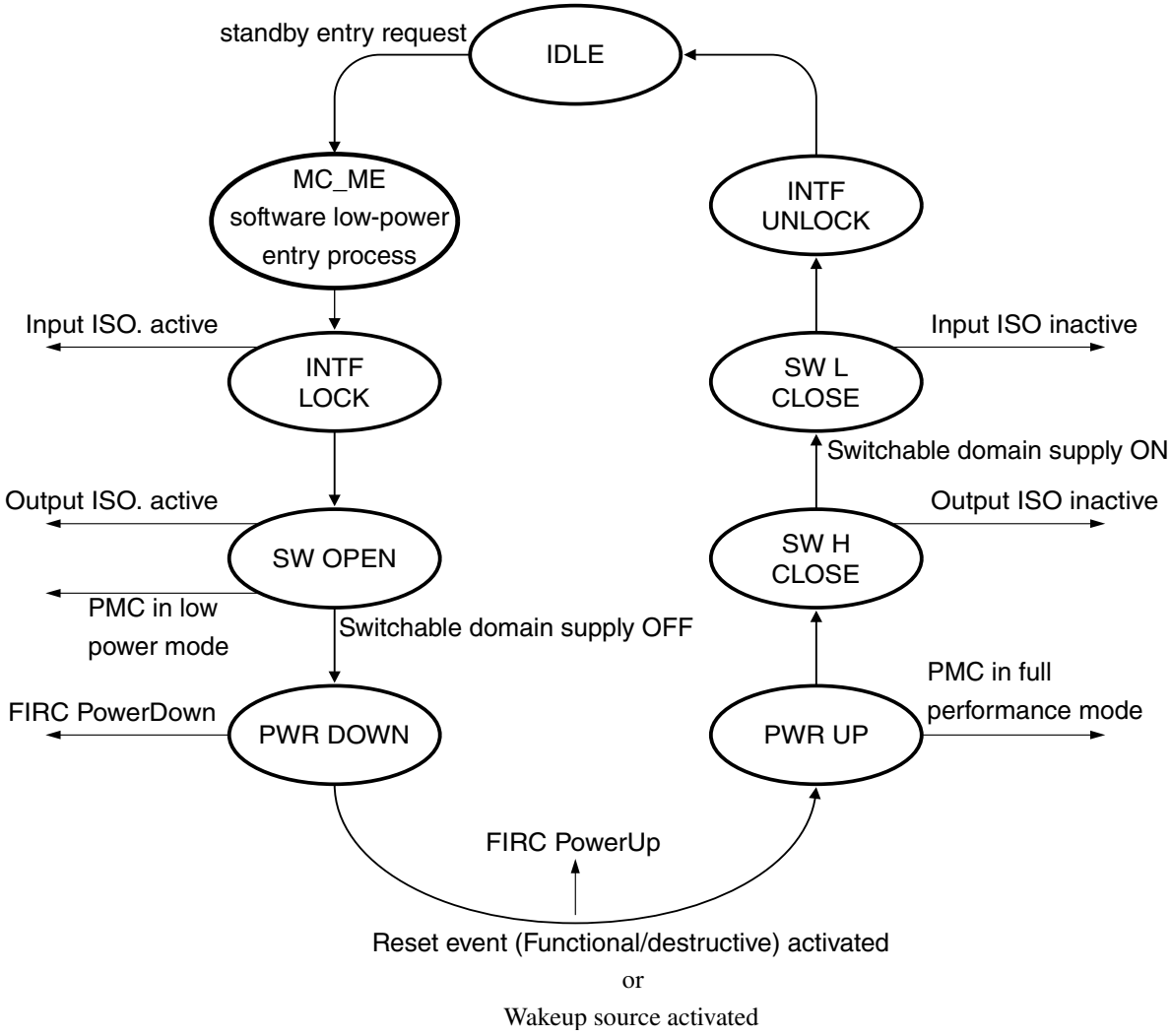



Figure 168. MC_PCU FSM

**NOTE**

When destructive reset is asserted from MC_RGM, MC_PCU FSM moves to the IDLE state immediately. Indication of this is not shown in the above figure.

Table 217. MC_PCU FSM transition description

| State | Name | Exit condition | Signal controlled | Signal monitored |
|---|---|---|---|---|
| IDLE | Idle state | Standby request received from MC_ME | - | - |
| MC_ME software low power entry process | MC_ME software low power entry process | Software entry sequence completed (SW4 process completed) | - | - |
| INTF LOCK | Interface lock state | Input isolation active | Input isolation activation | Input isolation active |

*Table continues on the next page...*

Table 217. MC_PCU FSM transition description (continued)

| State | Name | Exit condition | Signal controlled | Signal monitored |
|---|---|---|---|---|
| SW OPEN | Switch open state | Output isolation active, Switchable domain supply turned off | Output isolation activation, low power mode request to PMC | Output isolation active, Switchable domain supply |
| PWR DOWN | Power down state | Wakeup/functional reset occurrence | FIRC power down | Wakeup and functional reset |
| PWR UP | Power up state | PMC in full performance mode | - | - |
| SW H CLOSE | Switch H close state | Switchable domain supply turned on | - | Switchable domain supply |
| SW L CLOSE | Switch L close state | Input isolation inactive | Input isolation inactive | Input isolation active |
| INTF UNLOCK | Interface unlock | - | - | - |

## 44.3 Glossary

**FSM** Finite state machine

# Chapter 45
# Wakeup Unit (WKPU)

## 45.1 Chip-specific WKPU information

### 45.1.1 WKPU configuration on the chip

WKPU provides the following features:

- A configurable, low-power wake-up capability to the chip from multiple configurable asynchronous wake-up events.
- Support for four internal and 60 external sources that can generate interrupts or wake-up events.
- Support for an NMI input.

**Table 218. WKPU configuration on the chip**

| Number of sources, interrupts, and vectors | Value | Description |
|---|---|---|
| NMI sources | 1 | Single NMI pin routed to all application cores |
| Internal wake-up sources | 4 | • SWT_0 wake-up, RTC-API wake-up<br>• RTC timeout<br>• Analog comparator round robin wake-up (from LPCMP_0, 1, and 2 round robin wake-up)[1]<br>• RTI wake-up |
| External wake-up sources | 60 | 60 GPIO pins (WKPU[0]-WKPU[59]) support the wake-up functionality. See the IOMUX file attached to this document for details. |
| Glitch filters for external interrupts | 60 | Glitch filter on each external wake-up source |
| External interrupt vectors | 8 | — |

1. Both wake-up sources 0 (RTC-API) and 2 (CMP_*x* trigger mode wake-up) use RTC-API wake-up, with Trigger mode having a higher priority. This means, if you configure LPCMP_*x*.RRCR0[ERR_EN] for either of the comparators, the RTC_API wake-up is used only for the CMP_*x* trigger mode operation. The RTC-API wake-up does not cause wake-up from Standby mode in this scenario.

All of the aforementioned wake-up sources can be enabled or disabled. Also, you can configure these wake-up sources, by using WKPU configuration registers, to provide wake-up on rising or falling events. See the WKPU register memory map for details.

**NOTE**

You must use SIUL2 to perform WKPU pin configurations (PUE, PUS, and IBE). For this, you must first configure SIUL2.MSCR[IBE] for the corresponding pin. This chip does not support WKPU-provided pin configurations; it supports only bypass control.

### 45.1.2 WKPU NMI configuration

WKPU supports one external source that can cause non-maskable interrupts to on-chip cores and wake-up events to the system.

The following figure shows one application cores, Cortex-M7. In case of a wake-up event (internal or external), WKPU initiates the recovery of the chip and feeds an interrupt to the core depending on the configuration. The sections that follow provide details related to the associated configurations.

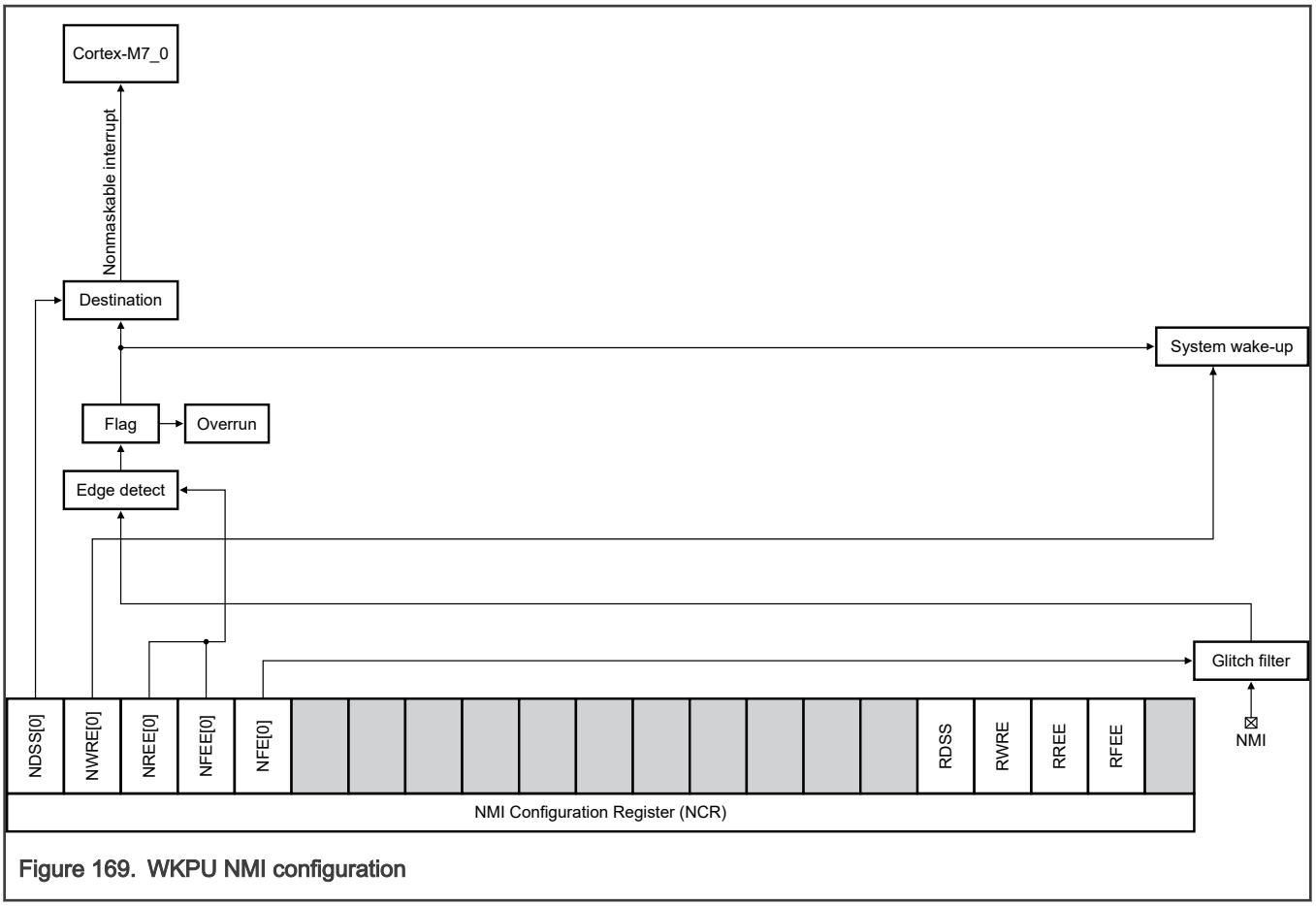



Figure 169. WKPU NMI configuration

### 45.1.3 WKPU wake-up source connectivity

WKPU allows the external NMI pin to assert the core NMIs on the chip. NMI supports NSR's status and overrun flags. This is what you could do using NCR:

- Control the NMI destination interrupt by configuring NCR[NDSS].
- Control the rising edge, falling edge, or either of the edge reactions to the NMI pin by using bits 2:0 of NCR[NFEE] and NCR[NREE]. The enabling of these edge reactions to the NMI pin is independent of each core.

WKPU supports the capturing of a second event per NMI input before the interrupt is cleared, thus reducing the chance of losing an NMI event.

The following figure shows routing of external wake-up events or interrupts with WKPU and the system interrupt controller.

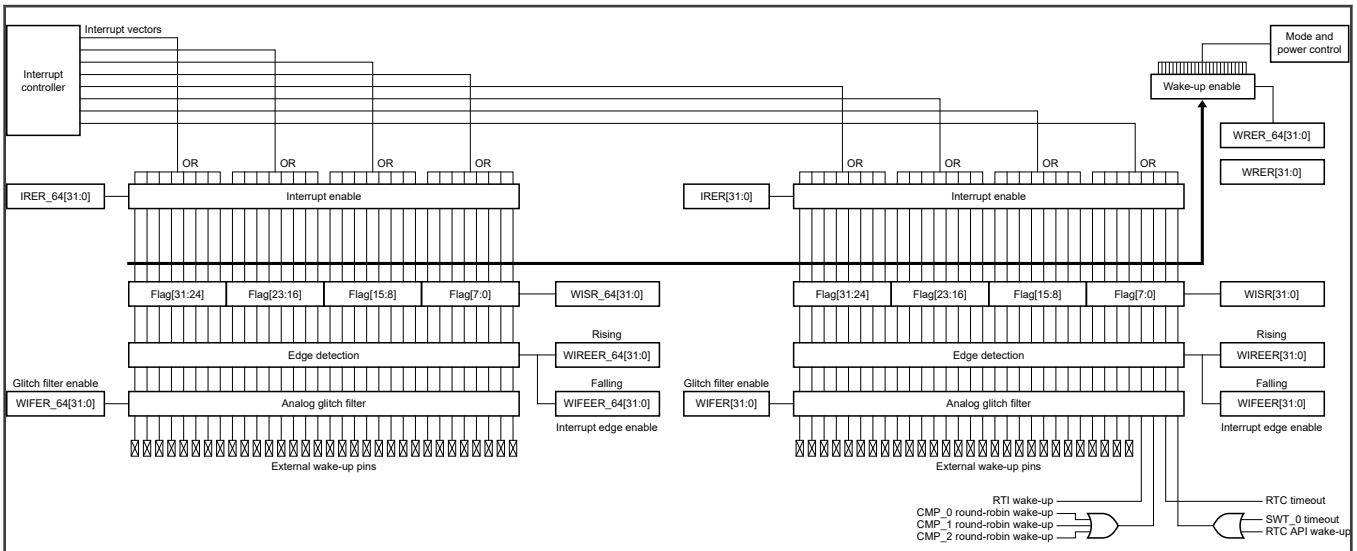



**Figure 170. WKPU wake-up source connectivity**

This is the wake-up source mapping to WKPU:

- Wake-up source 0 : SWT_0 timeout, RTC-API API wake-up
- Wake-up source 1 : RTC-API RTC timeout
- Wake-up source 2 : Round robin wake-up interrupt (Trigger mode interrupt) from LPCMP_0, LPCMP_1, or LPCMP_2
- Wake-up source 3 : RTI wake-up
- Wake-up source 4 : Wake-up source 63—external pin wake-up sources, WKPU[0] - WKPU[59]

If you configure any or all of the LPCMP_*x*.RRCR0[RR_EN] fields to be active, the corresponding CMP_*x* pins must be dedicated for the CMP_*x* operation. In case you are not using any of the CMPs, you can use SIUL2.MSCR*x*[SSS] to configure the pins for digital functionalities.

**NOTE**

You must enable WKPU (by using MC_ME.PRTN*x*_COFB*y*_CLKEN) before entering any of the chip's low-power modes.

## 45.2 Overview

WKPU supports 64 external sources that can generate interrupts or wake-up events, and 0 external sources that can cause nonmaskable interrupt requests or wake-up events. Additionally, it combines its wake-up events with those generated by other wake-up sources to supply a single wake-up to the system.

### 45.2.1 Block diagram

The following figure shows WKPU and its interfaces with the other system components.

**NOTE**

The signal widths in the following diagram do not depict a particular configuration of this chip. See the chip-specific WKPU information for details.

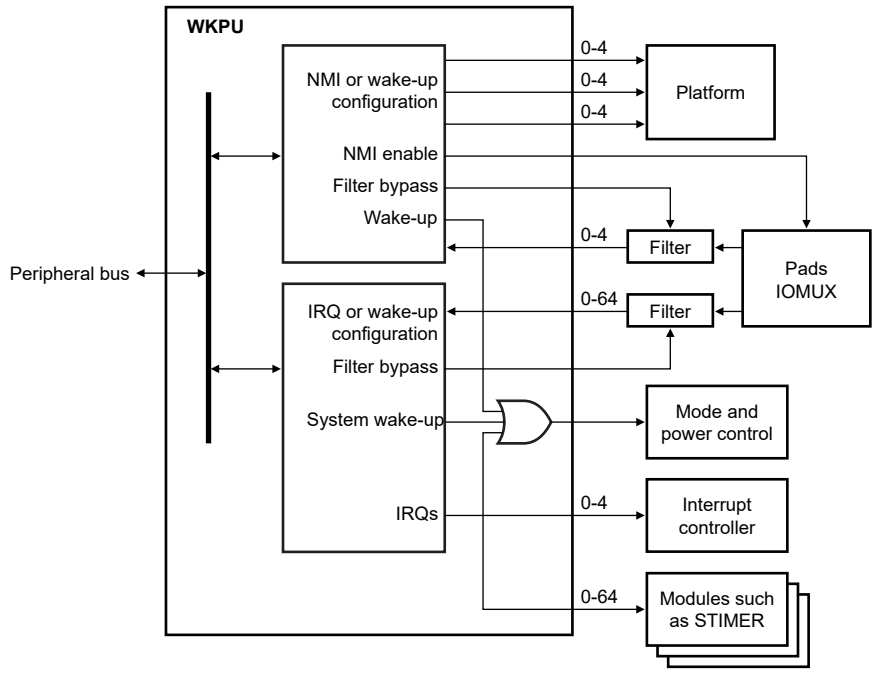



Figure 171. Block diagram

### 45.2.2 Features

- Supports nonmaskable interrupts including:
  - 2 NMI sources
  - 2 analog glitch filters
  - Independent interrupt destination for each core:
    - Nonmaskable interrupt
    - Critical interrupt
  - Active edge selection control (rise and fall) for events
  - Configurable system wake-up triggering from NMI sources
- Supports external wake-up and interrupts that include the following:
  - One System interrupt vector for interrupt sources
  - 64 analog glitch filters
  - Independent interrupt mask
  - Edge detection
  - Configurable system wake-up triggering from all interrupt sources

## 45.3 Functional description

### 45.3.1 Nonmaskable interrupts

WKPU supports the capturing of the second event according to the NMI input before the interrupt becomes 0. This reduces the chance of losing an NMI event, although it creates an overrun condition.

Each NMI passes through a bypassable analog glitch filter.

**NOTE**

When NMI is disabled, glitch filter control and pad configuration perform. This prevents erroneous triggering by glitches caused by the configuration process itself.

**NOTE**

The following figure represents a generic configuration and does not represent the configuration of this specific chip. See the chip-specific information for details on this chip's WKPU.

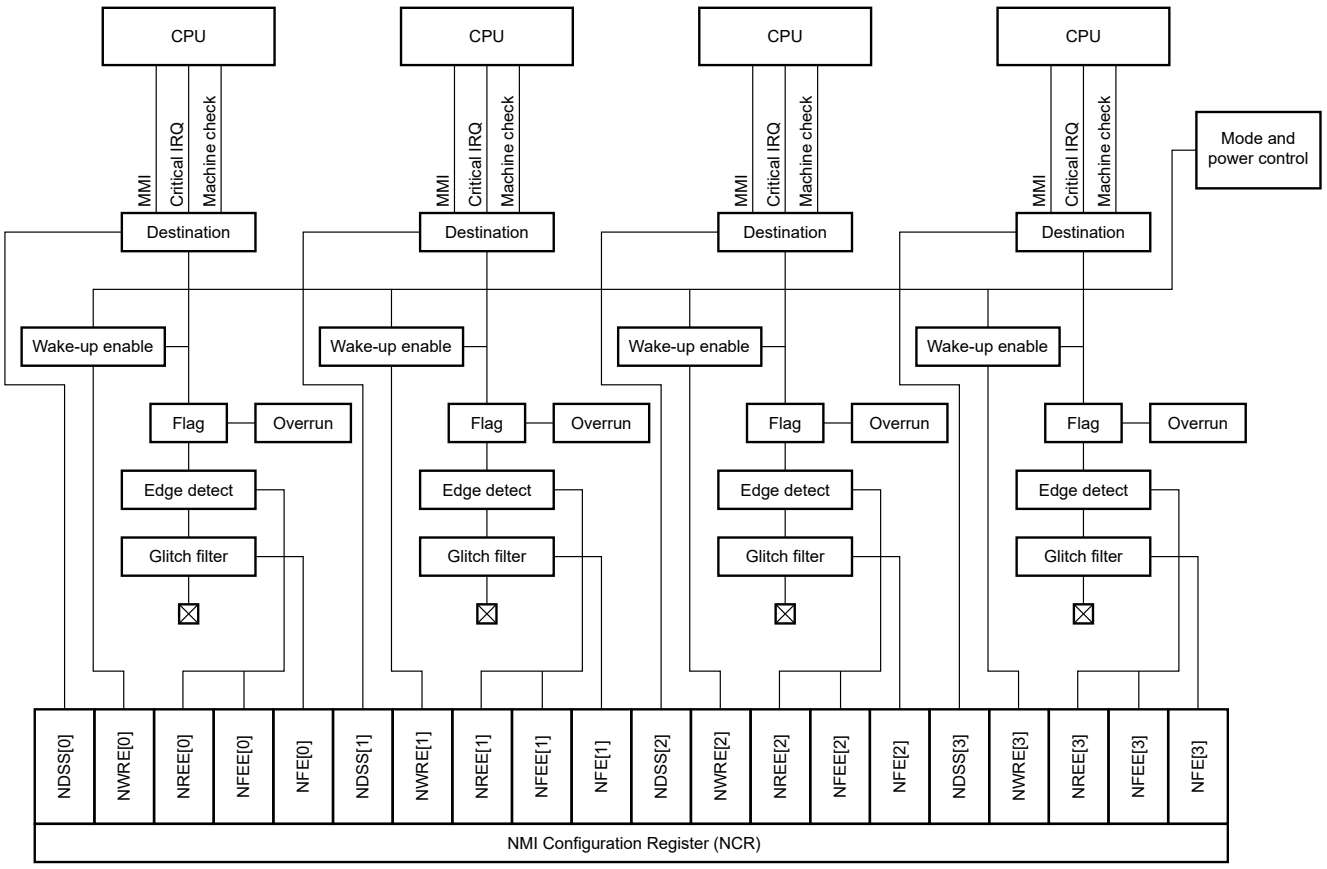



Figure 172. NMI pad diagram

### 45.3.1.1 NMI management

You can enable or disable each NMI independently. You can perform this by using the registers, NCR which are laid out to contain all configuration bits for a given NMI in a single byte (see NMI Configuration (NCR)). You can configure a pad defined as an NMI to recognize interrupts with an active rising edge, falling edge, or both edges being active. If both the edge events are disabled, it results in no interrupt being detected and does not get configured.

You can also control an active NMI edge through the configuration of the fields, NCR[NREE*n*] and NCR[NFEE*n*].

**NOTE**

After a reset, NREE and NFEE are set to 0, which disables the NMI functionality and software enables it explicitly.

After a pad's NMI functionality is enabled, the pad cannot be reconfigured to override or disable the NMI. See the chip-specific WKPU information for details of NMI implementation.

Additionally, you can control the NMI destination interrupt through the configuration of the field, NCR[NDSS*n*]. See NMI Configuration (NCR) for details.

Each NMI supports a status flag and an overrun flag, both of which are located in NMI Status Flag (NSR). This register prevents an inadvertent overwriting of the other flags in the same register. The status flag is set whenever an NMI event is detected. The overrun flag is set whenever an NMI event is detected when the status flag is still set (that is, the status flag has not yet been cleared).

**NOTE**

The overrun flag is cleared by writing 1 to the appropriate overrun field in NMI Status Flag (NSR). If the status field is 0 but the overrun field is still, the pending interrupt is not cleared.

During an NMI ISR, on wake-up of the chip from an NMI, any writes to the ECC-protected memory must have the correct ECC.

### 45.3.2 Clocking

This module has no clocking considerations.

WKPU has ipg_clk and ipg_clk_s as input clocks having the same clock frequency for register configuration and internal logic.

### 45.3.3 Interrupts

WKPU supports 4 interrupt vectors to the interrupt controller of the chip. Each interrupt vector supports multiple external interrupt sources from the device pads, with the total across all vectors being equal to the number of external interrupt sources. Each external interrupt source is assigned to exactly one interrupt vector. The interrupt vector assignment follows a sequence: one interrupt vector is for external interrupt sources 0 to N-1, the next is for N to N+M-1, and so on.

See the following figure for an overview of the external interrupt implementation, showing an example of four interrupt vectors with eight external interrupt sources each.

**NOTE**

The following figure represents a generic WKPU configuration. For details on this chip's configuration, see the chip-specific WKPU information.

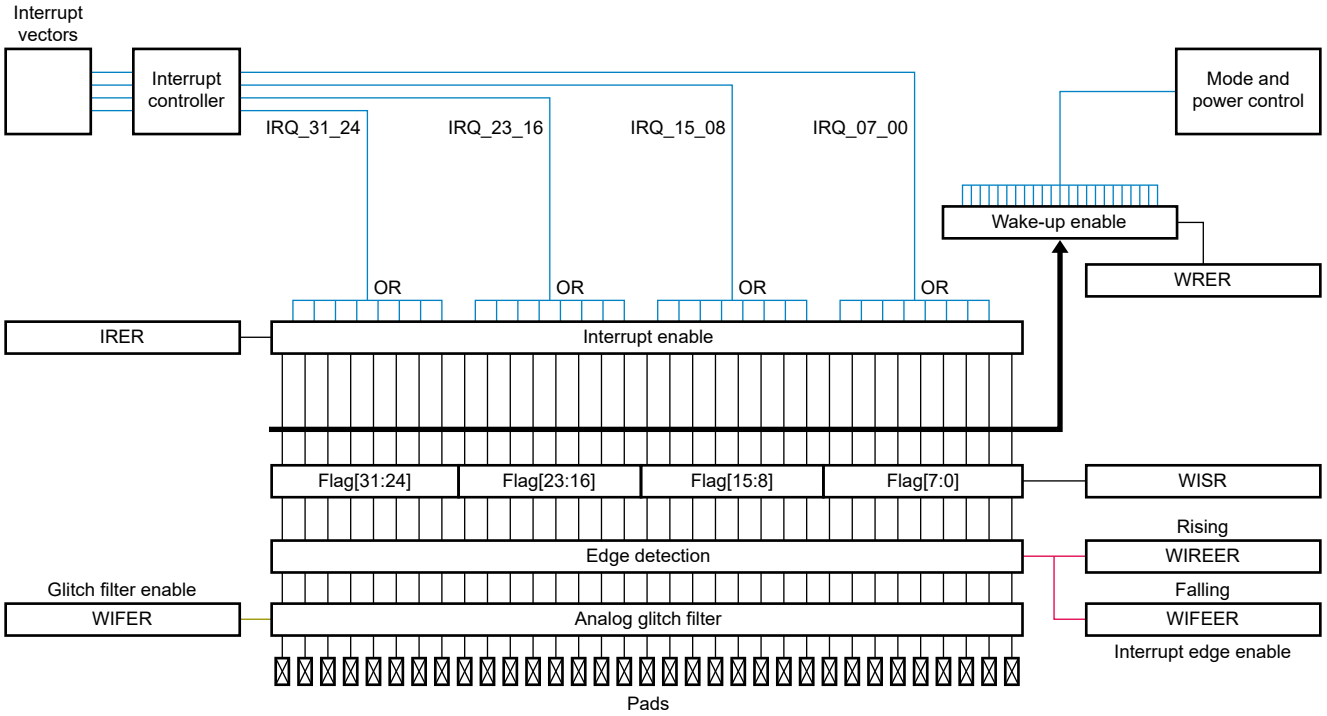



**Figure 173. External interrupt pad diagram**

All the external interrupt pads within a single group have equal priority. You must search through the group of sources in the most appropriate way for their application.

The priority of the vectors used by the external interrupt pads is set based on the platform and the priority levels of the interrupt controller. However, the chip can allow an independent configuration of pad allocation to each group of interrupts.

The external interrupt lines have a digital glitch filter applied to them.

#### 45.3.3.1 External interrupt management

You can enable or disable each external interrupt independently using a single rolled-up register (Interrupt Request Enable (IRER)). You can configure a pad defined as an external interrupt to recognize interrupts with an active rising edge, an active falling edge, or both edges being active.

**NOTE**

Writing 0 to both IREE[*n*] and IFEE[*n*] disables the external interrupt functionality for that pad completely (means no system wake-up or interrupt is generated from any activity on that pad).

You can control an active IRQ edge through the configuration of the registers, WIREER and WIFEER.

Each external interrupt supports an individual flag, which is held in the flag register, WISR. This W1C register prevents inadvertent overwriting of other flags in the same register.

### 45.3.4 External signals

This module has no external signals.

## 45.4 Initialization

To initialize this module, you must perform the following configuration:

- Glitch filter and pad configuration

- Nonmaskable interrupts
- Wake-up event

### 45.4.1 Glitch filter and pad configuration

You can perform glitch filter control and pad configuration when the NMI is disabled to avoid erroneous triggering by glitches caused by the configuration process.

When enabling the glitch filter, do not enable the rising and falling-edge events bits (that is, the NCR[NREE], NCR[NFEE], NCR[NREE], and NCR[NFEE]) in the same register write.

### 45.4.2 Nonmaskable interrupts

If IBE of NMI is tied off to 1, no false interrupt is expected.

### 45.4.3 Wake-up event

See the chip-specific WKPU information for wake-up event initialization.

## 45.5 WKPU memory map and registers

### 45.5.1 WKPU register descriptions

This section provides a detailed description of all the registers accessible in the WKPU module.

**NOTE**

Reserved registers read as zero and writes have no effect. A transfer error is generated when trying to access a completely reserved register space. The field length of external pad control registers depends on the number of WKPU channels implemented in a chip.

#### 45.5.1.1 WKPU memory map

WKPU base address: 402B_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | NMI Status Flag (NSR) | 32 | RW | 0000_0000h |
| 8h | NMI Configuration (NCR) | 32 | RW | 6060_0000h |
| 14h | Wake-Up and Interrupt Status Flag (WISR) | 32 | RW | 0000_0000h |
| 18h | Interrupt Request Enable (IRER) | 32 | RW | 0000_0000h |
| 1Ch | Wake-Up Request Enable (WRER) | 32 | RW | 0000_0000h |
| 28h | Wake-Up and Interrupt Rising-Edge Event Enable (WIREER) | 32 | RW | 0000_0000h |
| 2Ch | Wake-Up and Interrupt Falling-Edge Event Enable (WIFEER) | 32 | RW | 0000_0000h |
| 30h | Wake-Up and Interrupt Filter Enable (WIFER) | 32 | RW | 0000_0000h |
| 54h | Wake-Up and Interrupt Status Flag (WISR_64) | 32 | RW | 0000_0000h |
| 58h | Interrupt Request Enable (IRER_64) | 32 | RW | 0000_0000h |
| 5Ch | Wake-Up Request Enable (WRER_64) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 68h | Wake-Up and Interrupt Rising-Edge Event Enable (WIREER_64) | 32 | RW | 0000_0000h |
| 6Ch | Wake-Up and Interrupt Falling-Edge Event Enable (WIFEER_64) | 32 | RW | 0000_0000h |
| 70h | Wake-Up and Interrupt Filter Enable (WIFER_64) | 32 | RW | 0000_0000h |

### 45.5.1.2 NMI Status Flag (NSR)

#### Offset

| Register | Offset |
|---|---|
| NSR | 0h |

#### Function

Holds the nonmaskable interrupt status flags.

**NOTE**

This register is accessible by 8-, 16-, and 32-bit read/write operations.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | NIF0 | NOVF0 | Reserved | | | | | | NIF1 | NOVF1 | Reserved | | | | | |
| W | W1C | W1C | | | | | | | W1C | W1C | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

#### Fields

| Field | Function |
|---|---|
| 31<br>NIF0 | NMI Status Flag 0<br>Causes an interrupt request when an event, as defined by NREE0 and NFEE0, has occurred.<br>If NREE0 or NFEE0 is 1, this flag causes an interrupt request. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No event occurred on the pad<br>1b - An event occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 30<br>NOVF0 | NMI Overrun Status Flag 0<br>Indicates whether an overrun has occurred on NMI input 0.<br>This flag has the same current value as of NIF0 (when the NMI event occurs), indicating that the NMI occurred when the previous one was not serviced. If NREE0 or NFEE0 is 1, this flag causes an interrupt request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No overrun occurred<br>1b - An overrun occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 29-24<br>— | Reserved |
| 23<br>NIF1 | NMI Status Flag 1<br>Causes an interrupt request when an event, as defined by NREE1 and NFEE1, has occurred.<br>If NREE1 or NFEE1 is 1, this flag causes an interrupt request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No event occurred on the pad<br>1b - An event occurred<br>When writing |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No effect<br>1b - Clear the flag |
| 22<br>NOVF1 | NMI Overrun Status Flag 1<br>Indicates whether an overrun has occurred on NMI input 1.<br>This flag has the same current value as of NIF1 (when the NMI event occurs), indicating that the NMI occurred when the previous one was not serviced. If NREE1 or NFEE1 is 1, this flag causes an interrupt request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No overrun occurred<br>1b - An overrun occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 21-16<br>— | Reserved |
| 15-14<br>— | Reserved |
| 13-8<br>— | Reserved |
| 7-6<br>— | Reserved |
| 5-0<br>— | Reserved |

### 45.5.1.3 NMI Configuration (NCR)

**Offset**

| Register | Offset |
|---|---|
| NCR | 8h |

### Function

Holds the configuration fields for the nonmaskable interrupt settings.

**NOTE**

- This register is accessible by 8-, 16-, and 32-bit read/write operations.
- Writing 0 to both NREE[*n*] and NFEE[*n*] disables the NMI functionality completely (means no nonmaskable interrupt is generated on any pad activity).

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | NLOCK0 | NDSS0 | | NWRE0 | 0 | NREE0 | NFEE0 | NFE0 | NLOCK1 | NDSS1 | | NWRE1 | 0 | NREE1 | NFEE1 | NFE1 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31<br>NLOCK0 | NMI Configuration Lock Register 0<br>Specifies the lock configuration for the NMI.<br>When you write 1 to this field, it locks the configuration for the NMI until it is unlocked by a system reset or Standby0 mode exit.<br>0b - No effect<br>1b - Locks the configuration for the NMI |
| 30-29<br>NDSS0 | NMI Destination Source Select 0<br>Specifies the NMI destination interrupt to platform.<br>**NOTE**<br>As wake-up interrupt does not support another interrupt than NMI, the destination source select signal bits are reserved and always retain their reset value. This means no other request other than NMI can be generated.<br>00b - Nonmaskable interrupt<br>01b - Reserved<br>10b - Reserved<br>11b - Reserved. Reserved |
| 28 | NMI Wake-Up Request Enable 0 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| NWRE0 | Enables a system wake-up request when the corresponding NIF0 = 1 or NOVFO = 1.<br>0b - Disable<br>1b - Enable |
| 27<br>— | Reserved |
| 26<br>NREE0 | NMI Rising-Edge Events Enable 0<br>Enables the NMI rising-edge event.<br>0b - Disable<br>1b - Enable |
| 25<br>NFEE0 | NMI Falling-Edge Events Enable 0<br>Enables the NMI falling-edge event.<br>0b - Disable<br>1b - Enable |
| 24<br>NFE0 | NMI Filter Enable 0<br>Enables analog glitch filter on the NMI pad input.<br>0b - Disable<br>1b - Enable |
| 23<br>NLOCK1 | NMI Configuration Lock 1<br>Specifies the lock configuration for the NMI.<br>When you write 1 to this field, it locks the configuration for the NMI until it is unlocked by a system reset or Standby0 mode exit.<br>0b - No effect<br>1b - Locks the configuration for the NMI |
| 22-21<br>NDSS1 | NMI Destination Source Select 1<br>Specifies the NMI destination interrupt to platform.<br>**NOTE**<br>As wake-up does not support another interrupt than NMI, the destination source select signal bits are reserved and always retain their reset value. This means no other request other than NMI can be generated.<br>00b - Nonmaskable interrupt<br>01b - Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 10b - Reserved<br>11b - Reserved |
| 20<br>NWRE1 | NMI Wake-Up Request Enable 1<br>Enables a system wake-up request when the corresponding NIF1 = 1 or NOVF1 = 1.<br>0b - Disable<br>1b - Enable |
| 19<br>— | Reserved |
| 18<br>NREE1 | NMI Rising-Edge Events Enable 1<br>Enables the NMI rising-edge event.<br>0b - Disable<br>1b - Enable |
| 17<br>NFEE1 | NMI Falling-Edge Events Enable 1<br>Enables the NMI falling-edge event.<br>0b - Disable<br>1b - Enable |
| 16<br>NFE1 | NMI Filter Enable 1<br>Enables analog glitch filter on the NMI pad input.<br>0b - Disable<br>1b - Enable |
| 15-8<br>— | Reserved |
| 7-0<br>— | Reserved |

#### 45.5.1.4 Wake-Up and Interrupt Status Flag (WISR)

**Offset**

| Register | Offset |
|---|---|
| WISR | 14h |

### Function

Holds the wake-up and interrupt flags.

**NOTE**

- This register is accessible only by 32-bit read/write operations.
- The status fields associated with on-chip wake-up sources are located to the left of the external wake-up and interrupt status fields and are read-only. The wake-up for these sources must be configured and cleared at the on-chip wake-up source. Also, the configuration registers for the external interrupts/wake-ups do not have corresponding bits.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EIF | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EIF | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>EIF | External Wake-Up and Interrupt Status Flag<br>Specifies whether an event, as defined by WIREER and WIFEER, has occurred on the pad. If enabled (IRER[*n*]), EIF[*n*] causes an interrupt request.<br>0b - No event occurred<br>1b - An event occurred |

## 45.5.1.5 Interrupt Request Enable (IRER)

### Offset

| Register | Offset |
|---|---|
| IRER | 18h |

### Function

Enables the interrupt messaging from the wake-up and interrupt pads to the interrupt controller.

**NOTE**

This register is accessible only by 32-bit read/write operations.

If a pin is disabled through this register, you must write 0 to the corresponding fields in WIFEER and WIREER to ensure that the pin does not respond to any change.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EIRE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EIRE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>EIRE | External Interrupt Request Enable<br>Enables interrupt requests from the corresponding field, EIF[*n*].<br>0b - Disable<br>1b - Enable (with EIF[*n*] set) |

## 45.5.1.6 Wake-Up Request Enable (WRER)

### Offset

| Register | Offset |
|---|---|
| WRER | 1Ch |

### Function

Enables the system wake-up messaging from the wake-up and interrupt pads to the mode entry and power control modules.

**NOTE**

This register is accessible only by 32-bit read/write operations.

If a pin is disabled through this register, you must write 0 to the corresponding fields in WIFEER and WIREER to ensure that the pin does not respond to any change.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WRE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WRE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>WRE | External Wake-Up Request Enable<br>Enables system wake-up requests from the corresponding field, EIF[*n*].<br>0b - Disable<br>1b - Enable (with EIF[*n*] set) |

## 45.5.1.7 Wake-Up and Interrupt Rising-Edge Event Enable (WIREER)

### Offset

| Register | Offset |
|---|---|
| WIREER | 28h |

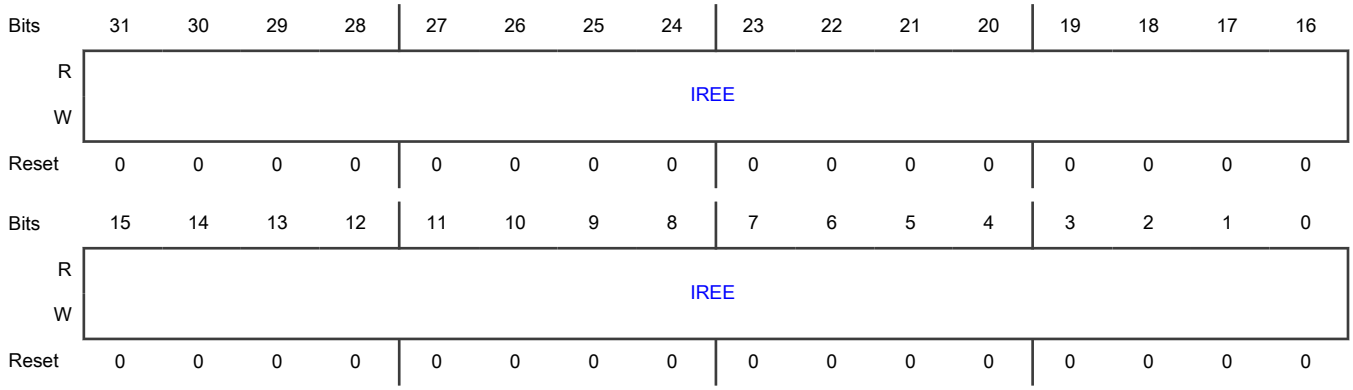
### Function

Enables rising-edge triggered events on the corresponding wake-up/interrupt pads.

**NOTE**

- This register is accessible only by 32-bit read/write operations.
- WIREER or WIFEER is configured for rising or falling edge after WRER to enable wake-up source.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>IREE | External Interrupt Rising-Edge Events Enable<br>Enables interrupt rising-edge event.<br>0b - Disable<br>1b - Enable |

## 45.5.1.8 Wake-Up and Interrupt Falling-Edge Event Enable (WIFEER)

### Offset

| Register | Offset |
|---|---|
| WIFEER | 2Ch |

### Function

Enables falling-edge triggered events on the corresponding wake-up and interrupt pads.

**NOTE**

- This register is accessible only by 32-bit read/write operations.
- WIREER or WIFEER is configured for rising or falling edge after WRER to enable wake-up source.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | IFEEx | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | IFEEx | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>IFEEx | External Interrupt Falling-Edge Events Enable<br>Enables interrupt falling-edge event.<br>0b - Disable<br>1b - Enable |

## 45.5.1.9 Wake-Up and Interrupt Filter Enable (WIFER)

### Offset

| Register | Offset |
|---|---|
| WIFER | 30h |

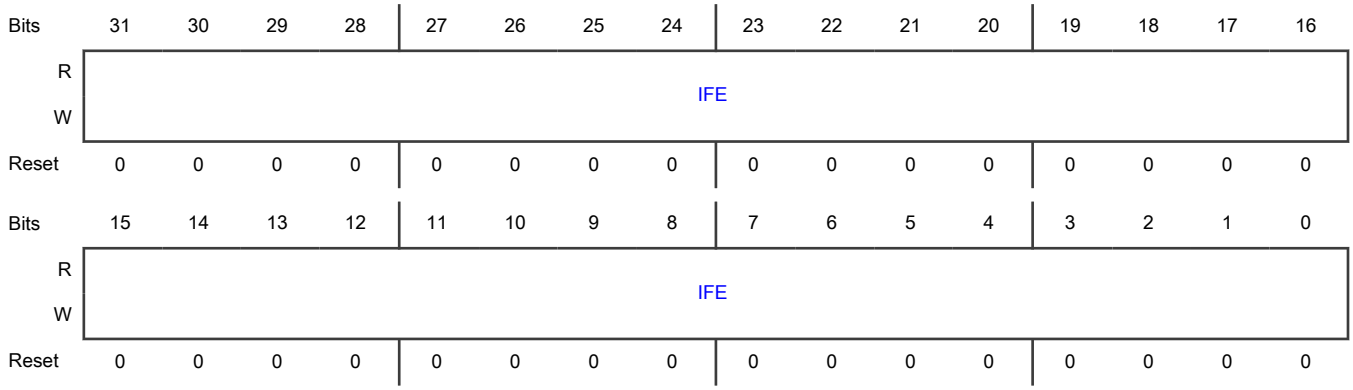
### Function

Enables an analog filter on the corresponding interrupt pads to filter out glitches on the inputs.

**NOTE**

This register is accessible only by 32-bit read/write operations.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>IFE | External Interrupt Filter Enable<br>Enables analog glitch filter on the external interrupt pad input.<br>0b - Disable<br>1b - Enable |

## 45.5.1.10 Wake-Up and Interrupt Status Flag (WISR_64)

### Offset

| Register | Offset |
|---|---|
| WISR_64 | 54h |

### Function

Holds the wake-up and interrupt flags.

**NOTE**

- This register is accessible only by 32-bit read/write operations.
- Status fields associated with on-chip wake-up sources are located to the left of the external wake-up/interrupt status fields and are read-only. The wake-up for these sources must be configured and cleared at the on-chip wake-up source. Also, the configuration registers for the external interrupts/wake-ups do not have corresponding fields.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EIF_1 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EIF_1 | | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>EIF_1 | External Wake-Up and Interrupt Status Flag<br>Specifies whether an event, as defined by WIREER and WIFEER, has occurred on the pad. If enabled (IRER[*n*]), EIF[*n*] causes an interrupt request.<br>0b - No event occurred<br>1b - An event occurred |

## 45.5.1.11 Interrupt Request Enable (IRER_64)

### Offset

| Register | Offset |
|---|---|
| IRER_64 | 58h |

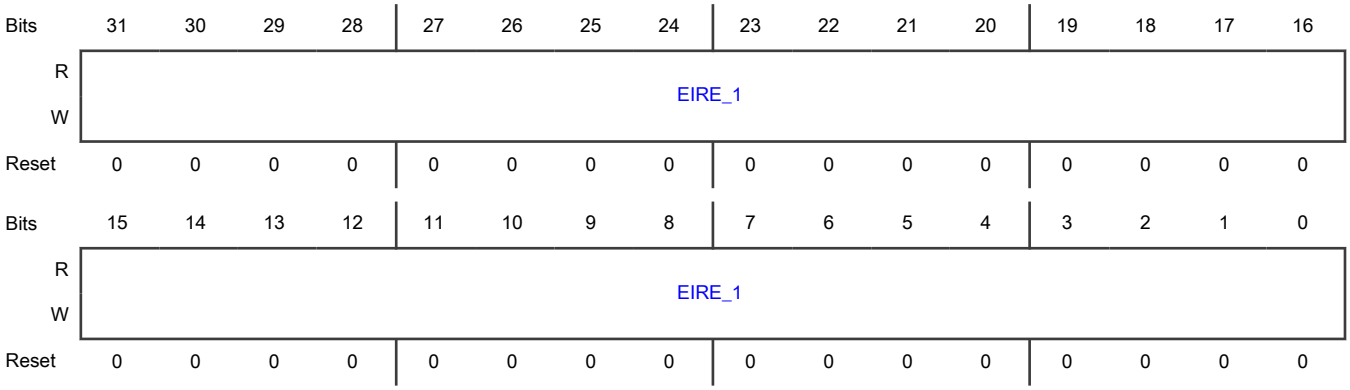
### Function

Enables interrupt messaging from the wake-up and interrupt pads to the interrupt controller.

**NOTE**

This register is accessible only by 32-bit read/write operations.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>EIRE_1 | External Interrupt Request Enable<br>Enables interrupt requests from the corresponding field, EIF[*n*].<br>0b - Disable<br>1b - Enable (with EIF[*n*] set) |

## 45.5.1.12 Wake-Up Request Enable (WRER_64)

### Offset

| Register | Offset |
|---|---|
| WRER_64 | 5Ch |

### Function

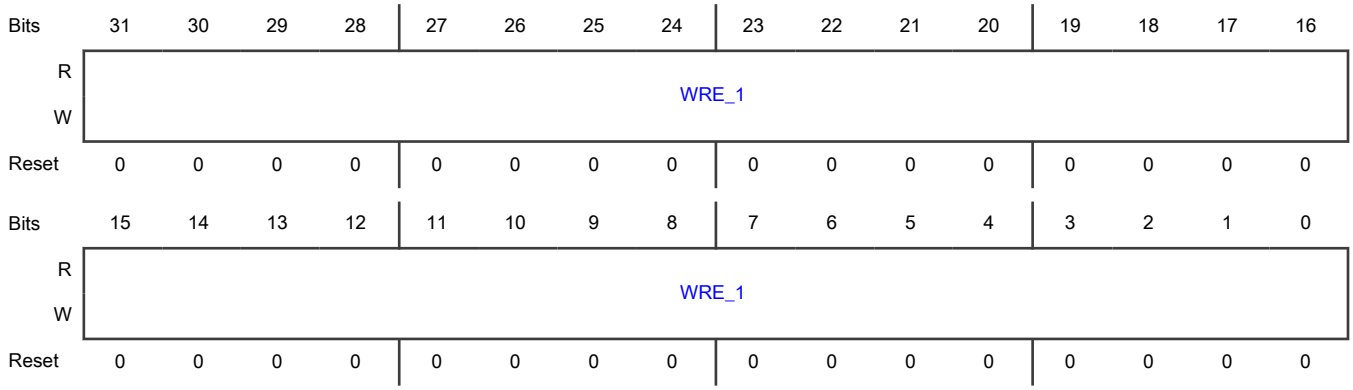
Enables system wake-up messaging from the wake-up and interrupt pads to the mode entry and power control modules.

**NOTE**

This register is accessible only by 32-bit read/write operations.

If a pin is disabled through this register, you must write 0 to the corresponding fields in WIFEER and WIREER to ensure that the pin does not respond to any change.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>WRE_1 | External Wake-Up Request Enable<br>Enables system wake-up requests from the corresponding field, EIF[*n*].<br>0b - Disable<br>1b - Enable (with EIF[*n*] set) |

## 45.5.1.13 Wake-Up and Interrupt Rising-Edge Event Enable (WIREER_64)

### Offset

| Register | Offset |
|---|---|
| WIREER_64 | 68h |

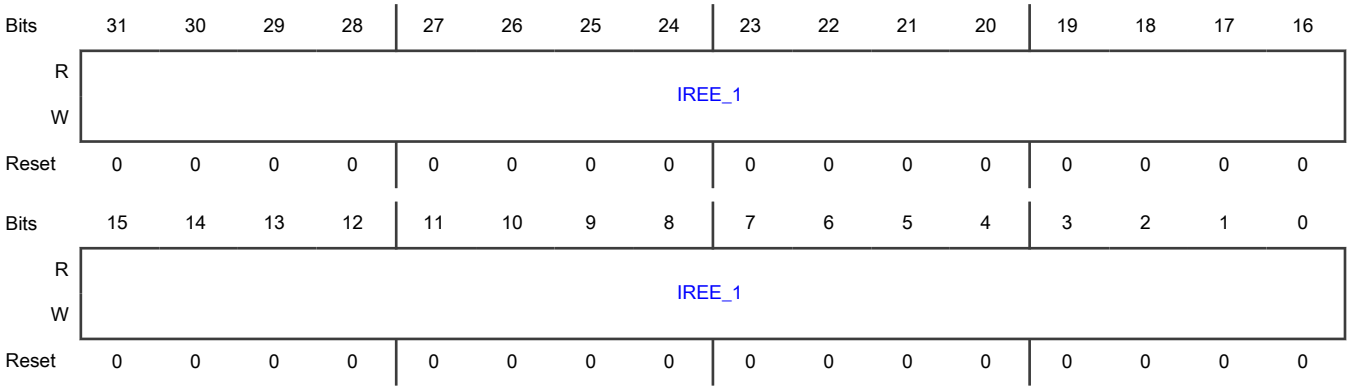
### Function

Enables rising-edge triggered events on the corresponding wake-up and interrupt pads.

**NOTE**

- This register is accessible only by 32-bit read/write operations.
- WIREER or WIFEER is configured for rising or falling-edge after WRER to enable wake-up source.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>IREE_1 | External Interrupt Rising-edge Events Enable<br>Enables an external interrupt rising-edge event.<br>0b - Disable<br>1b - Enable |

## 45.5.1.14 Wake-Up and Interrupt Falling-Edge Event Enable (WIFEER_64)

### Offset

| Register | Offset |
|---|---|
| WIFEER_64 | 6Ch |

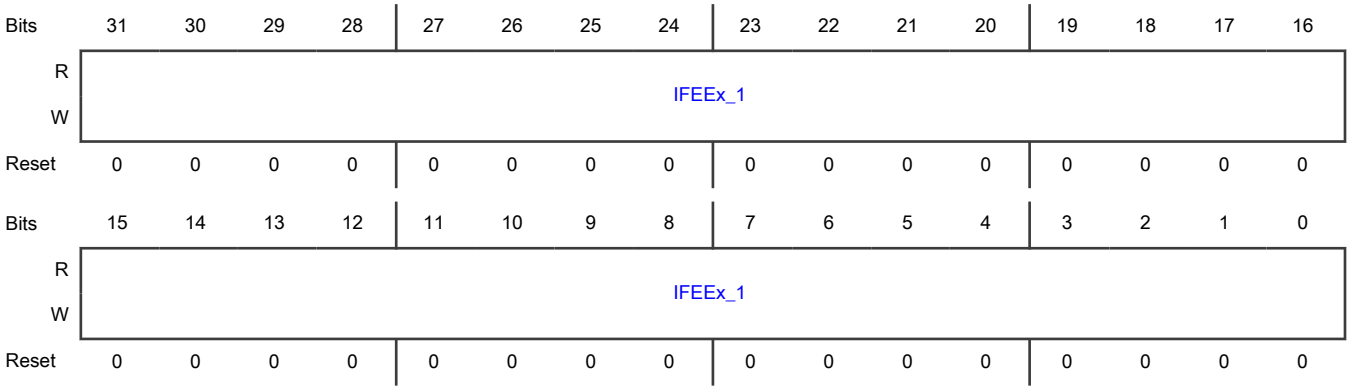
### Function

Enables falling-edge triggered events on the corresponding wake-up and interrupt pads.

**NOTE**

- This register is accessible only by 32-bit read/write operations.
- WIREER or WIFEER is configured for rising or falling-edge after WRER to enable a wake-up source.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>IFEEx_1 | External Interrupt Falling-Edge Events Enable<br>Enables an external interrupt falling-edge event.<br>0b - Disable<br>1b - Enable |

## 45.5.1.15 Wake-Up and Interrupt Filter Enable (WIFER_64)

### Offset

| Register | Offset |
|---|---|
| WIFER_64 | 70h |

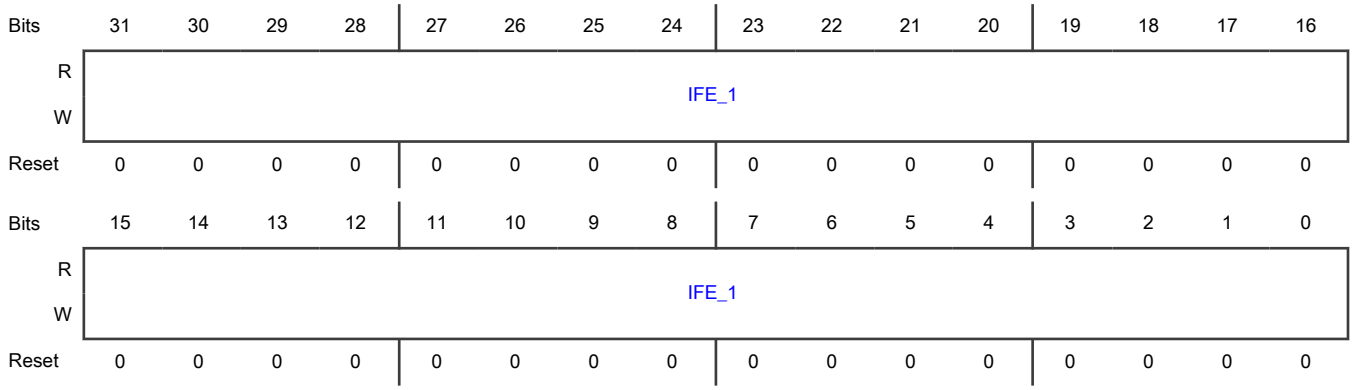
### Function

Enables an analog filter on the corresponding interrupt pads to filter out glitches on the inputs.

**NOTE**

This register is accessible only by 32-bit read/write operations.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>IFE_1 | External Interrupt Filter Enable<br>Enables an analog glitch filter on the external interrupt pad input.<br>0b - Disable<br>1b - Enable |

## 45.6 Glossary

**NMI** Nonmaskable interrupts

# Chapter 46
# Safety Overview

## 46.1 Introduction

This chip family is developed following the IEC 61508 standards, with derivatives targeting the chips that are operable in a system that fulfills the requirements of the SIL 2 safety integrity levels.

**Table 219. SIL levels**

| Chip | SIL level |
|---|---|
| MCXE315/MCXE316 | 2 |
| MCXE317 | |
| MCXE31B | |

The SIL level targets the safety processing, which means the software functions are executed as intended:

1. Read instructions from memory.
2. Execute instructions.
3. Read data from memory.
4. Process data.
5. Write back the result data into memory.

Some elements of the safety concept are based on the assumption that the chip is connected to an external SBC or PMIC. The SBC or PMIC performs observations and control functionalities that are essential to fulfill some related functional safety requirements. If you do not use an SBC or PMIC, you must ensure that your MCXE31 family chip provides an equivalent functionality that properly manages the corresponding interface(s).

The following figure illustrates the safety interface between the chip and the SBC or PMIC.

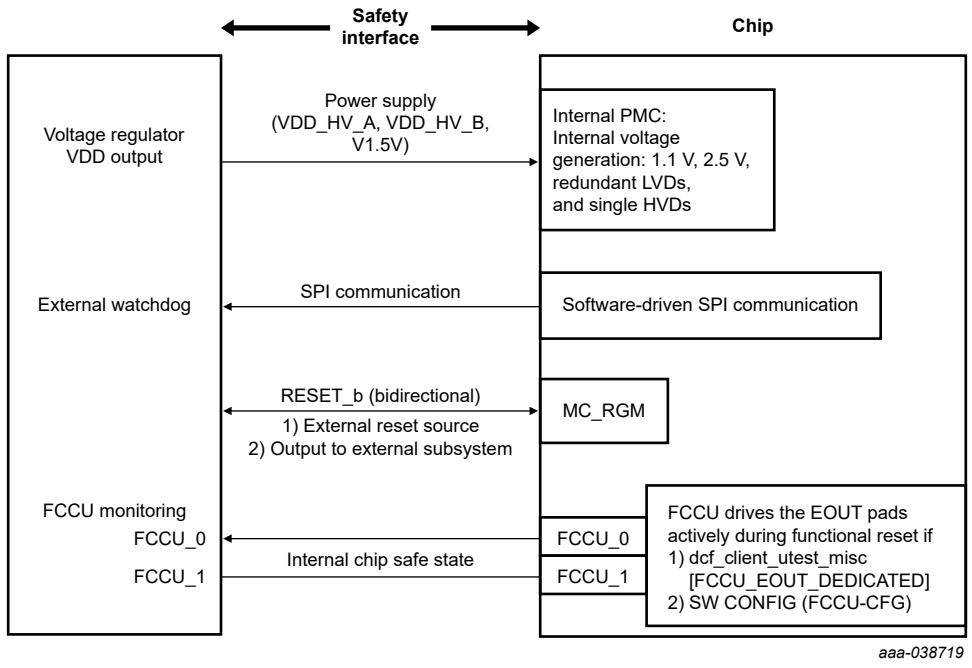



**Figure 174. Safety interface between the chip and the SBC or PMIC**

In the safety context, the chip interfaces can be classified into the groups shown in the table.

**Table 220. Chip interface groups**

| Interface | Description |
|---|---|
| Power supply | This interface is between the SBC and the chip. It ensures that the supply to the chip is in the correct range. In case of any low-voltage event, the chip has POR, LVR, and LVD circuits in place and in case of any HVD event, the chip raises an interrupt. The SBC must ensure that the voltage regulator outputs never exceed the allowed range (more specifically for HVD range). |
| Communication | Responsible for communication between the SBC and the chip. Though the chip supports multiple communication protocols (UART, LIN, SPI, I2C, FlexCAN, and so on), SPI is the preferred communication with SBC. This is relevant to initialize an external watchdog when the chip is inoperative for a considerably long time (this is indicated by the pin states of the communication interface). |
| Reset | The chip consists of a reset bidirectional pin interfaced with SBC. The SBC can initiate a chip reset via this pin as a safety reaction in case of:<br>• Extreme critical faults<br>• An inoperative chip<br>• Stuck cases based on the criticality and the application requirements<br>The SBC also samples the reset pin state to identify the chip condition (whether in running state or in reset). |
| FCCU | The chip indicates the chip faults to the SBC via FCCU EOUT pins through this interface. |

The interfaces in Table 220 ensure the chip's operational safety and integrity.

## 46.2 Safety architecture elements

The chip safety architecture consists of following elements that operate in an interconnected way to meet the SIL requirements:

- Cortex-M7 core complex
- eDMA controller
- Structural core self-test
- Internal windowed watchdogs with independent clock sources
- Power supply monitoring with redundant low-voltage detectors, single high-voltage detectors, and internal ADC connection to check the internal voltages during application
- Robust clock monitoring, including PLL loss of lock detection
- Embedded flash memory with ECC (single-bit data correction, double-bit data detection) and address encoding (parallel address path check)
- System RAM with ECC
  - Single-bit data correction

  - Double-bit data detection
  - Address detection
- Cortex-M7 cache memories with ECC (single-bit data correction, double-bit data detection)
- Peripherals memories (EMAC, FlexCAN) with ECC (single-bit data correction, double-bit data detection)
- End-to-End EDC (E2E EDC), with address encoding and monitoring of the control signals done by a dedicated module (XBIC). This ensures the safety of storage and of the data path to the internal storage (RAM, flash memory, core cache) and peripherals across the crossbar switch. See the XBIC chapter for details.
- Hardware CRC module that supports end-to-end data check integrity for any data transfer in the system (SRAM to peripherals via DMA transfer, external interfaces to SRAM/peripherals, and so on)
- Cortex-M7 MPUs
- XRDC for memory and peripheral protection
- AIPS_Lite peripheral protection with trusted master-slave connection
- Register protection mechanism for safety-critical registers
- On-chip temperature sensor for temperature monitoring
- Self-test:
  - LBIST to detect hidden dangerous faults in functional logic as well as in safety integrity mechanisms

**NOTE**

LBIST is not supported in MCXE315/MCXE316 and MCXE317.

  - MBIST to ensure integrity of the memories in the chip (SRAM, ITCM, DTCM, peripheral memories, and so on)
  - Check-the-checker software library
  - ADC self-test
- FCCU for error collection and reaction, including reporting error status to system; FCCU supports these programmable reaction types:
  - Interrupt
  - Functional reset
- Error pads indicate the chip's internal state to the external chip interface or SBC.
- EIM to inject errors into the memories and interface gaskets to verify the error-detection features of the memory controllers and the interface gaskets
- ERM to collect diagnostic information from memory controllers in case of an error event

## 46.3 I/O peripherals

The arrangement of I/O peripherals across peripheral bridges allows redundant use of peripherals while limiting possible causes of CCF. Redundant use includes using equivalent peripherals in a replicated way as well as using functionally different peripherals in, for example, feedback measurement loops. Comparison of redundant operation is the responsibility of the application software, not the safety hardware mechanism.

The peripherals are distributed evenly across the peripheral bridges (AIPS*n*), except for singular modules like EMAC, QuadSPI, and so on.

**NOTE**

EMAC and QuadSPI are not present in MCXE315/MCXE316 and MCXE317.

## 46.4 Self-test

The chip supports a self-test operation. Your safety software must initiate self-test by configuring STCU2; the chip does not initiate it. STCU2 then controls the self-test operation. Self-test supports both MBIST and LBIST. After the self-test operation completes, the chip enters a reset sequence. Self-test results are stored in STCU2 and your safety software can read the results after the reset sequence.

Figure 175 depicts the processing steps related to the self-test operation, and its linkage to the chip reset and an application start. You decide whether to run a self-test or to skip it before starting an application. You must also specify the self-test configuration before relinquishing control to STCU2 for performing the self-test operation. This processing finishes by reentering the chip reset sequence, depending on whether the chip encountered an unrecoverable fault during the self-test operation. When an unrecoverable fault is encountered during the self-test operation, the chip enters a reset sequence by performing a destructive reset. When no such fault is encountered, the self-test operation completes by entering the chip sequence with a functional reset sequence. You can prevent reset cycling by limiting the amount of resets permitted; the chip shuts down when this limit has been reached. See the "Functional reset escalation" and "Destructive reset escalation" sections in the MC_RGM chapter.

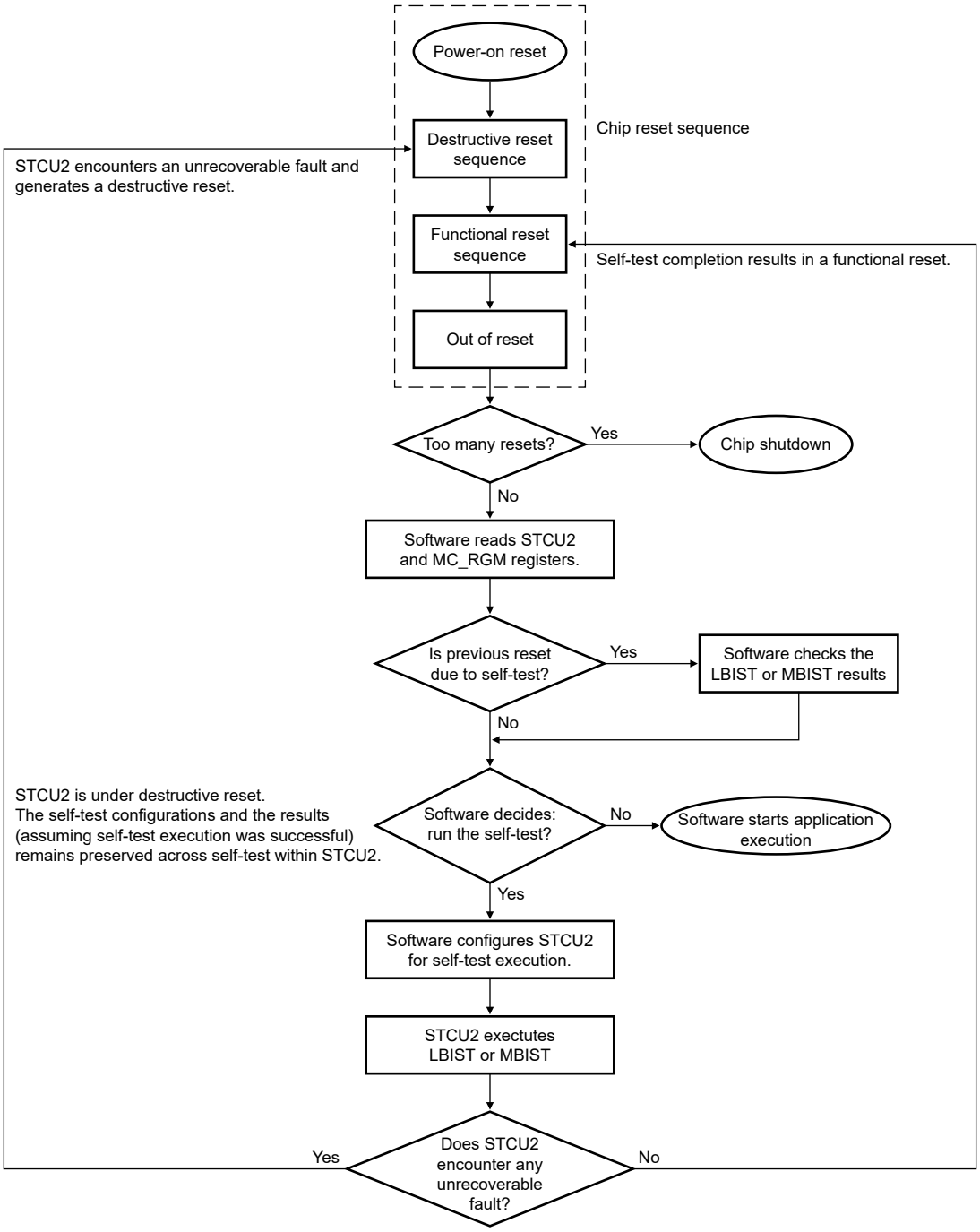



**Figure 175. LBIST/MBIST execution**

Figure 176 visualizes the top-level partitioning of the chip. The chip consists of two partitions, Run and Standby.

- Run: This partition consists of logic which is present in switchable domain and is shut off (has no supply) while the chip operates in low-power (Standby) mode.
- Standby: This partition consists of always-on logic which is functional even while the chip operates In low-power (Standby) mode.

The Run partition also contains the control logic that is essential for the chip self-test operation, as well as blocks that undergo self-test. The chip consists of a single LBIST partition, which is a subpartition within the Run partition. The LBIST subpartition

contains the logic over which self-test is executed. The logic outside the LBIST subpartition and within the Run partition consists of the LBIST control logic as well as logic which does not undergo LBIST.

The components within these different partitions are listed below the figure (they are not indicated in the figure for clarity).

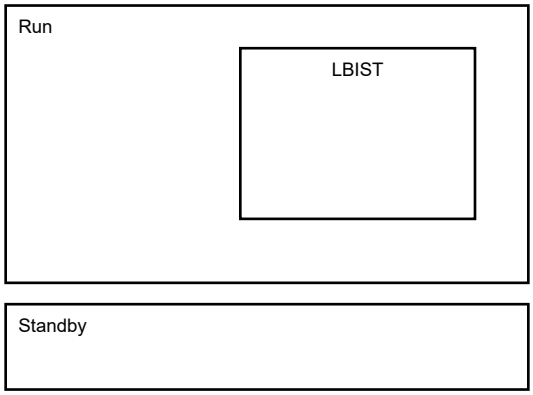



**Figure 176. MCXE31B chip top-level partition view**

The following list specifies the content of the LBIST partition and both power domains related to their participation in the self-test operation:

- Modules participating in the LBIST operation:

  **NOTE**

  LBIST is not supported in MCXE315/MCXE316 and MCXE317. The modules participating in the LBIST operation are out of LBIST in MCXE315/MCXE316 and MCXE317.

  — eDMA
  — EIM
  — ERM
  — XRDC
    - MDAC0 (Cortex-M7_0)
    - MDAC1 (eDMA)
    - MDAC2 (reserved)
    - MDAC3 (ELE_HSEB)
    - MDAC4 (Cortex-M7_1)
    - MDAC5 (EMAC)
    - MRC0 (flash memory)
    - MRC1 (PRAM)
    - MRC2 (QuadSPI)

      **NOTE**

      Cortex-M7, EMAC, and QuadSPI are not present in MCXE315/MCXE316 and MCXE317.

    - PDAC1
    - PDAC2
  — CRC
  — FCCU

- FOSU
- PFLASH_CTL
- PRAM_C0
- PRAM_C1
- SWT_1
- CMU
  - CMU_FC_0
  - CMU_FM_1
  - CMU_FM_2
  - CMU_FC_3
  - CMU_FC_4
  - CMU_FC_5
- IAHB gaskets

**NOTE**

An IAHB gasket is a frequency translation gasket in the system. These gaskets are used to add register walls and synchronize the timing domain across different operating frequencies. See the configurable gaskets in the 'Block diagram' in 'Introduction' chapter for the locations of the IAHB_GASKETs in the chip.

  - DMA_GSKT
  - HSE_GSKT
  - AIPS1_GSKT
  - AIPS2_GSKT
  - QSPI_GSKT

**NOTE**

AIPS2_GSKT and QSPI_GSKT are not present in MCXE315/MCXE316 and MCXE317.

  - TCM_GSKT
  - EMAC_GSKT
  - EDC master checker gaskets (See block diagram in Introduction chapter for details)
  - EDC slave checker gasket (See block diagram in Introduction chapter for details)
- Crossbar switches and bridges
  - AXBS_0 (main)
  - AXBS_1 (peripheral)
  - AXBS_2 (eDMA)
  - AXBS_3 (Cortex-M7 TCM)
  - AXBS_4 (HSE)
  - AIPS_Lite_1
  - AIPS_Lite_2

**NOTE**

AIPS2_Lite_ is not present in MCXE315/MCXE316 and MCXE317.

- Modules within the Standby domain (not participating in the LBIST operation):
  - 32 KB standby SRAM
  - LPCMP_0–LPCMP_2
  - Reset generation
    - MC_RGM
    - POR_WDG
  - Power management
    - PMC
    - MC_PCU
    - WKPU
    - Chip standby pins
  - Clock sources
    - FIRC
    - SIRC
    - FXOSC
    - SXOSC

**NOTE**

SXOSC is not present in MCXE315/MCXE316.

  - Timers
    - PIT_0
    - RTC
    - SWT_0
- Modules not participating in the LBIST operation (in the Run domain):
  - HSE subsystem
    - Engine
    - Memories
  - Cortex-M7_0
    - Core
    - I-cache
    - D-cache
    - ITCM
    - DTCM
  - Cortex-M7_1

**NOTE**

Cortex-M7_1 is not present in MCXE315/MCXE316 and MCXE317.

    - Core
    - I-cache

- D-cache
- ITCM
- DTCM

— CMUs
- CMU0
- CMU3

— Analog blocks
- ADC_0–ADC_2
- TempSense

**NOTE**

ADC_2 is not present in MCXE315/MCXE316 and MCXE317.

— Communication modules
- LPSPI_0–LPSPI_5
- LPI2C_0
- LPI2C_1
- LPUART_0–LPUART_15
- FlexCAN_0–FlexCAN_5
- SAI_0
- SAI_1

**NOTE**

SAI_0 and SAI_1 are not present in MCXE315/MCXE316 and MCXE317.

— Timers
- PIT_1
- PIT_2
- eMIOS
- BCTU
- STM_0
- STM_1

**NOTE**

PIT_2 and STM_1 are not present in MCXE315/MCXE316 and MCXE317.

— TRGMUX

— PLLDIG

— SIUL2

— LCU_0

— LCU_1

— STCU2

— DMAMUX

— MCM

— MCSM

— INTM

— SEMA42

— MU

**NOTE**

SEMA42 and chip MU are not present in MCXE315/MCXE316 and MCXE317.

— TSPC

**NOTE**

REG_PROT of an IP undergoes (or does not undergo) LBIST in conjunction with the protected module that undergoes (or does not undergo) LBIST.

The modules that are vital to the self-test operation are excluded from LBIST regions to allow LBIST to execute successfully.

You must run self-test with PLLDIG configured as the system clock. The LBIST clock controller controls the clock during serial shift, but returns clock control to the functional nodes during the self-test.

## 46.5 Glossary

| | |
|---|---|
| **SIL** | SIL / Safety Integrity Level. This is a risk classification scheme as defined by IEC 61508 functional |
| **CCF** | Common cause failure |
| **DTCM** | Data tightly coupled memory |
| **ECC** | Error correction code |
| **ITCM** | Instruction tightly coupled memory |
| **LBIST** | Logic built-in self-test |
| **MBIST** | Memory built-in self-test |
| **PLL** | Phase-locked loop oscillator |
| **PMIC** | Power management integrated chip |
| **SBC** | System basis chip |

# Chapter 47
# Error Injection Module (EIM)

## 47.1 Chip-specific EIM information

### 47.1.1 EIM channel mapping

EIM integrates with the memory controller and memory array to enable error injection in a controlled way. Each memory controller has its own EIM channel.

EMAC, AIPS2 gasket, Cortex-M7, and QuadSPI gasket are not available in the MCXE317 and MCXE315/MCXE316 product variants of the MCXE31 family.

**Table 221. EIM channel mapping**

| Channel # | Target | Data bits [1] | Check bits[1] | # of data bits | # of check bits |
|---|---|---|---|---|---|
| 0 | SRAM0 | Word1[31:0] – SRAM0 read data[63:32] | Word0[31:24] – SRAM0 read data ECC[7:0] | 64 | 8 |
| | | Word2[31:0] – SRAM0 read data[31:0] | | | |
| 1 | SRAM1[2] | Word1[31:0] – SRAM1 read data[63:32] | Word0[31:24] SRAM1 read data ECC[7:0] | 64 | 8 |
| | | Word2[31:0] – SRAM1 read data[31:0] | | | |
| 2 | DMA TCD | Word1[31:0] – DMA TCD RAM read data[63:32] | Word0[31:24] DMA TCD RAM read data checkbits[7:0] | 64 | 8 |
| | | Word2[31:0] – DMA TCD RAM read data[31:0] | | | |
| 3 | Cortex-M7_0 IC tag | Word1[12:0] – Cortex-M7_0 IC tag read data1[28:16] | Word0[31:25] – Cortex-M7_0 IC tag read data1[6:0] | 44 | 14 |
| | | Word2[31:22] – Cortex-M7_0 IC tag read data1[15:7] | | | |
| | | Word2[21:0] – Cortex-M7_0 IC tag read data0[28:7] | Word0[24:18] – Cortex-M7_0 IC tag read data1[6:0] | | |
| 4 | Cortex-M7_0 IC data | Word1[31:0] – Cortex-M7_0 IC data read data1[71:40] | Word0[31:24] – Cortex-M7_0 IC data data1[7:0] | 128 | 16 |
| | | Word2[31:0] – Cortex-M7_0 IC data read data1[39:8] | | | |
| | | Word3[31:0] – Cortex-M7_0 IC data read data0[71:40] | Word0[23:16] – Cortex-M7_0 IC data read data0[7:0] | | |
| | | Word4[31:0] – Cortex-M7_0 IC data read data0[39:8] | | | |

*Table continues on the next page...*

Table 221. EIM channel mapping (continued)

| Channel # | Target | Data bits [1] | Check bits[1] | # of data bits | # of check bits |
|---|---|---|---|---|---|
| 5 | Cortex-M7_0 DC tag | Word1[7:0] – Cortex-M7_0 DC tag read data3[32:25] | Word0[31:25] – Cortex-M7_0 DC tag read data3[6:0] | 104 | 28 |
| | | Word2[31:14] – Cortex-M7_0 DC tag read data3[24:7] | | | |
| | | Word2[13:0] – Cortex-M7_0 DC tag read data2[32:19] | Word0[24:18] – Cortex-M7_0 DC tag read data2[6:0] | | |
| | | Word3[31:20] – Cortex-M7_0 DC tag read data2[18:7] | | | |
| | | Word3[19:0] – Cortex-M7_0 DC tag read data1[32:13] | Word0[17:11] – Cortex-M7_0 DC tag read data1[6:0] | | |
| | | Word4[31:26] – Cortex-M7_0 DC tag read data1[12:7] | | | |
| | | Word4[25:0] – Cortex-M7_0 DC tag read data0[32:7] | Word0[10:4] – Cortex-M7_0 DC tag read data0[6:0] | | |
| 6 | Cortex-M7_0 DC data0 | Word1[31:0] – Cortex-M7_0 DC data0 read data3[38:7] | Word0[31:25] – Cortex-M7_0 DC data0 read data3[6:0] | 128 | 28 |
| | | Word2[31:0] – Cortex-M7_0 DC data0 read data2[38:7] | Word0[24:18] – Cortex-M7_0 DC data0 read data2[6:0] | | |
| | | Word3[31:0] – Cortex-M7_0 DC data0 read data1[38:7] | Word0[17:11] – Cortex-M7_0 DC data0 read data1[6:0] | | |
| | | Word4[31:0] – Cortex-M7_0 DC data0 read data0[38:7] | Word0[10:4] – Cortex-M7_0 DC data0 read data0[6:0] | | |
| 7 | Cortex-M7_0 DC data1 | Word1[31:0] – Cortex-M7_0 DC data1 read data3[38:7] | Word0[31:25] – Cortex-M7_0 DC data1 read data3[6:0] | 128 | 28 |
| | | Word2[31:0] – Cortex-M7_0 DC data1 read data2[38:7] | Word0[24:18] – Cortex-M7_0 DC data1 read data2[6:0] | | |
| | | Word3[31:0] – Cortex-M7_0 DC data1 read data1[38:7] | Word0[17:11] – Cortex-M7_0 DC data1 read data1[6:0] | | |
| | | Word4[31:0] – Cortex-M7_0 DC data1 read data0[38:7] | Word0[10:4] – Cortex-M7_0 DC data1 read data0[6:0] | | |
| 8 | Cortex-M7_1 IC tag | Word1[12:0] – Cortex-M7_1 IC tag read data1[28:16] | Word0[31:25] – Cortex-M7_1 IC tag read data1[6:0] | 44 | 14 |

*Table continues on the next page...*

Table 221. EIM channel mapping (continued)

| Channel # | Target | Data bits [1] | Check bits[1] | # of data bits | # of check bits |
|---|---|---|---|---|---|
| | | Word2[31:22] – Cortex-M7_1 IC tag read data1[15:7] | | | |
| | | Word2[21:0] – Cortex-M7_1 IC tag read data0[28:7] | Word0[24:18] – Cortex-M7_1 IC tag read data0[6:0] | | |
| 9 | Cortex-M7_1 IC tag | Word1[31:0] – Cortex-M7_1 IC data read data0[71:40] | Word0[31:24] – Cortex-M7_1 IC data read data1[7:0] | 128 | 16 |
| | | Word2[31:0] – Cortex-M7_1 IC data read data0[39:8] | | | |
| | | Word3[31:0] – Cortex-M7_1 IC data read data0[71:40] | Word0[23:16] – Cortex-M7_1 IC data read data0[7:0] | | |
| | | Word4[31:0] – Cortex-M7_1 IC data read data0[39:8] | | | |
| 10 | Cortex-M7_1 DC tag | Word1[7:0] – Cortex-M7_1 DC tag read data3[32:25] | Word0[31:25] – Cortex-M7_1 DC tag read data3[6:0] | 104 | 28 |
| | | Word2[31:14] – Cortex-M7_1 DC tag read data3[24:7] | | | |
| | | Word2[31:0] – Cortex-M7_1 DC tag read data2[32:19] | Word0[24:18] – Cortex-M7_1 DC tag read data2[6:0] | | |
| | | Word3[31:20] – Cortex-M7_1 DC tag read data1[18:7] | | | |
| | | Word3[19:0] – Cortex-M7_1 DC tag read data1[32:13] | Word0[17:11] – Cortex-M7_1 DC tag read data1[6:0] | | |
| | | Word4[31:26] – Cortex-M7_1 DC tag read data1[12:7] | | | |
| | | Word4[25:0] – Cortex-M7_1 DC tag read data0[32:7] | Word0[10:4] – Cortex-M7_1 DC tag read data0[6:0] | | |
| 11 | Cortex-M7_1 DC data0 | Word1[31:0] – Cortex-M7_1 DC data0 read data3[38:7] | Word0[31:25] – Cortex-M7_1 DC data0 read data3[6:0] | 128 | 28 |
| | | Word2[31:0] – Cortex-M7_1 DC data0 read data2[38:7] | Word0[24:18] – Cortex-M7_1 DC data0 read data2[6:0] | | |
| | | Word3[31:0] – Cortex-M7_1 DC data0 read data1[38:7] | Word0[17:11] – Cortex-M7_1 DC data0 read data1[6:0] | | |
| | | Word4[31:0] – Cortex-M7_1 DC data0 read data0[38:7] | Word0[10:4] – Cortex-M7_1 DC data0 read data0[6:0] | | |

*Table continues on the next page...*

Table 221. EIM channel mapping (continued)

| Channel # | Target | Data bits [1] | Check bits[1] | # of data bits | # of check bits |
|---|---|---|---|---|---|
| 12 | Cortex-M7_1 DC data1 | Word1[31:0] – Cortex-M7_1 DC data1 read data0[38:7] | Word0[31:25] – Cortex-M7_1 DC data1 read data3[6:0] | 128 | 28 |
| | | Word2[31:0] – Cortex-M7_1 DC data1 read data0[38:7] | Word0[24:18] – Cortex-M7_1 DC data1 read data2[6:0] | | |
| | | Word3[31:0] – Cortex-M7_1 DC data1 read data0[38:7] | Word0[17:11] – Cortex-M7_1 DC data1 read data1[6:0] | | |
| | | Word4[31:0] – Cortex-M7_1 DC data1 read data0[38:7] | Word0[10:4] – Cortex-M7_1 DC data1 read data0[6:0] | | |
| 13 | Cortex-M7_0 ITCM | Word1[31:0] – Cortex-M7_0 ITCM read data[63:32] | Word0[31:24] – Cortex-M7_0 ITCM read data ECC[7:0] | 64 | 8 |
| | | Word2[31:0] – Cortex-M7_0 ITCM read data[31:0] | | | |
| 14 | Cortex-M7_0 D0TCM | Word1[31:0] – Cortex-M7_0 D0TCM read data[31:0] | Word0[31:24] – Cortex-M7_0 D0TCM read data ECC[7:0] | 32 | 8 |
| 15 | Cortex-M7_0 D1TCM | Word1[31:0] – Cortex-M7_0 D1TCM read data[31:0] | Word0[31:24] – Cortex-M7_0 D1TCM read data ECC[7:0] | 32 | 8 |
| 16 | Cortex-M7_1 ITCM | Word1[31:0] – Cortex-M7_1 ITCM read data[63:32] | Word0[31:24] – Cortex-M7_1 ITCM read data ECC[7:0] | 64 | 8 |
| | | Word2[31:0] – Cortex-M7_1 ITCM read data[31:0] | | | |
| 17 | Cortex-M7_1 D0TCM | Word1[31:0] – Cortex-M7_1 D0TCM read data[31:0] | Word0[31:24] – Cortex-M7_1 D0TCM read data ECC[7:0] | 32 | 8 |
| 18 | Cortex-M7_1 D1TCM | Word1[31:0] – Cortex-M7_1 D1TCM read data[31:0] | Word0[31:24] – Cortex-M7_1 D1TCM read data ECC[7:0] | 32 | 8 |
| 19 | EMAC gasket | Word1[27:0] – EMAC AHB write data[63:36] | — | 188 | 0 |
| | | Word2[31:28] – EMAC AHB write data[35:32] | | | |
| | | Word2[27:0] – EMAC AHB write data[31:4] | | | |
| | | Word3[31:28] – EMAC AHB write data[3:0] | | | |
| | | Word3[27:0] – EMAC AHB read data[63:36] | | | |

*Table continues on the next page...*

Table 221. EIM channel mapping (continued)

| Channel # | Target | Data bits [1] | Check bits[1] | # of data bits | # of check bits |
|---|---|---|---|---|---|
| | | Word4[31:28] – EMAC AHB read data[35:32] | | | |
| | | Word4[27:0] – EMAC AHB read data[31:4] | | | |
| | | Word5[31:28] – EMAC AHB read data[3:0] | | | |
| | | Word5[27:0] – EMAC gasket monitor error injection[59:32] | | | |
| | | Word6[31:0] – EMAC gasket monitor error injection[31:0] | | | |
| 20 | Cortex-M7 TCM gasket | Word1[27:0] – TCM AHB write data[63:36] | — | 188 | 0 |
| | | Word2[31:28] – TCM AHB write data[35:32] | | | |
| | | Word2[27:0] – TCM AHB write data[31:4] | | | |
| | | Word3[31:28] – TCM AHB write data[3:0] | | | |
| | | Word3[27:0] – TCM AHB read data[63:36] | | | |
| | | Word4[31:28] – TCM AHB read data[35:32] | | | |
| | | Word4[27:0] – TCM AHB read data[31:4] | | | |
| | | Word5[31:28] – TCM AHB read data[3:0] | | | |
| | | Word5[27:0] – TCM gasket monitor error injection[59:32] | | | |
| | | Word6[31:0] – TCM gasket monitor error injection[31:0] | | | |
| 21 | DMA AXBS S0 gasket | Word1[27:0] – DMA AXBS S0 gasket monitor error injection[59:32] | — | 60 | 0 |
| | | Word2[31:0] – DMA AXBS S0 gasket monitor error injection[0:31] | | | |

*Table continues on the next page...*

Table 221. EIM channel mapping (continued)

| Channel # | Target | Data bits [1] | Check bits[1] | # of data bits | # of check bits |
|---|---|---|---|---|---|
| 22 | DMA AXBS S1 gasket | Word1[27:0] – DMA AXBS S1 gasket monitor error injection[59:32] | — | 60 | 0 |
| | | Word2[31:0] – DMA AXBS S1 gasket monitor error injection[31:0] | | | |
| 23 | HSE gasket | Word1[27:0] – HSE gasket monitor error injection[59:32] | — | 60 | 0 |
| | | Word2[31:0] – HSE gasket monitor error injection[31:0] | | | |
| 24 | QuadSPI gasket | Word1[27:0] – QuadSPI gasket monitor error injection[59:32] | — | 60 | 0 |
| | | Word2[31:0] – QuadSPI gasket monitor error injection[31:0] | | | |
| 25 | AIPS1 gasket | Word1[27:0] – AIPS1 gasket monitor error injection[59:32] | — | 60 | 0 |
| | | Word2[31:0] – AIPS1 gasket monitor error injection[31:0] | | | |
| 26 | AIPS2 gasket | Word1[27:0] – AIPS2 gasket monitor error injection[59:32] | — | 60 | 0 |
| | | Word2[31:0] – AIPS2 gasket monitor error injection[31:0] | | | |
| 27 | Cortex-M7 | Word1[29:0] – Cortex-M7 error injection[29:0] | — | 30 | 0 |
| 28 | ECC checking address | Word1[1:0] – Inject error on flash memory controller port 0 address checker | — | 24 | 0 |
| | | Word1[3:2] – Inject error on flash memory controller port 1 address checker | | | |
| | | Word1[4:4] – Inject error on flash memory controller port 2 address checker | | | |

*Table continues on the next page...*

Table 221. EIM channel mapping (continued)

| Channel # | Target | Data bits [1] | Check bits[1] | # of data bits | # of check bits |
|---|---|---|---|---|---|
| | | Word1[7:6] – Inject error on PRAM0 controller address checker | | | |
| | | Word1[9:8] – Inject error on PRAM1 controller address checker | | | |
| | | Word1[11:10] – Inject error on 64-bit TCM bus address checker | | | |
| | | Word1[13:12] – Inject error on QuadSPI path address checker | | | |
| | | Word1[15:14] – Inject error on AIPS0 address checker | | | |
| | | Word1[17:16] – Inject error on AIPS1 address checker | | | |
| | | Word1[19:18] – Inject error on AIPS2 address checker | | | |
| | | Word1[21:20] – Inject error on 32-bit TCM Cortex-M7_0 path address checker | | | |
| | | Word1[23:22] – Inject error on 32-bit TCM Cortex-M7_1 path address checker | | | |
| | | Word1[27:26] – Inject error on DMA AXBS S1 address parity checker | | | |
| 29 | EDC checking wdata | Word1[1:0] – Inject error on PRAM0 controller write data checker | — | 18 | 0 |
| | | Word1[3:2] – Inject error on PRAM1 controller write data checker | | | |
| | | Word1[5:4] – Inject error on 64-bit TCM bus write data checker | | | |
| | | Word1[7:6] – Reserved | | | |
| | | Word1[9:8] – Inject error on AIPS0 write data checker | | | |
| | | Word1[11:10] – Inject error on AIPS1 write data checker | | | |

*Table continues on the next page...*

Table 221. EIM channel mapping (continued)

| Channel # | Target | Data bits [1] | Check bits[1] | # of data bits | # of check bits |
|---|---|---|---|---|---|
| | | Word1[13:12] – Inject error on AIPS2 write data checker | | | |
| | | Word1[15:14] – Inject error on 32-bit TCM Cortex-M7_0 path write data checker | | | |
| | | Word1[17:16] – Inject error on 32-bit TCM Cortex-M7_1 path write data checker | | | |
| 30 | EDC checking rdata | Word1[1:0] – Inject error on Cortex-M7_0 AHBM read data checker | — | 18 | 0 |
| | | Word1[3:2] – Inject error on Cortex-M7_0 AHBP read data checker | | | |
| | | Word1[5:4] – Inject error on DMA read data checker | | | |
| | | Word1[7:6] – Inject error on STAM read data checker | | | |
| | | Word1[9:8] – Inject error on HSE read data checker | | | |
| | | Word1[11:10] – Inject error on EMAC read data checker | | | |
| | | Word1[13:12] – Inject error on Cortex-M7_1 AHBM read data checker | | | |
| | | Word1[15:14] – Inject error on Cortex-M7_1 AHBP read data checker | | | |
| | | Word1[17:16] – Inject error on 32-bit TCM bus path read data checker | | | |

1. You must write to EICHDi_WORDj registers to inject errors in the desired data and check bits. For details, see tables "Error injection channel descriptor: DATA_MASK details" and "DATA_MASK bit: Channel-word mapping" in this chapter.
2. SRAM1 is not available for the MCXE317, and MCXE315/MCXE316 variants.

The two enables, GEIEN and EICHEN*n*, enable the error injection functionality. The former enables it globally and the latter does it for a particular channel. This double-layer enable provides protection against accidental enabling and reconfiguration of the error injection function for each channel.

EIM provides support for inducing single-bit and multi-bit inversions on read data when accessing peripheral RAMs through its data mask registers.

**NOTE**

For enabling error injection on EDC gaskets (corresponding to channel 28, channel 29, and channel 30), you must also enable the fields corresponding to the required EDC gasket in the MSCM_ENEDC register before enabling the EIM channel.

EIM_EICHD1_CH01, EIM_EICHD1_CH08, EIM_EICHD1_CH09, EIM_EICHD1_CH10, EIM_EICHD1_CH11, EIM_EICHD1_CH12, EIM_EICHD1_CH16, EIM_EICHD1_CH17, EIM_EICHD1_CH18, EIM_EICHD1_CH19, EIM_EICHD1_CH21, EIM_EICHD1_CH22, EIM_EICHD1_CH24, EIM_EICHD1_CH26, EIM_EICHD1_CH27 are not present in MCXE317, hence the registers corresponding to these channels are also not present in MCXE317.

### 47.1.2 Behavior of EIM error injection on gaskets

The error injection on gaskets don't impact the actual data flow. The gasket read data, write data and the monitor error don't get changed when EIM error injection is done. The gasket compares the actual data with the modified data (with error injection) and flags the gasket alarm. In case of single-bit error also, the alarm is flagged.

The channels which depict such behavior are the gasket channels listed below:

- Channel 19: EMAC gasket
- Channel 20: Cortex-M7 TCM gasket
- Channel 21: DMA AXBS S0 gasket
- Channel 22: DMA AXBS S1 gasket
- Channel 23: HSE gasket
- Channel 24: QuadSPI gasket
- Channel 25: AIPS1 gasket
- Channel 26: AIPS2 gasket

## 47.2 Overview

The Error Injection Module (EIM) is mainly used for diagnostic purposes. It provides a method to test the diagnostics (memory ECC, interconnect parity) by error injection in the field. See the chip-specific EIM information to determine which functional safety features are supported by this method.

EIM enables you to inject artificial errors on error-checking mechanisms of a system, such as ECC for RAM read data and parity bits. For each such mechanism that EIM supports on the chip, EIM can inject single-bit and multi-bit inversions on data in the applicable target bus. Injecting faults on memory accesses can be used to exercise the SEC-DED ECC function of the related system.

**NOTE**

Terminology in this chapter has been updated as follows:

**Table 222. Updated terms**

| Updated term | Deprecated term |
|---|---|
| Controller | Master |

### 47.2.1 Features

The EIM includes these features:

- Supports 31 error injection channels. See the chip-specific EIM information for channel assignment details.
- Protection against accidental enable and reconfiguration error injection function via two-stage enable mechanism

### 47.2.2 Block diagram

The following diagram shows an example of EIM implementation with a 64-bit read data bus and an 8-bit checkbit bus.

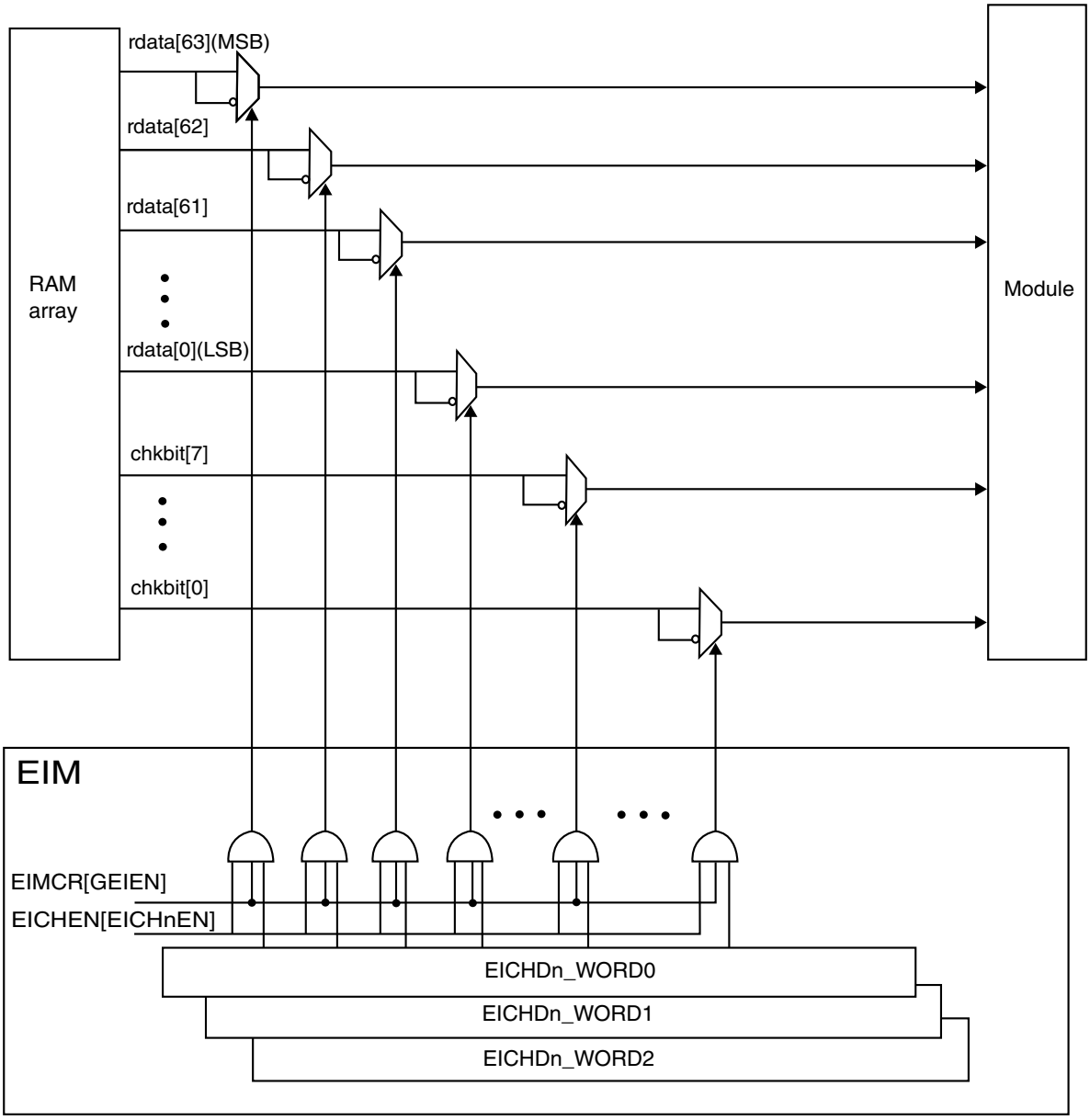



**Figure 177. EIM functional block diagram (64-bit read data bus and 8-bit check bit bus)**

Several memory elements are implemented within a device, which may not only be the large memory blocks (Flash and SRAM) but also smaller memories like caches, the TCD blocks, and the embedded peripheral memories. Some larger memories may actually be built from multiple memory elements, dependent on their size or function. Each of these memory elements implements its own control logic, the memory controller, that performs the accesses to the actual memory, the memory array. An EIM channel is associated with a memory controller and provides the capability to alter one or multiple signals in the read access path from the corresponding memory array(s). Only memory controllers controlling a safety related memory may be associated with an EIM channel.

## 47.3 Functional description

The EIM provides protection against accidental enabling and reconfiguration of the error injection function by enforcing a two-stage enablement mechanism. To properly enable the error injection mechanism for a channel:

- Write 1 to the EICHEN[EICH*n*EN] field, where *n* denotes the channel number.
- Write 1 to EIMCR[GEIEN].

**NOTE**

When the use case for a channel requires writing any EICHD*n*_WORD register, write the EICHD*n*_WORD register before executing the two-stage enablement mechanism. A successful write to any EICHD*n*_WORD register clears the corresponding EICHEN[EICH*n*EN] field.

The EIM supports 31 error injection channels. See the chip-specific EIM information for channel assignment details. Each channel:

- Can be assigned to a single memory array interface by intercepting the assigned memory read data bus and checkbit bus, and injects errors by inverting the value transmitted for selected bits on each bus line.
- Can be assigned to a redundant comparison unit by intercepting the signals being compared, and injecting errors by inverting the value transmitted for selected bits on each bus line.

On a memory read access, the applicable EICHD*n*_WORD registers define which bits of the read data and/or checkbit bus to invert.

Figure 177 depicts the interception and override of a 64-bit read data bus and an 8-bit checkbit data bus for an example memory array.

### Error injection scenarios

The EIM supports these cases of error injection:

- To generate a single-bit error, invert only 1 bit of the CHKBIT_MASK or DATA_MASK in the EICHD*n*_WORD registers.
- To generate a multi-bit error, invert only 2 bits of the CHKBIT_MASK or DATA_MASK in the EICHD*n*_WORD registers.

**NOTE**

An attempt to invert more than 2 bits in one operation might result in undefined behavior.

To enable error injection:

1. Set the EICHD*n*_WORD*m*[CHKBIT_MASK] and EICHD*n*_WORD*m*[B*a*_*b*DATA_MASK] fields for each channel that will be driving an injection.
2. Program the EICHEN register to enable the channels that will be injecting errors.
3. Set the EIMCR[GEIEN] field to globally allow all enabled channels to actively inject errors.

To disable error injection, either disable the EIMCR[GEIEN] field or disable the individual channel enable fields of the EICHEN register.

## 47.4 Initialization

This module does not require initialization.

## 47.6 EIM register descriptions

The EIM provides a programming model mapped to an on-platform peripheral slot.

### Programming model access

All system bus controllers can access the programming model:

- Only in supervisor mode

- Using only 32-bit (word) accesses

Any of the following attempted references to the programming model generates an error termination:

- In user mode
- Using non-32-bit access sizes
- To undefined (reserved) addresses

Attempted updates to the programming model while the EIM is in the midst of an operation result in non-deterministic behavior.

**Error injection channel descriptor: function and structure**

Each error injection channel descriptor:

- Specifies a mask that defines which bits of the read data and/or checkbit bus from target RAM are inverted on a read access.
- Consists of a 288-bit (36-byte) structure, composed of nine 32-bit words, in the EIM programming model. Unused words are not documented.
  - Word0 (EICHD*n*_WORD0), if present, defines the checkbit mask.
  - Word1 (EICHD*n*_WORD1) and additional words, if present, define the data mask. Word registers subsequent to Word1 are present only when required by the total width of the channel's data mask. Error injection channel descriptor: DATA_MASK details.

The multiple channel descriptors are organized sequentially.

**Error injection channel descriptor: DATA_MASK details**

For each channel: The following tables show the distribution of DATA_MASK's bits across the WORD registers. The first table shows the total width of DATA_MASK and the distribution of its bits across WORD1, WORD2, and WORD3. The second table shows the distribution of DATA_MASK's bits across WORD4 and subsequent registers.

**Table 223. Error injection channel descriptor: DATA_MASK details**

| Channel | DATA_MASK total width (bits) | Specific bits of DATA_MASK in | | |
|---|---|---|---|---|
| | | WORD1 | WORD2 | WORD3 |
| 0 | 64 | 63-32 | 31-0 | — |
| 1 | 64 | 63-32 | 31-0 | — |
| 2 | 64 | 63-32 | 31-0 | — |
| 3 | 44 | 43-32 | 31-0 | — |
| 4 | 128 | 127-96 | 95-64 | 63-32 |
| 5 | 104 | 103-96 | 95-64 | 63-32 |
| 6 | 128 | 127-96 | 95-64 | 63-32 |
| 7 | 128 | 127-96 | 95-64 | 63-32 |
| 8 | 44 | 43-32 | 31-0 | — |
| 9 | 128 | 127-96 | 95-64 | 63-32 |
| 10 | 104 | 103-96 | 95-64 | 63-32 |

*Table continues on the next page...*

Table 223. Error injection channel descriptor: DATA_MASK details (continued)

| Channel | DATA_MASK total width (bits) | Specific bits of DATA_MASK in | | |
|---|---|---|---|---|
| | | WORD1 | WORD2 | WORD3 |
| 11 | 128 | 127-96 | 95-64 | 63-32 |
| 12 | 128 | 127-96 | 95-64 | 63-32 |
| 13 | 64 | 63-32 | 31-0 | — |
| 14 | 32 | 31-0 | — | — |
| 15 | 32 | 31-0 | — | — |
| 16 | 64 | 63-32 | 31-0 | — |
| 17 | 32 | 31-0 | — | — |
| 18 | 32 | 31-0 | — | — |
| 19 | 188 | 187-160 | 159-128 | 127-96 |
| 20 | 188 | 187-160 | 159-128 | 127-96 |
| 21 | 60 | 59-32 | 31-0 | — |
| 22 | 60 | 59-32 | 31-0 | — |
| 23 | 60 | 59-32 | 31-0 | — |
| 24 | 60 | 59-32 | 31-0 | — |
| 25 | 60 | 59-32 | 31-0 | — |
| 26 | 60 | 59-32 | 31-0 | — |
| 27 | 30 | 29-0 | — | — |
| 28 | 24 | 23-0 | — | — |
| 29 | 18 | 17-0 | — | — |
| 30 | 18 | 17-0 | — | — |

Table 224. DATA_MASK bit: Channel-word mapping

| Channel | Specific bits of DATA_MASK in | | | | |
|---|---|---|---|---|---|
| | WORD4 | WORD5 | WORD6 | WORD7 | WORD8 |
| 4 | 31-0 | — | — | — | — |
| 5 | 31-0 | — | — | — | — |

*Table continues on the next page...*

Table 224. DATA_MASK bit: Channel-word mapping (continued)

| Channel | Specific bits of DATA_MASK in | | | | |
|---|---|---|---|---|---|
| | WORD4 | WORD5 | WORD6 | WORD7 | WORD8 |
| 6 | 31-0 | — | — | — | — |
| 7 | 31-0 | — | — | — | — |
| 9 | 31-0 | — | — | — | — |
| 10 | 31-0 | — | — | — | — |
| 11 | 31-0 | — | — | — | — |
| 12 | 31-0 | — | — | — | — |
| 19 | 95-64 | 63-32 | 31-0 | — | — |
| 20 | 95-64 | 63-32 | 31-0 | — | — |

### 47.6.1 EIM memory map

EIM base address: 4025_8000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Error Injection Module Configuration Register (EIMCR) | 32 | RW | 0000_0000h |
| 4h | Error Injection Channel Enable register (EICHEN) | 32 | RW | 0000_0000h |
| 100h | Error Injection Channel Descriptor 0, Word0 (EICHD0_WORD0) | 32 | RW | 0000_0000h |
| 104h | Error Injection Channel Descriptor 0, Word1 (EICHD0_WORD1) | 32 | RW | 0000_0000h |
| 108h | Error Injection Channel Descriptor 0, Word2 (EICHD0_WORD2) | 32 | RW | 0000_0000h |
| 140h | Error Injection Channel Descriptor 1, Word0 (EICHD1_WORD0) | 32 | RW | 0000_0000h |
| 144h | Error Injection Channel Descriptor 1, Word1 (EICHD1_WORD1) | 32 | RW | 0000_0000h |
| 148h | Error Injection Channel Descriptor 1, Word2 (EICHD1_WORD2) | 32 | RW | 0000_0000h |
| 180h | Error Injection Channel Descriptor 2, Word0 (EICHD2_WORD0) | 32 | RW | 0000_0000h |
| 184h | Error Injection Channel Descriptor 2, Word1 (EICHD2_WORD1) | 32 | RW | 0000_0000h |
| 188h | Error Injection Channel Descriptor 2, Word2 (EICHD2_WORD2) | 32 | RW | 0000_0000h |
| 1C0h | Error Injection Channel Descriptor 3, Word0 (EICHD3_WORD0) | 32 | RW | 0000_0000h |
| 1C4h | Error Injection Channel Descriptor 3, Word1 (EICHD3_WORD1) | 32 | RW | 0000_0000h |
| 1C8h | Error Injection Channel Descriptor 3, Word2 (EICHD3_WORD2) | 32 | RW | 0000_0000h |
| 200h | Error Injection Channel Descriptor 4, Word0 (EICHD4_WORD0) | 32 | RW | 0000_0000h |
| 204h | Error Injection Channel Descriptor 4, Word1 (EICHD4_WORD1) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 208h | Error Injection Channel Descriptor 4, Word2 (EICHD4_WORD2) | 32 | RW | 0000_0000h |
| 20Ch | Error Injection Channel Descriptor 4, Word3 (EICHD4_WORD3) | 32 | RW | 0000_0000h |
| 210h | Error Injection Channel Descriptor 4, Word4 (EICHD4_WORD4) | 32 | RW | 0000_0000h |
| 240h | Error Injection Channel Descriptor 5, Word0 (EICHD5_WORD0) | 32 | RW | 0000_0000h |
| 244h | Error Injection Channel Descriptor 5, Word1 (EICHD5_WORD1) | 32 | RW | 0000_0000h |
| 248h | Error Injection Channel Descriptor 5, Word2 (EICHD5_WORD2) | 32 | RW | 0000_0000h |
| 24Ch | Error Injection Channel Descriptor 5, Word3 (EICHD5_WORD3) | 32 | RW | 0000_0000h |
| 250h | Error Injection Channel Descriptor 5, Word4 (EICHD5_WORD4) | 32 | RW | 0000_0000h |
| 280h | Error Injection Channel Descriptor 6, Word0 (EICHD6_WORD0) | 32 | RW | 0000_0000h |
| 284h | Error Injection Channel Descriptor 6, Word1 (EICHD6_WORD1) | 32 | RW | 0000_0000h |
| 288h | Error Injection Channel Descriptor 6, Word2 (EICHD6_WORD2) | 32 | RW | 0000_0000h |
| 28Ch | Error Injection Channel Descriptor 6, Word3 (EICHD6_WORD3) | 32 | RW | 0000_0000h |
| 290h | Error Injection Channel Descriptor 6, Word4 (EICHD6_WORD4) | 32 | RW | 0000_0000h |
| 2C0h | Error Injection Channel Descriptor 7, Word0 (EICHD7_WORD0) | 32 | RW | 0000_0000h |
| 2C4h | Error Injection Channel Descriptor 7, Word1 (EICHD7_WORD1) | 32 | RW | 0000_0000h |
| 2C8h | Error Injection Channel Descriptor 7, Word2 (EICHD7_WORD2) | 32 | RW | 0000_0000h |
| 2CCh | Error Injection Channel Descriptor 7, Word3 (EICHD7_WORD3) | 32 | RW | 0000_0000h |
| 2D0h | Error Injection Channel Descriptor 7, Word4 (EICHD7_WORD4) | 32 | RW | 0000_0000h |
| 300h | Error Injection Channel Descriptor 8, Word0 (EICHD8_WORD0) | 32 | RW | 0000_0000h |
| 304h | Error Injection Channel Descriptor 8, Word1 (EICHD8_WORD1) | 32 | RW | 0000_0000h |
| 308h | Error Injection Channel Descriptor 8, Word2 (EICHD8_WORD2) | 32 | RW | 0000_0000h |
| 340h | Error Injection Channel Descriptor 9, Word0 (EICHD9_WORD0) | 32 | RW | 0000_0000h |
| 344h | Error Injection Channel Descriptor 9, Word1 (EICHD9_WORD1) | 32 | RW | 0000_0000h |
| 348h | Error Injection Channel Descriptor 9, Word2 (EICHD9_WORD2) | 32 | RW | 0000_0000h |
| 34Ch | Error Injection Channel Descriptor 9, Word3 (EICHD9_WORD3) | 32 | RW | 0000_0000h |
| 350h | Error Injection Channel Descriptor 9, Word4 (EICHD9_WORD4) | 32 | RW | 0000_0000h |
| 380h | Error Injection Channel Descriptor 10, Word0 (EICHD10_WORD0) | 32 | RW | 0000_0000h |
| 384h | Error Injection Channel Descriptor 10, Word1 (EICHD10_WORD1) | 32 | RW | 0000_0000h |
| 388h | Error Injection Channel Descriptor 10, Word2 (EICHD10_WORD2) | 32 | RW | 0000_0000h |
| 38Ch | Error Injection Channel Descriptor 10, Word3 (EICHD10_WORD3) | 32 | RW | 0000_0000h |
| 390h | Error Injection Channel Descriptor 10, Word4 (EICHD10_WORD4) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 3C0h | Error Injection Channel Descriptor 11, Word0 (EICHD11_WORD0) | 32 | RW | 0000_0000h |
| 3C4h | Error Injection Channel Descriptor 11, Word1 (EICHD11_WORD1) | 32 | RW | 0000_0000h |
| 3C8h | Error Injection Channel Descriptor 11, Word2 (EICHD11_WORD2) | 32 | RW | 0000_0000h |
| 3CCh | Error Injection Channel Descriptor 11, Word3 (EICHD11_WORD3) | 32 | RW | 0000_0000h |
| 3D0h | Error Injection Channel Descriptor 11, Word4 (EICHD11_WORD4) | 32 | RW | 0000_0000h |
| 400h | Error Injection Channel Descriptor 12, Word0 (EICHD12_WORD0) | 32 | RW | 0000_0000h |
| 404h | Error Injection Channel Descriptor 12, Word1 (EICHD12_WORD1) | 32 | RW | 0000_0000h |
| 408h | Error Injection Channel Descriptor 12, Word2 (EICHD12_WORD2) | 32 | RW | 0000_0000h |
| 40Ch | Error Injection Channel Descriptor 12, Word3 (EICHD12_WORD3) | 32 | RW | 0000_0000h |
| 410h | Error Injection Channel Descriptor 12, Word4 (EICHD12_WORD4) | 32 | RW | 0000_0000h |
| 440h | Error Injection Channel Descriptor 13, Word0 (EICHD13_WORD0) | 32 | RW | 0000_0000h |
| 444h | Error Injection Channel Descriptor 13, Word1 (EICHD13_WORD1) | 32 | RW | 0000_0000h |
| 448h | Error Injection Channel Descriptor 13, Word2 (EICHD13_WORD2) | 32 | RW | 0000_0000h |
| 480h | Error Injection Channel Descriptor 14, Word0 (EICHD14_WORD0) | 32 | RW | 0000_0000h |
| 484h | Error Injection Channel Descriptor 14, Word1 (EICHD14_WORD1) | 32 | RW | 0000_0000h |
| 4C0h | Error Injection Channel Descriptor 15, Word0 (EICHD15_WORD0) | 32 | RW | 0000_0000h |
| 4C4h | Error Injection Channel Descriptor 15, Word1 (EICHD15_WORD1) | 32 | RW | 0000_0000h |
| 500h | Error Injection Channel Descriptor 16, Word0 (EICHD16_WORD0) | 32 | RW | 0000_0000h |
| 504h | Error Injection Channel Descriptor 16, Word1 (EICHD16_WORD1) | 32 | RW | 0000_0000h |
| 508h | Error Injection Channel Descriptor 16, Word2 (EICHD16_WORD2) | 32 | RW | 0000_0000h |
| 540h | Error Injection Channel Descriptor 17, Word0 (EICHD17_WORD0) | 32 | RW | 0000_0000h |
| 544h | Error Injection Channel Descriptor 17, Word1 (EICHD17_WORD1) | 32 | RW | 0000_0000h |
| 580h | Error Injection Channel Descriptor 18, Word0 (EICHD18_WORD0) | 32 | RW | 0000_0000h |
| 584h | Error Injection Channel Descriptor 18, Word1 (EICHD18_WORD1) | 32 | RW | 0000_0000h |
| 5C4h | Error Injection Channel Descriptor 19, Word1 (EICHD19_WORD1) | 32 | RW | 0000_0000h |
| 5C8h | Error Injection Channel Descriptor 19, Word2 (EICHD19_WORD2) | 32 | RW | 0000_0000h |
| 5CCh | Error Injection Channel Descriptor 19, Word3 (EICHD19_WORD3) | 32 | RW | 0000_0000h |
| 5D0h | Error Injection Channel Descriptor 19, Word4 (EICHD19_WORD4) | 32 | RW | 0000_0000h |
| 5D4h | Error Injection Channel Descriptor 19, Word5 (EICHD19_WORD5) | 32 | RW | 0000_0000h |
| 5D8h | Error Injection Channel Descriptor 19, Word6 (EICHD19_WORD6) | 32 | RW | 0000_0000h |
| 604h | Error Injection Channel Descriptor 20, Word1 (EICHD20_WORD1) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 608h | Error Injection Channel Descriptor 20, Word2 (EICHD20_WORD2) | 32 | RW | 0000_0000h |
| 60Ch | Error Injection Channel Descriptor 20, Word3 (EICHD20_WORD3) | 32 | RW | 0000_0000h |
| 610h | Error Injection Channel Descriptor 20, Word4 (EICHD20_WORD4) | 32 | RW | 0000_0000h |
| 614h | Error Injection Channel Descriptor 20, Word5 (EICHD20_WORD5) | 32 | RW | 0000_0000h |
| 618h | Error Injection Channel Descriptor 20, Word6 (EICHD20_WORD6) | 32 | RW | 0000_0000h |
| 644h | Error Injection Channel Descriptor 21, Word1 (EICHD21_WORD1) | 32 | RW | 0000_0000h |
| 648h | Error Injection Channel Descriptor 21, Word2 (EICHD21_WORD2) | 32 | RW | 0000_0000h |
| 684h | Error Injection Channel Descriptor 22, Word1 (EICHD22_WORD1) | 32 | RW | 0000_0000h |
| 688h | Error Injection Channel Descriptor 22, Word2 (EICHD22_WORD2) | 32 | RW | 0000_0000h |
| 6C4h | Error Injection Channel Descriptor 23, Word1 (EICHD23_WORD1) | 32 | RW | 0000_0000h |
| 6C8h | Error Injection Channel Descriptor 23, Word2 (EICHD23_WORD2) | 32 | RW | 0000_0000h |
| 704h | Error Injection Channel Descriptor 24, Word1 (EICHD24_WORD1) | 32 | RW | 0000_0000h |
| 708h | Error Injection Channel Descriptor 24, Word2 (EICHD24_WORD2) | 32 | RW | 0000_0000h |
| 744h | Error Injection Channel Descriptor 25, Word1 (EICHD25_WORD1) | 32 | RW | 0000_0000h |
| 748h | Error Injection Channel Descriptor 25, Word2 (EICHD25_WORD2) | 32 | RW | 0000_0000h |
| 784h | Error Injection Channel Descriptor 26, Word1 (EICHD26_WORD1) | 32 | RW | 0000_0000h |
| 788h | Error Injection Channel Descriptor 26, Word2 (EICHD26_WORD2) | 32 | RW | 0000_0000h |
| 7C4h | Error Injection Channel Descriptor 27, Word1 (EICHD27_WORD1) | 32 | RW | 0000_0000h |
| 804h | Error Injection Channel Descriptor 28, Word1 (EICHD28_WORD1) | 32 | RW | 0000_0000h |
| 844h | Error Injection Channel Descriptor 29, Word1 (EICHD29_WORD1) | 32 | RW | 0000_0000h |
| 884h | Error Injection Channel Descriptor 30, Word1 (EICHD30_WORD1) | 32 | RW | 0000_0000h |

### 47.6.2 Error Injection Module Configuration Register (EIMCR)

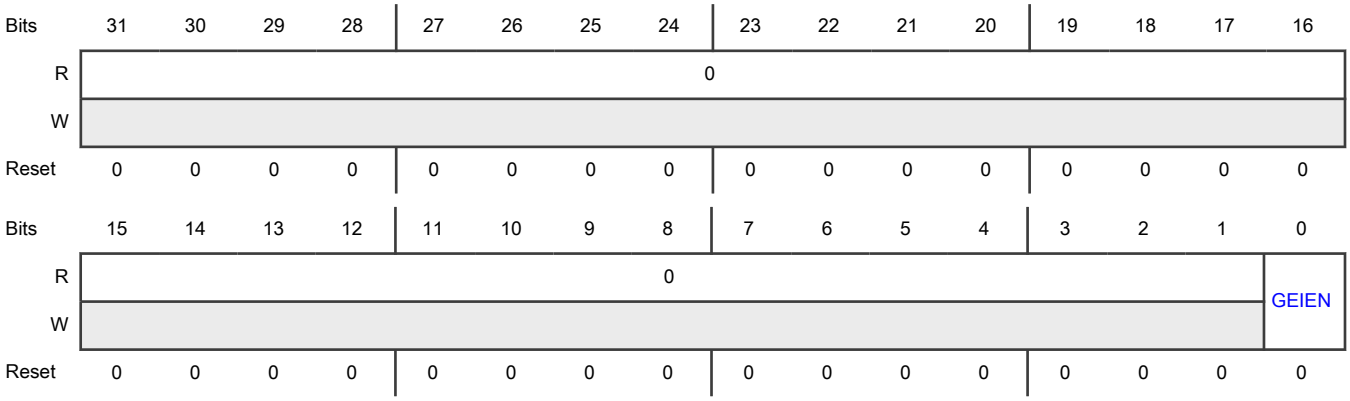
**Offset**

| Register | Offset |
|---|---|
| EIMCR | 0h |

**Function**

The EIM Configuration Register is used to globally enable/disable the error injection function.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>GEIEN | Global Error Injection Enable<br>This bit globally enables or disables the error injection function of the EIM. This field is initialized by hardware reset.<br>0b - Disabled<br>1b - Enabled |

## 47.6.3 Error Injection Channel Enable register (EICHEN)

### Offset

| Register | Offset |
|---|---|
| EICHEN | 4h |

### Function

Each field of the Error Injection Channel Enable register (EICHEN) is used to enable or disable the corresponding error injection channel.

**NOTE**

To enable an error injection channel, the Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EICH0EN | EICH1EN | EICH2EN | EICH3EN | EICH4EN | EICH5EN | EICH6EN | EICH7EN | EICH8EN | EICH9EN | EICH10EN | EICH11EN | EICH12EN | EICH13EN | EICH14EN | EICH15EN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EICH16EN | EICH17EN | EICH18EN | EICH19EN | EICH20EN | EICH21EN | EICH22EN | EICH23EN | EICH24EN | EICH25EN | EICH26EN | EICH27EN | EICH28EN | EICH29EN | EICH30EN | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>EICH0EN | Error Injection Channel 0 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 0<br>1b - Error injection is enabled on Error Injection Channel 0 |
| 30<br>EICH1EN | Error Injection Channel 1 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 1<br>1b - Error injection is enabled on Error Injection Channel 1 |
| 29<br>EICH2EN | Error Injection Channel 2 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 2<br>1b - Error injection is enabled on Error Injection Channel 2 |
| 28<br>EICH3EN | Error Injection Channel 3 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 3<br>1b - Error injection is enabled on Error Injection Channel 3 |
| 27<br>EICH4EN | Error Injection Channel 4 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 4<br>1b - Error injection is enabled on Error Injection Channel 4 |
| 26<br>EICH5EN | Error Injection Channel 5 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 5<br>1b - Error injection is enabled on Error Injection Channel 5 |
| 25<br>EICH6EN | Error Injection Channel 6 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 6<br>1b - Error injection is enabled on Error Injection Channel 6 |
| 24<br>EICH7EN | Error Injection Channel 7 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 7<br>1b - Error injection is enabled on Error Injection Channel 7 |
| 23<br>EICH8EN | Error Injection Channel 8 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 8<br>1b - Error injection is enabled on Error Injection Channel 8 |
| 22<br>EICH9EN | Error Injection Channel 9 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 9<br>1b - Error injection is enabled on Error Injection Channel 9 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 21<br>EICH10EN | Error Injection Channel 10 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 10<br>1b - Error injection is enabled on Error Injection Channel 10 |
| 20<br>EICH11EN | Error Injection Channel 11 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 11<br>1b - Error injection is enabled on Error Injection Channel 11 |
| 19<br>EICH12EN | Error Injection Channel 12 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 12<br>1b - Error injection is enabled on Error Injection Channel 12 |
| 18<br>EICH13EN | Error Injection Channel 13 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Error injection is disabled on Error Injection Channel 13<br>1b - Error injection is enabled on Error Injection Channel 13 |
| 17<br>EICH14EN | Error Injection Channel 14 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 14<br>1b - Error injection is enabled on Error Injection Channel 14 |
| 16<br>EICH15EN | Error Injection Channel 15 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 15<br>1b - Error injection is enabled on Error Injection Channel 15 |
| 15<br>EICH16EN | Error Injection Channel 16 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 16<br>1b - Error injection is enabled on Error Injection Channel 16 |
| 14<br>EICH17EN | Error Injection Channel 17 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 17<br>1b - Error injection is enabled on Error Injection Channel 17 |
| 13<br>EICH18EN | Error Injection Channel 18 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 18<br>1b - Error injection is enabled on Error Injection Channel 18 |
| 12<br>EICH19EN | Error Injection Channel 19 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 19<br>1b - Error injection is enabled on Error Injection Channel 19 |
| 11<br>EICH20EN | Error Injection Channel 20 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 20<br>1b - Error injection is enabled on Error Injection Channel 20 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 10<br>EICH21EN | Error Injection Channel 21 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 21<br>1b - Error injection is enabled on Error Injection Channel 21 |
| 9<br>EICH22EN | Error Injection Channel 22 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 22<br>1b - Error injection is enabled on Error Injection Channel 22 |
| 8<br>EICH23EN | Error Injection Channel 23 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 23<br>1b - Error injection is enabled on Error Injection Channel 23 |
| 7<br>EICH24EN | Error Injection Channel 24 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Error injection is disabled on Error Injection Channel 24<br>1b - Error injection is enabled on Error Injection Channel 24 |
| 6<br>EICH25EN | Error Injection Channel 25 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 25<br>1b - Error injection is enabled on Error Injection Channel 25 |
| 5<br>EICH26EN | Error Injection Channel 26 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 26<br>1b - Error injection is enabled on Error Injection Channel 26 |
| 4<br>EICH27EN | Error Injection Channel 27 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 27<br>1b - Error injection is enabled on Error Injection Channel 27 |
| 3<br>EICH28EN | Error Injection Channel 28 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 28<br>1b - Error injection is enabled on Error Injection Channel 28 |
| 2<br>EICH29EN | Error Injection Channel 29 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 29<br>1b - Error injection is enabled on Error Injection Channel 29 |
| 1<br>EICH30EN | Error Injection Channel 30 Enable<br>This field enables the corresponding error injection channel. The Global Error Injection Enable (EIMCR[GEIEN]) field must also be asserted to enable error injection.<br>After error injection is enabled, all subsequent read accesses incur one or more bit inversions as defined in the corresponding EICHD*n*_WORD registers. Error injection remains in effect until the error injection channel is manually disabled via software.<br>Any write to the corresponding EICHD*n*_WORD registers clears the corresponding EICHEN[EICH*n*EN] field, disabling the error injection channel.<br>0b - Error injection is disabled on Error Injection Channel 30<br>1b - Error injection is enabled on Error Injection Channel 30 |
| 0<br>— | Reserved |

## 47.6.4 Error Injection Channel Descriptor n, Word0 (EICHD0_WORD0 - EICHD2_WORD0)

**Offset**

| Register | Offset |
|---|---|
| EICHD0_WORD0 | 100h |
| EICHD1_WORD0 | 140h |

*Table continues on the next page...*

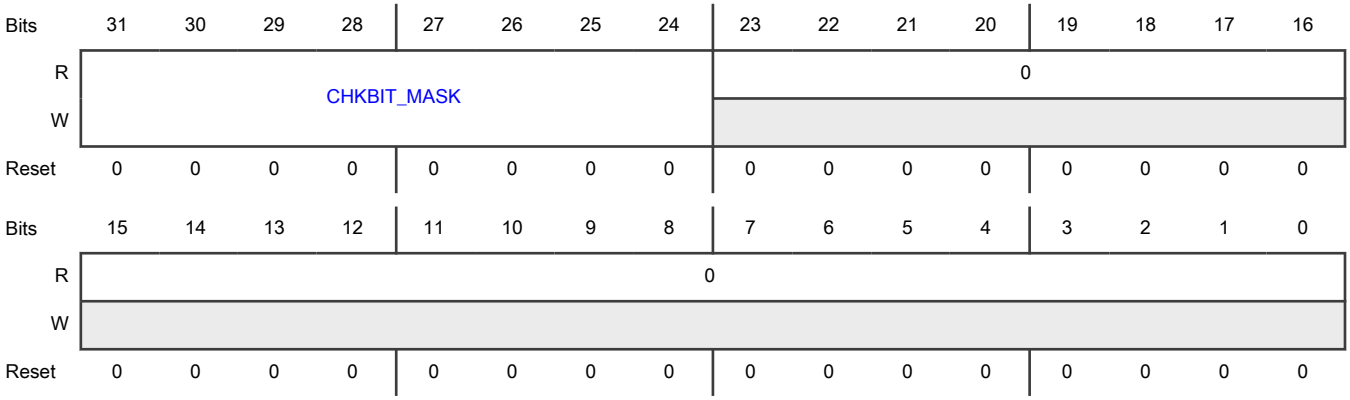
*Table continued from the previous page...*

| Register | Offset |
|---|---|
| EICHD2_WORD0 | 180h |

### Function

The first word of the Error Injection Channel Descriptor defines a left-justified mask field: CHKBIT_MASK. Each bit of CHKBIT_MASK specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified on read accesses. Successful write to this field clears the corresponding error injection channel valid bit, EICHEN[EICH*n*EN].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>CHKBIT_MASK | Checkbit Mask<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored.<br>For any unique details about the mapping of CHKBIT_MASK's bits to a channel's target RAM, see the chip-specific EIM information.<br>**NOTE**<br>Because CHKBIT_MASK is left-justified, the highest bit in the bit range is always in the position of the most significant bit. For CHKBIT_MASK[7:0] (8 bits wide), CHKBIT_MASK[7] is in the position of the most significant bit.<br>0b - The corresponding bit of the checkbit bus remains unmodified.<br>1b - The corresponding bit of the checkbit bus is inverted. |
| 23-0<br>— | Reserved |

## 47.6.5 Error Injection Channel Descriptor n, Word1 (EICHD0_WORD1 - EICHD2_WORD1)

### Offset

| Register | Offset |
|---|---|
| EICHD0_WORD1 | 104h |
| EICHD1_WORD1 | 144h |
| EICHD2_WORD1 | 184h |

### Function

The second word of the Error Injection Channel Descriptor defines a right-justified mask field. The bits in B0_3DATA_MASK correspond to bytes 0–3 of the target bus. Each bit specifies whether the corresponding bit of the target bus should be inverted or remain unmodified on read accesses. A successful write to this field clears the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | B0_3DATA_MASK | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | B0_3DATA_MASK | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>B0_3DATA_MASK | Data Mask Bytes 0-3<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored.<br>**NOTE**<br>For the specific DATA_MASK bits to which B0_3DATA_MASK corresponds, See Error injection channel descriptor: DATA_MASK details.<br>0b - The corresponding bit of bytes 0-3 on the read data bus remains unmodified.<br>1b - The corresponding bit of bytes 0-3 on the read data bus is inverted. |

### 47.6.6 Error Injection Channel Descriptor n, Word2 (EICHD0_WORD2 - EICHD26_WORD2)

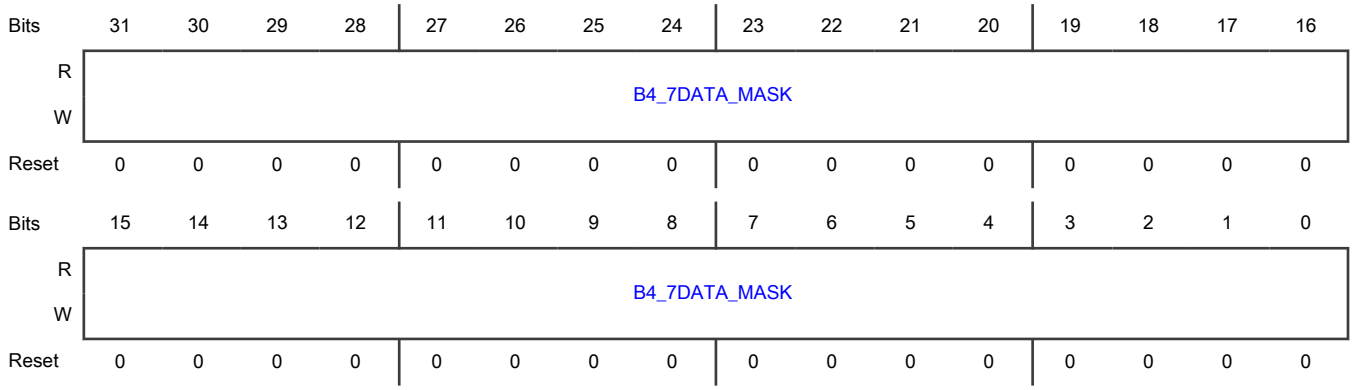
#### Offset

| Register | Offset |
|---|---|
| EICHD0_WORD2 | 108h |
| EICHD1_WORD2 | 148h |
| EICHD2_WORD2 | 188h |
| EICHD3_WORD2 | 1C8h |
| EICHD4_WORD2 | 208h |
| EICHD5_WORD2 | 248h |
| EICHD6_WORD2 | 288h |
| EICHD7_WORD2 | 2C8h |
| EICHD8_WORD2 | 308h |
| EICHD9_WORD2 | 348h |
| EICHD10_WORD2 | 388h |
| EICHD11_WORD2 | 3C8h |
| EICHD12_WORD2 | 408h |
| EICHD13_WORD2 | 448h |
| EICHD16_WORD2 | 508h |
| EICHD19_WORD2 | 5C8h |
| EICHD20_WORD2 | 608h |
| EICHD21_WORD2 | 648h |
| EICHD22_WORD2 | 688h |
| EICHD23_WORD2 | 6C8h |
| EICHD24_WORD2 | 708h |
| EICHD25_WORD2 | 748h |
| EICHD26_WORD2 | 788h |

#### Function

The third word of the Error Injection Channel Descriptor, when present, defines a right-justified mask field. The bits in B4_7DATA_MASK correspond to bytes 4–7 of the read data bus. Each bit specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified on read accesses. A successful write to this field clears the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>B4_7DATA_MASK | Data Mask Bytes 4-7<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified.<br>**NOTE**<br>For each channel: For the specific DATA_MASK bits to which B4_7DATA_MASK corresponds, See Error injection channel descriptor: DATA_MASK details.<br>0b - The corresponding bit of bytes 4-7 on the read data bus remains unmodified.<br>1b - The corresponding bit of bytes 4-7 on the read data bus is inverted. |

## 47.6.7 Error Injection Channel Descriptor 3, Word0 (EICHD3_WORD0)

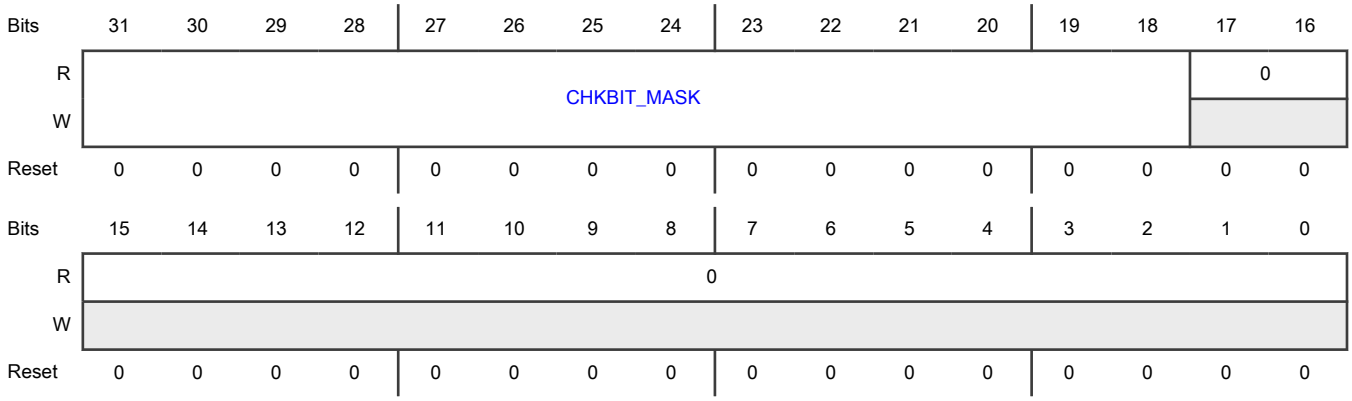
### Offset

| Register | Offset |
|---|---|
| EICHD3_WORD0 | 1C0h |

### Function

The first word of the Error Injection Channel Descriptor defines a left-justified mask field: CHKBIT_MASK. Each bit of CHKBIT_MASK specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified on read accesses. Successful write to this field clears the corresponding error injection channel valid bit, EICHEN[EICH*n*EN].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-18<br>CHKBIT_MASK | Checkbit Mask<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored.<br>For any unique details about the mapping of CHKBIT_MASK's bits to a channel's target RAM, see the chip-specific EIM information.<br>**NOTE**<br>Because CHKBIT_MASK is left-justified, the highest bit in the bit range is always in the position of the most significant bit. For CHKBIT_MASK[13:0] (14 bits wide), CHKBIT_MASK[13] is in the position of the most significant bit.<br>0b - The corresponding bit of the checkbit bus remains unmodified.<br>1b - The corresponding bit of the checkbit bus is inverted. |
| 17-0<br>— | Reserved |

## 47.6.8 Error Injection Channel Descriptor 3, Word1 (EICHD3_WORD1)

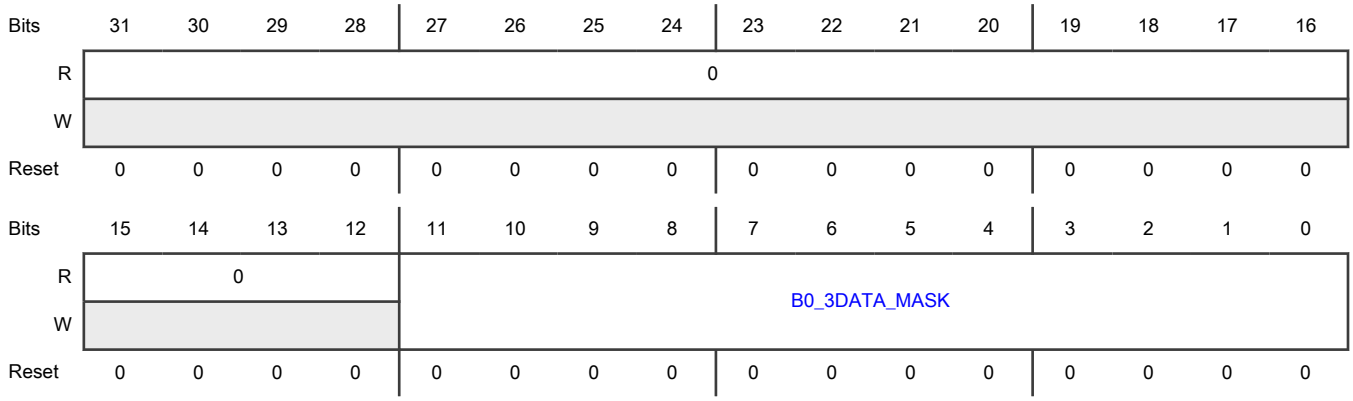
### Offset

| Register | Offset |
|---|---|
| EICHD3_WORD1 | 1C4h |

### Function

The second word of the Error Injection Channel Descriptor defines a right-justified mask field. The bits in B0_3DATA_MASK correspond to bytes 0–3 of the target bus. Each bit specifies whether the corresponding bit of the target bus should be inverted or remain unmodified on read accesses. A successful write to this field clears the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11-0<br>B0_3DATA_MASK | Data Mask Bytes 0-3<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored.<br>**NOTE**<br>For the specific DATA_MASK bits to which B0_3DATA_MASK corresponds, See Error injection channel descriptor: DATA_MASK details.<br>0b - The corresponding bit of bytes 0-3 on the read data bus remains unmodified.<br>1b - The corresponding bit of bytes 0-3 on the read data bus is inverted. |

## 47.6.9 Error Injection Channel Descriptor 4, Word0 (EICHD4_WORD0)

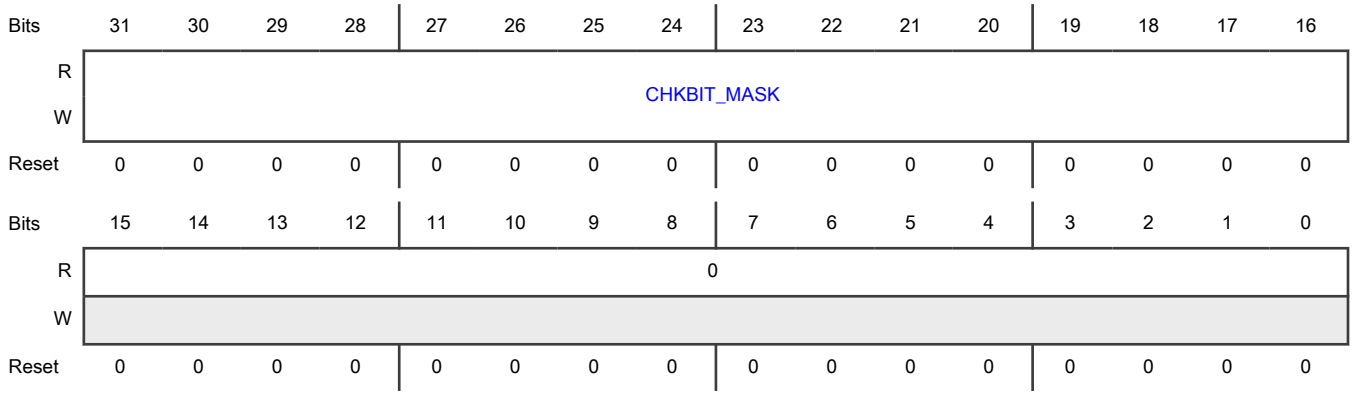
### Offset

| Register | Offset |
|---|---|
| EICHD4_WORD0 | 200h |

### Function

The first word of the Error Injection Channel Descriptor defines a left-justified mask field: CHKBIT_MASK. Each bit of CHKBIT_MASK specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified on read accesses. Successful write to this field clears the corresponding error injection channel valid bit, EICHEN[EICH*n*EN].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>CHKBIT_MASK | Checkbit Mask<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored.<br>For any unique details about the mapping of CHKBIT_MASK's bits to a channel's target RAM, see the chip-specific EIM information.<br>**NOTE**<br>Because CHKBIT_MASK is left-justified, the highest bit in the bit range is always in the position of the most significant bit. For CHKBIT_MASK[15:0] (16 bits wide), CHKBIT_MASK[15] is in the position of the most significant bit.<br>0b - The corresponding bit of the checkbit bus remains unmodified.<br>1b - The corresponding bit of the checkbit bus is inverted. |
| 15-0<br>— | Reserved |

## 47.6.10 Error Injection Channel Descriptor 4, Word1 (EICHD4_WORD1)

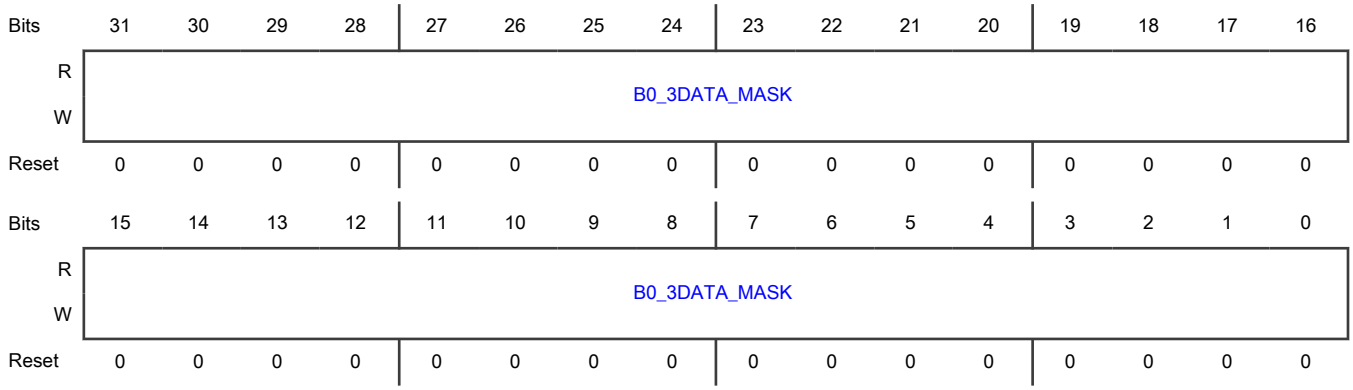
### Offset

| Register | Offset |
|---|---|
| EICHD4_WORD1 | 204h |

### Function

The second word of the Error Injection Channel Descriptor defines a right-justified mask field. The bits in B0_3DATA_MASK correspond to bytes 0–3 of the target bus. Each bit specifies whether the corresponding bit of the target bus should be inverted or remain unmodified on read accesses. A successful write to this field clears the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>B0_3DATA_MASK | Data Mask Bytes 0-3<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored.<br>**NOTE**<br>For the specific DATA_MASK bits to which B0_3DATA_MASK corresponds, See Error injection channel descriptor: DATA_MASK details.<br>0b - The corresponding bit of bytes 0-3 on the read data bus remains unmodified.<br>1b - The corresponding bit of bytes 0-3 on the read data bus is inverted. |

## 47.6.11 Error Injection Channel Descriptor n, Word3 (EICHD4_WORD3 - EICHD20_WORD3)

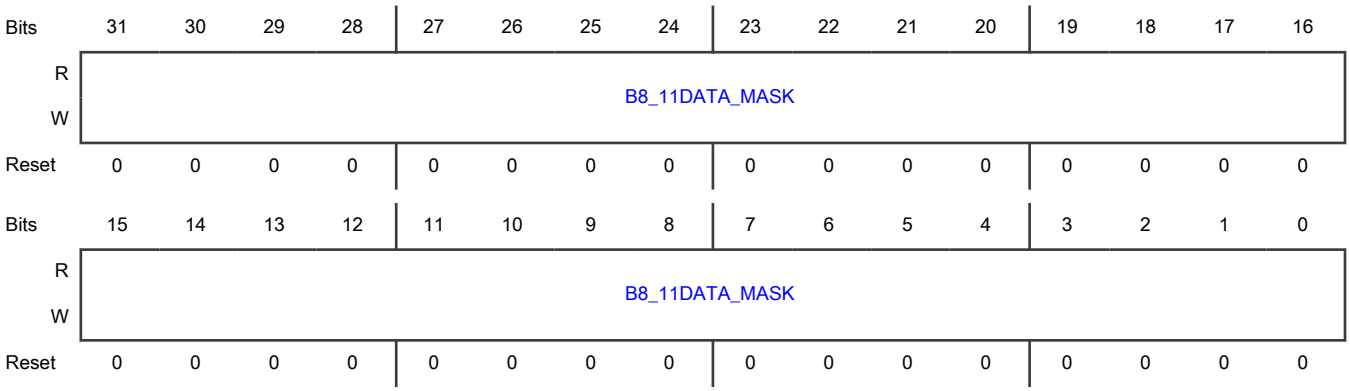
### Offset

| Register | Offset |
|---|---|
| EICHD4_WORD3 | 20Ch |
| EICHD5_WORD3 | 24Ch |
| EICHD6_WORD3 | 28Ch |
| EICHD7_WORD3 | 2CCh |
| EICHD9_WORD3 | 34Ch |
| EICHD10_WORD3 | 38Ch |
| EICHD11_WORD3 | 3CCh |
| EICHD12_WORD3 | 40Ch |
| EICHD19_WORD3 | 5CCh |
| EICHD20_WORD3 | 60Ch |

### Function

The fourth word of the Error Injection Channel Descriptor, when present, defines a right-justified mask field. The bits in B8_11DATA_MASK correspond to bytes 8–11 of the read data bus. Each bit specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified on read accesses. A successful write to this field clears the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>B8_11DATA_MASK | Data Mask Bytes 8-11<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified.<br>**NOTE**<br>For each channel: For the specific DATA_MASK bits to which B8_11DATA_MASK corresponds, See Error injection channel descriptor: DATA_MASK details.<br>0b - The corresponding bit of bytes 8-11 on the read data bus remains unmodified.<br>1b - The corresponding bit of bytes 8-11 on the read data bus is inverted. |

## 47.6.12 Error Injection Channel Descriptor n, Word4 (EICHD4_WORD4 - EICHD20_WORD4)

### Offset

| Register | Offset |
|---|---|
| EICHD4_WORD4 | 210h |
| EICHD5_WORD4 | 250h |
| EICHD6_WORD4 | 290h |
| EICHD7_WORD4 | 2D0h |
| EICHD9_WORD4 | 350h |
| EICHD10_WORD4 | 390h |

*Table continues on the next page...*

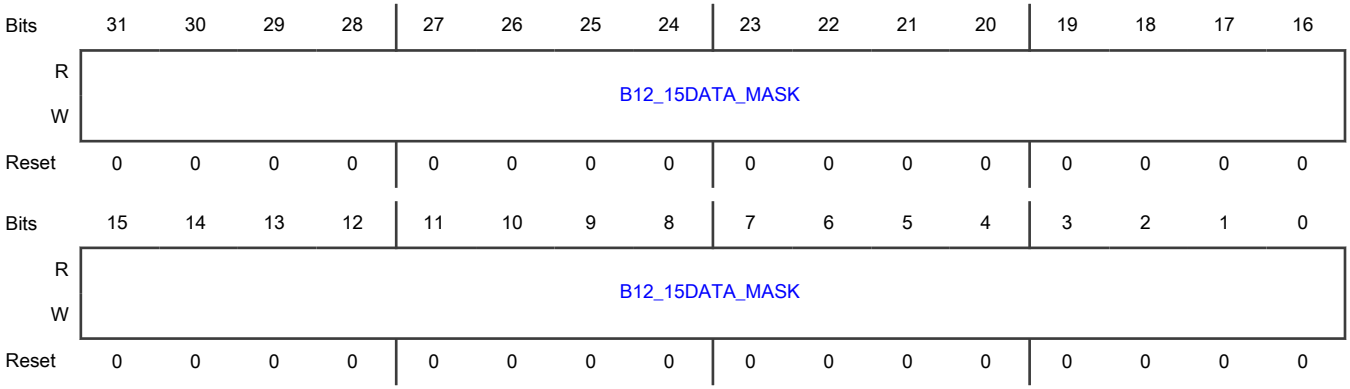
*Table continued from the previous page...*

| Register | Offset |
|---|---|
| EICHD11_WORD4 | 3D0h |
| EICHD12_WORD4 | 410h |
| EICHD19_WORD4 | 5D0h |
| EICHD20_WORD4 | 610h |

### Function

The fifth word of the Error Injection Channel Descriptor, when present, defines a right-justified mask field. The bits in B12_15DATA_MASK correspond to bytes 12–15 of the read data bus. Each bit specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified on read accesses. A successful write to this field clears the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>B12_15DATA_MASK | Data Mask Bytes 12-15<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified.<br>**NOTE**<br>For each channel: For the specific DATA_MASK bits to which B12_15DATA_MASK corresponds, See Error injection channel descriptor: DATA_MASK details.<br>0b - The corresponding bit of bytes 12-15 on the read data bus remains unmodified.<br>1b - The corresponding bit of bytes 12-15 on the read data bus is inverted. |

## 47.6.13 Error Injection Channel Descriptor n, Word0 (EICHD5_WORD0 - EICHD7_WORD0)

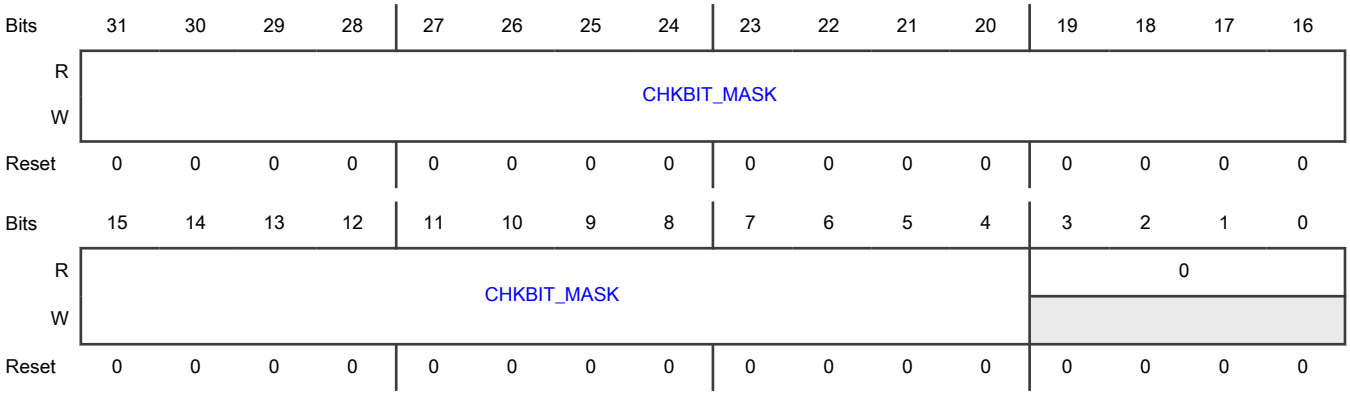
### Offset

| Register | Offset |
|---|---|
| EICHD5_WORD0 | 240h |
| EICHD6_WORD0 | 280h |
| EICHD7_WORD0 | 2C0h |

### Function

The first word of the Error Injection Channel Descriptor defines a left-justified mask field: CHKBIT_MASK. Each bit of CHKBIT_MASK specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified on read accesses. Successful write to this field clears the corresponding error injection channel valid bit, EICHEN[EICH*n*EN].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-4<br>CHKBIT_MASK | Checkbit Mask<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored.<br>For any unique details about the mapping of CHKBIT_MASK's bits to a channel's target RAM, see the chip-specific EIM information.<br>**NOTE**<br>Because CHKBIT_MASK is left-justified, the highest bit in the bit range is always in the position of the most significant bit. For CHKBIT_MASK[27:0] (28 bits wide), CHKBIT_MASK[27] is in the position of the most significant bit.<br>0b - The corresponding bit of the checkbit bus remains unmodified.<br>1b - The corresponding bit of the checkbit bus is inverted. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 3-0<br>— | Reserved |

## 47.6.14 Error Injection Channel Descriptor 5, Word1 (EICHD5_WORD1)

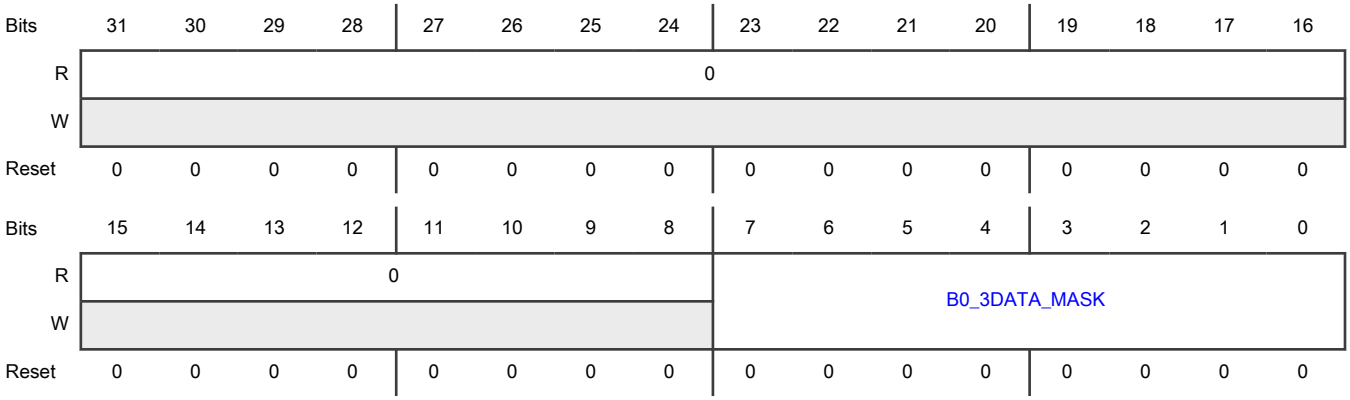
### Offset

| Register | Offset |
|---|---|
| EICHD5_WORD1 | 244h |

### Function

The second word of the Error Injection Channel Descriptor defines a right-justified mask field. The bits in B0_3DATA_MASK correspond to bytes 0–3 of the target bus. Each bit specifies whether the corresponding bit of the target bus should be inverted or remain unmodified on read accesses. A successful write to this field clears the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>B0_3DATA_MASK | Data Mask Bytes 0-3<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>For the specific DATA_MASK bits to which B0_3DATA_MASK corresponds, See Error injection channel descriptor: DATA_MASK details.<br>0b - The corresponding bit of bytes 0-3 on the read data bus remains unmodified.<br>1b - The corresponding bit of bytes 0-3 on the read data bus is inverted. |

## 47.6.15 Error Injection Channel Descriptor n, Word1 (EICHD6_WORD1 - EICHD7_WORD1)

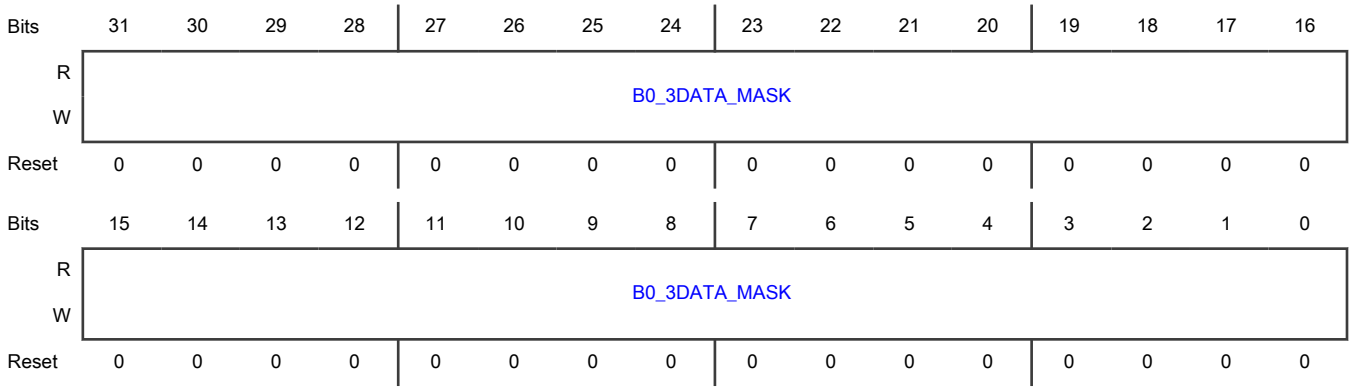
### Offset

| Register | Offset |
|---|---|
| EICHD6_WORD1 | 284h |
| EICHD7_WORD1 | 2C4h |

### Function

The second word of the Error Injection Channel Descriptor defines a right-justified mask field. The bits in B0_3DATA_MASK correspond to bytes 0–3 of the target bus. Each bit specifies whether the corresponding bit of the target bus should be inverted or remain unmodified on read accesses. A successful write to this field clears the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>B0_3DATA_MASK | Data Mask Bytes 0-3<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | **NOTE**<br>For the specific DATA_MASK bits to which B0_3DATA_MASK corresponds, See Error injection channel descriptor: DATA_MASK details.<br>0b - The corresponding bit of bytes 0-3 on the read data bus remains unmodified.<br>1b - The corresponding bit of bytes 0-3 on the read data bus is inverted. |

## 47.6.16 Error Injection Channel Descriptor 8, Word0 (EICHD8_WORD0)

### Offset

| Register | Offset |
|---|---|
| EICHD8_WORD0 | 300h |

### Function

The first word of the Error Injection Channel Descriptor defines a left-justified mask field: CHKBIT_MASK. Each bit of CHKBIT_MASK specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified on read accesses. Successful write to this field clears the corresponding error injection channel valid bit, EICHEN[EICH*n*EN].

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-18<br>CHKBIT_MASK | Checkbit Mask<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored.<br>For any unique details about the mapping of CHKBIT_MASK's bits to a channel's target RAM, see the chip-specific EIM information. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>Because CHKBIT_MASK is left-justified, the highest bit in the bit range is always in the position of the most significant bit. For CHKBIT_MASK[13:0] (14 bits wide), CHKBIT_MASK[13] is in the position of the most significant bit.<br>0b - The corresponding bit of the checkbit bus remains unmodified.<br>1b - The corresponding bit of the checkbit bus is inverted. |
| 17-0<br>— | Reserved |

## 47.6.17 Error Injection Channel Descriptor 8, Word1 (EICHD8_WORD1)

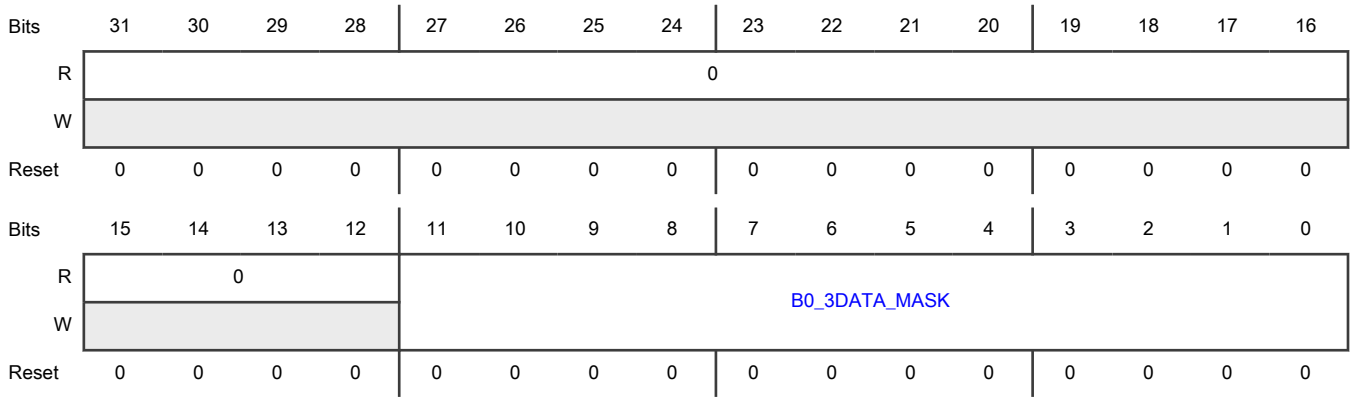
### Offset

| Register | Offset |
|---|---|
| EICHD8_WORD1 | 304h |

### Function

The second word of the Error Injection Channel Descriptor defines a right-justified mask field. The bits in B0_3DATA_MASK correspond to bytes 0–3 of the target bus. Each bit specifies whether the corresponding bit of the target bus should be inverted or remain unmodified on read accesses. A successful write to this field clears the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-12 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 11-0<br>B0_3DATA_MASK | Data Mask Bytes 0-3<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored.<br>**NOTE**<br>For the specific DATA_MASK bits to which B0_3DATA_MASK corresponds, See Error injection channel descriptor: DATA_MASK details.<br>0b - The corresponding bit of bytes 0-3 on the read data bus remains unmodified.<br>1b - The corresponding bit of bytes 0-3 on the read data bus is inverted. |

## 47.6.18 Error Injection Channel Descriptor 9, Word0 (EICHD9_WORD0)

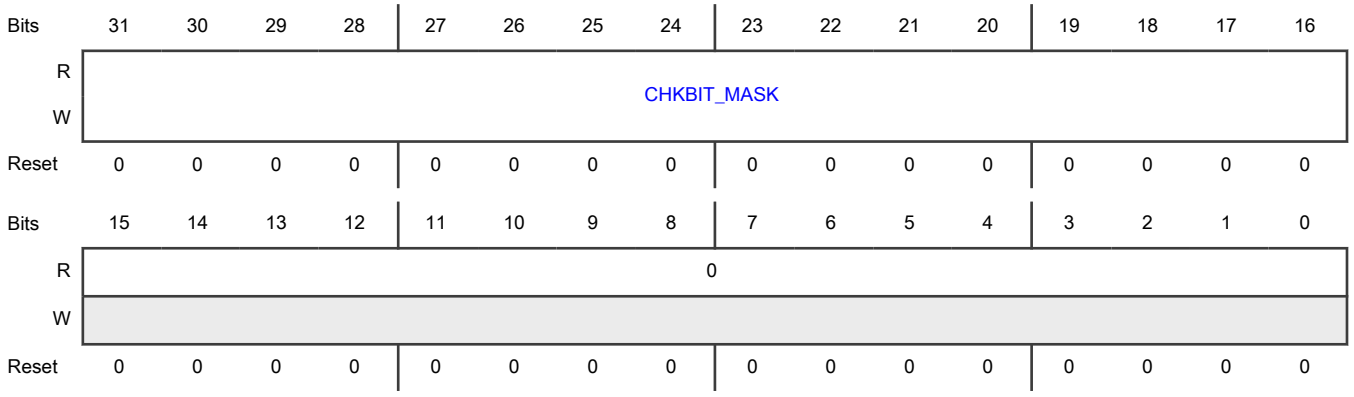
### Offset

| Register | Offset |
|---|---|
| EICHD9_WORD0 | 340h |

### Function

The first word of the Error Injection Channel Descriptor defines a left-justified mask field: CHKBIT_MASK. Each bit of CHKBIT_MASK specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified on read accesses. Successful write to this field clears the corresponding error injection channel valid bit, EICHEN[EICH*n*EN].

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-16<br>CHKBIT_MASK | Checkbit Mask<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored.<br>For any unique details about the mapping of CHKBIT_MASK's bits to a channel's target RAM, see the chip-specific EIM information.<br>**NOTE**<br>Because CHKBIT_MASK is left-justified, the highest bit in the bit range is always in the position of the most significant bit. For CHKBIT_MASK[15:0] (16 bits wide), CHKBIT_MASK[15] is in the position of the most significant bit.<br>0b - The corresponding bit of the checkbit bus remains unmodified.<br>1b - The corresponding bit of the checkbit bus is inverted. |
| 15-0<br>— | Reserved |

## 47.6.19 Error Injection Channel Descriptor 9, Word1 (EICHD9_WORD1)

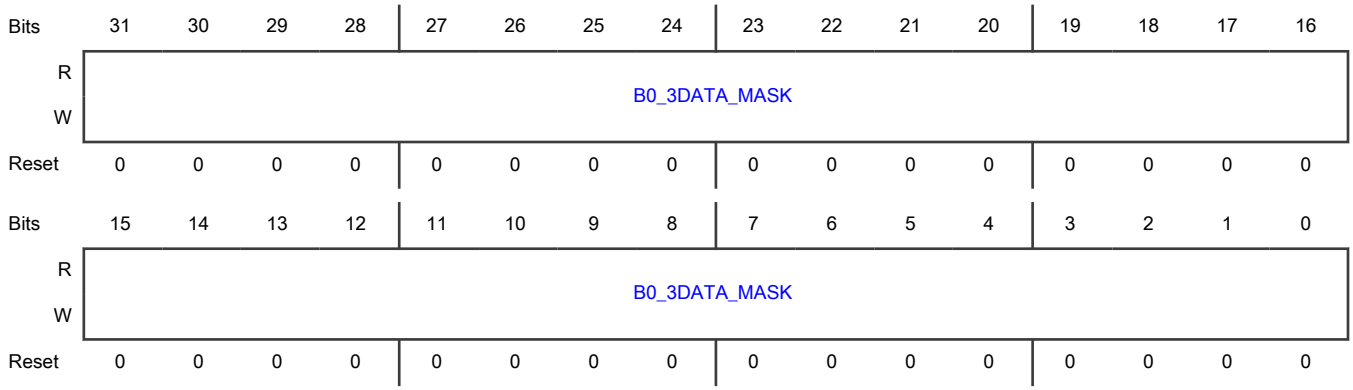
### Offset

| Register | Offset |
|---|---|
| EICHD9_WORD1 | 344h |

### Function

The second word of the Error Injection Channel Descriptor defines a right-justified mask field. The bits in B0_3DATA_MASK correspond to bytes 0–3 of the target bus. Each bit specifies whether the corresponding bit of the target bus should be inverted or remain unmodified on read accesses. A successful write to this field clears the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-0<br>B0_3DATA_MASK | Data Mask Bytes 0-3<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored.<br>**NOTE**<br>For the specific DATA_MASK bits to which B0_3DATA_MASK corresponds, See Error injection channel descriptor: DATA_MASK details.<br>0b - The corresponding bit of bytes 0-3 on the read data bus remains unmodified.<br>1b - The corresponding bit of bytes 0-3 on the read data bus is inverted. |

## 47.6.20 Error Injection Channel Descriptor n, Word0 (EICHD10_WORD0 - EICHD12_WORD0)

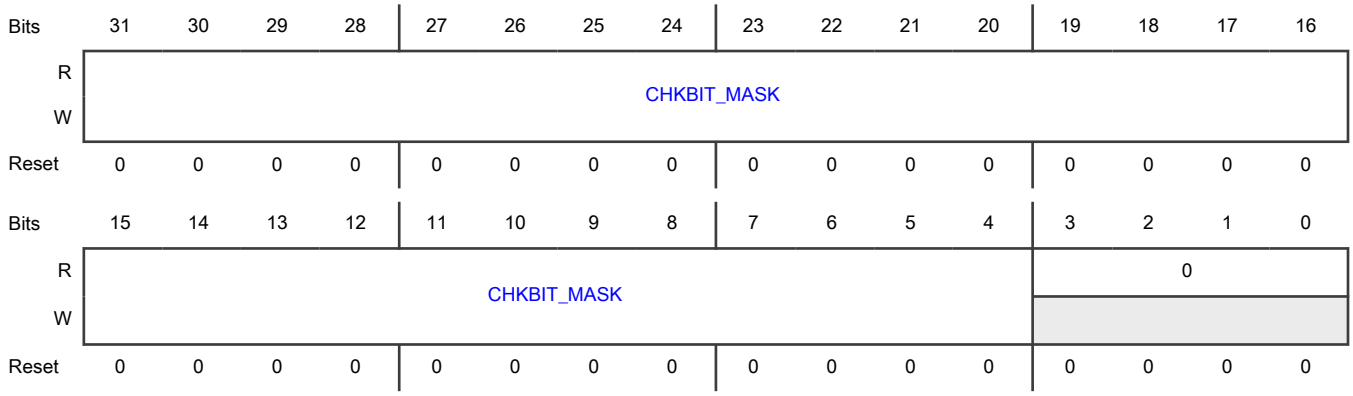
### Offset

| Register | Offset |
|---|---|
| EICHD10_WORD0 | 380h |
| EICHD11_WORD0 | 3C0h |
| EICHD12_WORD0 | 400h |

### Function

The first word of the Error Injection Channel Descriptor defines a left-justified mask field: CHKBIT_MASK. Each bit of CHKBIT_MASK specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified on read accesses. Successful write to this field clears the corresponding error injection channel valid bit, EICHEN[EICH*n*EN].

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-4<br>CHKBIT_MASK | Checkbit Mask<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored.<br>For any unique details about the mapping of CHKBIT_MASK's bits to a channel's target RAM, see the chip-specific EIM information.<br>**NOTE**<br>Because CHKBIT_MASK is left-justified, the highest bit in the bit range is always in the position of the most significant bit. For CHKBIT_MASK[27:0] (28 bits wide), CHKBIT_MASK[27] is in the position of the most significant bit.<br>0b - The corresponding bit of the checkbit bus remains unmodified.<br>1b - The corresponding bit of the checkbit bus is inverted. |
| 3-0<br>— | Reserved |

## 47.6.21 Error Injection Channel Descriptor 10, Word1 (EICHD10_WORD1)

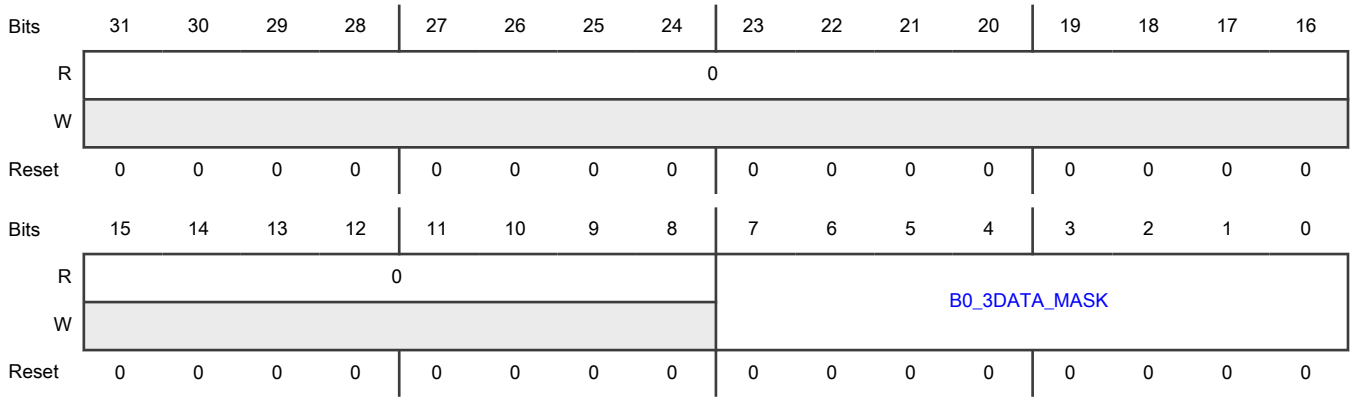
### Offset

| Register | Offset |
|---|---|
| EICHD10_WORD1 | 384h |

### Function

The second word of the Error Injection Channel Descriptor defines a right-justified mask field. The bits in B0_3DATA_MASK correspond to bytes 0–3 of the target bus. Each bit specifies whether the corresponding bit of the target bus should be inverted or remain unmodified on read accesses. A successful write to this field clears the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

### Diagram



**Fields**

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>B0_3DATA_MASK | Data Mask Bytes 0-3<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored.<br>**NOTE**<br>For the specific DATA_MASK bits to which B0_3DATA_MASK corresponds, See Error injection channel descriptor: DATA_MASK details.<br>0b - The corresponding bit of bytes 0-3 on the read data bus remains unmodified.<br>1b - The corresponding bit of bytes 0-3 on the read data bus is inverted. |

## 47.6.22 Error Injection Channel Descriptor n, Word1 (EICHD11_WORD1 - EICHD18_WORD1)

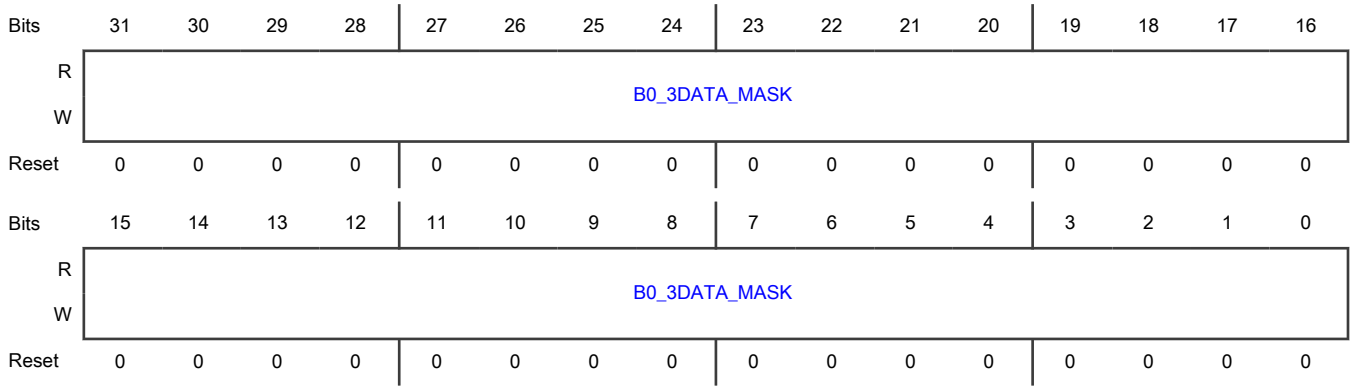
**Offset**

| Register | Offset |
|---|---|
| EICHD11_WORD1 | 3C4h |
| EICHD12_WORD1 | 404h |
| EICHD13_WORD1 | 444h |
| EICHD14_WORD1 | 484h |
| EICHD15_WORD1 | 4C4h |
| EICHD16_WORD1 | 504h |
| EICHD17_WORD1 | 544h |
| EICHD18_WORD1 | 584h |

**Function**

The second word of the Error Injection Channel Descriptor defines a right-justified mask field. The bits in B0_3DATA_MASK correspond to bytes 0–3 of the target bus. Each bit specifies whether the corresponding bit of the target bus should be inverted or remain unmodified on read accesses. A successful write to this field clears the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>B0_3DATA_MASK | Data Mask Bytes 0-3<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored.<br>**NOTE**<br>For the specific DATA_MASK bits to which B0_3DATA_MASK corresponds, See Error injection channel descriptor: DATA_MASK details.<br>0b - The corresponding bit of bytes 0-3 on the read data bus remains unmodified.<br>1b - The corresponding bit of bytes 0-3 on the read data bus is inverted. |

## 47.6.23 Error Injection Channel Descriptor n, Word0 (EICHD13_WORD0 - EICHD18_WORD0)

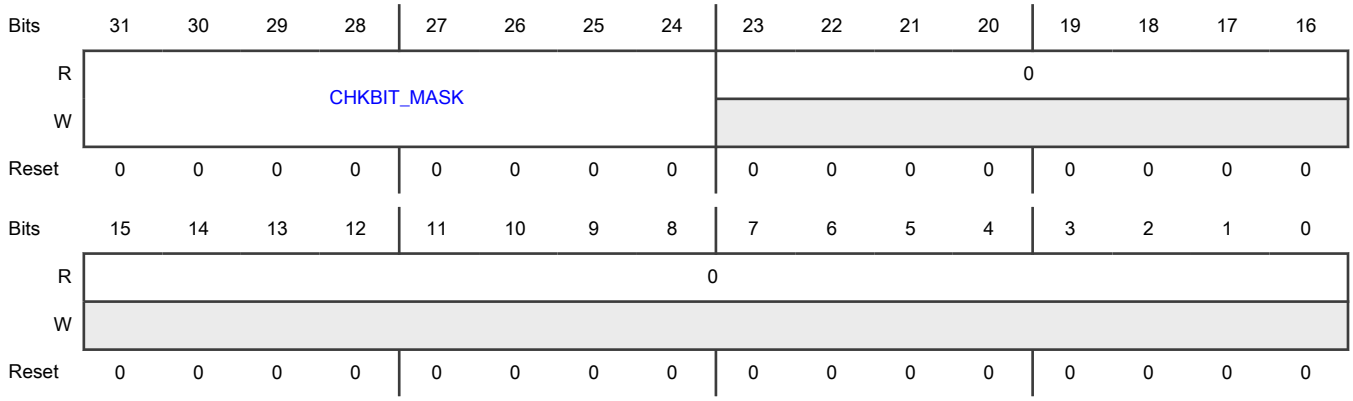
### Offset

| Register | Offset |
|---|---|
| EICHD13_WORD0 | 440h |
| EICHD14_WORD0 | 480h |
| EICHD15_WORD0 | 4C0h |
| EICHD16_WORD0 | 500h |
| EICHD17_WORD0 | 540h |
| EICHD18_WORD0 | 580h |

### Function

The first word of the Error Injection Channel Descriptor defines a left-justified mask field: CHKBIT_MASK. Each bit of CHKBIT_MASK specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified on read accesses. Successful write to this field clears the corresponding error injection channel valid bit, EICHEN[EICH*n*EN].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>CHKBIT_MASK | Checkbit Mask<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the checkbit bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored.<br>For any unique details about the mapping of CHKBIT_MASK's bits to a channel's target RAM, see the chip-specific EIM information.<br>**NOTE**<br>Because CHKBIT_MASK is left-justified, the highest bit in the bit range is always in the position of the most significant bit. For CHKBIT_MASK[7:0] (8 bits wide), CHKBIT_MASK[7] is in the position of the most significant bit.<br>0b - The corresponding bit of the checkbit bus remains unmodified.<br>1b - The corresponding bit of the checkbit bus is inverted. |
| 23-0<br>— | Reserved |

## 47.6.24 Error Injection Channel Descriptor n, Word1 (EICHD19_WORD1 - EICHD26_WORD1)

### Offset

| Register | Offset |
|---|---|
| EICHD19_WORD1 | 5C4h |
| EICHD20_WORD1 | 604h |
| EICHD21_WORD1 | 644h |
| EICHD22_WORD1 | 684h |
| EICHD23_WORD1 | 6C4h |

*Table continues on the next page...*

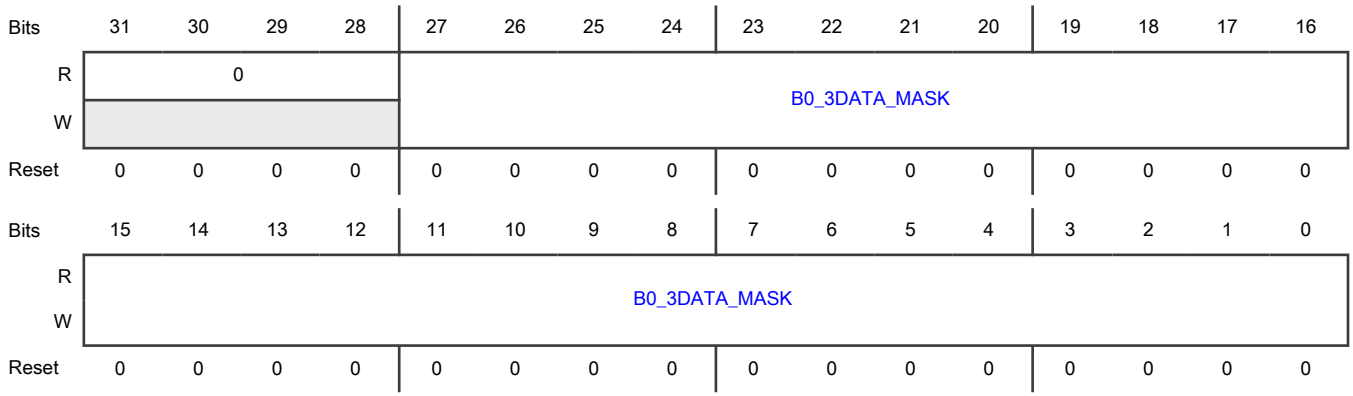
*Table continued from the previous page...*

| Register | Offset |
|---|---|
| EICHD24_WORD1 | 704h |
| EICHD25_WORD1 | 744h |
| EICHD26_WORD1 | 784h |

### Function

The second word of the Error Injection Channel Descriptor defines a right-justified mask field. The bits in B0_3DATA_MASK correspond to bytes 0–3 of the target bus. Each bit specifies whether the corresponding bit of the target bus should be inverted or remain unmodified on read accesses. A successful write to this field clears the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-0<br>B0_3DATA_MASK | Data Mask Bytes 0-3<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored.<br>**NOTE**<br>For the specific DATA_MASK bits to which B0_3DATA_MASK corresponds, See Error injection channel descriptor: DATA_MASK details.<br>0b - The corresponding bit of bytes 0-3 on the read data bus remains unmodified.<br>1b - The corresponding bit of bytes 0-3 on the read data bus is inverted. |

## 47.6.25 Error Injection Channel Descriptor n, Word5 (EICHD19_WORD5 - EICHD20_WORD5)

### Offset

| Register | Offset |
|---|---|
| EICHD19_WORD5 | 5D4h |
| EICHD20_WORD5 | 614h |

### Function

The sixth word of the Error Injection Channel Descriptor, when present, defines a right-justified mask field. The bits in B16_19DATA_MASK correspond to bytes 16–19 of the read data bus. Each bit specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified on read accesses. A successful write to this field clears the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | B16_19DATA_MASK | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | B16_19DATA_MASK | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>B16_19DATA_MASK | Data Mask Bytes 16-19<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified.<br>**NOTE**<br>For each channel: For the specific DATA_MASK bits to which B16_19DATA_MASK corresponds, See Error injection channel descriptor: DATA_MASK details.<br>0b - The corresponding bit of bytes 16-19 on the read data bus remains unmodified.<br>1b - The corresponding bit of bytes 16-19 on the read data bus is inverted. |

## 47.6.26 Error Injection Channel Descriptor n, Word6 (EICHD19_WORD6 - EICHD20_WORD6)

### Offset

| Register | Offset |
|---|---|
| EICHD19_WORD6 | 5D8h |
| EICHD20_WORD6 | 618h |

### Function

The seventh word of the Error Injection Channel Descriptor, when present, defines a right-justified mask field. The bits in B20_23DATA_MASK correspond to bytes 20–23 of the read data bus. Each bit specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified on read accesses. A successful write to this field clears the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | B20_23DATA_MASK | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | B20_23DATA_MASK | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>B20_23DATA_MASK | Data Mask Bytes 20-23<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified.<br>**NOTE**<br>For each channel: For the specific DATA_MASK bits to which B20_23DATA_MASK corresponds, See Error injection channel descriptor: DATA_MASK details.<br>0b - The corresponding bit of bytes 20-23 on the read data bus remains unmodified.<br>1b - The corresponding bit of bytes 20-23 on the read data bus is inverted. |

## 47.6.27 Error Injection Channel Descriptor 27, Word1 (EICHD27_WORD1)

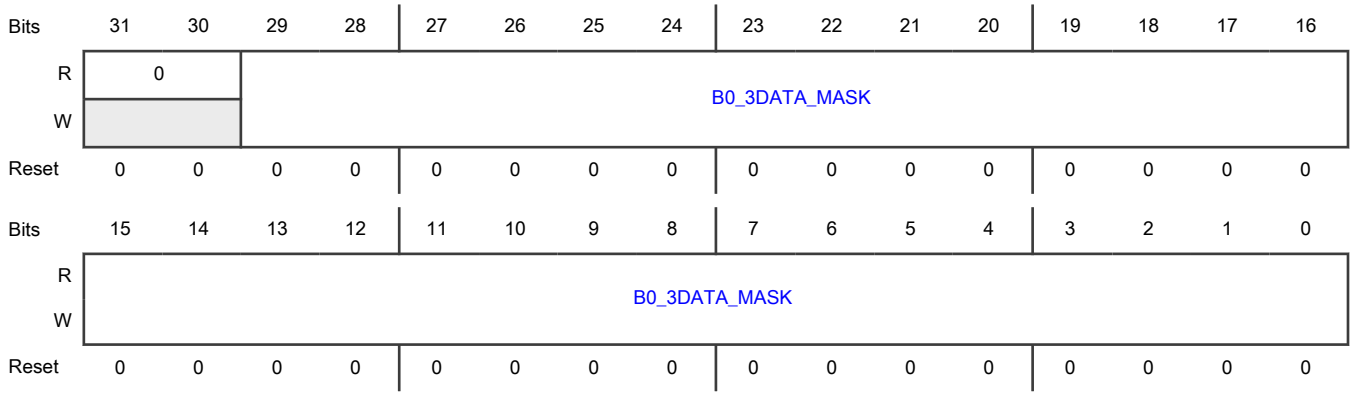
### Offset

| Register | Offset |
|---|---|
| EICHD27_WORD1 | 7C4h |

### Function

The second word of the Error Injection Channel Descriptor defines a right-justified mask field. The bits in B0_3DATA_MASK correspond to bytes 0–3 of the target bus. Each bit specifies whether the corresponding bit of the target bus should be inverted or remain unmodified on read accesses. A successful write to this field clears the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-0<br>B0_3DATA_MASK | Data Mask Bytes 0-3<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored.<br>**NOTE**<br>For the specific DATA_MASK bits to which B0_3DATA_MASK corresponds, See Error injection channel descriptor: DATA_MASK details.<br>0b - The corresponding bit of bytes 0-3 on the read data bus remains unmodified.<br>1b - The corresponding bit of bytes 0-3 on the read data bus is inverted. |

## 47.6.28 Error Injection Channel Descriptor 28, Word1 (EICHD28_WORD1)

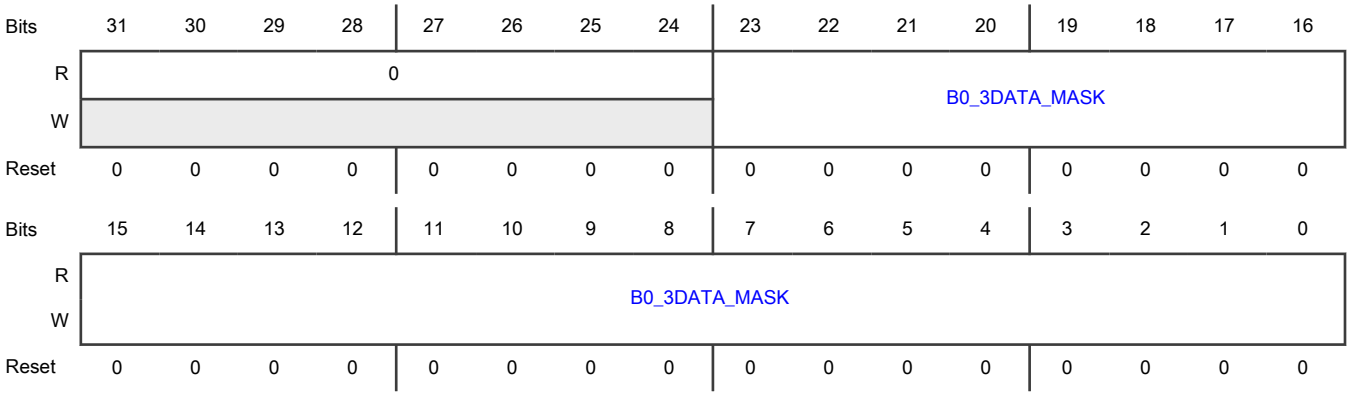
### Offset

| Register | Offset |
|---|---|
| EICHD28_WORD1 | 804h |

### Function

The second word of the Error Injection Channel Descriptor defines a right-justified mask field. The bits in B0_3DATA_MASK correspond to bytes 0–3 of the target bus. Each bit specifies whether the corresponding bit of the target bus should be inverted or remain unmodified on read accesses. A successful write to this field clears the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>B0_3DATA_MASK | Data Mask Bytes 0-3<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored.<br>**NOTE**<br>For the specific DATA_MASK bits to which B0_3DATA_MASK corresponds, See Error injection channel descriptor: DATA_MASK details.<br>0b - The corresponding bit of bytes 0-3 on the read data bus remains unmodified.<br>1b - The corresponding bit of bytes 0-3 on the read data bus is inverted. |

## 47.6.29 Error Injection Channel Descriptor n, Word1 (EICHD29_WORD1 - EICHD30_WORD1)

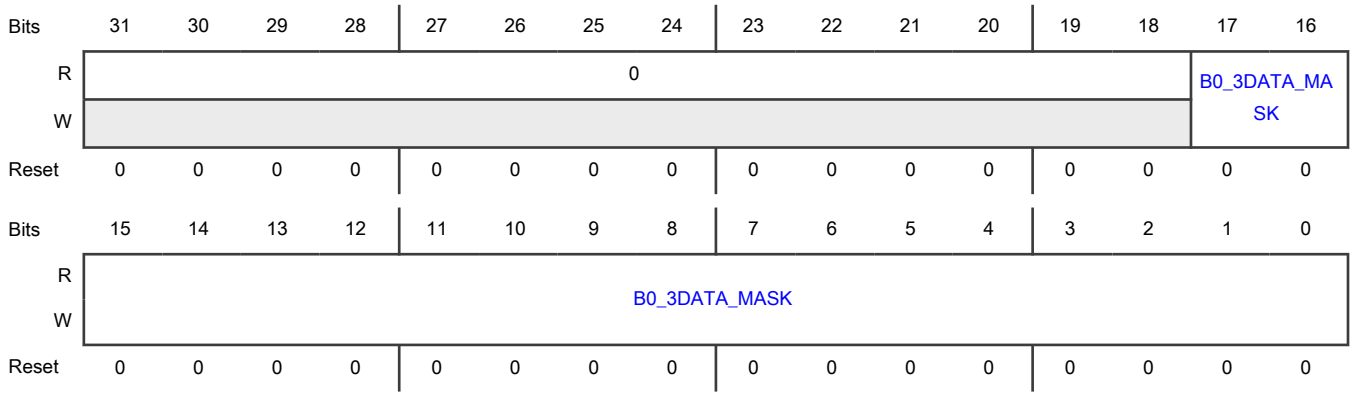
### Offset

| Register | Offset |
|---|---|
| EICHD29_WORD1 | 844h |
| EICHD30_WORD1 | 884h |

### Function

The second word of the Error Injection Channel Descriptor defines a right-justified mask field. The bits in B0_3DATA_MASK correspond to bytes 0–3 of the target bus. Each bit specifies whether the corresponding bit of the target bus should be inverted or remain unmodified on read accesses. A successful write to this field clears the corresponding error injection channel valid field, EICHEN[EICH*n*EN].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-18<br>— | Reserved |
| 17-0<br>B0_3DATA_MASK | Data Mask Bytes 0-3<br>This field defines a bit-mapped mask that specifies whether the corresponding bit of the read data bus from the target RAM should be inverted or remain unmodified. Writes to unimplemented bits are ignored.<br>**NOTE**<br>For the specific DATA_MASK bits to which B0_3DATA_MASK corresponds, See Error injection channel descriptor: DATA_MASK details.<br>0b - The corresponding bit of bytes 0-3 on the read data bus remains unmodified.<br>1b - The corresponding bit of bytes 0-3 on the read data bus is inverted. |

## 47.7 Glossary

| | |
|---|---|
| **SEC-DED** | Single error correction – double error detection |
| **ECC** | Error correction code |

# Chapter 48
# Error Reporting Module (ERM)

## 48.1 Chip-specific ERM information

### 48.1.1 ERM channel mapping

ERM provides information on memory events associated with ECC and parity. It also provides you an option to enable interrupt notification for these events.

**Table 225. ERM channel mapping**

| Channel # | Module | Captured status |
|---|---|---|
| 00 | SRAM0 | Single-bit error, multi-bit error, syndrome, absolute error address aligned to double-word (64-bit) boundary |
| 01 | SRAM1 [1] | Single-bit error, multi-bit error, syndrome, absolute error address+18000h aligned to double-word (64-bit) boundary[2] |
| 02 | Cortex-M7_0 I-cache tag RAM | Single-bit error, multi-bit error[3] |
| 03 | Cortex-M7_0 I-cache data RAM | Single-bit error, multi-bit error[3] |
| 04 | Cortex-M7_0 D-cache tag RAM | Single-bit error, multi-bit error[3] |
| 05 | Cortex-M7_0 D-cache data RAM | Single-bit error, multi-bit error[3] |
| 10 | Cortex-M7_0 ITCM | Single-bit error, multi-bit error, syndrome, offset error address |
| 11 | Cortex-M7_0 D0TCM | Single-bit error, multi-bit error, syndrome, offset error address |
| 12 | Cortex-M7_0 D1TCM | Single-bit error, multi-bit error, syndrome, offset error address[4] |
| 13 | Cortex-M7_1 ITCM | Single-bit error, multi-bit error, syndrome, offset error address |
| 14 | Cortex-M7_1 D0TCM | Single-bit error, multi-bit error, syndrome, offset error address |
| 15 | Cortex-M7_1 D1TCM | Single-bit error, multi-bit error, syndrome, offset error address[4] |
| 16 | DMA TCD | Single-bit error, multi-bit error, syndrome, offset error address[5] |
| 17 | Flash memory port p0 | Single-bit error, multi-bit error, absolute error address |
| 18 | Flash memory port p1 | Single-bit error, multi-bit error, absolute error address |
| 19 | Flash memory port p2 [6] | Single-bit error, multi-bit error, absolute error address |

1. SRAM1 is not available for the MCXE315/MCXE316/MCXE317 product variants.
2. The size of SRAM0 is 160 KB, and therefore, to align addresses to the next power of 2, an address space of 256 KB is reserved for error reporting. However, SRAM0 and SRAM1 are in contiguous locations in the memory map. So, you must subtract 18000h (256 KB-160 KB = 96 KB that corresponds to 18000h) from the reported address to get an absolute address. See the "Memory and Memory Interfaces" chapter for SRAM details on different MCXE31 variants.
3. The cache controller does not report error addresses and syndrome.
4. For address reporting, bit 2 of the address is masked because this bit decides whether the access (read or write) is for D0TCM or D1TCM. For example, if the offset address is 0h or 8h, it is routed to D0TCM, but if the offset address is 4h or Ch, it is routed to D1TCM. The errors that are latched for these offset addresses are as follows:
   - Offset 0h : Address 0h is latched into the ERM channel corresponding to D0TCM.
   - Offset 4h : Address 0h is latched into the ERM channel corresponding to D1TCM.
   - Offset 8h : Address 8h is latched into the ERM channel corresponding to D0TCM.
   - Offset Ch : Address 8h is latched into the ERM channel corresponding to D1TCM.

5. Bits [31:10] and [2:0] are always 0 because they are not connected.

   Bits [9:5] indicate the corresponding TCD out of all the 32 implemented TCDs.

   Bits [4:3] indicate an offset of TCDs with respect to a 64-bit aligned boundary.

   - If [4:3] is 00, it indicates that an error is on offset address 20h.
   - If [4:3] is 01, it indicates that an error is on offset address 28h.
   - If [4:3] is 10, it indicates that an error is on offset address 30h.
   - If [4:3] is 11, it indicates that an error is on offset address 38h.
6. Flash memory port p2 is not applicable for the MCXE315/MCXE316/MCXE317 product variants.

## 48.2 Overview

The Error Reporting Module (ERM) provides information and optional interrupt notification on memory error events associated with ECC and parity. The ERM collects error events on memory accesses for memory arrays, such as flash memory, system RAM, or peripheral RAMs. ERM supports various channels for memory sources where each ERM channel is associated with a different memory module. See the chip-specific ERM information for details about supported memory sources and specific memory channel assignments. If memory supports ECC then ERM syndrome and error address information is captured along with error event. ERM does not receive this information in case of cache or memory with parity along with error event.

### 48.2.1 Features

The ERM includes these features:

- Optional interrupt notification on captured error events
- Capturing of address and syndrome information on single-bit correction and non-correctable ECC events
- Support for error event capturing for memory sources, with individual reporting fields and interrupt configuration per memory channel
- Recording the count value of the number of corrected error events

### 48.2.2 Block diagram

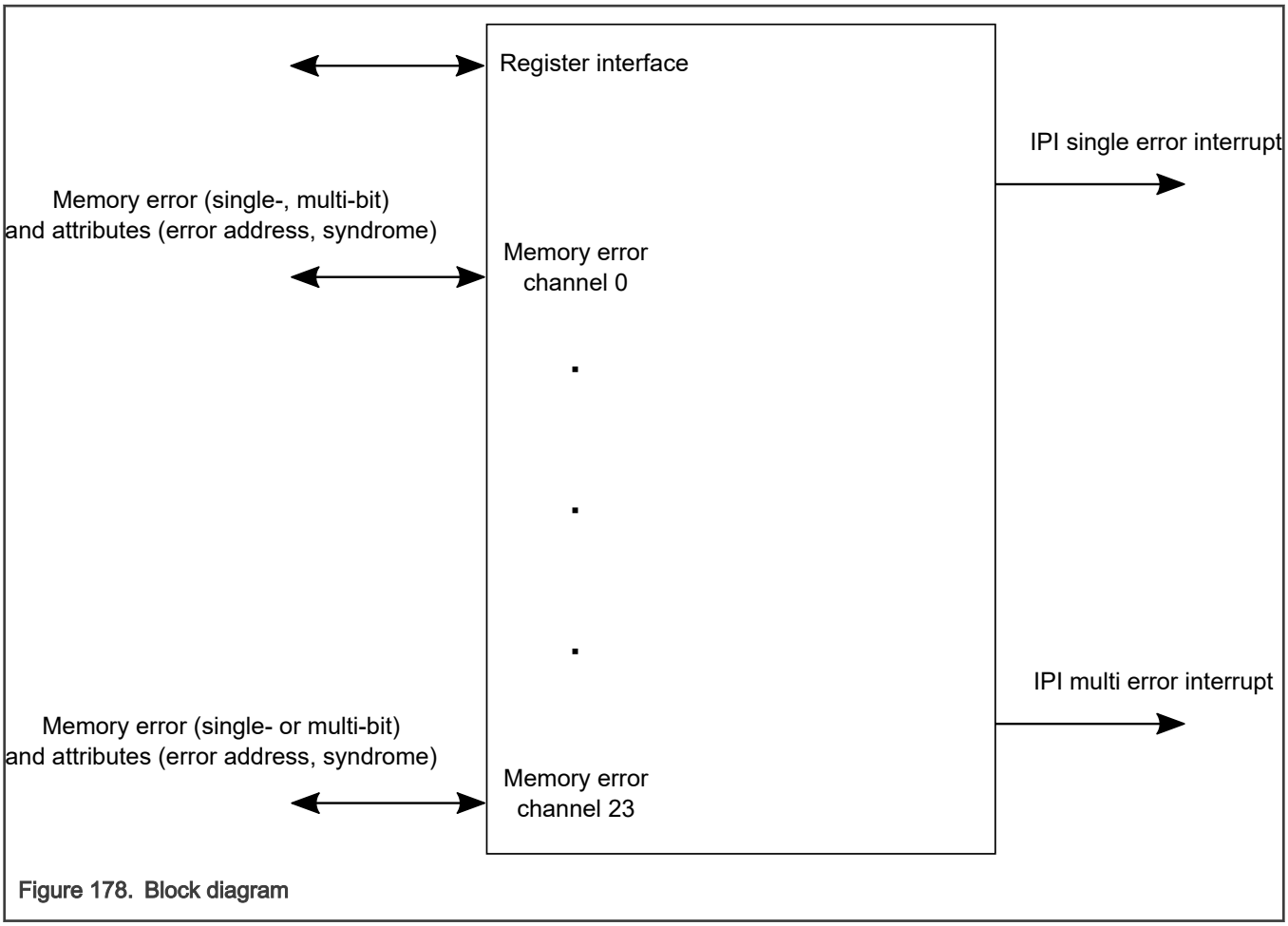



Figure 178. Block diagram

## 48.3 Functional description

### 48.3.1 Single-bit correction events

When a single-bit correction event on Memory *n* is detected, the ERM:

- Records the event by changing the value of the applicable Status Register bit SR*x*[SBC*n*] to 1.
- Increments the correctable error count value (until the counter reaches its maximum value): CORR_ERR_CNT*n*[COUNT].
- Records the corresponding access address that initiated the event in the Memory *n* Error Address Register: EAR*n* (if this register is present for the channel).
- Stores the corresponding ECC syndrome in the Memory *n* Error Syndrome Register: SYN*n* (if this register is present for the channel). This register identifies the bit position of the corrected data on single-bit data inversion.

The ERM holds event information only for the last reported event.

To clear the record of an event, write 1 to SR*x*[SBC*n*] to change its value to 0.

To reset the correctable error count value, write all zeros to CORR_ERR_CNT*n*[COUNT].

#### Optional interrupt notification for single-bit correction events

The ERM provides an option to generate an interrupt notification upon the report of a single-bit correction event. To enable single-bit correction interrupts for a channel:

1. To enable interrupt notification for single-bit correction events on Memory *n*, set CR*x*[ESCIE*n*] to 1.
2. Subsequently, a single-bit correction event on Memory *n* is detected, the ERM:
   - Records the event and address, and stores the ECC syndrome as usual.
   - Additionally sends an interrupt notification corresponding to the event.
3. To clear both the record of an event and the corresponding interrupt notification, write 1 to SR*x*[SBC*n*] to change its value to 0.

### 48.3.2 Non-correctable error events

When a non-correctable ECC error event on Memory *n* is detected, the ERM:

- Records the event by changing the value of the applicable Status Register bit: SR*x*[NCE*n*] to 1.
- Records the corresponding access address that initiated the event in the Memory *n* Error Address Register: EAR*n* (if this register is present for the channel).
- Stores the corresponding ECC syndrome in the Memory *n* Error Syndrome Register: SYN*n* (if this register is present for the channel).
  - In the event of a non-correctable address bit inversion, SYN*n* identifies the pertinent address bit position.
  - In the event of a non-correctable, multi-bit data inversion, the syndrome value does not provide any additional diagnostic information.

The ERM holds event information only for the last reported event.

To clear the record of an event, write 1 to SR*x*[NCE*n*] to change its value to 0.

#### Optional interrupt notification for non-correctable error events

The ERM provides an option to generate an interrupt notification upon the report of a non-correctable ECC event. To enable non-correctable error interrupts for a channel:

1. To enable interrupt notifications for non-correctable error events on Memory *n*, set CR*x*[ENCIE*n*] to 1.
2. Subsequently, when a non-correctable error event on Memory *n* is detected, the ERM:
   - Records the event and address and stores the ECC syndrome as usual.
   - Additionally sends an interrupt notification corresponding to the event.
3. To clear both the record of an event and the corresponding interrupt notification, write 1 to SR*x*[NCE*n*] to change its value to 0.

**NOTE**

Parity errors can be mapped to non-correctable errors where error attributes like SYNDROME, ADDRESS are not provided.

## 48.4 Initialization

For each ERM channel supporting memory with ECC, prepare the corresponding memory array before enabling ERM interrupts about errors for that memory.

1. Initialize the memory to a known value so that the correct corresponding ECC codeword is stored.
2. During the memory's initialization, if the ERM captures information about any ECC error event, clear the corresponding SR*x*[SBC*n*] or SR*x*[NCE*n*] field that stores the record of the event.
3. Program the applicable CR*x*[ESCIE*n*] and CR*x*[ENCIE*n*] fields to enable ERM interrupts as desired.

## 48.5 ERM register descriptions

You can access the programming model:

- Only in supervisor mode
- Using only 32-bit (word) accesses

Any of the following attempted references to the programming model generates an error termination:

- In user mode
- Using non-32-bit access sizes

Based on the design implementation, the following XFR error behavior is evident at the IPS interface.

- Within the ERM memory map, an XFR error is evident at reserved addresses from location 20h to FFh.
- No XFR error is evident at reserved addresses in memory spaces allocated to each channel. For example: For channel 0, for read/write accesses to reserved address 10Ch, the XFR error is 0.
- For accesses to locations beyond the addresses allocated for the final channel, the XFR error is 1.

**NOTE**

- See the chip-specific ERM information at the beginning of this chapter for details on Memory channel mapping.
- To access the channel registers, corresponding memory channel clock must be enabled.

### 48.5.1 ERM memory map

ERM base address: 4025_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | ERM Configuration Register 0 (CR0) | 32 | RW | 0000_0000h |
| 4h | ERM Configuration Register 1 (CR1) | 32 | RW | 0000_0000h |
| 8h | ERM Configuration Register 2 (CR2) | 32 | RW | 0000_0000h |
| 10h | ERM Status Register 0 (SR0) | 32 | RW | 0000_0000h |
| 14h | ERM Status Register 1 (SR1) | 32 | RW | 0000_0000h |
| 18h | ERM Status Register 2 (SR2) | 32 | RW | 0000_0000h |
| 100h | ERM Memory 0 Error Address Register (EAR0) | 32 | R | 0000_0000h |
| 104h | ERM Memory 0 Syndrome Register (SYN0) | 32 | R | 0000_0000h |
| 108h | ERM Memory 0 Correctable Error Count Register (CORR_ERR_CNT0) | 32 | RW | 0000_0000h |
| 110h | ERM Memory 1 Error Address Register (EAR1) | 32 | R | 0000_0000h |
| 114h | ERM Memory 1 Syndrome Register (SYN1) | 32 | R | 0000_0000h |
| 118h | ERM Memory 1 Correctable Error Count Register (CORR_ERR_CNT1) | 32 | RW | 0000_0000h |
| 128h | ERM Memory 2 Correctable Error Count Register (CORR_ERR_CNT2) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 138h | ERM Memory 3 Correctable Error Count Register (CORR_ERR_CNT3) | 32 | RW | 0000_0000h |
| 148h | ERM Memory 4 Correctable Error Count Register (CORR_ERR_CNT4) | 32 | RW | 0000_0000h |
| 158h | ERM Memory 5 Correctable Error Count Register (CORR_ERR_CNT5) | 32 | RW | 0000_0000h |
| 168h | ERM Memory 6 Correctable Error Count Register (CORR_ERR_CNT6) | 32 | RW | 0000_0000h |
| 178h | ERM Memory 7 Correctable Error Count Register (CORR_ERR_CNT7) | 32 | RW | 0000_0000h |
| 188h | ERM Memory 8 Correctable Error Count Register (CORR_ERR_CNT8) | 32 | RW | 0000_0000h |
| 198h | ERM Memory 9 Correctable Error Count Register (CORR_ERR_CNT9) | 32 | RW | 0000_0000h |
| 1A0h | ERM Memory 10 Error Address Register (EAR10) | 32 | R | 0000_0000h |
| 1A4h | ERM Memory 10 Syndrome Register (SYN10) | 32 | R | 0000_0000h |
| 1A8h | ERM Memory 10 Correctable Error Count Register (CORR_ERR_CNT10) | 32 | RW | 0000_0000h |
| 1B0h | ERM Memory 11 Error Address Register (EAR11) | 32 | R | 0000_0000h |
| 1B4h | ERM Memory 11 Syndrome Register (SYN11) | 32 | R | 0000_0000h |
| 1B8h | ERM Memory 11 Correctable Error Count Register (CORR_ERR_CNT11) | 32 | RW | 0000_0000h |
| 1C0h | ERM Memory 12 Error Address Register (EAR12) | 32 | R | 0000_0000h |
| 1C4h | ERM Memory 12 Syndrome Register (SYN12) | 32 | R | 0000_0000h |
| 1C8h | ERM Memory 12 Correctable Error Count Register (CORR_ERR_CNT12) | 32 | RW | 0000_0000h |
| 1D0h | ERM Memory 13 Error Address Register (EAR13) | 32 | R | 0000_0000h |
| 1D4h | ERM Memory 13 Syndrome Register (SYN13) | 32 | R | 0000_0000h |
| 1D8h | ERM Memory 13 Correctable Error Count Register (CORR_ERR_CNT13) | 32 | RW | 0000_0000h |
| 1E0h | ERM Memory 14 Error Address Register (EAR14) | 32 | R | 0000_0000h |
| 1E4h | ERM Memory 14 Syndrome Register (SYN14) | 32 | R | 0000_0000h |
| 1E8h | ERM Memory 14 Correctable Error Count Register (CORR_ERR_CNT14) | 32 | RW | 0000_0000h |
| 1F0h | ERM Memory 15 Error Address Register (EAR15) | 32 | R | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1F4h | ERM Memory 15 Syndrome Register (SYN15) | 32 | R | 0000_0000h |
| 1F8h | ERM Memory 15 Correctable Error Count Register (CORR_ERR_CNT15) | 32 | RW | 0000_0000h |
| 200h | ERM Memory 16 Error Address Register (EAR16) | 32 | R | 0000_0000h |
| 204h | ERM Memory 16 Syndrome Register (SYN16) | 32 | R | 0000_0000h |
| 208h | ERM Memory 16 Correctable Error Count Register (CORR_ERR_CNT16) | 32 | RW | 0000_0000h |
| 210h | ERM Memory 17 Error Address Register (EAR17) | 32 | R | 0000_0000h |
| 218h | ERM Memory 17 Correctable Error Count Register (CORR_ERR_CNT17) | 32 | RW | 0000_0000h |
| 220h | ERM Memory 18 Error Address Register (EAR18) | 32 | R | 0000_0000h |
| 228h | ERM Memory 18 Correctable Error Count Register (CORR_ERR_CNT18) | 32 | RW | 0000_0000h |
| 230h | ERM Memory 19 Error Address Register (EAR19) | 32 | R | 0000_0000h |
| 238h | ERM Memory 19 Correctable Error Count Register (CORR_ERR_CNT19) | 32 | RW | 0000_0000h |

### 48.5.2 ERM Configuration Register 0 (CR0)

#### Offset

| Register | Offset |
|---|---|
| CR0 | 0h |

#### Function

This 32-bit control register configures the interrupt notification capability for available channels.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ESCIE0 | ENCIE0 | 0 | | ESCIE1 | ENCIE1 | 0 | | ESCIE2 | ENCIE2 | 0 | | ESCIE3 | ENCIE3 | 0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ESCIE4 | ENCIE4 | 0 | | ESCIE5 | ENCIE5 | 0 | | ESCIE6 | ENCIE6 | 0 | | ESCIE7 | ENCIE7 | 0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>ESCIE0 | ESCIE0<br>Enable Memory 0 Single Correction Interrupt Notification<br>0b - Interrupt notification of Memory 0 single-bit correction events is disabled.<br>1b - Interrupt notification of Memory 0 single-bit correction events is enabled. |
| 30<br>ENCIE0 | ENCIE0<br>Enable Memory 0 Non-Correctable Interrupt Notification<br>0b - Interrupt notification of Memory 0 non-correctable error events is disabled.<br>1b - Interrupt notification of Memory 0 non-correctable error events is enabled. |
| 29-28<br>— | Reserved |
| 27<br>ESCIE1 | ESCIE1<br>Enable Memory 1 Single Correction Interrupt Notification<br>0b - Interrupt notification of Memory 1 single-bit correction events is disabled.<br>1b - Interrupt notification of Memory 1 single-bit correction events is enabled. |
| 26<br>ENCIE1 | ENCIE1<br>Enable Memory 1 Non-Correctable Interrupt Notification<br>0b - Interrupt notification of Memory 1 non-correctable error events is disabled.<br>1b - Interrupt notification of Memory 1 non-correctable error events is enabled. |
| 25-24<br>— | Reserved |
| 23 | ESCIE2 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| ESCIE2 | Enable Memory 2 Single Correction Interrupt Notification<br>0b - Interrupt notification of Memory 2 single-bit correction events is disabled.<br>1b - Interrupt notification of Memory 2 single-bit correction events is enabled. |
| 22<br>ENCIE2 | ENCIE2<br>Enable Memory 2 Non-Correctable Interrupt Notification<br>0b - Interrupt notification of Memory 2 non-correctable error events is disabled.<br>1b - Interrupt notification of Memory 2 non-correctable error events is enabled. |
| 21-20<br>— | Reserved |
| 19<br>ESCIE3 | ESCIE3<br>Enable Memory 3 Single Correction Interrupt Notification<br>0b - Interrupt notification of Memory 3 single-bit correction events is disabled.<br>1b - Interrupt notification of Memory 3 single-bit correction events is enabled. |
| 18<br>ENCIE3 | ENCIE3<br>Enable Memory 3 Non-Correctable Interrupt Notification<br>0b - Interrupt notification of Memory 3 non-correctable error events is disabled.<br>1b - Interrupt notification of Memory 3 non-correctable error events is enabled. |
| 17-16<br>— | Reserved |
| 15<br>ESCIE4 | ESCIE4<br>Enable Memory 4 Single Correction Interrupt Notification<br>0b - Interrupt notification of Memory 4 single-bit correction events is disabled.<br>1b - Interrupt notification of Memory 4 single-bit correction events is enabled. |
| 14<br>ENCIE4 | ENCIE4<br>Enable Memory 4 Non-Correctable Interrupt Notification<br>0b - Interrupt notification of Memory 4 non-correctable error events is disabled.<br>1b - Interrupt notification of Memory 4 non-correctable error events is enabled. |
| 13-12<br>— | Reserved |
| 11 | ESCIE5<br>Enable Memory 5 Single Correction Interrupt Notification |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| ESCIE5 | 0b - Interrupt notification of Memory 5 single-bit correction events is disabled.<br>1b - Interrupt notification of Memory 5 single-bit correction events is enabled. |
| 10<br>ENCIE5 | ENCIE5<br>Enable Memory 5 Non-Correctable Interrupt Notification<br>0b - Interrupt notification of Memory 5 non-correctable error events is disabled.<br>1b - Interrupt notification of Memory 5 non-correctable error events is enabled. |
| 9-8<br>— | Reserved |
| 7<br>ESCIE6 | ESCIE6<br>Enable Memory 6 Single Correction Interrupt Notification<br>0b - Interrupt notification of Memory 6 single-bit correction events is disabled.<br>1b - Interrupt notification of Memory 6 single-bit correction events is enabled. |
| 6<br>ENCIE6 | ENCIE6<br>Enable Memory 6 Non-Correctable Interrupt Notification<br>0b - Interrupt notification of Memory 6 non-correctable error events is disabled.<br>1b - Interrupt notification of Memory 6 non-correctable error events is enabled. |
| 5-4<br>— | Reserved |
| 3<br>ESCIE7 | ESCIE7<br>Enable Memory 7 Single Correction Interrupt Notification<br>0b - Interrupt notification of Memory 7 single-bit correction events is disabled.<br>1b - Interrupt notification of Memory 7 single-bit correction events is enabled. |
| 2<br>ENCIE7 | ENCIE7<br>Enable Memory 7 Non-Correctable Interrupt Notification<br>0b - Interrupt notification of Memory 7 non-correctable error events is disabled.<br>1b - Interrupt notification of Memory 7 non-correctable error events is enabled. |
| 1-0<br>— | Reserved |

## 48.5.3 ERM Configuration Register 1 (CR1)

### Offset

| Register | Offset |
|---|---|
| CR1 | 4h |

### Function

This 32-bit control register configures the interrupt notification capability for available channels.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ESCIE8 | ENCIE8 | 0 | | ESCIE9 | ENCIE9 | 0 | | ESCIE10 | ENCIE10 | 0 | | ESCIE11 | ENCIE11 | 0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ESCIE12 | ENCIE12 | 0 | | ESCIE13 | ENCIE13 | 0 | | ESCIE14 | ENCIE14 | 0 | | ESCIE15 | ENCIE15 | 0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>ESCIE8 | ESCIE8<br>Enable Memory 8 Single Correction Interrupt Notification<br>0b - Interrupt notification of Memory 8 single-bit correction events is disabled.<br>1b - Interrupt notification of Memory 8 single-bit correction events is enabled. |
| 30<br>ENCIE8 | ENCIE8<br>Enable Memory 8 Non-Correctable Interrupt Notification<br>0b - Interrupt notification of Memory 8 non-correctable error events is disabled.<br>1b - Interrupt notification of Memory 8 non-correctable error events is enabled. |
| 29-28<br>— | Reserved |
| 27<br>ESCIE9 | ESCIE9<br>Enable Memory 9 Single Correction Interrupt Notification<br>0b - Interrupt notification of Memory 9 single-bit correction events is disabled. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Interrupt notification of Memory 9 single-bit correction events is enabled. |
| 26<br>ENCIE9 | ENCIE9<br>Enable Memory 9 Non-Correctable Interrupt Notification<br>0b - Interrupt notification of Memory 9 non-correctable error events is disabled.<br>1b - Interrupt notification of Memory 9 non-correctable error events is enabled. |
| 25-24<br>— | Reserved |
| 23<br>ESCIE10 | ESCIE10<br>Enable Memory 10 Single Correction Interrupt Notification<br>0b - Interrupt notification of Memory 10 single-bit correction events is disabled.<br>1b - Interrupt notification of Memory 10 single-bit correction events is enabled. |
| 22<br>ENCIE10 | ENCIE10<br>Enable Memory 10 Non-Correctable Interrupt Notification<br>0b - Interrupt notification of Memory 10 non-correctable error events is disabled.<br>1b - Interrupt notification of Memory 10 non-correctable error events is enabled. |
| 21-20<br>— | Reserved |
| 19<br>ESCIE11 | ESCIE11<br>Enable Memory 11 Single Correction Interrupt Notification<br>0b - Interrupt notification of Memory 11 single-bit correction events is disabled.<br>1b - Interrupt notification of Memory 11 single-bit correction events is enabled. |
| 18<br>ENCIE11 | ENCIE11<br>Enable Memory 11 Non-Correctable Interrupt Notification<br>0b - Interrupt notification of Memory 11 non-correctable error events is disabled.<br>1b - Interrupt notification of Memory 11 non-correctable error events is enabled. |
| 17-16<br>— | Reserved |
| 15<br>ESCIE12 | ESCIE12<br>Enable Memory 12 Single Correction Interrupt Notification<br>0b - Interrupt notification of Memory 12 single-bit correction events is disabled. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Interrupt notification of Memory 12 single-bit correction events is enabled. |
| 14<br>ENCIE12 | ENCIE12<br>Enable Memory 12 Non-Correctable Interrupt Notification<br>0b - Interrupt notification of Memory 12 non-correctable error events is disabled.<br>1b - Interrupt notification of Memory 12 non-correctable error events is enabled. |
| 13-12<br>— | Reserved |
| 11<br>ESCIE13 | ESCIE13<br>Enable Memory 13 Single Correction Interrupt Notification<br>0b - Interrupt notification of Memory 13 single-bit correction events is disabled.<br>1b - Interrupt notification of Memory 13 single-bit correction events is enabled. |
| 10<br>ENCIE13 | ENCIE13<br>Enable Memory 13 Non-Correctable Interrupt Notification<br>0b - Interrupt notification of Memory 13 non-correctable error events is disabled.<br>1b - Interrupt notification of Memory 13 non-correctable error events is enabled. |
| 9-8<br>— | Reserved |
| 7<br>ESCIE14 | ESCIE14<br>Enable Memory 14 Single Correction Interrupt Notification<br>0b - Interrupt notification of Memory 14 single-bit correction events is disabled.<br>1b - Interrupt notification of Memory 14 single-bit correction events is enabled. |
| 6<br>ENCIE14 | ENCIE14<br>Enable Memory 14 Non-Correctable Interrupt Notification<br>0b - Interrupt notification of Memory 14 non-correctable error events is disabled.<br>1b - Interrupt notification of Memory 14 non-correctable error events is enabled. |
| 5-4<br>— | Reserved |
| 3<br>ESCIE15 | ESCIE15<br>Enable Memory 15 Single Correction Interrupt Notification<br>0b - Interrupt notification of Memory 15 single-bit correction events is disabled. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Interrupt notification of Memory 15 single-bit correction events is enabled. |
| 2<br>ENCIE15 | ENCIE15<br>Enable Memory 15 Non-Correctable Interrupt Notification<br>0b - Interrupt notification of Memory 15 non-correctable error events is disabled.<br>1b - Interrupt notification of Memory 15 non-correctable error events is enabled. |
| 1-0<br>— | Reserved |

### 48.5.4 ERM Configuration Register 2 (CR2)

#### Offset

| Register | Offset |
|---|---|
| CR2 | 8h |

#### Function

This 32-bit control register configures the interrupt notification capability for available channels.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ESCIE16 | ENCIE16 | 0 | | ESCIE17 | ENCIE17 | 0 | | ESCIE18 | ENCIE18 | 0 | | ESCIE19 | ENCIE19 | 0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

#### Fields

| Field | Function |
|---|---|
| 31<br>ESCIE16 | ESCIE16<br>Enable Memory 16 Single Correction Interrupt Notification |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Interrupt notification of Memory 16 single-bit correction events is disabled.<br>1b - Interrupt notification of Memory 16 single-bit correction events is enabled. |
| 30<br>ENCIE16 | ENCIE16<br>Enable Memory 16 Non-Correctable Interrupt Notification<br>0b - Interrupt notification of Memory 16 non-correctable error events is disabled.<br>1b - Interrupt notification of Memory 16 non-correctable error events is enabled. |
| 29-28<br>— | Reserved |
| 27<br>ESCIE17 | ESCIE17<br>Enable Memory 17 Single Correction Interrupt Notification<br>0b - Interrupt notification of Memory 17 single-bit correction events is disabled.<br>1b - Interrupt notification of Memory 17 single-bit correction events is enabled. |
| 26<br>ENCIE17 | ENCIE17<br>Enable Memory 17 Non-Correctable Interrupt Notification<br>0b - Interrupt notification of Memory 17 non-correctable error events is disabled.<br>1b - Interrupt notification of Memory 17 non-correctable error events is enabled. |
| 25-24<br>— | Reserved |
| 23<br>ESCIE18 | ESCIE18<br>Enable Memory 18 Single Correction Interrupt Notification<br>0b - Interrupt notification of Memory 18 single-bit correction events is disabled.<br>1b - Interrupt notification of Memory 18 single-bit correction events is enabled. |
| 22<br>ENCIE18 | ENCIE18<br>Enable Memory 18 Non-Correctable Interrupt Notification<br>0b - Interrupt notification of Memory 18 non-correctable error events is disabled.<br>1b - Interrupt notification of Memory 18 non-correctable error events is enabled. |
| 21-20<br>— | Reserved |
| 19<br>ESCIE19 | ESCIE19<br>Enable Memory 19 Single Correction Interrupt Notification |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Interrupt notification of Memory 19 single-bit correction events is disabled.<br>1b - Interrupt notification of Memory 19 single-bit correction events is enabled. |
| 18<br>ENCIE19 | ENCIE19<br>Enable Memory 19 Non-Correctable Interrupt Notification<br>0b - Interrupt notification of Memory 19 non-correctable error events is disabled.<br>1b - Interrupt notification of Memory 19 non-correctable error events is enabled. |
| 17-16<br>— | Reserved |
| 15-14<br>— | Reserved |
| 13-12<br>— | Reserved |
| 11-10<br>— | Reserved |
| 9-8<br>— | Reserved |
| 7-6<br>— | Reserved |
| 5-4<br>— | Reserved |
| 3-2<br>— | Reserved |
| 1-0<br>— | Reserved |

### 48.5.5 ERM Status Register 0 (SR0)

**Offset**

| Register | Offset |
|---|---|
| SR0 | 10h |

### Function

This 32-bit status register reports error events for available channels.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SBC0 | NCE0 | 0 | | SBC1 | NCE1 | 0 | | SBC2 | NCE2 | 0 | | SBC3 | NCE3 | 0 | |
| W | W1C | W1C | | | W1C | W1C | | | W1C | W1C | | | W1C | W1C | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SBC4 | NCE4 | 0 | | SBC5 | NCE5 | 0 | | SBC6 | NCE6 | 0 | | SBC7 | NCE7 | 0 | |
| W | W1C | W1C | | | W1C | W1C | | | W1C | W1C | | | W1C | W1C | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>SBC0 | SBC0<br>Memory 0 Single-Bit Correction Event<br>Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ESCIE0] is enabled.<br>0b - No single-bit correction event on Memory 0 detected.<br>1b - Single-bit correction event on Memory 0 detected. |
| 30<br>NCE0 | NCE0<br>Memory 0 Non-Correctable Error Event<br>Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ENCIE0] is enabled.<br>0b - No non-correctable error event on Memory 0 detected.<br>1b - Non-correctable error event on Memory 0 detected. |
| 29-28<br>— | Reserved |
| 27<br>SBC1 | SBC1<br>Memory 1 Single-Bit Correction Event<br>Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ESCIE1] is enabled.<br>0b - No single-bit correction event on Memory 1 detected.<br>1b - Single-bit correction event on Memory 1 detected. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 26<br>NCE1 | NCE1<br>Memory 1 Non-Correctable Error Event<br>Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ENCIE1] is enabled.<br>0b - No non-correctable error event on Memory 1 detected.<br>1b - Non-correctable error event on Memory 1 detected. |
| 25-24<br>— | Reserved |
| 23<br>SBC2 | SBC2<br>Memory 2 Single-Bit Correction Event<br>Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ESCIE2] is enabled.<br>0b - No single-bit correction event on Memory 2 detected.<br>1b - Single-bit correction event on Memory 2 detected. |
| 22<br>NCE2 | NCE2<br>Memory 2 Non-Correctable Error Event<br>Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ENCIE2] is enabled.<br>0b - No non-correctable error event on Memory 2 detected.<br>1b - Non-correctable error event on Memory 2 detected. |
| 21-20<br>— | Reserved |
| 19<br>SBC3 | SBC3<br>Memory 3 Single-Bit Correction Event<br>Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ESCIE3] is enabled.<br>0b - No single-bit correction event on Memory 3 detected.<br>1b - Single-bit correction event on Memory 3 detected. |
| 18<br>NCE3 | NCE3<br>Memory 3 Non-Correctable Error Event<br>Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ENCIE3] is enabled.<br>0b - No non-correctable error event on Memory 3 detected. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Non-correctable error event on Memory 3 detected. |
| 17-16<br>— | Reserved |
| 15<br>SBC4 | SBC4<br>Memory 4 Single-Bit Correction Event<br>Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ESCIE4] is enabled.<br>0b - No single-bit correction event on Memory 4 detected.<br>1b - Single-bit correction event on Memory 4 detected. |
| 14<br>NCE4 | NCE4<br>Memory 4 Non-Correctable Error Event<br>Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ENCIE4] is enabled.<br>0b - No non-correctable error event on Memory 4 detected.<br>1b - Non-correctable error event on Memory 4 detected. |
| 13-12<br>— | Reserved |
| 11<br>SBC5 | SBC5<br>Memory 5 Single-Bit Correction Event<br>Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ESCIE5] is enabled.<br>0b - No single-bit correction event on Memory 5 detected.<br>1b - Single-bit correction event on Memory 5 detected. |
| 10<br>NCE5 | NCE5<br>Memory 5 Non-Correctable Error Event<br>Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ENCIE5] is enabled.<br>0b - No non-correctable error event on Memory 5 detected.<br>1b - Non-correctable error event on Memory 5 detected. |
| 9-8<br>— | Reserved |
| 7 | SBC6 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| SBC6 | Memory 6 Single-Bit Correction Event<br>Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ESCIE6] is enabled.<br>0b - No single-bit correction event on Memory 6 detected.<br>1b - Single-bit correction event on Memory 6 detected. |
| 6<br>NCE6 | NCE6<br>Memory 6 Non-Correctable Error Event<br>Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ENCIE6] is enabled.<br>0b - No non-correctable error event on Memory 6 detected.<br>1b - Non-correctable error event on Memory 6 detected. |
| 5-4<br>— | Reserved |
| 3<br>SBC7 | SBC7<br>Memory 7 Single-Bit Correction Event<br>Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ESCIE7] is enabled.<br>0b - No single-bit correction event on Memory 7 detected.<br>1b - Single-bit correction event on Memory 7 detected. |
| 2<br>NCE7 | NCE7<br>Memory 7 Non-Correctable Error Event<br>Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR0[ENCIE7] is enabled.<br>0b - No non-correctable error event on Memory 7 detected.<br>1b - Non-correctable error event on Memory 7 detected. |
| 1-0<br>— | Reserved |

## 48.5.6 ERM Status Register 1 (SR1)

**Offset**

| Register | Offset |
|---|---|
| SR1 | 14h |

### Function

This 32-bit status register reports error events for available channels.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SBC8 | NCE8 | 0 | | SBC9 | NCE9 | 0 | | SBC10 | NCE10 | 0 | | SBC11 | NCE11 | 0 | |
| W | W1C | W1C | | | W1C | W1C | | | W1C | W1C | | | W1C | W1C | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SBC12 | NCE12 | 0 | | SBC13 | NCE13 | 0 | | SBC14 | NCE14 | 0 | | SBC15 | NCE15 | 0 | |
| W | W1C | W1C | | | W1C | W1C | | | W1C | W1C | | | W1C | W1C | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>SBC8 | SBC8<br>Memory 8 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR1[ESCIE8] is enabled.<br>0b - No single-bit correction event on Memory 8 detected.<br>1b - Single-bit correction event on Memory 8 detected. |
| 30<br>NCE8 | NCE8<br>Memory 8 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR1[ENCIE8] is enabled.<br>0b - No non-correctable error event on Memory 8 detected.<br>1b - Non-correctable error event on Memory 8 detected. |
| 29-28<br>— | Reserved |
| 27<br>SBC9 | SBC9<br>Memory 9 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR1[ESCIE9] is enabled.<br>0b - No single-bit correction event on Memory 9 detected.<br>1b - Single-bit correction event on Memory 9 detected. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 26<br>NCE9 | NCE9<br>Memory 9 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR1[ENCIE9] is enabled.<br>0b - No non-correctable error event on Memory 9 detected.<br>1b - Non-correctable error event on Memory 9 detected. |
| 25-24<br>— | Reserved |
| 23<br>SBC10 | SBC10<br>Memory 10 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR1[ESCIE10] is enabled.<br>0b - No single-bit correction event on Memory 10 detected.<br>1b - Single-bit correction event on Memory 10 detected. |
| 22<br>NCE10 | NCE10<br>Memory 10 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR1[ENCIE10] is enabled.<br>0b - No non-correctable error event on Memory 10 detected.<br>1b - Non-correctable error event on Memory 10 detected. |
| 21-20<br>— | Reserved |
| 19<br>SBC11 | SBC11<br>Memory 11 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR1[ESCIE11] is enabled.<br>0b - No single-bit correction event on Memory 11 detected.<br>1b - Single-bit correction event on Memory 11 detected. |
| 18<br>NCE11 | NCE11<br>Memory 11 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR1[ENCIE11] is enabled.<br>0b - No non-correctable error event on Memory 11 detected. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Non-correctable error event on Memory 11 detected. |
| 17-16<br>— | Reserved |
| 15<br>SBC12 | SBC12<br>Memory 12 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR1[ESCIE12] is enabled.<br>0b - No single-bit correction event on Memory 12 detected.<br>1b - Single-bit correction event on Memory 12 detected. |
| 14<br>NCE12 | NCE12<br>Memory 12 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR1[ENCIE12] is enabled.<br>0b - No non-correctable error event on Memory 12 detected.<br>1b - Non-correctable error event on Memory 12 detected. |
| 13-12<br>— | Reserved |
| 11<br>SBC13 | SBC13<br>Memory 13 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR1[ESCIE13] is enabled.<br>0b - No single-bit correction event on Memory 13 detected.<br>1b - Single-bit correction event on Memory 13 detected. |
| 10<br>NCE13 | NCE13<br>Memory 13 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR1[ENCIE13] is enabled.<br>0b - No non-correctable error event on Memory 13 detected.<br>1b - Non-correctable error event on Memory 13 detected. |
| 9-8<br>— | Reserved |
| 7 | SBC14 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| SBC14 | Memory 14 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR1[ESCIE14] is enabled.<br>0b - No single-bit correction event on Memory 14 detected.<br>1b - Single-bit correction event on Memory 14 detected. |
| 6<br>NCE14 | NCE14<br>Memory 14 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR1[ENCIE14] is enabled.<br>0b - No non-correctable error event on Memory 14 detected.<br>1b - Non-correctable error event on Memory 14 detected. |
| 5-4<br>— | Reserved |
| 3<br>SBC15 | SBC15<br>Memory 15 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR1[ESCIE15] is enabled.<br>0b - No single-bit correction event on Memory 15 detected.<br>1b - Single-bit correction event on Memory 15 detected. |
| 2<br>NCE15 | NCE15<br>Memory 15 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR1[ENCIE15] is enabled.<br>0b - No non-correctable error event on Memory 15 detected.<br>1b - Non-correctable error event on Memory 15 detected. |
| 1-0<br>— | Reserved |

## 48.5.7 ERM Status Register 2 (SR2)

**Offset**

| Register | Offset |
|---|---|
| SR2 | 18h |

### Function

This 32-bit status register reports error events for available channels.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SBC16 | NCE16 | 0 | | SBC17 | NCE17 | 0 | | SBC18 | NCE18 | 0 | | SBC19 | NCE19 | 0 | |
| W | W1C | W1C | | | W1C | W1C | | | W1C | W1C | | | W1C | W1C | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31<br>SBC16 | SBC16<br>Memory 16 Single-Bit Correction Event<br>Write 1 to clear this field.This write also clears the corresponding interrupt notification, if CR2[ESCIE16] is enabled.<br>0b - No single-bit correction event on Memory 16 detected.<br>1b - Single-bit correction event on Memory 16 detected. |
| 30<br>NCE16 | NCE16<br>Memory 16 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR2[ENCIE16] is enabled.<br>0b - No non-correctable error event on Memory 16 detected.<br>1b - Non-correctable error event on Memory 16 detected. |
| 29-28<br>— | Reserved |
| 27<br>SBC17 | SBC17<br>Memory 17 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR2[ESCIE17] is enabled.<br>0b - No single-bit correction event on Memory 17 detected.<br>1b - Single-bit correction event on Memory 17 detected. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 26<br>NCE17 | NCE17<br>Memory 17 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR2[ENCIE17] is enabled.<br>0b - No non-correctable error event on Memory 17 detected.<br>1b - Non-correctable error event on Memory 17 detected. |
| 25-24<br>— | Reserved |
| 23<br>SBC18 | SBC18<br>Memory 18 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR2[ESCIE18] is enabled.<br>0b - No single-bit correction event on Memory 18 detected.<br>1b - Single-bit correction event on Memory 18 detected. |
| 22<br>NCE18 | NCE18<br>Memory 18 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR2[ENCIE18] is enabled.<br>0b - No non-correctable error event on Memory 18 detected.<br>1b - Non-correctable error event on Memory 18 detected. |
| 21-20<br>— | Reserved |
| 19<br>SBC19 | SBC19<br>Memory 19 Single-Bit Correction Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR2[ESCIE19] is enabled.<br>0b - No single-bit correction event on Memory 19 detected.<br>1b - Single-bit correction event on Memory 19 detected. |
| 18<br>NCE19 | NCE19<br>Memory 19 Non-Correctable Error Event<br>Write 1 to clear this field. This write also clears the corresponding interrupt notification, if CR2[ENCIE19] is enabled.<br>0b - No non-correctable error event on Memory 19 detected. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Non-correctable error event on Memory 19 detected. |
| 17-16<br>— | Reserved |
| 15-12<br>— | Reserved |
| 11-8<br>— | Reserved |
| 7-4<br>— | Reserved |
| 3-0<br>— | Reserved |

### 48.5.8 ERM Memory a Error Address Register (EAR0 - EAR19)

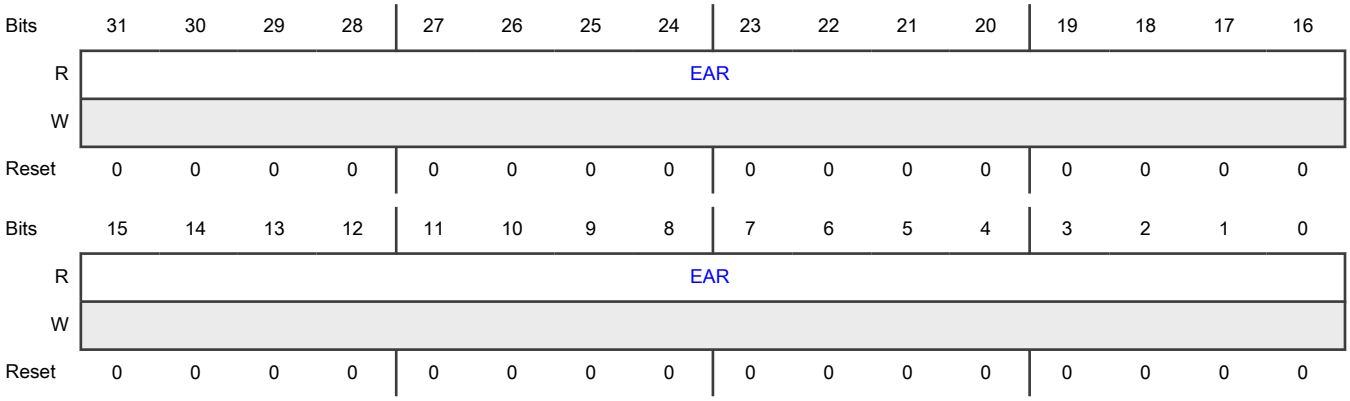
#### Offset

| Register | Offset |
|---|---|
| EAR0 | 100h |
| EAR1 | 110h |
| EAR10 | 1A0h |
| EAR11 | 1B0h |
| EAR12 | 1C0h |
| EAR13 | 1D0h |
| EAR14 | 1E0h |
| EAR15 | 1F0h |
| EAR16 | 200h |
| EAR17 | 210h |
| EAR18 | 220h |
| EAR19 | 230h |

#### Function

Each ERM Memory n Error Address Register is a 32-bit register for capturing the address of the last ECC event in Memory*n*, where*n* denotes the memory channel. Any attempted write to EAR*n* is ignored.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>EAR | EAR<br>Memory*n* Error Address — This field contains the faulting system address of the last recorded ECC event on Memory*n*. |

## 48.5.9 ERM Memory a Syndrome Register (SYN0 - SYN16)

### Offset

| Register | Offset |
|---|---|
| SYN0 | 104h |
| SYN1 | 114h |
| SYN10 | 1A4h |
| SYN11 | 1B4h |
| SYN12 | 1C4h |
| SYN13 | 1D4h |
| SYN14 | 1E4h |
| SYN15 | 1F4h |
| SYN16 | 204h |

### Function

The ERM Memory *n* Syndrome Register is a 32-bit register for capturing the calculated syndrome of the last ECC event on Memory*n*, where*n* denotes the memory channel. Any attempted write to SYN*n* is ignored. The syndrome value identifies the pertinent bit position on a correctable, single-bit data inversion or a non-correctable, single-bit address inversion. The syndrome value does not provide any additional diagnostic information on non-correctable, multi-bit inversions.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-24<br>SYNDROME | SYNDROME<br>Memory*n* Syndrome — This field contains the ECC syndrome associated with the last recorded ECC event on Memory*n*. |
| 23-0<br>— | Reserved |

## 48.5.10 ERM Memory a Correctable Error Count Register (CORR_ERR_CNT0 - CORR_ERR_CNT19)

### Offset

| Register | Offset |
|---|---|
| CORR_ERR_CNT0 | 108h |
| CORR_ERR_CNT1 | 118h |
| CORR_ERR_CNT2 | 128h |
| CORR_ERR_CNT3 | 138h |
| CORR_ERR_CNT4 | 148h |
| CORR_ERR_CNT5 | 158h |
| CORR_ERR_CNT6 | 168h |
| CORR_ERR_CNT7 | 178h |
| CORR_ERR_CNT8 | 188h |
| CORR_ERR_CNT9 | 198h |
| CORR_ERR_CNT10 | 1A8h |
| CORR_ERR_CNT11 | 1B8h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | Offset |
|---|---|
| CORR_ERR_CNT12 | 1C8h |
| CORR_ERR_CNT13 | 1D8h |
| CORR_ERR_CNT14 | 1E8h |
| CORR_ERR_CNT15 | 1F8h |
| CORR_ERR_CNT16 | 208h |
| CORR_ERR_CNT17 | 218h |
| CORR_ERR_CNT18 | 228h |
| CORR_ERR_CNT19 | 238h |

### Function

Each 32-bit ERM Memory *n* Correctable Error Count Register records the count value of the number of correctable ECC error events for Memory*n*, where*n* denotes the memory channel.

**NOTE**

Non-correctable errors are considered a serious fault, so the ERM does not provide any mechanism to count non-correctable errors. Only correctable errors are counted.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>COUNT | Memory n Correctable Error Count |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | For each correctable error event, the ERM increments this field's error count value until the counter reaches its maximum value FFh. COUNT value will stop when it reaches maximum value FFh and will not wrap even if additional errors occur.<br>Read this field to determine the correctable error count value so far.<br>Write all zeros to this field to reset the counter. Writing a non-zero value has no effect. |

## 48.6 Glossary

**ECC** Error correction code

# Chapter 49
# Fault Collection and Control Unit (FCCU)

## 49.1 Chip-specific FCCU information

### 49.1.1 FCCU NCF slots

**Table 226. FCCU NCF slots**

| Slot number | Source module (error type) |
|---|---|
| NCF[0] | • Cortex-M7 LS and core lockup<br>• ELE_HSEB lockup |
| NCF[1] | Interconnect:<br>• All EDC bus gaskets<br>• XBIC monitors and platform gaskets |
| NCF[2] | ECC errors:<br>• PRAMC<br>• TCMs<br>• Caches<br>• eDMA<br>• EDC after ECC<br>• ELE_HSEB RAM errors |
| NCF[3] | All flash memory errors:<br>• FMU<br>• PFLASH<br>• DCM flash memory |
| NCF[4] | Voltage-related errors:<br>• PMC 1.1 V and 2.5 V GNG<br>• Pad overvoltage |
| NCF[5] | Debug and test monitoring |
| NCF[6] | INTM |
| NCF[7] | Software notification |

### 49.1.2 Chip-boundary FCCU signals

**Table 227. Chip-boundary FCCU signals**

| FCCU signal | Chip signal |
|---|---|
| EOUT0 | FCCU_ERR0 |
| EOUT1 | FCCU_ERR1 |

### 49.1.3 FCCU clocking

Table 228. FCCU clocking

| FCCU clock signal | Chip clock signal |
|---|---|
| CLKSAFE | FIRC_CLK |
| CLKPRIM | AIPS_PLAT_CLK |

### 49.1.4 FOSU timer interval

The FOSU_COUNT value determines the FOSU timer interval. On this chip, FOSU_COUNT is 69780h.

### 49.1.5 Supported internal chip reactions

The short functional reset discussed in this chapter is equivalent to the chip functional reset. See the interrupt map file attached to this document for interrupts from FCCU to NVIC.

**NOTE**

After STCU2 completes the self-testing procedure, the chip reboots and FCCU resets.

### 49.1.6 Supported FCCU EOUT fault output modes (protocols)

This chip supports only the Bistable protocol of FCCU EOUT, and does not support the other protocols discussed in this chapter, Dual-Rail and Time-Switching.

### 49.1.7 Recommended reaction programming for faults

You can upgrade or downgrade the reaction of the faults according to the recommended reaction discussed in the fault map file attached to this document.

In case you upgrade a reaction, no issues are expected in the behavior. If you downgrade the reaction, the functionality is not guaranteed.

This is the recommendation for faults caused by lockstep errors:

1. The recommended reaction for the lockstep error fault is a functional reset.
2. Program the core to perform the following steps after rebooting:
    a. Initialize TCM (because TCMs can become corrupted)
    b. Invalidate the cache (because caches can become corrupted)
3. Write 1 to the corresponding bit in FCCU.NCF_S0 register to clear the non-critical fault status.
4. If the FCCU fault gets cleared, you have nothing else to do for fault handling (this means, the FCCU lockstep error is recovered).
5. If the error persists, it indicates that the internal registers of the two cores have different values. To bring them to the same value, you could perform any of these steps:
    a. Initialize destructive reset through software by using MC_ME.MODE_CONF[DEST_RST] to recover from lockstep. This initializes the FCCU with no pending faults.
    b. Reconfigure the Cortex-M7 debug configurations (which would have been lost in previous step due to destructive reset in the system).

### 49.1.8 FCCU NCF handling architecture

This chip supports eight NCFs. Therefore, multiple faults of similar nature are ORed together and then connected as a single NCF to FCCU. To control individual fault, there are enable/disable controls provided within DCM in registers DCMRWD3, DCMRWD4 and DCMRWD5.

Similarly, the FCCU.NCF_S0 captures the status of fault within FCCU. This fault might have resulted due to any fault mapped on the corresponding FCCU NCF channel. The DCM status registers, DCMROD3, DCMROD4 and DCMROD5 capture the status of individual faults which are mapped onto FCCU NCFs. Figure 179 shows this arrangement.

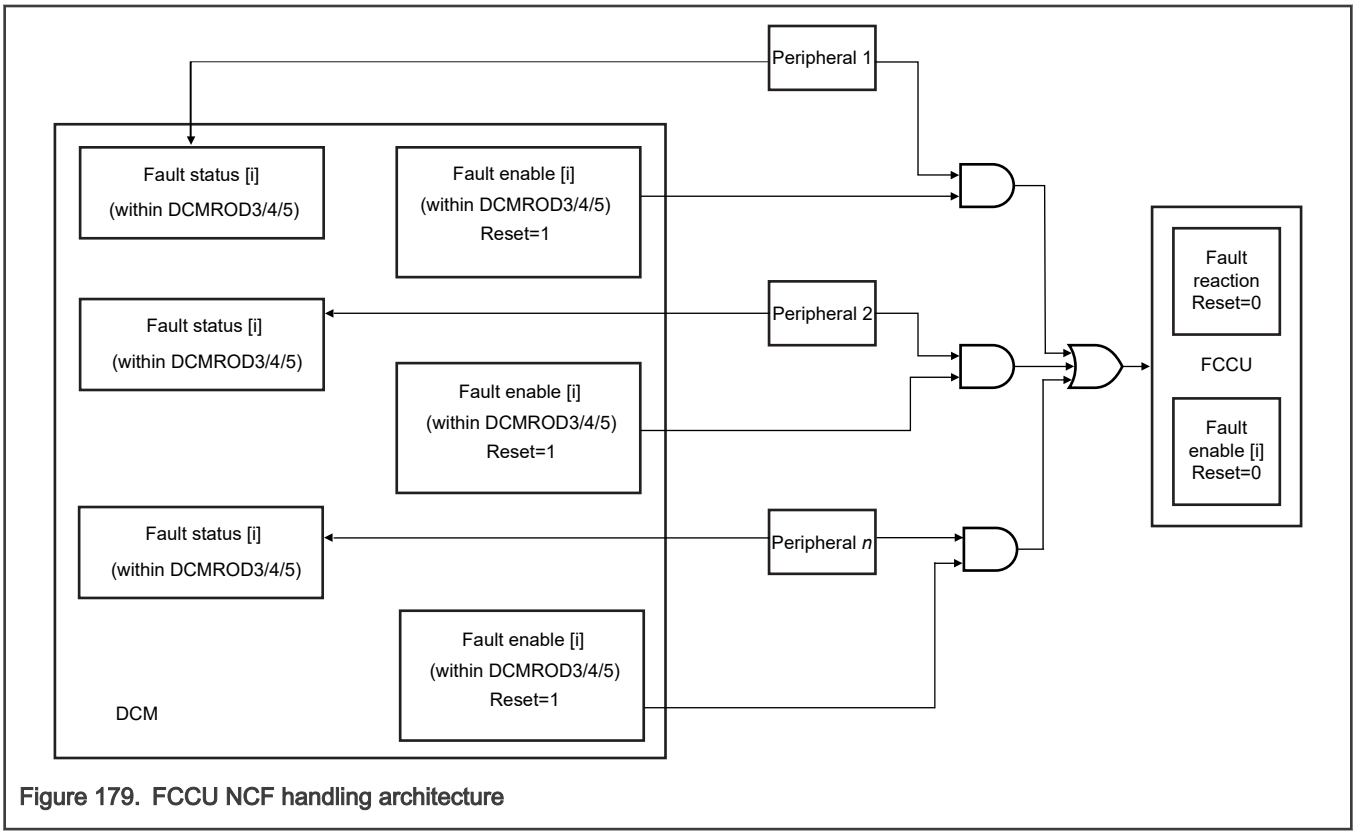



**Figure 179. FCCU NCF handling architecture**

## 49.2 Overview

The FCCU provides a hardware interface to collect faults and to place the device into a safe state when a failure is detected in the device. No CPU intervention is requested for collection and control operations. FCCU offers a systematic approach to fault collection and control.

### 49.2.1 Block diagram

The following figure represents a top-level diagram of the FCCU module.

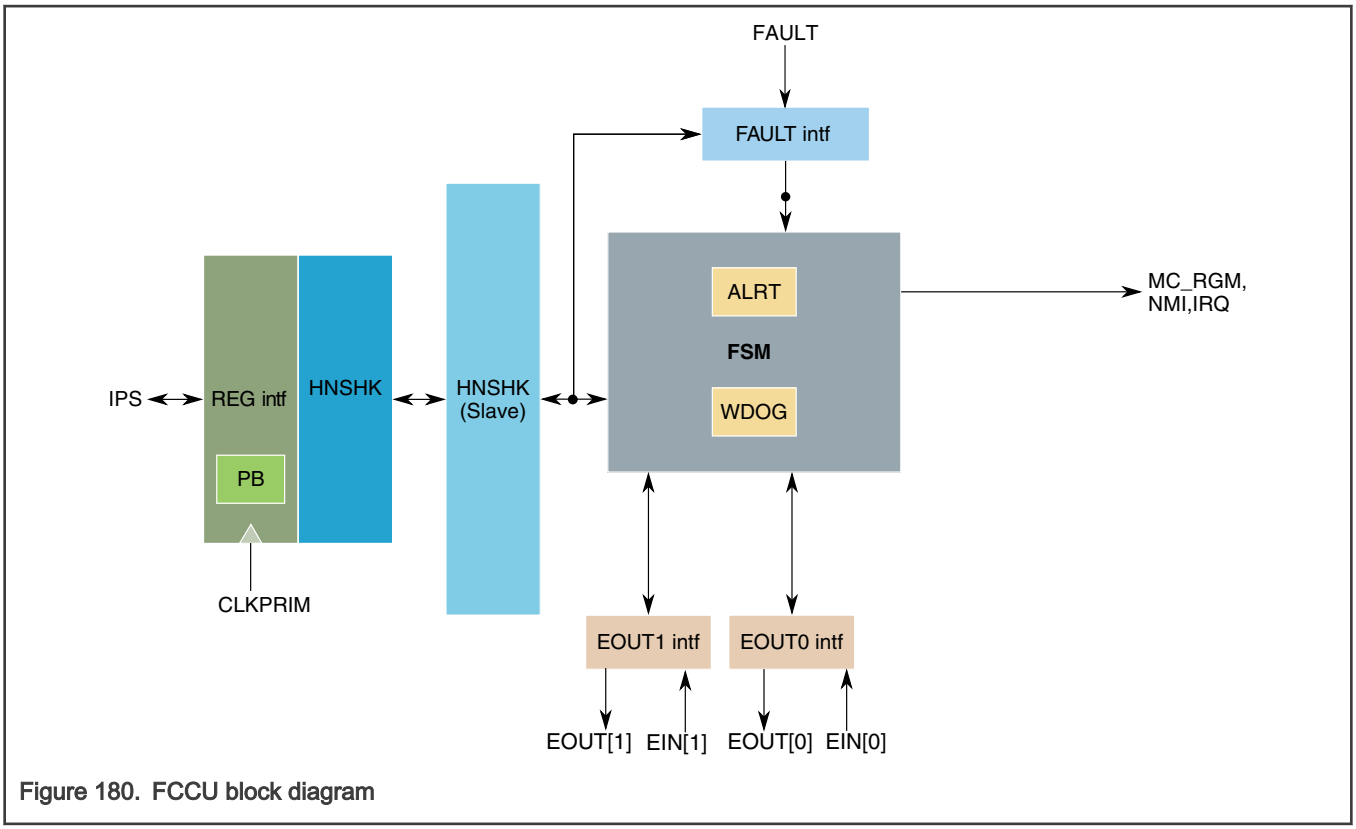



Figure 180. FCCU block diagram

This table describes the FCCU submodules.

Table 229. FCCU submodules

| Submodule | Description |
|---|---|
| REG intf | Includes the register file, the IPS bus interface, the IRQ interface and the parity block (PB) for the configuration registers |
| HNSHK blocks (master and slave blocks) | Includes the FSM ability to support the handshake between the REG interface and the FSM unit because of the usage of two asynchronous clocks [CLKPRIM(module clock) and CLKSAFE(RC oscillator clock)] |
| FSM unit | Implements the main functions of FCCU. The FSM also includes the:<br>• Watchdog timer (WDG)<br>• Alarm timer (ALRT) |
| FAULT intf | Implements the interface for fault conditioning and management |
| EOUTx units | Implement the output stage to manage the EOUT interfaces |

## 49.2.2 Features

The key features of the FCCU module are these:

- Management of non-critical faults
- HW or SW fault recovery management
- Fault collection from safety relevant modules on the device

- Fault injection (fake faults)
- Collection of test results
- Lockable configuration
  - Changes are only possible after entering the CONFIG state
  - Supports a transient and a permanent lock
  - Configuration changes observed by a watchdog timer
- Configurable fault control
- External reaction (FAULT state): EOUT signaling. Error indication via the pin(s) is controlled by FCCU.
- Internal chip reactions (ALARM state): interrupt request
- Configurable internal chip reactions for each NCF (FAULT state):
  - Short functional reset request pulse
  - NMI
  - No reaction
  - IRQ
- In Bi-Stable operational mode, one of the EOUT signals is high to indicate an OK operational state of FCCU.
- After power on, the EOUT signals have high impedance.[6] They indicate an operational state only after the software configures them.
- In case of a failure event or on software request for error pin indication, the pin(s) are set to faulty state for a minimum time $T_{min}$ (see DELTA_T[DELTA_T]), even if the software tries to release it before (for the case of error pin configured in Bi-Stable mode only).

The self-checking procedure checks the FCCU circuitry at the start up. The FCCU is operational with the default configuration immediately after the completion of the self-checking procedure. Internal (short functional reset request pulse, interrupt request) and external (EOUT signaling) reactions are statically defined or programmable. The default configuration can be modified only in the Configuration (CONFIG) state. FCCU is designed to function when CLKPRIM is faster than the CLKSAFE clocks.

## 49.3 Functional description

### 49.3.1 Definitions

In general, the following definitions are applicable for fault management:

- HW recoverable fault: The fault indication is a level-sensitive signal that remains asserted until the fault cause is deasserted. That is, if logical 0 on the fault signal indicates fault, then the status flags are valid as long as the fault line stays at 0. The status is automatically cleared when the fault signal goes to 1. Typically the fault signal is latched external to the FCCU in the module where the fault occurred. The FCCU state transitions are consequently executed on the state changes of the input fault signal. No SW intervention in the FCCU is required to recover the fault condition.
- SW recoverable fault: The fault indication is a signal asserted without a defined time duration. The fault signal is latched in the FCCU. The fault recovery is executed following a SW recovery procedure (status/flag register clearing).

HW recoverable is an option to exclude the handling of error sources by FCCU management SW, in case it is known that the fault is recoverable by itself when the fault condition is corrected.

For details related to reset interface, see the reset interface section in the chip reference manual.

### 49.3.2 FSM description

The functionality of FCCU is depicted by the FSM state diagram (see Figure 181).

[6] Actual value depends on device-specific setting at pad level.

FCCU has four states that are identified with the following meaning:

- CONFIG: Used only to modify the configuration of FCCU from its default. A subset of the FCCU registers, dedicated to define the FCCU configuration (global configuration, reactions to fault, timeout, non-critical fault masking) can be accessed in write mode only in the CONFIG state.

  The CONFIG state is accessible only in the NORMAL state and if the configuration is not locked. A permanent configuration lock can be disabled by a reset that also resets the FCCU. The transient lock register is unlocked by writing BCh into it. FCCU gets transiently locked again if an invalid key is written into TRANS_LOCK[TRANSKEY] (that is, other than BCh). To lock FCCU for configuration, write FFh to PERMNT_LOCK[PERMNTKEY].

  After the release of reset, the state of the transient lock is locked, and the state of the permanent lock is unlocked.

  The locking feature only restricts the FSM movement into CONFIG state. After the user enters the CONFIG state and then tries to lock the configuration, the locking of configuration is effective only after FCCU moves to the NORMAL state; it will not be effective in the current CONFIG state.

  The CONFIG to NORMAL state transition can be executed by SW or automatically following a timeout condition of the watchdog. In case the timeout information and the SW request for state change to NORMAL appears at the same time, watchdog timeout has the priority and hence the configuration registers (those that are writable only in the CONFIG state) are reset to their default values. The movement to the NORMAL state is made.

  The incoming faults, occurring during the configuration phase (CONFIG state) are latched in order to process them when FCCU is moved to the NORMAL state, according to the new configuration.

  All pending faults that occur during the CONFIG state result in both of the following:

  — Highest-priority state transition

  — Interrupt generation (NMI or alarm IRQ)

  If the state transition occurs, it gives the reset reaction corresponding to the worst case based on all the faults (pending or non-pending faults) that occurred during the CONFIG state.

- NORMAL: This is FCCU's operating state when no faults are occurring. It is also the default state on the reset exit. Following state transitions occur on one of the following events:

  — Unmasked non-critical faults with the timeout disabled: FCCU moves to the FAULT state.

  — Unmasked non-critical faults with the timeout enabled: FCCU moves to the ALARM state.

  — Masked non-critical faults: FCCU stays in the NORMAL state.

- ALARM: FCCU moves into the ALARM state when an unmasked non-critical fault occurs and the timeout is enabled. Transition to the ALARM state goes along with an interrupt alarm, if enabled. By definition, this fault may be recovered within a programmable timeout period, before it generates a transition to the FAULT state. The timeout is reinitialized if FCCU enters the NORMAL state. The timeout restarts following the recovery from the FAULT state.
- FAULT: FCCU moves into the FAULT state when one of the following condition occurs:

  — Timeout related to a non-critical fault when FCCU is in the ALARM state

  — Unmasked non-critical faults with the timeout disabled

The transition from the NORMAL or ALARM to the FAULT state goes along with the generation of:

- Internal chip reaction—NMI interrupt (optional)
- External reaction—EOUT signaling (optional)
- Internal chip reaction—SW option: Soft reaction (Short functional reset request pulse if configured)
- Non Maskable Interrupt (NMI) is routed to all cores.

After moving to the FAULT state, if there is either a previous pending fault or a new fault for which NMI is enabled, NMI generation takes place.

Multiple faults can occur at the same time.

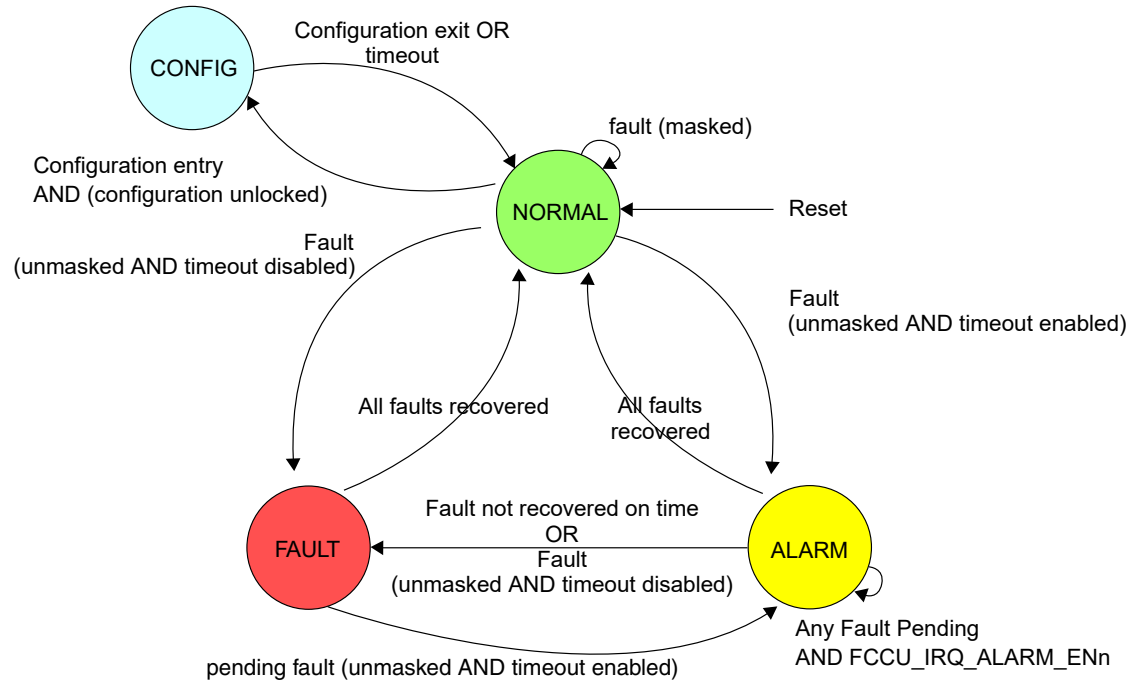



Figure 181. FCCU state diagram

### 49.3.3 Fault priority scheme and nesting

The FAULT state has a higher priority than the ALARM state in case of concurrent fault events (non-critical) that occur in the NORMAL state.

The ALARM to FAULT state transition occurs if a non-critical fault (unmasked and with timeout disabled) is asserted in the ALARM state.

The ALARM to NORMAL state transition occurs only if all the non-critical faults (including the faults that have been collected after the entry in the ALARM state) have been cleared (SW or HW recovery); otherwise the FCCU remains in the ALARM state.

The FAULT to NORMAL state transition occurs only if all the non-critical faults (including the faults that have been collected after the entry in the FAULT/ALARM state) have been cleared (SW or HW recovery); otherwise the FCCU moves to the ALARM state (if any non-critical fault is still pending and the timeout is not elapsed).

In general, no fault nesting is supported except for the non-critical faults that cause an ALARM to FAULT state transition. In this case, the NCF timer is stopped until the FAULT state is recovered. If FCCU is in the ALARM state and another fault occurs, which has its alarm timeout enabled, then the alarm timer shall not reload and shall not start again.

### 49.3.4 Fault recovery

The following timing diagrams describe the main use cases of FCCU in terms of fault events and related recovery.

A typical sequence related to non-critical fault management (ALARM state), see Figure 182 and Figure 183, is as follows:

1. Non-critical fault assertion
2. FCCU state transition (automatic): NORMAL to ALARM
   - Alarm interrupt request (if enabled)
   - Timeout running
3. System state: RUN
4. Alarm interrupt management: fault recovery

- FCCU state transition: ALARM to NORMAL

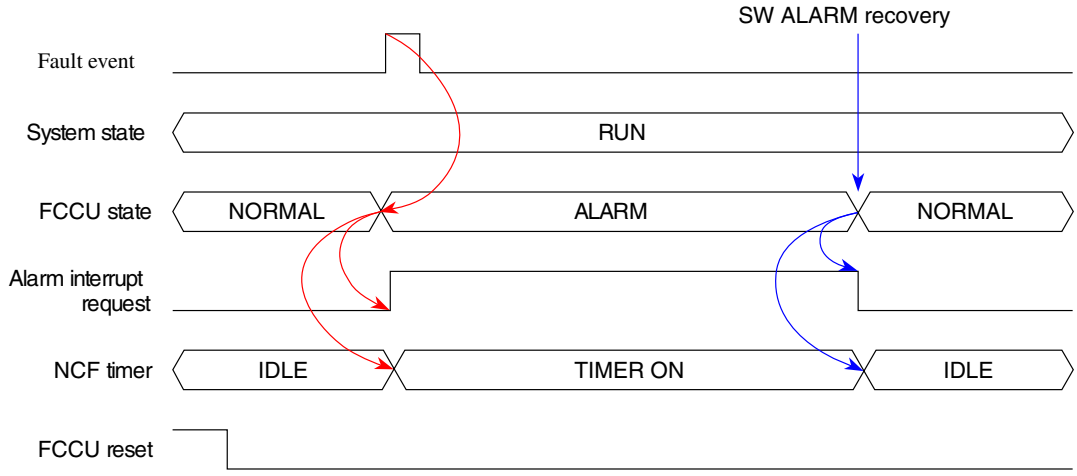



**Figure 182. Non-critical fault (ALARM state) SW recovery**

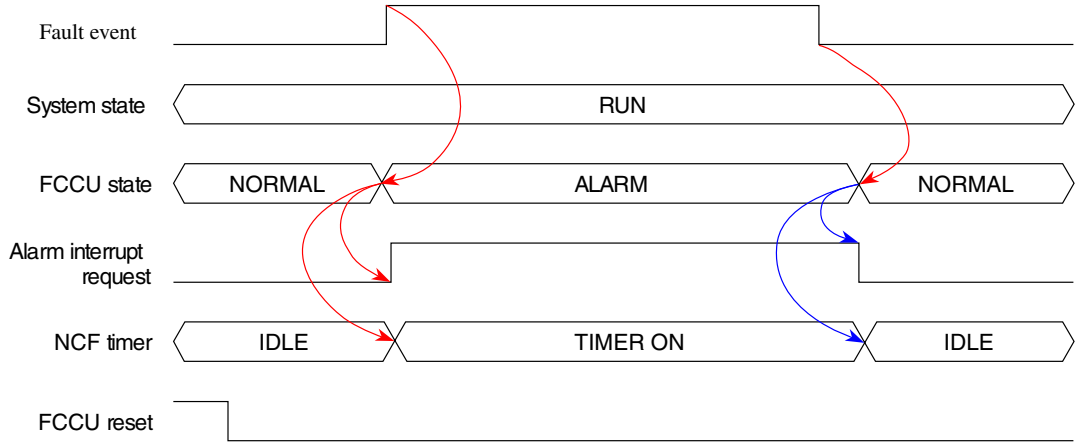



**Figure 183. Non-critical fault (ALARM state) HW recovery**

A typical sequence related to non-critical fault management (ALARM to FAULT state), see Figure 184, is as follows:

1. Non-critical fault assertion
2. FCCU state transition (automatic): NORMAL to ALARM
   - Alarm interrupt request (if enabled)
   - Timeout running
3. FCCU state transition (following the timeout trigger): ALARM to FAULT
   - NMI assertion (if enabled)
4. NMI interrupt management (if enabled)
   - Fault recovery (by software): FCCU state transition: FAULT to NORMAL

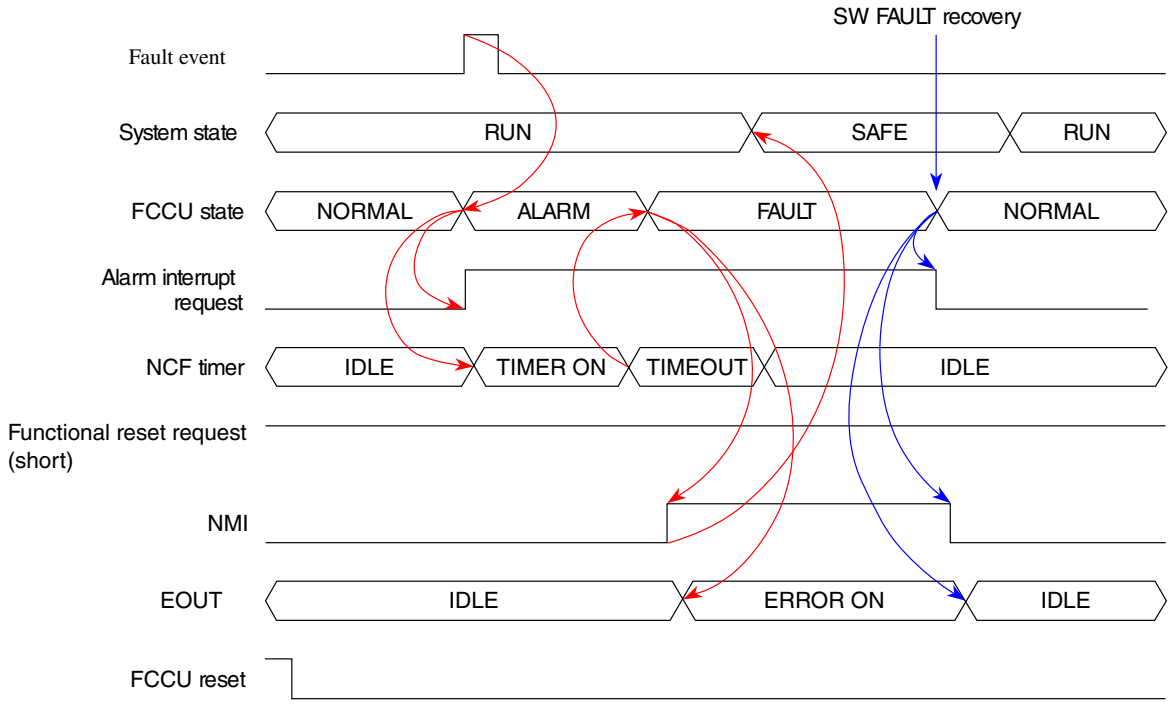



**Figure 184. Non-critical fault (ALARM to FAULT state) recovery**

## 49.3.5 EOUT interface

### Introduction

You use the EOUT[1:0] signals to indicate FCCU's condition to off-chip logic.

**NOTE**

For information on the availability and names of these FCCU signals on the boundary of this chip, see the chip-specific FCCU information.

### The FCCU conditions

There are three FCCU conditions:

| Condition | Description |
|---|---|
| Faulty | All of the following are true:<br>• The fault-output (EOUT) timer is running (see How the fault-output (EOUT) timer works in Bi-Stable fault-output mode).<br>• FCCU is in FAULT state. |
| Non-faulty | All of the following are true:<br>• The fault-output (EOUT) timer is not running.<br>• FCCU is in ALARM or NORMAL state. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Condition | Description |
|---|---|
| Configuration | All of the following are true:<br>• The fault-output (EOUT) timer is not running.<br>• FCCU is in CONFIG state. |

### How the fault-output (EOUT) timer works in Bi-Stable fault-output mode

In Bi-Stable fault-output mode (FOM), FCCU starts the fault-output (EOUT) timer when all of the following are true:

- If the EOUT signals are in Bi-Stable FOM, and the EOUT signals are not programmed to be always low (CFG[FCCU_SET_CLEAR]).
- The EOUT timer is not already running.
- FCCU enters the FAULT state as the result of a fault.

When the fault-output (EOUT) timer is already running and a new fault occurs:

- If FCCU is in the CONFIG state: FCCU does not restart the EOUT timer.
- If FCCU is in the NORMAL or ALARM state:
  - And ALARM state is enabled for the fault (non-critical): FCCU enters (or remains in) the ALARM state but does not restart the EOUT timer.
  - And ALARM state is disabled for the fault (non-critical): FCCU enters the FAULT state and restarts the EOUT timer.
- If FCCU is in the FAULT state: FCCU restarts the fault-output (EOUT) timer.

FCCU stops and reinitializes the fault-output (EOUT) timer when all of the following are true:

- If the EOUT signals are in Bi-Stable fault-output mode (CFG[FOM]), and $T_{min}$ (see DELTA_T[DELTA_T]) has expired.
- All faults that caused FCCU to enter or remain in the FAULT state since FCCU started the fault-output (EOUT) timer have been cleared, causing FCCU to return to the NORMAL state.

### Prepare the EOUT signals to indicate FCCU's condition

- If the EOUT signals are in Bi-Stable fault-output mode (CFG[FOM]), ensure that the EOUT signals are controlled by FCCU's FSM (CFG[FCCU_SET_CLEAR]).
- Ensure that the EOUT signals are active (CFG[FCCU_SET_AFTER_RESET]).

**NOTE**

If the EOUT signals are in Bi-Stable fault-output mode, you must deactivate and then reactivate the EOUT signals (CFG[FCCU_SET_AFTER_RESET]) to correctly initialize them so they have opposite states.

### More about the EOUT interface

Different fault-output modes (protocols) for the fault-output (EOUT) interface are supported (CFG[FOM]):

- Bi-Stable

**NOTE**

See the chip-specific FCCU information for the fault-output modes supported by this chip.

You can further configure the fault-output modes using the following attributes:

| Attribute | Field | Setting used in the example diagrams and tables that follow |
|---|---|---|
| Configuration mode | CFG[CM] | Different |
| Polarity selection | CFG[PS] | For the faulty indication, EOUT1 is high, and EOUT0 is low. |

**EOUT frequency:** This frequency is generated by dividing the CLKSAFE frequency by a fixed factor of $2^{18}$.

$$EOUT_{freq} = \frac{CLKSAFE_{freq}}{2^{18}}$$

For example, with a CLKSAFE frequency of 16 MHz, this drives a signal of 61 Hz on EOUT.

In case of a failure event or on software request for EOUT indication, the signal(s) are set to the faulty state for a minimum time ($T_{min}$), even if software tries to release it before. If software configures the error pins to OK(1), and if a fault comes trying to drive the pin to NOK(0), then priority is given to the fault indication and the error signals indicate NOK, such as an incoming fault is not masked even when software has set the error signal to high. Also, if the error signals are forced to low by software by writing to CFG[FCCU_SET_CLEAR], then the signals shall remain low (or high) for the entire duration of $T_{min}$. During the $T_{min}$ by a non-software fault, the FCCU FSM moves independently of this signal state (low), and as soon as the timer expires, the pin behavior is dictated by the state in which the FSM finds itself in, and it is not possible to set the signals to OK by software moving FCCU to the CONFIG state, as long as this timer is running. No software intervention is needed to bring the signal from the low state.

Software can bring the pin back to OK state by clearing the faults and waiting for the $T_{min}$ interval to expire, after which the FCCU automatically enters the NORMAL state and the error signal indicates OK.

In case another failure event happens within $T_{min}$ after a first one, the $T_{min}$ counter is restarted.

#### 49.3.5.1 Bi-Stable protocol

The encoding scheme is provided in Table 230 and the related timing diagram is shown in Figure 185.

**Table 230. Bi-Stable encoding**

| Condition | EOUT[1:0] (CFG[PS] is 0) | EOUT[1:0] (CFG[PS] is 1) |
|---|---|---|
| Non-faulty | Static 01 | Static 10 |
| Faulty | Static 10 | Static 01 |
| Reset | High-impedance (no toggling)[1] | High-impedance (no toggling) |
| Configuration (CFG[CM] is 1 and CFG[FCCU_SET_AFTER_RESET] is 0) | High-impedance (no toggling)[2] | High-impedance (no toggling) |
| Configuration (When CFG[CM] is 1 and CFG[FCCU_SET_AFTER_RESET] is 1) | Static 01 | Static 10 |

1. Final value depends on device specific settings at pad level.
2. Ensure that the EOUT signals are active (CFG[FCCU_SET_AFTER_RESET]); otherwise, the EOUT signals stay in a high-impedance state after reset lifts.

**NOTE**

Figure 185 is formatted to display the behavior in all four conditions (reset, non-faulty, faulty, and configuration), not to imply transitions between one condition and another. In particular, a transition from the faulty condition to the configuration condition is not possible.

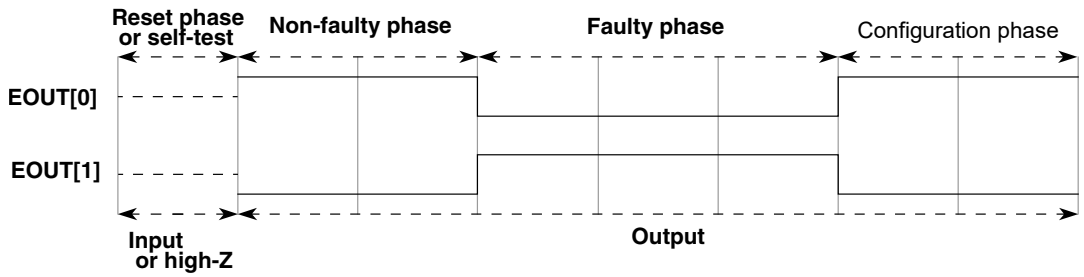



Figure 185. Bi-Stable protocol

## 49.3.6 Modes of operation

### 49.3.6.1 Put FCCU in the NORMAL state

#### 49.3.6.1.1 Introduction

You put FCCU in the NORMAL state to save changes to the configuration, and to allow FCCU to enter the ALARM or FAULT state when a fault occurs on an enabled fault channel.

#### 49.3.6.1.2 About putting FCCU in NORMAL state

When putting FCCU in the NORMAL state:

- If you attempt to lock the configuration while FCCU is in CONFIG state, FCCU does not actually lock the configuration until FCCU leaves CONFIG state—that is, either you put FCCU in the NORMAL state, or FCCU puts itself in the NORMAL state because the Configuration-state timeout interval (CFG_TO[TO]) expires.
- After you permanently lock the configuration, you must reset FCCU before you can put FCCU in the CONFIG state.

#### 49.3.6.1.3 Put FCCU in the NORMAL state

1. Check the FCCU status (STAT[STATUS]).
   - If the FCCU status is NORMAL, go to step 6.
   - If the FCCU status is CONFIG, go to step 2.
   - If the FCCU status is ALARM or FAULT, go to step 4.
2. Run the OP2 operation (see Run an operation).
3. Check the operation status (CTRL[OPS]).
   - If the operation status is Successful, FCCU is in the NORMAL state. Go to step 6.
   - If the operation status is Aborted, go to step 2.
4. Recover all faults (see Fault recovery).
5. Go to step 1.
6. Lock the configuration if you want to prevent any changes to it:
   - To require a key to unlock the configuration, from Supervisor mode, temporarily lock the configuration (TRANS_LOCK[TRANSKEY]).

- To require a reset of FCCU to unlock the configuration, from Supervisor mode, permanently lock the configuration (PERMNT_LOCK[PERMNTKEY]).

  The configuration is permanently locked until FCCU is reset.

#### 49.3.6.2 Manage faults

##### 49.3.6.2.1 Introduction

After saving changes to the configuration, you are ready to use FCCU to manage faults.

##### 49.3.6.2.2 Determine if there are any unrecovered non-critical faults

Check the unrecovered-fault indicators for the non-critical faults (NCF_Sa[NCFSn]).

##### 49.3.6.2.3 Recover a software-recoverable non-critical fault

1. Resolve the source of the software-recoverable non-critical fault.
2. Unlock the NCF_Sa registers (NCFK[NCFK]) using a 32-bit write.
3. Initiate clearing of the unrecovered-fault indicator for the software-recoverable non-critical fault (NCF_Sa[NCFSn]) using a 32-bit write.

   FCCU initiates the OP12 operation.

   **NOTE**

   If you want to clear multiple unrecovered-fault indicators and those indicators reside in different NCF_Sa registers, you must perform steps 2 and 3 for each individual register.

4. Check the operation status (CTRL[OPS]).
   - If the operation status is In Progress, go to step 4.
   - If the operation status is Successful, go to step 5.
   - If the operation status is Aborted, go to step 2.
5. Check the unrecovered-fault indicator for the software-recoverable non-critical fault (NCF_Sa[NCFSn]).
   - If the indicator indicates no unrecovered fault, the fault has been recovered. Stop.
   - If the indicator still indicates an unrecovered fault, go to step 2.

##### 49.3.6.2.4 Clear the freeze-status indicators

1. Run the OP13 operation (see Run an operation).
2. Check the operation status (CTRL[OPS]).
   - If the operation status is Successful, stop .
   - If the operation status is Aborted, go to step 1.

#### 49.3.6.3 Run operations

##### 49.3.6.3.1 Introduction

You run operations to perform actions such as putting FCCU in the CONFIG state or setting the operation status to Idle. For a complete list of operations you can run, see CTRL[OPR].

#### 49.3.6.3.2 About running operations

When running operations:

- FCCU ignores any operations initiated while the operation status is In Progress.
- Certain operations must be unlocked before you can initiate them. After you initiate them, they are locked again.

#### 49.3.6.3.3 Run an operation

1. Check the operation status (CTRL[OPS]).
2. Go to step 1 if the operation status is In Progress.
3. Unlock the operation (CTRLK[CTRLK]) using a 32-bit write, if the operation must be unlocked before you can initiate it.

   **NOTE**

   The Control Key (CTRLK) register used in this step and the Control (CTRL) register used in the next step must be written with consecutive instructions. Do not use read-modify-write instructions, such as bit-field instructions, to modify these registers.

4. Initiate the operation (CTRL[OPR]) using a 32-bit write.
5. Check the operation status (CTRL[OPS]).
6. Go to step 5 if the operation status is In Progress.

The operation status is now Idle (OP15 only), Aborted, or Successful.

### 49.3.7 FOSU

The FOSU provides a supervision of the primary fault notification path by analyzing FCCU's behavior for correctness. It waits for any reaction of the FCCU in a fixed time window after a fault is signaled.

The intention of the FOSU is to provide a secondary fault reaction path in most cases when the FCCU fails but not to needlessly propagate a fault which is already handled by the FCCU in a full chip reset. Only a failed primary fault reaction (that is, FCCU's failure) is a reason for the secondary reaction to take over (and generate a destructive reset request).

There is a 'do nothing' input coming from FCCU which indicates that the FCCU is programmed for no reaction for ALL FAULTS. It is a "static" input in the sense that it does not change after FCCU configuration. The FOSU masks the incoming faults with the 'do nothing' control from the FCCU, meaning that a fault is not captured by the FOSU if the 'do nothing' signal is asserted (that is, a disabled fault). There is no minimum pulse width requirement on the fault indication other than what is required by the technology, which is the same as that of the FCCU. FOSU does not monitor FCCU for the case of faults occurring during the CONFIG state.

The FOSU contains a timer with a duration of FOSU_COUNT, driven by CLKSAFE. The timer is initialized and started on any captured, enabled fault. While the timer is running, any subsequent captured fault will neither restart nor reinitialize the timer. The timer is stopped when the FCCU shows any of the following reactions (the FOSU does not check whether the reaction is the configured one for the faults which occurred):

- Reset: short functional reset
- IRQ (triggered by ALARM state)
- NMI
- Error out triggered (by FCCU or by SW)

When the timer is stopped, the fault capture logic is cleared to ensure that the timer is not restarted because of faults still 'stuck' in the capture logic. The timer is then restarted by the next new failure indication. When the timer expires, the FOSU's failure indicator output is asserted after it ensures that the fault is enabled and the static "fccu program to do nothing" signal is deasserted. This is because FCCU uses settings after it exits CONFIG state, even if fault captured before the exit.

The FOSU's failure indicator output is connected to one of the MC_RGM's 'destructive' reset inputs, so its assertion will cause a reset sequence to be initiated starting at DEST0. The FOSU module is reset with the same reset as is used by the FCCU. When

this reset is asserted, the FOSU's capture logic is cleared, its timer is kept stopped and in a non-expired state, and its failure indicator output is deasserted.

**NOTE**

FOSU is triggered on assertion of enabled fault. In case the triggering fault is disabled, FOSU times out without reaction.

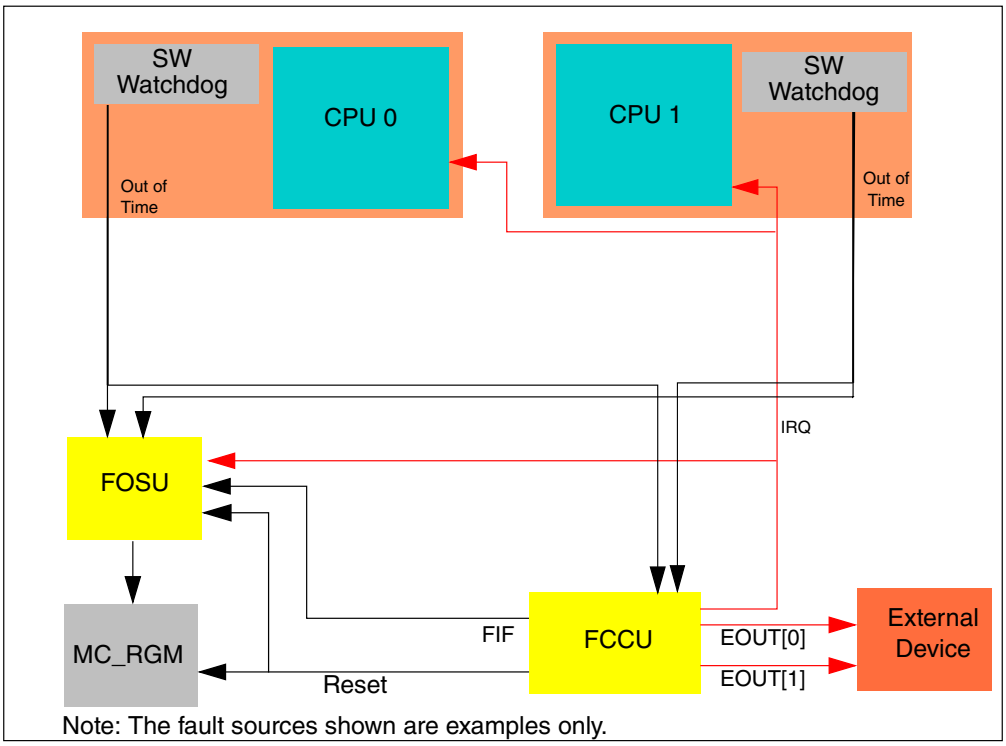



**Figure 186. FOSU connections to the FCCU and MC_RGM**

### 49.3.8 Clocking

This module has no clocking considerations.

### 49.3.9 Interrupts

This module has no interrupts.

## 49.4 External signals

FCCU interfaces with the EOUT pin. Signaling on the EOUT pin depends on whether the module is processing an error or is idle.

## 49.5 Initialization

### 49.5.1 Prepare FCCU for configuration

#### 49.5.1.1 Introduction

You prepare FCCU for configuration by first configuring the CONFIG state and then putting FCCU in that state.

#### 49.5.1.2 About preparing FCCU for configuration

When preparing FCCU for configuration, keep the following in mind:

- To put FCCU in CONFIG state, FCCU must be in NORMAL state.
- After FCCU is reset, the configuration is temporarily locked. You must temporarily unlock the configuration before you can put FCCU in the CONFIG state.
- When FCCU is in the CONFIG state, FCCU does not actually save the changes you make to the configuration. To save changes to the configuration, you must manually put FCCU in the NORMAL state. If FCCU automatically leaves the CONFIG state and enters the NORMAL state because the Configuration-state timeout interval (CFG_TO[TO]) expires (called a Configuration-state timeout), FCCU changes the value of the Configuration (CFG) register to its Configuration-state-timeout value and the value of each of the other configuration registers to its reset value. FCCU also changes the value of the Configuration-State Timeout Interval (CFG_TO) register to its reset value. For information on the Configuration-state timeout value, see CFG register bit value at different events. For a list of configuration registers, see Configuration registers.

#### 49.5.1.3 Configure the CONFIG state

1. Set the Configuration-state timeout interval (CFG_TO[TO]).
2. Enable the Configuration-state-timeout interrupt signal (IRQ_EN[CFG_TO_IEN]), if you want FCCU to request an interrupt when a Configuration-state timeout occurs.

#### 49.5.1.4 Put FCCU in the CONFIG state

1. Unlock the configuration temporarily (TRANS_LOCK[TRANSKEY]) in Supervisor mode.
2. Check the FCCU status (STAT[STATUS]).
   - If the FCCU status is CONFIG, FCCU is in the CONFIG state. Stop.
   - If the FCCU status is NORMAL, go to step 3.
   - If the FCCU status is ALARM or FAULT, go to step 5.
3. Run the OP1 operation (see Run an operation).
4. Check the operation status (CTRL[OPS]).
   - If the operation status is Successful, FCCU is in the CONFIG state. Stop.
   - If the operation status is Aborted, the configuration is probably permanently locked. Go to step 7.
5. Recover all faults (see Fault recovery).
6. Go to step 2.
7. Reset FCCU.

### 49.5.2 Configure FCCU

#### 49.5.2.1 Introduction

You configure FCCU so it functions according to the needs of your particular application.

#### 49.5.2.2 About configuring FCCU

When configuring FCCU:

- If you enable a non-critical fault channel but disable all reactions for that channel, FCCU changes state when necessary but does not perform any reaction because reactions are disabled. If you enable reactions for a non-critical fault channel but disable that channel, and FCCU is in the NORMAL state when a fault occurs on the channel, FCCU does not enter the ALARM or FAULT state and therefore does not perform any reaction.

#### 49.5.2.3 Configure the non-critical fault channels

For each non-critical fault channel that you want FCCU to monitor:

1. Set the recovery type (NCF_CFGa[NCFCn]).
2. Enable at least one type of Fault-state reaction:
   - Chip functional reset (NCFS_CFGa[NCFSCn])
   - Non-maskable interrupt (NMI_ENa[NMIENn])
   - EOUT signaling (EOUT_SIG_ENa[EOUTENn])

   **NOTE**

   If you enable the chip functional reset as the type of Fault-state reaction for a channel, enable at least one other type of Fault-state reaction for the channel or enable the ALARM state (step 4) for the channel.
3. Set the Alarm-state timeout interval (NCF_TO[TO]), if you plan to enable the ALARM state for any non-critical fault channel.

   **NOTE**

   Ensure that the Alarm-state timeout interval is less than the FOSU module's timeout interval; otherwise, FOSU generates a chip reset every time a fault occurs on the channel. The FOSU timeout interval (FOSU_COUNT) is a chip-specific value. See the chip-specific FCCU information.
4. Enable the ALARM state (NCF_TOEa[NCFTOEn]) for any non-critical fault channel for which you want FCCU to enter the ALARM state before entering the FAULT state.
5. Enable the Alarm-state reaction (IRQ_ALARM_ENa[IRQENn]) for each non-critical fault channel for which you enabled the ALARM state.
6. Enable the corresponding NCF_Ea[NCFEn] field for each non-critical fault channel that you want FCCU to monitor.

## 49.6 Application information

### 49.6.1 Use cases and limitations

#### Configuration guidelines

Follow these guidelines to configure FCCU:

- If you want FCCU to react to a fault on a non-critical fault channel:
  - Enable the channel (Non-critical Fault Enable (NCF_E0)).
  - Enable at least one type of Fault-state reaction for the channel: chip reset (Non-critical Fault-State Configuration (NCFS_CFG0)), fault-output (EOUT) signaling (Non-critical Fault-State EOUT Signaling Enable (EOUT_SIG_EN0)), or non-maskable interrupt (Non-critical Fault-State Non-maskable-Interrupt-Request Enable (NMI_EN0)).
  - If you enable chip reset as the type of Fault-state reaction for the channel (Non-critical Fault-State Configuration (NCFS_CFG0)), enable either ALARM state (Non-critical-Fault Alarm-State Timeout Enable (NCF_TOE0)) or at least one other type of Fault-state reaction for the channel: fault-output (EOUT) signaling (Non-critical Fault-State EOUT Signaling Enable (EOUT_SIG_EN0)) or non-maskable interrupt (Non-critical Fault-State Non-maskable-Interrupt-Request Enable (NMI_EN0)).
  - If you enable ALARM state for the channel (Non-critical-Fault Alarm-State Timeout Enable (NCF_TOE0)), enable the Alarm-state reaction (Non-critical Alarm-State Interrupt-Request Enable (IRQ_ALARM_EN0)).
  - If you enable ALARM state for the channel (Non-critical-Fault Alarm-State Timeout Enable (NCF_TOE0)), make sure the Alarm-state timer interval (Non-critical-Fault Alarm-State Timeout Interval (NCF_TO)) is less than the FOSU module's timer interval; otherwise, FOSU generates a chip reset every time a fault occurs on the channel. The FOSU timer interval (FOSU_COUNT) is chip-specific. See the chip-specific FCCU information.

**Recommendations to configure FCCU**

1. After a power on, or 'destructive' reset (when initiated by the assertion of the chip reset pin, RESET_B), where both system and FCCU are reset, the following steps could be followed to configure FCCU:
    a. Check and clear any pending fault status
    b. Verify FCCU is in NORMAL state, else repeat step(a) above
    c. Configure FCCU
2. After any 'functional' reset of the system, arising out of a reset request from FCCU or other sources, the following steps could be followed to reconfigure FCCU:
    a. If active, wait for the Error out $T_{min}$ to expire
    b. Check and clear fault status
    c. Error pin moves to "non faulty" state, once fault status is cleared and $T_{min}$ expires
    d. Verify FCCU is in NORMAL state, else repeat step(a) above
    e. Read and verify value in NCF_En
    f. Reconfigure FCCU, if necessary

## 49.7 Register descriptions

### 49.7.1 FCCU register descriptions

The FCCU registers are listed in the table below. Any address offset not explicitly mentioned in this table is reserved.

The FCCU supports word (32-bit), half-word (16-bit), and byte (8-bit) read and write accesses.

Follow these register-access guidelines:

- Do not read from or write to any addresses that are not shown in the following table. Doing so may or may not result in a transfer error.
- Do not write to any of the configuration registers unless FCCU is in the CONFIG state. Doing so results in a transfer error.
- Do not write to the Transient Configuration Lock (TRANS_LOCK) or Permanent Configuration Lock (PERMNT_LOCK) registers unless your code runs in the Supervisor mode. Doing so results in a transfer error.

For each possible NCF failure source, a different reaction—including no reaction—is configurable through the use of NMI, IRQ, and short reset selection registers. It is not possible for a single event upset to switch off all reactions on failures as implementation is per fault source (but it will be possible to switch them all off by SW if intended). Failures themselves are not able to disable all reactions and indications.

#### 49.7.1.1 FCCU memory map

FCCU base address: 4038_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Control (CTRL) | 32 | RW | 0000_00C0h |
| 4h | Control Key (CTRLK) | 32 | W | 0000_0000h |
| 8h | Configuration (CFG) | 32 | RW | 0000_0000h |
| 1Ch | Non-critical Fault Configuration (NCF_CFG0) | 32 | RW | 0000_00FFh |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 4Ch | Non-critical Fault-State Configuration (NCFS_CFG0) | 32 | RW | 0000_0000h |
| 80h | Non-critical Fault Status (NCF_S0) | 32 | RW | 0000_0000h |
| 90h | Non-critical Fault Key (NCFK) | 32 | RW | 0000_0000h |
| 94h | Non-critical Fault Enable (NCF_E0) | 32 | RW | 0000_0000h |
| A4h | Non-critical-Fault Alarm-State Timeout Enable (NCF_TOE0) | 32 | RW | 0000_00FFh |
| B4h | Non-critical-Fault Alarm-State Timeout Interval (NCF_TO) | 32 | RW | 0003_A980h |
| B8h | Configuration-State Timeout Interval (CFG_TO) | 32 | RW | 0000_0005h |
| BCh | IO Control (EINOUT) | 32 | RW | See section |
| C0h | Status (STAT) | 32 | R | 0000_0010h |
| C4h | Normal-to-Alarm Freeze Status (N2AF_STATUS) | 32 | R | 0000_0000h |
| C8h | Alarm-to-Fault Freeze Status (A2FF_STATUS) | 32 | R | 0000_0000h |
| CCh | Normal-to-Fault Freeze Status (N2FF_STATUS) | 32 | R | 0000_0000h |
| D0h | Fault-to-Alarm Freeze Status (F2AF_STATUS) | 32 | R | 0000_0000h |
| DCh | Non-critical Fault Fake (NCFF) | 32 | W | 0000_0000h |
| E0h | IRQ Status (IRQ_STAT) | 32 | RW | 0000_0000h |
| E4h | IRQ Enable (IRQ_EN) | 32 | RW | 0000_0000h |
| F0h | Transient Configuration Lock (TRANS_LOCK) | 32 | W | 0000_0000h |
| F4h | Permanent Configuration Lock (PERMNT_LOCK) | 32 | W | 0000_0000h |
| F8h | Delta T (DELTA_T) | 32 | RW | 0000_0000h |
| FCh | Non-critical Alarm-State Interrupt-Request Enable (IRQ_ALARM_EN0) | 32 | RW | 0000_0000h |
| 10Ch | Non-critical Fault-State Non-maskable-Interrupt-Request Enable (NMI_EN0) | 32 | RW | 0000_0000h |
| 11Ch | Non-critical Fault-State EOUT Signaling Enable (EOUT_SIG_EN0) | 32 | RW | 0000_0000h |
| 12Ch | Alarm-State Timer (TMR_ALARM) | 32 | R | 0003_A980h |
| 134h | Configuration-State Timer (TMR_CFG) | 32 | R | 000F_FFFFh |
| 138h | Fault-Output Timer (TMR_ETMR) | 32 | R | 0000_0000h |

### 49.7.1.2 Control (CTRL)

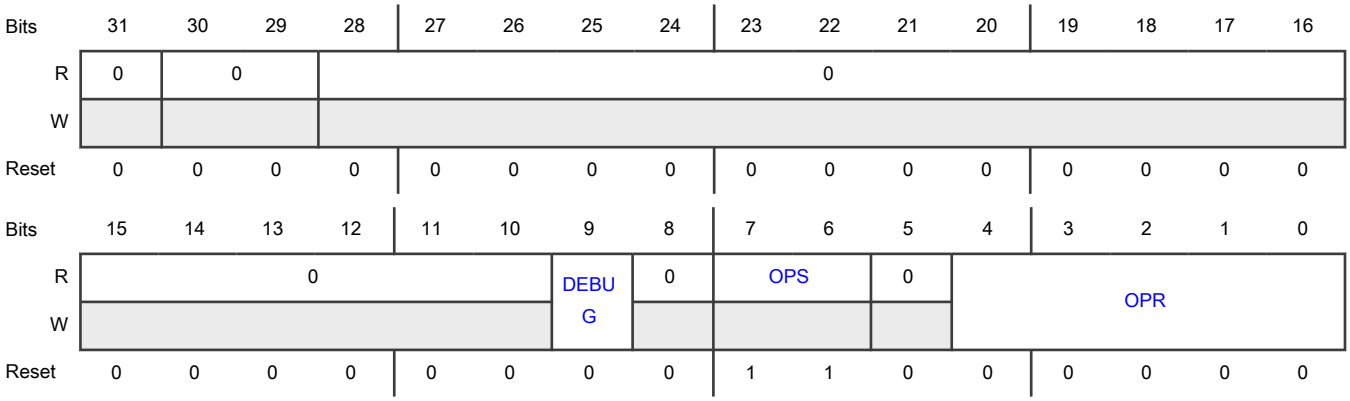
#### Offset

| Register | Offset |
|---|---|
| CTRL | 0h |

#### Function

Initiates and indicates the status of operations—and enables the Debug mode.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-29<br>— | Reserved |
| 28-10<br>— | Reserved |
| 9<br>DEBUG | Debug Mode Enable<br>Specifies whether the Debug mode is enabled. If so, FCCU enters the Debug mode when the Debug signal is asserted. When FCCU enters the Debug mode, it halts operation and remains in the state it was in before it entered this mode.<br>**NOTE**<br>FOSU does not halt when FCCU enters the Debug mode. Therefore, FOSU can still cause a reset if a fault occurs while FCCU is in the Debug mode.<br>0b - Disabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Enabled |
| 8<br>— | Reserved |
| 7-6<br>OPS | Operation Status<br>This field can be read and cleared (via OP15 operation) by the software.<br>00b - Idle<br>01b - In progress<br>10b - Aborted<br>11b - Successful |
| 5<br>— | Reserved |
| 4-0<br>OPR | Operation Run<br>Initiates operations that perform actions such as putting FCCU in the CONFIG state or setting the operation status to Idle. For information on how to run operations, see Run operations.<br>FCCU ignores any write to this field while the operation status (CTRL[OPS]) is "In progress". After completion of an operation, FCCU sets this field to OP0.<br>The following events result in an operation status (CTRL[OPS]) of "Aborted":<br>• Writing to a NCF_Sa register (which automatically initiates the OP12 operation) without first successfully unlocking the register (NCFK[NCFK])<br>• Initiating an OP1 operation when FCCU is not in the NORMAL state or the configuration is locked<br>• Initiating an OP1, or OP2, operation without first unlocking the operation (CTRLK[CTRLK])<br>00000 OP0—No operation<br>00001 OP1—Applies only when the configuration is unlocked, when FCCU is in the NORMAL state, and immediately after you unlock the operation (CTRLK[CTRLK]). Put FCCU in the CONFIG state.<br>00010 OP2—Applies only immediately after you unlock the operation (CTRLK[CTRLK]). Put FCCU in the NORMAL state.<br>00011 Reserved<br>00100 Reserved<br>00101 Reserved<br>00110 Reserved<br>00111 Reserved<br>01000 Reserved<br>01001 Reserved<br>01010 Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 01011 Reserved<br>01100 OP12—Do not initiate this operation; it is automatically initiated by the FCCU. A NCF_Sa register status clear operation is in progress.<br>01101 OP13—Clear the freeze status registers.<br>01110 OP14—Do not initiate this operation; it is automatically initiated by the FCCU. A Configuration-state timeout is in progress. For more information, see Configuration registers.<br>01111 OP15—Set the operation status (CTRL[OPS]) to Idle.<br>10000 Reserved<br>10001 Reserved<br>10010 Reserved<br>10011 Reserved<br>10100 Reserved<br>10101—11110 Forbidden. Writing any of these values returns an operation status (CTRL[OPS]) of "Aborted" with no side effect.<br>11111 Reserved |

### 49.7.1.3 Control Key (CTRLK)

#### Offset

| Register | Offset |
|---|---|
| CTRLK | 4h |

#### Function

See CTRLK[CTRLK].

#### Diagram

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

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

**Fields**

| Field | Function |
|---|---|
| 31-0<br>CTRLK | Locked-Operation Control Key<br>Writable only with a 32-bit write. Unlocks locked operations (CTRL[OPR]) so you can initiate them. For information on how to unlock locked operations before you initiate them, see Run an operation.<br>**NOTE**<br>• You must initiate an operation in the FCCU register access that immediately follows the one that unlocks it; otherwise, the operation is again locked.<br>• Reading from this register always returns the value 0000_0000h.<br>Operations not listed here are not locked and do not need to be unlocked.<br>9137_56AFh: Unlock OP1.<br>825A_132Bh: Unlock OP2.<br>Any other value: Do nothing. |

### 49.7.1.4 Configuration (CFG)

**Offset**

| Register | Offset |
|---|---|
| CFG | 8h |

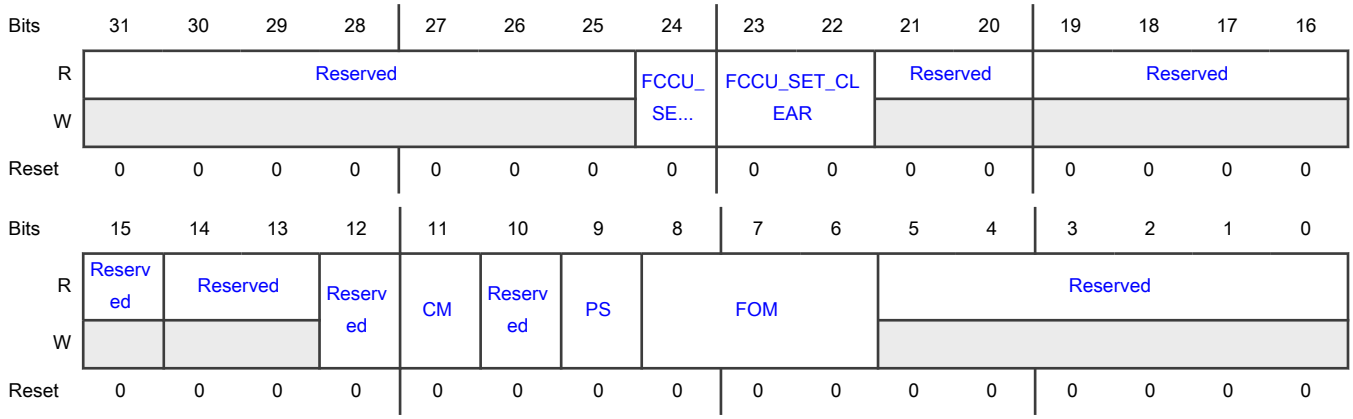
**Function**

Writable only when FCCU is in the CONFIG state. Changed by FCCU to another value when the chip resets FCCU, a Configuration-state timeout occurs, or you run an OP31 operation. See CFG register bit value at different events for more information. Specifies the global configuration for FCCU.

**NOTE**

If you specify a new value for any of the fields in this register that affect the EOUT signals while the fault-output (EOUT) timer is running (FCCU is indicating a fault on the EOUT signals), FCCU does not use the new settings you specified until after the fault-output (EOUT) timer expires (FCCU stops indicating a fault on the EOUT signals).

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved |
| 24<br>FCCU_SET_AFTER_RESET | Fault-Output (EOUT) Activate<br>For fault-output (EOUT) signaling, controls whether the EOUT signals are active.<br>0b - Inactive (the EOUT signals are in a high-impedance state)<br>1b - Active (the EOUT signals indicate FCCU's condition) |
| 23-22<br>FCCU_SET_CLEAR | Fault-Output (EOUT) Control<br>Applies only to Bi-Stable fault-output mode (CFG[FOM]) and when the EOUT signals are active (CFG[FCCU_SET_AFTER_RESET]). Controls whether the fault-output (EOUT) signals are managed by FCCU's FSM.<br>00b - Controlled by the FSM<br>01b - Always low<br>10b - Controlled by the FSM<br>11b - High until a fault occurs on a channel, regardless of whether that fault is disabled; thereafter, controlled by the FSM. Note: FCCU ignores an attempt to write this value if the fault-output (EOUT) timer is already running. |
| 21-20<br>— | Reserved |
| 19-16<br>— | Reserved |
| 15<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 14-13<br>— | Reserved |
| 12<br>— | Reserved<br>Always write the reset value to this field. |
| 11<br>CM | Fault-Output (EOUT) Configuration-Indication Mode<br>For fault-output (EOUT) signaling, this field controls whether the configuration indication is the same as the non-faulty indication.<br>0b - Different<br>1b - Same |
| 10<br>— | Reserved |
| 9<br>PS | Fault-Output (EOUT) Polarity Selection<br>Applies to fault-output (EOUT) signaling and controls the polarity of the signals for fault-output mode indications that hold the signals low or high (versus toggling them or placing them in a high-impedance state). Applies only to Bi-Stable fault-output mode (for all indications).<br>0b - For the faulty indication, EOUT1 is high, and EOUT0 is low.<br>1b - For the faulty indication, EOUT1 is low, and EOUT0 is high. |
| 8-6<br>FOM | Fault-Output (EOUT) Mode<br>For fault-output (EOUT) signaling, controls the protocol of the signaling.<br>000b - Reserved<br>001b - Reserved<br>010b - Bi-Stable<br>011b - Reserved<br>100b - Reserved<br>101b - Test 0 (controlled by the EINOUT register; EOUT1 is an output; EOUT0 is an input)<br>110b - Test 1 (controlled by the EINOUT register; EOUT1 and EOUT0 are both outputs)<br>111b - Test 2 (controlled by the EINOUT register; EOUT1 is an input; EOUT0 is an output) |
| 5-0<br>— | Reserved |

### 49.7.1.5 Non-critical Fault Configuration (NCF_CFG0)

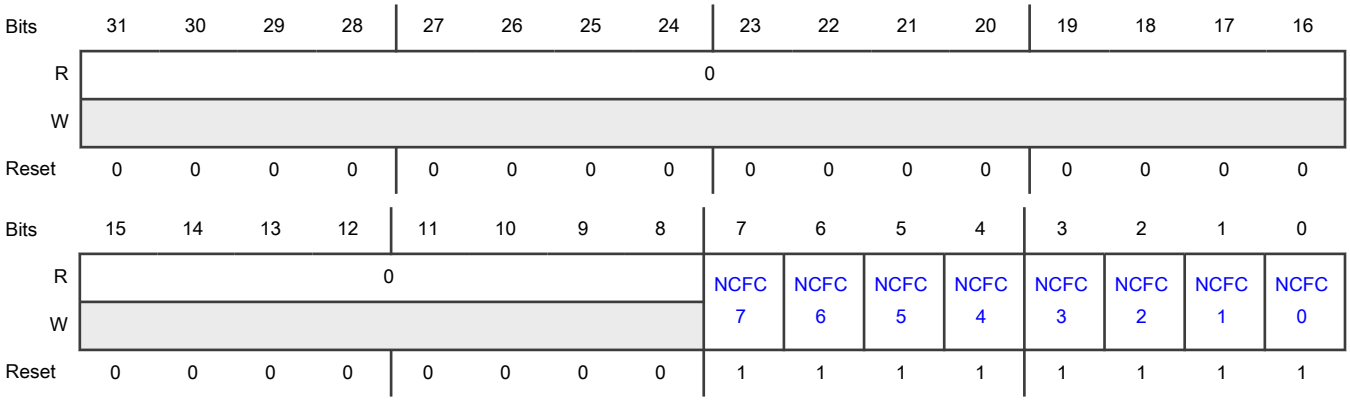
#### Offset

| Register | Offset |
|---|---|
| NCF_CFG0 | 1Ch |

#### Function

See NCF_CFGa[NCFCn].

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>NCFCn | Non-critical Fault Configuration n<br>Writable only when FCCU is in the CONFIG state. Changed by FCCU to its reset value when a Configuration-state timeout occurs. Controls the recovery type (HW or SW) of the associated non-critical fault channel (n). For information on how to configure the non-critical fault channels, see Configure the non-critical fault channels.<br>**NOTE**<br>Configure a non-critical fault channel as hardware-recoverable only if the source continues to indicate a fault on the fault channel's input (NCFn) until the condition that caused the fault is no longer true; otherwise, configure the non-critical fault channel as software-recoverable.<br>0b - Hardware-recoverable<br>1b - Software-recoverable |

### 49.7.1.6 Non-critical Fault-State Configuration (NCFS_CFG0)

#### Offset

| Register | Offset |
|---|---|
| NCFS_CFG0 | 4Ch |

#### Function

See NCFS_CFGa[NCFSCn].

#### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | NCFSC7 | | NCFSC6 | | NCFSC5 | | NCFSC4 | | NCFSC3 | | NCFSC2 | | NCFSC1 | | NCFSC0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-14: NCFSC7<br>13-12: NCFSC6<br>11-10: NCFSC5<br>9-8: NCFSC4<br>7-6: NCFSC3<br>5-4: NCFSC2<br>3-2: NCFSC1<br>1-0: NCFSC0 | Non-critical Fault-State Configuration n<br>Writable only when FCCU is in the CONFIG state. Changed by FCCU to its reset value when a Configuration-state timeout occurs. Controls whether the chip functional reset is enabled as a Fault-state reaction for the associated non-critical fault channel (n). When the chip functional reset is enabled for an enabled non-critical fault channel, a fault on that channel causes FCCU to assert the rst_sfunc_b signal when FCCU enters the FAULT state. For information on how to configure the non-critical fault channels, see Configure the non-critical fault channels.<br>00b - Disabled<br>01b - Enabled (rst_sfunc_b) (short)<br>10b - Reserved<br>11b - Disabled |

### 49.7.1.7 Non-critical Fault Status (NCF_S0)

#### Offset

| Register | Offset |
|---|---|
| NCF_S0 | 80h |

#### Function

See NCF_Sa[NCFSn].

#### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | NCFS7 | NCFS6 | NCFS5 | NCFS4 | NCFS3 | NCFS2 | NCFS1 | NCFS0 |
| W | | | | | | | | | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>NCFSn | Non-critical Fault Status n<br>Indicates whether there is an unrecovered fault on the associated non-critical fault channel (n).<br>**NOTE**<br>To recover a software-recoverable non-critical fault, which includes clearing its unrecovered-fault indicator, see Recover a software-recoverable non-critical fault. FCCU clears the unrecovered-fault indicator for a hardware-recoverable non-critical fault automatically when the source no longer indicates a fault on the fault channel's input signal; if you attempt to clear the unrecovered-fault indicator for a hardware-recoverable non-critical fault, FCCU does not clear the indicator and does not indicate an error.<br>0b - No unrecovered fault<br>1b - Unrecovered fault |

### 49.7.1.8 Non-critical Fault Key (NCFK)

#### Offset

| Register | Offset |
|---|---|
| NCFK | 90h |

#### Function

See NCFK[NCFK].

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-0<br>NCFK | Non-critical Fault Key<br>Writable only with a 32-bit write. Unlocks the NCF_Sa registers so you can write to them while recovering a software-recoverable non-critical fault. For information on how to unlock the NCF_Sa registers before writing to them, see Recover a software-recoverable non-critical fault.<br>**NOTE**<br>• You must write to one of the NCF_Sa registers immediately after unlocking it (that is, in the FCCU register access that immediately follows the one that unlocks them); otherwise the registers are again locked. If you want to write to multiple NCF_Sa registers, you must unlock each register immediately before you write to it.<br>• Reading from this register always returns the value 0000_0000h.<br>AB34_98FEh: Unlock.<br>Any other value: Do nothing. |

### 49.7.1.9 Non-critical Fault Enable (NCF_E0)

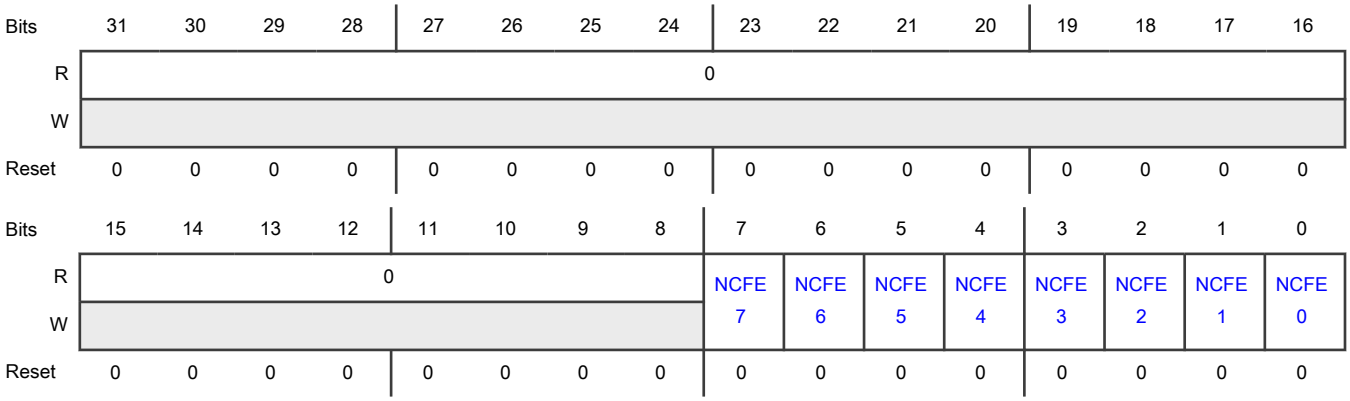
#### Offset

| Register | Offset |
|---|---|
| NCF_E0 | 94h |

#### Function

See NCF_Ea[NCFEn].

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>NCFEn | Non-critical Fault Enable n<br>Writable only when FCCU is in the CONFIG state. Changed by FCCU to its reset value when a Configuration-state timeout occurs. Controls whether the associated non-critical fault channel (n) is enabled. When a non-critical fault channel is enabled, a fault on that channel causes FCCU to leave the NORMAL state and enter the FAULT state (or ALARM state if enabled for the channel). For information on how to configure the non-critical fault channels, see Configure the non-critical fault channels.<br>0b - Disabled<br>1b - Enabled |

### 49.7.1.10 Non-critical-Fault Alarm-State Timeout Enable (NCF_TOE0)

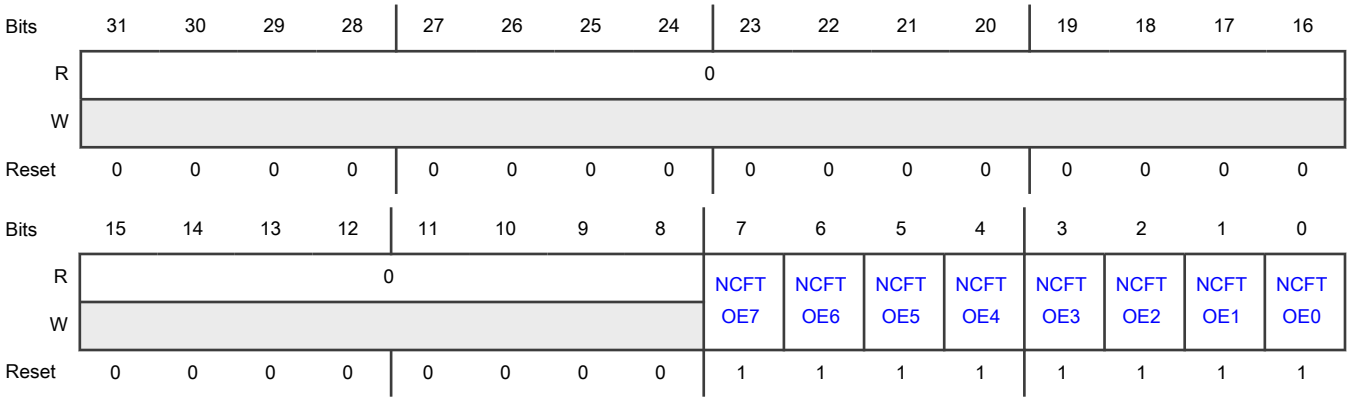
#### Offset

| Register | Offset |
|---|---|
| NCF_TOE0 | A4h |

### Function

See NCF_TOEa[NCFTOEn].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>NCFTOEn | Non-critical-Fault Alarm-State Timeout Enable n<br>Writable only when FCCU is in the CONFIG state. Changed by FCCU to its reset value when a Configuration-state timeout occurs. Controls whether the ALARM state is enabled for the associated non-critical fault channel (n). When the ALARM state is enabled for an enabled non-critical fault channel, a fault on that channel causes FCCU to leave the NORMAL state and enter the ALARM state instead of FAULT state. If the fault is not recovered within the Alarm-state timeout interval, then FCCU leaves the ALARM state and enters the FAULT state. For information on how to configure the non-critical fault channels, see Configure the non-critical fault channels.<br>0b - Disabled<br>1b - Enabled |

## 49.7.1.11 Non-critical-Fault Alarm-State Timeout Interval (NCF_TO)

### Offset

| Register | Offset |
|---|---|
| NCF_TO | B4h |

### Function

See NCF_TO[TO]

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TO | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>TO | Non-critical-Fault Alarm-State Timeout Interval<br>Writable only when FCCU is in the CONFIG state. Changed by FCCU to its reset value when a Configuration-state timeout occurs. Controls the maximum amount of time that FCCU can be in the ALARM state ($T_{Max\ Alarm}$) according to this equation:<br>$T_{Max\ Alarm} = TO \times T_{CLKSAFE}$<br>where $T_{CLKSAFE}$ is the safe-clock period.<br>If FCCU enters the ALARM state (because a fault occurs on an enabled non-critical fault channel for which the ALARM state is enabled) and this timeout interval expires (called an Alarm-state timeout), then FCCU leaves the ALARM state and enters the FAULT state.<br>**NOTE**<br>Make sure the Alarm-state timeout interval is less than the FOSU module's timeout interval; otherwise, a fault that occurs while FCCU is in the ALARM state can cause FOSU to generate a chip reset. The FOSU timeout interval (FOSU_COUNT) is chip-specific. See the chip-specific FCCU information. |

## 49.7.1.12 Configuration-State Timeout Interval (CFG_TO)

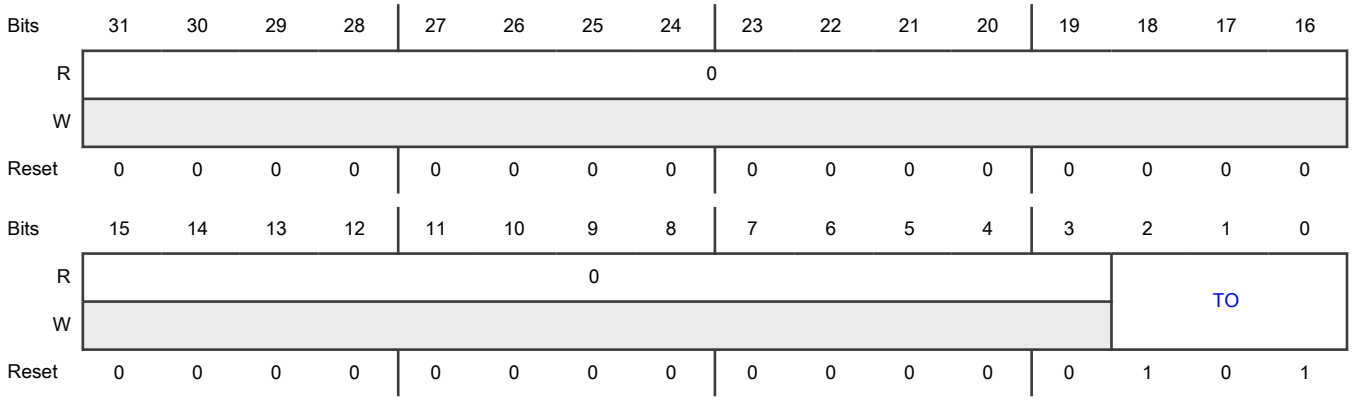
### Offset

| Register | Offset |
|---|---|
| CFG_TO | B8h |

### Function

See CFG_TO[TO]

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2-0<br>TO | Configuration-State Timeout Interval<br>Writable only when FCCU is in the NORMAL, ALARM, or FAULT state (not in the CONFIG state). Changed by FCCU to its reset value when a Configuration-state timeout occurs. Not accessible while a Configuration-state timeout (OP14 operation) is in progress. Controls the maximum amount of time that FCCU can be in the CONFIG state ($T_{\text{Max configuration}}$) according to this equation:<br>$T_{\text{Max configuration}} = T_{\text{CLKSAFE}} \times 2^{(\text{TO} + 13)}$<br>where $T_{\text{CLKSAFE}}$ is the safe-clock period.<br>If you put FCCU in the CONFIG state and this timeout interval expires (called a Configuration-state timeout), then FCCU:<br>• Automatically leaves the CONFIG state and enters the NORMAL state<br>• Changes the value of the Configuration (CFG) register to its Configuration-state-timeout value and the value of each of the other configuration registers to its reset value. For information on the Configuration (CFG) register's Configuration-state-timeout value, see CFG register bit value at different events. For a list of configuration registers, see Configuration registers. |

## 49.7.1.13 IO Control (EINOUT)

### Offset

| Register | Offset |
|---|---|
| EINOUT | BCh |

### Function

The EINOUT register allows the following operations typically in the NORMAL state:

- To control the EOUT[1] output level when the FCCU is configured in "Test1" or "Test0" fault output mode (CFG[FOM])

- To control the EOUT[0] output level when the FCCU is configured in "Test1" or "Test2" fault output mode (CFG[FOM])
- to observe the state of signals at EIN[1:0] pins

The following table shows Bi-Stable encoding.

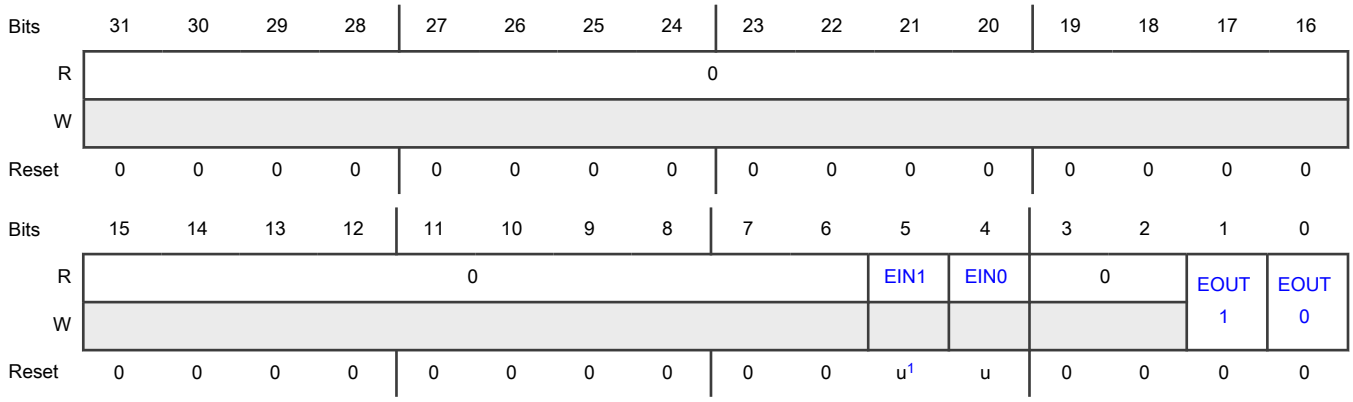
**Table 231. Bi-Stable encoding**

| Mode = CFG[FOM] | EOUT[0] | EOUT[1] |
|---|---|---|
| Test1 | output | output |
| Test2 | output | input |
| Test0 | input | output |

**NOTE**

Because of the resynchronization stage of the EOUT interface, there is a latency of a few CLKSAFE cycles following a write/read operation of the EINOUT register.

### Diagram




1. Reset value varies as per the corresponding EIN signal.

### Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved |
| 5<br>EIN1 | Error Input 1<br>Indicates the state of the EIN1 signal.<br>**NOTE**<br>When IP's SET_AFTER_RESET bit is 0, then corresponding pad is in High-Z state, in this case values sampled on EIN1 will depend on board/pad pull up/down values.<br>0b - Low<br>1b - High |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 4<br>EIN0 | Error Input 0<br>Indicates the state of the EIN0 signal.<br>**NOTE**<br>When IP's SET_AFTER_RESET bit is 0, then corresponding pad is in High-Z state, in this case values sampled on EIN0 will depend on board/pad pull up/down values.<br>0b - Low<br>1b - High |
| 3-2<br>— | Reserved |
| 1<br>EOUT1 | EOUT1<br>Error out 1 (significant only if the CFG.FOM = Test1 or Test0 => EOUT[1] configured in output mode). The EOUT1 set/clear the respective EOUT[1] output signal if CFG.FOM = 110 or 101, otherwise it is a "don't-care" value.<br>**NOTE**<br>When the configuration watchdog timer expires, FCCU changes the value of this field to its reset value.<br>0b - force EOUT[1] = 0<br>1b - force EOUT[1] = 1 |
| 0<br>EOUT0 | EOUT0<br>Error out 0 (significant only if the CFG.FOM = Test1 or Test2 => EOUT[0] configured in output mode). The EOUT0 set/clear the respective EOUT[0] output signal if CFG.FOM = 110 or 111, otherwise it is a "don't care" value.<br>**NOTE**<br>When the configuration watchdog timer expires, FCCU changes the value of this field to its reset value.<br>0b - force EOUT[0] = 0<br>1b - force EOUT[0] = 1 |

### 49.7.1.14 Status (STAT)

**Offset**

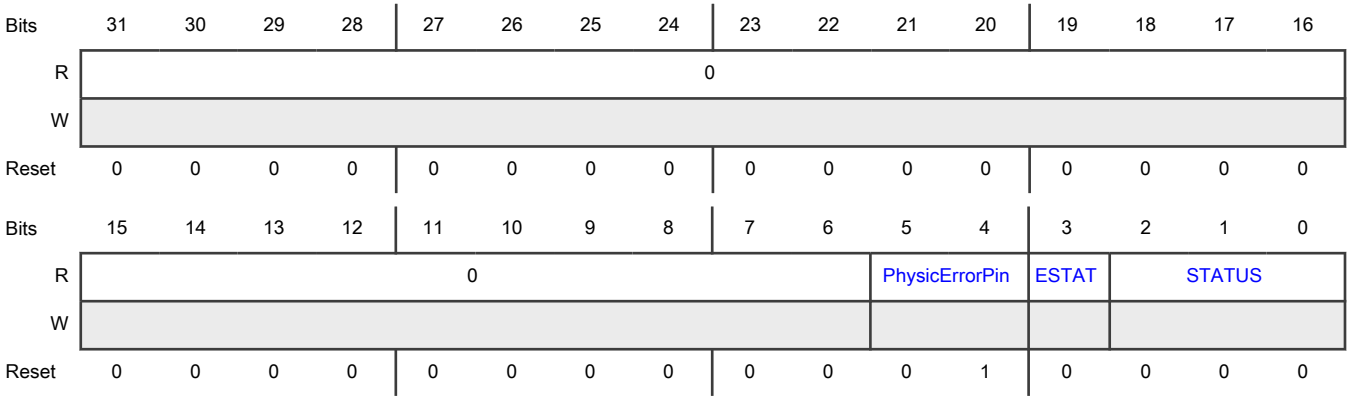
| Register | Offset |
|---|---|
| STAT | C0h |

### Function

This register indicates the following:

- States that FCCU is driving on the EOUT signals
- Whether FCCU is in a faulty condition
- Current state of FCCU

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved |
| 5-4<br>PhysicErrorPin | EOUT Signal States<br>Applies only when the EOUT signals are active (CFG[FCCU_SET_AFTER_RESET]). Indicates the states that FCCU is driving on the EOUT signals.<br>00b - EOUT1 is low; EOUT0 is low.<br>01b - EOUT1 is low; EOUT0 is high.<br>10b - EOUT1 is high; EOUT0 is low.<br>11b - EOUT1 is high; EOUT0 is high. |
| 3<br>ESTAT | FCCU Faulty Condition<br>Indicates whether FCCU is in faulty condition (as indicated by the EOUT signals). For more information, see The FCCU conditions.<br>0b - Not in faulty condition (in non-faulty or configuration condition)<br>1b - In faulty condition |
| 2-0<br>STATUS | FCCU State<br>Indicates the current state of FCCU<br>000b - NORMAL |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 001b - CONFIG<br>010b - ALARM<br>011b - FAULT<br>100b - Reserved<br>101b - Reserved<br>110b - Reserved<br>111b - Reserved |

### 49.7.1.15 Normal-to-Alarm Freeze Status (N2AF_STATUS)

#### Offset

| Register | Offset |
|---|---|
| N2AF_STATUS | C4h |

#### Function

See N2AF_STATUS[NAFS].

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0 | Normal-to-Alarm Freeze Status |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| NAFS | Used only for testing and debugging. Indicates whether FCCU left the NORMAL state and entered the ALARM state since the last time this register was cleared and, if so, which non-critical fault caused FCCU to do so.<br>**NOTE**<br>To clear this register and the other freeze status registers, see Clear the freeze-status indicators.<br>00h: No Normal-to-Alarm-state transition (cleared)<br>01h: NCF0<br>10h: NCF1<br>...<br>7Fh: NCF126<br>80h: NCF127<br>...<br>FFh: Multiple Normal-to-Alarm-state transitions |

### 49.7.1.16 Alarm-to-Fault Freeze Status (A2FF_STATUS)

#### Offset

| Register | Offset |
|---|---|
| A2FF_STATUS | C8h |

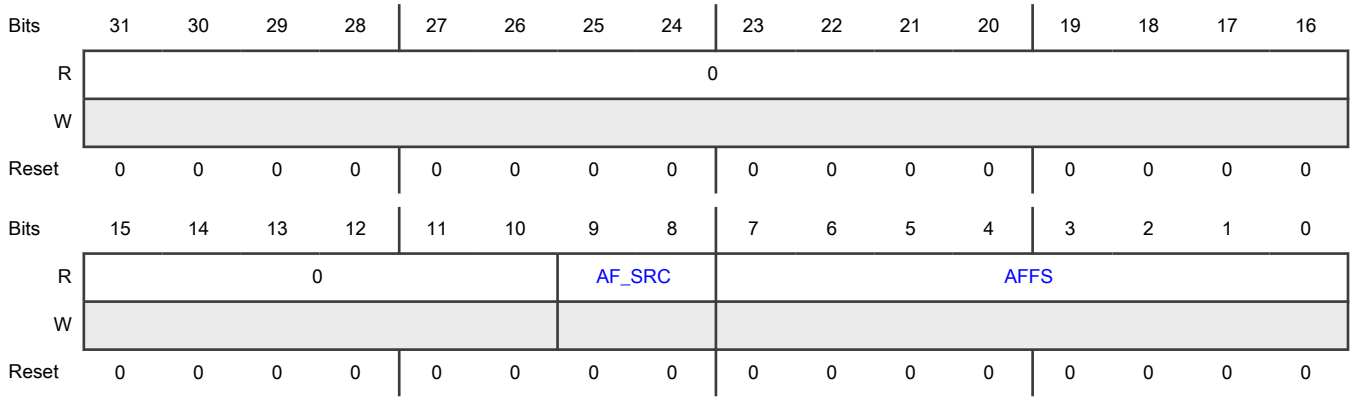
#### Function

Used only for testing and debugging. Indicates whether FCCU left the ALARM state and entered the FAULT state since the last time this register was cleared and, if so, which type of fault caused FCCU to do so.

**NOTE**

To clear this register and the other freeze status registers, see Clear the freeze-status indicators.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9-8<br>AF_SRC | Alarm-to-Fault Source<br>Used only for testing and debugging. Indicates the type of fault that caused FCCU to leave the ALARM state and enter the FAULT state since the last time this register was cleared.<br>**NOTE**<br>To clear this register and the other freeze status registers, see Clear the freeze-status indicators.<br>00b - No Alarm-to-Fault-state fault<br>01b - Reserved<br>10b - Non-critical fault<br>11b - Multiple Alarm-to-Fault-state faults |
| 7-0<br>AFFS | Alarm-to-Fault Freeze Status<br>Used only for testing and debugging. Indicates whether FCCU left the ALARM state and entered the FAULT state since the last time this register was cleared and, if so, which fault caused FCCU to do so.<br>**NOTE**<br>To clear this register and the other freeze status registers, see Clear the freeze-status indicators.<br>00h: No Alarm-to-Fault-state transition (cleared)<br>01h: NCF0 (due to an Alarm-state timeout)<br>10h: NCF1 (due to an Alarm-state timeout)<br>...<br>7Fh: NCF126 (due to an Alarm-state timeout) |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 80h: NCF127 (due to an Alarm-state timeout)<br>...<br>FFh: Multiple Alarm-to-Fault-state transitions |

### 49.7.1.17 Normal-to-Fault Freeze Status (N2FF_STATUS)

#### Offset

| Register | Offset |
|---|---|
| N2FF_STATUS | CCh |

#### Function

Used only for testing and debugging. Indicates whether FCCU left the NORMAL state and entered the FAULT state since the last time this register was cleared and, if so, which type of fault caused FCCU to do so.

**NOTE**

To clear this register and the other freeze status registers, see Clear the freeze-status indicators.

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9-8<br>NF_SRC | Normal-to-Fault Source<br>Used only for testing and debugging. Indicates the type of fault that caused FCCU to leave the NORMAL state and enter the FAULT state since the last time this register was cleared. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>To clear this register and the other freeze status registers, see Clear the freeze-status indicators.<br>00b - No Normal-to-Fault-state fault<br>01b - Reserved<br>10b - Non-critical fault<br>11b - Multiple Normal-to-Fault-state faults |
| 7-0<br>NFFS | Normal-to-Fault Freeze Status<br>Used only for testing and debugging. Indicates whether FCCU left the NORMAL state and entered the FAULT state since the last time this register was cleared and, if so, which fault caused FCCU to do so.<br>**NOTE**<br>To clear this register and the other freeze status registers, see Clear the freeze-status indicators.<br>00h: No Normal-to-Fault-state transition (cleared)<br>01h: NCF0<br>10h: NCF1<br>...<br>7Fh: NCF126<br>80h: NCF127<br>...<br>FFh: Multiple Normal-to-Fault-state transitions |

### 49.7.1.18 Fault-to-Alarm Freeze Status (F2AF_STATUS)

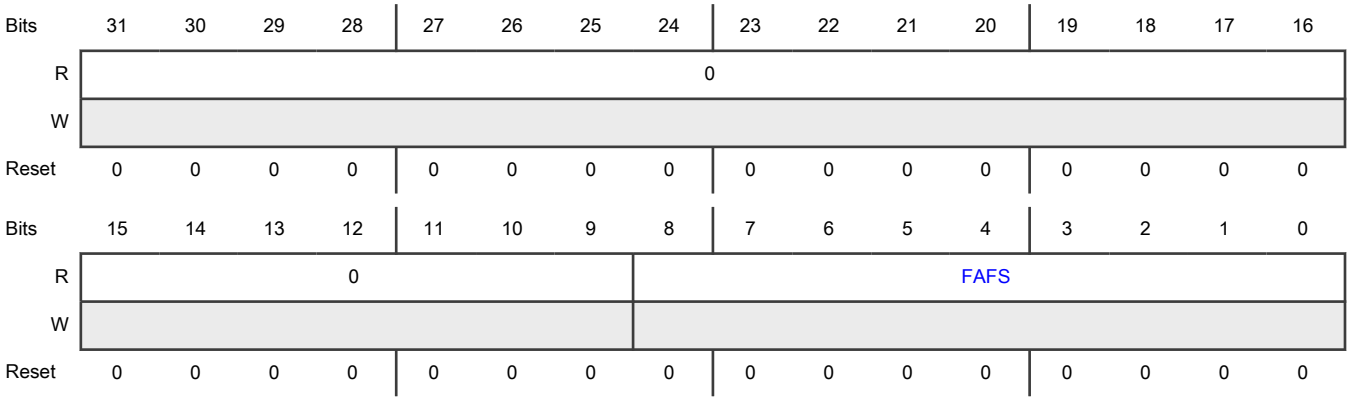
#### Offset

| Register | Offset |
|---|---|
| F2AF_STATUS | D0h |

#### Function

See F2AF_STATUS[FAFS].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8-0<br>FAFS | Fault-to-Alarm Freeze Status<br>Used only for testing and debugging. Indicates whether FCCU left the FAULT state and entered the ALARM state since the last time this register was cleared and, if so, which non-critical fault caused FCCU to do so.<br>**NOTE**<br>To clear this register and the other freeze status registers, see Clear the freeze-status indicators.<br>00h: No Fault-to-Alarm-state transition (cleared)<br>01h: NCF0<br>10h: NCF1<br>...<br>7Fh: NCF126<br>80h: NCF127<br>...<br>FFh: Multiple Fault-to-Alarm-state transitions |

## 49.7.1.19 Non-critical Fault Fake (NCFF)

### Offset

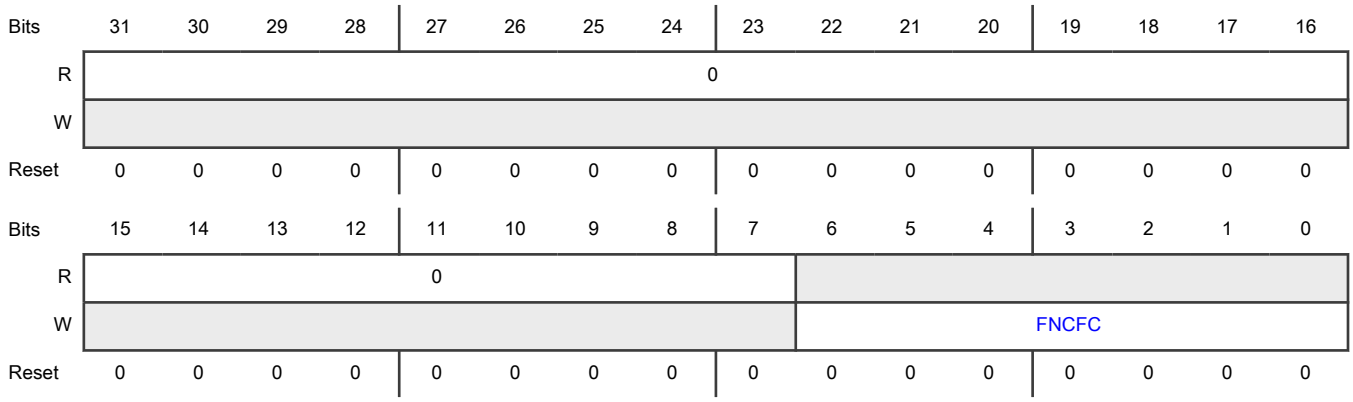
| Register | Offset |
|---|---|
| NCFF | DCh |

### Function

This register contains a unique code to set a non-critical fault in mutually exclusive mode by the external FAULT interface (signal setting). It allows the SW emulation of the non-critical faults, by injecting the fault directly in the FAULT root, to verify the entire path and reaction. The reaction following a fake non-critical fault cannot be masked.

This is a write-only register with a set of codes corresponding to each non-critical fault injection.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-7<br>— | Reserved |
| 6-0<br>FNCFC | FNCFC<br>Fake non-critical fault code<br>**NOTE**<br>Writing to this field injects fake faults; writing 00 and the default value being 0 renders different results.<br>00h: Fake non-critical fault injection at non-critical fault source 0<br>01h: Fake non-critical fault injection at non-critical fault source 1<br>02h: Fake non-critical fault injection at non-critical fault source 2<br>...<br>7Fh: Fake non-critical fault injection at non-critical fault source 127 |

## 49.7.1.20 IRQ Status (IRQ_STAT)

### Offset

| Register | Offset |
|---|---|
| IRQ_STAT | E0h |

### Function

This register provides the FCCU interrupt status related to the following events:

- Configuration-state timeout error
- Alarm interrupt
- NMI interrupt

The configuration-state timeout interrupt is asserted if both IRQ_STAT[CFG_TO_STAT] and IRQ_EN[CFG_TO_IEN] bits are asserted. It is cleared when a 1 is written to the IRQ_STAT[CFG_TO_STAT] bit.

The NMI and ALARM interrupts are asserted and cleared according to the FCCU state. The status bits of the IRQ_STAT trace the status of the related interrupt lines.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | Reserved | Reserved | NMI_STAT | ALRM_ST... | CFG_TO_... |
| W | | | | | | | | | | | | | | | | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-5<br>— | Reserved |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>NMI_STAT | NMI Interrupt Status<br>0b - NMI interrupt is OFF<br>1b - NMI interrupt is ON |
| 1<br>ALRM_STAT | Alarm Interrupt Status<br>0b - Alarm interrupt is OFF<br>1b - Alarm interrupt is ON |
| 0 | Configuration-State Timeout Status |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| CFG_TO_STAT | 0b - No configuration-stat timeout error<br>1b - Configuration-state timeout error |

### 49.7.1.21 IRQ Enable (IRQ_EN)

#### Offset

| Register | Offset |
|---|---|
| IRQ_EN | E4h |

#### Function

This register is used to configure enabling of interrupt related to the "Configuration-state timeout error".

The configuration-state timeout interrupt is asserted if both the IRQ_STAT[CFG_TO_STAT] and IRQ_EN[CFG_TO_IEN] fields are set to 1.

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3<br>— | Reserved |
| 2-1<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 0<br>CFG_TO_IEN | Configuration-State Timeout Interrupt Enable<br>0b - Configuration-state timeout interrupt disabled<br>1b - Configuration-state timeout interrupt enabled |

### 49.7.1.22 Transient Configuration Lock (TRANS_LOCK)

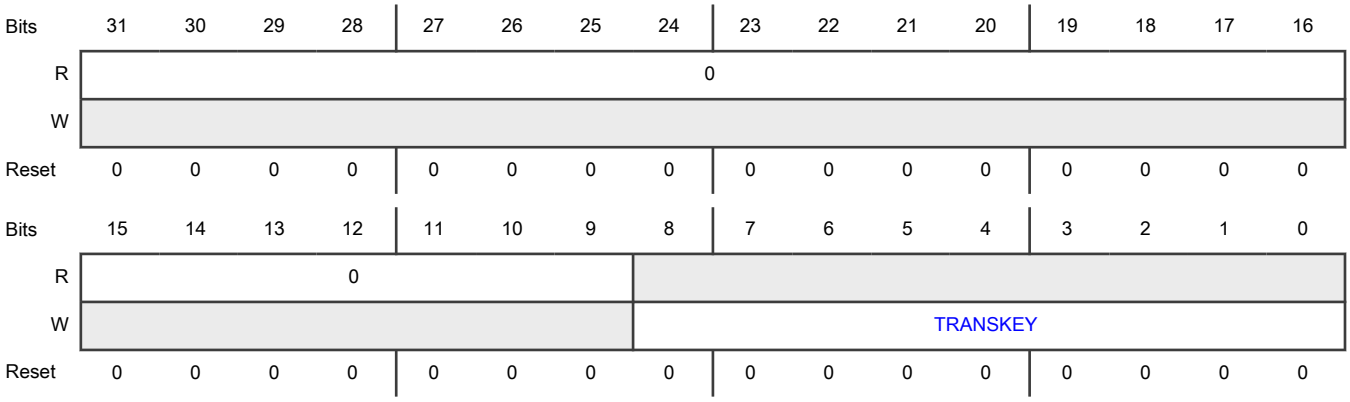
#### Offset

| Register | Offset |
|---|---|
| TRANS_LOCK | F0h |

#### Function

See TRANS_LOCK[TRANSKEY]

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8-0<br>TRANSKEY | Transient Configuration Lock<br>Writable only by code running in Supervisor mode. Temporarily locks and unlocks the configuration. Locking the configuration prevents FCCU from entering the CONFIG state. For information about putting FCCU in configuration, see Prepare FCCU for configuration and Configure FCCU. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>You can write to this field when FCCU is in any state, but the lock will not get into effect until FCCU is in the NORMAL state.<br>BCh: Unlock.<br>Any other value: Lock. |

### 49.7.1.23 Permanent Configuration Lock (PERMNT_LOCK)

#### Offset

| Register | Offset |
|---|---|
| PERMNT_LOCK | F4h |

#### Function

See PERMNT_LOCK[PERMNTKEY]

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8-0<br>PERMNTKEY | Permanent Configuration Lock<br>Writable only by code running in the Supervisor mode. Permanently locks the configuration, which prevents FCCU from entering the CONFIG state until FCCU is reset. For information about putting FCCU in configuration, see Prepare FCCU for configuration and Configure FCCU. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>You can write to this field when FCCU is in any state, but the lock will not get into effect until FCCU is in NORMAL state.<br>FFh: Lock.<br>Any other value: Do nothing. |

### 49.7.1.24 Delta T (DELTA_T)

#### Offset

| Register | Offset |
|---|---|
| DELTA_T | F8h |

#### Function

The DELTA_T register is used for programming the value of delta_T constant (see DELTA_T), in microseconds.

**NOTE**
This register can be written only when the FCCU is in the CONFIG state.

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-16 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 15-14<br>— | Reserved |
| 13-0<br>DELTA_T | Minimum Fault-Output (EOUT) Timer Interval<br>Applies only to Bi-Stable mode (CFG[FOM]). Controls the minimum amount of time ( $T_{min}$) that the fault-output (EOUT) timer runs according to this equation:<br>$T_{min}$ = (250+ DELTA_T) µs * (48000/CLKSAFE freq KHz )<br>**NOTE**<br>The durations shown for the DELTA_T values depend on CLKSAFE signals. For frequency of CLKSAFE signals see the chip-specific FCCU information). Also see chip's data sheet for the trimmed frequency variation (for example, $\delta F_{var}$). |

### 49.7.1.25 Non-critical Alarm-State Interrupt-Request Enable (IRQ_ALARM_EN0)

#### Offset

| Register | Offset |
|---|---|
| IRQ_ALARM_EN0 | FCh |

#### Function

See IRQ_ALARM_ENa[IRQENn].

#### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>IRQENn | Non-critical Alarm-State Interrupt-Request Enable n<br>Writable only when FCCU is in the CONFIG state. Changed by FCCU to its reset value when a Configuration-state timeout occurs. Controls whether the interrupt request is enabled as the Alarm-state reaction for the associated non-critical fault channel (n). When the ALARM state and the Alarm-state interrupt request are enabled for an enabled non-critical fault channel, a fault on that channel causes FCCU to assert the irq_alarm signal when FCCU enters the ALARM state; irq_alarm remains asserted until FCCU is in the NORMAL state. For information on how to configure the non-critical fault channels, see Configure the non-critical fault channels.<br>0b - Disabled<br>1b - Enabled |

## 49.7.1.26 Non-critical Fault-State Non-maskable-Interrupt-Request Enable (NMI_EN0)

### Offset

| Register | Offset |
|---|---|
| NMI_EN0 | 10Ch |

### Function

See NMI_ENa[NMIENn].

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>NMIENn | Non-critical Fault-State Non-maskable-Interrupt-Request Enable n<br>Writable only when FCCU is in the CONFIG state. Changed by FCCU to its reset value when a Configuration-state timeout occurs. Controls whether the non-maskable interrupt request is enabled as a Fault-state reaction for the associated non-critical fault channel (n). When the non-maskable interrupt request is enabled for an enabled non-critical fault channel, a fault on that channel causes FCCU to assert the NMIOUT signal when FCCU enters the FAULT state; NMIOUT remains asserted until FCCU exits FAULT state. For information on how to configure the non-critical fault channels, see Configure the non-critical fault channels.<br>0b - Disabled<br>1b - Enabled |

## 49.7.1.27 Non-critical Fault-State EOUT Signaling Enable (EOUT_SIG_EN0)

### Offset

| Register | Offset |
|---|---|
| EOUT_SIG_EN0 | 11Ch |

### Function

See EOUT_SIG_ENa[EOUTENn].

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | EOUT EN7 | EOUT EN6 | EOUT EN5 | EOUT EN4 | EOUT EN3 | EOUT EN2 | EOUT EN1 | EOUT EN0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Fields**

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>EOUTENn | Non-critical Fault-State EOUT Signaling Enable n<br>Writable only when FCCU is in the CONFIG state. Changed by FCCU to its reset value when a Configuration-state timeout occurs. Applies only when the EOUT signals are active (CFG[FCCU_SET_AFTER_RESET]). When FCCU is configured for Bi-Stable fault-output mode (CFG[FOM]), controls whether EOUT signaling is enabled as a Fault-state reaction for the associated non-critical fault channel (n). (For other fault-output modes, EOUT signaling is always enabled, regardless of the value of this field.) When EOUT signaling is enabled for an enabled non-critical fault channel, a fault on that channel causes FCCU to indicate the faulty condition on the EOUT[1:0] signals when FCCU enters the FAULT state. For information on how to configure the non-critical fault channels, see Configure the non-critical fault channels.<br>For all fault-output modes, also controls whether FCCU asserts the FIF signal when a fault on the associated non-critical fault channel (n) causes FCCU to enter the FAULT state.<br>**NOTE**<br>For all fault-output modes, you must set this field to enabled to ensure that FCCU asserts the FIF signal when FCCU enters FAULT state as the result of a fault on the associated non-critical fault channel (n) so the FOSU module does not mistakenly generate a destructive chip reset.<br>0b - In Bi-Stable fault-output mode, both EOUT signaling and FIF assertion are disabled; in other fault-output modes, EOUT signaling is enabled and FIF assertion is disabled.<br>1b - Both EOUT signaling and FIF assertion are enabled in all fault-output modes. |

### 49.7.1.28 Alarm-State Timer (TMR_ALARM)

**Offset**

| Register | Offset |
|---|---|
| TMR_ALARM | 12Ch |

**Function**

See TMR_ALARM[COUNT].

This table shows how the Alarm-state timer's state and value vary by FCCU state:

**Table 232. TMR_ALARM reset value**

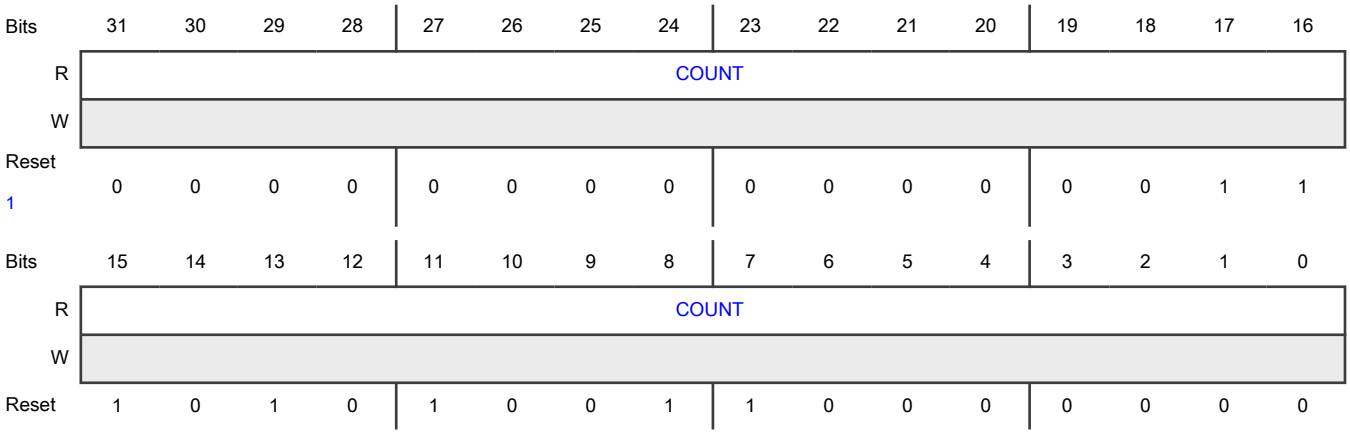
| FCCU state | Timer state | Timer value |
|---|---|---|
| CONFIG | Idle | 0000_0000h |
| NORMAL | Idle | Initial value: NCF_TO[TO] |

*Table continues on the next page...*

Table 232. TMR_ALARM reset value (continued)

| FCCU state | Timer state | Timer value |
|---|---|---|
| ALARM | Running | Value when read |
| FAULT | Idle | End of count |

Diagram




1. The default reset value is provided by NCF_TO[TO]. See Table 232 for the reset value at different FCCU states.

Fields

| Field | Function |
|---|---|
| 31-0<br>COUNT | Alarm-State Timer Count<br>Specifies the value of the Alarm-state timer in CLKSAFE periods. |

### 49.7.1.29 Configuration-State Timer (TMR_CFG)

Offset

| Register | Offset |
|---|---|
| TMR_CFG | 134h |

Function

See TMR_CFG[COUNT].

This table shows how the Configuration-state timer's state and value vary by FCCU state:

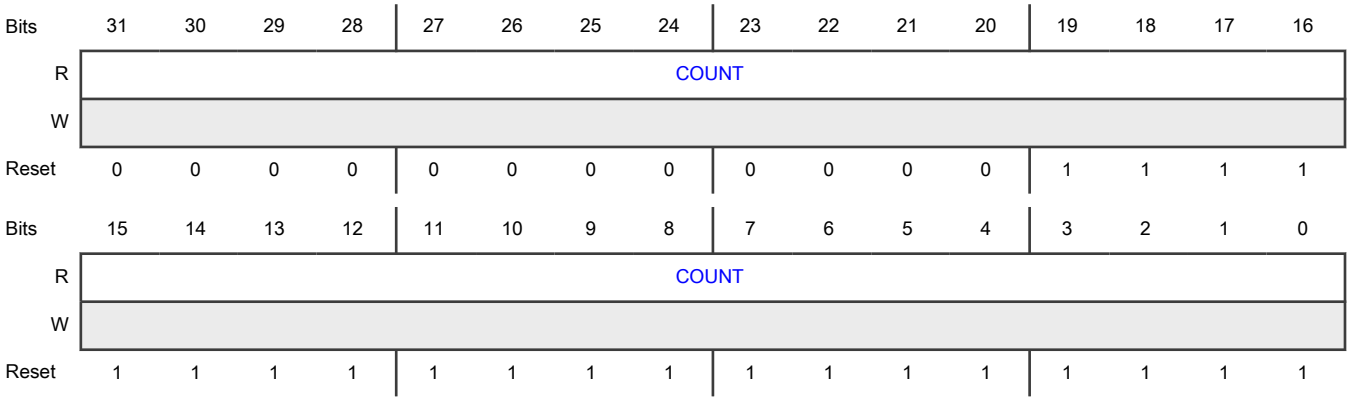
| FCCU state | Timer state | Timer value |
|---|---|---|
| CONFIG | Running | Value when read |

*Table continues on the next page...*

*Table continued from the previous page...*

| FCCU state | Timer state | Timer value |
|---|---|---|
| NORMAL | Idle | 000F_FFFFh |
| ALARM | Idle | 000F_FFFFh |
| FAULT | Idle | 000F_FFFFh |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>COUNT | Configuration-State Timer Count<br>Specifies the value of the Configuration-state timer in CLKSAFE periods. |

## 49.7.1.30 Fault-Output Timer (TMR_ETMR)

### Offset

| Register | Offset |
|---|---|
| TMR_ETMR | 138h |

### Function

See TMR_ETMR[COUNT].

This table shows how the fault-output timer's state and value vary by FCCU state:

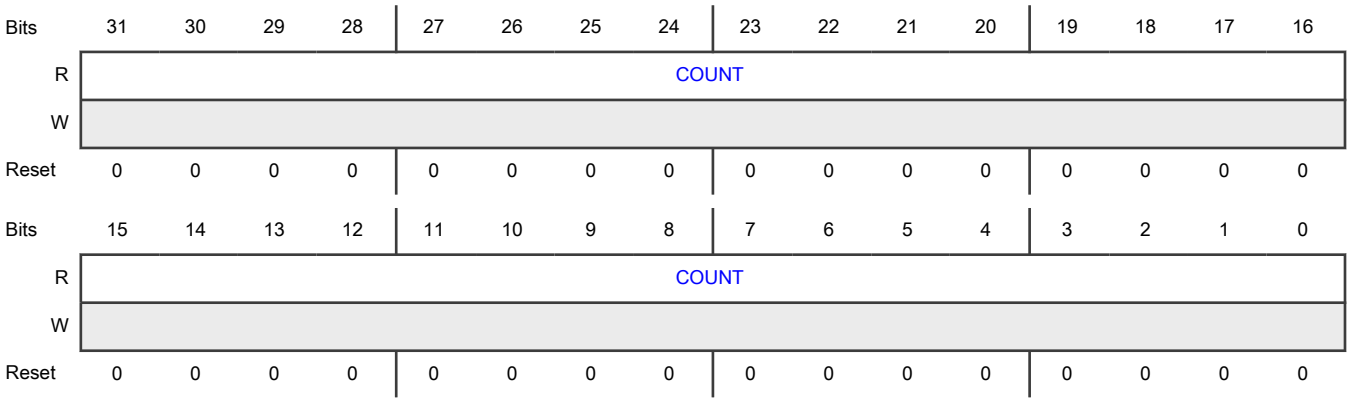
| FCCU state | Timer state (value) |
|---|---|
| CONFIG | Idle (0000_0000h) |

*Table continues on the next page...*

*Table continued from the previous page...*

| FCCU state | Timer state (value) |
|---|---|
| NORMAL | Idle (0000_0000h) |
| ALARM | Idle (0000_0000h) |
| FAULT | Running (value when read. It is an up-counter which rollbacks to zero after reaching maximum value and then it begins counting again.) |

**Diagram**




**Fields**

| Field | Function |
|---|---|
| 31-0<br>COUNT | Fault-Output Timer Count<br>Specifies the value of the fault-output timer in CLKSAFE periods. |

## 49.7.2 Configuration registers

### 49.7.2.1 Definition

*Configuration registers* are registers that:

- Let you configure FCCU's Alarm-state timer interval, fault channels, and fault-output (EOUT) signals.
- You can write to only when the configuration is not locked and FCCU is in CONFIG state (see Put FCCU in the CONFIG state).
- Save the values you write to them while FCCU is in CONFIG state only after you manually put FCCU in NORMAL state. If FCCU automatically leaves CONFIG state and enters NORMAL state because the configuration-timer interval (CFG_TO[TO]) expires (called a Configuration-state timeout), FCCU changes the value of the Configuration (CFG) register to its Configuration-state-timeout value and the value of each of the other configuration registers to its reset value; FCCU also changes the value of the Configuration-State Timeout Interval (CFG_TO) register to its reset value. For information on the Configuration-state timeout value, see CFG register bit value at different events.

### 49.7.2.2 Configuration registers

Following is the list of configuration registers in this module. They are listed in an offset order from lowest to highest.

- Configuration (CFG)
- Non-critical Fault Configuration (NCF_CFGa)
- Non-critical Fault-State Configuration (NCFS_CFG0)
- Non-critical Fault Enable (NCF_E0)
- Non-critical-Fault Alarm-State Timeout Enable (NCF_TOE0)
- Non-critical-Fault Alarm-State Timeout Interval (NCF_TO)
- Delta T (DELTA_T)
- Non-critical Alarm-State Interrupt-Request Enable (IRQ_ALARM_EN0)
- Non-critical Fault-State Non-maskable-Interrupt-Request Enable (NMI_EN0)
- Non-critical Fault-State EOUT Signaling Enable (EOUT_SIG_EN0)

### 49.7.3 CFG register bit value at different events

In this chip, there are no events that affect the bits in the configuration register.

## 49.8 Glossary

| | |
|---|---|
| **CF** | Critical fault |
| **EOUT** | Error out |
| **FOSU** | FCCU output supervision unit |
| **FSM** | Finite state machine |
| **intf** | Interface |
| **IRQ** | Interrupt request |
| **NCF** | Non-critical fault |
| **NMI** | Non-maskable interrupt |

# Chapter 50
# Self-test General-Purpose Registers (SELFTEST_GPR)

## 50.1 Introduction

SELFTEST_GPR configures the STCU2 LBIST shift cycles. Programming self-test GPRs is a part of the self-test programming sequence. See the "Self-test programming sequence" section in the STCU2 chapter.

NXP supports only STCU2 Built-In Self-Test (BIST) sequences that have been validated by NXP for in-field testing usage.

The detailed programming sequences for supported configurations are available in a separate application note (contact your NXP sales representative).

The values for SELFTEST_GPR registers must be aligned with the NXP supported configurations only.

**NOTE**

SELFTEST_GPR is not present in MCXE315/MCXE316 and MCXE317.

**NOTE**

While accessing SELFTEST_GPR space, software must ensure that the STCU2 block has its clock enabled, i.e., MC_ME.PRTN1_COFB3_CLKEN[REQ104] configured as 1'b1. Not ensuring this might result in unpredictable device behaviour.

## 50.2 Peripheral shift clock switching (PCS)

The PCS feature is used to avoid current surges at the start and/or end of LBIST. Since LBIST can cause sudden current surges which might lead to chip malfunction.

With PCS enabled, there are additional (SELFTEST_GPR.CONFIG_GPR[PCS_STEP_SIZE] + 1)*16 patterns beyond the configured count. If enabled at the start of LBIST (using SELFTEST_GPR.CONFIG_GPR[PCS_ENABLE_START]), it would cause the shift clock to scale up in this duration from divide by 16 to the original frequency (by changing divider in each step).

Similarly, if configured at the end of LBIST (using SELFTEST_GPR.CONFIG_GPR[PCS_ENABLE_END]), the shift clock frequency will be scaled down from original frequency to divide by 16.

## 50.3 SELFTEST_GPR register descriptions

### 50.3.1 SELFTEST_GPR memory map

SELFTEST.SELFTEST base address: 403B_0000h

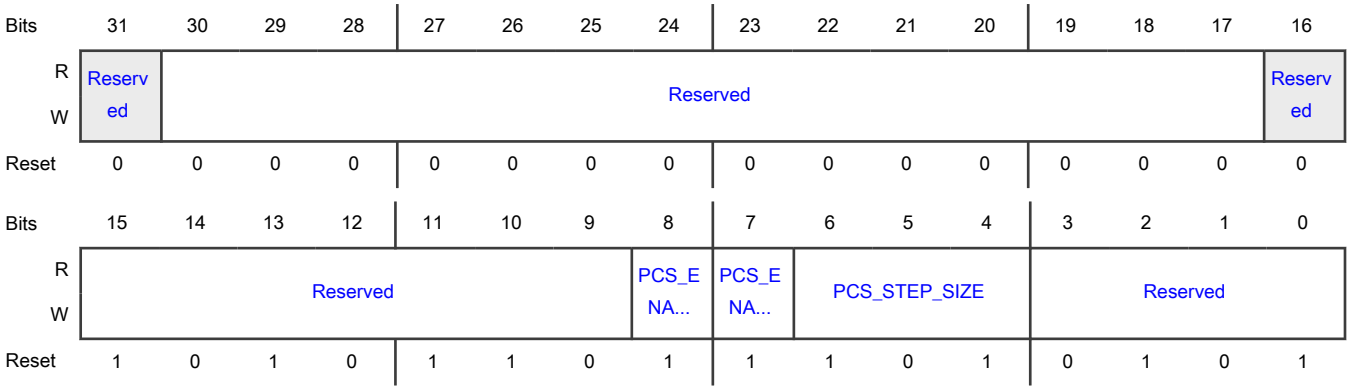
| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Configuration register (CONFIG_REG) | 32 | RW | 0000_ADD5h |
| 14h | LBIST Program (LBIST_PROG_REG) | 32 | RW | 0003_0050h |

## 50.3.2 Configuration register (CONFIG_REG)

### Offset

| Register | Offset |
|---|---|
| CONFIG_REG | 0h |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-17<br>— | Reserved |
| 16<br>— | Reserved |
| 15-9<br>— | Reserved |
| 8<br>PCS_ENABLE_END | PCS Enable End<br>Enables Progressive Shift Clock Switching for LBIST at end. The feature gets enabled when this bit is set to 1. It is recommended to run LBIST patterns by enabling this feature. |
| 7<br>PCS_ENABLE_START | PCS Enable Start<br>Enables Progressive Shift Clock Switching for LBIST at start. The feature gets enabled when this bit is set to 1. It is recommended to run LBIST patterns by enabling this feature. |
| 6-4 | PCS Step Size<br>pcs_step_size[2:0] controls the step size for progressive increase of shift frequency: |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| PCS_STEP_SIZE | 000 - step size of 1 pattern<br>001 - step size of 2 patterns<br>...<br>111 - step size of 8 patterns |
| 3-0<br>— | Reserved |

### 50.3.3 LBIST Program (LBIST_PROG_REG)

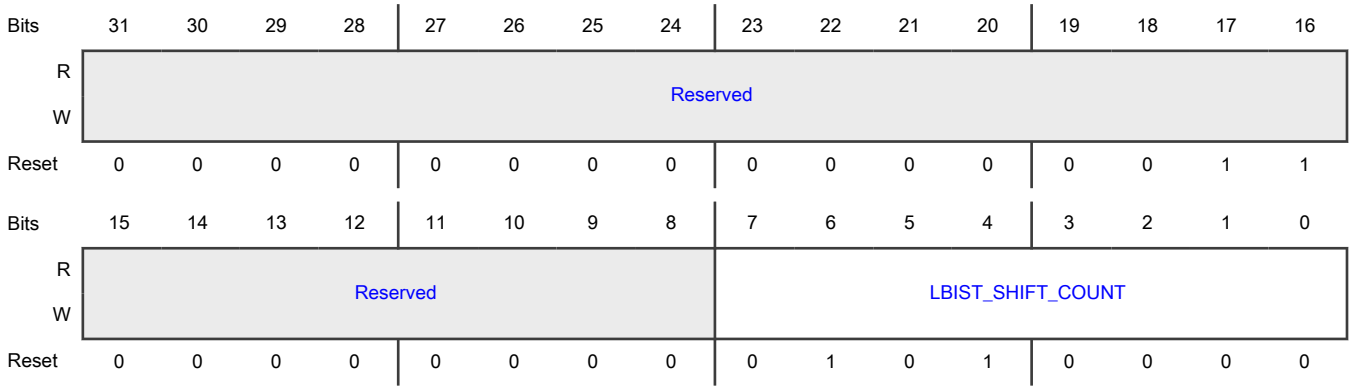
#### Offset

| Register | Offset |
|---|---|
| LBIST_PROG_REG | 14h |

#### Function

Configures the LBIST shift count.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>LBIST_SHIFT_COUNT | LBIST Shift Count<br>Specifies the number of shift cycles in a scan chain in one pattern count for the LBIST partition. |

## 50.4 Glossary

| | |
|---|---|
| **BIST clock** | The BIST clock refers to the BIST operational clock which is the clock source used while LBIST is operational by the LBIST controller, which is the MC_CGM clock node LBIST_CLK for the chip. See the system block diagram for details. |
| **LBIST** | Logic built-in self-test |
| **Pattern count** | The pattern count refers to an empirically determined value of shift patterns run in LBIST on a scan chain to result in predetermined fault coverage, based on device safety integrity level. Since the LBIST scheme is Pseudo Random Testing, a sufficiently good value of pattern count is used which provides the required fault coverage. |
| **Scan chain** | A sequential chain of flops in scan mode over which one scan pattern (generated by PRPG, that is, Pseudo Random Pattern Generator) in LBIST will be shifted and the output MISR (Multi Input Signature Register) will be matched with earlier known value to determine pass or fail criterion of that particular chain. |

# Chapter 51
# Self-Test Control Unit (STCU2)

## 51.1 Chip-specific STCU2 information

### 51.1.1 Supported BIST sequences

This chip supports only STCU2 BIST sequences that NXP has validated for in-field testing usage. The detailed programming sequences for supported configurations are available in a separate application note (contact your NXP sales representative for this information).

### 51.1.2 Self-test overview

STCU2 provides the overall control of both LBIST and MBIST, in serial or parallel, depending on the power, timing, or coverage constraints.

**NOTE**

LBIST is not available for MCXE317 and MCXE315/MCXE316

The application software initiates the self-test sequence or procedure (please use whichever term is correct), and has provisions to bypass the self-test configuration to directly move to normal Run mode.

There is a programmable BIST-sequence-execution watchdog timer that specifies the maximum time allowed for the execution of a BIST sequence. In case the selected LBISTs or MBISTs are not complete during the assigned time, the current LBISTs or MBISTs execution is interrupted and a failure is flagged into ERR_STAT[WDTOSW] and MBESWx or LBESW.

The STCU2 execution includes three phases:

- Software configuration. Load or program the test configuration parameters to be executed, depending on SIL target and performance requirements. The self-test configuration parameters allow the self-test sequence to run with different times depending on pattern numbers and types, achieving different levels of coverage.
- Start of BIST execution. This involves software register configuration in STCU2. The BIST completion is followed by a functional reset.
- End of BIST execution. After BIST execution and functional reset are complete, the BIST results are made available and you decide how to continue (trigger a new reset or start a safety application). The BIST execution completion is signalized in an internal STCU2 register (LBESWn) bit which is not affected by the functional reset. The BIST test results are also not affected by the functional reset.

The following figure shows a pictorial representation of two LBIST partitions having two and one MBIST groups, respectively. STCU2 interacts with LBIST_CTRL for the respective partitions to configure and control the LBIST sequence and compare results. It also interacts with MTR controller (MCT) for MBIST. Each LBIST partition has its corresponding controller whereas a single MCT can cater to multiple MBIST groups.

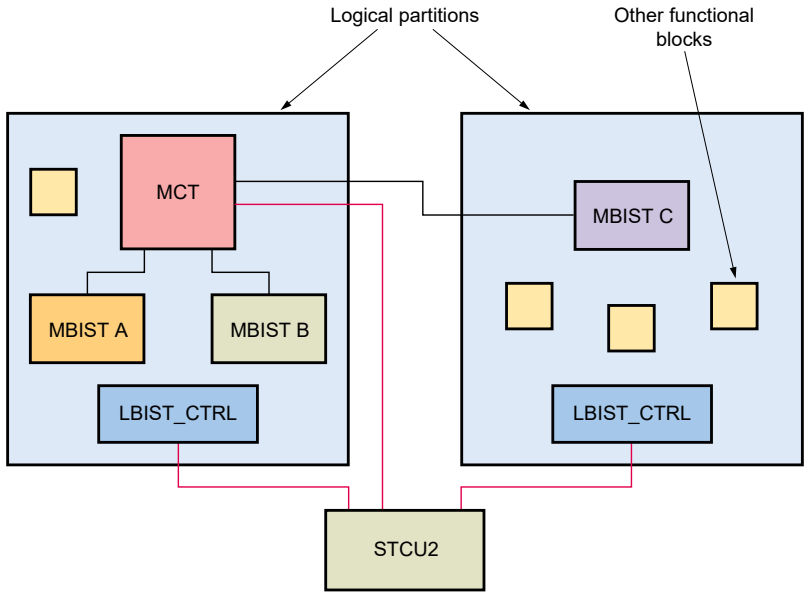



Figure 187. Self-test overview

**NOTE**

LBIST_CTRL is not available for MCXE317 and MCXE315/MCXE316.

Each LBIST partition also includes a block called Self-test GPR involving a set of registers that provide miscellaneous configuration and parameter details for an LBIST operation.

When you trigger a self-test sequence, the chip becomes unavailable for software application during the self-test run.

### 51.1.3 STCU2 LBIST/MBIST mapping

This topic covers STCU2 LBIST/MBIST mapping.

#### LBIST mapping

The below table lists the LBIST mapping of the chip. The NLBIST value is 1.

Table 233. LBIST mapping

| BIST ID (NLBIST) | BIST instance name | Modules |
|---|---|---|
| 0 | LBIST | DMA:<br>• DMA controller |
| | | EIM |
| | | ERM |
| | | XRDC<br>• MDAC0<br>• MDAC1<br>• MDAC2 |

*Table continues on the next page...*

Table 233. LBIST mapping (continued)

| BIST ID (NLBIST) | BIST instance name | Modules |
|---|---|---|
| | | • MDAC3<br>• MDAC4<br>• MDAC5<br>• MRC0<br>• MRC1<br>• MRC2<br>• PDAC0<br>• PDAC1<br>• PDAC2 |
| | | CRC |
| | | FCCU |
| | | FOSU |
| | | PFLASH_CTL |
| | | PRAMC_0 |
| | | PRAMC_1<br>**NOTE**<br>PRAMC_1 is not present in MCXE317, and MCXE315/MCXE316. |
| | | SWT_1<br>**NOTE**<br>SWT_1 is not present in MCXE317 and MCXE315/MCXE316. |
| | | CMU:<br>• CMU1<br>• CMU2<br>• CMU4<br>• CMU5 |
| | | iAHB Gaskets:<br>• DMA_GSKT<br>• HSE_GSKT<br>• AIPS1_GSKT<br>• AIPS2_GSKT<br>• QSPI_GSKT |

*Table continues on the next page...*

Table 233. LBIST mapping (continued)

| BIST ID (NLBIST) | BIST instance name | Modules |
|---|---|---|
| | | **NOTE**<br>AIPS2_GSKT and QSPI_GSKT are not present in MCXE317 and MCXE315/MCXE316.<br>• TCM_GSKT<br>• EMAC_GSKT<br>• EDC master checker gaskets (See block diagram in Introduction chapter for details)<br>• EDC slave checker gasket (See block diagram in Introduction chapter for details) |
| | | Crossbars<br>• AXBS<br>• DMA_AXBS<br>• HSE_AXBS<br>• PERIPHERAL_AXBS<br>• TCM_AXBS |

### MBIST mapping

The below tables lists the MBIST mapping of different variants of this chip. The NMCUT value is 12.

Table 234. MCXE31B MBIST mapping

| BIST index (NMCUT) | BIST name | Memory index | Memory instance name |
|---|---|---|---|
| 0 | SYS0_RAMS | 0 | PRAM0 block 1 |
| 0 | SYS0_RAMS | 1 | PRAM0 block 2 |
| 1 | SYS1_RAMS | 0 | PRAM1 block 1 |
| 1 | SYS1_RAMS | 1 | PRAM1 block 2 |
| 2 | DMA_TCD_RAM | 0 | DMA TCD RAM |
| 3 | CM7_0_TOP | 0 | CM7_0 instruction cache data block 1 |
| 3 | CM7_0_TOP | 1 | CM7_0 instruction cache data block 2 |
| 3 | CM7_0_TOP | 2 | CM7_0 instruction cache tag block 1 |
| 3 | CM7_0_TOP | 3 | CM7_0 instruction cache tag block 2 |
| 3 | CM7_0_TOP | 4 | CM7_0 data cache data block 1 |
| 3 | CM7_0_TOP | 4 | CM7_0 data cache data block 2 |
| 3 | CM7_0_TOP | 5 | CM7_0 data cache data block 3 |
| 3 | CM7_0_TOP | 5 | CM7_0 data cache data block 4 |
| 3 | CM7_0_TOP | 6 | CM7_0 data cache data block 5 |

*Table continues on the next page...*

Table 234. MCXE31B MBIST mapping (continued)

| BIST index (NMCUT) | BIST name | Memory index | Memory instance name |
|---|---|---|---|
| 3 | CM7_0_TOP | 6 | CM7_0 data cache data block 6 |
| 3 | CM7_0_TOP | 7 | CM7_0 data cache data block 7 |
| 3 | CM7_0_TOP | 7 | CM7_0 data cache data block 8 |
| 3 | CM7_0_TOP | 8 | CM7_0 data cache tag block 1 |
| 3 | CM7_0_TOP | 9 | CM7_0 data cache tag block 2 |
| 3 | CM7_0_TOP | 10 | CM7_0 data cache tag block 3 |
| 3 | CM7_0_TOP | 11 | CM7_0 data cache tag block 4 |
| 3 | CM7_0_TOP | 12 | CM7_0 instruction TCM (Tightly Coupled Memory) |
| 3 | CM7_0_TOP | 13 | CM7_0 data TCM block 1 |
| 3 | CM7_0_TOP | 13 | CM7_0 data TCM block 2 |
| 4 | CM7_1_TOP | 0 | CM7_1 instruction cache data block 1 |
| 4 | CM7_1_TOP | 1 | CM7_1 instruction cache data block 2 |
| 4 | CM7_1_TOP | 2 | CM7_1 instruction cache tag block 1 |
| 4 | CM7_1_TOP | 3 | CM7_1 instruction cache tag block 2 |
| 4 | CM7_1_TOP | 4 | CM7_1 data cache data block 1 |
| 4 | CM7_1_TOP | 4 | CM7_1 data cache data block 2 |
| 4 | CM7_1_TOP | 5 | CM7_1 data cache data block 3 |
| 4 | CM7_1_TOP | 5 | CM7_1 data cache data block 4 |
| 4 | CM7_1_TOP | 6 | CM7_1 data cache data block 5 |
| 4 | CM7_1_TOP | 6 | CM7_1 data cache data block 6 |
| 4 | CM7_1_TOP | 7 | CM7_1 data cache data block 7 |
| 4 | CM7_1_TOP | 7 | CM7_1 data cache data block 8 |
| 4 | CM7_1_TOP | 8 | CM7_0 data cache tag block 1 |
| 4 | CM7_1_TOP | 9 | CM7_0 data cache tag block 2 |
| 4 | CM7_1_TOP | 10 | CM7_0 data cache tag block 3 |
| 4 | CM7_1_TOP | 11 | CM7_0 data cache tag block 4 |
| 4 | CM7_1_TOP | 12 | CM7_0 instruction TCM |
| 4 | CM7_1_TOP | 13 | CM7_0 data TCM block 1 |
| 4 | CM7_1_TOP | 13 | CM7_0 data TCM block 2 |
| 5 | FLEX_CAN_RAMS | 0 | FLEXCAN0 MB memory |
| 5 | FLEX_CAN_RAMS | 1 | FLEXCAN1 MB memory |
| 5 | FLEX_CAN_RAMS | 2 | FLEXCAN2 MB memory |

*Table continues on the next page...*

Table 234. MCXE31B MBIST mapping (continued)

| BIST index (NMCUT) | BIST name | Memory index | Memory instance name |
|---|---|---|---|
| 5 | FLEX_CAN_RAMS | 3 | FLEXCAN3 MB memory |
| 5 | FLEX_CAN_RAMS | 4 | FLEXCAN4 MB memory |
| 5 | FLEX_CAN_RAMS | 5 | FLEXCAN5 MB memory |
| 6 | QSPI_PERI_RAMS | 0 | QSPI RAM |
| 6 | QSPI_PERI_RAMS | 1 | QSPI TX memory |
| 7 | EMAC_TSN_RAM | 0 | EMAC Timestamp Memory |
| 8 | EMAC_RAMS | 0 | EMAC RXParser |
| 8 | EMAC_RAMS | 1 | EMAC TX block 1 |
| 8 | EMAC_RAMS | 2 | EMAC TX block 1 |
| 8 | EMAC_RAMS | 3 | EMAC TX block 2 |
| 8 | EMAC_RAMS | 4 | EMAC RX block 2 |
| 9 | b03_ETF_RAMS | 0 | HTM ETF |
| 9 | b03_ETF_RAMS | 1 | Shared ETF |
| 9 | b03_ETF_RAMS | 2 | CM7 cluster Instruction ETF |
| 9 | b03_ETF_RAMS | 3 | CM7 cluster Data ETF |
| 10 | HSE_RAMS | 0 | HSE secure RAM |
| 10 | HSE_RAMS | 1 | HSE PKC memory block 1 |
| 10 | HSE_RAMS | 2 | HSE PKC memory block 2 |
| 10 | HSE_RAMS | 3 | HSE MTB memory |
| 11 | HSE_ROMS | 0 | BOOTROM block 1 |
| 11 | HSE_ROMS | 0 | BOOTROM block 2 |
| 11 | HSE_ROMS | 0 | BOOTROM block 3 |
| 11 | HSE_ROMS | 0 | BOOTROM block 4 |
| 11 | HSE_ROMS | 0 | BOOTROM block 5 |

Table 235. MCXE317 MBIST mapping

| BIST index (NMCUT) | BIST name | Memory index | Memory instance name |
|---|---|---|---|
| 0 | HSE_ROMS | 0 | BOOTROM block 1 |
| 0 | HSE_ROMS | 0 | BOOTROM block 2 |
| 0 | HSE_ROMS | 0 | BOOTROM block 3 |
| 0 | HSE_ROMS | 0 | BOOTROM block 4 |
| 0 | HSE_ROMS | 0 | BOOTROM block 5 |

*Table continues on the next page...*

Table 235. MCXE317 MBIST mapping (continued)

| BIST index (NMCUT) | BIST name | Memory index | Memory instance name |
|---|---|---|---|
| 1 | HSE_RAMS | 0 | HSE secure RAM |
| 1 | HSE_RAMS | 1 | HSE PKC memory block 1 |
| 1 | HSE_RAMS | 2 | HSE PKC memory block 2 |
| 1 | HSE_RAMS | 3 | HSE MTB memory |
| 2 | SYS0_DMA_RAMS | 0 | PRAM0 block 1 |
| 2 | SYS0_DMA_RAMS | 1 | PRAM0 block 2 |
| 2 | SYS0_DMA_RAMS | 2 | DMA TCD RAM |
| 3 | CM7_0_TOP | 0 | CM7_0 instruction cache data block 1 |
| 3 | CM7_0_TOP | 1 | CM7_0 instruction cache data block 2 |
| 3 | CM7_0_TOP | 2 | CM7_0 instruction cache tag block 1 |
| 3 | CM7_0_TOP | 3 | CM7_0 instruction cache tag block 2 |
| 3 | CM7_0_TOP | 4 | CM7_0 data cache data block 1 |
| 3 | CM7_0_TOP | 4 | CM7_0 data cache data block 2 |
| 3 | CM7_0_TOP | 5 | CM7_0 data cache data block 3 |
| 3 | CM7_0_TOP | 5 | CM7_0 data cache data block 4 |
| 3 | CM7_0_TOP | 6 | CM7_0 data cache data block 5 |
| 3 | CM7_0_TOP | 6 | CM7_0 data cache data block 6 |
| 3 | CM7_0_TOP | 7 | CM7_0 data cache data block 7 |
| 3 | CM7_0_TOP | 7 | CM7_0 data cache data block 8 |
| 3 | CM7_0_TOP | 8 | CM7_0 data cache tag block 1 |
| 3 | CM7_0_TOP | 9 | CM7_0 data cache tag block 2 |
| 3 | CM7_0_TOP | 10 | CM7_0 data cache tag block 3 |
| 3 | CM7_0_TOP | 11 | CM7_0 data cache tag block 4 |
| 3 | CM7_0_TOP | 12 | CM7_0 instruction TCM (Tightly Coupled Memory) |
| 3 | CM7_0_TOP | 13 | CM7_0 data TCM block 1 |
| 3 | CM7_0_TOP | 13 | CM7_0 data TCM block 2 |
| 4 | FLEX_CAN_RAMS | 0 | FLEXCAN0 MB memory |
| 4 | FLEX_CAN_RAMS | 1 | FLEXCAN1 MB memory |
| 4 | FLEX_CAN_RAMS | 2 | FLEXCAN2 MB memory |
| 4 | FLEX_CAN_RAMS | 3 | FLEXCAN3 MB memory |
| 4 | FLEX_CAN_RAMS | 4 | FLEXCAN4 MB memory |
| 4 | FLEX_CAN_RAMS | 5 | FLEXCAN5 MB memory |

### 51.1.4 Clock connectivity

STCU2 works on the system clock domain. The scan operations while LBIST are done on TCK clock.

Table 236. Clock connectivity

| STCU2 clock | Clock connected to the chip | Remarks |
|---|---|---|
| CORE_CLK | AIPS_SLOW_CLK | System clock divider output clock node for STCU2 operations and register interface accesses |
| REG_INTF_CLK | | |
| SHIFT_CLK | JTAG_TCK | JTAG_TCK used for shift operations when self-test is in progress |

### 51.1.5 Memory map for self-test GPRs

See the memory map file attached to this document and 'SELFTEST_GPR' chapter for details.

### 51.1.6 Handling unrecoverable faults

Only recoverable and non-critical faults from STCU2 are mapped to FCCU. The unrecoverable and critical faults are mapped to MC_RGM. Therefore, ignore any references to mapping of unrecoverable and critical faults with FCCU in this chapter.

### 51.1.7 Indicating fault state during the main reset domain Self-Test

FCCU EOUT indicates the fault state during the Self-Test as soon as the UTEST_MISC[FCCU_EOUT_ DEDICATED] is enabled and DCMRWD2[EOUT_STAT_DUR_STEST] is enabled at the time of Self-Test setup steps.

The fault state remains asserted until the bit DCMRWD2[EOUT_STAT_DUR_STEST] is cleared post Self-Test reset sequence.

In case it is not required to indicate the fault state during the Self-Test, then the bit DCMRWD2[EOUT_STAT_DUR_STEST] should not be programmed.

### 51.1.8 Error events management in main reset domain Self-Test

Error events should be mapped to unrecoverable faults in STCU2_ERR_FM register for main reset domain Self-Test.

### 51.1.9 AUTOLOCK_VALUE for register write access via STCU2_SKC

Register write-access watchdog timer explains that access to STCU2 registers via the STCU2_SKC security-key mechanism times out after a number of STCU2 clock cycles equal to AUTOLOCK_VALUE. On this chip, AUTOLOCK_VALUE is 32'hFFFF_FFFF.

### 51.1.10 PLL loss of lock during Self-Test

Detection of PLL loss of lock during LBIST or MBIST causes a destructive reset in the system.

CMU0 programming is required before the Self-Test programming begins.

All the registers present in LBIST domains do not retain their content after Self-Test reset.

### 51.1.11 STCU2 BIST start (BSTART) register description

Attempted accesses with the undefined reserved values in the BSTART register can result in undefined behavior. The following table defines the BSTART field on this chip.

Table 237. BSTART description

| Field value | Description |
|---|---|
| 000b | NOP (reset value) |
| 001b | RUN_ONLY: Run BISTs.<br>MTR controller (MCT) runs the selected BISTs without programming them first. |
| 010b | Reserved |
| 011b | Reserved |
| 100b | PROG_ONLY: Program BISTs only.<br>MCT only programs BISTs and does not start them. This enables different BISTs to be programmed with different algorithms before all the selected BISTs are started using the RUN_ONLY command. |
| 101b | PROG_RUN: Program BISTs and start them. |
| 110b | Reserved |
| 111b | Reserved |

### 51.1.12 STCU2 algorithm select (ALGOSEL) register description

This register selects any one of the predefined algorithms. The value used in this register is mapped to a predefined algorithm register in BIST. After MCT has programmed information on this register and starts some BIST operation, this value is transferred to the predefined algorithm register in every BIST that is selected.

If multiple predefined algorithms are selected, the order of execution is from LSB to MSB.

Any field that does not have a predefined algorithm associated with it is a reserved bit. No data is written to reserved bits and when you read these bits, they always return 0. These predefined algorithms include:

- Backgrounds
- Address modes and other register programming that may be necessary
- The number of BIST runs needed to perform the exact coverage requested.

MCT decodes this information through a series of look-up tables. The following table describes the high-level BIST algorithms that can be invoked through the MCT. The columns in that table have the following meaning:

- Index: The position in this MCT register enables this algorithm.
- Name: The symbolic name for this algorithm.
- Description: The intent and use model for this algorithm.
- BIST runs: The sequence of BIST invocations comprising this algorithm.
- Sequences: The ordered list of march elements to run. Each march element consists of a direction indicator (for increasing addresses and for decreasing addresses) and a set of march phases. A march phase consists of reading (R) or writing (W), the non-inverted (0) or inverted (1) background pattern.
- Used by default: Whether this algorithm is selected in the post-reset state of this register.

Table 238. ALGOSEL description

| Index | Name | Description | BIST runs | Sequences | Used by default |
|---|---|---|---|---|---|
| 3 | March C+ single | March C+ with column fast and solid background | MarchC+ column fast 1x (#4) | W0, R0W1, R1W0R0W1, R1W0, R0W1R1W0, R0 | Yes |
| 13 | BasicChk | Basic selfcontained checkerboard, using column fast | BasicChk (#18) | R0W1R1 | - |

### 51.1.13 Self-Test programming sequence

This section describes a high level programming sequence for executing Self-Test. The detailed programming sequences for supported configurations are available in a separate application note (contact your NXP sales representative).

- Program clock sources and MC_CGMs as per the clock configuration.

  **NOTE**

  While running MBIST on EMAC timestamp memory, MC_CGM.MUX_9_DC_0[DIV] should be appropriately configured to ensure EMAC_CLK_TS should be atleast 1.5 times the AIPS_SLOW_CLK frequency.

- Program SELFTEST_GPR[CONFIG_REG] and SELFTEST_GPR[LBIST_PROG_REG] as per the LBIST configurations.

  **NOTE**

  This is not applicable for MCXE317 and MCXE315/MCXE316 wherein LBIST is not supported.

- Program STCU2 as per NXP recommended.
- Program dcf_client_utest_misc[FCCU_EOUT_DEDICATED] and DCMRWD2[EOUT_STAT_DUR_STEST] to configure EOUT pins as dedicated and to indicate FCCU error states on EOUT pins during Self-Test respectively.
- Configure ERCTRL[ERASSERT] register bit in MC_RGM to achieve desired behaviour of Reset Pin as documented in Reset Section and Pad States Section.
- Disable functional reset sources to avoid false Self-Test abort scenarios. For example, software watchdog timers cannot be serviced while Self-Test because the software is not accessible and can cause invalid timeout reset. See Reset events and configurations while Self-Test.
- Program the PCS_ENABLE[8:7] fields of the CONFIG_REG register (SELFTEST_GPR) to 0 to disable PCS.
- Trigger Self-Test execution by programming RUNSW register in STCU2.
- At the end of the Self-Test run, a functional reset is generated. Out of reset, the software can check the status registers inside STCU2 and take actions accordingly.

### 51.1.14 Modules operation during/post Self-Test

The clock, reset, power generation modules, such as RGM, PMC, CGM, FIRC, PLL, and SIRC will be available during and post Self-Test. However, because the chip undergoes the reset sequence, PLL and CGM will be reset.

CMU0 and CMU3 will be available during Self-Test if enabled prior to Self-Test, but will be Reset post Self-Test.

STCU2 will also be available during Self-Test and the results can be read post Self-Test.

### 51.1.15 Reset events and configurations while Self-Test

As Self-Test is a software initiated Self-Test, it is expected by software to ensure all the peripherals are disabled as documented in Software considerations before Self-Test.

The functional reset sources should be appropriately configured to avoid any functional reset while Self-Test is running (disable SWT_RST and HSE_SWT_RST. JTAGC should not be in EXTEST, HIGHZ or CLAMP instructions). This ensures no selftest abort while running selftest due to functional reset source. The below table specifies the state of functional reset sources for Self-Test.

**Table 239. Reset events and configurations while self-test**

| Reset source | Reset description | Configurations/remarks for selftest |
|---|---|---|
| FCCU_RST | FCCU reset reaction | The FCCU is a part of logic which is under Self-Test and thus it would be disabled during the Self-Test |
| ST_DONE | Self-Test done reset | The selftest done indication resets the device post selftest. While Self-Test is running, the Self-Test done indication will be in disabled state |
| SWT_RST | SWT reset request | SWT should be disabled in Self-Test configuration sequence. |
| JTAG_RST | JTAG Reset | JTAG communication should be inactive while Self-Test and JTAGC state machine should not be in EXTEST, HIGHZ or CLAMP instructions |
| HSE_SWT_RST | HSE SWT timeout | HSE_SWT should be disabled in Self-Test configuration sequence. |
| SW_FUNC | software 'functional' reset | The software is inactive during Self-Test and hence software functional reset will not arrive while the chip is under Self-Test. |
| DEBUG_FUNC | debug 'functional' reset | MDM DAP should not be configured for debug functional reset generation while Self-Test is running. |

The destructive reset sources are available while in Self-Test. In case of any destructive reset event, the device undergoes a destructive reset sequence, thereby resetting the STCU2 and Self-Test logic itself.

After completion of Self-Test (LBIST or MBIST), a functional reset sequence is executed by MC_RGM.

### 51.1.16 Software considerations before Self-Test

1. As all the communication peripherals are not in LBIST partitions; hence, it is required by SW to stop all communications and disable all the PCTLs of the communication peripherals before initiating Self-Test as the device will be unusable in Self-Test. This is needed so that there is graceful safe stating of inputs of non-lbisted peripheral modules.
2. Configure RGM functional reset sources as interrupts before Self-Test as per the considerations mentioned in Reset events and configurations while Self-Test, i.e., the SWTs should be disabled, the JTAGC should not be in states which can cause a JTAG reset event (EXTEST, HIGHZ, CLAMP instructions) and the software should not trigger a functional reset event via MC_ME or MDM_AP while the selftest is ongoing. This is to avoid any abort scenario due to any functional reset source as Self-Test is initiated by SW.

There is no issue with Reset pin as Reset pin is a source of destructive reset and will reset STCU2 and Self-Test related logic etc.

### 51.1.17 End of Self-Test

The Self-Test completion is a source of functional reset to the device, by which the device executes a functional reset sequence. The logic which is LBISTed additionally undergoes a complete POR sequence to ensure that the random content shifted during BIST sequence get cleared.

**At the end of MBIST**, the software should execute below steps:

1. Software reads the MBIST end flag to identify that the MBIST sequence had ended properly without any interruption or abort, by reading STCU2.MBESWn.
2. Software reads the MBIST status to identify whether the MBIST was successful or not, by reading STCU2.MBSSWn.

**NOTE**

The MBIST status should always be read in accordance with MBIST end flag. The STCU2.MBSSWn status is irrelevant if STCU2.MBESWn indicates that MBIST execution is incomplete.

**At the end of LBIST**, the software should execute below steps:

1. Software reads the LBIST end flag to identify that the LBIST sequence had ended properly without any interruption or abort, by reading STCU2.LBESWn.
2. Software reads the LBIST status to identify whether the LBIST was successful or not, by reading STCU2.LBSSWn.

**NOTE**

The LBIST status should always be read in accordance with LBIST end flag. The STCU2.LBSSWn status is irrelevant if STCU2.LBESWn indicates that LBIST execution is incomplete.

## 51.2 Introduction

STCU2 is a comprehensive programmable hardware module that:

- Manages the execution of BISTs
- Indicates whether each BIST passed
- Manages the chip's LBIST and the MBIST blocks

## 51.3 Main features

STCU2 includes:

- System interface for reading from and writing to the STCU2 registers (online) via the CPU
- Programmable scheduler for BIST execution
- Control over LBIST concurrent or sequential execution
- Control over MBIST concurrent or sequential execution
- Programmable LBIST delayed concurrent start
- Programmable internal clock prescaler to reduce internal and BIST clocks
- PLL lock signal monitoring during BIST sequences
- Programmable BIST-sequence-execution watchdog timer that specifies the maximum time allowed for the execution of a BIST sequence
- Fixed register-write-access watchdog timer that specifies the maximum amount of time that STCU2 allows you to write to its registers after you unlock them
- Fields that indicate the pass/fail indication of each BIST, see Types of BIST sequences and Types of BIST partitions
- Fields that indicate the status of each type of online STCU2 internal error condition
- Programmable fault mapping of each BIST for controlling the type of fault that STCU2 reports (recoverable or unrecoverable) when the BIST fails to execute
- Programmable fault mapping of each STCU2 internal error condition for controlling the type of fault that STCU2 reports (recoverable or unrecoverable) when the condition occurs
- Signals to report recoverable and unrecoverable faults to the FCCU module
- Redundant recoverable and unrecoverable fault-generation logic to improve reliability
- FCCU recoverable and unrecoverable fault-injection mechanism
- Global register write-protection mechanism that requires two security key codes
- Global automatic power saving after a BIST sequence is completed when watchdog timer time-out is detected

- Watchdog automatic clock wake-up mechanism when software unlocks the STCU2 registers for write access
- Watchdog automatic power saving when the fixed register-write-access watchdog timer times out

## 51.4 Block diagram

This diagram shows the parts of STCU2:

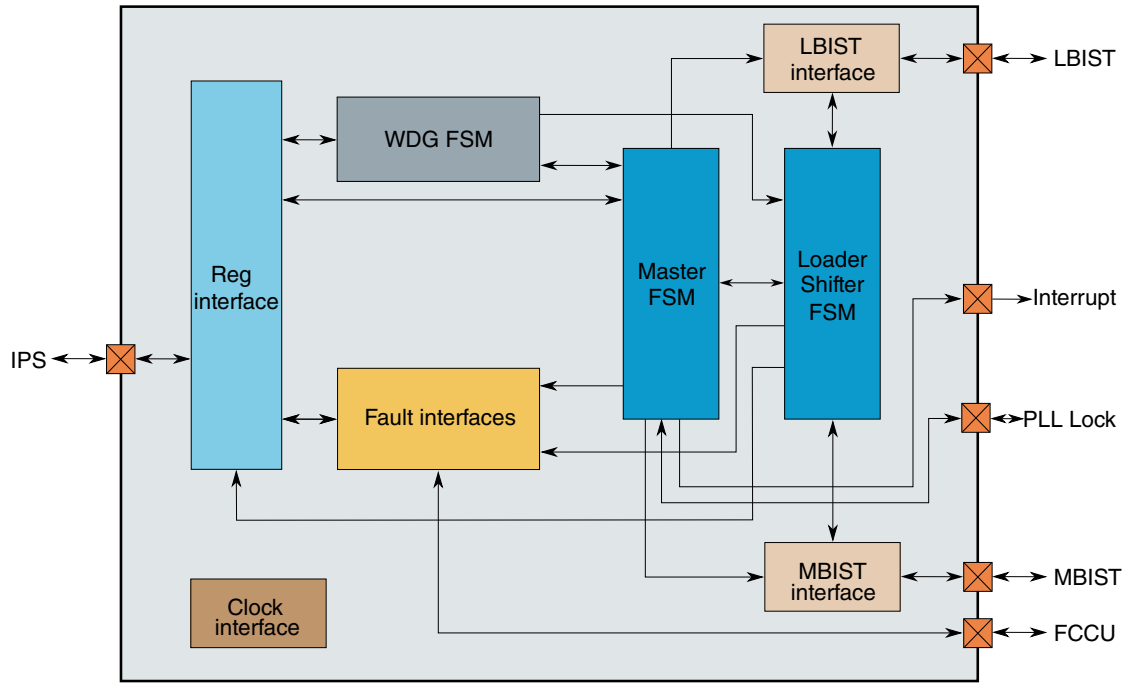



**Figure 188. STCU2 block diagram**

This table describes the parts of STCU2:

**Table 240. STCU2 parts**

| Part | Function |
|---|---|
| Reg interface | Provides access to:<br>• Registers<br>• The security key logic<br>• The IPS interface |
| Fault interfaces | Performs the following tasks:<br>• Collects the fault conditions caused by STCU2 internal error conditions and each BIST executed in a sequence<br>• Sets the global recoverable or unrecoverable status flag, depending on the fault mapping for a given BIST<br>• Manages the recoverable and unrecoverable fault lines to and from the FCCU module |

*Table continues on the next page...*

Table 240. STCU2 parts (continued)

| Part | Function |
|---|---|
| | • Manages the set/clear injection mechanism provided by the FCCU module<br>To improve the intrinsic reliability of this critical logic, the generation logic is duplicated. |
| Clock interface | Manages the internal and the BIST clock prescaler, the internal clock-gating power saving, and the wake-up clock feature |
| WDG FSM | Provides the following:<br>• A programmable BIST-sequence-execution watchdog timer that specifies the maximum time allowed for the execution of a BIST sequence<br>• A fixed register-write-access watchdog timer that specifies the maximum amount of time that STCU2 allows you to write to its registers after you unlock them |
| Master FSM | Coordinates and schedules all of the operations performed during a BIST sequence |
| Loader Shifter FSM | Programs the BIST registers and reads back the data to be checked at the end of each test operation |
| LBIST interface | Provides the interface between the chip's LBIST engines and the STCU2 controllers |
| MBIST interface | Provides the interface between the MBIST controller and the STCU2 controller |

## 51.5 Peripheral bus interface

The peripheral bus interface is a slave bus used for configuration purposes via CPU. The module supports the following bus read operations:

- Word (32 bits) data read operations to any registers
- Any other operation is not supported.

Word, low and high half-words, byte operations are supported in write mode only:

- Write access is allowed by software only when CFG[WRP] is cleared. The default value of CFG[WRP] is cleared, exceptions are the following read/write fields: CFG[WRP], ERR_STAT[UFSF] and ERR_STAT[RFSF].
- STCU2 Configuration (CFG) should only be accessed in 32 bit (partial access not allowed).

The STCU2 module generates a transfer error in the following cases:

- Any read access to the registers after writing Key1 and before writing Key2 (in online ) and before register write-access watchdog timer expires.
- Any write/read access to the register addresses not mapped on the peripheral but included in the address space of the peripheral
- Any write/read operation different from byte/halfword/word (free byte enables or other operations) on each register
- Any write operation on Double Security Key register applying a wrong sequence of keys (the two write operations cannot be interleaved with other access to STCU2 registers)
- Any write operation performed on a register when the Double Security keys have not been applied

- Any write operation performed on registers when CFG[WRP] is set and the access is performed through software (internal peripheral interface). The exceptions are SKC register (for valid keys value and sequence) in which STCU2 does not assert transfer error.
- Any write operation performed on Read Only registers

The registers of the STCU2 module are accessible (read/write) in each access mode: user, supervisor, or test.

In case there are write operations on bits marked as reserved, the transfer error is not generated.

**NOTE**

See the chip-specific STCU2 information for the wait time necessary to write to the online registers.

## 51.6 BISTs and BIST partitions

### 51.6.1 Definition: BIST

A BIST is a test that the chip can execute to verify the functional integrity of a part of itself. The chip uses STCU2 and other on-chip hardware to execute a BIST.

### 51.6.2 Definition: BIST partition

A BIST partition is a part of a chip for which a BIST has been defined. The hardware that is included in a given BIST partition is chip-specific. For a list of the hardware included in each of the BIST partitions on this chip, see the chip-specific STCU2 information.

### 51.6.3 Example: BIST partitions on a chip

This is an example of a chip with six BIST partitions:

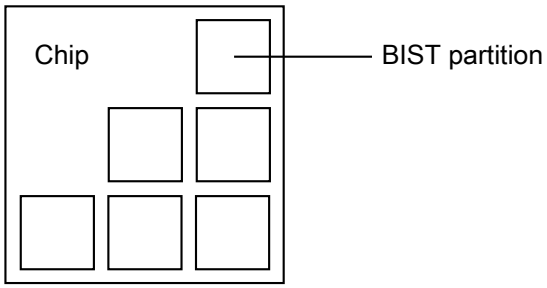



**Figure 189. BIST partitions on a chip**

### 51.6.4 Types of BIST partitions

The STCU2 module on this chip supports the following types of BIST partitions:

**Table 241. Types of BIST partitions**

| BIST partition type | Description |
|---|---|
| MBIST partition | An SRAM or ROM block |
| LBIST partition | One or more digital modules |

## 51.7 BIST sequences

### 51.7.1 Definition: BIST sequence

A BIST sequence is a programmable series of one or more phases, each of which executes one or more individual BISTs.

### 51.7.2 STCU2 executes MBISTs before LBISTs

If a BIST sequence includes both MBISTs and LBISTs; STCU2 executes the MBISTs first.

### 51.7.3 Example: Single-phase BIST sequence

This is an example of a single-phase BIST sequence in which STCU2 executes only one BIST:

**Table 242. Example: Single-phase BIST sequence**

| Phase | BIST executed |
|---|---|
| 0 | MBIST 16 |

### 51.7.4 Example: Multi-phase BIST sequence

This is an example of a multiphase BIST sequence in which STCU2 executes more than one BIST in parallel in some phases:

**Table 243. Example: Multi-phase BIST sequence**

| Phase | BISTs executed |
|---|---|
| 0 | MBISTs 7, 30 |
| 1 | MBIST 23 |
| 2 | MBISTs 6, 16, 29, 23 |
| 3 | LBIST 12 |
| 4 | LBISTs 13, 8, 11 |
| 5 | LBISTs 19, 6 |

### 51.7.5 Types of BIST sequences

The STCU2 module on this chip supports these types of BIST sequences:

**Table 244. Example: Types of BIST sequences**

| BIST sequence type | Description |
|---|---|
| Online BIST sequence | A BIST sequence that STCU2 executes during runtime at the request of software. Application software configures and initiates execution of the BIST sequence by loading values into STCU2 registers. |

### 51.7.6 Supported BIST sequences

NXP supports only a specific set of BIST sequences for this chip. For the list of supported BIST sequences, see the chip-specific STCU2 information.

## 51.8 Functional description

### 51.8.1 FSM description

The module has three FSM that work together: Master FSM, Loader/Shifter FSM, and the Watchdog FSM. Basically the Master FSM is the core unit of the STCU2 module. It coordinates all the self-test operations and the other FSM. The Loader/Shifter FSM is used to program the MBIST and the LBIST parameters and to retrieve the related results depending on the parameters stored into the STCU2 registers and under the control of the Master FSM. The Watchdog FSM evaluates all the schedule time for MBIST and LBIST and the time-out in case of wrong STCU2 programming.

### 51.8.2 BIST scheduling

STCU2 is designed to program the parallel/serial execution of the MBIST or LBIST depending on the power, timing, and coverage constraints.

The mechanism used to provide this flexibility is a linked list of BIST descriptors where the starting pointer is defined in CFG[PTR]. The first LBIST is mapped on 0, the second on 1, and so on. A BIST descriptor identifies a LBIST or MBIST via an index that is associated with its control register (STCU2 LBIST Control (LB_CTRL0) or STCU2 MBIST Control (MB_CTRL0 - MB_CTRL11)). The first MBIST is mapped on (00000080h + 0), the second on (00000080h + 1), and so on. The additional pointers of the linked list are in LB_CTRL*n*[PTR] for LBIST *n* (where *n* is the selected LBIST) and MB_CTRL*m*[PTR] for MBIST *m* (where *m* is the selected MBIST) and must be populated depending on the selected run sequence. The additional fields LB_CTRL[CSM] and MB_CTRL[CSM] provide the flexibility to run concurrently or sequentially the chosen set of the LBIST or MBIST or to close the linked list by setting the NIL pointer.

### 51.8.3 FCCU interface

The FCCU interface is the hardware flag mechanism towards the system, to indicate the occurrence of an Unrecoverable fault and/or a Recoverable fault failure during the Self Test sequence. Two independent signals have been used to mitigate the common cause faults.

To diagnose physical defects on the two fault signals, a fault injection mechanism is also provided. In this case, the FCCU interface allows the user application to check the integrity of the UF and RF connection lines between the STCU2 and the FCCU. Refer to the description of FCCU fault injection mechanism to understand how the UF/RF set/clear mechanism works.

### 51.8.4 Watchdogs

The STCU2 implements different watchdogs to ensure that operations are finished in time.

#### 51.8.4.1 BIST watchdog timer

The LBIST and MBIST execution time has to be configured as described in STCU2 Watchdog Granularity (WDG), to account for the overall execution time of the self test sequence. In case the selected LBISTs or MBISTs are not yet completed during assigned time, the current LBISTs or MBISTs execution is interrupted and a failure is flagged into ERR_STAT[WDTOSW] and MBESWx or LBESW.

In case of multiple sequential run in the same online session and the time-out happens in the middle of a sequential run, the next sequential run will be skipped and the execution ends with the current updated status of the registers reported above.

#### 51.8.4.2 Register write-access watchdog timer

As explained in the STCU2 SK Code (SKC):

- A key mechanism protects STCU2 registers during the self-test configuration phase by preventing any unwanted access.
- A hardware watchdog timer locks register-write access after a number of STCU2 clock cycles. To refresh the hardware watchdog timer before it times out, write Key1 and Key2 sequences.

AUTOLOCK_VALUE is the number of STCU2 clock cycles after which the hardware watchdog timer locks register-write access. See the chip-specific STCU2 information for the value of AUTOLOCK_VALUE.

The hardware watchdog timer is particularly useful in case CFG[WRP] is 0 during the software self-test configuration. In this case, the software application might enable write access to the STCU2 registers.

## 51.9 STCU2 register descriptions

The STCU2 registers are listed in this section.

**NOTE**

Always write the reset value for the Reserved fields.

During online self-test, after the STCU2 registers are configured, they must not be overridden via internal peripheral system until the self-test is complete.

### 51.9.1 STCU2 memory map

STCU base address: 403A_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 4h | STCU2 Run Software (RUNSW) | 32 | RW | 0000_0000h |
| 8h | STCU2 SK Code (SKC) | 32 | W | 0000_0000h |
| Ch | STCU2 Configuration (CFG) | 32 | RW | 0000_0000h |
| 14h | STCU2 Watchdog Granularity (WDG) | 32 | RW | 0000_FFFFh |
| 24h | STCU2 Error (ERR_STAT) | 32 | RW | 0000_0000h |
| 28h | STCU2 Error FM (ERR_FM) | 32 | RW | 0000_0000h |
| 4Ch | STCU2 Online LBIST Status (LBSSW0) | 32 | R | 0000_0000h |
| 5Ch | STCU2 Online LBIST End Flag (LBESW0) | 32 | R | 0000_0000h |
| 7Ch | STCU2 Online LBIST Unrecoverable FM (LBUFM0) | 32 | RW | 0000_0000h |
| 10Ch | STCU2 Online MBIST Status (MBSSW0) | 32 | R | 0000_0000h |
| 14Ch | STCU2 Online MBIST End Flag (MBESW0) | 32 | R | 0000_0000h |
| 18Ch | STCU2 MBIST Unrecoverable FM (MBUFM0) | 32 | RW | 0000_0000h |
| 200h | STCU2 LBIST Control (LB_CTRL0) | 32 | RW | 0000_0000h |
| 204h | STCU2 LBIST PC Stop (LB_PCS0) | 32 | RW | 0000_0000h |
| 220h | STCU2 Online LBIST MISR Expected Low (LB_MISRELSW0) | 32 | RW | FFFF_FFFFh |
| 224h | STCU2 Online LBIST MISR Expected High (LB_MISREHSW0) | 32 | RW | FFFF_FFFFh |
| 228h | STCU2 Online LBIST MISR Read Low (LB_MISRRLSW0) | 32 | R | 0000_0000h |
| 22Ch | STCU2 Online LBIST MISR Read High (LB_MISRRHSW0) | 32 | R | 0000_0000h |
| 2200h | STCU2 Algorithm Select (ALGOSEL) | 32 | RW | 0000_0000h |
| 220Ch | STCU2 MBIST Stagger (STGGR) | 32 | RW | 0000_0000h |
| 2210h | STCU2 BIST Start (BSTART) | 32 | RW | 0000_0000h |
| 2214h - 2240h | STCU2 MBIST Control (MB_CTRL0 - MB_CTRL11) | 32 | RW | 0000_0000h |

## 51.9.2 STCU2 Run Software (RUNSW)

### Offset

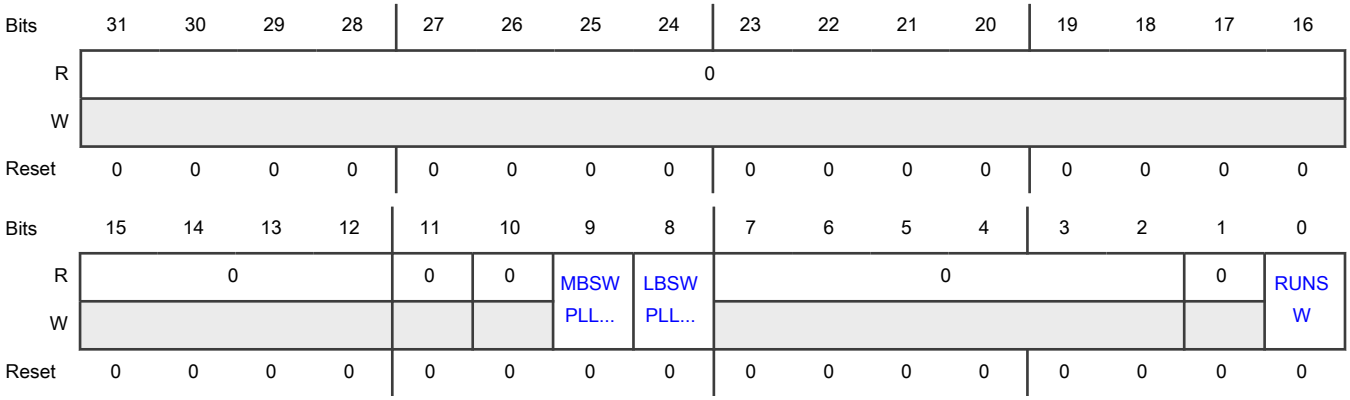
| Register | Offset |
|---|---|
| RUNSW | 4h |

### Function

The RUNSW register defines the RUNSW bit to start the online self-testing procedure.

The R/W fields in this register are readable at any time. However, you can write to these fields only when CFG[WRP] = 0.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11<br>— | Reserved |
| 10<br>— | Reserved |
| 9<br>MBSWPLLEN | Online MBIST with PLL Enabled<br>0b - Online MBIST is executed without using the on-chip PLL.<br>1b - Online MBIST is executed using the PLL configuration provided by software. STCU2 does not take the PLL control but monitors the PLL lock signal to check if PLL is working correctly. |
| 8<br>LBSWPLLEN | Online LBIST with PLL Enabled<br>0b - Online LBIST is executed without using the on-chip PLL. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Online LBIST is executed using the PLL configuration provided by the software. STCU2 does not take the PLL control but monitors the PLL lock signal to check if PLL is working correctly. |
| 7-2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>RUNSW | The RUNSW bit is automatically cleared by STCU2 when the online self-testing procedure is complete.<br>0b - Idle<br>1b - Online self-testing procedure is running |

### 51.9.3 STCU2 SK Code (SKC)

#### Offset

| Register | Offset |
|---|---|
| SKC | 8h |

#### Function

The SKC register implements the security key code mechanism needed to access in write mode to the other STCU2 registers.

To unlock STCU2 access after the STCU2 asynchronous reset and at the end of the STCU2 run, the software (IPS bus) need to apply the following sequence:

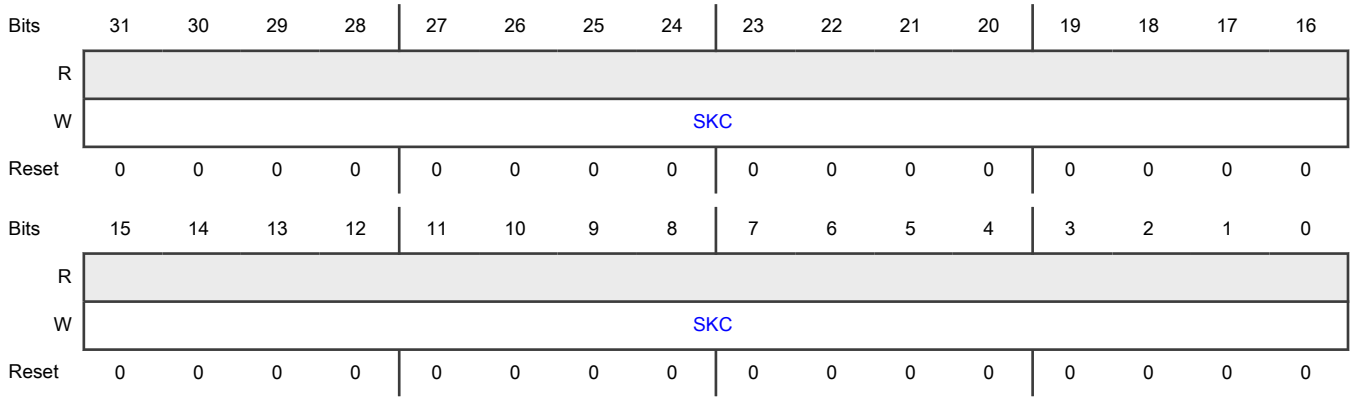
- write the key1 into the SKC register
- write the key2 into the SKC register

Depending on the online test step, the two keys are different. Byte write operation is not allowed because the full key has to be recognized as one unit.

A hard-coded WDG counter (Register write-access watchdog timer) starts decrementing its value right after POR is deasserted. Therefore, user must ensure to complete unlocking and programming sequence before its expiry. In case the STCU2 register access lasts more cycles than the one defined in the hard-coded WDG timeout, the STCU2 write access is locked and the WDG and register interface clocks are switched off. Also, in this case, to enable the write access to the STCU2 again and the WDG and register interface clocks, it is required to apply the sequence again. In case of invalid access or sequence (Key1/2 have to be applied consecutively), a transfer error on the IPS is asserted. The STCU2 write access is locked and to unlock the access, the sequence has to be applied again. This can also happen if watchdog expires between key1 and key2. In this case, user needs to apply the key1-key2 sequence again. In case it is required to extend the STCU2 register access cycles before the hard-coded WDG timeout expires, Key1 and Key2 sequences need to be applied. The effect of this write operation is to re-initialize the WDG timeout counter. The STCU2 write access is locked and to unlock the access, the sequence has to be applied again.

The SKC register is not readable. The value 00000000h is always returned in case of read operation.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>SKC | STCU2 SK Code<br>STCU2 security key code for online test<br>= 753F924Eh: Key1 to unlock the write access the STCU2<br>= 8AC06DB1h: Key2 to unlock the write access the STCU2 |

## 51.9.4 STCU2 Configuration (CFG)

### Offset

| Register | Offset |
|---|---|
| CFG | Ch |

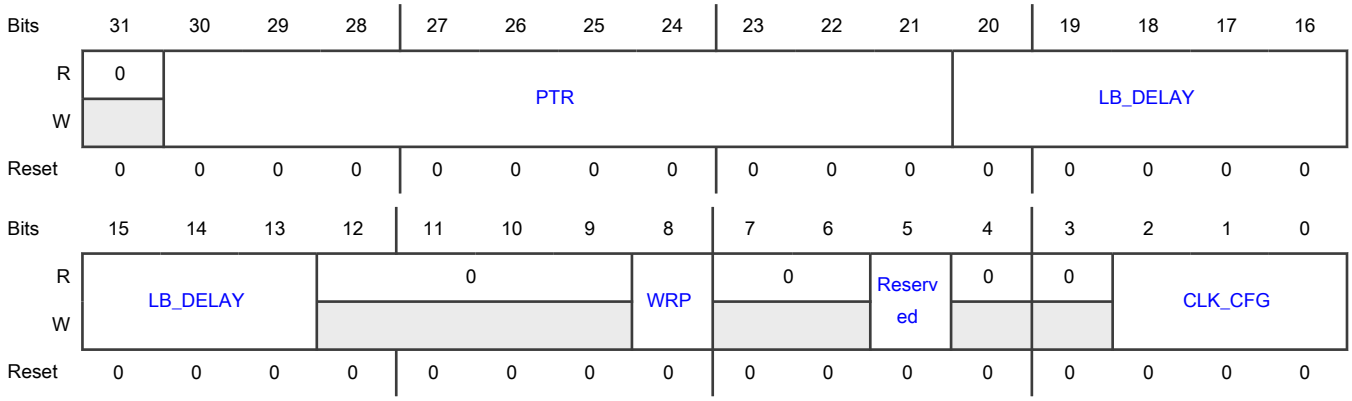
### Function

The CFG register includes the global configuration of the STCU2 and can be updated in the online test steps.

The access to this register is described in the following figure. It further depends on the state of the WRP bit as follows:

- When WRP = 0: The register can be written during the online self-test case without restrictions.
- When WRP = 1:
  - The only bit that can be written without any restriction is WRP.
  - In case there are software operations that write all the register's bits, a transfer error is raised only if the value of the selected byte written differs from the current status of the register. This functionality has been implemented to prevent potential compilation behavior that might invalidate this single bit clean capability.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-21<br>PTR | First LBIST or MBIST pointer<br>PTR defines the logical pointer to the first LBIST or MBIST to be scheduled when the self-testing procedure is enabled. PTR is the entry pointer to a linked list of BIST descriptors. If PTR = 0 then the LBIST0 is initially scheduled. See BIST scheduling for details.<br>0h to (01h - 1): pointer to LBIST<br>00000080h to (00000080h + 00Ch - 1): pointer to MBIST<br>3FFh: pointer to NIL. No BIST execution.<br>others: invalid pointer => an error is set into the ERR_STAT[INVPSW]. |
| 20-13<br>LB_DELAY | Delay LBIST run<br>LB_DELAY defines the delay between the LBIST starts when more than a single LBIST is selected to be executed concurrently with the purpose of smoothing the power consumption transient. The allowed delay time are these:<br>00h: No delay<br>01h: 1 x 16 STCU2 CORE_CLK cycles<br>02h: 2 x 16 STCU2 CORE_CLK cycles<br>03h: 3 x 16 STCU2 CORE_CLK cycles<br>...<br>FDh: 253 x 16 STCU2 CORE_CLK cycles<br>FEh: 254 x 16 STCU2 CORE_CLK cycles<br>FFh: 255 x 16 STCU2 CORE_CLK cycles |
| 12-9 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 8<br>WRP | Write Protection<br>0: Specific STCU2 registers can be written through IPS bus interface<br>1: STCU2 registers cannot be written through IPS, preventing any user application write operation |
| 7-6<br>— | Reserved |
| 5<br>— | Reserved |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2-0<br>CLK_CFG | Logic, Memory BIST, and STCU2 CORE_CLK configuration<br>CLK_CFG defines the ratio between the sys_clk and the internal clock used to program both the LBIST and the MBIST and the STCU2 CORE_CLK. The punch-out mechanism is used to generate the derived clocks. The following configurations are allowed:<br>000b - sys_clk/1<br>001b - sys_clk/2<br>010b - sys_clk/3<br>011b - sys_clk/4<br>100b - sys_clk/5<br>101b - sys_clk/6<br>110b - sys_clk/7<br>111b - sys_clk/8 |

### 51.9.5 STCU2 Watchdog Granularity (WDG)

#### Offset

| Register | Offset |
|---|---|
| WDG | 14h |

#### Function

The WDG register defines the time budget of LBIST and MBIST execution providing a protection mechanism in case of dead-lock or endless conditions during the self-test procedure.

In case online self-test sequence is run, it defines the timeout of the execution run.

The R/W fields in this register are readable at any time. You can write to these fields when online self-test phase is active and CFG[WRP] = 0.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WDGEOC | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WDGEOC | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>WDGEOC | Watchdog End of Count Timer<br>This value has to be set to define the time budget related to the online self-test execution and check that everything is correctly working within this slot of time. The delay time slots that are allowed are as follows:<br>0000 0000h: 1 x 16 STCU2 CORE_CLK cycles<br>0000 0001h: 2 x 16 STCU2 CORE_CLK cycles<br>0000 0002h: 3 x 16 STCU2 CORE_CLK cycles<br>...<br>FFFF FFFDh: 4294967294 x 16 STCU2 CORE_CLK cycles<br>FFFF FFFEh: 4294967295 x 16 STCU2 CORE_CLK cycles<br>FFFF FFFFh: 4294967296 x 16 STCU2 CORE_CLK cycles |

## 51.9.6 STCU2 Error (ERR_STAT)

### Offset

| Register | Offset |
|---|---|
| ERR_STAT | 24h |

### Function

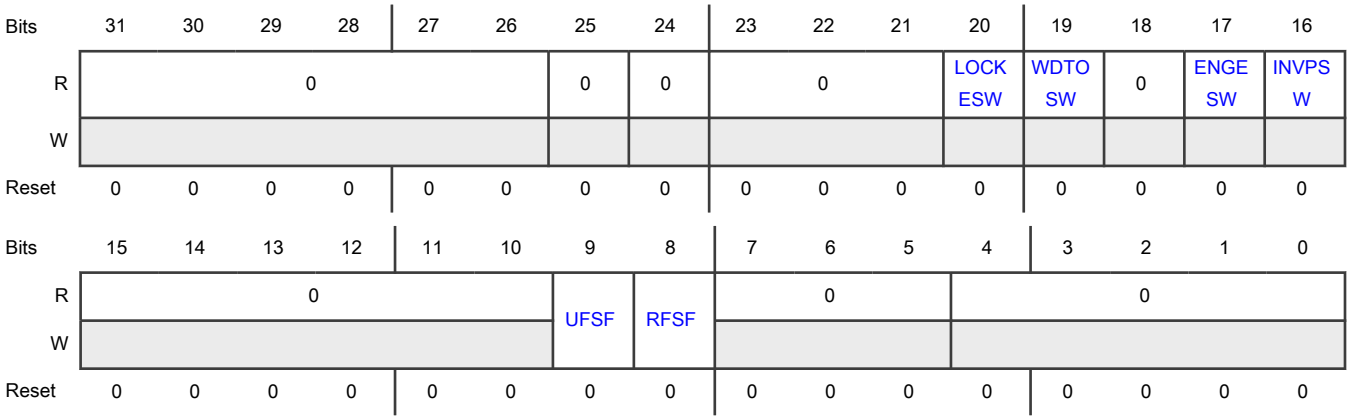
The ERR_STAT register includes the status flags related to the STCU2 internal error conditions occurred during the configuration or the online self-testing execution.

The UFSF and RFSF can be set/cleared using the FCCU dedicated channels.

The access to this register is described in the following figure and as follows:

- If you select the byte write capability to write only UFSF and RFSF, then there is no restriction in writing these bits.
- If your software performs the write operations on other bits besides UFSF/RFSF, then a transfer error is generated only if the value you are writing to the other bits differs from their value currently stored in the register. This functionality has been implemented to prevent potential compilation behavior that might invalidate the UFSF/RFSF single bit set/clean capability.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25<br>— | Reserved |
| 24<br>— | Reserved |
| 23-21<br>— | Reserved |
| 20<br>LOCKESW | Online LOCK error<br>You can always read this field. The content of this field is initialized to its reset value when RUNSW[RUNSW] is set to 1.<br>0b - In case PLL is enabled, it is correctly locked during the self-test sequence<br>1b - When the PLL is enabled, this flag highlights that there has been an unexpected PLL unlock(loss-of-lock) event during the online self-test sequence execution. The online self-test run is stopped and the status of the currently running LBISTs or MBISTs is saved in the related registers. The LOCK signal is monitored during the LBIST run when RUNSW[LBSWPLLEN] is set and/or during the MBIST run when RUNSW[MBSWPLLEN] = 1. |
| 19<br>WDTOSW | Online watchdog timeout |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | You can always read this field. The content of this field is initialized to its reset value when RUNSW[RUNSW] is set to 1.<br>0b - LBIST and MBIST time slots completed within the assigned watchdog time.<br>1b - LBIST and MBIST time slots not completed within the assigned watchdog time or there are internal mismatches among End of Execution signals. |
| 18<br>— | Reserved |
| 17<br>ENGESW | Online engine error<br>You can always read this field. The content of this field is initialized to its reset value when RUNSW[RUNSW] is set to 1.<br>0b - Valid engine execution<br>1b - Invalid engine execution. The error conditions that set this bit are FSM, protocol error, and so on. |
| 16<br>INVPSW | Online invalid pointer<br>You can always read this field. The content of this field is initialized to its reset value when RUNSW[RUNSW] is set to 1.<br>0b - Valid linked pointer list<br>1b - Invalid linked pointer list. The following conditions set this bit: Initial LBIST or MBIST pointer is out of range; LBIST is selected when MBIST is concurrently running or vice versa; Error in the LBIST/MBIST linking (execution generates an infinite loop). |
| 15-10<br>— | Reserved |
| 9<br>UFSF | Unrecoverable Faults Status Flag<br>This flag reports the global status of the Unrecoverable Faults(UF). This field can be set or cleared using the FCCU dedicated channel, and can also be set or cleared by software.<br>0b - No errors that trigger the UF condition.<br>1b - There are errors that trigger the UF condition. |
| 8<br>RFSF | Recoverable Faults Status Flag<br>This flag reports the global status of the Recoverable Fault (RF). This field can be set or cleared using the FCCU dedicated channel, and can also be set or cleared by software.<br>0b - No errors that trigger the Recoverable Faults condition<br>1b - There are errors that trigger the Recoverable Faults condition |
| 7-5<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 4-0<br>— | Reserved<br>Write may occur on these fields but writes have no effect. |

## 51.9.7 STCU2 Error FM (ERR_FM)

### Offset

| Register | Offset |
|---|---|
| ERR_FM | 28h |

### Function

The ERR_FM register defines the fault mapping of the STCU2 faults described in the register ERR_STAT in terms of UF or RF. All sources of internal faults can be routed to UF and RF.

The R/W fields in this register are readable at any time. You can write to these fields when online self-test phase is active and CFG[WRP] = 0.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | LOCKEUFM | WDTOUFM | Reserved | ENGEUFM | INVPUFM |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-5<br>— | Reserved |
| 4<br>LOCKEUFM | PLL LOCK Unrecoverable Fault Mapping<br>0b - Recoverable Fault Mapping<br>1b - Unrecoverable Fault Mapping |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 3<br>WDTOUFM | Watchdog Timeout Unrecoverable Fault Mapping<br>0b - Recoverable Fault Mapping<br>1b - Unrecoverable Fault Mapping |
| 2<br>— | Reserved |
| 1<br>ENGEUFM | Engine Error Unrecoverable Fault Mapping<br>0b - Recoverable Fault Mapping<br>1b - Unrecoverable Fault Mapping |
| 0<br>INVPUFM | Invalid Pointer Unrecoverable Fault Mapping<br>0b - Recoverable Fault Mapping<br>1b - Unrecoverable Mapping |

## 51.9.8 STCU2 Online LBIST Status (LBSSW0)

### Offset

| Register | Offset |
|---|---|
| LBSSW0 | 4Ch |

### Function

This register includes the results corresponding to the execution of the selected online LBIST.

The size of the register depends on the number of LBIST .

The content of this register is initialized to its reset value when RUNSW[RUNSW] is set to 1.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | LBSSW0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>LBSSWn | LBSSWn<br>online status of the selected LBIST<br>0b - Failed LBIST execution<br>1b - Successful LBIST execution |

## 51.9.9 STCU2 Online LBIST End Flag (LBESW0)

### Offset

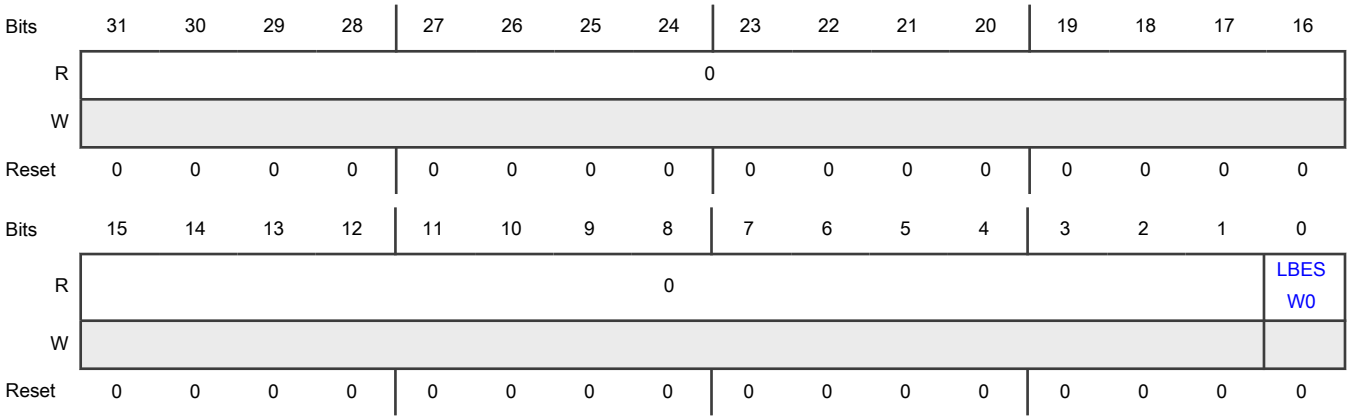
| Register | Offset |
|---|---|
| LBESW0 | 5Ch |

### Function

This register includes the results corresponding to the execution of the selected online LBIST.

The size of the register depends on the number of LBIST .

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 0<br>LBESWn | LBESW<br>LBESWx: online LBIST end status<br>0b - LBIST execution not yet completed<br>1b - LBIST execution finished |

## 51.9.10 STCU2 Online LBIST Unrecoverable FM (LBUFM0)

### Offset

| Register | Offset |
|---|---|
| LBUFM0 | 7Ch |

### Function

This register defines the fault mapping of each LBIST in terms of UF or RF.

The size of the register 0 depends on the number of LBIST .

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | LBUFM0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>LBUFMn | LBIST Unrecoverable Fault Mapping<br>0b - Recoverable Fault mapping<br>1b - Unrecoverable Fault mapping |

## 51.9.11 STCU2 Online MBIST Status (MBSSW0)

### Offset

| Register | Offset |
|---|---|
| MBSSW0 | 10Ch |

### Function

This register includes the results corresponding to the execution of the selected online MBIST in the range 0:11.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value when RUNSW[RUNSW] is set to 1.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | MBSSW11 | MBSSW10 | MBSSW9 | MBSSW8 | MBSSW7 | MBSSW6 | MBSSW5 | MBSSW4 | MBSSW3 | MBSSW2 | MBSSW1 | MBSSW0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11-0<br>MBSSWn | MBSSW<br>Online status of the selected MBISTn (where n = 11:0).<br>0b - Failed MBIST execution<br>1b - Successful MBIST execution |

## 51.9.12 STCU2 Online MBIST End Flag (MBESW0)

### Offset

| Register | Offset |
|---|---|
| MBESW0 | 14Ch |

### Function

This register includes the End Flag related to the execution of the selected online MBIST in the range 0:11.

The size of the register depends on the number of BISTed RAMs/ROMs.

The content of this register is initialized to its reset value when RUNSW[RUNSW] is set to 1.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | MBESW11 | MBESW10 | MBESW9 | MBESW8 | MBESW7 | MBESW6 | MBESW5 | MBESW4 | MBESW3 | MBESW2 | MBESW1 | MBESW0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11-0<br>MBESWn | Online MBISTn (where n = 11:0) end status.<br>0b - MBIST execution not yet completed<br>1b - MBIST execution finished |

## 51.9.13 STCU2 MBIST Unrecoverable FM (MBUFM0)

### Offset

| Register | Offset |
|---|---|
| MBUFM0 | 18Ch |

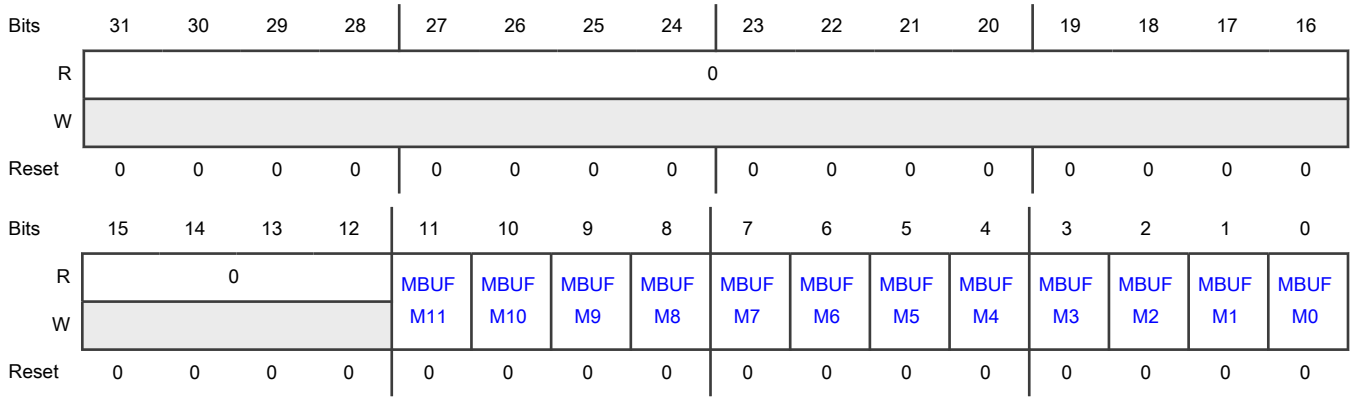
### Function

This register defines the fault mapping, in terms of UF or RF, of the MBIST in the range 0:11

The size of the register depends on the number of BISTed RAMs/ROMs.

The R/W fields in this register are readable at any time. You can write to these fields when online self-test phase is active and CFG[WRP] = 0.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11-0<br>MBUFMn | MBESW<br>Online end status of MBISTn (where n = 11:0).<br>0b - Recoverable fault mapping<br>1b - Unrecoverable fault mapping |

## 51.9.14 STCU2 LBIST Control (LB_CTRL0)

### Offset

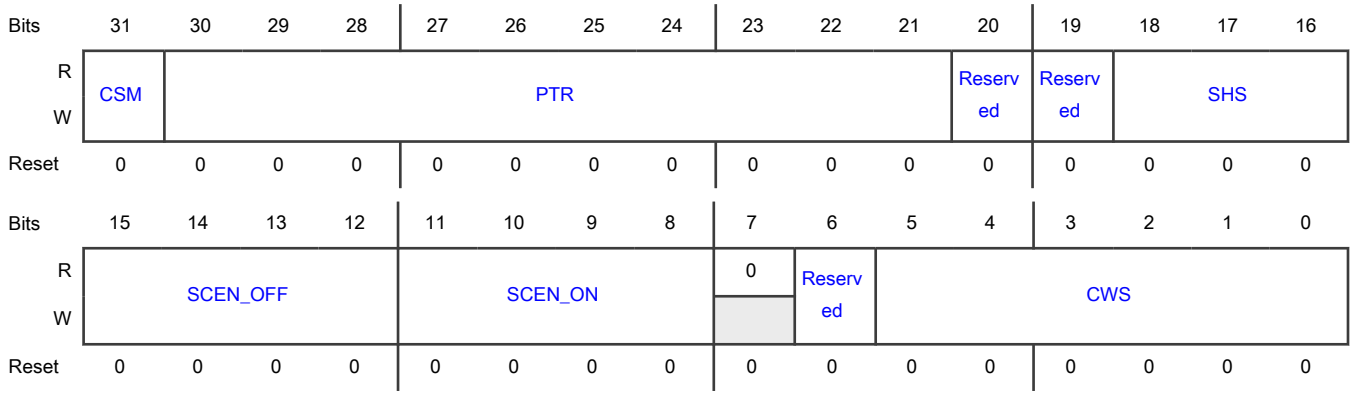
| Register | Offset |
|---|---|
| LB_CTRL0 | 200h |

### Function

This register defines the control setting of each LBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when online self-test phase is active and CFG[WRP] = 0.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>CSM | Concurrent/sequential mode<br>The next LBIST is scheduled concurrently to the current one if the CSM bit is set to 1; otherwise, it is scheduled sequentially to the completion of the current LBIST execution.<br>0b - Sequential mode<br>1b - Concurrent mode |
| 30-21<br>PTR | Next LBIST or MBIST pointer<br>PTR defines the logical pointer to the next LBIST or MBIST to be scheduled. The next LBIST or MBIST is scheduled concurrently to the current one if the CSM bit is set to 1, otherwise it is scheduled sequentially to the completion of the current LBIST execution. In case of NIL pointer, the CSM bit has to be set Sequential (0) to define this is the end of the list. The self-testing procedure stops after the last BIST in the configuration chain is complete. See BIST scheduling for details.<br>0h to (01h - 1): pointer to NLBIST-1<br>00000080h to (00000080h + 00Ch - 1): pointer to MBIST<br>3FFh: pointer to NIL. No BIST execution.<br>others: invalid pointer => an error is set into the ERR_STAT[INVPSW]. |
| 20<br>— | Reserved<br>**NOTE**<br>This reserved field is writable but do not write any value to it other than its reset value |
| 19<br>— | Reserved<br>**NOTE**<br>This reserved field is writable but do not write any value to it other than its reset value |
| 18-16<br>SHS | Shift speed<br>SHS defines the shift speed |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 000b - Shift at full rate (BIST clock).<br>001b - Shift at 1/2 rate (BIST clock).<br>010b - Shift at 1/3 rate (BIST clock).<br>011b - Shift at 1/4 rate (BIST clock).<br>100b - Shift at 1/5 rate (BIST clock).<br>101b - Shift at 1/6 rate (BIST clock).<br>110b - Shift at 1/7 rate (BIST clock).<br>111b - Shift at 1/8 rate (BIST clock). |
| 15-12<br>SCEN_OFF | Scan enable OFF<br>SCEN_OFF information is used to configure the lbist controller hardware to generate off_cycles, delay cycles during the scan enable off transition.<br>**NOTE**<br>SCEN_OFF must be programmed to a value >=1.<br>0000b - 0 delay cycles<br>0001b - 1 delay cycle<br>0010b - 2 delay cycles<br>0011b - 3 delay cycles<br>0100b - 4 delay cycles<br>0101b - 5 delay cycles<br>0110b - 6 delay cycles<br>0111b - 7 delay cycles<br>1000b - 8 delay cycles<br>1001b - 9 delay cycles<br>1010b - 10 delay cycles<br>1011b - 11 delay cycles<br>1100b - 12 delay cycles<br>1101b - 13 delay cycles<br>1110b - 14 delay cycles<br>1111b - 15 delay cycles |
| 11-8<br>SCEN_ON | Scan enable ON<br>SCEN_ON information is used to configure the lbist controller hardware to generate on_cycles, delay cycles during the scan enable on transition, |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>SCEN_ON delay register value must be programmed to a value >=1<br>0000b - 0 delay cycles<br>0001b - 1 delay cycle<br>0010b - 2 delay cycles<br>0011b - 3 delay cycles<br>0100b - 4 delay cycles<br>0101b - 5 delay cycles<br>0110b - 6 delay cycles<br>0111b - 7 delay cycles<br>1000b - 8 delay cycles<br>1001b - 9 delay cycles<br>1010b - 10 delay cycles<br>1011b - 11 delay cycles<br>1100b - 12 delay cycles<br>1101b - 13 delay cycles<br>1110b - 14 delay cycles<br>1111b - 15 delay cycles |
| 7<br>— | Reserved |
| 6<br>— | Reserved |
| 5-0<br>CWS | Capture window size<br>CWS defines the capture window size.<br>00_0000b - Illegal<br>00_0001b - Controller waits 1 shift cycle for capture to finish.<br>00_0010b - Controller waits 2 shift cycles for capture to finish.<br>00_0011b - Controller waits 3 shift cycles for capture to finish.<br>00_0100b - Controller waits 4 shift cycles for capture to finish.<br>00_0101b - Controller waits 5 shift cycles for capture to finish.<br>00_0110b - Controller waits 6 shift cycles for capture to finish.<br>00_0111b - Controller waits 7 shift cycles for capture to finish. |

## 51.9.15 STCU2 LBIST PC Stop (LB_PCS0)

### Offset

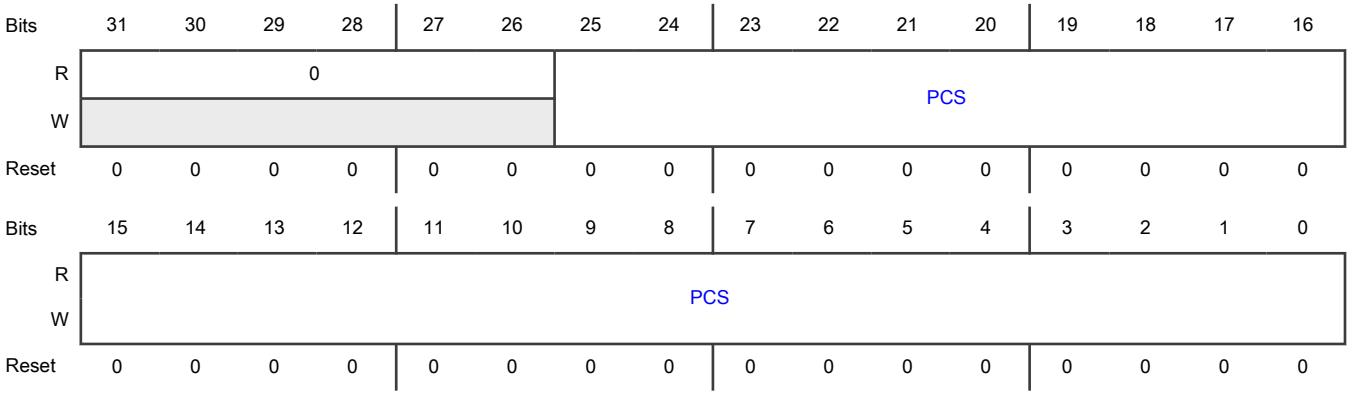
| Register | Offset |
|---|---|
| LB_PCS0 | 204h |

### Function

This register defines the pattern counter stop of each LBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when online self-test phase is active and CFG[WRP] = 0.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25-0<br>PCS | PCS<br>Pattern counter stop<br>PCS defines the pattern counter stop value. |

## 51.9.16 STCU2 Online LBIST MISR Expected Low (LB_MISRELSW0)

### Offset

| Register | Offset |
|---|---|
| LB_MISRELSW0 | 220h |

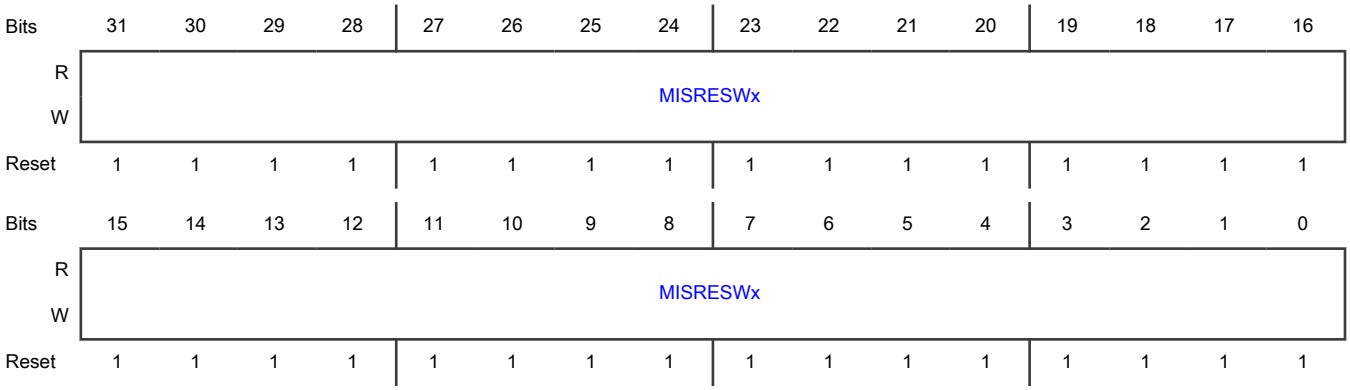
### Function

This register defines bits 32 of the expected MISR of the online LBIST controller.

The size of the register depends on the number of MISR bits of the related LBIST. .

The R/W fields in this register are readable at any time. You can write to these fields when online self-test phase is active and CFG[WRP] = 0.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>MISRESWx | Online MISR expected low bits<br>This field defines 32 bits of the expected MISR. |

## 51.9.17 STCU2 Online LBIST MISR Expected High (LB_MISREHSW0)

### Offset

| Register | Offset |
|---|---|
| LB_MISREHSW0 | 224h |

### Function

The size of the register depends on the number of MISR bits of the related LBIST. .

The R/W fields in this register are readable at any time. You can write to these fields when online self-test phase is active and CFG[WRP] = 0.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MISRESWx | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>MISRESWx | Online MISR Expected High Bits<br>This field defines the 32 bits of the expected MISR. |

## 51.9.18 STCU2 Online LBIST MISR Read Low (LB_MISRRLSW0)

### Offset

| Register | Offset |
|---|---|
| LB_MISRRLSW0 | 228h |

### Function

This register reports 32 bits of the MISR obtained at the end of the online LBIST controller execution.

The size of the register depends on the number of MISR bits of the related LBIST. .

The content of this register is initialized to its reset value when RUNSW[RUNSW] is set to 1.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>MISRRSWx | MISRRSWx<br>Online MISR Read Low Bin<br>This field is equivalent to 32 bits of the MISR obtained at the end of the online LBIST execution. |

## 51.9.19 STCU2 Online LBIST MISR Read High (LB_MISRRHSW0)

### Offset

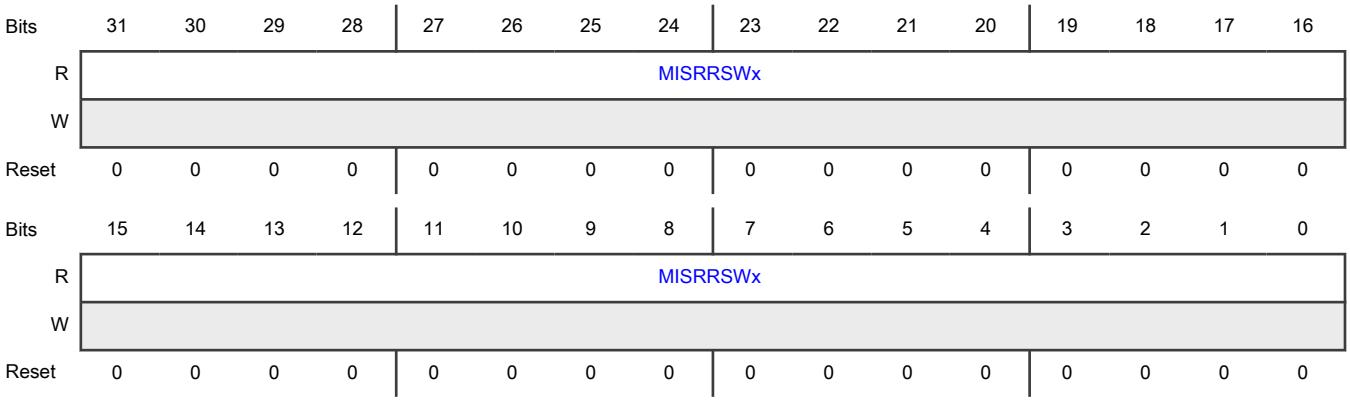
| Register | Offset |
|---|---|
| LB_MISRRHSW0 | 22Ch |

### Function

The size of the register depends on the number of MISR bits of the related LBIST. .

The content of this register is initialized to its reset value when RUNSW[RUNSW] is set to 1.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>MISRRSWx | MISRRSWx<br>Online MISR Read High Bits<br>This field is equivalent to 32 bits of the MISR obtained at the end of the online LBIST execution. |

## 51.9.20 STCU2 Algorithm Select (ALGOSEL)

### Offset

| Register | Offset |
|---|---|
| ALGOSEL | 2200h |

### Function

This is a 32-bit register intended to be programmed by the user to select algorithms to be run on BIST. See the chip-specific STCU2 information for details of this register.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ALGO SEL... | ALGO SEL... | ALGO SEL... | ALGO SEL... | ALGO SEL... | ALGO SEL... | ALGO SEL... | ALGO SEL... | ALGO SEL... | ALGO SEL... | ALGO SEL... | ALGO SEL... | ALGO SEL... | ALGO SEL... | ALGO SEL... | ALGO SEL... |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ALGO SEL... | ALGO SEL... | ALGO SEL... | ALGO SEL... | ALGO SEL... | ALGO SEL... | ALGO SEL9 | ALGO SEL8 | ALGO SEL7 | ALGO SEL6 | ALGO SEL5 | ALGO SEL4 | ALGO SEL3 | ALGO SEL2 | ALGO SEL1 | ALGO SEL0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>ALGOSELn | Algorithm Select |

## 51.9.21 STCU2 MBIST Stagger (STGGR)

### Offset

| Register | Offset |
|---|---|
| STGGR | 220Ch |

### Function

This register allows one to program number of clock cycles between execution of one BIST and the next one.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | STAG | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | STAG | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>STAG | STAG<br>Number of STCU2 CORE_CLK cycles between execution of one BIST and the next one. |

## 51.9.22 STCU2 BIST Start (BSTART)

### Offset

| Register | Offset |
|---|---|
| BSTART | 2210h |

### Function

This is a 32-bit register intended to be programmed by the user to run BISTs with different configuration. See the chip-specific STCU2 information for details of this register.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BSTART31 | BSTART30 | BSTART29 | BSTART28 | BSTART27 | BSTART26 | BSTART25 | BSTART24 | BSTART23 | BSTART22 | BSTART21 | BSTART20 | BSTART19 | BSTART18 | BSTART17 | BSTART16 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BSTART15 | BSTART14 | BSTART13 | BSTART12 | BSTART11 | BSTART10 | BSTART9 | BSTART8 | BSTART7 | BSTART6 | BSTART5 | BSTART4 | BSTART3 | BSTART2 | BSTART1 | BSTART0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>BSTARTn | BIST Start |

## 51.9.23 STCU2 MBIST Control (MB_CTRL0 - MB_CTRL11)

### Offset

For a = 0 to 11:

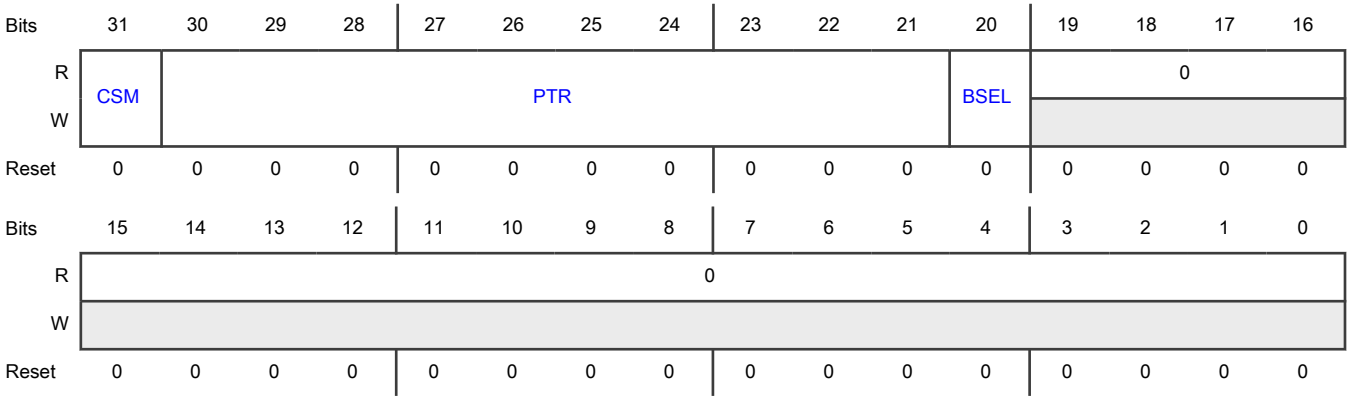
| Register | Offset |
|---|---|
| MB_CTRLa | 2214h + (a × 4h) |

### Function

The MB_CTRL register defines the control setting of MBIST controller.

The R/W fields in this register are readable at any time. You can write to these fields when online self-test phase is active and CFG[WRP] = 0.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>CSM | CSM<br>Concurrent/sequential mode<br>0b - Sequential mode<br>1b - Concurrent mode |
| 30-21 | PTR |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| PTR | Next LBIST or MBIST pointer<br>PTR defines the logical pointer to the next LBIST or MBIST to be scheduled. The next LBIST or MBIST is scheduled concurrently to the current one if the CSM bit is set to 1; otherwise it is scheduled sequentially to the completion of the current MBIST execution. In case of NIL pointer the CSM bit must be set Sequential (0) to define this is the end of the list. The self-testing procedure stops after the last BIST in the configuration chain is complete. See BIST scheduling for details.<br>**NOTE**<br>If the pointer is invalid and MBIST is scheduled to run concurrently than this invalid scenario is handled by watchdog timeout feature and corresponding watchdog timeout status bit will be updated in ERR_STAT[WDTOSW]. In this particular case ERR_STAT[INVPSW] will not be set.<br>0h to (01h - 1): pointer to LBIST<br>00000080h to (00000080h + 00Ch - 1): pointer to MBIST<br>3FFh: pointer to NIL. No next BIST execution.<br>others: invalid pointer => an error is set into the ERR_STAT[INVPSW]. |
| 20<br>BSEL | BSEL<br>BIST Select<br>0b - Selected BIST is not selected for execution.<br>1b - Selected BIST is selected for execution. |
| 19-0<br>— | Reserved |

## 51.10 Glossary

| | |
|---|---|
| **BIST** | Built-in self-test |
| **BIST Clock** | BIST controller clock corresponding to the specific BIST |
| **CORE_CLK** | Clock specified by the CFG register |
| **FSM** | Finite state machine |
| **IPS** | Internal peripheral system |
| **LBIST** | Logic BIST |
| **MBIST** | Memory BIST |
| **NLBIST** | Number of logic built-in self-test controller |
| **NMCUT** | Number of memory checked using memory built-in self-test controller |
| **Online self-test phase** | In this condition, system is alive and the SW can program STCU2 |
| **RF** | Recoverable faults |
| **UF** | Unrecoverable faults |

| | |
|---|---|
| **WDG FSM** | Watchdog finite state machine |

# Chapter 52
# Register Protection (REG_PROT)

## 52.1 Chip-specific REG_PROT information

### 52.1.1 REG_PROT configuration

The MCU safety relevant configuration registers are protected against unauthorized HW/S changes during read/write/clear access by implementing them with register protection module. See the REG_PROT details file attached to this document.

#### 52.1.1.1 CMU_BRIDGE register protection

The CMUs in the device reside on a common peripheral slot as CMU_BRIDGE in the system memory map. Figure 190 indicates the locations of the CMUs alongwith the memory map.

Based on the CMU location in system peripheral memory map, the below locations in the CMU 16KB space are reserved and any access on these locations results in a bus transfer error.

402B_C018 – 402B_C01F

402B_C030 – 402B_C03F

402B_C050 – 402B_C05F

402B_C078 – 402B_C07F

402B_C098 – 402B_C09F

402B_C0B8 – 402B_CFFF

Since the register protection is available in the device for CMU_BRIDGE, the 4KB offset locations (refers as the Area 2 in register protection) are also reserved and accesses over these regions also result in bus transfer error.

402B_D018 – 402B_D01F

402B_D030 – 402B_D03F

402B_D050 – 402B_D05F

402B_D078 – 402B_D07F

402B_D098 – 402B_D09F

402B_D0B8 – 402B_DFFF

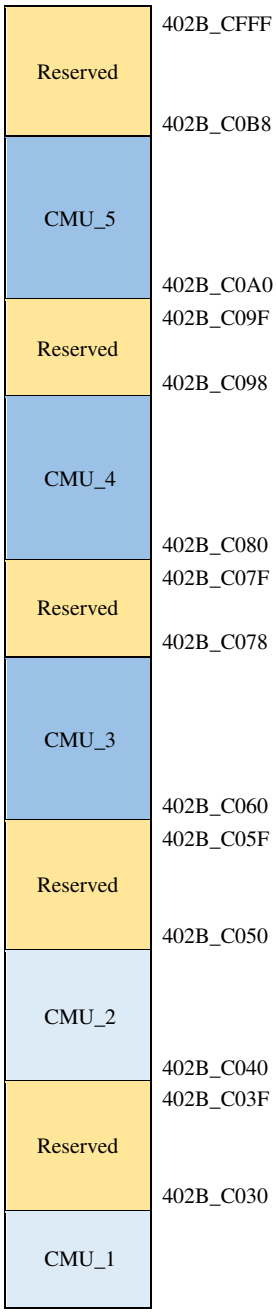



Figure 190. Location of CMUs

## 52.2 Introduction

REG_PROT offers a mechanism to protect defined memory-mapped address locations, in a module under protection, from being written.

REG_PROT is a protection module that is located between the module under protection and the peripheral interface. The address locations to be protected exist in the module under protection and the address locations that can be protected are module specific. Writes to these protected address locations are locked using a Soft Lock Bit Register (SLBR*n*). Any access to address locations in the module under protection is restricted if their corresponding Soft lock fields in a Soft Lock Bit Register (SLBR*n*) are 1.

The configured soft-lock fields can also be write-restricted by using GCR[HLB]. When GCR[HLB]=1, you cannot write to the soft-lock fields. The address locations in the module under protection can be restricted to Supervisor mode access using GCR[UAA]. This is shown in the following figure.

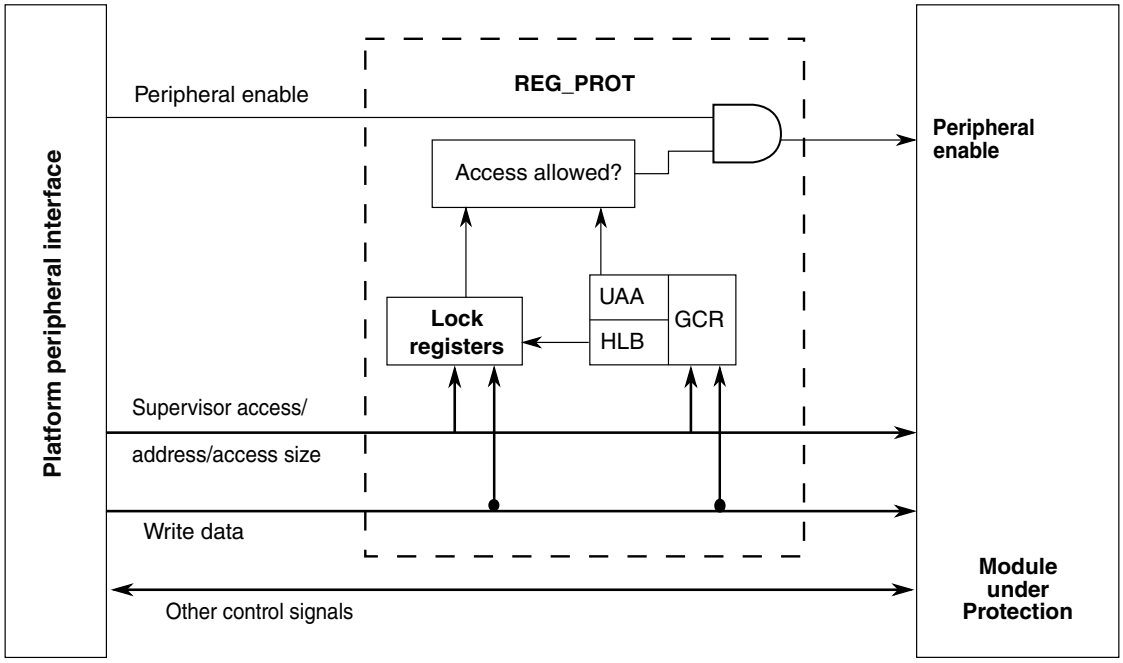



Figure 191. Block diagram

## 52.3 Features

REG_PROT includes these distinctive features:

- Write accesses for the module under protection can be restricted to the supervisor mode only.
- Registers of modules with their register slot size occupying 1 KB to 32 KB of memory-mapped address space, depending on the PROT_MEM parameter, can be dynamically locked. See Memory map and register definition.
- Multiple ways are present to set the lock bits.
- After the lock bits are configured, they can be protected from changes.

## 52.4 Modes of operation

The REG_PROT module is operable when the module under protection is operable.

## 52.5 External signal description

There are no external signals.

## 52.6 Functional description

The following sections describe the functional characteristics of the REG_PROT module.

### 52.6.1 General

This module provides a generic register (address) write-protection mechanism. The register protection size can be:

- 32 bits (address == multiples of 4)

- 16 bits (address == multiples of 2)
- 8 bits (address == multiples of 1)

The addresses that are protected and the register protection size depends on the chip and/or module.

For all addresses that are protected, there are SLBR*n*[SLB*m*] bits that specify whether the address is locked. When an address is locked, it can only be read but not written in any mode (supervisor/normal). If an address is unprotected, the corresponding SLBR*n*[SLB*m*] bit is always 0, regardless of what the software writes to that field.

**NOTE**

For more information on register protection specification, see the REG_PROT details file attached to this document.

## 52.6.2 Change lock settings

To change the setting whether an address is locked or unlocked, the corresponding SLBR*n*[SLB*m*] bit needs to be changed. This can be done using the following methods:

- Set the SLBR*n*[SLB*m*] bit(s) by writing to area #2
- Modify the SLBR*n*[SLB*m*] directly by writing to area #3

Both methods are explained in the following sections.

### 52.6.2.1 Change lock settings via area #2

You can lock a register after writing to it. Note that reading area #2 does not affect soft lock bits. To do so, you can use area # 2.

The following shows an example with PROT_MEM = 8 KB where SLBs are modified using area #2.

When writing 16-bit to address 0008h, MR9 and MR8 in the protected module are updated. The corresponding lock fields remain unchanged (see the left part of Figure 192).

When writing 16-bit to address 2008h, MR9 and MR8 in the protected module are updated. The corresponding lock fields SLBR2[SLB0:1] become 1, and the lock fields SLBR2[SLB2:3] remain unchanged (see the right part of Figure 192).

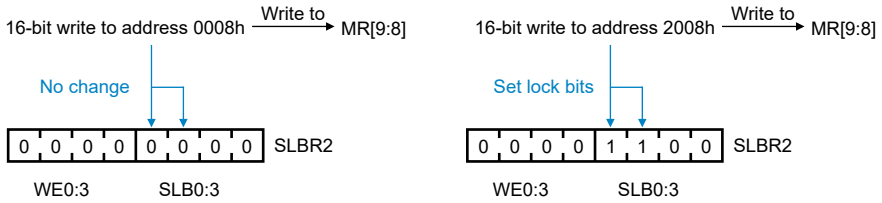



**Figure 192. Enable locking via area #2**

The following figure shows an example where some addresses are protected and some are not.

In Figure 193, addresses 000Ch and 000Dh are unprotected. Therefore, their corresponding lock fields SLBR3[SLB0:1] are always 0 (shown in bold). During a 32-bit write access to address 200Ch, the lock fields in SLBR3 change as follows:

- SLBR3[SLB2:3] become 1.
- SLBR3[SLB0:1] remain 0.

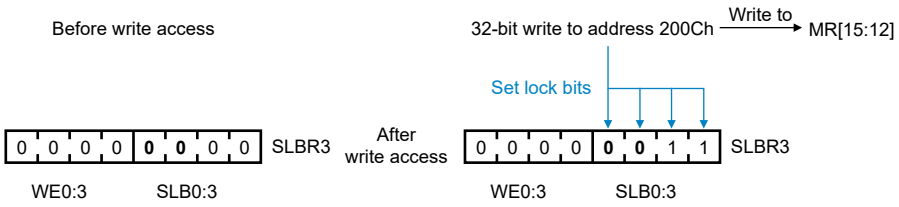



**Figure 193. Enable locking for protected and unprotected addresses**

### 52.6.2.2 Change lock settings via area #3

The lock bits are located in area #3 (see Table 245). You can modify them by writing to them. Each SLB*m* field in Soft Lock Bit Register (SLBR*n*) has a mask field, WE*m*, that protects SLB*m* from modification. This masking makes read-modify-write operations unnecessary.

Figure 194 shows two modification examples for registers with 8-bit protection. In part A, a write access to SLBR*n* specifies a mask value that allows modification of all SLBR*n*[SLB0:3] fields. Part B specifies a mask that allows modification only to SLBR*n*[SLB1:3].

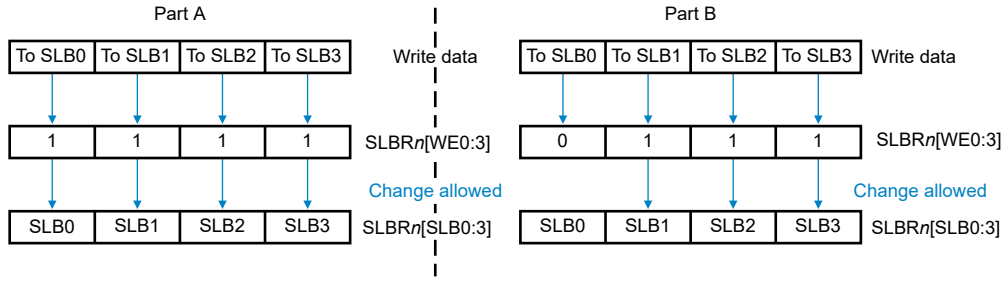



**Figure 194. Change lock settings directly via area #3**

Figure 195 shows examples for registers with 16-bit protection.

Part A of Figure 195 shows that for SLBR*n*:

- The data written to SLBR*n*[SLB0] is automatically written to SLBR*n*[SLB1].
- The data written to SLBR*n*[SLB2] is automatically written to SLBR*n*[SLB3].

This is because the address reflected by SLBR*n*[SLB0] and SLBR*n*[SLB2] are protected 16-bit wise. Note that in this case, the write enable SLBR*n*[WE0] and SLBR*n*[WE2] must be set while SLBR*n*[WE1] and SLBR*n*[WE3] don't matter.

Part B of Figure 195 shows that the data written to SLBR*n*[SLB0] is automatically written to SLBR*n*[SLB1]. This is done because the address reflected by SLBR*n*[SLB0] has 16-bit protection. In this case, SLBR*n*[WE0] must be 1 and SLBR*n*[WE1] does not matter. SLBR*n*[SLB2] and SLBR*n*[SLB3] remain unchanged because SLBR*n*[WE2] and SLBR*n*[WE3] are 0.

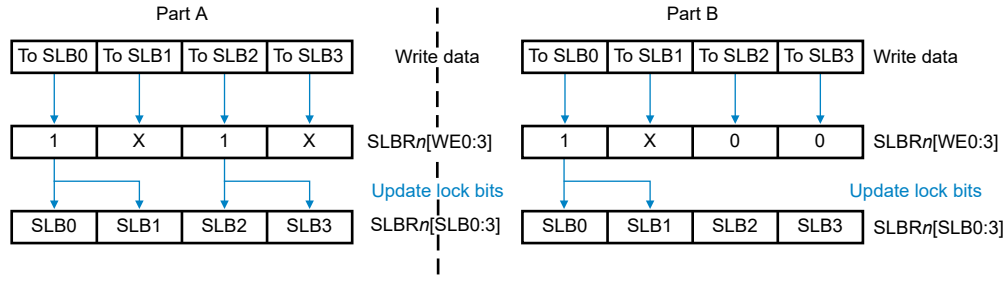



**Figure 195. Change lock settings for addresses with 16-bit protection**

Figure 196 shows a register with 32-bit protection. In SLBR*n*:

- When SLBR*n*[WE0]=1, the data written to SLBR*n*[SLB0] is automatically written to SLBR*n*[SLB1:3] as well.
- When SLBR*n*[WE0]=0, then SLBR*n*[SLB0:3] remain unchanged. Note that in this case, the write enable SLBR*n*[WE0] must be set while SLBR*n*[WE1:3] does not matter.

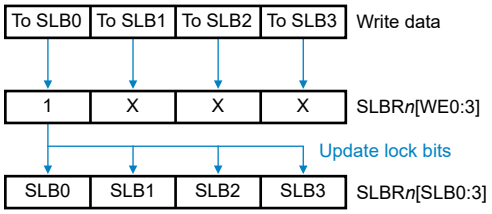



**Figure 196. Change lock settings for addresses with 32-bit protection**

The following figure shows a mixed protection size configuration.

In SLBR*n*:

- The data written to SLBR*n*[SLB0] is mirrored to SLBR*n*[SLB1] because the corresponding register has 16-bit protection.
- The data written to SLBR*n*[SLB2] is blocked because the corresponding register is unprotected.
- The data written to SLBR*n*[SLB3] is successfully written to SLBR*n*[SLB3] because the corresponding register has 8-bit protection.

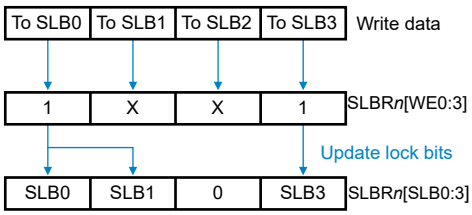



**Figure 197. Change lock settings for mixed protection**

### 52.6.2.3 Write protection for locking bits

Changing the locking bits through any of the procedures mentioned in Change lock settings via area #3 and Change lock settings via area #2 is only possible as long as the field GCR.HLB is cleared. After you write 1 to this field, the locking bits cannot be modified until there is a system reset.

## 52.6.3 Access errors

REG_PROT generates transfer errors under several circumstances, as described below. For the area definition, see Memory map and register definition.

- If accessing area #1 or area #2, REG_PROT sends any access error from the underlying module under protection.
- If User mode is not allowed, the attempted User mode writes to all areas cause a transfer error, and the writes are blocked.
- If accessing the reserved area, a transfer error is asserted.
- If accessing unimplemented 32-bit registers in areas #3 and #4, a transfer error is asserted.
- If writing to a register in areas #1 and #2, and an SLB field is 1 for any of the affected bytes, then:
  - A transfer error is asserted.
  - The write is blocked.
  - The complete write operation to non-protected bytes in this word is ignored.

- If writing to an SLBR in area #3 with GCR[HLB]=1, a transfer error is asserted.
- Any write operation in any access mode to area #2, when GCR[HLB]=1, is not allowed.

### 52.6.4 Sequence

You must configure the REG_PROT first. And then, you need to write 1 to the HLB field in REG_PROT. This ensures that registers under protection are not updated in the user access mode. You need to follow these steps:

1. Enable XRDC protection for the module to enable writes in the supervisor mode only. This step is mandatory, if protected registers of the module are expected to be configured multiple times.
2. Configure REG_PROT to enable write on registers of module under protection.
3. Configure the registers of module under protection.
4. Configure the REG_PROT for the required SLB fields.
5. Set the HLB field in REG_PROT. This step is mandatory, if protected registers of module are expected to be configured only once.

## 52.7 Memory map and register definition

This section provides a detailed description of the memory map of a module using REG_PROT.

The PROT_MEM size varies per chips. It can be:

- 1 KB
- 2 KB
- 4 KB
- 8 KB
- 16 KB
- 32 KB

The resulting memory space is divided into four areas. Memory locations within area #1 and area #2 are a part of the module under protection. Area #3 and area #4 are a part of REG_PROT.

The proper register slot size for each module is mentioned in the Memory Map file attached to this document.

Following is the memory map for a module that is protected by the REG_PROT module. Reserved registers in area #1 are handled as specified in the protected module's documentation. There is a single register in configuration space that corresponds to area #4 while the size mentioned in Area size column is to turn the whole area multiple of 4 KB. See the Offset column to know the exact location of the GCR register.

**Table 245. Area offset based on protection size**

| PROT_MEM | Area | Area size | Use | Offset |
|---|---|---|---|---|
| 1 KB | Area 1 | 1 KB | Module Register(MR0—MR1023) | 0000h—03FFh |
| | Area 2 | 1 KB | Module Register and Set Soft Lock Bit (LMR0—LMR1023) | 0400h—07FFh |
| | Area 3 | 256 B | Soft Lock Bit Register (SLBR0—255) | 0800h—08FFh |
| | Area 4 | 256 B | Global Configuration Register (GCR) | 0900h—09FFh |
| | | 1.5 KB | Reserved | 0A00h— 0FFFh |
| 2 KB | Area 1 | 2 KB | Module Register (MR0—MR2047) | 0000h—07FFh |
| | Area 2 | 2KB | Module Register and Set Soft Lock Bit (LMR0—LMR2047) | 0800h—0FFFh |

*Table continues on the next page...*

Table 245. Area offset based on protection size (continued)

| PROT_MEM | Area | Area size | Use | Offset |
|---|---|---|---|---|
| | Area 3 | 512 B | Soft Lock Bit Register (SLBR0—511) | 1000h—11FFh |
| | Area 4 | 256 B | Global Configuration Register (GCR) | 1200h—12FFh |
| | | 3.25 KB | Reserved | 1300h—1FFFh |
| 4 KB | Area 1 | 4 KB | Module Register (MR0—MR4095) | 0000h—0FFFh |
| | Area 2 | 4 KB | Module Register and Set Soft Lock Bit (LMR0—LMR4095) | 1000h—1FFFh |
| | Area 3 | 1 KB | Soft Lock Bit Register (SLBR0—1023) | 2000h—23FFh |
| | Area 4 | 256 B | Global Configuration Register (GCR) | 2400h—24FFh |
| | | 6.75 KB | Reserved | 2500h—3FFFh |
| 8 KB | Area 1 | 8 KB | Module Register (MR0—MR8191) | 0000h—1FFFh |
| | Area 2 | 8 KB | Module Register and Set Soft Lock Bit (LMR0—LMR8191) | 2000h—3FFFh |
| | Area 3 | 2 KB | Soft Lock Bit Register (SLBR0—2047) | 4000h—47FFh |
| | Area 4 | 256 B | Global Configuration Register (GCR) | 4800h—48FFh |
| | | 1.75 KB | Reserved | 4900h—4FFFh |
| 16 KB | Area 1 | 16 KB | Module Register (MR0—MR16383) | 0000h—3FFFh |
| | Area 2 | 16 KB | Module Register and Set Soft Lock Bit (LMR0—LMR16383) | 4000h—7FFFh |
| | Area 3 | 4 KB | Soft Lock Bit Register (SLBR0—4095) | 8000h—8FFFh |
| | Area 4 | 4 KB | Global Configuration Register (GCR) | 9000h—9FFFh |
| 32 KB | Area 1 | 32 KB | Module Register (MR0—MR32767) | 0000h—7FFFh |
| | Area 2 | 32 KB | Module Register and Set Soft Lock Bit (LMR0—LMR32767) | 8000h—FFFFh |
| | Area 3 | 8 KB | Soft Lock Bit Register (SLBR0—8191) | 10000h—11FFFh |
| | Area 4 | 4 KB | Global Configuration Register (GCR) | 12000h—12FFFh |

- Area #1 can take the memory space of 1 KB/2 KB/4 KB/8 KB/16 KB/32 KB, which holds the normal functional module registers and is transparent for all read/write operations.
- Area #2 can take the memory space of 1 KB/2 KB/4 KB/8 KB/16 KB/32 KB and is a mirror of area #1. For an example, if area #1 takes 8 KB of memory space, then read/write access to an offset 2000+X reads/writes the register at offset X. However, a write access to offset 2000+X additionally sets the optional soft lock bits for this offset X in the same cycle as the register at offset X is written. This provides for an automatic write and lock operation. Not all registers in area #1 need to have protection defined by the associated soft lock bits. For unprotected registers at offset Y, accesses to offset 2000+Y are identical to accesses at offset Y.
- Area #3 can take the memory space of 256 B/512 B/1 KB/2 KB/4 KB/8 KB and hold soft lock bits, one bit per byte in area #1. The four soft lock bits associated with one module register word are arranged at byte boundaries in the memory map. The Soft Lock Bit registers can be directly written using a bit mask.
- Area #4 can take the memory space of 1.75 KB/3.5 KB/7 KB/2 KB/4 KB/4 KB and holds the configuration bits of the protection mode. There is one configuration Hard lock bit per module that prevents all further modifications to the soft lock bits and can only be cleared by a system reset after it is set. When user access allowed bit is set, it allows user access to the protected module.

If you access a locked byte with a write transaction, a bus error is issued to the system and the write transaction is not executed. This is true even if not all accessed bytes are locked.

Accessing unimplemented 32—bit registers in area #3 results in a bus transfer error. In area #4, there will not be any transfer error for the address range mentioned in the table above because these address spaces of area #4 are mapped to the GCR register. Accesses in area #4, beyond the address range mentioned in the Module memory map table, are prohibited and might or might not result into a bus abort.

### 52.7.1 Module register (MR)

Each functional module has its own unique set of registers. This chapter refers to these registers generically as module registers (MR), which exist in the lower module memory space (memory area 1), and receive protection from the REG_PROT module.

### 52.7.2 Module Register and Set Soft Lock Bit (LMR)

This is memory area #2 that provides mirrored access to module registers with the side-effect of setting soft lock bits in case of a write access to a register that is defined as protectable by the locking mechanism. Each MR byte is protectable by one associated bit in SLBR*n* [SLB*m*], according to the mapping described in Soft Lock Bit Register (SLBR*n*).

## 52.8 REG_PROT register descriptions

This register information documents memory area #3 and area #4.

### 52.8.1 REG_PROT memory map

REG_PROT base address: base address of protected module instance

**Table 246. REG_PROT memory map**

| PROT_MEM | Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|---|
| 1kB | 0x800 – 0x8FF | Soft Lock Bit Register (SLBR0-255) | 8 | RW | 00h |
| | 0x900 – 0x9FF | Global Configuration Register (GCR) | 32 | RW | 0000_0000h |
| 2kB | 0x1000 – 0x11FF | Soft Lock Bit Register (SLBR0-511) | 8 | RW | 00h |
| | 0x1200 – 0x12FF | Global Configuration Register (GCR) | 32 | RW | 0000_0000h |
| 4kB | 0x2000 – 0x23FF | Soft Lock Bit Register (SLBR0-1023) | 8 | RW | 00h |
| | 0x2400 – 0x24FF | Global Configuration Register (GCR) | 32 | RW | 0000_0000h |
| 8kB | 0x4000 – 0x47FF | Soft Lock Bit Register (SLBR0-2047) | 8 | RW | 00h |
| | 0x4800 – 0x48FF | Global Configuration Register (GCR) | 32 | RW | 0000_0000h |
| 16kB | 0x8000 – 0x8FFF | Soft Lock Bit Register (SLBR0-4095) | 8 | RW | 00h |
| | 0x9000-0x9FFF | Global Configuration Register (GCR) | 32 | RW | 0000_0000h |
| 32kB | 0x10000-0x11FFF | Soft Lock Bit Register (SLBR0-8191) | 8 | RW | 00h |
| | 0x12000 – 0x12FFF | Global Configuration Register (GCR) | 32 | RW | 0000_0000h |

### 52.8.2 Soft Lock Bit Register (SLBR*n*)

#### Offset

See REG_PROT memory map for SLBR*n* offset ranges. For SLBR*n* addresses, see the REG_PROT details file attached to this document.

### Function

These registers hold the Soft Lock Bits (SLBs) for the protected registers in memory area #1, which is the normal register address space of the protected module. Multiple Soft Lock Bit Registers (SLBR*n*) can be implemented, depending on the number of protected module register bytes. Each SLBR*n* has four soft lock fields (SLB0-SLB3), each of which controls write access to a byte in memory area #1. Each soft lock field also has a corresponding write enable field in the same register that controls whether the soft lock field can be written. The following table shows the mapping between the soft lock fields to the bytes in memory area #1.

**Table 247. SLBs vs. protected address**

| SLB | Protected address |
|---|---|
| SLBR0[SLB0] | MR0 |
| SLBR0[SLB1] | MR1 |
| SLBR0[SLB2] | MR2 |
| SLBR0[SLB3] | MR3 |
| SLBR1[SLB0] | MR4 |
| SLBR1[SLB1] | MR5 |
| SLBR1[SLB2] | MR6 |
| SLBR1[SLB3] | MR7 |
| SLBR2[SLB0] | MR8 |
| ... | ... |

### Diagram

| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | SLB0 | SLB1 | SLB2 | SLB3 |
| W | WE0 | WE1 | WE2 | WE3 | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 7<br>WE0 | Write enable fields for SLB<br>WE0 enables writing to SLB0.<br>0b - SLB is not modified.<br>1b - Value is written to SLB. |
| 6<br>WE1 | Write enable fields for SLB<br>WE1 enables writing to SLB1.<br>0b - SLB is not modified.<br>1b - Value is written to SLB. |
| 5 | Write enable fields for SLB |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| WE2 | WE2 enables writing to SLB2.<br>0b - SLB is not modified.<br>1b - Value is written to SLB. |
| 4<br>WE3 | Write enable bits for SLB<br>WE3 enables writing to SLB3.<br>0b - SLB is not modified.<br>1b - Value is written to SLB. |
| 3<br>SLB0 | Soft lock fields for one MRn register<br>SLB0 can block accesses to MR[n *4 + 0].<br>0b - Associated MRn byte is unprotected and writable.<br>1b - Associated MRn byte is locked against write accesses. |
| 2<br>SLB1 | Soft lock bits for one MRn register<br>SLB1 can block accesses to MR[n *4 + 1].<br>0b - Associated MRn byte is unprotected and writable.<br>1b - Associated MRn byte is locked against write accesses. |
| 1<br>SLB2 | Soft lock fields for one MRn register<br>SLB2 can block accesses to MR[n *4 + 2].<br>0b - Associated MRn byte is unprotected and writable.<br>1b - Associated MRn byte is locked against write accesses. |
| 0<br>SLB3 | Soft lock fields for one MRn register<br>SLB3 can block accesses to MR[n *4 + 3].<br>0b - Associated MRn byte is unprotected and writable.<br>1b - Associated MRn byte is locked against write accesses. |

### 52.8.3 Global Configuration Register (GCR)

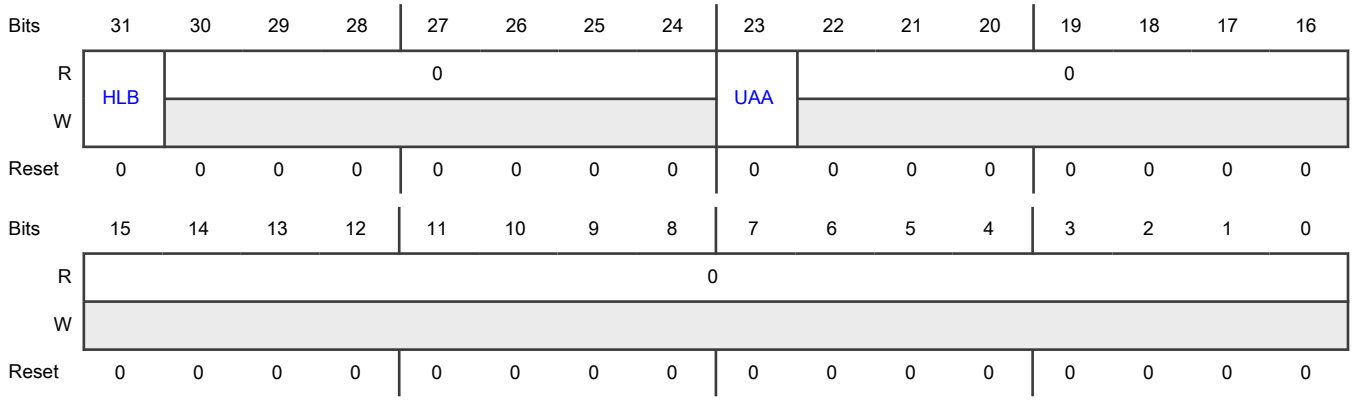
#### Offset

See REG_PROT memory map for GCR offset.

#### Function

This register controls module level configurations.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>HLB | Hard Lock Bit<br>This field cannot be cleared after it is set by the software. It can only be cleared by a system reset.<br>0b - All SLB fields are accessible and can be modified.<br>1b - All SLB fields are write protected and cannot be modified. |
| 30-24<br>— | Reserved |
| 23<br>UAA | User Access Allowed<br>Controls the user and supervisor mode access to registers of the module under protection.<br>0b - The registers in the module under protection can only be written in the supervisor mode. All the write accesses in non-supervisor mode are not executed and a transfer error is issued. This access restriction is in addition to any access restrictions imposed by the protected IP module.<br>1b - The registers in the module under protection can be accessed in the mode defined for the module registers without any additional restrictions. |
| 22-0<br>— | Reserved |

## 52.9 Glossary

**PROT_MEM** Integration parameter that specifies the size of the module register slot protected. The PROT_MEM size depends on the chip or module.

**Hard lock** A lock that restricts access to any register bit. You can unlock or clear this lock through the hardware reset.

**Soft lock** A lock that restricts access to any register bit. You can unlock or clear this lock through the software.

# Chapter 53
# Clock Monitoring Unit – Frequency Check (CMU_FC)

## 53.1 Chip-specific CMU_FC information

### 53.1.1 CMU_FC configuration

The device consists of a robust clock monitoring architecture targeted to monitor various clock nodes to ensure device clock robustness. There are 6 clock monitoring units present in the device out of which 4 are of type CMU_FC. See 'Clock monitoring' section in Clocking chapter for details on device clock monitoring architecture.

**NOTE**

CMU_5 is accessible by Cortex-M0+ core only.

**CMU_FC combined spec variation:**The combined spec variation for CMU (used for configuring CMU_FCx.HTCR[HFREF] and CMU_FCx.HTCR[LFREF]) is the absolute sum of frequency variations of the clock sources. Refer device datasheet for frequency variations for the corresponding clock sources.

## 53.2 Overview

CMU_FC ensures the clock integrity of selected clocks and allows continuous clock integrity verification.

CMU_FC checks if the frequency of a monitored clock (*monitored_clock*) is within a programmable frequency range specified by the user. If the frequency is outside the specified limits, it could lead to performance, protocol, and timing failures. CMU_FC requires a clock used as a base reference for the frequency check operation. This reference clock (*reference_clock*) must be within documented limits for correct CMU_FC operation.

### 53.2.1 Block diagram

CMU_FC counts clock cycles of the monitored clock during a user programmable time duration of *n* clock cycles of the reference clock. There is a comparison between the final count of the monitored clock and the user programmable upper and lower threshold count limits. The control logic determines whether the cycle count is within the programmed limits. If not it sets the Frequency Higher than High frequency reference threshold event status (FHH) field or the Frequency Lower than Low frequency reference threshold event status (FLL) field in the Status Register (SR). If you enable interrupts, an interrupt asserts when the status flag sets.

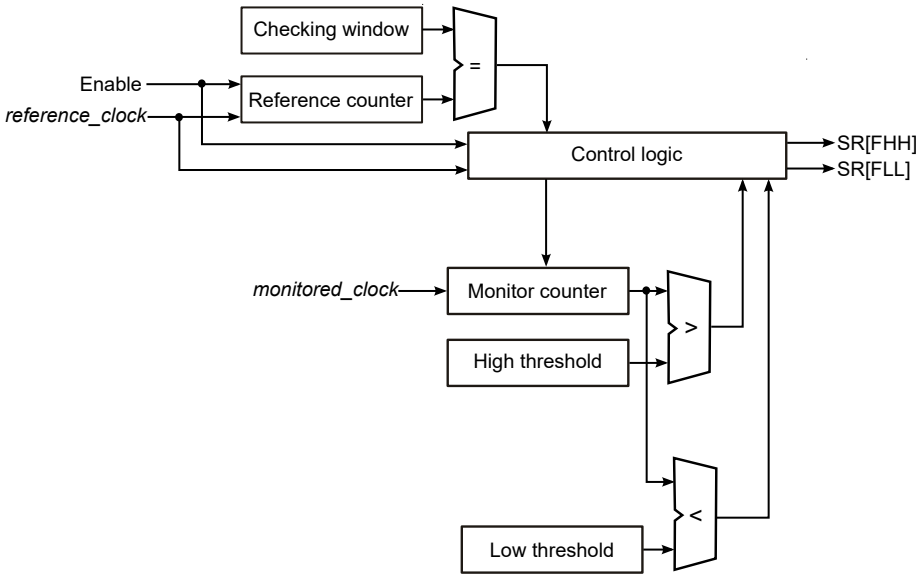



Figure 198. CMU_FC block diagram

CMU_FC cannot detect the following:

- Monitored clock duty cycle variations
- Instantaneous monitored clock frequency variations that do not cross upper or lower threshold limits

### 53.2.2 Features

The CMU_FC features are:

- Programmable duration of reference clock cycles
- Initiates an event if a monitored clock frequency is higher than high frequency reference (FHH)
- Initiates an event if a monitored clock frequency is lower than low frequency reference (FLL)
- FLL event (if enabled) is generated when monitored clock is very slow or stops functioning
- Masking of FHH event interrupt
- Masking of FLL event interrupt

## 53.3 Functional description

### 53.3.1 Operating modes

CMU_FC supports the following operating modes:

- Frequency checking

See the succeeding sections for detailed descriptions of the functions.

### 53.3.2 Frequency checking

Frequency checking starts when software writes GCR[FCE] = 1. The Run Status field SR[RS] shows 1 after frequency check operation starts.

CMU_FC performs continuous periodic clock checking of the monitored clock. Each checking window involves the following operations:

- Stage 1 – Reference clock counter runs for RCCR[REF_CNT] reference clock cycles. Monitored clock counter runs in parallel for the same time duration as reference clock counter.
- Stage 2 – After stage 1, the module compares the final monitored clock count against the programmed thresholds.

If the monitored clock count is greater than the high threshold value in HTCR[HFREF], an FHH event occurs.

If the monitored clock count is lower than the low threshold value in LTCR[LFREF], an FLL event occurs.

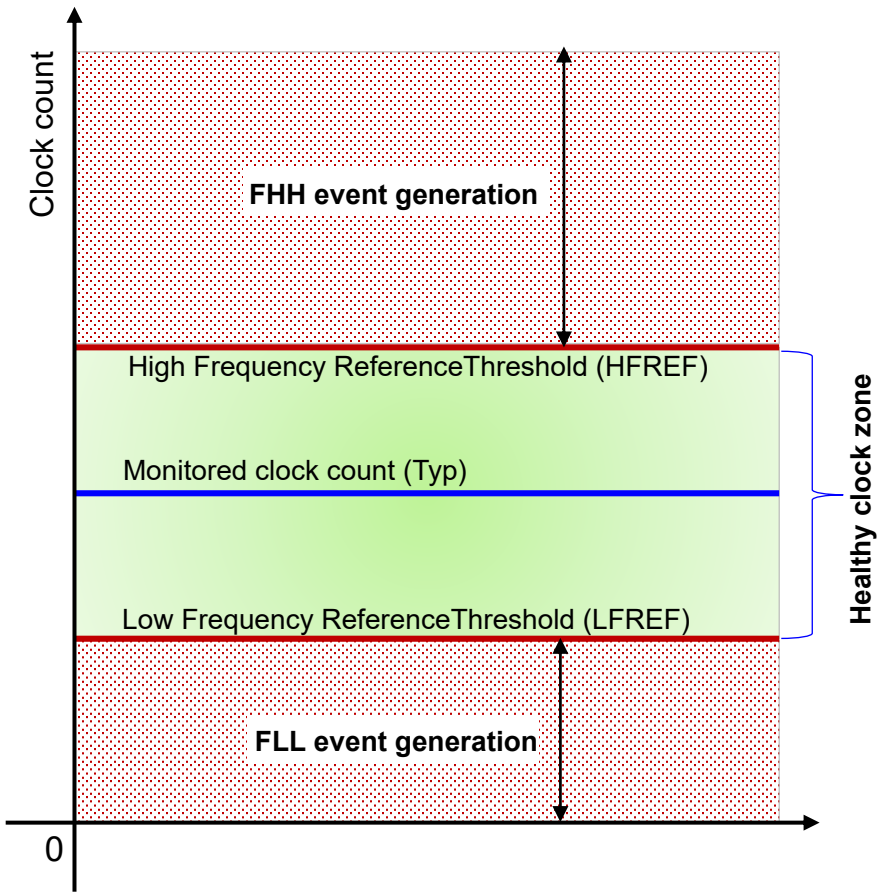



Figure 199. Frequency checking HFREF and LFREF thresholds

### 53.3.2.1 Monitored clock lost

If a monitored clock ceases operation before evaluation of that clock (Stage 2, see Frequency checking), CMU_FC waits for RCCR[REF_CNT] reference clock periods by extending Stage 2 before triggering an FLL event.

- If *monitored_clock* is not running when enabling CMU_FC:

| Worst case FLL response time = (2 × RCCR[REF_CNT] + 12) *reference_clock* cycles + 15 *bus_clock* cycles |
|---|

- If *monitored_clock* stops when the CMU_FC is running:

| Worst case FLL response time = (2 × RCCR[REF_CNT] + 2) *reference_clock* cycles + 3 *bus_clock* cycles |
|---|

### 53.3.3 Clocking

This module has the following clock sources.

**Table 248. Clock description**

| Clock | I/O | Description |
|---|---|---|
| *reference_clock* | I | Clock signal that the module uses as a reference to evaluate *monitored_clock* |
| *monitored_clock* | I | Clock signal on which frequency check is performed |
| *bus_clock* | I | Clock signal on which register read/write operation is performed |

**NOTE**

See the Clocking chapter for details of monitored and reference clocks used on this chip.

### 53.3.4 Reset

This module has the following reset sources.

- Global hard reset input
- Reset port synchronized to reference clock domain input
- Reset port synchronized to monitored clock domain input

## 53.4 External signals

This module has no external signals.

## 53.5 Programming guidelines

### 53.5.1 Programming RCCR[REF_CNT]

RCCR[REF_CNT] defines the duration of the checking operation in number of *reference_clock* cycles. The minimum RCCR[REF_CNT] value can be calculated using the following formula:

**Minimum RCCR [REF_CNT]** = CEILING value of MAX $\left(3 \text{ X } \left(\frac{f_{reference_clock}}{f_{bus_clock}}\right), 8 + 5 \text{ X } \left(\frac{f_{reference_clock}}{f_{monitored_clock}}\right)\right)$

...where:

- $f_{reference_clock}$ is the frequency of the reference clock
- $f_{bus_clock}$ is the frequency of the bus clock
- $f_{monitored_clock}$ is the frequency of the monitored clock

**Example to determine RCCR[REF_CNT]:**

- $f_{bus_clock}$ = 16 MHz
- $f_{reference_clock}$ = 8 MHz
- $f_{monitored_clock}$ = 48 MHz

| | |
|---|---|
| Formula #1: | $3 \times \left( \frac{f_{reference_clock}}{f_{bus_clock}} \right)$ |
| Inserting the values: | 3 × (8 / 16) = 1.5 |
| Formula #2: | $8 + 5 \times \left( \frac{f_{reference_clock}}{f_{monitored_clock}} \right)$ |
| Inserting the values: | 8 + 5 × (8 / 48) ≈ 8.83 |
| MAX (1.5, 8.83): | 8.83 |
| CEILING value of MAX: | 9 |
| Therefore, minimum RCCR[REF_CNT]: | 9 |

Programmed RCCR[REF_CNT] value must be greater than or equal to the calculated minimum value and should be decided after considering application requirements of frequency check completion response time and overall accuracy required from CMU_FC.

- Higher values of RCCR[REF_CNT] results in longer measurement window, leading to better accuracy in monitored clock check.
- Lower values of RCCR[REF_CNT] results in shorter measurement window, leading to faster FHH and FLL event response, but higher inaccuracy in reported result.

For measurement window calculation, see step 8 in Programming HFREF and LFREF

### 53.5.2 Programming HFREF and LFREF

You must consider the following when programming HFREF/LFREF:

- Tolerable monitored clock frequency variation
- Maximum reference clock frequency variation
- Inherent module inaccuracy

CMU_FC has an expected maximum deviation of ± 3 *monitored_clock* cycles ($CMU_FC_{VAR+}$ = 3, $CMU_FC_{VAR-}$ = –3).

The following procedure shows how to calculate optimum values for HTCR[HFREF] and LTCR[LFREF] (numbers shown are from example in table):

1. Determine the ideal *monitored_clock* frequency ($f_{monitored_clock}$ (ideal), 48 MHz).
2. Determine the specified variation of the *monitored_clock* (1.1%).
3. Calculate *monitored_clock* frequency variation: .

   $$f_{monitored_clock}\,(ideal) \times (1 \pm variation\,(\text{step 2}))$$

   - $f_{monitored_clock}$ (max) = 48 MHz × (1 + (1.1 ÷ 100)) = 48.53 MHz
   - $f_{monitored_clock}$ (min)= 48 MHz × (1 – (1.1 ÷ 100)) = 47.47 MHz
4. Determine the ideal *reference_clock* frequency ($f_{reference_clock}$ (ideal), 8 MHz).
5. Determine the specified variation of the *reference_clock* (3.3%).

6. Calculate ideal *reference_clock* frequency variation:

$$f_{reference_clock}\ (ideal) \times (1 \pm variation\ (\text{step 5}))$$

   - $f_{reference_clock}$ (max) = 8 MHz × (1 + (3.3 ÷ 100)) = 8.26 MHz
   - $f_{reference_clock}$ (min) = 8 MHz × (1 – (3.3 ÷ 100)) = 7.74 MHz
7. Select the value of RCCR[REF_CNT] needed (80). See Programming RCCR[REF_CNT].
8. Calculate ideal measurement window:

$$\frac{RCCR\,[REF_CNT]}{f_{reference_clock}\ (ideal)}$$

   - 80 ÷ 8 MHz = 10000 ns
9. Calculate ideal *monitored_clock* count:

$$\frac{RCCR[REF_CNT]}{f_{reference_clock}(ideal)} \times f_{monitored_clock}\ (ideal)$$

   - 80 ÷ 8 MHz × 48 MHz = 480.00 (round off this value, 480 in this case)
10. Calculate high threshold value (HTCR[HFREF]):

$$\text{Ceiling value of } \left(\frac{f_{monitored_clock}\ (max)}{f_{reference_clock}\ (min)} \times RCCR[REF_CNT]\right) + CMU_FC_{VAR+}$$

    - HTCR[HFREF] = Ceiling(48.53 MHz ÷ 7.74 MHz × 80 + 3) = 505
11. Calculate low threshold value (LTCR[LFREF]):

$$\text{Floor value of } \left(\frac{f_{monitored_clock}\ (min)}{f_{reference_clock}\ (max)} \times RCCR[REF_CNT]\right) - CMU_FC_{VAR-}$$

    - LTCR[LFREF] = Floor(47.47 MHz ÷ 8.26 MHz × 80 – 3) = 456

**Table 249. HTCR[HFREF] and LTCR[LFREF] calculation example**

| Property | Value | Unit |
|---|---|---|
| Bus clock frequency ($f_{bus_clock}$) | 16 | MHz |
| Monitored clock frequency ($f_{monitored_clock}$ (ideal)) | 48 | MHz |
| Specified variation of monitored clock | 1.1 | % |
| Monitored clock frequency (maximum after specified variation, $f_{monitored_clock}$ (max)) | 48.53 | MHz |

*Table continues on the next page...*

Table 249. HTCR[HFREF] and LTCR[LFREF] calculation example (continued)

| Property | Value | Unit |
|---|---|---|
| Monitored clock frequency (minimum after specified variation, $f_{monitored_clock}$ (min)) | 47.47 | MHz |
| | | |
| Reference clock frequency ($f_{reference_clock}$ (ideal)) | 8 | MHz |
| Specified variation of reference clock | 3.3 | % |
| Reference clock frequency (maximum after specified variation, $f_{reference_clock}$ (max)) | 8.26 | MHz |
| Reference clock frequency (minimum after specified variation, $f_{reference_clock}$ (min)) | 7.74 | MHz |
| | | |
| Minimum reference count (RCCR[REF_CNT]) | 9 | — |
| Programmed reference count (RCCR[REF_CNT])<br>(must be ≥ minimum RCCR[REF_CNT]) | 80 | — |
| Measurement window | 10000.00 | ns |
| | | |
| Calculated monitored clock count | 480.00 | — |
| Calculated monitored clock Count (rounded) | 480 | — |
| | | |
| Module positive variation ($CMU_FC_{VAR+}$) | 3 | — |
| Module negative variation ($CMU_FC_{VAR-}$) | -3 | — |
| | | |
| Programmed higher threshold (HTCR[HFREF]) | 505 | — |
| Programmed lower threshold (LTCR[LFREF]) | 456 | — |

## 53.5.3 Module programming sequence

### 53.5.3.1 Module programming sequence

The recommended programming sequence for CMU_FC frequency checking is:

1. In any order, complete the following:
   - Program RCCR[REF_CNT] to define the frequency measuring window duration in *reference_clock* cycles.
   - Program LTCR[LFREF] and HTCR[HFREF] to set the permissible frequency range of the *monitored_clock*.
   - Program IER to enable interrupt.
2. Write GCR[FCE] = 1 to start the frequency check operation.

   **NOTE**

   When GCR[FCE] = 1, you must not write to RCCR, HTCR, LTCR, or IER. Attempting to do so generates a bus transfer error.

3. To reconfigure RCCR, LTCR, HTCR , and IER or to stop frequency check operation, wait for SR[RS] = 1, then write GCR[FCE] = 0.

# 53.6 CMU_FC register descriptions

### 53.6.1 CMU_FC memory map

This section describes the address order of all the CMU_FC registers. Each description includes a standard register diagram and associated field descriptions.

CMU_0 base address: 402B_C000h

CMU_3 base address: 402B_C060h

CMU_4 base address: 402B_C080h

CMU_5 base address: 402B_C0A0h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Global Configuration Register (GCR) | 32 | RW | 0000_0000h |
| 4h | Reference Count Configuration Register (RCCR) | 32 | RW | 0000_0000h |
| 8h | High Threshold Configuration Register (HTCR) | 32 | RW | 00FF_FFFFh |
| Ch | Low Threshold Configuration Register (LTCR) | 32 | RW | 0000_0000h |
| 10h | Status Register (SR) | 32 | RW | 0000_0000h |
| 14h | Interrupt Enable Register (IER) | 32 | RW | 0000_0000h |

### 53.6.2 Global Configuration Register (GCR)

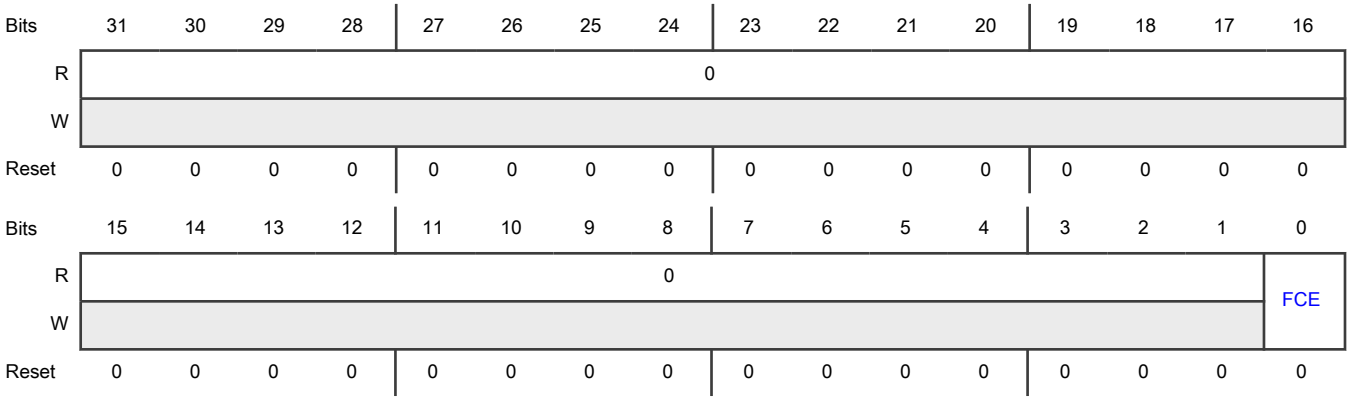
#### Offset

| Register | Offset |
|---|---|
| GCR | 0h |

#### Function

Controls module level configurations such as enabling frequency check.

#### Diagram



**Fields**

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>FCE | Frequency Check Enable<br>Starts or stops frequency checking. FCE is disabled by default. Software can enable FCE at any time. To stop the ongoing operation, write 0 to FCE only when SR[RS] = 1.<br>0b - Stops frequency checking<br>1b - Starts frequency checking |

## 53.6.3 Reference Count Configuration Register (RCCR)

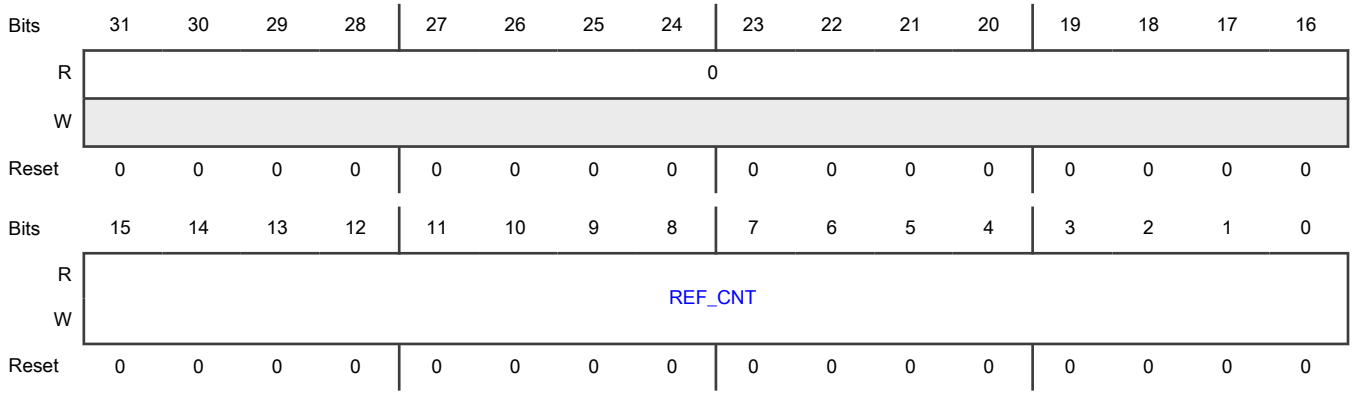
**Offset**

| Register | Offset |
|---|---|
| RCCR | 4h |

**Function**
Programs reference count duration of the frequency check window.

**NOTE**
Write to RCCR only when GCR[FCE] = 0. A bus transfer error results if software writes RCCR when GCR[FCE] = 1.

**Diagram**




**Fields**

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 15-0<br>REF_CNT | Reference clock count<br>Total number of counts of *reference_clock* for which frequency check runs. This field defines the duration of one frequency check window. See Programming RCCR[REF_CNT] for RCCR calculation. |

### 53.6.4 High Threshold Configuration Register (HTCR)

#### Offset

| Register | Offset |
|---|---|
| HTCR | 8h |

#### Function
Determines the high threshold limit of the monitored clock counter.

**NOTE**
Write HTCR only when GCR[FCE] = 0. A bus transfer error results if software writes HTCR when GCR[FCE] = 1.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | HFREF | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

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

#### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>HFREF | High frequency reference threshold<br>HFREF determines the high reference value for the monitored clock frequency.<br>See Programming HFREF and LFREF for HFREF calculation. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>Do not program HFREF to a value greater than 0x00FFFFFC. |

### 53.6.5 Low Threshold Configuration Register (LTCR)

#### Offset

| Register | Offset |
|---|---|
| LTCR | Ch |

#### Function

Determines the low threshold limit of the monitored clock counter.

**NOTE**

Write LTCR only when GCR[FCE] = 0. A bus transfer error results if software writes LTCR when GCR[FCE] = 1.

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>LFREF | Low Frequency Reference Threshold<br>LFREF determines the low reference value for the monitored clock frequency.<br>See Programming HFREF and LFREF for LFREF calculation.<br>**NOTE**<br>Do not program LFREF to a value less than 0x00000003. |

## 53.6.6 Status Register (SR)

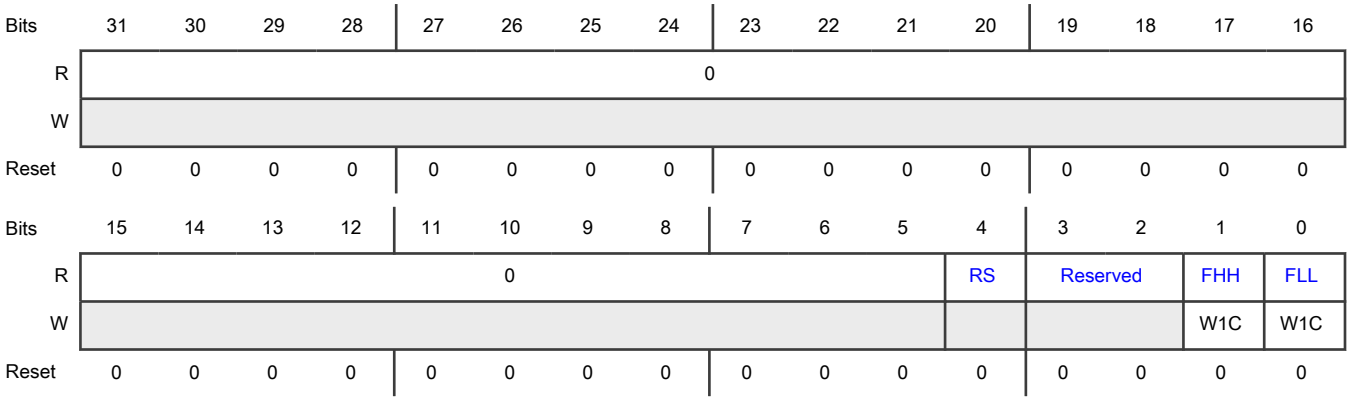
### Offset

| Register | Offset |
|---|---|
| SR | 10h |

### Function

Provides the internal status of the module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-5<br>— | Reserved |
| 4<br>RS | Run Status<br>Shows the running status of module internal operation. After enabling the frequency check operation (GCR[FCE] = 1), there is a fixed delay before this field shows running status. If the system wants to disable the frequency check operation, then the software should read this register and write GCR[FCE] = 0 only when this field is 1.<br>0b - Frequency check stopped<br>1b - Frequency check running |
| 3-2<br>— | Reserved |
| 1<br>FHH | Frequency higher than high frequency reference threshold event status<br>Hardware writes 1 to FHH when the monitored clock frequency becomes higher than the high threshold value in HTCR[HFREF]. FHH clears when software writes 1. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No FHH event<br>1b - FHH event occurred |
| 0<br>FLL | Frequency lower than low frequency reference threshold event status<br>Hardware writes 1 to FLL when the monitored clock frequency becomes lower than the low threshold value in LTCR[LFREF]. FLL clears when software writes 1.<br>0b - No FLL event<br>1b - FLL event occurred |

### 53.6.7 Interrupt Enable Register (IER)

#### Offset

| Register | Offset |
|---|---|
| IER | 14h |

#### Function

Enables CMU_FC interrupts.

**NOTE**

Write IER only when GCR[FCE] = 0. A bus transfer error results if software writes IER when GCR[FCE] = 1.

Enable only either asynchronous FHH event interrupt or synchronous FHH event interrupt at a time.

For example:

- If IER[FHHIE] is set to 1 then set IER[FHHAIE] to 0.
- If IER[FHHAIE] is set to 1 then set IER[FHHIE] to 0.

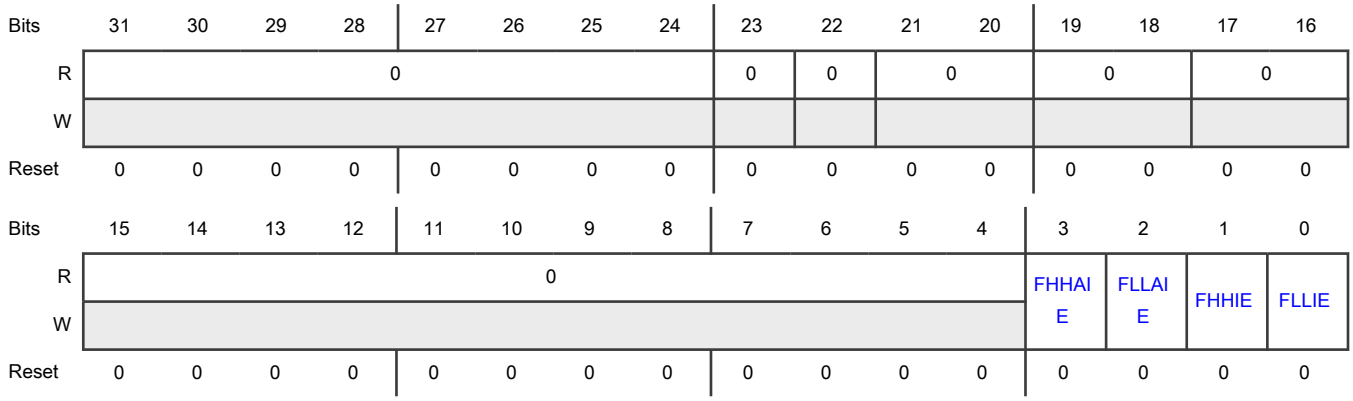
Enable only either asynchronous FLL event interrupt or synchronous FLL event interrupt at a time.

For example:

- If IER[FLLIE] is set to 1 then set IER[FLLAIE] to 0.
- If IER[FLLAIE] is set to 1 then set IER[FLLIE] to 0.

See Clocking chapter for interrupt usage information based on CMU_FC instances.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22<br>— | Reserved |
| 21-20<br>— | Reserved |
| 19-18<br>— | Reserved |
| 17-16<br>— | Reserved |
| 15-4<br>— | Reserved |
| 3<br>FHHAIE | Frequency Higher than High Frequency Reference Threshold Asynchronous Interrupt Enable<br>FHHAIE enables FHH asynchronous interrupt at the module boundary.<br>0b - Asynchronous FHH event interrupt disabled<br>1b - Asynchronous FHH event interrupt enabled |
| 2<br>FLLAIE | Frequency Lower than Low Frequency Reference Threshold Asynchronous Interrupt Enable<br>FLLAIE enables FLL asynchronous interrupt at the module boundary. |

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>0b - Asynchronous FLL event interrupt disabled<br>1b - Asynchronous FLL event interrupt enabled</td></tr>
<tr><td>1<br>FHHIE</td><td>Frequency Higher than High Frequency Reference Threshold Synchronous Interrupt Enable<br>FHHIE enables an FHH synchronous interrupt at the module boundary.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>CMU_0</td><td>IER</td><td>—</td></tr><tr><td>CMU_3</td><td>—</td><td>IER</td></tr><tr><td>CMU_4</td><td>—</td><td>IER</td></tr><tr><td>CMU_5</td><td>—</td><td>IER</td></tr></table>0b - Synchronous FHH event interrupt disabled<br>1b - Synchronous FHH event interrupt enabled</td></tr>
<tr><td>0<br>FLLIE</td><td>Frequency Lower than Low Frequency Reference Threshold Synchronous Interrupt Enable<br>FLLIE enables an FLL synchronous interrupt at the module boundary.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>CMU_0</td><td>IER</td><td>—</td></tr><tr><td>CMU_3</td><td>—</td><td>IER</td></tr><tr><td>CMU_4</td><td>—</td><td>IER</td></tr><tr><td>CMU_5</td><td>—</td><td>IER</td></tr></table>0b - Synchronous FLL event interrupt disabled<br>1b - Synchronous FLL event interrupt enabled</td></tr>
</table>

## 53.7 Glossary

**FHH** The state that the monitored clock frequency is higher than the high frequency reference. FHH is also the name of the register field that indicates this state.

**FHH event** The event that the module triggers when it detects FHH.

**FLL** The state that the monitored clock frequency is lower than the lower frequency reference. FLL is also the name of the register field that indicates this state.

**FLL event** The event that the module triggers when it detects FLL.

# Chapter 54
# Clock Monitoring Unit – Frequency Meter (CMU_FM)

## 54.1 Chip-specific CMU_FM information

### 54.1.1 CMU_FM configuration

The device consists of a robust clock monitoring architecture targeted to monitor various clock nodes to ensure device clock robustness. There are 6 clock monitoring units present in the device out of which 2 are of type CMU_FM. See 'Clock monitoring' section in Clocking chapter for details on device clock monitoring architecture.

**NOTE**

CMU metering interrupt is not generated if the clock which is being metered is lost. A timeout status flag, CMU.SR[FMTO] is updated in this case.

**NOTE**

Software must ensure that CMU1 interrupt is serviced within POR_WDG timeout. In absence of CMU1 interrupt servicing, the POR_WDG detects this as an FIRC failure and initiates a POR recovery.

## 54.2 Overview

CMU_FM measures the frequency of the metered clock (*metered_clock*) using another clock as reference. This reference clock (*reference_clock*) must be within documented limits for correct CMU_FM operation. CMU_FM operation is only guaranteed if a reference clock is within normal operating parameters.

### 54.2.1 Block diagram

The CMU_FM counts the clock cycles of a metered clock in the user programmable time duration *n* number of clock cycles of the reference clock. After completion of a metered clock operation, a register field sets, indicating the operation has completed. The count of the metered clock also updates in the Status Register (SR) after completion of a metered clock operation.

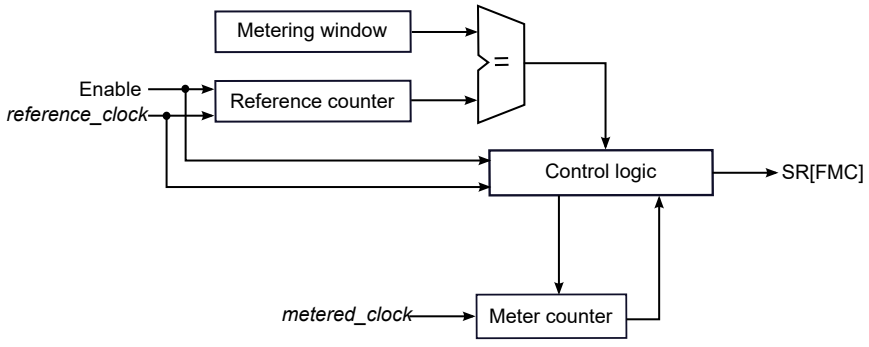



**Figure 200. Block diagram**

### 54.2.2 Features

Features of the CMU_FM are:

- Programmable duration of *reference_clock* cycles
- Maskable interrupt generation when frequency metering completes
- Timeout feature indicates loss of *metered_clock*

## 54.3 Functional description

The frequency meter operation initiates when software writes GCR[FME] = 1. When the operation starts, hardware writes SR[RS] = 1. Safety critical applications must poll SR[RS] to determine when metering operation starts.

Frequency metering window involves these operations:

1. Stage 1 — The reference clock counter runs for RCCR[REF_CNT] cycles of *reference_clock*. The metered clock counter runs in parallel for the same time duration as the *reference_clock* counter.
2. Stage 2 — At the end of stage 1, SR[FMC] = 1, SR[MET_CNT] updates with the *metered_clock* count, and CMU_FM writes GCR[FME] = 0.

### 54.3.1 Clocking

The following table lists the clock sources for this module.

**Table 250. Clock description**

| Clock | I/O | Description |
|---|---|---|
| *reference_clock* | I | Clock signal that the module uses as a reference to evaluate *metered_clock* |
| *metered_clock* | I | Clock signal on which frequency metering is performed |
| *bus_clock* | I | Clock signal on which register read/write operation is performed |

**NOTE**

See the Clocking chapter for details of metered and reference clocks used on this chip.

### 54.3.2 Reset

This module has the following reset sources.

- Global hard reset input
- Reset port synchronized to reference clock domain input
- Reset port synchronized to monitored clock domain input

## 54.4 Programming guidelines

### 54.4.1 Programming RCCR[REF_CNT]

RCCR[REF_CNT] defines the duration of the metering operation in number of *reference_clock* cycles. The minimum RCCR[REF_CNT] value can be calculated using the following formula:

$$\textbf{Minimum RCCR [REF_CNT]} = \text{CEILING value of MAX}\left(3 \text{ X } \left(\frac{f_{reference_clock}}{f_{bus_clock}}\right), 8 + 5 \text{ X } \left(\frac{f_{reference_clock}}{f_{monitored_clock}}\right)\right)$$

...where:

- $f_{reference_clock}$ is the frequency of the reference clock
- $f_{bus_clock}$ is the frequency of the bus clock
- $f_{metering_clock}$ is the frequency of the metered clock

**Example to determine RCCR[REF_CNT]:**

- $f_{bus_clock}$ = 16 MHz
- $f_{reference_clock}$ = 8 MHz
- $f_{metered_clock}$ = 48 MHz

Formula #1:

$$3 \text{ X } \left( \frac{f_{reference_clock}}{f_{bus_clock}} \right)$$

Inserting the values: 3 × (8 / 16) = 1.5

Formula #2:

$$8 + 5 \text{ X } \left( \frac{f_{reference_clock}}{f_{monitored_clock}} \right)$$

Inserting the values: 8 + 5 × (8 / 48) ≈ 8.83

MAX (1.5, 8.83): 8.83

CEILING value of MAX: 9

Therefore, minimum RCCR[REF_CNT]: 9

Programmed RCCR[REF_CNT] value must be greater than or equal to the calculated minimum value and should be decided after considering application requirements of frequency metering completion response time and overall accuracy required from CMU_FM.

- Higher values of RCCR[REF_CNT] results in longer metering window, leading to better accuracy in metered clock measurement.
- Lower values of RCCR[REF_CNT] results in shorter metering window, leading to faster FMC response, but higher inaccuracy in reported result.

### 54.4.2 Module programming sequence

The recommended programming sequence of CMU_FM is:

1. In any order, complete the following:
   - Program RCCR[REF_CNT] to define the frequency metering window duration in *reference_clock* cycles.
   - Program IER[FMCIE] to enable interrupt.
2. Write GCR[FME] = 1 to start frequency metering operation. Writes to RCCR and IER are disabled after GCR[FME] = 1. Attempting a write to any of them after GCR[FME] = 1 generates a bus transfer error.
3. To reconfigure RCCR and IER or to stop frequency metering operation, wait for SR[RS] = 1, then write GCR[FME] = 0.

Upon completion of metering operations:

- Hardware writes SR[FMC] = 1
- SR[MET_CNT] updates with metered count value
- Hardware writes GCR[FME] = 0

## 54.5 External signals

This module has no external signals.

## 54.6 CMU_FM register descriptions

### 54.6.1 CMU_FM memory map

This section describes the address order of all the CMU_FM registers. Each description includes a standard register diagram and associated field descriptions.

CMU_1 base address: 402B_C020h

CMU_2 base address: 402B_C040h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Global Configuration Register (GCR) | 32 | RW | 0000_0000h |
| 4h | Reference Count Configuration Register (RCCR) | 32 | RW | 0000_0000h |
| 8h | Status Register (SR) | 32 | RW | 0000_0000h |
| Ch | Interrupt Enable Register (IER) | 32 | RW | 0000_0000h |

### 54.6.2 Global Configuration Register (GCR)

#### Offset

| Register | Offset |
|---|---|
| GCR | 0h |

#### Function

Controls module level configurations such as enabling frequency metering.

#### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>FME | Frequency Meter Enable<br>Starts or stops frequency metering. FME is disabled by default. Software can enable FME at any time. To stop the ongoing operation, write 0 to FME only when SR[RS] = 1. FME automatically clears upon frequency metering completion or timeout event.<br>0b - Stops frequency metering<br>1b - Starts frequency metering |

## 54.6.3 Reference Count Configuration Register (RCCR)

### Offset

| Register | Offset |
|---|---|
| RCCR | 4h |

### Function

Programs reference count duration of the frequency meter window.

**NOTE**

Write to RCCR only when GCR[FME] = 0. A bus transfer error results if software writes RCCR when GCR[FME] = 1.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | REF_CNT | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-16 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 15-0<br>REF_CNT | Reference Clock Count<br>Total number of counts of *reference_clock* for which frequency metering runs. This field defines the duration of one frequency metering window. See Programming RCCR[REF_CNT] for RCCR calculation. |

### 54.6.4 Status Register (SR)

#### Offset

| Register | Offset |
|---|---|
| SR | 8h |

#### Function

Provides the internal status of the module and metered clock count.

#### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MET_CNT | | | | | | | | 0 | | | RS | Reserved | | FMTO | FMC |
| W | | | | | | | | | | | | | | | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-8<br>MET_CNT | Meter Clock Count<br>This field stores the count value of the metered clock cycles when frequency metering operation is complete, in other words, SR[FMC] = 1. CMU_FM has a maximum deviation of ± 3 *metered_clock* cycles over the ideal *metered_clock* cycles expected by the user. |
| 7-5<br>— | Reserved |
| 4 | Run Status |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| RS | Shows the running status of module internal operation. After enabling the frequency metering operation (GCR[FME] = 1), there is a fixed delay before this field shows running status. If the system wants to disable the frequency metering operation, then the software should read this register and write GCR[FME] = 0 only when this field is 1.<br>0b - Frequency meter stopped<br>1b - Frequency meter running |
| 3-2<br>— | Reserved |
| 1<br>FMTO | Frequency Meter Time Out<br>Sets if there is a loss of metered clock during the metering operation. After metering window, the module waits for a timeout duration of RCCR[REF_CNT] cycles before writing one to this field. GCR[FME] = 0 and SR[FMC] = 0 when FMTO = 1. Writing one clears this field. |
| 0<br>FMC | Frequency Meter Operation Complete<br>On completion of frequency metering operation, FMC = 1, SR[MET_CNT] field updates, and GCR[FME] clears. Writing one clears this field. |

## 54.6.5 Interrupt Enable Register (IER)

### Offset

| Register | Offset |
|---|---|
| IER | Ch |

### Function

Enables CMU_FM interrupts.

**NOTE**

Write IER only when GCR[FME] = 0. A bus transfer error results if software writes IER when GCR[FME] = 1.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>FMCIE | Frequency Meter Complete Interrupt Enable<br>This bit is used to enable/disable Frequency Meter complete interrupt at the module boundary.<br>0b - Frequency Meter complete interrupt is disabled<br>1b - Frequency Meter complete interrupt is enabled |

# Chapter 55
# Cyclic Redundancy Check (CRC)

## 55.1 Chip-specific CRC information

This chip supports one instance of CRC module.

## 55.2 Overview

CRC generates 16-bit or 32-bit CRC codes output for error detection. You can calculate these codes for up to 32 bits input data at a time.

CRC provides a programmable polynomial and other parameters to meet 16-bit or 32-bit CRC standards.

### 55.2.1 Block diagram

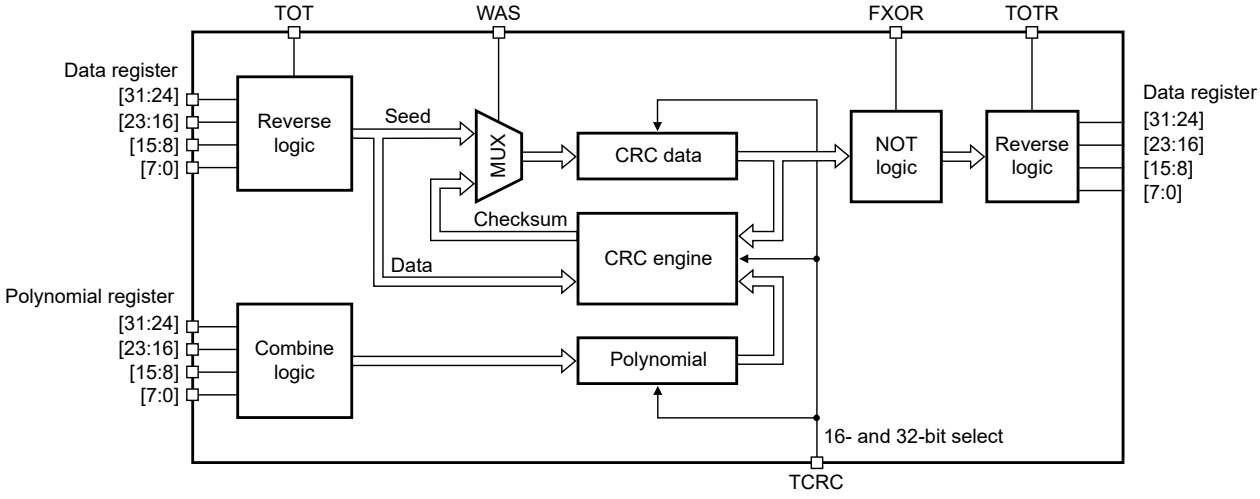



Figure 201. Block diagram

### 55.2.2 Features

CRC has the following features:

- Hardware CRC generator circuit using 16-bit or 32-bit programmable shift registers
- Programmable initial seed value and polynomial
- Transpose of input or output data (CRC result) in bitwise or bytewise (this option is required for certain CRC standards. You cannot perform a bytewise transpose operation when accessing DATA via 8-bit access.)
- Invert final CRC result
- 32-bit CPU register programming interface

## 55.3 Functional description

### 55.3.1 Modes of operation

The following sections describe various modes of operation that affect the functionality of CRC: Run mode and Low power mode.

#### 55.3.1.1 Run mode

Run mode is the basic mode of operation.

#### 55.3.1.2 Low-power modes

When the chip enters the low-power mode, the CRC module clock (ipg_clk and ipg_clk_s) is disabled and the in-progress CRC calculation stops. The calculation resumes after the CRC module clock is enabled or the chip exits low power mode via system reset.

**NOTE**

See the chip-specific power management chapter to know the specific low-power modes supported on your device.

### 55.3.2 CRC calculations

In 16-bit and 32-bit CRC modes, you can program data values as 8 bits, 16 bits, or 32 bits at a time, provided all bytes are contiguous. Noncontiguous bytes lead to an incorrect CRC calculation.

#### 55.3.2.1 Calculating a 16-bit CRC

Perform these steps to calculate a 16-bit CRC:

1. Write 0 to CTRL[TCRC] to enable 16-bit CRC mode.
2. Program the transpose and complement options in Control (CTRL) as required for the CRC calculation.
3. Write a 16-bit polynomial to GPOLY[LOW].

   GPOLY[HIGH] is not usable in 16-bit CRC mode.
4. Write 1 to CTRL[WAS] to program the seed value.
5. Write a 16-bit seed to DATA[LU] and DATA[LL].

   DATA[HU] and DATA[HL] are not used.
6. Write 0 to CTRL[WAS] to start writing data values.
7. Write data values into DATA[LU], and DATA[LL]

   CRC is calculated on every data value write and the intermediate CRC result is stored back into DATA[LU] and DATA[LL]
8. After writing all the data values, read the final CRC result from DATA[LU] and DATA[LL].

CRC is calculated bytewise and two clocks are required to complete one CRC calculation.

The transpose and complement operations are performed on-the-fly when reading or writing values. See Transpose feature and Result complement for details.

#### 55.3.2.2 Calculating a 32-bit CRC

Perform these steps to calculate a 32-bit CRC:

1. Write 1 to CTRL[TCRC] to enable 32-bit CRC mode.
2. Program the transpose and complement options in Control (CTRL) as required for CRC calculation. See Transpose feature and Result complement for details.
3. Write a 32-bit polynomial to GPOLY[HIGH] and GPOLY[LOW] .
4. Write 1 to CTRL[WAS] to program the seed value.
5. Write a 32-bit seed to DATA[HU], DATA[HL], DATA[LU], and DATA[LL].
6. Write 0 to CTRL[WAS] to start writing data values.
7. Write data values into DATA[HU], DATA[HL], DATA[LU], and DATA[LL]

CRC is computed on every data value write and the intermediate CRC result is stored back into DATA .

8. After writing all the values, read the final CRC result from DATA[HU], DATA[HL], DATA[LU], and DATA[LL].

   CRC is calculated bytewise and two clocks are required to complete one CRC calculation.

The transpose and complement operations are performed on-the-fly when reading or writing values. See Transpose feature and Result complement for details.

### 55.3.3 Transpose feature

Transpose is not enabled by default. However, CRC requires input data and/or final checksum to be transposed. You have an option to configure each transpose operation separately to meet CRC standards. The data is transposed on-the-fly while being read or written.

Some protocols use the little-endian format for data stream to calculateCRC. In this case, transpose flips bits.

#### 55.3.3.1 Types of transpose

CRC provides several types of transpose to flip bits and/or bytes for both writing input data and reading result separately using the CTRL[TOT] and CTRL[TOTR] according to the CRC calculation being used.

The following types of transpose are available for writing to and reading from DATA.

1. CTRL[TOT] or CTRL[TOTR] is 0.

   No transposition occurs.

2. CTRL[TOT] or CTRL[TOTR] is 1.

   Bits in a byte are transposed when bytes are not transposed.

   reg[31:0] becomes {reg[24:31], reg[16:23], reg[8:15], reg[0:7]}.

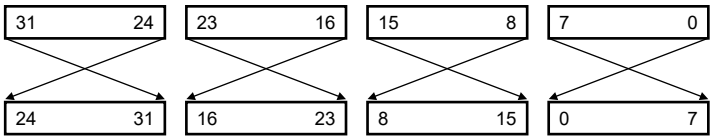



**Figure 202. Transpose type 1b**

3. CTRL[TOT] or CTRL[TOTR] is 10b.

   Both bits in bytes and bytes are transposed.

   reg[31:0] becomes {reg[0:7], reg[8:15], reg[16:23], reg[24:31]}.

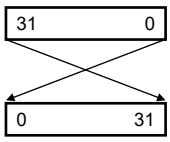



**Figure 203. Transpose type 10b**

4. CTRL[TOT] or CTRL[TOTR] is 11b.

   Bytes are transposed but bits are not transposed.

   reg[31:0] becomes {reg[7:0], reg[15:8], reg[23:16], reg[31:24]}.

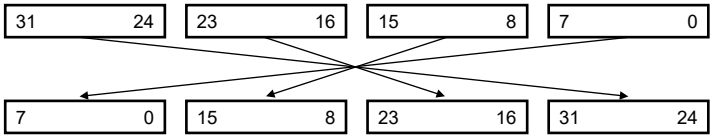



Figure 204. Transpose type 11b

**NOTE**

For 8-bit and 16-bit write accesses to Data (DATA), the data is transposed with 0s on the unused byte or bytes (taking 32 bits as a whole), but CRC is calculated on the valid byte(s) only. When reading the Data (DATA) for a 16-bit CRC result and using transpose options 10 and 11, the resulting value after transposition resides in DATA[HU] and DATA[HL]. You must account for this situation when reading the 16-bit CRC result, so reading 32 bits is preferred.

### 55.3.4 Result complement

When CTRL[FXOR] = 1, the checksum is complemented. The CRC result complement function outputs the complement of the checksum value stored in Data (DATA) every time Data (DATA) is read. When CTRL[FXOR] = 0,, reading Data (DATA) accesses the raw checksum value.

### 55.3.5 Clocking

Table 251. CRC clocks

| Type of clock | Description |
|---|---|
| Bus clock (ipg_clk/ipg_clk_s) | ipg_clk_s controls the access to the CRC registers. ipg_clk and ipg_clk_s function the CRC module. |

**NOTE**

See the chip-specific clocking chapter, to know the specific CRC clocks supported on your device.

### 55.3.6 Interrupts

This module has no interrupts.

## 55.4 External signals

There is no CRC signal that connects off chip.

## 55.5 Initialization

To enable CRC calculation, you must program:

- CTRL[WAS] .
- Polynomial (GPOLY).
- Parameters for transposition and CRC result inversion in the applicable registers.

Writing 1 to CTRL[WAS] enables you to program the seed value into CRC Data registers.

After writing all the data, you must wait for at least two clock cycles to read the data from CRC Data (DATA) register.

After a CRC calculation completes, you can reinitialize the module for a new CRC computation by again writing 1 to CTRL[WAS] and programming a new, or previously used, seed value. You must set all other parameters before programming the seed value and subsequent data values.

## 55.6 Use cases

The following tables use the little-endian format.

### 55.6.1 CTRL programming

The following table shows Control (CTRL) programming for 16-bit CRC.

**Table 252. CTRL programming for 16-bit CRC**

| Algorithm | Polynomial | Seed | Ref in | Ref out | XOR out | CTRL[TOT] | CTRL[TOTR] | CTRL[FXOR] |
|---|---|---|---|---|---|---|---|---|
| CRC-16_CCITT_FALSE | 1021h | FFFFh | 0 | 0 | 0000h | 0h | 0h | 0h |
| CRC-16_ARC | 8005h | 0000h | 1 | 1 | 0000h | 1h | 2h | 0h |
| CRC-16_AUG_CCITT | 1021h | 1D0Fh | 0 | 0 | 0000h | 0h | 0h | 0h |
| CRC-16_BUYPASS | 8005h | 0000h | 0 | 0 | 0000h | 0h | 0h | 0h |
| CRC-16_CCITT_ZERO | 1021h | 0000h | 0 | 0 | 0000h | 0h | 0h | 0h |
| CRC-16_CDMA2000 | C867h | FFFFh | 0 | 0 | 0000h | 0h | 0h | 0h |
| CRC-16_DDS_110 | 8005h | 800Dh | 0 | 0 | 0000h | 0h | 0h | 0h |
| CRC-16_DECT_X | 589h | 0000h | 0 | 0 | 0000h | 0h | 0h | 0h |
| CRC-16_DNP | 3D65h | 0000h | 1 | 1 | FFFFh | 1h | 2h | 1h |
| CRC-16_EN_13757 | 3D65h | 0000h | 0 | 0 | FFFFh | 0h | 0h | 1h |
| CRC-16_GENIBUS | 1021h | FFFFh | 0 | 0 | FFFFh | 0h | 0h | 1h |
| CRC-16_MAXIM | 8005h | 0000h | 1 | 1 | FFFFh | 1h | 2h | 1h |
| CRC-16_MCRF4XX | 1021h | FFFFh | 1 | 1 | 0000h | 1h | 2h | 0h |
| CRC-16_RIELLO | 1021h | B2AAh | 1 | 1 | 0000h | 1h | 2h | 0h |
| CRC-16_T10_DIF | 8BB7h | 0000h | 0 | 0 | 0000h | 0h | 0h | 0h |
| CRC-16_TELEDISK | A097h | 0000h | 0 | 0 | 0000h | 0h | 0h | 0h |
| CRC-16_TMS37157 | 1021h | 89ECh | 1 | 1 | 0000h | 1h | 2h | 0h |
| CRC-16_USB | 8005h | FFFFh | 1 | 1 | FFFFh | 1h | 2h | 1h |
| CRC-16_A | 1021h | C6C6h | 1 | 1 | 0000h | 1h | 2h | 0h |
| CRC-16_KERMIT | 1021h | 0000h | 1 | 1 | 0000h | 1h | 2h | 0h |
| CRC-16_MODBUS | 8005h | FFFFh | 1 | 1 | 0000h | 1h | 2h | 0h |
| CRC-16_X_25 | 1021h | FFFFh | 1 | 1 | FFFFh | 1h | 2h | 1h |
| CRC-16_XMODEM | 1021h | 0000h | 0 | 0 | 0000h | 0h | 0h | 0h |

The following table shows Control (CTRL) programming for 32-bit CRC.

Table 253. CTRL programming for 32-bit CRC

| Algorithm | Polynomial | Seed | Ref in | Ref out | XOR out | CTRL[TOT] | CTRL[TOTR] | CTRL[FXOR] |
|---|---|---|---|---|---|---|---|---|
| CRC-32 | 04C11DB7h | FFFFFFFFh | 1 | 1 | FFFF_FFFFh | 1h | 2h | 1h |
| CRC-32_BZIP2 | 04C11DB7h | FFFFFFFFh | 0 | 0 | FFFF_FFFFh | 0h | 0h | 1h |
| CRC-32C | 1EDC6F41h | FFFFFFFFh | 1 | 1 | FFFF_FFFFh | 1h | 2h | 1h |
| CRC-32D | A833982Bh | FFFFFFFFh | 1 | 1 | FFFF_FFFFh | 1h | 2h | 1h |
| CRC-32_MPEG-2 | 04C11DB7h | FFFFFFFFh | 0 | 0 | 0000_0000h | 0h | 0h | 0h |
| CRC-32_POSIX | 04C11DB7h | 00000000h | 0 | 0 | FFFF_FFFFh | 0h | 0h | 1h |
| CRC-32Q | 814141ABh | 00000000h | 0 | 0 | 0000_0000h | 0h | 0h | 0h |
| CRC-32_JAMCRC | 04C11DB7h | FFFFFFFFh | 1 | 1 | 0000_0000h | 1h | 2h | 0h |
| CRC-32_XFER | 000000AFh | 00000000h | 0 | 0 | 0000_0000h | 0h | 0h | 0h |

### 55.6.2 Expected read data fields

The following table shows the expected read data fields for 16-bit CRC.

Table 254. Expected read data fields for 16-bit CRC

| Algorithm | Data (DATA) |
|---|---|
| CRC16_CCITT_FALSE | [31:16] = Unknown[15:0] = Valid data |
| CRC16_ARC | [31:16] = Valid data [15:0] = Unknown |
| CRC16_AUG_CCITT | [31:16] = Unknown [15:0] = Valid data |
| CRC16_BUYPASS | [31:16] = Unknown [15:0] = Valid data |
| CRC16_CCITT_ZERO | [31:16] = Unknown [15:0] = Valid data |
| CRC16_CDMA2000 | [31:16] = Unknown [15:0] = Valid data |
| CRC16_DDS_110 | [31:16] = Unknown [15:0] = Valid data |
| CRC16_DECT_X | [31:16] = Unknown [15:0] = Valid data |
| CRC16_DNP | [31:16] = Valid data [15:0] = Unknown |
| CRC-16_EN_13757 | [31:16] = Unknown [15:0] = Valid data |
| CRC-16_GENIBUS | [31:16] = Unknown [15:0] = Valid data |
| CRC-16_MAXIM | [31:16] = Valid data [15:0] = Unknown |
| CRC-16_MCRF4XX | [31:16] = Valid data [15:0] = Unknown |
| CRC-16_RIELLO | [31:16] = Valid data [15:0] = Unknown |
| CRC-16_T10_DIF | [31:16] = Unknown [15:0] = Valid data |
| CRC-16_TELEDISK | [31:16] = Unknown [15:0] = Valid data |
| CRC-16_TMS37157 | [31:16] = Valid data [15:0] = Unknown |
| CRC-16_USB | [31:16] = Valid data [15:0] = Unknown |

*Table continues on the next page...*

Table 254. Expected read data fields for 16-bit CRC (continued)

| Algorithm | Data (DATA) |
|---|---|
| CRC-16_A | [31:16] = Valid data [15:0] = Unknown |
| CRC-16_KERMIT | [31:16] = Valid data [15:0] = Unknown |
| CRC-16_MODBUS | [31:16] = Valid data [15:0] = Unknown |
| CRC-16_X_25 | [31:16] = Valid data [15:0] = Unknown |
| CRC-16_XMODEM | [31:16] = Unknown [15:0] = Valid data |

The following table shows the expected read data fields for 32-bit CRC.

Table 255. Expected read data fields for 32-bit CRC

| Algorithm | Data (DATA) |
|---|---|
| CRC-32 | [31:0] = Valid data |
| CRC-32_BZIP2 | [31:0] = Valid data |
| CRC-32C | [31:0] = Valid data |
| CRC-32D | [31:0] = Valid data |
| CRC-32_MPEG-2 | [31:0] = Valid data |
| CRC-32_POSIX | [31:0] = Valid data |
| CRC-32Q | [31:0] = Valid data |
| CRC-32_JAMCRC | [31:0] = Valid data |
| CRC-32_XFER | [31:0] = Valid data |

## 55.7 Memory map and register descriptions

**NOTE**

You must reconfigure CRC engine in case an IPS transfer error occurs (ips_xfr_err).

The CRC module generates a transfer error in the following cases:

- Write accesses to the register addresses that are not mapped to the peripherals but included in the address spaces of the peripherals.

### 55.7.1 CRC register descriptions

#### 55.7.1.1 CRC memory map

CRC base address: 4038_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Data (DATA) | 32 | RW | FFFF_FFFFh |
| 4h | Polynomial (GPOLY) | 32 | RW | 0000_1021h |
| 8h | Control (CTRL) | 32 | RW | 0000_0000h |

### 55.7.1.2 Data (DATA)

#### Offset

| Register | Offset |
|---|---|
| DATA | 0h |

#### Function

Configures the value of seed, data, and checksum. When CTRL[WAS] = 1, any write to this register is regarded as the seed value. When CTRL[WAS] becomes 0, any write to this register is regarded as data for general CRC calculation.

In 16-bit CRC mode, DATA[HU] and DATA[HL] are not used for programming the seed value, and reads of these fields return an indeterminate value. In 32-bit CRC mode, all fields are used for programming the seed value.

When programming data values, you can program to write 8 bits, 16 bits, or 32 bits in big endian order, provided all bytes are contiguous, with the MSB of data value written first.

After writing all data values, you can read the CRC result from DATA register. In 16-bit CRC mode, the CRC result is available in DATA[LU] and DATA[LL]. In 32-bit CRC mode, all fields contain the result. After writing all data, you must wait for at least two clock cycles to read the data from CRC data (DATA) register.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | HU | | | | | | | | HL | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LU | | | | | | | | LL | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

#### Fields

| Field | Function |
|---|---|
| 31-24<br>HU | Upper Part of High Byte<br>Generates CRC checksum in both 16-bit and 32-bit CRC modes if CTRL[WAS] = 0.<br>• In 16-bit CRC mode (CTRL[TCRC] = 0), this field is not used for programming a seed value.<br>• In 32-bit CRC mode (CTRL[TCRC] = 1), the values written to this field are part of the seed value when CTRL[WAS] = 1. |
| 23-16<br>HL | Lower Part of High Byte<br>Generates CRC checksum in both 16-bit and 32-bit CRC modes if CTRL[WAS] = 0.<br>• In 16-bit CRC mode (CTRL[TCRC] = 0), this field is not used for programming a seed value. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • In 32-bit CRC mode (CTRL[TCRC] = 1), the values written to this field are part of the seed value when CTRL[WAS] = 1. |
| 15-8<br>LU | Upper Part of Low Byte<br>Generates CRC checksum when CTRL[WAS] = 0.<br>When CTRL[WAS] = 1,the values written to this field are part of the seed value. |
| 7-0<br>LL | Lower Part of Low Byte<br>Generates CRC checksum when CTRL[WAS] = 0.<br>When CTRL[WAS] = 1, the values written to this field are part of the seed value. |

### 55.7.1.3 Polynomial (GPOLY)

#### Offset

| Register | Offset |
|---|---|
| GPOLY | 4h |

#### Function

Configures polynomial value for CRC calculation.

- Sets the upper 16 bits of polynomial that are used only in 32-bit CRC mode. Writes to this field are ignored in 16-bit CRC mode.
- Sets the lower 16 bits of polynomial that are used in both 16-bit and 32-bit CRC modes.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | HIGH | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| R<br>W | LOW | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |

### Fields

| Field | Function |
|---|---|
| 31-16<br>HIGH | High Half-Word<br>Writable and readable in 32-bit CRC mode (CTRL[TCRC] = 1). You cannot write to this field in 16-bit CRC mode (CTRL[TCRC] = 0). |
| 15-0<br>LOW | Low Half-Word<br>Writable and readable in both 16-bit and 32-bit CRC modes. |

## 55.7.1.4 Control (CTRL)

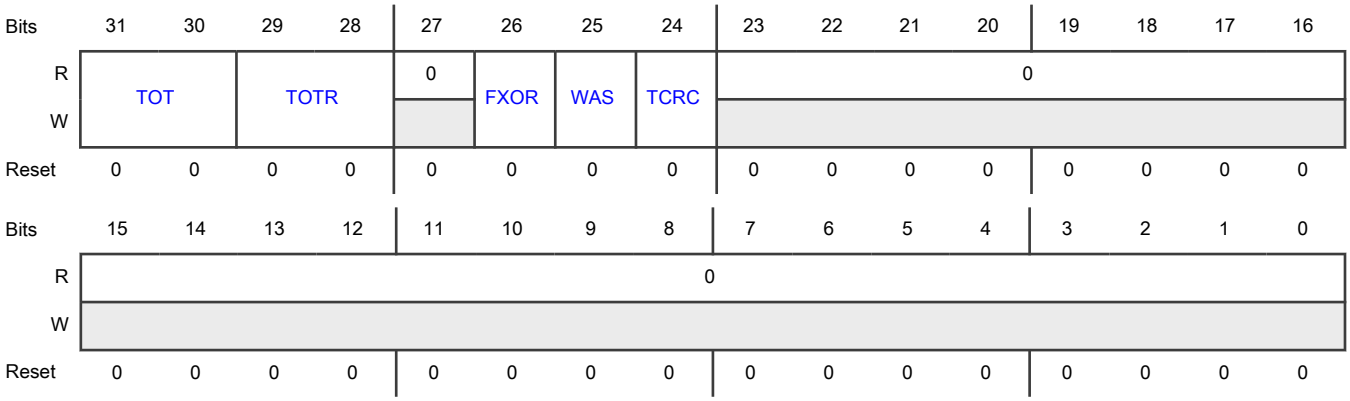
### Offset

| Register | Offset |
|---|---|
| CTRL | 8h |

### Function

Sets control for CRC. You must write 1 to the appropriate fields of this register before starting a new CRC calculation, which you can initialize by writing 1 to CTRL[WAS] and then writing the seed into DATA.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>TOT | Transpose Type for Write<br>Sets transpose type for the values written to DATA. See Transpose feature for the available transpose options.<br>00b - No transposition<br>01b - Bits in bytes are transposed, but bytes are not transposed. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 10b - Both bits in bytes and bytes are transposed.<br>11b - Only bytes are transposed, no bits in a byte are transposed. |
| 29-28<br>TOTR | Transpose Type for Read<br>Sets transpose type for the values read from DATA . See Transpose feature for the available transpose options.<br>00b - No transposition<br>01b - Bits in bytes are transposed, but bytes are not transposed.<br>10b - Both bits in bytes and bytes are transposed.<br>11b - Only bytes are transposed, no bits in a byte are transposed. |
| 27<br>— | Reserved |
| 26<br>FXOR | Complement Read of CRC Data Register<br>Enables on-the-fly complementing of read data.<br>Some CRC protocols require the final checksum to be XORed with FFFFFFFFh or FFFFh.<br>0b - Disables XOR on reading data.<br>1b - Inverts or complements the read value of the CRC Data. |
| 25<br>WAS | Write as Seed<br>Specifies whether writes to DATA are data values or seed values.<br>When this field = 1, the value that you write to is considered as seed value. When this field = 0, the value that you write to is considered as data for CRC calculation.<br>0b - Data values<br>1b - Seed values |
| 24<br>TCRC | TCRC<br>Defines the width of CRC.<br>0b - 16 bits<br>1b - 32 bits |
| 23-0<br>— | Reserved |

## 55.8 Glossary

| | |
|---|---|
| **Transpose** | Flipping input or output data bits (or bytes) |
| **Seed value** | Initial value of CRC calculation |

# Chapter 56
# Power Conversion and Motor Control (PCMC)

## 56.1 Introduction

The PCMC subsystem is a group of modules that can be interconnected to perform a variety of real-time tasks, such as:

- Motor control
- Power conversion
- Advanced PWM generation
- Lighting control

The modules that make up PCMC are:

- ADC
- BCTU
- eMIOS
- LCU
- LPCMP
- TRGMUX

See Feature comparison for the number of instances of each module in your chip.

## 56.2 Block diagram

The figure and table below show a typical configuration.

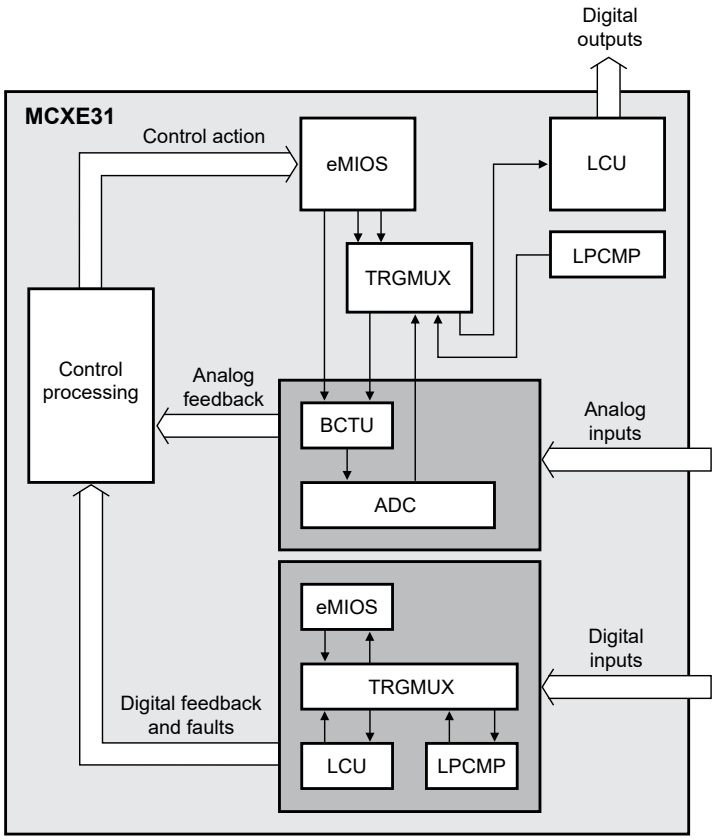



Figure 205. Block diagram

Table 256. Block diagram components

| Component | Description |
|---|---|
| ADC | Measures external analog signals |
| BCTU | Performs advanced ADC triggering |
| Control processing | Arm Cortex-M7 core:<br>• Processes digital and analog inputs to generate control outputs<br>• Writes control outputs to peripherals |
| eMIOS | • Generates high-resolution PWM outputs<br>• Measures time-based digital inputs |
| LCU | Performs programmable logic functions and fault control |
| LPCMP | Compares analog values to detect faults |
| TRGMUX | • Routes signals between internal peripherals<br>• Routes external signals to internal peripherals |

## 56.3 Functional overview

### 56.3.1 Analog signal measurement

ADC measures the voltage of external analog signals and converts that voltage to a digital value. Each ADC instance operates independently of other instances. Software or hardware can trigger conversions. Together with BCTU and eMIOS, ADC can preempt normal measurements by injecting high-priority measurements when needed.

See Analog-to-Digital Converter.

### 56.3.2 Advanced ADC triggering

BCTU works with ADC to initiate conversions triggered by outputs from other modules in the PCMC subsystem, such as PWM signals from eMIOS or LCU sequential logic outputs.

With BCTU, you can group measurements into configuration lists containing up to 32 items. Each item contains information such as:

- ADC instance to use for the conversion
- The ADC channel
- The trigger event for initiating the next conversion
- The conversion list address

BCTU can store conversion results by routing them through DMA or putting them in FIFO queues. The FIFO queues enable conversions to occur seamlessly by storing conversion results without software intervention.

BCTU has 72 trigger inputs, 69 of which come from the eMIOS channel outputs. The three remaining trigger inputs come from other on-chip modules via TRGMUX. You can also trigger conversions by software through the BCTU register interface.

BCTU can issue a single conversion or a list of conversions based on a single trigger occurrence. BCTU signals the eMIOS timers when ADC receives a single command or a list of commands. ADC sends conversion data back to BCTU, which loads it in the ADC*n* Result Data (ADC*n*DR) register for the CPU to read.

BCTU-to-ADC interface illustrates the connections between BCTU and an ADC.

See also:

- Body cross-triggering unit (BCTU)
- PCDR*n*, ICDR*n*, and ECDR*n* register descriptions in ADC register descriptions

### 56.3.3 BCTU-to-ADC interface

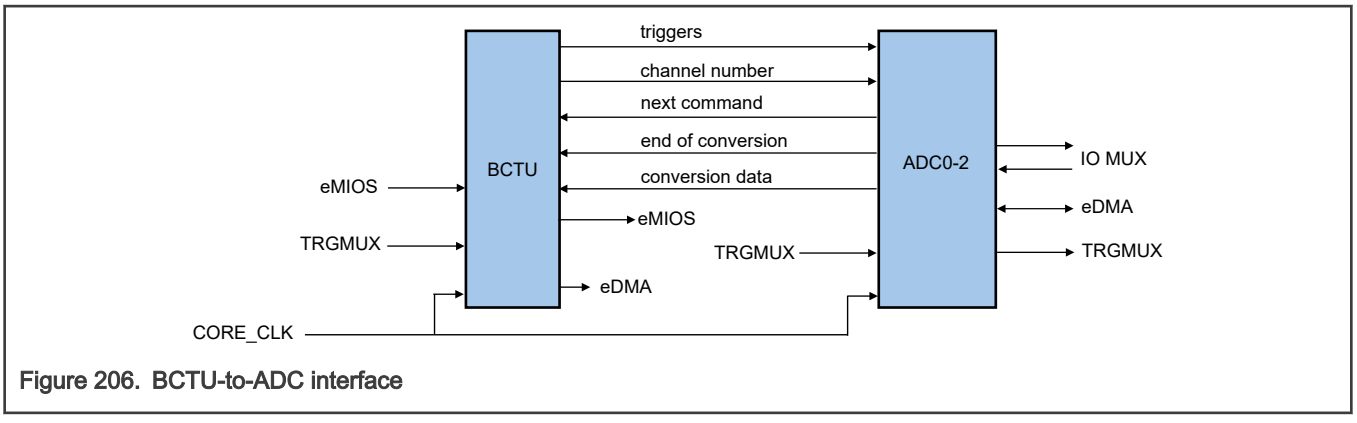



Figure 206. BCTU-to-ADC interface

### 56.3.4 Generate PWM outputs and measure digital inputs

eMIOS generates high-resolution PWM outputs. Each eMIOS instance has 23 dedicated channel outputs. You can configure each channel to operate in a different mode. Depending on the mode, eMIOS channels can:

- Measure the period of an input signal
- Measure the width of an input pulse
- Function as general-purpose I/O
- Count pulses or edges
- Capture input values
- Compare outputs

For example, eMIOS can generate periodic ADC triggers. Using TRGMUX, you can route eMIOS triggers to other on-chip peripherals such as LCU, another eMIOS instance, external chip I/O, and so on.

eMIOS connections illustrates the input and output connections for eMIOS.

See also:

- Enhanced Modular IO Subsystem (eMIOS)
- Body cross-triggering unit (BCTU)
- Trigger Mux (TRGMUX)

### 56.3.5 eMIOS connections

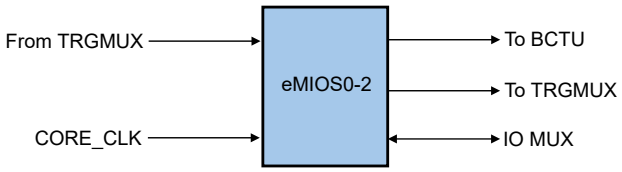



Figure 207. eMIOS connections

### 56.3.6 Create combinatorial and sequential logic functions

LCU enables you to implement hardware-based combinatorial and sequential logic on the chip. This capability reduces cost and decreases application size.

Each LCU instance has 12 dedicated inputs and 12 outputs. You can use LCU to implement a variety of combinatorial and sequential logic functions with no CPU intervention, including a simple OR gate, flip flops, and motor control logic. As detailed in Fault detection, LCU also plays a significant role in fault management.

LCU connections shows the input and output connections for LCU.

See also:

- Logic Control Unit (LCU)
- LCU use case examples
- Trigger MUX (TRGMUX)
- TRGMUX connectivity file attached to this document

### 56.3.7 LCU connections

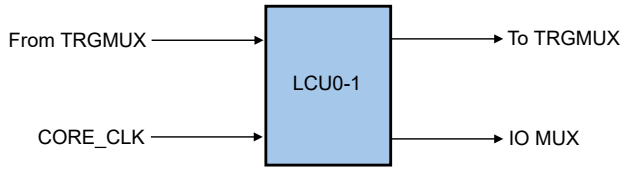



Figure 208. LCU connections

### 56.3.8 Fault detection

The PCMC subsystem uses LPCMP and LCU to implement fault detection.

LPCMP can detect fault conditions (such as short circuits, over-voltages, and excessive temperature) by comparing an external input voltage to a programmable reference voltage. It produces a digital output that indicates whether the input voltage is higher or lower than the reference voltage.

The chip can detect faults via external digital or analog signals. TRGMUX routes digital signals from SIUL to a peripheral to trigger the appropriate behavior.

LCU has four force inputs that you can use to manage fault assertion via its own internal logic mechanism.

The typical reaction to the presence of fault is to put the system in a safe state by:

- Deasserting output signals such as PWM outputs controlling a power stage.
- Disconnecting an entire circuit by switching off a relay.

See also Low Power Comparator (LPCMP).

### 56.3.9 Programmable module interconnections

TRGMUX enables you to interconnect on-chip peripherals to create application-specific configurations. TRGMUX has 128 inputs and 111 outputs, which you can interconnect in hundreds of combinations through software during runtime.

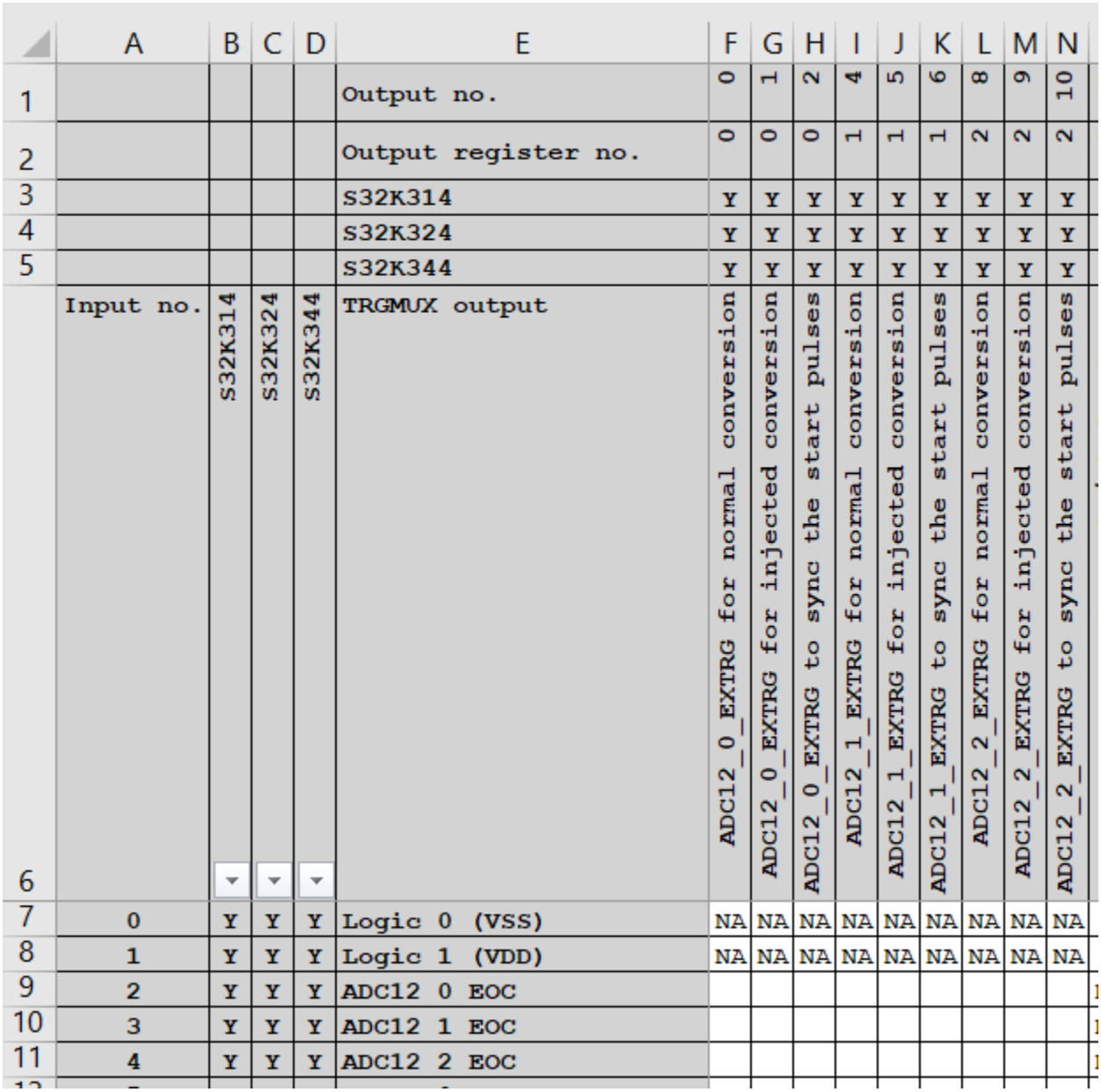
You can also route on-chip peripheral outputs to chip I/O pins to enable signal debugging. Similarly, you can route external signals to on-chip peripheral inputs.

**NOTE**

There is a two-cycle AIPS_SLOW_CLK delay through TRGMUX.

For details of PCMC interconnections, see PCMC connections diagram.

The attached TRGMUX connectivity file, a portion of which appears below, shows the interconnections possible between module instances. Column B in the spreadsheet lists TRGMUX inputs. Row 3 lists TRGMUX outputs. A connection between an input and an output is possible if the intersecting cell of the input row and the output column is blank. "NA" indicates that a connection is not possible.



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | Output no. | 0 | 1 | 2 | 4 | 5 | 6 | 8 | 9 | 10 |
| 2 | | | | | Output register no. | 0 | 0 | 0 | 1 | 1 | 1 | 2 | 2 | 2 |
| 3 | | | | | S32K314 | Y | Y | Y | Y | Y | Y | Y | Y | Y |
| 4 | | | | | S32K324 | Y | Y | Y | Y | Y | Y | Y | Y | Y |
| 5 | | | | | S32K344 | Y | Y | Y | Y | Y | Y | Y | Y | Y |
| 6 | Input no. | S32K314 | S32K324 | S32K344 | TRGMUX output | ADC12_0_EXTRG for normal conversion | ADC12_0_EXTRG for injected conversion | ADC12_0_EXTRG to sync the start pulses | ADC12_1_EXTRG for normal conversion | ADC12_1_EXTRG for injected conversion | ADC12_1_EXTRG to sync the start pulses | ADC12_2_EXTRG for normal conversion | ADC12_2_EXTRG for injected conversion | ADC12_2_EXTRG to sync the start pulses |
| 7 | 0 | Y | Y | Y | Logic 0 (VSS) | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| 8 | 1 | Y | Y | Y | Logic 1 (VDD) | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| 9 | 2 | Y | Y | Y | ADC12 0 EOC | | | | | | | | | |
| 10 | 3 | Y | Y | Y | ADC12 1 EOC | | | | | | | | | |
| 11 | 4 | Y | Y | Y | ADC12 2 EOC | | | | | | | | | |

Figure 209. Portion of TRGMUX connectivity

See also:

- Trigger Mux (TRGMUX)
- TRGMUX connectivity file attached to this document

#### 56.3.9.1 PCMC connections diagram

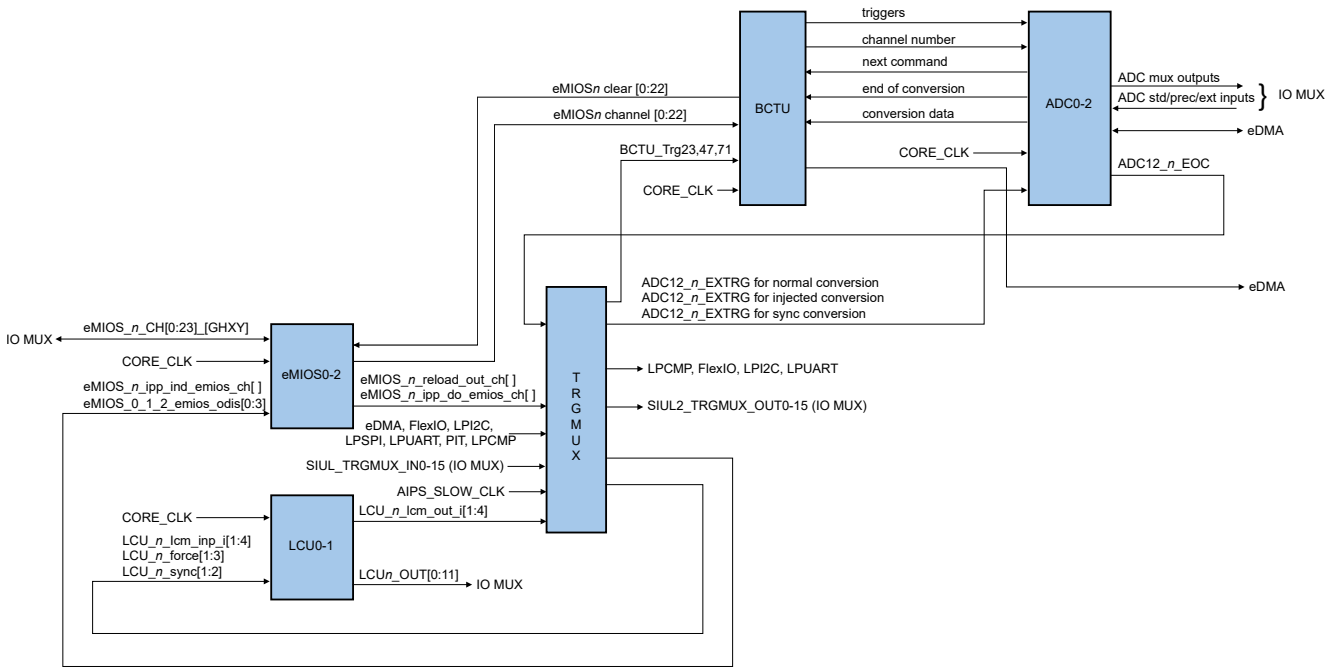



Figure 210. PCMC connections diagram

## 56.4 Reset, clocking, and other considerations

### 56.4.1 Reset

ADC, BCTU, eMIOS, LCU, and TRGMUX fully reset during a POR, destructive reset, or functional reset.

See also:

- Module reset status
- Software reset

### 56.4.2 Clocking

Detailed clocking information is given in the links below. In summary:

- CORE_CLK clocks ADC, BCTU, eMIOS, and LCU.
- AIPS_SLOW_CLK clocks TRGMUX.
- In general, you must set AIPS_SLOW_CLK to one-fourth or one-half the frequency of CORE_CLK.

See also:

- ADC and motor control modules
- Clocking use case examples
- System clock frequency limitations
- Clock frequency

### 56.4.3 Power modes

All ADC, BCTU, eMIOS, LCU, and TRGMUX instances operate only in Run mode. None of the instances operate in Standby mode.

See also:

- Module reset status
- ADC modes of operation
- LCU modes of operation

## 56.5 Use case examples

### 56.5.1 Motor control use case

The following sections show:

- A PMSM control system
- A BLDC control system

Both PMSM and BLDC use a rotating magnetic stator field to turn the stator consisting of permanent magnets. The stator is driven by 3-phase AC voltage. The difference between PMSM and BLDC is in how stator position is calculated.

- PMSM calculates stator position by measuring back EMF from the motor.
- BLDC calculates stator position based on feedback from motor sensors.

#### 56.5.1.1 PMSM motor control use case

##### 56.5.1.1.1 Generic PMSM motor control

The following figure and table illustrate a generic PMSM motor control system. The feedback loops in the system create a motor control loop.

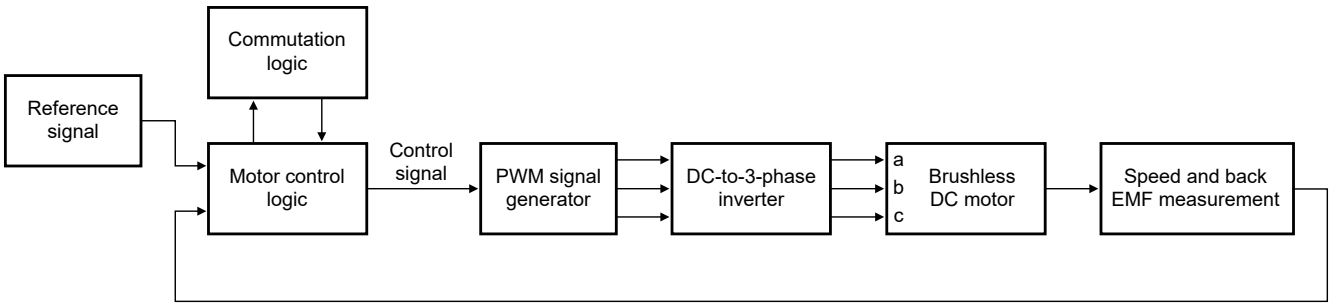



Figure 211. Generic PMSM motor control

Table 257. Block diagram components

| Block | Description |
|---|---|
| Angular position sensor | Generates data that enables commutation logic to calculate rotor position. |
| Brushless DC motor | Electronically commutated electric motor. |
| Commutation logic | Compares values from the rotor angular position sensors to entries in a lookup table, and applies voltage to stator winding corresponding to same set of values in lookup table. |

*Table continues on the next page...*

Table 257. Block diagram components (continued)

| Block | Description |
|---|---|
| DC-to-3-phase inverter | Switching network that transmits power to the motor stator windings according to a control signal. |
| Motor control logic | • Compares speed measurement to required speed and adjusts the PWM signal generator.<br>• Generates the assertions for the PWM signal generator according to commutation logic input. |
| PWM signal generator | Generates PWM signal voltage source. |
| Reference signal | Signal that indicates required motor speed. |
| Speed and back EMF measurement | An ADC is linked to a shunt resistor to perform measurements and detect over-current conditions. If one is found, then hardware generates a fault interrupt. |

#### 56.5.1.1.2 PMSM motor control

The PCMC subsystem consists of a group of modules that enable you to implement PMSM motor control loops in a variety of ways. The following figure and table illustrate one possible implementation.

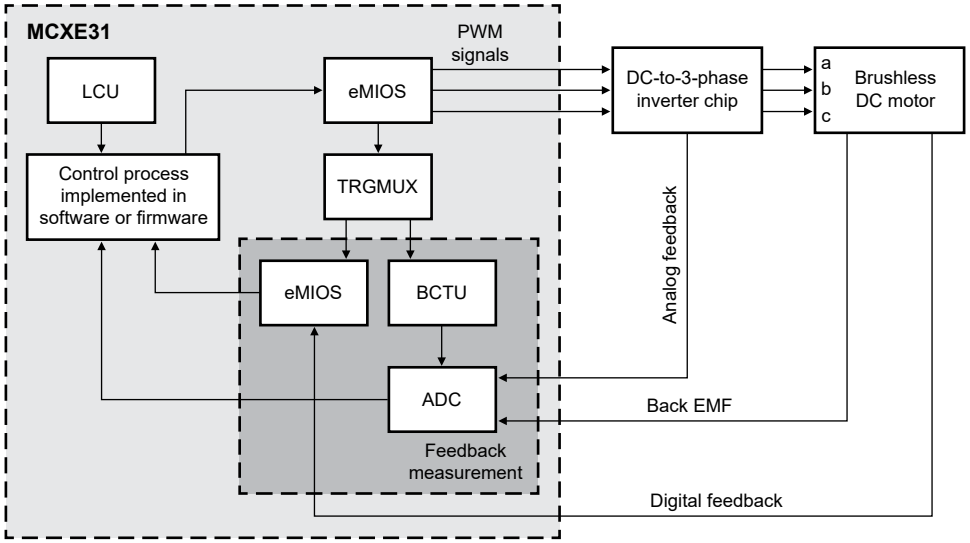



Figure 212. PMSM motor control

Table 258. Block diagram components

| Block | Description |
|---|---|
| ADC | Measures analog inputs (currents and/or voltages) and converts to digital values. Typically, it measures currents and can be configured to generate a fault interrupt when a measured input is outside a pre-defined acceptable range. |
| BCTU | Triggers conversions by ADC. |
| Control process | Provides top-level system control functions for FOC or another feasible control algorithm: |

*Table continues on the next page...*

Table 258. Block diagram components (continued)

| Block | Description |
|---|---|
| | • Compares speed measurement to required speed and adjusts the PWM signal generator.<br>• Generates the assertions for the PWM signal generator according to commutation logic input. |
| DC-to-3-phase inverter | Switching network that transmits power to the motor stator windings according to a control signal. |
| eMIOS | • Generates PWM input for DC to 3-phase inverter.<br>• Measures digital feedback inputs—BLDC position sensors, for example. |
| LCU | Implements commutation lookup table. Control Process compares values from the angular position sensors to entries in the lookup table. It then applies voltage to stator winding corresponding to same set of values in lookup table. |
| TRGMUX | • Enables interconnect to the required set of on-chip peripherals.<br>• Indirectly triggers ADC sampling via BCTU.<br>• Triggers eMIOS to measure digital feedback signals. |

### 56.5.1.2 Sensored BLDC motor control use case

#### 56.5.1.2.1 Generic sensored BLDC motor control

The following figure and table illustrate a generic sensored BLDC motor control system. The feedback loops in the system create a motor control loop.

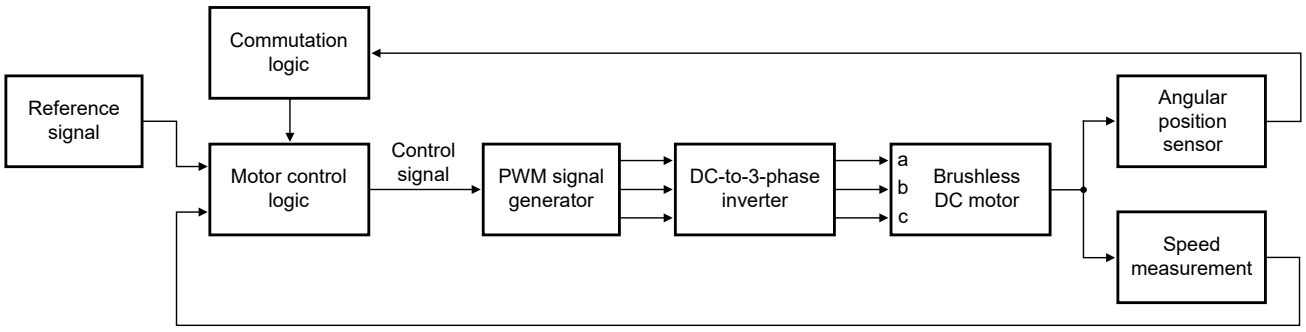



Figure 213. Generic sensored BLDC motor control

Table 259. Block diagram components

| Block | Description |
|---|---|
| Angular position sensor | Hall effect sensors or optical decoders detect rotor position |
| Commutation logic | Compares values from the angular position sensors to entries in a lookup table and applies voltage to stator winding corresponding to same set of values in lookup table |
| Reference signal | Signal that indicates required motor speed |

*Table continues on the next page...*

Table 259. Block diagram components (continued)

| Block | Description |
|---|---|
| DC-to-3-phase inverter | Switching network that transmits power to the stator windings according to a control signal |
| Motor control logic | • Compares speed measurement to required speed and adjusts the PWM signal generator<br>• Generates the assertions for the PWM signal generator according to commutation logic input |
| Speed measurement | Uses an ADC linked to a shunt resistor to perform measurements and detect over-current conditions, in which case hardware generates an interrupt. |

#### 56.5.1.2.2 PCMC sensored field-oriented control (FOC) of 3-phase BLDC motor

FOC is a VFD control method in which the stator currents of a 3-phase AC electric motor (such as BLDC, PMCM, reluctance motor, or other spinning electrical machine) is identified as a vector.

The PCMC subsystem consists of a group of modules that enable you to implement sensored FOC BLDC motor control loops in a variety of ways. The following figure and table show one possible implementation.

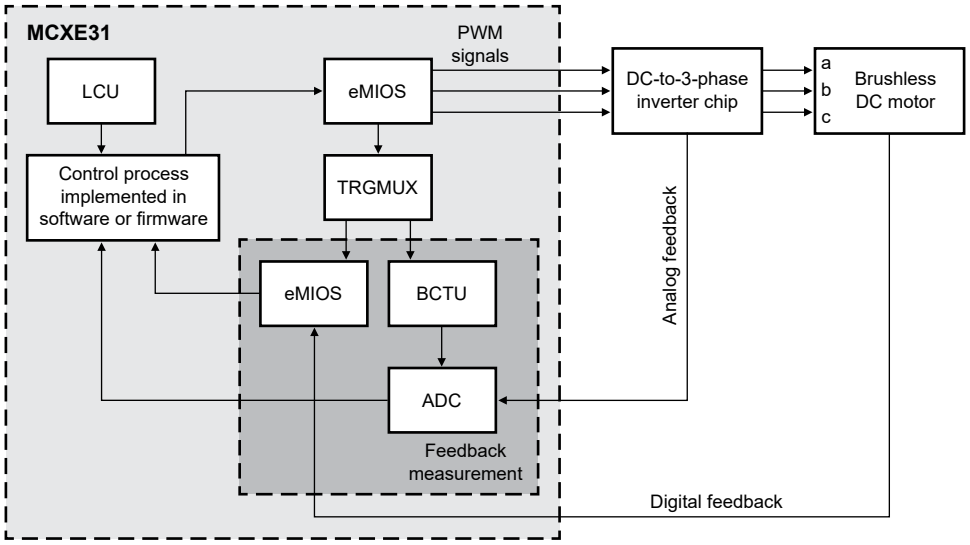



Figure 214. PCMC sensored BLDC motor control

Table 260. Block diagram components

| Block | Description |
|---|---|
| ADC | Measures analog inputs (currents and/or voltages) and converts to digital values. Typically, it measures currents, and can be configured to generate a fault interrupt when a measured input is outside a pre-defined acceptable range. |
| BCTU | Triggers conversions by ADC. |
| Control process | Provides top-level system control functions for FOC control: |

*Table continues on the next page...*

Table 260. Block diagram components (continued)

| Block | Description |
|---|---|
| | • Compares speed measurement to required speed and adjusts the PWM signal generator.<br>• Generates the assertions for the PWM signal generator according to commutation logic input. |
| DC-to-3-phase inverter | Switching network that transmits power to the motor stator windings according to a control signal. |
| eMIOS | • Generates PWM input for DC to 3-phase inverter.<br>• Measures digital feedback inputs—BLDC position sensors, for example. |
| LCU | Implements commutation lookup table. Control process compares values from the angular position sensors to entries in the lookup table. It then applies voltage to stator winding corresponding to same set of values in lookup table. |
| TRGMUX | • Enables interconnect of the required set on on-chip peripherals.<br>• Indirectly triggers ADC sampling via BCTU.<br>• Triggers eMIOS to measure digital feedback signals. |

## 56.6 Design considerations

### 56.6.1 ADC design considerations

The following topics discuss factors that affect ADC function in applications.

#### 56.6.1.1 ADC measurement accuracy

The FOC motor control algorithm requires a high degree of confidence in the reliability of conversion data. Here are some best practices that contribute to ADC precision.

- Use precision channels (see ADC precision channels).
- Minimize input noise (see ADC input noise).
- Use BCTU triggering (see ADC triggering).

##### 56.6.1.1.1 ADC precision channels

See the chip-specific ADC configuration information for the number of precision channels for each ADC instance.

See the channel mapping table for the channel numbers for the precision channels.

##### 56.6.1.1.2 ADC input noise

Regardless of the application, ADCs are inherently sensitive to input noise. The following best practices help minimize input noise.

- Avoid simultaneous conversions.
- Consider using a low-pass filter on input pins.

All variants of this chip include multiple, independent ADC instances, so it is easily possible to have simultaneous sampling by different ADC instances. This should be avoided, though, because ADC sampling can cause noise on adjacent device ADC inputs. Avoidance of simultaneous sampling should be ensured by software design. This is important for implementations of the FOC algorithm.

Additionally, an RC low-pass filter on ADC input pins can help reduce noise and enhance accuracy.

#### 56.6.1.1.3 ADC triggering

Some applications, including FOC motor control, must use BCTU for precise triggering of ADC measurement. This is because the FOC algorithm requires sampling at specific times to implement the “observer/predictor”. BCTU makes it possible to perform precisely-timed ADC measurement.

For less exigent algorithms or non-closed-loop controlled motors, BCTU triggering and precision ADC channels are not necessarily required.

### 56.6.1.2 Example ADC configurations

Following are some ADC configurations that might be used for FOC motor control.

**Table 261. Example ADC configurations**

| Description | Settings |
|---|---|
| Single-shot mode | • Enable Overwrite (OWREN=1)<br>• Disable Write Left-Aligned (WLSIDE=0)<br>• Select single shot (MODE=0)<br>• Disable External Trigger (TRGEN=0)<br>• Disable Injection Trigger (JTRGEN=0)<br>• Disable BCTU triggering (BCTUEN=0)<br>• Disable averaging (AVGEN=0)<br>• Select conversion clock = 1/2 module clock (ADCCLKSEL=01) |
| Normal conversion scan mode | • Enable Overwrite (OWREN=1)<br>• Disable Write Left-Aligned (WLSIDE=0)<br>• Select looped conversions (MODE=1)<br>• Disable External Trigger (TRGEN=0)<br>• Disable Injection Trigger (JTRGEN=0)<br>• Disable BCTU triggering (BCTUEN=0)<br>• Disable averaging (AVGEN=0)<br>• Select conversion clock = 1/2 module clock (ADCCLKSEL=01) |
| BCTU control mode | • Enable Overwrite (OWREN=1)<br>• Disable Write Left-Aligned (WLSIDE=0)<br>• Select single shot (MODE=0)<br>• Disable External Trigger (TRGEN=0)<br>• Disable Injection Trigger (JTRGEN=0)<br>• Enable BCTU triggering (BCTUEN=1)<br>• Disable averaging (AVGEN=0)<br>• Select conversion clock = 1/2 module clock (ADCCLKSEL=01) |

*Table continues on the next page...*

Table 261. Example ADC configurations (continued)

| Description | Settings |
|---|---|
| BCTU trigger mode | • Enable Overwrite (OWREN=1)<br>• Disable Write Left-Aligned (WLSIDE=0)<br>• Select single shot (MODE=0)<br>• Disable External Trigger (TRGEN=0)<br>• Disable Injection Trigger (JTRGEN=0)<br>• Enable BCTU triggering (BCTUEN=1)<br>• Enable all trigger sources (BCTU_MODE=1)<br>• Disable averaging (AVGEN=0)<br>• Select conversion clock = 1/2 module clock (ADCCLKSEL=01) |

## 56.6.2 BCTU design considerations

For reasons discussed in ADC triggering, some applications require precise ADC triggering and BCTU provides that capability. The following topics discuss some design considerations when using BCTU.

### 56.6.2.1 BCTU or ADC data retrieval

Although the primary function of BCTU is to provide precisely timed triggers for ADC conversions, there is some overlap in BCTU and ADC function in that conversion data can be retrieved from either module. For applications in which frequent conversions are performed, such as real-time motor control, BCTU offers a distinct advantage over ADC, because BCTU trigger sources can be configured to route conversion results to a BCTU internal FIFO. Doing so prevents subsequent conversion data from overwriting previous data, making it available for either direct access or DMA transfer to system RAM.

### 56.6.2.2 Disabling BCTU triggers

There are fault scenarios in which an application must disable BCTU triggers as part of putting the system into a safe state. Examples include:

- Over-voltage condition
- Under-voltage condition
- Calibration parameter mismatch between the embedded motor control system and the motor

Application fault reaction design must include the disabling of BCTU triggers in scenarios in which either system conditions indicate an impending failure, or ADC conversion data is outside control boundaries.

## 56.6.3 eMIOS design considerations

In a BLDC motor control application, eMIOS plays a central role, providing the driving PWM signal as well as performing feedback signal measurements. The following topics discuss eMIOS considerations in the application design.

### 56.6.3.1 PWM considerations

In a BLDC motor control application, PWM/power signal requirements are affected by a number of parameters, including:

- Displacement Power Factor
- DF
- THD
- DVA

- CF
- FF

Ensure that the above parameters are included when choosing the timebase and prescaler combination for eMIOS channels producing PWM signals.

#### 56.6.3.2 Input measurement

In a BLDC motor control application, eMIOS measures feedback signals for the motor and provides critical information to the control system. Selecting a timebase and prescaler for eMIOS channels performing input measurement tasks is usually a tradeoff between application requirements for precision and the computational cost in relation to the selected clock speed.

For example, some applications like electric steering demand an extremely precise control for position and torque. This is different from a compressor e-motor, where speed and torque do not demand rigid precision requirements.

Other considerations are related to control of:

- THD
- FF
- GM
- pf correction

The above considerations are significant from the perspective of the PWM reaction speed for the output signal for BLDC applications. These parameters must be considered in the design.

#### 56.6.3.3 eMIOS channel modes useful in motor control

**Table 262. eMIOS channel modes useful in motor control**

| Mode | Description | Use in motor control |
|---|---|---|
| MCB | Modulus Counter Buffered (Up/Down) | Time base<br>• PWM frequency<br>• Reload |
| IPWM | Input Pulse Width Measurement | Capture modes:<br>• Speed measurement<br>• Position measurement<br>• Voltage (duty cycle) |
| IPM | Input Period Measurement | Speed measurement |
| PEC | Pulse Edge Counting | Position measurement |
| OPWMCB | Center aligned Output PWM Buffered with dead time | PWM duty cycle |
| OPWMB | Output Pulse Width Modulation Buffered | Trigger placement within the period |
| OPWMT | Output Pulse Width Modulation Trigger | Trigger placement within the period |

##### 56.6.3.3.1 MCB PWM and trigger generation

The following figure shows one option for PWM and trigger generation.

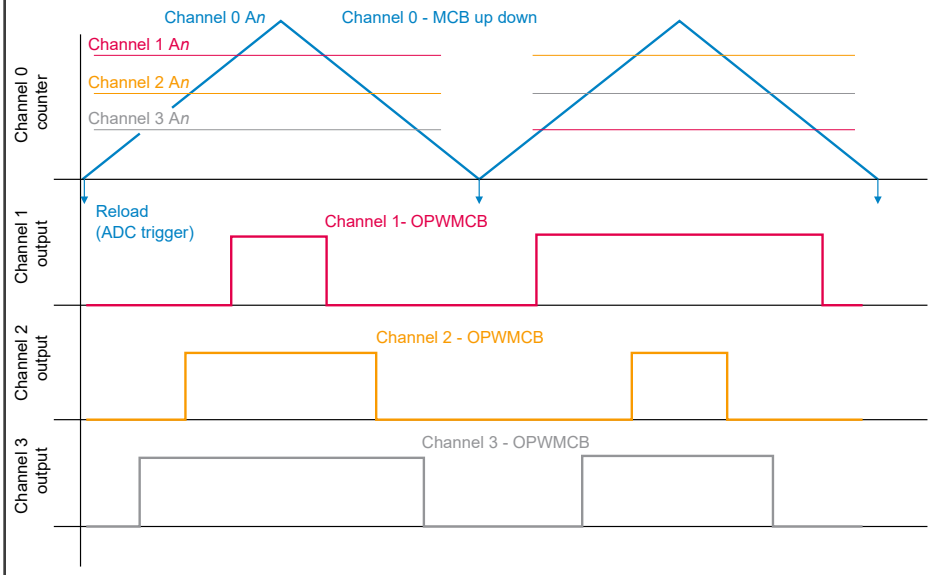



Figure 215. MCB PWM trigger generation

### 56.6.3.3.2 OPWMCB PWM and trigger generation

The following figure shows another option for PWM and trigger generation.

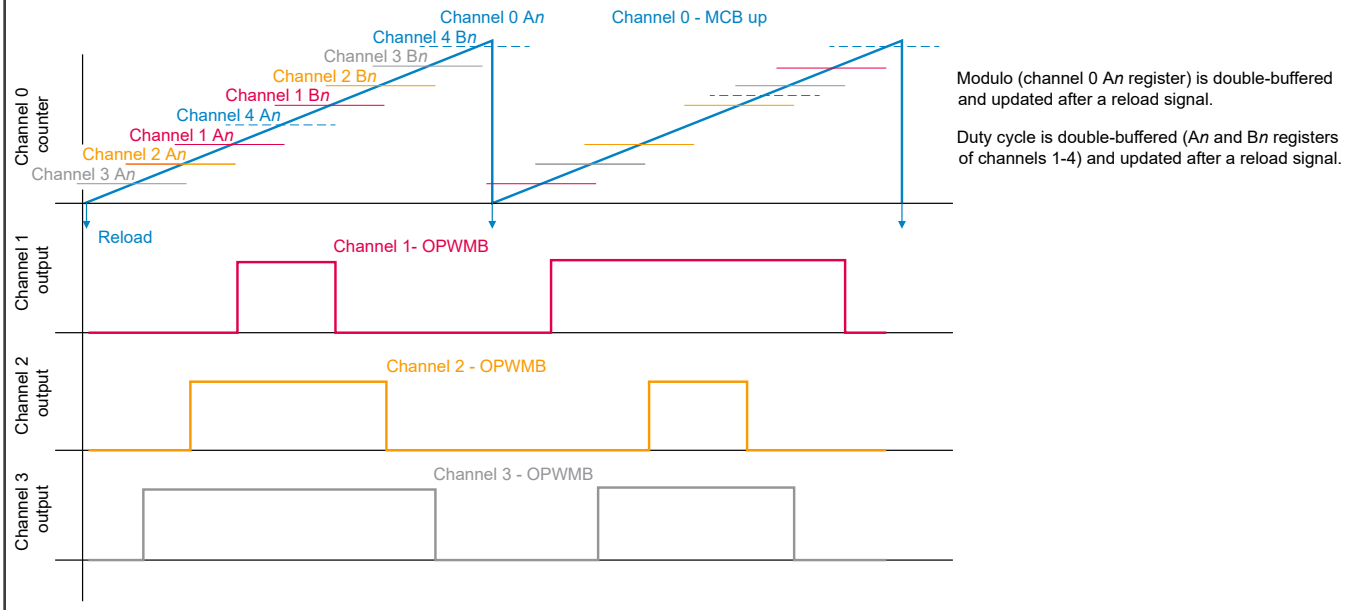



Figure 216. OPWMCB PWM trigger generation

## 56.7 Glossary

| | |
|---|---|
| **BLDC** | Brushless DC. An electronically commutated DC motor. |
| **CF** | Crest Factor. A waveform parameter that is the ratio of peak values to its RMS value. |
| **DF** | Distortion Factor. |

| | |
|---|---|
| **Displacement Power Factor** | In power electronics, the phase relationship between voltage and current: $\cos(\theta_v - \theta_i)$, where $\theta_v$ is the phase of the voltage and $\theta_i$ is the phase of the current. |
| **DVA** | Distortion Volt-Amps. |
| **FF** | Form Factor. |
| **FOC** | Field-Oriented Control. |
| **GM** | Gain Margin. |
| **pf** | Power factor. In power electronics, it is the ratio of average power to RMS power. |
| **PMSM** | Permanent Magnet Synchronous Motor. |
| **THD** | Total Harmonic Distortion. |
| **VFD** | Variable-Frequency Drive. |

# Chapter 57
# Analog-to-Digital Converter (ADC)

## 57.1 Chip-specific ADC information

### 57.1.1 ADC Configuration

This chip has up to 3 instances of ADC. ADC channels are divided into 3 groups - Precision, Standard and External (each with independent configuration settings and different accuracy/performance level). The following table shows the ADC configuration:

**Table 263. ADC instances**

| Instance | MCXE31B | MCXE315/MCXE316 and MCXE317 |
|---|---|---|
| ADC_0 | Yes | Yes |
| ADC_1 | Yes | Yes |
| ADC_2 | Yes | No |

**Table 264. ADC configuration**

| Feature | ADC_0 | ADC_1 | ADC_2 |
|---|---|---|---|
| No. of precision channels | 8 | 8 | 8 |
| No. of standard channels | 16 | 16 | 16 |
| No. of special internal channels | 1[1] | 0 | 0 |
| No. of external channels | 32 | 32 | 0 |
| BCTU trigger support | Yes | Yes | Yes |
| DMA support | Yes | Yes | Yes |
| Hardware interleaving | Yes | Yes | Yes |
| No. of watchdogs | 4 | 4 | 4 |
| No. of Hardware trigger | 3 | 3 | 3 |

1. CHAN_INT0 - channel 50(S)

**Table 265. ADC maximum clock frequency**

| Feature | MCXE31B | MCXE317 | MCXE315/ MCXE316 |
|---|---|---|---|
| ADC module clock frequency (f_mc) | 160 / 120 MHz | 120 MHz | 120 MHz |

**NOTE**

For register configurations need for different frequencies, see 'Functional description' section in this chapter.

**ADC Channel mapping Table**

Table 266. ADC channel mapping

| Channel Number | ADC_0 | ADC_1 | ADC_2 |
|---|---|---|---|
| 0 | ADC0_P0 | ADC1_P0 | ADC2_P0 |
| 1 | ADC0_P1 | ADC1_P1 | ADC2_P1 |
| 2 | ADC0_P2 | ADC1_P2 | ADC2_P2 |
| 3 | ADC0_P3 | ADC1_P3 | ADC2_P3 |
| 4 | ADC0_P4 | ADC1_P4 | ADC2_P4 |
| 5 | ADC0_P5 | ADC1_P5 | ADC2_P5 |
| 6 | ADC0_P6 | ADC1_P6 | ADC2_P6 |
| 7 | ADC0_P7 | ADC1_P7 | ADC2_P7 |
| 8 | —[1] | — | — |
| 9 | — | — | — |
| 10 | — | — | — |
| 11 | — | — | — |
| 12 | — | — | — |
| 13 | — | — | — |
| 14 | — | — | — |
| 15 | — | — | — |
| 16 | — | — | — |
| 17 | — | — | — |
| 18 | — | — | — |
| 19 | — | — | — |
| 20 | — | — | — |
| 21 | — | — | — |
| 22 | — | — | — |
| 23 | — | — | — |
| 24 | — | — | — |
| 25 | — | — | — |
| 26 | — | — | — |
| 27 | — | — | — |
| 28 | — | — | — |
| 29 | — | — | — |
| 30 | — | — | — |
| 31 | — | — | — |

*Table continues on the next page...*

Table 266. ADC channel mapping (continued)

| Channel Number | ADC_0 | ADC_1 | ADC_2 |
|---|---|---|---|
| 32 | ADC0_S[8] | ADC1_S[8] | ADC2_S[8] |
| 33 | ADC0_S[9] | ADC1_S[9] | ADC2_S[9] |
| 34 | ADC0_S[10] | ADC1_S[10] | ADC2_S[10] |
| 35 | ADC0_S[11] | ADC1_S[11] | ADC2_S[11] |
| 36 | ADC0_S[12] | ADC1_S[12] | ADC2_S[12] |
| 37 | ADC0_S[13] | ADC1_S[13] | ADC2_S[13] |
| 38 | ADC0_S[14] | ADC1_S[14] | ADC2_S[14] |
| 39 | ADC0_S[15] | ADC1_S[15] | ADC2_S[15] |
| 40 | ADC0_S[16] | ADC1_S[16] | ADC2_S[16] |
| 41 | ADC0_S[17] | ADC1_S[17] | ADC2_S[17] |
| 42 | ADC0_S[18] | ADC1_S[18] | ADC2_S[18] |
| 43 | ADC0_S[19] | ADC1_S[19] | ADC2_S[19] |
| 44 | ADC0_S[20] | ADC1_S[20] | ADC2_S[20] |
| 45 | ADC0_S[21] | ADC1_S[21] | ADC2_S[21] |
| 46 | ADC0_S[22] | ADC1_S[22] | ADC2_S[22] |
| 47 | ADC0_S[23] | ADC1_S[23] | ADC2_S[23] |
| 48 | Bandgap | Bandgap | Bandgap |
| 49 | Tempsensor output | Tempsensor output | Tempsensor output |
| 50 | ANAMUX_OUT | — | — |
| 51 | — | — | — |
| 52 | — | — | — |
| 53 | — | — | — |
| 54 | VREFL | VREFL | VREFL |
| 55 | VREFH | VREFH | VREFH |
| 56 | — | — | — |
| 57 | — | — | — |
| 58 | — | — | — |
| 59 | — | — | — |
| 60 | — | — | — |
| 61 | — | — | — |
| 62 | — | — | — |
| 63 | — | — | — |

*Table continues on the next page...*

Table 266. ADC channel mapping (continued)

| Channel Number | ADC_0 | ADC_1 | ADC_2 |
|---|---|---|---|
| 64 | ADC0_X[0]<br>ADC0_MA[2:0]=3'h0 | ADC1_X[0]<br>ADC1_MA[2:0]=3'h0 | — |
| 65 | ADC0_X[0]<br>ADC0_MA[2:0]=3'h1 | ADC1_X[0]<br>ADC1_MA[2:0]=3'h1 | — |
| 66 | ADC0_X[0]<br>ADC0_MA[2:0]=3'h3 | ADC1_X[0]<br>ADC1_MA[2:0]=3'h3 | — |
| 67 | ADC0_X[0]<br>ADC0_MA[2:0]=3'h2 | ADC1_X[0]<br>ADC1_MA[2:0]=3'h2 | — |
| 68 | ADC0_X[0]<br>ADC0_MA[2:0]=3'h6 | ADC1_X[0]<br>ADC1_MA[2:0]=3'h6 | — |
| 69 | ADC0_X[0]<br>ADC0_MA[2:0]=3'h7 | ADC1_X[0]<br>ADC1_MA[2:0]=3'h7 | — |
| 70 | ADC0_X[0]<br>ADC0_MA[2:0]=3'h5 | ADC1_X[0]<br>ADC1_MA[2:0]=3'h5 | — |
| 71 | ADC0_X[0]<br>ADC0_MA[2:0]=3'h4 | ADC1_X[0]<br>ADC1_MA[2:0]=3'h4 | — |
| 72 | ADC0_X[1]<br>ADC0_MA[2:0]=3'h0 | ADC1_X[1]<br>ADC1_MA[2:0]=3'h0 | — |
| 73 | ADC0_X[1]<br>ADC0_MA[2:0]=3'h1 | ADC1_X[1]<br>ADC1_MA[2:0]=3'h1 | — |
| 74 | ADC0_X[1]<br>ADC0_MA[2:0]=3'h3 | ADC1_X[1]<br>ADC1_MA[2:0]=3'h3 | — |
| 75 | ADC0_X[1]<br>ADC0_MA[2:0]=3'h2 | ADC1_X[1]<br>ADC1_MA[2:0]=3'h2 | — |
| 76 | ADC0_X[1]<br>ADC0_MA[2:0]=3'h6 | ADC1_X[1]<br>ADC1_MA[2:0]=3'h6 | — |
| 77 | ADC0_X[1]<br>ADC0_MA[2:0]=3'h7 | ADC1_X[1]<br>ADC1_MA[2:0]=3'h7 | — |
| 78 | ADC0_X[1] | ADC1_X[1] | — |

*Table continues on the next page...*

Table 266. ADC channel mapping (continued)

| Channel Number | ADC_0 | ADC_1 | ADC_2 |
|---|---|---|---|
| | ADC0_MA[2:0]=3'h5 | | |
| 79 | ADC0_X[1]<br>ADC0_MA[2:0]=3'h4 | ADC1_X[1]<br>ADC1_MA[2:0]=3'h4 | — |
| 80 | ADC0_X[2]<br>ADC0_MA[2:0]=3'h0 | ADC1_X[2]<br>ADC1_MA[2:0]=3'h0 | — |
| 81 | ADC0_X[2]<br>ADC0_MA[2:0]=3'h1 | ADC1_X[2]<br>ADC1_MA[2:0]=3'h1 | — |
| 82 | ADC0_X[2]<br>ADC0_MA[2:0]=3'h3 | ADC1_X[2]<br>ADC1_MA[2:0]=3'h3 | — |
| 83 | ADC0_X[2]<br>ADC0_MA[2:0]=3'h2 | ADC1_X[2]<br>ADC1_MA[2:0]=3'h2 | — |
| 84 | ADC0_X[2]<br>ADC0_MA[2:0]=3'h6 | ADC1_X[2]<br>ADC1_MA[2:0]=3'h6 | — |
| 85 | ADC0_X[2]<br>ADC0_MA[2:0]=3'h7 | ADC1_X[2]<br>ADC1_MA[2:0]=3'h7 | — |
| 86 | ADC0_X[2]<br>ADC0_MA[2:0]=3'h5 | ADC1_X[2]<br>ADC1_MA[2:0]=3'h5 | — |
| 87 | ADC0_X[2]<br>ADC0_MA[2:0]=3'h4 | ADC1_X[2]<br>ADC1_MA[2:0]=3'h4 | — |
| 88 | ADC0_X[3]<br>ADC0_MA[2:0]=3'h0 | ADC1_X[3]<br>ADC1_MA[2:0]=3'h0 | — |
| 89 | ADC0_X[3]<br>ADC0_MA[2:0]=3'h1 | ADC1_X[3]<br>ADC1_MA[2:0]=3'h1 | — |
| 90 | ADC0_X[3]<br>ADC0_MA[2:0]=3'h3 | ADC1_X[3]<br>ADC1_MA[2:0]=3'h3 | — |
| 91 | ADC0_X[3]<br>ADC0_MA[2:0]=3'h2 | ADC1_X[3]<br>ADC1_MA[2:0]=3'h2 | — |
| 92 | ADC0_X[3]<br>ADC0_MA[2:0]=3'h6 | ADC1_X[3]<br>ADC1_MA[2:0]=3'h6 | — |

*Table continues on the next page...*

Table 266. ADC channel mapping (continued)

| Channel Number | ADC_0 | ADC_1 | ADC_2 |
|---|---|---|---|
| 93 | ADC0_X[3]<br>ADC0_MA[2:0]=3'h7 | ADC1_X[3]<br>ADC1_MA[2:0]=3'h7 | — |
| 94 | ADC0_X[3]<br>ADC0_MA[2:0]=3'h5 | ADC1_X[3]<br>ADC1_MA[2:0]=3'h5 | — |
| 95 | ADC0_X[3]<br>ADC0_MA[2:0]=3'h4 | ADC1_X[3]<br>ADC1_MA[2:0]=3'h4 | — |

1. Channels marked with “-“ means they are reserved and access to any reserved channel may result in inappropriate data.

**NOTE**

The decoding signals for the selection of external ADC channels (ADC_X) are gray-encoded.

**NOTE**

Value of ANAMUX_OUT is selected by DCMRWF1[SUPPLY_MON_SEL] field.

**NOTE**

The module clock mentioned in this chapter is CORE_CLK for this chip.

### 57.1.2 ANAMUX for internal supply monitoring

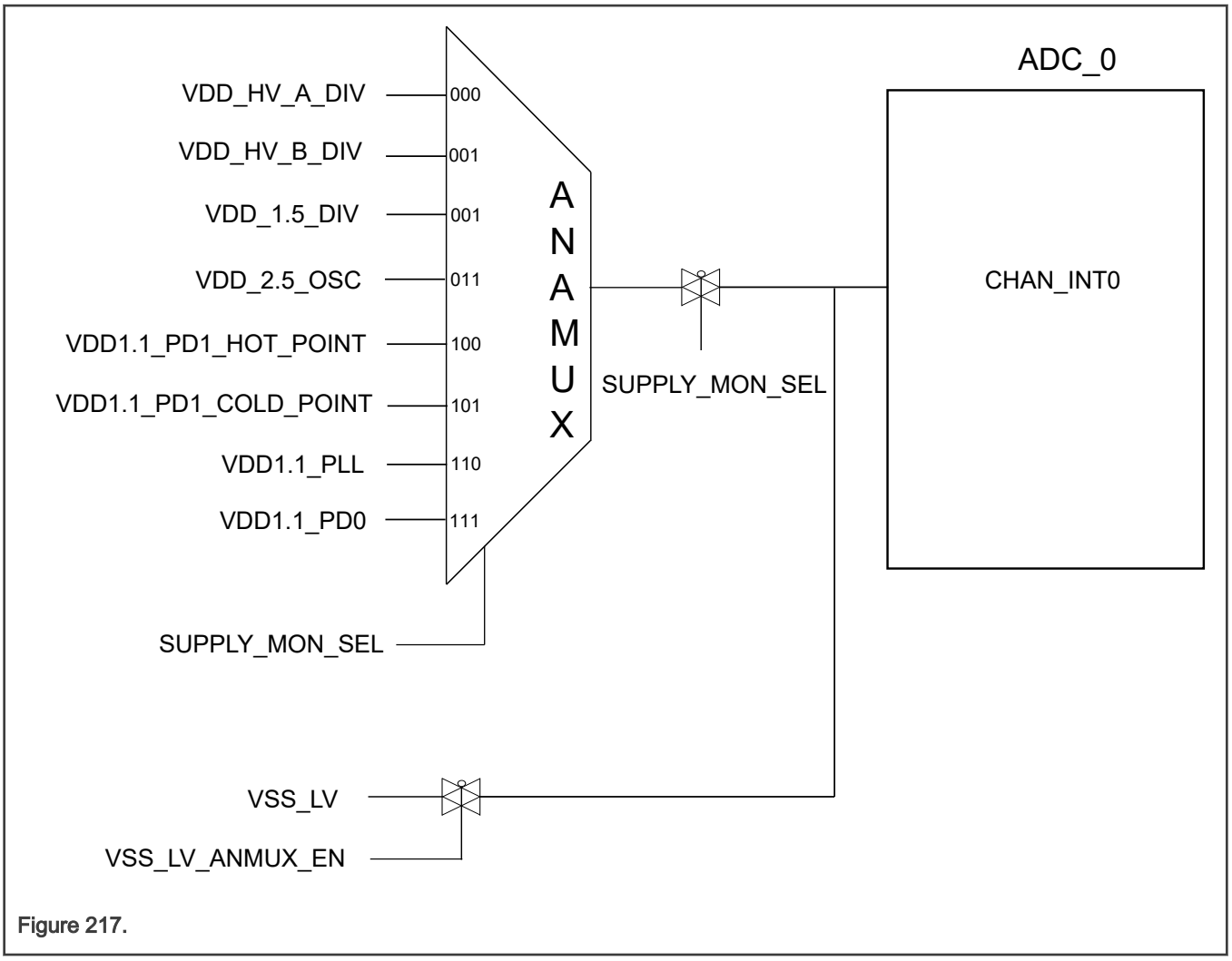



Figure 217.

**NOTE**

VDD_HV_B_DIV, VDD_1.5_DIV are not available in MCXE317 and MCXE315/MCXE316.

### 57.1.3 BCTU Interface

All 3 instances of ADC are triggered from the BCTU. The BCTU provides channel conversion commands to the ADC (includes channel number information). In addition, the ADC provides the conversion result back to the BCTU.

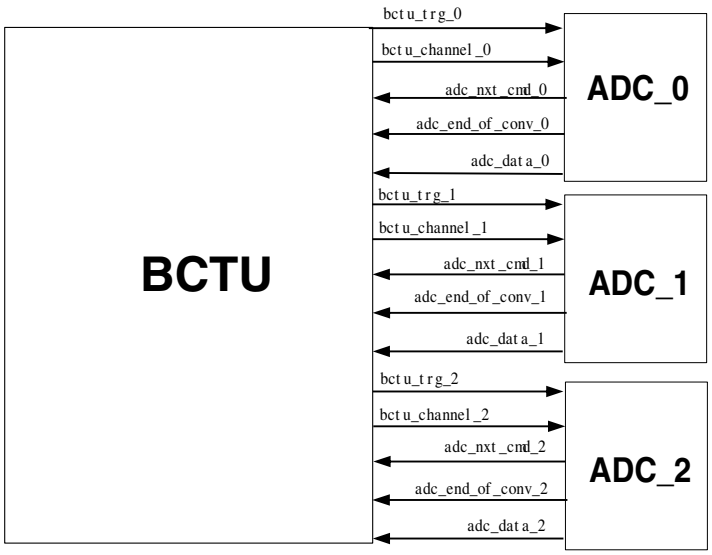



Figure 218. ADC Cross-triggering Unit

BCTU gives trigger pulse to all the 72 channels of ADC to initiate a conversion based on BCTU channel number.

#### BCTU-ADC result register

ADC result registers (ADCx_PCDRn[CDATA], ADCx_ICDRn[CDATA] and ADCx_ECDRn[CDATA]) store 15 bit conversion data. The BCTU ADC registers store 15-bit conversion data (BCTU_ADC0DR[ADC0_DATA], BCTU_ADC1DR[ADC1_DATA] and BCTU_ADC2DR[ADC2_DATA]).

### 57.1.4 DMA interface

Each ADC supports a single DMA request that can be requested after the conversion of every channel. The conversion result is stored into a register DMAR and is transferred via DMA or host access.

**NOTE**

The last ongoing conversion will be aborted as soon as ADC gets STOP request from the software and it will move to STOP mode.

### 57.1.5 Hardware triggers

In this chip, hardware trigger signals can be provided from TRGMUX outputs. This feature enables synchronous conversion of two independent ADC instances in parallel. If ADC is in Idle state (that is, no conversion phase ongoing and the MCR[PWDN] and MCR[ACKO] fields are 0) and the MCR[XSTRTEN] is set, an event on the external start signal causes ADC to start either normal or injected conversion operation. The MCR[NSTART]/MCR[JSTART] field is automatically set respectively. The normal conversion sync pulse is used with normal conversion trigger to synchronize the start timing of normal conversion between multiple ADC instances.

Table 267. Hardware triggers

| TRGMUX output number | Hardware trigger functions |
|---|---|
| 0 | ADC_0 Normal Conversion |
| 1 | ADC_0 Injected Conversion |
| 2 | ADC_0 Normal Conversion Sync Pulse |

*Table continues on the next page...*

Table 267. Hardware triggers (continued)

| TRGMUX output number | Hardware trigger functions |
|---|---|
| 4 | ADC_1 Normal Conversion |
| 5 | ADC_1 Injected Conversion |
| 6 | ADC_1 Normal Conversion Sync Pulse |
| 8 | ADC_2 Normal Conversion |
| 9 | ADC_2 Injected Conversion |
| 10 | ADC_2 Normal Conversion Sync Pulse |

### 57.1.6 ADC Self-Test

It is important to check at regular intervals if the ADC is operating correctly. For this purpose a self test feature is provided. When self-test is enabled, the ADC automatically checks its components and flags errors.

### 57.1.7 ADC mux-mode channels

There are some channels in ADC which are driven by multiple pads. For this, GPR bits from DCM are used to define which pad is connected to a particular ADC channel.

For example as below, where ADC0 S8 is driven by GPIO_PAD0 (default) and GPIO_PAD45 (mux_mode). ADC2 S8 is driven by GPIO_PAD133 (default) and GPIO_PAD45 (mux_mode). ADC1 S8 has a dedicated pad GPIO45, so its TGATE doesn't need GPR bit gating. Similar implementation is for ADC0_S9 and ADC2_S9. Following DCM GPR bits have been used:

Table 268. ADC mux_mode GPR

| ADC channel | GPR_bit | Connected pad |
|---|---|---|
| ADC0 Standard channel 8 (ADC0 S8) | DCM.DCMRWF4[1] | 0: GPIO PAD0 (default)<br>1: GPIO PAD45 |
| ADC0 Standard channel 9 (ADC0 S9) | DCM.DCMRWF4[2] | 0: GPIO PAD1 (default)<br>1: GPIO PAD46 |
| ADC 1 standard channel 14 (ADC1 S14) | DCM.DCMRWF4[3] | 0: GPIO PAD69 (default)<br>1: GPIO PAD32 |
| ADC 1 standard channel 15 (ADC1 S15) | DCM.DCMRWF4[4] | 0: GPIO PAD4 (default)<br>1: GPIO PAD33 |
| ADC1 Standard channel 22 (ADC1 S22) | DCM.DCMRWF4[5] | 0: GPIO PAD124 (default)<br>1: GPIO PAD145 |
| ADC1 Standard channel 23 (ADC1 S23) | DCM.DCMRWF4[6] | 0: GPIO PAD125 (default)<br>1: GPIO PAD146 |
| ADC2 Standard channel 8 (ADC2 S8) | DCM.DCMRWF4[9] | 0: GPIO PAD133 (default)<br>1: GPIO PAD45 |
| ADC2 Standard channel 9 (ADC2 S9) | DCM.DCMRWF4[10] | 0: GPIO PAD132 (default)<br>1: GPIO PAD46 |

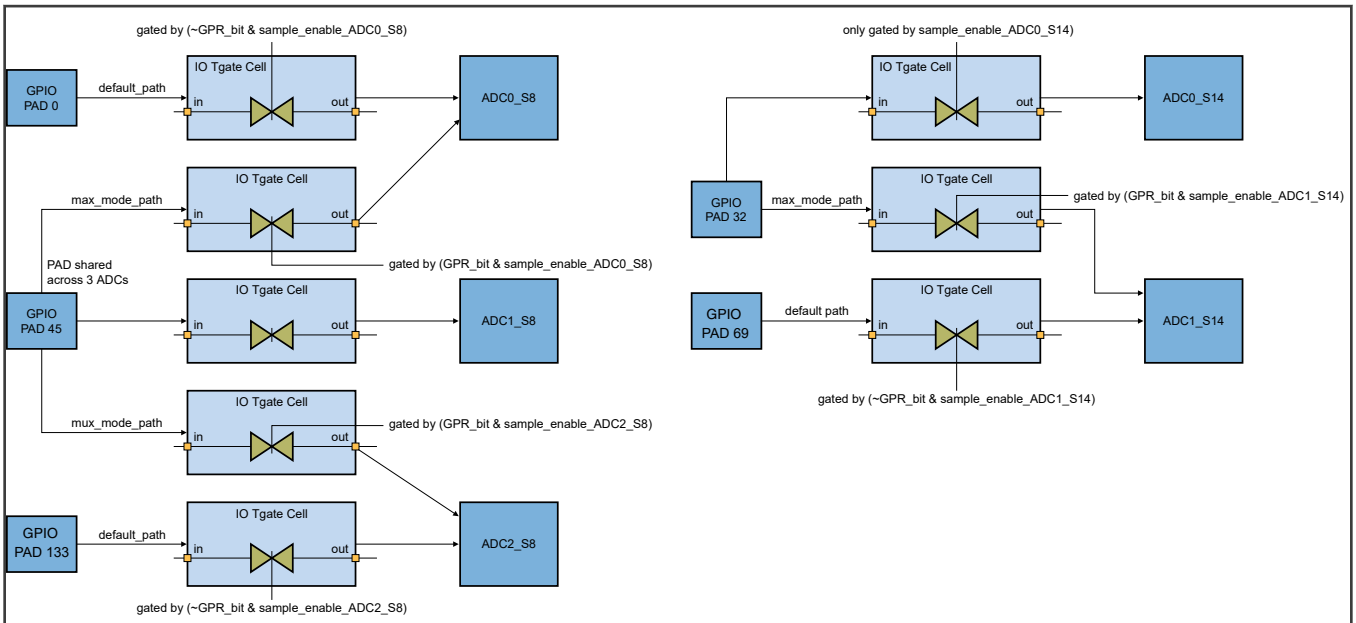



Figure 219. ADC channel interleaving

## 57.2 Overview

ADC produces a digital value from an input analog voltage using the SAR algorithm. It features various methods to trigger a conversion and offers flexibility in the selection of input channels. ADC is useful for a wide range of applications.

A configurable self-test capability supports use cases with increased requirements for functional safety.

## 57.2.1 Block diagram

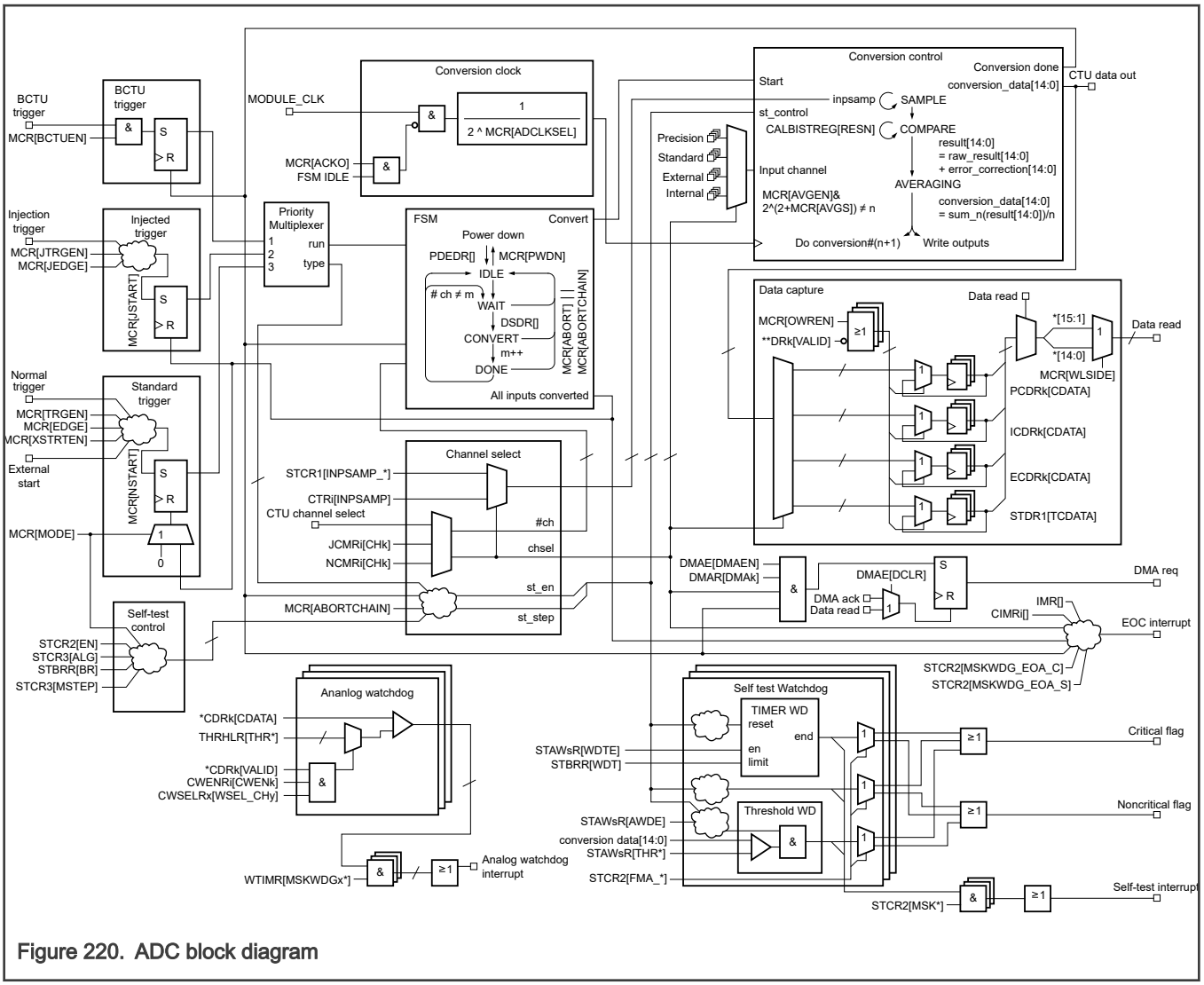



Figure 220. ADC block diagram

Guide to the diagram:

- The signal flow through the diagram is from left to right: inputs to blocks are on the left side, outputs are on the right.
- Block names have the first letters in uppercase and the other letters in lowercase—"Conversion Control," for example).
- Register names are in uppercase followed by a field name in brackets ("MCR[NSTART]," for example). Letters in lowercase are indexes. Asterisks are placeholders for letters (wildcards). Register names in the block diagram map to the names in ADC register descriptions.
- Signal names are in lowercase ("conversion done," for example).
- All digital signals shown are synchronous signals. Asynchronous inputs to the ADC are internally synchronized (synchronizers not shown).

The block diagram shows the signal flow through ADC from the left to the right and from the top to the bottom:

1. When ADC receives one of the three different types of triggers (BCTU, injected, or normal), the priority multiplexer initiates the conversion.
2. The input to be converted is selected in the Channel Select subblock, based on the type of trigger and on the self-test configuration.

3. The state machine (FSM) transitions from Idle state to Convert state. In case the channel to convert changed since the last conversion and this new channel is an external channel, the conversion starts after a delay configured by DSDR[DSD].
4. After a conversion begins, the Conversion Control block holds the internal capacitance network CDAC in the sample phase for a time defined by CTRi[INPSAMP].
5. During the compare phase, CALBISTREG[RESN] defines the number of steps used to execute the successive approximation algorithm. The error_correction values, determined during calibration, are added to the raw result. If averaging is enabled (MCR[AVGEN] = 1), up to 32 (specified via MCR[AVGS]) conversion results are averaged to obtain the result.
6. When the conversion completes and all averaging steps have completed:
    a. The conversion done signal indicates completion.
    b. Conversion_data[14:0] is written to the result registers in the Data Capture subblock.
    c. The FSM transitions into Done state.
7. The next input in the Channel Select block is selected and the next conversion is started by the transition of the FSM to Convert state. This continues until all selected inputs have been converted.
8. If an analog watchdog is enabled, the conversion result is compared to the configured threshold. If the limits are exceeded, an interrupt can be triggered.
9. Self-test can check the integrity of ADC, either interleaved with normal conversions or as a standalone check.
10. If an ongoing conversion is aborted (by writing 1 to MCR[ABORT]) or an ongoing conversion of a set of input channels is aborted (by writing 1 to MCR[ABORTCHAIN]), the FSM transitions into Idle state. The Conversion Control block stops the current conversion if MCR[ABORT] was written with 1, or finishes the current conversion if MCR[ABORTCHAIN] was written with 1. In both cases, the conversion done output signal of the Conversion Control block is set to 1.

Some details are not shown in the block diagram:

- Calibration must be performed to determine the error_correction values (see Calibration) before ADC can do meaningful conversions.
- An input can presample the internal low or high reference voltages to avoid a memory effect that may be present in the comparison due to a previous conversion (see Presampling).
- Monitor ADC status via status flags.
    - See Main Status (MSR) for the FSM state.
    - See Interrupt Status (ISR) to monitor interrupts.
    - See Channel End Of Conversion Flag For Precision Inputs (CEOCFR0) for conversion phase per input.
    - See the following registers to monitor analog watchdogs:
        - Analog Watchdog Out Of Range For Precision Inputs (AWORR0)
        - Analog Watchdog Out Of Range For Standard Inputs (AWORR1)
        - Analog Watchdog Out Of Range For External Inputs (AWORR2)

      See Analog Watchdog Threshold Interrupt Status (WTISR) to monitor analog watchdog interrupts.
    - See Self-Test Status 1 (STSR1) – Self-Test Status 4 (STSR4) to monitor self-test.

### 57.2.2 Features

- Selectable resolution (8-, 10-, 12-, 14-bit). Note that the conversion result is always 15 bits wide, even though the selected resolution is smaller (see CALBISTREG[RESN])
- Conversion data captured in a separate register for each input channel.
- Option to improve accuracy via averaging, which calculates conversion data by averaging the data of up to 32 conversions.

- Conversion triggers:
  - Normal conversion trigger converts a number of input channels, either once per trigger or continuously.
  - Injected conversion trigger interrupts an ongoing normal conversion and converts another set of input channels.
  - BCTU conversion trigger interrupts an ongoing conversion and converts an input channel, in which the input is selected and the conversion is started via the BCTU.
- An analog watchdog optionally monitors conversion data for each input channel and issues an interrupt if the converted data is below or above configurable limits.
- DMA functionality transfers conversion data to other modules.
- Programmable interrupts optionally issue an interrupt when conversion of one or of a set of input channels is finished.
- Self-test functions validate ADC structural integrity during functional operation and generate events with different severities on any finding.
- Conversion clock (AD_clk) control enables the use of ADC in systems with a higher clock frequency by using internal clock dividers.
- Auto turn-off of the conversion clock when ADC is idle.

## 57.3 Functional description

### 57.3.1 Clock

ADC is controlled by one clock signal, the module clock. Internally, the conversion circuit is controlled by the conversion clock, which is derived from the module clock.

The frequency of the conversion clock has to be within the limits defined in the data sheet. If the module clock frequency is higher than the maximum frequency of the conversion clock allowed during the functional conversion or during the calibration (see the chip data sheet), then you must configure the ADC conversion clock divider (MCR[ADCLKSEL]) so that the frequency of the conversion clock is within allowed limits. See Table 269.

**Table 269. Clock configuration for highest conversion speeds**

| Module Clock Frequency f_mc[1] | Configuration during calibration | Configuration during functional conversion |
|---|---|---|
| 40 MHz < f_mc ≤ 80 MHz | MCR[ADCLKSEL] = 1h<br>AMSIO[HSEN] = 0h<br>AMSIO[CMPCTRL0] = 0b | MCR[ADCLKSEL] = 0h<br>AMSIO[HSEN] = 0h<br>AMSIO[CMPCTRL0] = 0b |
| 80 MHz < f_mc ≤ 120 MHz | MCR[ADCLKSEL] = 1h<br>AMSIO[HSEN] = 1h<br>AMSIO[CMPCTRL0] = 1b | MCR[ADCLKSEL] = 0h<br>AMSIO[HSEN] = 1h<br>AMSIO[CMPCTRL0] = 1b |
| 120 MHz < f_mc ≤ 160 MHz | MCR[ADCLKSEL] = 2h<br>AMSIO[HSEN] = 0h<br>AMSIO[CMPCTRL0] = 0b | MCR[ADCLKSEL] = 1h<br>AMSIO[HSEN] = 0h<br>AMSIO[CMPCTRL0] = 0b |
| 160 MHz < f_mc ≤ 240 MHz | MCR[ADCLKSEL] = 2h<br>AMSIO[HSEN] = 1h<br>AMSIO[CMPCTRL0] = 1b | MCR[ADCLKSEL] = 1h<br>AMSIO[HSEN] = 1h<br>AMSIO[CMPCTRL0] = 1b |

*Table continues on the next page...*

Table 269. Clock configuration for highest conversion speeds (continued)

| Module Clock Frequency f_mc[1] | Configuration during calibration | Configuration during functional conversion |
|---|---|---|
| 240 MHz < f_mc ≤ 320 MHz | MCR[ADCLKSEL] = 3h<br>AMSIO[HSEN] = 0h<br>AMSIO[CMPCTRL0] = 0b | MCR[ADCLKSEL] = 2h<br>AMSIO[HSEN] = 0h<br>AMSIO[CMPCTRL0] = 0b |
| 320 MHz < f_mc ≤ 480 MHz | MCR[ADCLKSEL] = 3h<br>AMSIO[HSEN] = 1h<br>AMSIO[CMPCTRL0] = 1b | MCR[ADCLKSEL] = 2h<br>AMSIO[HSEN] = 1h<br>AMSIO[CMPCTRL0] = 1b |

1. Lower resulting frequencies are allowed—see the chip data sheet.

When ADC is not converting (state machine is in Idle state), the conversion clock can be turned off (MCR[ACKO]).

### 57.3.2 Modes of operation

ADC is always in Functional mode. No other mode selection exists.

- Put ADC into Power Down state by writing 1 to MCR[PWDN] to reduce power consumption.
- Gate the clock signal by writing 1 to MCR[ACKO] when ADC is in Idle state.

### 57.3.3 Conversion

ADC measures the voltage of a signal—it converts an analog input value into a digital representation. When you read the conversion result, you must divide this conversion result by the number of codes (defined by selected conversion resolution in CALBISTREG[RESN]), and multiply it with the reference voltage value. This calculation provides the voltage value of the converted signal.

To convert an analog input voltage into its digital representation, ADC:

1. Receives a signal, referred to as a trigger, indicating it is to perform a conversion.
2. Captures the voltage of a signal via internal capacitances in a process referred to as sampling.
3. Compares the voltage to the reference voltage, using the SAR algorithm.

Before a conversion can be performed, you must select:

- The trigger source(s) to use (on the chip level)
- The input channel(s) to convert (NCMR0–2, JCMR0–2)

Additionally, you can tailor the conversion to meet your needs by changing the default configuration. For example, you can:

- Configure ADC to convert the selected input channel(s) only once or continuously in a loop (MCR[MODE]).
- Specify the duration that ADC samples the input signal (CTR0/1/2).
- Specify the resolution of the conversion result by selecting the number of bits (CALBISTREG[RESN]).
- Enable an analog watchdog to generate an interrupt when a conversion result falls outside a specified range (see Analog watchdog functions).
- Enable the DMA interface to send requests and receive acknowledgments (see DMA functionality).
- Check the operational integrity of ADC by running a self-test (see Self-test).

After power-up or a functional reset, ADC remains in Power Down state until enabled by writing 0 to MCR[PWDN]. After the system is powered on, you must run the calibration (see Calibration), before you can do a meaningful conversion with ADC.

Some configuration fields (listed below) are writeable only in Power Down state. If you intend to write to them, you must do so before exiting Power Down state.

- Select the conversion clock frequency to be within the allowed limits for a conversion.

### 57.3.4 Normal trigger

Select the input channels to convert by configuring the fields in NCMR0, NCMR1, and NCMR2.

These registers must be programmed before the start of conversion. You cannot reconfigure them until the conversion of all selected channels is complete. The sequence is always to convert the selected input channels in this order:

1. Start from the lowest input channel of the precision input channels.
2. Proceed through the standard input channels in ascending order.
3. End with the highest input channel of the external input channels.

#### 57.3.4.1 Starting conversions in normal conversion mode

After programming the required fields of the following registers, you can start a conversion by writing 1 to MCR[NSTART]:

- Normal Conversion Enable For Precision Inputs (NCMR0)
- Normal Conversion Enable For Standard Inputs (NCMR1)
- Normal Conversion Enable For External Inputs (NCMR2)
- Main Configuration (MCR)

A hardware trigger can also start a normal conversion.

- If the external trigger is enabled (MCR[TRGEN] = 1), an external trigger enable is detected to start the conversion. The enable is checked only during start of conversion.
- Detection of an active edge defined by MCR[EDGE] (rising = 1, falling = 0) on an external trigger transitions MCR[NSTART] to 1 and starts the normal conversion.
- When MCR[TRGEN] = 1 (for example, when external trigger is enabled), conversion can be started by software.
- Any external trigger or software initiated conversion in normal conversion mode is ignored during any ongoing conversion chain.

**NOTE**

To have reliable operation there must be a gap of 3 clock cycles between configuration of NCMRx and the start of a conversion (by writing 1 to MCR[NSTART], or arrival of an external or internal hardware start trigger). ADC calculates the number of channels to be converted during this period.

#### 57.3.4.2 Operation modes in normal conversions

Two operational modes are available during normal conversion:

- One-Shot mode
- Scan mode

For normal conversion, select the mode via MCR[MODE]. The first phase of the conversion process involves sampling the analog channel. The next phase is the conversion phase, curing which the sampled analog value is converted to digital as shown in the following figure.

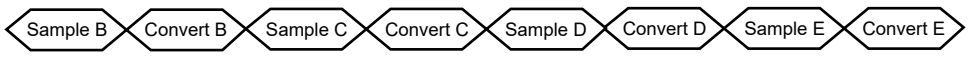



Figure 221. Normal conversion flow

In One-Shot mode (MCR[MODE] = 0), a sequential conversion specified in the NCMR*n* mask registers is performed only once. At the end of each conversion, the digital result of the conversion is stored in the corresponding data register (CDR*n*).

For example: Channels A→B→C→D→E→F→G→H are present in a device where channels B→D→E are to be converted in One-Shot mode. Conversion begins with channel B, followed by conversion of channels D and E. At the end of conversion of channel E, the scanning of channels stops.

MSR[NSTART] automatically transitions to 1 when a normal conversion starts. At the same time, MCR[NSTART] is reset to zero by hardware, enabling software to program a new start of conversion in advance. In this case, the new requested conversion starts after completion of the running/current conversion. However, for correct functioning of the device, you should program MCR[NSTART] for the new conversion only after completion of the current conversion. The application can wait for an ECH interrupt prior to writing 1 to MCR[NSTART] again.

If an external trigger starts the conversion chain, ADC does not observe the trigger input until the conversion chain is finished. After the chain is finished, the next chain can be triggered by another hardware trigger edge.

In Scan mode operation (MCR[MODE]=1), a sequential conversion of N channels specified in the NCMR*n* registers is continuously performed. At the end of each conversion, the result is stored in the corresponding data register, as in One-Shot mode.

MSR[NSTART] automatically transitions to 1 when a normal conversion starts. Unlike One-Shot mode, MCR[NSTART] is not reset to 0 in Scan mode. It can be reset by software when you must exit Scan mode. In that case, ADC completes the current chain and after the last conversion, it resets MCR[NSTART] to 0.

For example: Channels A→B→C→D→E→F→G→H are present in the device where channels B→D→E are to be converted in Scan mode. MCR[MODE] = 1 selects Scan mode operation. Conversion starts from channel B followed by conversion of channels D→E. After conversion of channel E, the scanning of channel B starts followed by conversion of the channels D→E. This sequence repeats itself until MCR[NSTART] is reset to 0 by software.

If an external trigger starts a conversion, then MCR[NSTART] does not transition to 1. This scan chain can be stopped by writing 0 to MCR[NSTART]. The conversion stops when the ongoing chain is finished. Consequently, after it starts, the only way to stop Scan mode conversion is to write 0 to MCR[MODE].

**End of conversion**

In both modes, at the end of each conversion, an End of Conversion (EOC) interrupt is issued if enabled by the corresponding mask fields in CIMR*n* and IMR).

After conversion of all selected channels in NCMR*n* is complete, the conversion operation is considered finished. Then ISR[ECH] transitions to 1 and the End of Chain (ECH) interrupt is issued (if enabled in IMR[MSKECH]).

The corresponding channel field in the CEOCFR*n* register(s) is updated to indicate that data is available on data register CDR*n* of the respective channel.

If there is no channel selected in NCMR*n* and there is a start-of-conversion trigger, then ISR[ECH] is set to 1 and the End of Chain (ECH) interrupt is immediately issued (if enabled).

### 57.3.5 Injected trigger

An injected trigger enables you to convert a set of input channels although the standard trigger has already started conversion of another set of input channels. Each channel can be individually enabled by writing 1 to the corresponding field in JCMR0, JCMR1, or JCMR2.

The set of input channels of the injected trigger is converted only once. A continuous conversion of a set of input channels in a loop is only possible when started by a standard trigger. When an injected trigger is received, an ongoing conversion started by a standard trigger is interrupted. After the set of input channels of the injected trigger is converted once, the interrupted conversion of the set of input channels of the standard trigger resumes from the input channel that was interrupted, as shown in the following figure.

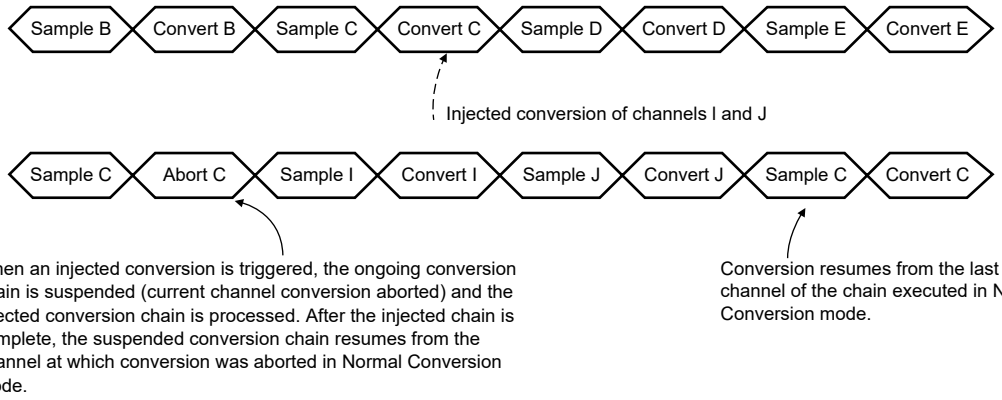



**Figure 222. Injected trigger conversion sequence**

#### 57.3.5.1 Starting conversions in injected conversion mode

An injected conversion chain can be started:

- By software: When external triggering is disabled (MCR[JTRGEN] = 0), you can write 1 to MCR[JSTART], which causes the current conversion chain processed in Normal Conversion mode to be suspended and the injected chain to be processed.
- By external trigger: When external triggering is enabled (MCR[JTRGEN] = 1), a programmed event (rising/falling edge depending on the value of MCR[JEDGE]) on the injection external input starts the injected conversion.

MCR[JSTART] automatically transitions to 1 when the injected conversion chain starts. At the same time MCR[JSTART] is reset to 0, enabling software to program a new start of conversion in advance. In that case the new requested conversion starts after the running conversion completes.

At the end of each conversion, an End Of Injected Conversion (JEOC) interrupt is issued, if enabled by the corresponding mask field in Interrupt Mask (IMR). At the end of a chain, an End Of Injected Chain (JECH) interrupt is issued, if enabled by the corresponding mask field. Additionally, ADC writes 1 to ISR[JECH].

**NOTE**

If the content of all the Injected Conversion mask registers (JCMR*n*) is zero, that is, no channel is selected, the JECH interrupt is immediately issued after the start of conversion.

To have reliable ADC operation, there must be a gap of 3 cycles between configuration of JCMRx and starting of conversion (writing 1 to MCR[JSTART] or arrival of either an external or internal hardware start trigger). ADC calculates the total channel numbers to be converted during this period.

The corresponding channel bit in the CEOCFR*n* register is updated to indicate data is available in the channel's conversion data register, CDR*n*.

After starting, an injected chain conversion cannot be interrupted.

### 57.3.6 BCTU interface

The BCTU interface enhances ADC's injected conversion capability. It contains control inputs to select the channels to be converted from the appropriate event configuration register. Figure 223 shows the interface.

The BCTU generates a trigger (bctu_trigger) and a channel number (bctu_numchannel) to be converted. A single channel is converted for each request. After performing the conversion, ADC returns the result on the bctu_dataout bus together with two output signals named bctu_nextcmd and bctu_push. The assertion of signal bctu_nextcmd means ADC is ready to accept the next trigger from BCTU. The bctu_push signal is asserted at the end of conversion, meaning that conversion is finished and the conversion result available at output bctu_dataout is valid.

The conversion result is also saved in the corresponding channel's data register and is compared with analog watchdog thresholds if requested.

The signals bctu_trigger, bctu_nextcmd and bctu_push are all of type single-cycle active-high-pulse in the ADC clock domain. The channel number provided from BCTU must be valid when bctu trigger is active-high. The result data from ADC is valid with bctu_push high.

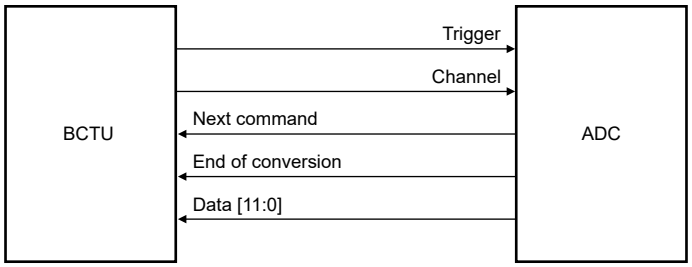



**Figure 223. BCTU interface**

The BCTU interface has two modes of operation:

- Trigger
- Control

To enable the BCTU interface, write 1 to MCR[BCTUEN]. The operating mode (Trigger or Control) can be fixed or programmable.

### 57.3.6.1 BCTU Trigger mode

In Trigger mode, normal and injected conversions can also be performed. All types of conversions can be initiated in this mode. The priorities among the three types of conversions are discussed below.

When a trigger is received, the channel number is taken as an injected channel value and the triggered injected conversion starts. MSR[BCTUSTART] transitions to 1 automatically at this point and it is also automatically reset to 0 when the triggered injected conversion is completed.

BCTU conversions must be requested only after successful ADC calibration. The application must prevent any BCTU trigger during calibration.

If a BCTU trigger is received during an ongoing injected conversion, the injected chain is aborted immediately and only the BCTU-triggered injected conversion proceeds. Additionally, MSR[JSTART] is reset to zero. The abort of an injected conversion is indicated by MSR[JABORT].

If a BCTU trigger is received during an ongoing normal conversion, the ongoing normal channel conversion is suspended and the BCTU-triggered injected conversion is processed. Normal conversion resumes from the suspended channel after completion of the BCTU-triggered conversion.

If a normal conversion is requested during BCTU conversion (MSR[BCTUSTART] = 1), the normal conversion starts after the BCTU conversion completes (MSR[BCTUSTART] is reset to zero).

Any injected conversion is discarded if requested during BCTU conversion and MCR[JSTART] is immediately reset to zero.

### 57.3.6.2 BCTU Control mode

In BCTU control mode (MCR[BCTU_MODE] = 0), only the BCTU can start a conversion. All other trigger sources are ignored.

Along with BCTU trigger, the information provided with signal bctu_numchannel is taken as the channel number for injected channel and BCTU triggered conversion starts. MSR[BCTUSTART] transitions to 1 automatically at the start of the conversion and it remains 1 unless BCTU is disabled by writing 0 to MCR[BCTUEN].

The conversion must be requested (generating bctu_trigger) when calibration has finished successfully. If a BCTU trigger is received during calibration execution, calibration is stopped immediately in order to satisfy the BCTU request. Calibration fails in this case.

### 57.3.7 Aborting a conversion

Two abort functions are provided:

- Abort of a single channel
- Abort of a chain

To abort an ongoing conversion, write 1 to MCR[ABORT]. The current conversion aborts and conversion of the next channel of the chain begins immediately.

Depending on the current state of a conversion, an Abort action may take 1 to 4 cycles of the bus clock. At the start of any conversion, and at the end of a particular conversion when internal state counters are changing, an abort action is delayed by a maximum of 3 cycles to put all states in a stable condition.

To ensure that the abort action completes, do not change MCR[ABORT] for the next 3 cycles.

Avoid writing 1 to MCR[ABORT] along with, and within 3 cycles of, writing 1 to MCR[NSTART] or MCR[JSTART].

During an abort:

- MCR[NSTART] and MCR[JSTART] remain 1, if not in the last channel of the chain.
- MCR[ABORT] is reset to 0 when the channel is aborted.
- The EOC interrupt corresponding to the aborted channel is not generated. This behavior applies to normal and injected conversions. If the last channel of a chain is aborted, the end of chain is reported by generating an ECH interrupt.

It is possible to abort the current chain of conversions by writing 1 to MCR[ABORTCHAIN]. In this case, the behavior of ADC depends on the value of MCR[MODE] (One-Shot/Scan operation modes). If Scan operation mode is disabled, MCR[NSTART] is automatically reset to 0 along with MCR[ABORTCHAIN]. Otherwise, in Scan Operation mode, a new chain is started. The EOC interrupt of the current aborted conversion is not generated but an ECH interrupt is generated to signal the end of the chain.

**NOTE**

For a single channel chain, an ECH interrupt is not generated for an abort or abort chain.

When an abort chain is requested while an injected chain is running over a suspended normal chain, both the injected and normal chains are aborted, and both MCR[NSTART] and MCR[JSTART] are reset to 0.

### 57.3.8 Configuring ADC clock divider and sample time settings

You can scale the AD_clk frequency via MCR[ADCLKSEL]. ADCLKSEL can only be written in Power Down state (MCR[PWDN] = 1). Depending on the frequency of the module clock, there might be different settings necessary for functional conversion and for calibration. See the chip data sheet for ADC electrical characteristics.

Three conversion timing registers are used (CTR*n*) to support different sampling times for the different types of channels. An exception exists for the temperature sensor channel, which always uses the Conversion Timing For Standard Inputs (CTR1) value. See the register description for details.

### 57.3.9 Presampling

The presampling feature enables pre-charge or discharge of the ADC internal sample capacitor node to a defined level before sampling of the selected analog input channel starts. This is useful for resetting information (history effect/offset) from the most recent conversion. During presampling, ADC samples the internally generated voltage. During the sampling phase, ADC samples analog input coming from pads.

Enable presampling on a per-channel basis by setting the corresponding fields in the PSR*n* registers to one.

After enabling presampling for a channel, the normal sequence of operation for the channel is:

1. Presampling
2. Sampling
3. Evaluation

You can bypass sampling of all channels by writing 1 to PSCR[PRECONV]. When sampling of a channel is bypassed, the sampled and stored internal voltage applied during the presampling phase is converted (Figure 224). See Conversion time for the timing equation for conversion.

Presampling does not apply for self-test channels.

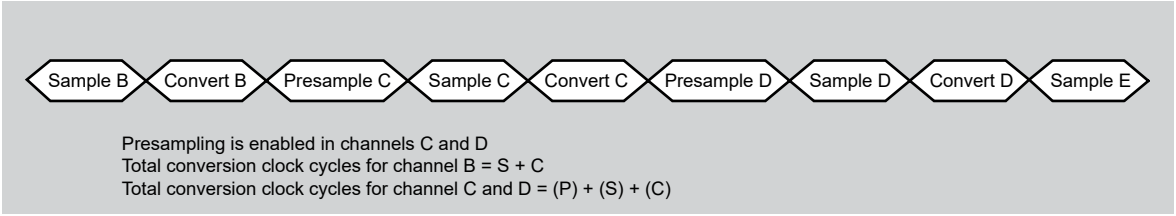



**Figure 224. Presampling sequence**

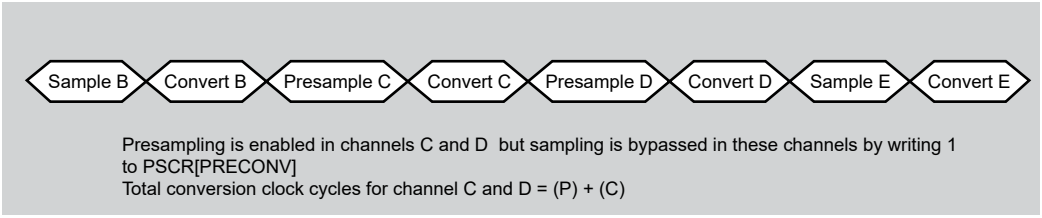



**Figure 225. Presampling sequence with PRECONV=1**

### 57.3.9.1 Enabling presampling per channel

Enable presampling for specific channels by programming Presampling Control (PSCR). You can select between two internally generated voltages by configuring the PREVAL*n* fields in Presampling Control (PSCR) to the appropriate values, as given in the following table.

**Table 270. Presampling voltage selection via PREVAL*n***

| PREVAL*n* | Presampling voltage |
|---|---|
| 0 | Presample voltage – 0: VREFL |
| 1 | Presample voltage – 1: VREFH |

The presampling configuration field(s) (PSCR[PREVAL0], PSCR[PREVAL1], PSCR[PREVAL2]) are located in Presampling Control (PSCR). These fields enable selection of the presampling channel.

The temperature sensor channel has an exception. This channel is mapped with a fixed channel depending on device configuration, but it uses the presampling configuration defined for channel group 2. So, when the temperature sensor channel is selected, the presampling voltage defined by PSCR[PREVAL1] is selected. This voltage might be different from the one selected for channel group 1 via PSCR[PREVAL0].

## 57.3.10 Analog watchdog functions

When enabled, an analog watchdog monitors conversion results for an associated channel and reports an issue if it determines a result is outside customer-defined limits (as shown in Analog watchdog configuration (up to 16 watchdogs) ). These limits are specified by an upper and a lower threshold value named THRH and THRL respectively.

After the conversion of the selected channel a comparison is performed between the converted value and the threshold values. If the converted value is outside that threshold value, then ADC generates a corresponding threshold violation interrupt.

The comparison result is stored as HAWIFx and LAWIFx fields in Analog Watchdog Threshold Interrupt Status (WTISR), as explained in the following table. Depending on the values of WTIMR[LAWIFENx] and WTIMR[HDWIFENx] mask fields, an interrupt is generated on a threshold violation.

**Table 271. Values of HAWIFx and LAWIFx**

| HAWIFx | LAWIFx | Conversion result |
|---|---|---|
| 1 | 0 | Conversion result > THRH |
| 0 | 1 | Conversion result < THRL |
| 0 | 0 | THRH ≥ conversion result ≥ THRL |

Depending on the chip's factory settings, up to 16 analog watchdogs are available. See the chip-specific configuration information for availability.

There are two types of mutually exclusive settings for analog watchdogs. Both types are described in the following sections. The availability depends on device configuration. See the chip-specific configuration information for availability.

Different capabilities are available with the two settings, as described below.

### 57.3.10.1 Channel analog watchdog select registers

ADC configuration includes selecting the analog watchdog threshold register (THRHLR*n*) that provides limits to monitor inputs of a given type. You select the THRHLR*n* using CWSELR*a*l*n* registers.

**Table 272. CWSELR*a*l*n* register naming**

| Index | Description |
|---|---|
| *a* | ADC input type associated with a CWSELR*a*l*n* register.<br>• P = Precision inputs<br>• S = Standard inputs<br>• E = External inputs |
| *n* | 0-indexed register number. |

### 57.3.10.2 Analog watchdog configuration (up to 16 watchdogs)

ADC can have up to 16 watchdogs in the factory configuration. (See the chip-specific configuration information for availability.) For each watchdog there is one threshold value register, THRHLR*n*, that contains the high and low thresholds.

You can independently enable the analog watchdog for each channel by programming the appropriate CWENR*n* register.

The threshold values for each channel can be selected independently from a maximum of 16 threshold registers (THRHLR*n*) using the corresponding WSEL_CH*n* field in the appropriate CWSELR*a*l*n*, where *a* indicates the input type:

- P = precision
- S = standard
- E = external

Each CWSELR*a*l*n* register is associated with 8 consecutive channels. CWSELR*a*l0 holds 8 WSEL_CH fields for channels 0 to 7. CWSELR*a*l1 is for channels 8 to 15 and so on. The availability of fields and registers depends on device configuration. See the chip-specific configuration information for availability.

If the conversion result of a selected channel is outside the range specified by threshold values, then the corresponding field in the Analog Watchdog Out of Range register (AWORR*n*) transitions to 1. For example, if channel 7 is to be monitored with the threshold values in THRHLR3, then CWSELR*a*l0[WSEL_CH7] must be programmed with 3. Enabling the watchdog is done by writing 1 to the corresponding field for channel 7 in CWENR0.

In this configuration, a set of threshold values (THRHLR*n*) can be linked to several ADC channels.

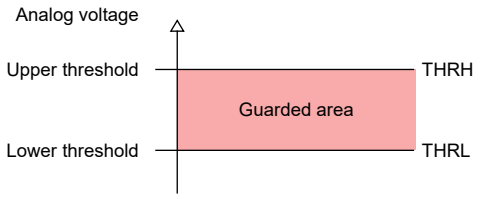



**Figure 226. Guarded area**

**NOTE**

If the higher threshold of the analog watchdog is programmed below the lower threshold and the conversion result is below the lower threshold, then the WDG*n*L interrupt for the low threshold violation transitions to 1. If the conversion result is greater than the lower threshold (and therefore also greater than the higher threshold), then the interrupt WDG*n*H for high threshold violation is generated. Thus you should take care to avoid that situation, as it could lead to misinterpretation of the watchdog interrupts.

### 57.3.11 DMA functionality

Conversion result data from any channel can be transferred from a register to system memory via Direct Memory Access (DMA). DMA transfers are enabled by writing 1 to DMAE[DMAEN]. After being enabled, the on-chip DMA controller can receive a DMA request after the conversion of each channel by writing 1 to the respective masking field in:

- DMA Request Enable For Precision Inputs (DMAR0)
- DMA Request Enable For Standard Inputs (DMAR1)
- DMA Request Enable For External Inputs (DMAR2)

DMA masking registers must be programmed before starting any conversion.

A DMA request to the DMA controller can be cleared at different times in two modes:

- Mode-1: Clearing of DMA request on acknowledgment from DMA controller (DMAE[DCLR] = 0)
- Mode-2: Clearing of DMA request on read to data registers (DMAE[DCLR] = 1)

The figures below show the operation of DMA in two modes (cycle counts are typical values).

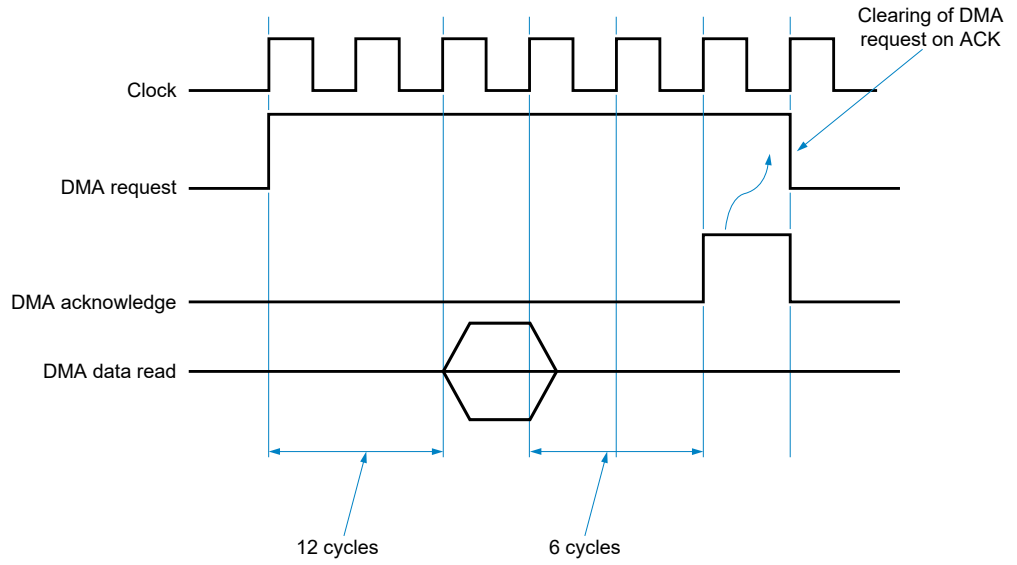



Figure 227. DMA operation Mode-1 (DMAE[DCLR] = 0)

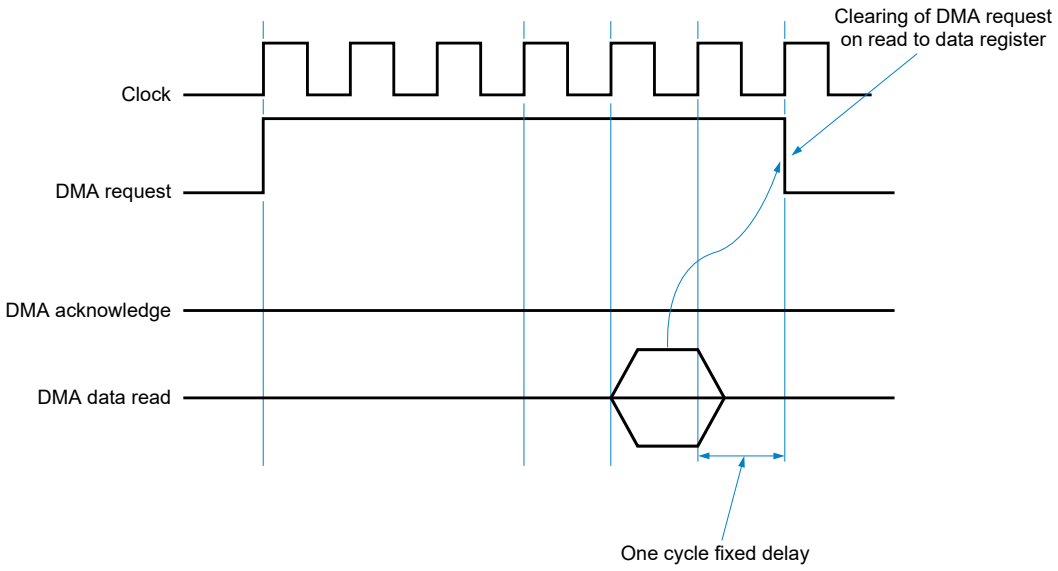



Figure 228. DMA operation Mode-2 (DMAE[DCLR] = 1)

### 57.3.12 Interrupts

ADC generates the following maskable interrupts:

- EOC (End of Conversion)
- ECH (End of Chain)
- JEOC (End of Injected Conversion)
- JECH (End of Injected Chain)
- EOBCTU (End of BCTU-triggered Conversion)
- LAWIFx and HDWIFx (Watchdog threshold)

- Self-testing interrupts

Interrupts are generated during the conversion process to signal events such as End Of Conversion (EOC), and so on, as described in Interrupt Status (ISR). ISR and another register, Interrupt Mask (IMR), are provided to check and enable the interrupt request to an external interrupt controller.

Interrupts can be individually enabled on a per-channel basis by programming EOC Interrupt Enable For Precision Inputs (CIMR0).

CEOCFR*n* are used for pending End of Conversion interrupts, with one field per channel to indicate completed conversions on a per-channel basis.

Interrupts generated due to the analog watchdogs are managed by two 32-bit registers, Analog Watchdog Threshold Interrupt Status (WTISR) and Analog Watchdog Threshold Interrupt Enable (WTIMR), to check and enable the interrupt request. A watchdog interrupt causes two corresponding fields to transition to 1 for each channel monitored. One field is in WTISR[WDG*n*H] and another in WTISR[WDG*n*L].

Generated events are logged into status registers and used to generate interrupts if enabled in corresponding Mask registers.

There are three different interrupt outputs. The interrupt structure is shown in Figure 229.

Each interrupt is generated by combining a group of events.

The following interrupts are combined (logic OR) into one interrupt output.

- EOC
- ECH
- EOBCTU
- JEOC
- JECH
- End of self-test algorithm (WDG_EOA_S and WDG_EOA_C)

Interrupts of all HDWIFx and LAWIFx are combined on the second interrupt output.

Interrupts of all self-test errors (watchdog threshold, sequence, and timer violations) in Self-Test Status 1 (STSR1) are combined on the third interrupt output. The errors are:

- WDSERR
- WDTERR
- ERR_S0
- ERR_S1
- ERR_S2
- ERR_C

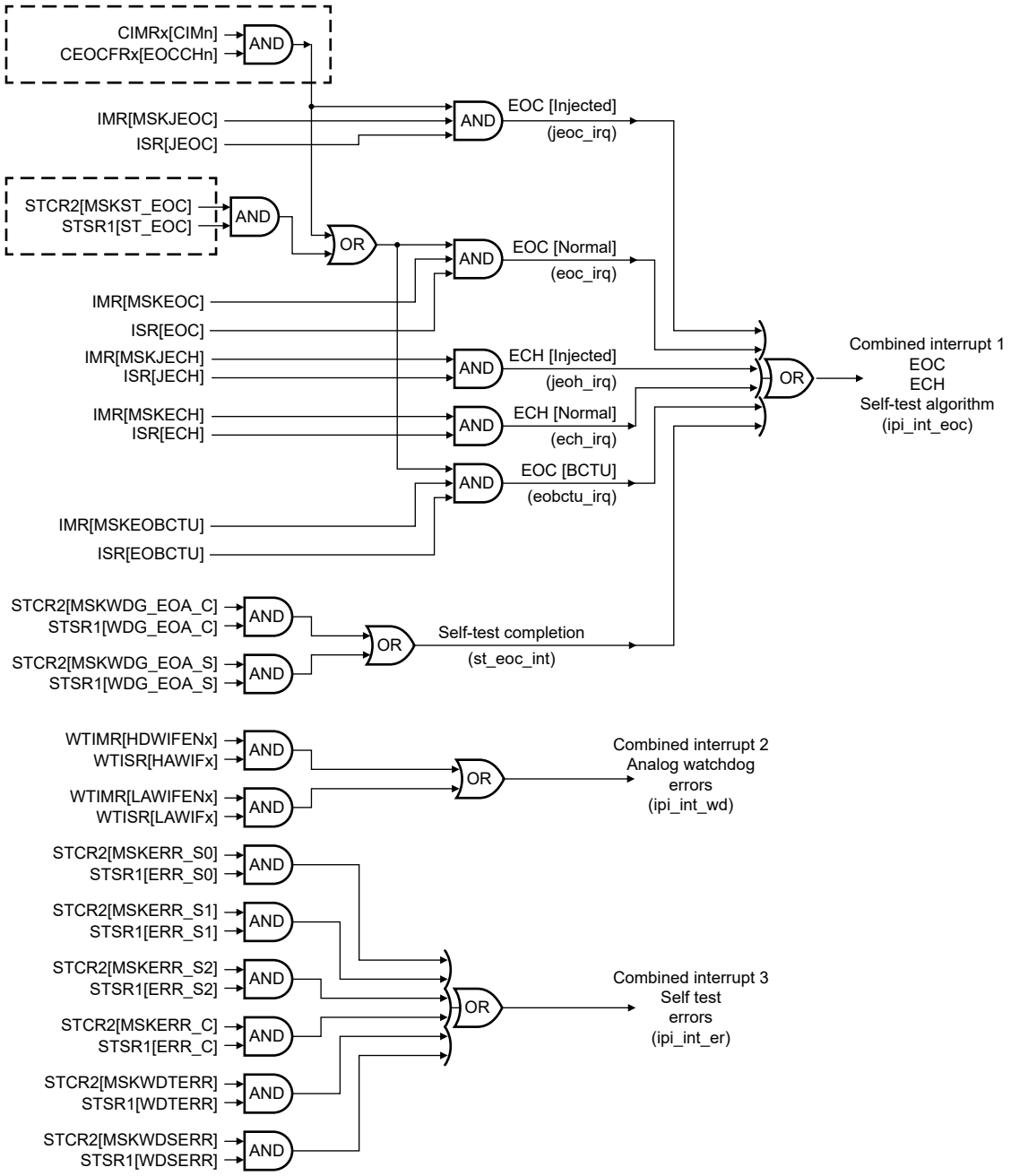



**Figure 229. Interrupt structure of ADC**

Interrupt Status (ISR) Indicates the interrupt pending request status. Write 1 to the corresponding status field to clear the pending interrupt flag (at this write operation all the other fields in Interrupt Status (ISR) must be maintained at 0).

ISR checks for three types of interrupts.

The first type, Interrupt 1, relates to the following:

- End of normal or injected conversion (EOC or JEOC)
- End of normal or injected conversion chain (ECH or JECH)
- End of algorithm C or algorithm S self-test (WD_EOA_C or WD_EOA_S)

Perform the following actions for this type of interrupt:

1. Read Interrupt Status (ISR) to determine whether the event is an EOC, JEOC, ECH, JECH, or EOBCTU event.
2. Read Self-Test Status 1 (STSR1) to identify the end of self-test algorithm events (WDG_EOA_S or WDG_EOA_C), if present.
3. If the interrupt is an EOC or JEOC event:
   - Clear the respective channel pending field in the appropriate CEOCFR*n* register for an EOC by writing a 1 to it.
   - Clear ISR[EOC] or ISRJEOC] by writing a 1 to it.
4. If the interrupt is an ECH or JECH, clear ISR[ECH] or ISR[JECH] by writing 1 to it.
5. If the interrupt is EOBCTU, clear ISR[EOBCTU] by writing 1 to it.
6. If the interrupt is for self-test end of algorithm, clear STSR1[WDG_EOA_S] or STSR1[WDG_EOA_C] by writing 1 to it.

The second type, Interrupt 2, relates to analog watchdog errors. Perform the following actions for this type of interrupt:

1. Read Analog Watchdog Threshold Interrupt Status (WTISR) to determine which watchdog triggered the interrupt.
2. Read the AWORRx register(s) to determine which channel's data caused the interrupt (if present).
3. Clear the respective field of the WTISR[WDG*n*H/WDG*n*L] register, where *n* = watchdog number, by writing 1 to it.
4. Clear the respective field of the AWORR*n*[AWOR_CH*n*] register, where x = register, n = channel, by writing 1 to it.

The third type, Interrupt 3, relates to self-test errors. Perform the following actions for this type of interrupt:

1. Read Self-Test Status 1 (STSR1) to determine the type of error encountered in self-test (WDSERR, WDTERR, ERR_S0, ERR_S1, ERR_S2, ERR_C, and so on) if present.
2. Clear the respective field in Self-Test Status 1 (STSR1) by writing 1 to it.

The connectivity of interrupt lines at the chip level (external to ADC) determines whether software must read all the registers for all three interrupts or any one or combination of any two.

### 57.3.13 External decode signals delay

The ADC provides three external decode signals which it uses to select one channel out of eight in the external analog multiplexers. There can be a maximum of four such multiplexers used to connect the 32 external channels. The ADC automatically sets the decode signals to control these external analog multiplexers, based on the current channel selected for conversion. Delay Start Of Data Conversion (DSDR) is provided to take the switching time of the external analog multiplexer into account. It enables you to program the delay between the decoding signal selection and the actual start of conversion.

If presampling is enabled, the DSD delay starts after the presample phase (presampling time is the same duration as sample time).

### 57.3.14 Power Down state

The analog part, along with the ADC controller, can be put in Power Down state by writing 1 to MCR[PWDN]. This shuts down ADC and stops the clock to the ADC controller. After release of reset, MCR[PWDN] remains 1, so the ADC analog module is kept in Power Down state by default. You must exit this state before starting any operation, by writing 0 to MCR[PWDN]. In Power Down state, no conversion can be started. If a BCTU trigger pulse is received during this state, it is discarded. You must ensure that a BCTU trigger is not initiated to ADC before writing 0 to MCR[PWDN], as it may cause the BCTU module to be stuck in a wait state.

You can set MCR[PWDN] by writing 1 to it at any time. If a conversion is ongoing, ADC does not move into Power Down state immediately. ADC enters Power Down state only after completion of the ongoing conversion. During scan operation, the ongoing operation should be aborted manually by writing 0 to MCR[NSTART] before or after writing 1 to MCR[PWDN].

ADC Power Down state status is indicated by MSR[ADCSTATUS] when ADC enters into Power Down state.

If the BCTU is enabled and MSR[BCTUSTART] = 1, then you cannot write 1 to PWDN. When BCTU Trigger mode is enabled, the application must wait for the end of conversion (MSR[BCTUSTART] is automatically reset to 0). When BCTU Control mode is enabled, before entering Power Down state the application must write 0 to BCTUEN also.

If Power Down state is entered by writing 1 to MCR[PWDN], the process running before entry into Power Down state must be restarted manually (by writing 1 to the appropriate START field in Main Configuration (MCR)) after exiting Power Down state.

**NOTE**

Writing 0 to MCR[PWDN] and writing 1 to MCR[NSTART] or MCR[JSTART] during the same cycle is not supported.

### 57.3.15 Low Power mode support

For low-power operation, ADC must be in Idle state and then put into Power Down state to ensure proper signal condition at the analog boundary.

From any of the states listed below, follow the sequence of steps shown to put ADC into Power Down state.

1. Idle condition
    a. Verify that ADC is in Idle state (MSR[ADCSTATUS] = 000b).
    b. Write 1 to MCR[PWDN], to shut off the clock to the hard macro with the proper state of signals required for Low Power mode.
    c. Verify that ADC is in Power Down state (MSR[ADCSTATUS] = 001b).
2. During calibration

    **NOTE**

    Writing 1 to MCR[PWDN] during calibration is prohibited. After calibration is started it should be allowed to finish normally. Alternatively, it can be terminated by starting a normal conversion before writing 1 to MCR[PWDN].

    a. Check the current ADC state either via MSR[ADCSTATUS] (000b = Idle, 011b = Calibration), or via CALBISTREG[C_T_BUSY] (1 = calibration is in progress).
    b. When ADC is in Idle state, write 1 to MCR[PWDN] to shut off the clock to the hard macro with the proper state of signals required for Low Power mode.
    c. Verify that ADC is in Power Down state (MSR[ADCSTATUS] = 001b).
3. During conversion (normal [one-shot]/Injected/BCTU):
    a. Write 1 to MCR[PWDN].
    b. ADC enters Power Down mode after completing the current conversion chain (for normal and injected conversions) or the conversion for conversion triggered by BCTU.
    c. Check the ADC state via MSR[ADCSTATUS] (000b = Idle state, 001b = Power Down state).
4. During Scan mode operation of normal conversion

    **NOTE**

    Writing 1 to MCR[PWDN] during scan mode operation has no effect other than to prevent software from starting a new normal or injected conversion. It does not stop a conversion in progress and cannot put ADC into Power Down state.

    a. Software must write 0 to MCR[NSTART] to stop the ongoing scan conversion before (or after) writing 1 to MCR[PWDN]. This stops conversion at the current chain boundary.
    b. ADC enters Power Down mode after coming to an Idle state after MCR[PWDN] is written with 1. This shuts the clock off to the hard macro and puts analog inputs in the proper state required for Low Power mode.
    c. Verify that ADC is in Power Down state (MSR[ADCSTATUS] = 001b).

### 57.3.16 Auto Clock Off mode

To reduce power consumption during Idle state (without going into Power Down state), an auto-clock-off feature can be enabled by writing 1 to MCR[ACKO]. When enabled, the internal ADC operating clock (AD_Clk) is automatically switched off during an idle period (for example, no conversion is programmed).

### 57.3.17 Calibration and self-test

#### 57.3.17.1 Calibration

ADC must be calibrated before it can do meaningful conversions. The application must prevent any start of conversion before the ongoing calibration has finished successfully. No conversion trigger may be received during calibration.

During calibration, a known reference voltage is sampled and converted under controlled conditions to determine the correction values (calibration values) for offset, gain, and capacitor mismatch.

These calibration values (except gain calibration) are used in a post-processing step after conversion, reducing or eliminating the various error contribution effects. The gain calibration is used during sample phase to define the additional charge to be loaded to compensate for gain failure.

You must run calibration after every power-up reset (check device-specific reset connectivity for appropriate reset applied). You must also run calibration if it is indicated by the self-test (see Self-test).

Configuration and start of calibration is done by programming the Control And Calibration Status (CALBISTREG). During calibration, internal hardware averaging function should be enabled. Maximum averaging is recommended, and 16 should be the minimum.

Calibration status is captured by a status flag (CALBISTREG[TEST_FAIL]) to indicate any fault detected during calibration execution or if calibration was prematurely aborted. Calibration is aborted if any normal conversion is initiated during the calibration execution, which is called premature termination. (Flag TEST_FAIL = 1 indicates a calibration fail, for example when a calibration value is out of internally defined limits.)

To execute the calibration algorithm, perform the following steps:

1. Select the conversion clock frequency to be within the allowed limits for the calibration.
2. Bring ADC from Power Down state to active conversion (program MCR[PWDN] = 0b).
3. Configure the Control And Calibration Status (CALBISTREG). The default values are set for maximum accuracy (recommended).
4. Disable the error correction for the smaller capacitances in the CDAC of the ADC (write 0h to CAL2[ENX]).
5. Start calibration (program CALBISTREG[TEST_EN] = 1b), and calibration starts immediately.
6. Poll the status of CALBISTREG[C_T_BUSY] for 0 (wait until it transitions to 0).
7. Check CALBISTREG[TEST_FAIL] to determine the final status. If 1, then calibration failed.
8. Check the status of MSR[CALIBRTD]. If calibration was successful this field is 1.

#### 57.3.17.2 Self-test

For safety applications, it is important to verify correct operation at regular intervals. ADC has a self-test feature for this purpose. When self-test is enabled, ADC automatically checks its components and flags errors, if any are found.

Tests can be enabled to check the reference (VrefH) and calibrated values.

The following test algorithms have been implemented:

- Supply self-test (algorithm S): It includes the conversion of the band gap and VREF voltages. It includes a sequence of num_supply_steps test conversions (steps). The supply test conversions must be an atomic operation (that is, all supply algorithm conversions must be performed one after another with no functional conversions in between).

- Capacitive self-test (algorithm C): It includes a sequence of test steps per definition of algorithm C (see Table 273) which executes the capacitive matrix of the CDAC used for sampling and conversion.

Individual steps can take up to 1 µs at 80 MHz ADC clock frequency.

**Table 273. Self-test steps**

| STCR3[ALG] | STCR3 [step](per algorithm definition) | Description | Outcome | Comment |
|---|---|---|---|---|
| 00 (algorithm S) | 0 | Supply self-test (band gap voltage) | Measures the internal band gap voltage | |
| | 1 | Supply self-test (reference voltage high) | Measures ADC high reference voltage | |
| | 2 | Supply self-test (reference voltage high) | Measures ADC high reference voltage | |
| 01 (Reserved) | Reserved | | | |
| 10 (algorithm C) | 0 – 11 | Capacitive self-test. One of the calibration steps is being run. | The difference/error of the individual offset value from the previously calibrated value is being returned as an ADC result that is compared with the programmed value in the self-test analog watchdog register(s) to detect any fault. | The ideal result value is expected to be 0. If the returned value is higher, it indicates a runtime fault or accuracy shift. The error is indicated in the status register and can be used to generate an interrupt also. It that indicates ADC should be recalibrated to check whether the reported error can be managed by calibration or that permanent damage occurred to ADC. |
| 11 (algorithm S + C) | 0 – 2 (S), 0 – 11 (C) | Supply for One-Shot mode and (supply + capacitive) for Scan mode only. | Same as above. | |

ADC:

- Schedules self-testing algorithms using configuration registers.
- Monitors the converted data using analog watchdog registers.
- Flags any errors to a fault control unit (FCCU), if one exists.

Self-test steps can be activated from software (CPU) or BCTU.

| ADC mode | TEST algorithm (CPU) | TEST algorithm/step (BCTU) |
|---|---|---|
| CPU mode (MCR[BCTUEN] = 0) | Yes | No |

*Table continues on the next page...*

*Table continued from the previous page...*

| ADC mode | TEST algorithm (CPU) | TEST algorithm/step (BCTU) |
|---|---|---|
| | • One-Shot operation mode<br>• Scan operation mode | |
| BCTU mode | No | Yes |

#### 57.3.17.2.1 CPU mode

In this mode, self-test conversion is similar to normal conversion. You enable self-test by writing 1 to STCR2[EN].

Self-test conversions execute along with the functional conversions. The sequencing of steps of the selected algorithm depends on the operating mode of normal conversions (selected by MCR[MODE]).

In One-Shot operation mode, if self-test is enabled, only one step of the selected self-testing algorithm is executed at the end of the chain. The step number and algorithm to be executed are programmed in Self-Test Status 3 (STSR3). In One-Shot operation mode the sequence is:

1. Program NCMR*n* to select channels to be converted for normal conversion.
2. Write 0 to MCR[MODE] to select one-shot operation.
3. Configure the self-test algorithm threshold values (see Table 274).
4. Program sampling duration values in STCR1[INPSAMP*m*] (*m* = S, C).
5. Select the self-testing algorithm (in STCR3[ALG]), and step (in STCR3[MSTEP]). The default is algorithm S, step 0.
6. Enable self-testing by writing 1 to STCR2[EN].
7. Start the normal conversion by writing 1 to MCR[NSTART].
8. All normal conversions are executed as usual.
9. At the end of all normal conversions in the chain, the step number (programmed in STCR3[MSTEP]) of the self-testing algorithm (selected by STCR3[ALG]), is executed similar to a normal functional channel.
10. At the end of the conversion for the self-test channel:
    - The result is written to STDR1[TCDATA].
    - STDR1[VALID] transitions to 1.
    - ISR[EOC], ISR[ECH], and STSR1[ST_EOC] transition to 1.
11. ADC returns to Idle state.

For example: Channels A→B→C→D→E→F→G→H are present in the device where channels B→D→E are to be converted in One-Shot operation mode. At the end of conversion for channels B→D→E, self-test conversion is performed and ISR[ECH] and ISR[EOC] transition to 1. The sequence is B→D→E→self-test step.

MSR[NSTART] transitions to 1 when the normal conversion starts and transitions to 0 at the end of conversion for the test channel.

In Scan operation mode, consecutive steps of the selected self-test algorithm are converted continuously at the end of each chain of normal conversions. The number of channels converted at the end of each chain is 1 (except for algorithm S, in which all the steps are performed sequentially without any functional conversion interleaved). So, in Scan operation mode the sequence is:

1. Program NCMR*n* to select the channels to be converted for normal conversion.
2. Write 1 to MCR[MODE] to select Scan operation mode.
3. Configure the self-test algorithm threshold values (see Table 274).
4. Program sampling duration values in STCR1[INPSAMP*m*] field (*m* = S, C).

5. Select the self-testing algorithm in STCR3[ALG]. By default, all algorithms (supply and capacitive) are selected, that is, all algorithms are executed step-by-step, one after the other.
6. Enable self-testing by writing 1 to STCR2[EN].
7. Start the normal conversion by writing 1 to MCR[NSTART].
8. All normal conversions are executed as usual.
9. At the end of the chain of normal conversions (assuming the default value of STCR3[ALG]), all steps of algorithm S are performed (as algorithm S is always atomic). MSR[SELF_TEST_S] transitions to 1.
10. At the end of the conversion of the self-test channel for the last step of algorithm S, the result is written to STDR1[TCDATA] and STDR1[VALID] transitions to 1. At the same time, MSR[SELF_TEST_S] transitions to 0. The following registers also transition to 1:
    - ISR[EOC]
    - ISR[ECH]
    - STSR1[ST_EOC]
11. The next chain of normal conversions then starts.
12. At end of the chain of normal conversions, ADC executes step 0 of algorithm C.
13. At the end of the conversion of the test channel for step 0 of algorithm C, the result is stored in STDR1[TCDATA] and STDR1[VALID] transitions to 1 (if STSR1[OVERWR] is 1). Also, ISR[EOC] and ISR[ECH] transition to 1. STSR1[ST_EOC] also transitions to 1.
14. The next chain of normal conversion then starts.
15. At the end of the normal conversion chain, ADC executes step1 of algorithm C.

This process continues for all steps of all three algorithms. The state machine returns to Idle state when MCR[NSTART] transitions to 0.

For example: channels A→B→C→D→E→F→G→H are present in the device where channels B→D→E are to be converted in Scan operation mode. At the end of every conversion chain for channels B→D→E, one step of the self-test conversion is performed and ISR[ECH] and ISR[EOC] transition to 1.

The sequence is:

B→D→E→ST-S0→ST-S1→ST-S2→B→D→E→ST-C0→B→D→E→ST-C1→B→D→E→ST-C2→ . . . B→D→E→ST-C11→B→D→E→ST-S0 . . .

ST-Sx - Self-test supply step x (x = 0, 1, 2)

ST-Cx - Self-test capacitive step x (x = 0 to 11)

MSR[NSTART] transitions to 1 when normal conversion starts.

**NOTE**

- If instead of starting normal conversion by software (by writing 1 to MCR[NSTART]), it is started by an external trigger, the self-test behavior remains the same.
- Self-test channel conversion is not performed for injected conversions. It is performed only during normal conversions.
- If, during a test channel conversion, an injected conversion arrives, the test conversion is aborted (just as a normal functional channel) and the injected conversion is performed. After the injected conversion is complete, the test conversion resumes from the step at which it was aborted. In this case, MSR[SELF_TEST_S] remains 1 during the injected conversion.
- For self-test, MCR[MODE] should be programmed (at least one cycle) before writing 1 MCR[NSTART] and should not be changed until the conversion is finished or terminated.

Table 274. Recommended self-test threshold values

| Register field | Hex value | Decimal value | Comment |
|---|---|---|---|
| STAW0R[THRH] | 3669h | 13929 | The threshold values depend on the stability of the supply and the reference voltage. When you see voltage variations, widen the limits. |
| STAW0R[THRL] | 5 V reference: 1A7Fh<br>3 V reference: 2B27h | 5 V reference: 6784<br>3 V reference: 11047 | |
| STAW1R[THRL] | 3FF9h | 16377 | |
| STAW2R[THRL] | 3FF9h | 16377 | |
| STAW4R[THRH] | 20h | 32 | |
| STAW4R[THRL] | 7FE0h | –32 | |
| STAW5R[THRH] | 20h | 32 | |
| STAW5R[THRL] | 7FE0h | –32 | |

#### 57.3.17.2.2 BCTU mode

BCTU mode is enabled by writing 1 to MCR[BCTUEN].

Self-test conversion behaves in the same way for all BCTU operating modes (Trigger mode and Control mode). In both cases, if MCR[BCTUEN] = 1, the self-test conversion can be started only by BCTU—software cannot start it. MCR[BCTUEN] must not be changed during normal conversion. The interface between BCTU and ADC (for self-test) is shown in the following figure.

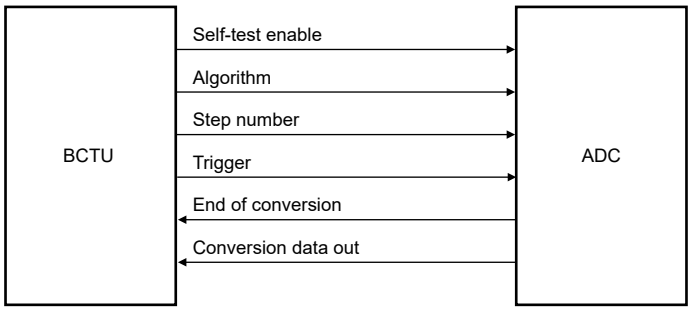



Figure 230. Self-test interface between ADC and BCTU

For self-test conversions in BCTU mode, BCTU asserts self-test enable along with the Trigger command. The algorithm and the step number to be executed is put on algorithm and step number, respectively. The other commands (Trigger, End Of Conversion, and Conversion Data Out) have the same meaning, like normal BCTU functional conversion.

As already mentioned for algorithm S, the three steps must be atomic. In BCTU mode, this is managed by BCTU, that is, BCTU sends three triggers (one for each step) asserting self-test enable and updating test_step for each step.

#### 57.3.17.2.3 Abort and abort chain for self-testing channel

Writing 1 to MCR[ABORT] during self-test channel conversion has no effect.

In One-Shot operation mode, if you write 1 to MCR[ABORTCHAIN] during the self-test channel conversion, the self-test channel is aborted and ISR[ECH] transitions to 1. In this case, ISR[EOC] for the self-test channel is not generated.

In Scan operation mode, if MCR[ABORTCHAIN] = 1 when self-test channel step *n* is ongoing, self-test channel step *n* is aborted and the next chain conversion begins. At the end of the chain, the step *n* conversion is performed again. (For algorithm S, the full algorithm is executed again).

#### 57.3.17.2.4 Self-test analog watchdog

ADC provides monitoring options for conversion data generated by self-test algorithms. Analog watchdogs determine whether the conversion results for self-test algorithms are in the range of a particular guard area. ADC provides separate self-test analog watchdog registers for each algorithm.

After the conversion of each step of a self-test algorithm, ADC compares the converted value and the threshold values if the analog watchdog feature has been enabled by writing 1 to STAW*n*R[AWDE] (for example, STAW0R[AWDE]). If the converted value is not between the upper and lower threshold values specified by the Self-Test Analog Watchdog Register for the particular algorithm, the corresponding error flag, STSR1[ERR_x], transitions to 1 and the step number in which error occurred is captured in STSR1[STEP_x] (for algorithm C). Erroneous data is written to STSR4[DATA*n*]. The STSR1[ERR_x] flags generate an interrupt if enabled by the corresponding mask bit in Self-Test Configuration 2 (STCR2). The fault indication is also forwarded to the fault control unit, if present, so that necessary action can be taken at the chip level. The fault type (critical or noncritical) is specified by the configuration in Self-Test Configuration 2 (STCR2).

The analog watchdog feature works differently for algorithm S. Algorithm S is always an atomic operation. Self-Test Status 1 (STSR1) has a separate error field for each step of algorithm S to avoid overwrite in case an error occurs in more than one step. Therefore, there are separate mask bits for each step in STCR1. For the same reason, the status registers (Self-Test Status 2 (STSR2) and Self-Test Status 3 (STSR3)) have separate fields for each step to store erroneous data.

In step 0 of the supply algorithm, ADC measures the band gap voltage (1.2 V), which is assumed to be stable. The conversion result of step 0 is compared against high (THRH) and low (THRL) thresholds as defined in Self-Test Analog Watchdog S0 (STAW0R), if enabled (STAW0R[AWDE] = 1). The STSR1[ERR_S0] flag transitions to 1 if any of the thresholds are violated. STSR1[ERR_S0] is cleared by writing 1 to it. The conversion result of ADC can be calculated with the following formula:

- ADC data (decimal) = ( Vin × $2^{15}$ ÷ (Vrh - Vrl))

Where:

- Vin: input voltage (band gap voltage in this case)
- Vrh: ADC reference high voltage
- Vrl: ADC reference low voltage

The upper and lower threshold value can be calculated with the above formula using maximum and minimum values of Vin, Vrh, Vrl (Vin is the band gap supply voltage in this case):

- Upper threshold (THRH, decimal) = (Vin[max] × $2^{15}$) ÷ (Vrh[min] - Vrl[max])
- Lower threshold (THRL, decimal) = (Vin[min] × $2^{15}$) ÷ (Vrh[max] - Vrl[min])

The minimum and maximum voltages can be obtained from the chip data sheet. The band gap voltage tolerance, as well as the actual reference voltage used and its tolerances, must be considered when calculating high and low thresholds.

For algorithm S steps 1 and 2, (VREF/VREF) is measured in order to check the integrity of the sampling signal. For these particular conversions, no higher threshold value is required as the ideal value is FFFh. Only the lower threshold value is programmed in Self-Test Analog Watchdog S1 (STAW1R) and Self-Test Analog Watchdog S2 (STAW2R).

For algorithm C, a separate register is provided for step 0. In step 0 an offset for other steps is measured. The converted data is compared with the threshold values in Self-Test Analog Watchdog C (STAW5R) if STAW4R[AWDE] = 1. For other steps, this offset is subtracted from converted data before performing watchdog checks.

#### 57.3.17.2.5 Watchdog timer

The watchdog timer is an additional check that monitors the sequence of the self-testing algorithm that has been implemented, and also checks whether the algorithm is completed within a safe time period. The watchdog timers are enabled for CPU as well as BCTU conversions. Each algorithm has an independent watchdog timer. The watchdog timer for a particular algorithm is enabled by writing 1 to STAWxR[WDTE]. The safe time value is programmed in STBRR[WDT] (the default value is 10 ms, assuming an 80 MHz clock).

The safe time is measured starting from step 0 of the algorithm (including all normal chain conversions in between) to the point where step 0 of the same algorithm starts again.

The programming sequence is:

1. Program NCMR*n* to select the channels for normal conversion in Scan operation mode (MCR[MODE] = 1).
2. Select the self-test algorithm in STCR3[ALG]. By default, all algorithms (supply and capacitive) are selected (all algorithms are executed step-by-step, one after the other).
3. Enable the self-test channel by writing 1 to STCR2[EN].
4. Program the safe period value in STBRR[WDT].
5. Enable the watchdog timer by writing 1 to STAWxR[WDTE]. Assume writing 1 to STAWxR[WDTE] occurs at time *t*0. It is important to do all configuration programming before writing 1 to STAWxR[WDTE], because the safe time check is also performed between writing 1 to STAWxR[WDTE] and the start of step 0. This is to verify that the algorithm has started within the safe time.
6. Start the normal conversion by writing 1 to MCR[NSTART].
7. At the end of the first conversion chain, three steps (steps 0, 1, and 2) of algorithm S are executed in sequence. Assume the start of Step 0 occurs at time *t*1.
8. After completion of algorithm S, ADC performs conversion of the next chain.
9. At the end of the conversion chain, the first step (step 0) of algorithm C is performed.
10. In this way, one step of algorithm C is performed after completion of one conversion chain until all steps are finished. Steps are executed in sequence (steps 0–11)
11. After the last step of algorithm C, another chain conversion is executed. At the end of this chain conversion, step 0 of algorithm S is started again, repeating the whole sequence. Assume this time (starting of step 0 of the supply algorithm) to be at time *t*2.
12. For algorithm S, if (*t*1 - *t*0) > safe period or (*t*2 - *t*1) > safe period, the watchdog timer flags an error and STSR1[WDTERR] transitions to 1. ADC asserts a fault signal per the criticality configuration (critical or noncritical) and an interrupt is generated, if enabled (STCR2[MSKWDTERR] = 1). Otherwise, the watchdog timer counter is reset and starts again to monitor the next sequence.
13. A similar sequence is followed for watchdog timers for algorithm C.

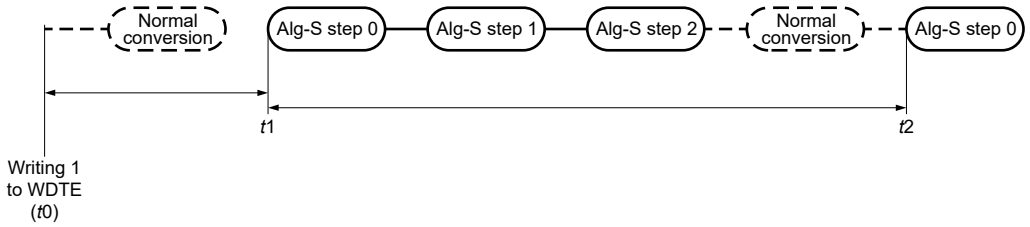



**Figure 231. Watchdog timer monitor for algorithm S**

**NOTE**

1. Because BCTU may not incorporate any safe period checking mechanism, the watchdog timers can also be enabled for BCTU conversions.
2. You must disable watchdog timers for self-test algorithms that are not executed.

#### 57.3.17.2.5.1 Watchdog sequence checking

The self-test watchdog timer incorporates sequence checking features to verify that the steps of each algorithm are executed in the correct order. If the steps are not in the correct order, STSR1[WDSERR] transitions to 1 to indicate the error. A fault is indicated by an interrupt flag if STCR2[MSKWDSERR] = 1. A fault is also asserted as per criticality configuration (critical or noncritical) and an interrupt generated if STCR2[MSKWDSERR] = 1.

A watchdog sequence error is flagged in the following cases:

- The steps of any algorithm are not executed in proper order.
- The step numbers provided by BCTU for a BCTU conversion are not in order. Watchdog sequence checking is significant only for BCTU burst mode.
- When an abort chain occurs during a self-test channel conversion, that step must be repeated at the end of the next chain. This generates a sequence error as soon as the self-test channel conversion resumes. There is an exception, however. If an abort chain occurs during the last step of algorithm S, the sequence error is not flagged because the whole algorithm must be repeated.

If an injected conversion occurs during self-test channel conversion, a watchdog sequence error is not flagged, although the ongoing step number is aborted and is repeated.

**NOTE**

The watchdog timer feature is applicable for Scan operation mode but not for One-Shot operation mode.

#### 57.3.17.2.6 Baud rate control for test channel

The baud rate control feature controls the scheduling of self-test channel conversions between normal conversion chains. The scheduling rate is specified by STBRR[BR].

By default, if the self-test channel is enabled, one step of the selected algorithm is executed after each normal conversion chain. The bandwidth consumed by the self-test channel depends on the number of channels in the normal chain. For example, if you have 50 normal conversions in a chain, then the self-test channel consumes only 2% of the total bandwidth, which is very small. If the number decreases to just 3 channels, the bandwidth consumed by the self-test channel is 25%, which is significant (and may not be desirable because it slows down the normal conversion rate).

STBRR[BR] provides flexibility by scheduling the self-test channel conversion to be performed not at the end of every chain, but at the end of BR + 1 chains. For example, if STBRR[BR] = 5, a single step of the selected algorithm for the test channel is performed after 6 chain conversions. The next step is performed at the end of the next six chain conversions, and so on. By default, STBRR[BR] = 0.

**NOTE**

This feature is applicable only for Scan operation mode and not for One-Shot operation mode. STBRR[BR] should be written with 0 for One-Shot operation mode.

To use the baud rate control feature in Scan operation mode, NCMR*n* should have a nonzero value.

##### 57.3.17.2.6.1 Abort chain when baud rate is nonzero

As already described, for a nonzero value of STBRR[BR], the self-test channel conversion is performed at the end of BR + 1 chains. If an abort chain occurs during the chain in which a self-test channel conversion is scheduled to be performed, the self-test channel conversion is performed after the next BR + 1 chains.

For example, if STBRR[BR] = 2, the sequence is two normal chains (without any self-test channel conversion), followed by a chain with the self-test channel converted at the end. If an abort chain occurs during the first two chains, it is treated as a normal chain abort and the self-test channel is converted at the end of the third chain only (as in the case without any abort chain). However, if an abort chain occurs during the third chain in which the self-test channel is scheduled to be converted, the self-test channel is converted after the next three chains (for example, at the end of the sixth chain, counting from the beginning).

### 57.3.18 Conversion time

Total conversion time depends on the conversion clock frequency, which is configured by programming MCR[ADCLKSEL].

The components of conversion time and affecting configurations are:

- PST
- ST
- CT

- DP
- TPT

Presample phase time is equal to the sample time with a one AD_clk cycle delay for phase transition from presample phase to sample phase. CTR*n*[INPSAMP] controls sample time for different types of channels and is specified in terms of AD_clk cycles, so the presampling time is sample time plus one cycle of AD_clk. The minimum value of sample time is 8. If the value programmed is less than 8, it has no effect on sample time and the sample time is 8 AD_clk cycles.

Compare phase time is controlled by:

- The evaluation time of a single bit
- The number of bits converted

For *n*-bit conversions, the value of CT is *n* multiplied by the evaluation time of a single bit in terms of AD_clk cycles.

The evaluation time of a single bit is 4 AD_clk cycles if AMSIO[HSEN] = 0h and AMSIO[CMPCTRL0] = 0b. The evaluation time of a single bit is 6 AD_clk cycles if AMSIO[HSEN] = 1h and AMSIO[CMPCTRL0] = 1b.

The data processing time is 2 cycles of AD_clk. During these cycles raw converted data is corrected for offset, gain, capacitor mismatch, and so on.

Trigger processing time consists of:

- One module_clock cycle, which is required to prepare the channel and calculate the initial gain value used by ADC for the first conversion.
- BCTU triggering time:
  - Triggers from the synchronous BCTU interface require 1 cycle of the module clock for processing. One more cycle is required to register the BCTU trigger in synchronous mode.

The total conversion time, in terms of the module clock cycles, is calculated using the following equation for normal and injected conversions:

Total_conversion_time = ( [(PST + ST + CT + DP) × chain_length] + TPT) × TAD_clk

Example:

The ADC controller clock is equal to the module clock (80 MHz, clock cycle = 12.5 ns)

- ADC resolution 12 bit + 1 bit for a special capacitor (CS)
- Three channels are programmed in NCMR*n*, so a chain of three is to be converted
- Default sample time (22 cycles) is specified
- No presampling
- Conversion time (4 cycles per bit)

The total time for the three conversions = [(0 + 22 + (4×13) + 2) × 3] + 1 = 229 cycles ~= 2.862 µs

Conversion timing in Calibration for full test:

- Single conversion time = [sample time (CALBISTREG[TSAMP]) + compare time (11 × 4) + data proc(2)]
- Inter-conversion gap = 1 cycle
- Total number of tests = 12
- Averaging = 32

One test duration = (single conversion time + inter-conversion gap) × averaging samples

Total test duration = ( ( one test duration ) × total number of tests).

Example: sample time = 22

Total test time = ( ( ( (22 + (11×4) +2)+1) × 32) × 12) = 26496 cycles × 12.5 ns (80 MHz) = 331200 ns

Conversion time in self-test:

Self-test conversion time equals one test period multiplied by the number of consecutive (atomic) steps. For supply self-test, the atomic step is 3 and for other tests it is 1.

One test period for self-test is as below.

- For supply tests: ( [ sample_time + (13 × 4) + 2] )
- For other algorithms: ( [ sample_time + (11 × 4) + 2] )

**NOTE**

Algorithm S steps (conversions) always run in atomic operation, required for an algorithmic procedure. In Scan operation mode, when all algorithms are selected, three steps of algorithm S run together after the end of the conversion chain. algorithm C steps are not required to be run together, so after each conversion chain one step is executed serially. Conversion time with self-test must also be calculated accordingly.

Example.

- Channels in chain A→B→C.
- Scan operation mode.
- All self-test algorithms are selected.

So, the resulting conversion order is:

[A→B→C→AS_step0→AS_step1→AS_step2→A→B→C→AC_step0→A→B→C→AC_step1→A→B→C→AC_step2→.....→A→B→C→AC_step11]→[same sequence repeats]

**NOTE**

- AS = Algorithm S (supply self-test)
- AC = Algorithm C (capacitive self-test)

### 57.3.19 Conversion data processing

Raw converted data contains many types of errors, such as offset, gain, and so on. ADC processes raw conversion data before writing the result to a data register, to reduce or eliminate error contributions. The process of error correction happens in parallel with bit-by-bit evaluation, using the correction values generated during the offset determination and calibration process.

The error correction value is subtracted from the raw conversion result whenever the comparator output for one of the calibrated capacitors evaluates high during the conversion period.

To reduce data processing time, error correction is done in parallel during each bit evaluation of a conversion. Lower-weighted capacitors (smaller than C9) are not calibrated, so no error correction is performed for these capacitors. However, they are self-tested to guarantee that the error contribution is below specified accuracy limits.

The final result is checked for any overflow or underflow. When the processed data is above the maximum value that can be represented by ADC resolution, the output data is forced to all 1s (FFFh for 12-bit resolution). Similarly, if the processed data is negative then output data is forced to all 0s (000h for 12-bit resolution).

### 57.3.20 User-defined offset and gain values

In addition to the ADC-calculated offset value and gain value, ADC enables you to specify another set of offset and gain values in Offset And Gain User (OFSGNUSR). These values must be stored in two's complement format, where the leftmost bit (the most significant bit) is the sign bit.

ADC subtracts these values from the final processed conversion result. The values are used to removed fixed DC bias or any other known errors—for instance those caused by board design, sensors, and so on.

## 57.4 External signals

This module has no external signals.

## 57.5 Clock frequency

ADC's internal blocks are designed for an input clock frequency up to the Fin (Input Signal Bandwidth) maximum (see the ADC electrical specification section in the chip data sheet). The specified speed can be achieved with this clock frequency. If the input clock frequency exceeds the specified maximum frequency, ADC accuracy degrades and, in the worst case scenario, ADC malfunctions. ADC accuracy also degrades if the input clock frequency is below the specified minimum frequency. Sampling time has an absolute maximum value to guarantee the specified parameters and accuracy. For slower clocks, the sampling clock count must be programmed accordingly.

## 57.6 Memory map and register definition

ADC has a wide range of configurations to tailor its operation to application requirements. This comes with the drawback of possible corner cases. To avoid unintended behavior, configure ADC before starting a conversion and change any configuration only when ADC is idle, in other words when MSR[ADCSTATUS] = 0.

### 57.6.1 Transfer error description

The following register accesses cause a transfer error and do not change register content:

- Any access to an unused address.
- Any write access to a read-only register.

### 57.6.2 ADC register descriptions

#### 57.6.2.1 ADC memory map

ADC_0 base address: 400A_0000h

ADC_1 base address: 400A_4000h

ADC_2 base address: 400A_8000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Main Configuration (MCR) | 32 | RW | 0000_0001h |
| 4h | Main Status (MSR) | 32 | R | 0000_0001h |
| 10h | Interrupt Status (ISR) | 32 | RW | 0000_0000h |
| 14h | Channel End Of Conversion Flag For Precision Inputs (CEOCFR0) | 32 | RW | 0000_0000h |
| 18h | Channel End Of Conversion Flag For Standard Inputs (CEOCFR1) | 32 | RW | 0000_0000h |
| 1Ch | Channel End Of Conversion Flag For External Inputs (CEOCFR2) | 32 | RW | 0000_0000h |
| 20h | Interrupt Mask (IMR) | 32 | RW | 0000_0000h |
| 24h | EOC Interrupt Enable For Precision Inputs (CIMR0) | 32 | RW | 0000_0000h |
| 28h | EOC Interrupt Enable For Standard Inputs (CIMR1) | 32 | RW | 0000_0000h |
| 2Ch | EOC Interrupt Enable For External Inputs (CIMR2) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 30h | Analog Watchdog Threshold Interrupt Status (WTISR) | 32 | RW | 0000_0000h |
| 34h | Analog Watchdog Threshold Interrupt Enable (WTIMR) | 32 | RW | 0000_0000h |
| 40h | Direct Memory Access Configuration (DMAE) | 32 | RW | 0000_0000h |
| 44h | DMA Request Enable For Precision Inputs (DMAR0) | 32 | RW | 0000_0000h |
| 48h | DMA Request Enable For Standard Inputs (DMAR1) | 32 | RW | 0000_0000h |
| 4Ch | DMA Request Enable For External Inputs (DMAR2) | 32 | RW | 0000_0000h |
| 60h - 6Ch | Analog Watchdog Threshold Values (THRHLR0 - THRHLR3) | 32 | RW | 7FFF_0000h |
| 80h | Presampling Control (PSCR) | 32 | RW | 0000_0000h |
| 84h | Presampling Enable For Precision Inputs (PSR0) | 32 | RW | 0000_0000h |
| 88h | Presampling Enable For Standard Inputs (PSR1) | 32 | RW | 0000_0000h |
| 8Ch | Presampling Enable For External Inputs (PSR2) | 32 | RW | 0000_0000h |
| 94h | Conversion Timing For Precision Inputs (CTR0) | 32 | RW | 0000_0016h |
| 98h | Conversion Timing For Standard Inputs (CTR1) | 32 | RW | 0000_0016h |
| 9Ch | Conversion Timing For External Inputs (CTR2) | 32 | RW | 0000_0016h |
| A4h | Normal Conversion Enable For Precision Inputs (NCMR0) | 32 | RW | 0000_0000h |
| A8h | Normal Conversion Enable For Standard Inputs (NCMR1) | 32 | RW | 0000_0000h |
| ACh | Normal Conversion Enable For External Inputs (NCMR2) | 32 | RW | 0000_0000h |
| B4h | Injected Conversion Enable For Precision Inputs (JCMR0) | 32 | RW | 0000_0000h |
| B8h | Injected Conversion Enable For Standard Inputs (JCMR1) | 32 | RW | 0000_0000h |
| BCh | Injected Conversion Enable For External Inputs (JCMR2) | 32 | RW | 0000_0000h |
| C4h | Delay Start Of Data Conversion (DSDR) | 32 | RW | 0000_0000h |
| C8h | Power Down Exit Delay (PDEDR) | 32 | RW | 0000_0000h |
| 100h - 11Ch | Precision Input n Conversion Data (PCDR0 - PCDR7) | 32 | R | 0000_0000h |
| 180h - 1DCh | Standard Input n Conversion Data (ICDR0 - ICDR23) | 32 | R | 0000_0000h |
| 200h - 27Ch | External Input n Conversion Data (ECDR0 - ECDR31) | 32 | R | 0000_0000h |
| 2B0h | Channel Analog Watchdog Select For Precision Inputs (CWSELRPI0) | 32 | RW | 0000_0000h |
| 2B4h | Channel Analog Watchdog Select For Precision Inputs (CWSELRPI1) | 32 | R | 0000_0000h |
| 2C0h | Channel Analog Watchdog Select For Standard Inputs (CWSELRSI0) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 2C4h | Channel Analog Watchdog Select For Standard Inputs (CWSELRSI1) | 32 | RW | 0000_0000h |
| 2C8h | Channel Analog Watchdog Select For Standard Inputs (CWSELRSI2) | 32 | RW | 0000_0000h |
| 2D0h | Channel Analog Watchdog Select For External inputs (CWSELREI0) | 32 | RW | 0000_0000h |
| 2D4h | Channel Analog Watchdog Select For External inputs (CWSELREI1) | 32 | RW | 0000_0000h |
| 2D8h | Channel Analog Watchdog Select For External inputs (CWSELREI2) | 32 | RW | 0000_0000h |
| 2DCh | Channel Analog Watchdog Select For External inputs (CWSELREI3) | 32 | RW | 0000_0000h |
| 2E0h | Channel Watchdog Enable For Precision Inputs (CWENR0) | 32 | RW | 0000_0000h |
| 2E4h | Channel Watchdog Enable For Standard Inputs (CWENR1) | 32 | RW | 0000_0000h |
| 2E8h | Channel Watchdog Enable For External Inputs (CWENR2) | 32 | RW | 0000_0000h |
| 2F0h | Analog Watchdog Out Of Range For Precision Inputs (AWORR0) | 32 | RW | 0000_0000h |
| 2F4h | Analog Watchdog Out Of Range For Standard Inputs (AWORR1) | 32 | RW | 0000_0000h |
| 2F8h | Analog Watchdog Out Of Range For External Inputs (AWORR2) | 32 | RW | 0000_0000h |
| 340h | Self-Test Configuration 1 (STCR1) | 32 | RW | 1818_2507h |
| 344h | Self-Test Configuration 2 (STCR2) | 32 | RW | 0000_0005h |
| 348h | Self-Test Configuration 3 (STCR3) | 32 | RW | 0000_0300h |
| 34Ch | Self-Test Baud Rate (STBRR) | 32 | RW | 0005_0000h |
| 350h | Self-Test Status 1 (STSR1) | 32 | RW | 0000_0000h |
| 354h | Self-Test Status 2 (STSR2) | 32 | R | 0000_0000h |
| 358h | Self-Test Status 3 (STSR3) | 32 | R | 0000_0000h |
| 35Ch | Self-Test Status 4 (STSR4) | 32 | R | 0000_0000h |
| 370h | Self-Test Conversion Data 1 (STDR1) | 32 | R | 0000_0000h |
| 380h | Self-Test Analog Watchdog S0 (STAW0R) | 32 | RW | 0727_04C5h |
| 388h | Self-Test Analog Watchdog S1 (STAW1R) | 32 | RW | 0000_3FF9h |
| 38Ch | Self-Test Analog Watchdog S2 (STAW2R) | 32 | RW | 0000_3FF9h |
| 394h | Self-Test Analog Watchdog C0 (STAW4R) | 32 | RW | 0010_3FF0h |
| 398h | Self-Test Analog Watchdog C (STAW5R) | 32 | RW | 0010_3FF0h |
| 39Ch | Analog Miscellaneous In/Out register (AMSIO) | 32 | RW | 0000_0811h |
| 3A0h | Control And Calibration Status (CALBISTREG) | 32 | RW | See section |
| 3A8h | Offset And Gain User (OFSGNUSR) | 32 | RW | 0004_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 3B4h | Calibration Value 2 (CAL2) | 32 | RW | 4300_8243h |

### 57.6.2.2 Main Configuration (MCR)

#### Offset

| Register | Offset |
|---|---|
| MCR | 0h |

#### Function

Configures most ADC features. You must change field values only when ADC is in Idle state, except for the ABORTCHAIN and ABORT fields.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | OWREN | WLSIDE | MODE | 0 | TRGEN | EDGE | XSTRTEN | NSTART | 0 | JTRGEN | JEDGE | JSTART | 0 | | BCTUEN | BCTU_MO... |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | 0 | | | AVGEN | AVGS | | Reserved | ABORTCH... | ABORT | ACKO | 0 | ADCLKSEL | | | PWDN |
| W | STCL | | | | | | | | W1C | W1C | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

#### Fields

| Field | Function |
|---|---|
| 31<br>OWREN | Overwrite Enable<br>Specifies whether a conversion data register accepts new data before the current conversion data has been read. When enabled, the new conversion result overwrites the older data, regardless of the validity of the older conversion data. When PCDR*n*[VALID] = 1, the conversion data has not been read.<br>0b - Disable<br>1b - Enable |
| 30<br>WLSIDE | Write Left-Aligned |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Specifies whether conversion data is right-aligned or left-aligned when written to one of the conversion data registers:<br>• PCDR*n*<br>• ICDR0*n*<br>• ECDR*n*<br>Right-aligned data occupies *CDR[14:0]. Left-aligned data occupies *CDR[15:1],<br>0b - Right aligned<br>1b - Left-aligned |
| 29<br>MODE | Normal Conversion Mode<br>Specifies whether the set of input channels selected for a normal conversion are converted only once per start event or continuously after a start event.<br>If a conversion is done only once, ADC enters an idle state after conversion is complete.<br>For continuous conversion, all input channels selected for the normal conversion are converted continuously in a loop. Stop the loop by writing 0 to NSTART.<br>0b - Single conversion<br>1b - Continuous conversion |
| 28<br>— | Reserved |
| 27<br>TRGEN | External Trigger Enable<br>Specifies whether the normal trigger input starts a conversion.<br>0b - Normal trigger input does not start a conversion<br>1b - Normal trigger input starts a conversion |
| 26<br>EDGE | External Trigger Edge Selection<br>Selects which edge of the normal trigger input starts a conversion.<br>0b - Falling edge<br>1b - Rising edge |
| 25<br>XSTRTEN | Auxiliary External Start Enable<br>Enables the auxiliary normal trigger source to start a conversion. You can use this field to synchronize the start of a conversion of two ADC instances.<br>0b - Disable<br>1b - Enable |
| 24<br>NSTART | Start Normal Conversion |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Starts a normal conversion. If the continuous mode is selected (MODE = 1), the value of this field remains 1. Write 0 to this field to stop the in-progress conversion loop after conversion of the last input channel of the selected set of input channels is complete.<br>If continuous mode is not selected (MODE = 0), this field automatically resets to 0 after writing 1.<br>0b - No effect<br>1b - Starts conversion |
| 23<br>— | Reserved |
| 22<br>JTRGEN | Injection Trigger Enable<br>Enables the injected trigger input as a source to start a conversion.<br>0b - Disable<br>1b - Enable |
| 21<br>JEDGE | Injected Trigger Edge Selection<br>Selects which edge of the injected trigger input starts an injected conversion.<br>0b - Falling edge<br>1b - Rising edge |
| 20<br>JSTART | Injected Start<br>Interrupts any ongoing normal conversion and starts an injected conversion. This field automatically resets to 0. If an injected conversion is already ongoing, the field value remains 1 until the next injected conversion starts. This field can only be written with 1.<br>0b - Injected conversion can be started<br>1b - Starts an injected conversion |
| 19-18<br>— | Reserved |
| 17<br>BCTUEN | Body Cross Trigger Unit Enable<br>Enables BCTU as a trigger source.<br>0b - Disable<br>1b - Enable |
| 16<br>BCTU_MODE | Body Cross Trigger Unit Mode Select<br>Specifies whether sources other than BCTU can start a conversion when the BCTU is enabled as a trigger source (BCTUEN = 1). This field is writeable only when ADC is in Power Down state (PWDN = 1).<br>0b - Only BCTU can trigger conversion<br>1b - All trigger sources can trigger conversion |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 15<br>STCL | Self-Test Configuration Lock<br>Protects the following registers from writing:<br>• STCR1<br>• STCR2<br>• STCR3<br>• STBRR<br>• STAW0R<br>• STAW1R<br>• STAW2R<br>• STAW4R<br>• STAW5R<br>0b - Registers are writeable<br>1b - Registers are read-only |
| 14-12<br>— | Reserved |
| 11<br>AVGEN | Averaging Enable<br>Enables conversion averaging.<br>0b - Disable<br>1b - Enable |
| 10-9<br>AVGS | Averaging Select<br>Specifies the number of conversions ADC uses to calculate the conversion result.<br>00b - 4 conversions<br>01b - 8 conversions<br>10b - 16 conversions<br>11b - 32 conversions |
| 8<br>— | Reserved |
| 7<br>ABORTCHAIN | Abort Chain<br>Aborts the conversion of the selected set of input channels (chain of input channels). The currently ongoing conversion completes.<br>This field always reads 0.<br>This field cannot be programmed during BCTU conversion. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Undefined<br>When writing<br>0b - Conversion continues<br>1b - Conversion aborted |
| 6<br>ABORT | Abort Conversion<br>Aborts an ongoing conversion. This field always reads 0 and cannot be written during BCTU conversion.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Undefined<br>When writing<br>0b - Conversion continues<br>1b - Conversion aborted |
| 5<br>ACKO | Auto Clock Off<br>Reduces power consumption by turning off the clock of the analog part of ADC when it is in Idle state.<br>0b - Clock always active<br>1b - Clock gated |
| 4-3<br>— | Reserved |
| 2-1<br>ADCLKSEL | Conversion Clock (AD_clk) Frequency Selection<br>Selects the frequency for the clock signal of the conversion circuit (AD_clk = module clock ÷ ($2^{ADCLKSEL}$)). This field can be written only when ADC is in Power Down state (PWDN = 1).<br>00b - Module clock frequency<br>01b - Module clock frequency ÷ 2<br>10b - Module clock frequency ÷ 4<br>11b - Module clock frequency ÷ 8 |
| 0<br>PWDN | Power Down<br>Reduces power consumption by turning off power to the analog portion of ADC. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - ADC enters a functional state<br>1b - ADC enters Power Down state |

### 57.6.2.3 Main Status (MSR)

#### Offset

| Register | Offset |
|---|---|
| MSR | 4h |

#### Function

Contains flags that indicate the current ADC state.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CALIBRTD | 0 | | | | | | NSTART | JABORT | 0 | | JSTART | 0 | SELF_TE... | Reserved | BCTUSTA... |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CHADDR | | | | | | | 0 | | | ACKO | 0 | | ADCSTATUS | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

#### Fields

| Field | Function |
|---|---|
| 31<br>CALIBRTD | Calibration Status<br>Indicates ADC calibration status.<br>0b - Uncalibrated or calibration unsuccessful<br>1b - Calibrated |
| 30-25<br>— | Reserved |
| 24<br>NSTART | Normal Conversion Started<br>Shows whether a normal conversion is in progress. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Not in progress<br>1b - In progress |
| 23<br>JABORT | Injected Conversion Aborted<br>Indicates whether the conversion of a set of inputs selected for injected conversion has been aborted. This field resets to 0 when a new injected conversion is started.<br>0b - Not aborted<br>1b - Aborted |
| 22-21<br>— | Reserved |
| 20<br>JSTART | Injected Conversion Started<br>Indicates whether an ongoing conversion was started by the injection trigger.<br>0b - Not an injected conversion<br>1b - Injected conversion |
| 19<br>— | Reserved |
| 18<br>SELF_TEST_S | Indicates whether an ongoing conversion is for self-test.<br>0b - Not self-test<br>1b - Self-test |
| 17<br>— | Reserved |
| 16<br>BCTUSTART | BCTU Conversion Started<br>Indicates whether a BCTU conversion is ongoing. This field is 1 when a BCTU trigger event is received and the BCTU conversion starts. When the BCTU trigger mode is selected, this field is automatically reset to 0 when conversion is completed. Otherwise, if BCTU control mode is selected, this field resets to 0 when the BCTU is disabled (BCTUEN = 0).<br>0b - Conversion was not triggered by BCTU<br>1b - Ongoing conversion was triggered by BCTU |
| 15-9<br>CHADDR | Input Under Measure<br>Contains the number of the input that is currently converted.<br>000_0000b-011_1111b - Input number |
| 8-6<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 5<br>ACKO | Auto Clock-Off On<br>Indicates whether the auto clock-off feature is active, and the conversion circuit is not receiving a clock signal.<br>When auto clock-off is inactive, the conversion circuit is clocked.<br>0b - Inactive<br>1b - Active |
| 4-3<br>— | Reserved |
| 2-0<br>ADCSTATUS | ADC State<br>Indicates the current state of the ADC Finite State Machine (FSM).<br>000b - Idle<br>001b - Power Down<br>010b - Wait<br>011b - Calibrate<br>100b - Convert<br>110b - Done |

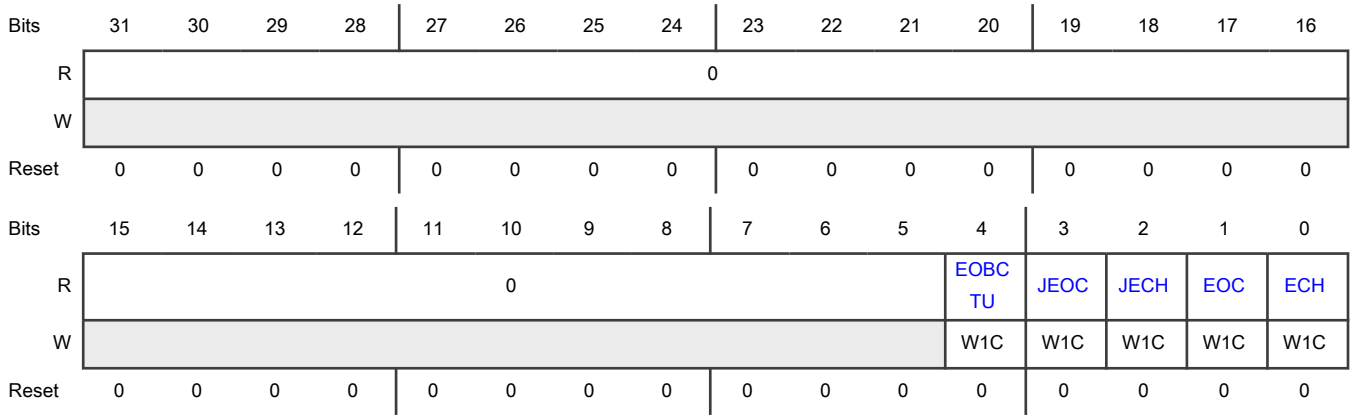
### 57.6.2.4 Interrupt Status (ISR)

**Offset**

| Register | Offset |
|---|---|
| ISR | 10h |

**Function**
Contains flags that indicate whether an interrupt has been generated.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-5<br>— | Reserved |
| 4<br>EOBCTU | End Of BCTU Conversion<br>Indicates the status of the End of BCTU Conversion (EOBCTU) interrupt, which is generated after a BCTU conversion completes.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No EOBCTU interrupt generated<br>1b - EOBCTU interrupt generated<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 3<br>JEOC | End Of Injected Conversion<br>Indicates the status of the End of Injected Conversion (JEOC) interrupt, which is generated after an injected conversion completes.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No JEOC interrupt generated<br>1b - JEOC interrupt generated<br>When writing |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No effect<br>1b - Clears flag |
| 2<br>JECH | End Of Injected Chain Conversion<br>Indicates the status of the End of Injected Chain (JECH) interrupt, which is generated after injected conversion of a set of inputs completes.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No JECH interrupt generated<br>1b - JECH interrupt generated<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 1<br>EOC | End Of Conversion<br>Indicates the status of the End of Conversion (EOC) interrupt, which is generated after a normal conversion completes.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No EOC interrupt generated<br>1b - Interrupt generated<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 0<br>ECH | End Of Chain Conversion<br>Indicates the status of the End of Chain (ECH) conversion interrupt, which is generated after a set of inputs has been converted normally.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Indicates no ECH interrupt generated<br>1b - Indicates an ECH interrupt has been generated<br>When writing |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No effect<br>1b - Clears flag |

### 57.6.2.5 Channel End Of Conversion Flag For Precision Inputs (CEOCFR0)

#### Offset

| Register | Offset |
|---|---|
| CEOCFR0 | 14h |

#### Function

Contains flags that indicate whether conversion is complete for a precision input.

#### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | Reserved | Reserved | PIEOCF7 | PIEOCF6 | PIEOCF5 | PIEOCF4 | PIEOCF3 | PIEOCF2 | PIEOCF1 | PIEOCF0 |
| W | | | | | | | | | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>— | Reserved |
| 8<br>— | Reserved |
| 7 | Precision Input End Of Conversion Flag 7 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| PIEOCF7 | Indicates whether conversion of a precision input 7 has completed.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion not complete<br>1b - Conversion complete<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 6<br>PIEOCF6 | Precision Input End Of Conversion Flag 6<br>Indicates whether conversion of a precision input 6 has completed.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion not complete<br>1b - Conversion complete<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 5<br>PIEOCF5 | Precision Input End Of Conversion Flag 5<br>Indicates whether conversion of a precision input 5 has completed.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion not complete<br>1b - Conversion complete<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 4<br>PIEOCF4 | Precision Input End Of Conversion Flag 4<br>Indicates whether conversion of a precision input 4 has completed. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion not complete<br>1b - Conversion complete<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 3<br>PIEOCF3 | Precision Input End Of Conversion Flag 3<br>Indicates whether conversion of a precision input 3 has completed.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion not complete<br>1b - Conversion complete<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 2<br>PIEOCF2 | Precision Input End Of Conversion Flag 2<br>Indicates whether conversion of a precision input 2 has completed.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion not complete<br>1b - Conversion complete<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 1<br>PIEOCF1 | Precision Input End Of Conversion Flag 1<br>Indicates whether conversion of a precision input 1 has completed.<br>**NOTE**<br>This field behaves differently for register reads and writes. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When reading<br>0b - Conversion not complete<br>1b - Conversion complete<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 0<br>PIEOCF0 | Precision Input End Of Conversion Flag 0<br>Indicates whether conversion of a precision input 0 has completed.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion not complete<br>1b - Conversion complete<br>When writing<br>0b - No effect<br>1b - Clears flag |

#### 57.6.2.6 Channel End Of Conversion Flag For Standard Inputs (CEOCFR1)

**Offset**

| Register | Offset |
|---|---|
| CEOCFR1 | 18h |

**Function**
Contains flags that indicate whether conversion is complete for a standard input.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | SIEOCF23 | SIEOCF22 | SIEOCF21 | SIEOCF20 | SIEOCF19 | SIEOCF18 | SIEOCF17 | SIEOCF16 |
| W | | | | | | | | | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SIEOCF15 | SIEOCF14 | SIEOCF13 | SIEOCF12 | SIEOCF11 | SIEOCF10 | SIEOCF9 | SIEOCF8 | SIEOCF7 | SIEOCF6 | SIEOCF5 | SIEOCF4 | SIEOCF3 | SIEOCF2 | SIEOCF1 | SIEOCF0 |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>SIEOCFn | Standard Input End Of Conversion Flag<br>Indicates whether conversion on standard input *n* is complete.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion not complete<br>1b - Conversion complete<br>When writing<br>0b - No effect<br>1b - Clears flag |

## 57.6.2.7 Channel End Of Conversion Flag For External Inputs (CEOCFR2)

### Offset

| Register | Offset |
|---|---|
| CEOCFR2 | 1Ch |

### Function

Contains flags that indicate whether conversion is complete for an external input.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| ADC_0 | CEOCFR2 | — |
| ADC_1 | CEOCFR2 | — |
| ADC_2 | — | CEOCFR2 |

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EIEOCF31 | EIEOCF30 | EIEOCF29 | EIEOCF28 | EIEOCF27 | EIEOCF26 | EIEOCF25 | EIEOCF24 | EIEOCF23 | EIEOCF22 | EIEOCF21 | EIEOCF20 | EIEOCF19 | EIEOCF18 | EIEOCF17 | EIEOCF16 |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EIEOCF15 | EIEOCF14 | EIEOCF13 | EIEOCF12 | EIEOCF11 | EIEOCF10 | EIEOCF9 | EIEOCF8 | EIEOCF7 | EIEOCF6 | EIEOCF5 | EIEOCF4 | EIEOCF3 | EIEOCF2 | EIEOCF1 | EIEOCF0 |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>EIEOCFn | External Input End Of Conversion Flag<br>Indicates whether conversion on external input *n* is complete.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion not complete<br>1b - Conversion complete<br>When writing<br>0b - No effect<br>1b - Clears flag |

## 57.6.2.8 Interrupt Mask (IMR)

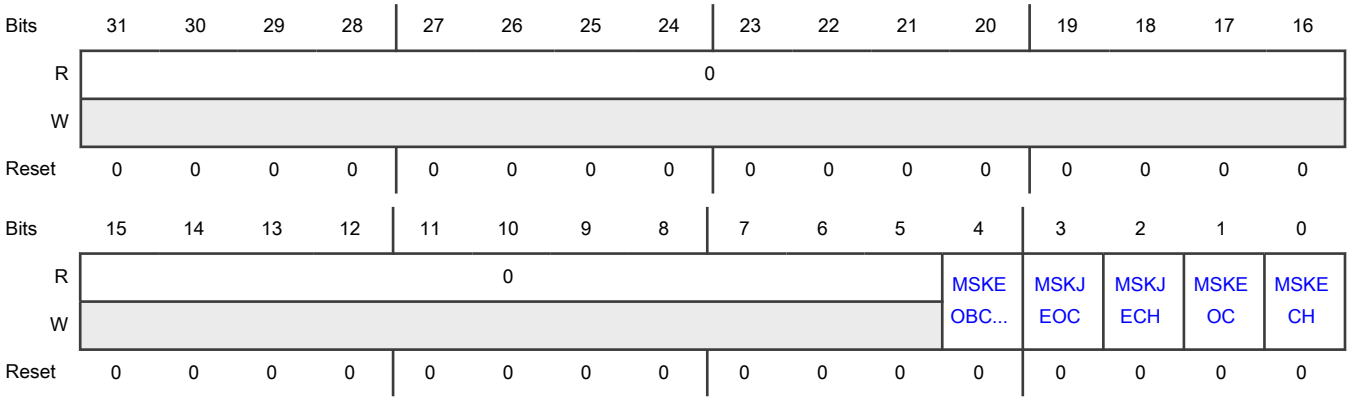
### Offset

| Register | Offset |
|---|---|
| IMR | 20h |

### Function

Enables flagging of interrupts.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-5<br>— | Reserved |
| 4<br>MSKEOBCTU | EOBCTU Interrupt Flag Enable<br>Specifies whether a completed BCTU conversion flags the EOBCTU interrupt.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |
| 3<br>MSKJEOC | JEOC Interrupt Flag Enable<br>Specifies whether completion of an injected conversion flags the JEOC interrupt.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |
| 2<br>MSKJECH | JECH Interrupt Flag Enable<br>Specifies whether completion of an injected conversion of a set of inputs flags the JECH interrupt.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |
| 1<br>MSKEOC | EOC Interrupt Flag Enable<br>Specifies whether completion of a normal conversion flags the EOC interrupt.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |
| 0 | ECH Interrupt Flag Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| MSKECH | Specifies whether completion of a normal conversion of a set of inputs flags the ECH interrupt.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |

### 57.6.2.9 EOC Interrupt Enable For Precision Inputs (CIMR0)

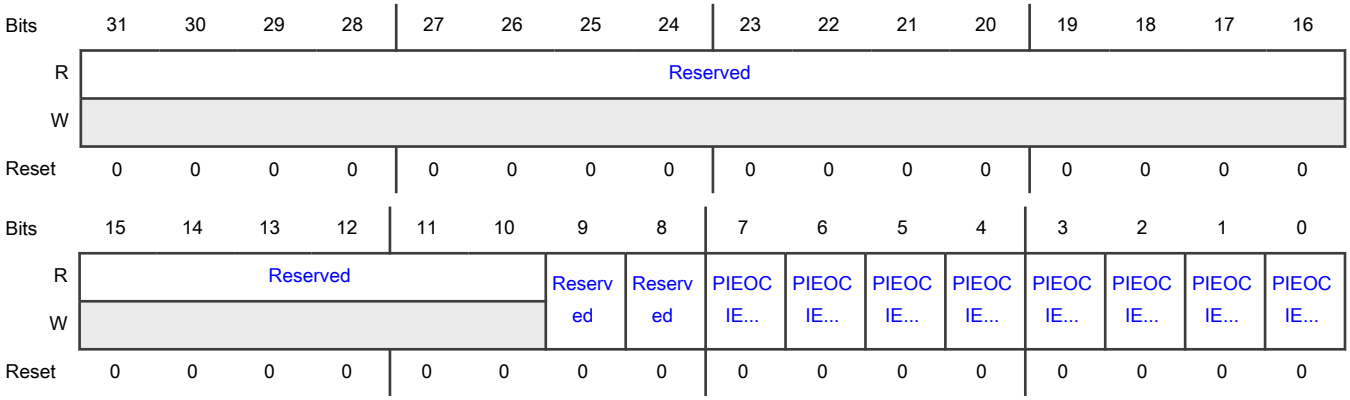
#### Offset

| Register | Offset |
|---|---|
| CIMR0 | 24h |

#### Function

Specifies whether a completed conversion of a precision input flags an EOC interrupt.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>— | Reserved |
| 8<br>— | Reserved |
| 7 | Precision Input EOC Interrupt Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| PIEOCIEN7 | Specifies whether a completed conversion of precision input *n* flags an EOC interrupt.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |
| 6<br>PIEOCIEN6 | Precision Input EOC Interrupt Enable<br>Specifies whether a completed conversion of precision input *n* flags an EOC interrupt.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |
| 5<br>PIEOCIEN5 | Precision Input EOC Interrupt Enable<br>Specifies whether a completed conversion of precision input *n* flags an EOC interrupt.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |
| 4<br>PIEOCIEN4 | Precision Input EOC Interrupt Enable<br>Specifies whether a completed conversion of precision input *n* flags an EOC interrupt.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |
| 3<br>PIEOCIEN3 | Precision Input EOC Interrupt Enable<br>Specifies whether a completed conversion of precision input *n* flags an EOC interrupt.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |
| 2<br>PIEOCIEN2 | Precision Input EOC Interrupt Enable<br>Specifies whether a completed conversion of precision input *n* flags an EOC interrupt.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |
| 1<br>PIEOCIEN1 | Precision Input EOC Interrupt Enable<br>Specifies whether a completed conversion of precision input *n* flags an EOC interrupt.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |
| 0<br>PIEOCIEN0 | Precision Input EOC Interrupt Enable<br>Specifies whether a completed conversion of precision input *n* flags an EOC interrupt.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |

### 57.6.2.10 EOC Interrupt Enable For Standard Inputs (CIMR1)

#### Offset

| Register | Offset |
|---|---|
| CIMR1 | 28h |

#### Function
Specifies whether a completed conversion of a standard input flags an EOC interrupt.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | SIEOCIE... | SIEOCIE... | SIEOCIE... | SIEOCIE... | SIEOCIE... | SIEOCIE... | SIEOCIE... | SIEOCIE... |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

#### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>SIEOCIENn | Standard Input EOC Interrupt Enable<br>Specifies whether a completed conversion of standard input *n* flags an EOC interrupt.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |

### 57.6.2.11 EOC Interrupt Enable For External Inputs (CIMR2)

#### Offset

| Register | Offset |
|---|---|
| CIMR2 | 2Ch |

#### Function
Specifies whether a completed conversion of an external input flags an EOC interrupt.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| ADC_0 | CIMR2 | — |
| ADC_1 | CIMR2 | — |
| ADC_2 | — | CIMR2 |

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... | EIEOCIE... |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>EIEOCIENn | External Input EOC Interrupt Enable<br>Specifies whether a completed conversion of external input *n* flags an EOC interrupt.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |

## 57.6.2.12 Analog Watchdog Threshold Interrupt Status (WTISR)

### Offset

| Register | Offset |
|---|---|
| WTISR | 30h |

### Function

Contains flags that indicate the result of a comparison between the conversion result and the threshold value defined in the analog watchdog.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | HAWIF16 | LAWIF16 | HAWIF15 | LAWIF15 | HAWIF14 | LAWIF14 | HAWIF13 | LAWIF13 | HAWIF12 | LAWIF12 | HAWIF11 | LAWIF11 | HAWIF10 | LAWIF10 | HAWIF9 | LAWIF9 |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | HAWIF8 | LAWIF8 | HAWIF7 | LAWIF7 | HAWIF6 | LAWIF6 | HAWIF5 | LAWIF5 | HAWIF4 | LAWIF4 | HAWIF3 | LAWIF3 | HAWIF2 | LAWIF2 | HAWIF1 | LAWIF1 |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>HAWIF16 | High Analog Watchdog Interrupt Flag 16<br>Indicates whether the conversion result is higher than the high threshold value of the analog watchdog 16.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is lower than the specified high threshold<br>1b - Conversion result is higher than the specified high threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 30<br>LAWIF16 | Low Analog Watchdog Interrupt Flag 16<br>Indicates whether the conversion result is lower than the low threshold value of the analog watchdog 16.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is greater than the specified low threshold<br>1b - Conversion result is lower than the specified low threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 29 | High Analog Watchdog Interrupt Flag 15 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| HAWIF15 | Indicates whether the conversion result is higher than the high threshold value of the analog watchdog 15.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is lower than the specified high threshold<br>1b - Conversion result is higher than the specified high threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 28<br>LAWIF15 | Low Analog Watchdog Interrupt Flag 15<br>Indicates whether the conversion result is lower than the low threshold value of the analog watchdog 15.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is greater than the specified low threshold<br>1b - Conversion result is lower than the specified low threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 27<br>HAWIF14 | High Analog Watchdog Interrupt Flag 14<br>Indicates whether the conversion result is higher than the high threshold value of the analog watchdog 14.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is lower than the specified high threshold<br>1b - Conversion result is higher than the specified high threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 26<br>LAWIF14 | Low Analog Watchdog Interrupt Flag 14<br>Indicates whether the conversion result is lower than the low threshold value of the analog watchdog 14. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is greater than the specified low threshold<br>1b - Conversion result is lower than the specified low threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 25<br>HAWIF13 | High Analog Watchdog Interrupt Flag 13<br>Indicates whether the conversion result is higher than the high threshold value of the analog watchdog 13.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is lower than the specified high threshold<br>1b - Conversion result is higher than the specified high threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 24<br>LAWIF13 | Low Analog Watchdog Interrupt Flag 13<br>Indicates whether the conversion result is lower than the low threshold value of the analog watchdog 13.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is greater than the specified low threshold<br>1b - Conversion result is lower than the specified low threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 23<br>HAWIF12 | High Analog Watchdog Interrupt Flag 12<br>Indicates whether the conversion result is higher than the high threshold value of the analog watchdog 12. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is lower than the specified high threshold<br>1b - Conversion result is higher than the specified high threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 22<br>LAWIF12 | Low Analog Watchdog Interrupt Flag 12<br>Indicates whether the conversion result is lower than the low threshold value of the analog watchdog 12.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is greater than the specified low threshold<br>1b - Conversion result is lower than the specified low threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 21<br>HAWIF11 | High Analog Watchdog Interrupt Flag 11<br>Indicates whether the conversion result is higher than the high threshold value of the analog watchdog 11.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is lower than the specified high threshold<br>1b - Conversion result is higher than the specified high threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 20<br>LAWIF11 | Low Analog Watchdog Interrupt Flag 11<br>Indicates whether the conversion result is lower than the low threshold value of the analog watchdog 11. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is greater than the specified low threshold<br>1b - Conversion result is lower than the specified low threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 19<br>HAWIF10 | High Analog Watchdog Interrupt Flag 10<br>Indicates whether the conversion result is higher than the high threshold value of the analog watchdog 10.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is lower than the specified high threshold<br>1b - Conversion result is higher than the specified high threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 18<br>LAWIF10 | Low Analog Watchdog Interrupt Flag 10<br>Indicates whether the conversion result is lower than the low threshold value of the analog watchdog 10.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is greater than the specified low threshold<br>1b - Conversion result is lower than the specified low threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 17<br>HAWIF9 | High Analog Watchdog Interrupt Flag 9<br>Indicates whether the conversion result is higher than the high threshold value of the analog watchdog 9. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is lower than the specified high threshold<br>1b - Conversion result is higher than the specified high threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 16<br>LAWIF9 | Low Analog Watchdog Interrupt Flag 9<br>Indicates whether the conversion result is lower than the low threshold value of the analog watchdog 9.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is greater than the specified low threshold<br>1b - Conversion result is lower than the specified low threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 15<br>HAWIF8 | High Analog Watchdog Interrupt Flag 8<br>Indicates whether the conversion result is higher than the high threshold value of the analog watchdog 8.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is lower than the specified high threshold<br>1b - Conversion result is higher than the specified high threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 14<br>LAWIF8 | Low Analog Watchdog Interrupt Flag 8<br>Indicates whether the conversion result is lower than the low threshold value of the analog watchdog 8.<br>**NOTE**<br>This field behaves differently for register reads and writes. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When reading<br>0b - Conversion result is greater than the specified low threshold<br>1b - Conversion result is lower than the specified low threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 13<br>HAWIF7 | High Analog Watchdog Interrupt Flag 7<br>Indicates whether the conversion result is higher than the high threshold value of the analog watchdog 7.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is lower than the specified high threshold<br>1b - Conversion result is higher than the specified high threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 12<br>LAWIF7 | Low Analog Watchdog Interrupt Flag 7<br>Indicates whether the conversion result is lower than the low threshold value of the analog watchdog 7.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is greater than the specified low threshold<br>1b - Conversion result is lower than the specified low threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 11<br>HAWIF6 | High Analog Watchdog Interrupt Flag 6<br>Indicates whether the conversion result is higher than the high threshold value of the analog watchdog 6.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Conversion result is lower than the specified high threshold<br>1b - Conversion result is higher than the specified high threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 10<br>LAWIF6 | Low Analog Watchdog Interrupt Flag 6<br>Indicates whether the conversion result is lower than the low threshold value of the analog watchdog 6.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is greater than the specified low threshold<br>1b - Conversion result is lower than the specified low threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 9<br>HAWIF5 | High Analog Watchdog Interrupt Flag 5<br>Indicates whether the conversion result is higher than the high threshold value of the analog watchdog 5.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is lower than the specified high threshold<br>1b - Conversion result is higher than the specified high threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 8<br>LAWIF5 | Low Analog Watchdog Interrupt Flag 5<br>Indicates whether the conversion result is lower than the low threshold value of the analog watchdog 5.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is greater than the specified low threshold |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Conversion result is lower than the specified low threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 7<br>HAWIF4 | High Analog Watchdog Interrupt Flag 4<br>Indicates whether the conversion result is higher than the high threshold value of the analog watchdog 4.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is lower than the specified high threshold<br>1b - Conversion result is higher than the specified high threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 6<br>LAWIF4 | Low Analog Watchdog Interrupt Flag 4<br>Indicates whether the conversion result is lower than the low threshold value of the analog watchdog 4.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is greater than the specified low threshold<br>1b - Conversion result is lower than the specified low threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 5<br>HAWIF3 | High Analog Watchdog Interrupt Flag 3<br>Indicates whether the conversion result is higher than the high threshold value of the analog watchdog 3.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is lower than the specified high threshold<br>1b - Conversion result is higher than the specified high threshold<br>When writing |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No effect<br>1b - Clears flag |
| 4<br>LAWIF3 | Low Analog Watchdog Interrupt Flag 3<br>Indicates whether the conversion result is lower than the low threshold value of the analog watchdog 3.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is greater than the specified low threshold<br>1b - Conversion result is lower than the specified low threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 3<br>HAWIF2 | High Analog Watchdog Interrupt Flag 2<br>Indicates whether the conversion result is higher than the high threshold value of the analog watchdog 2.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is lower than the specified high threshold<br>1b - Conversion result is higher than the specified high threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 2<br>LAWIF2 | Low Analog Watchdog Interrupt Flag 2<br>Indicates whether the conversion result is lower than the low threshold value of the analog watchdog 2.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is greater than the specified low threshold<br>1b - Conversion result is lower than the specified low threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 1<br>HAWIF1 | High Analog Watchdog Interrupt Flag 1<br>Indicates whether the conversion result is higher than the high threshold value of the analog watchdog 1.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is lower than the specified high threshold<br>1b - Conversion result is higher than the specified high threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 0<br>LAWIF1 | Low Analog Watchdog Interrupt Flag 1<br>Indicates whether the conversion result is lower than the low threshold value of the analog watchdog 1.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion result is greater than the specified low threshold<br>1b - Conversion result is lower than the specified low threshold<br>When writing<br>0b - No effect<br>1b - Clears flag |

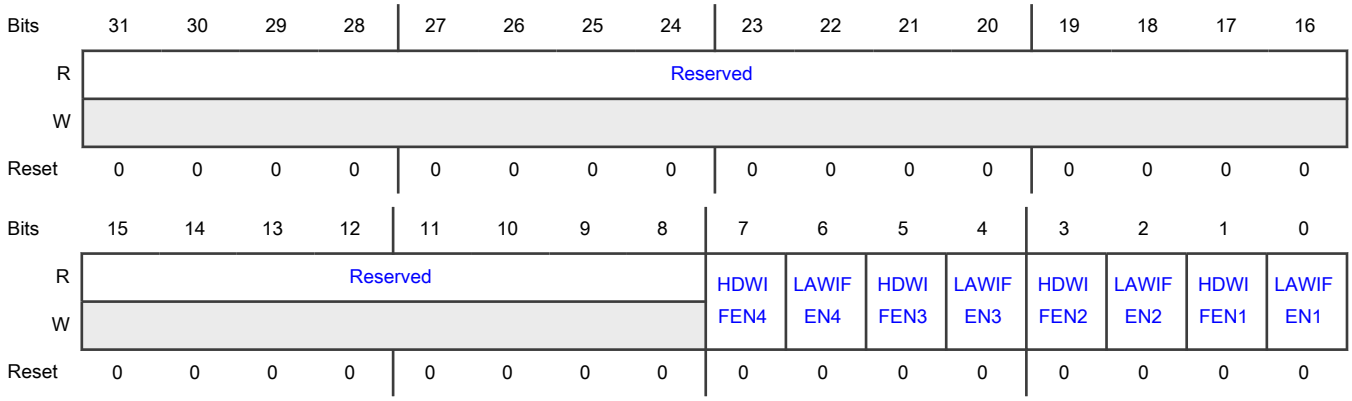
### 57.6.2.13 Analog Watchdog Threshold Interrupt Enable (WTIMR)

**Offset**

| Register | Offset |
|---|---|
| WTIMR | 34h |

**Function**
Enables the interrupt for each analog watchdog threshold value.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7<br>HDWIFEN4 | High Data Watchdog Interrupt Flag Enable n<br>Enables flagging of an interrupt when a conversion result is higher than the high threshold value of the analog watchdog *n*.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |
| 6<br>LAWIFEN4 | Low Analog Watchdog Interrupt Flag Enable n<br>Enables flagging of an interrupt when a conversion result is lower than the low threshold value of analog watchdog *n*.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |
| 5<br>HDWIFEN3 | High Data Watchdog Interrupt Flag Enable n<br>Enables flagging of an interrupt when a conversion result is higher than the high threshold value of the analog watchdog *n*.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |
| 4<br>LAWIFEN3 | Low Analog Watchdog Interrupt Flag Enable n<br>Enables flagging of an interrupt when a conversion result is lower than the low threshold value of analog watchdog *n*.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |
| 3 | High Data Watchdog Interrupt Flag Enable n |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| HDWIFEN2 | Enables flagging of an interrupt when a conversion result is higher than the high threshold value of the analog watchdog *n*.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |
| 2<br>LAWIFEN2 | Low Analog Watchdog Interrupt Flag Enable n<br>Enables flagging of an interrupt when a conversion result is lower than the low threshold value of analog watchdog *n*.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |
| 1<br>HDWIFEN1 | High Data Watchdog Interrupt Flag Enable n<br>Enables flagging of an interrupt when a conversion result is higher than the high threshold value of the analog watchdog *n*.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |
| 0<br>LAWIFEN1 | Low Analog Watchdog Interrupt Flag Enable n<br>Enables flagging of an interrupt when a conversion result is lower than the low threshold value of analog watchdog *n*.<br>0b - Interrupt is not flagged<br>1b - Interrupt is flagged |

### 57.6.2.14 Direct Memory Access Configuration (DMAE)

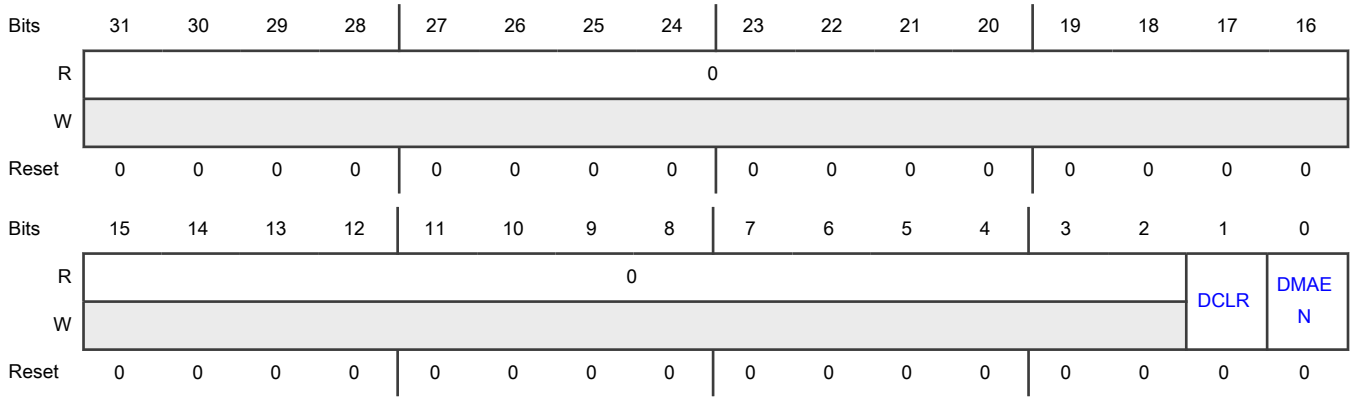
**Offset**

| Register | Offset |
|---|---|
| DMAE | 40h |

**Function**

Configures the DMA feature.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>DCLR | DMA Clear Request<br>Selects one of the following events to clear a DMA request:<br>• DMA controller acknowledges the request.<br>• Application reads the conversion data register.<br>0b - DMA controller acknowledges the request<br>1b - Conversion data register is read |
| 0<br>DMAEN | DMA Enable<br>Enables DMA.<br>0b - Disable<br>1b - Enable |

## 57.6.2.15 DMA Request Enable For Precision Inputs (DMAR0)

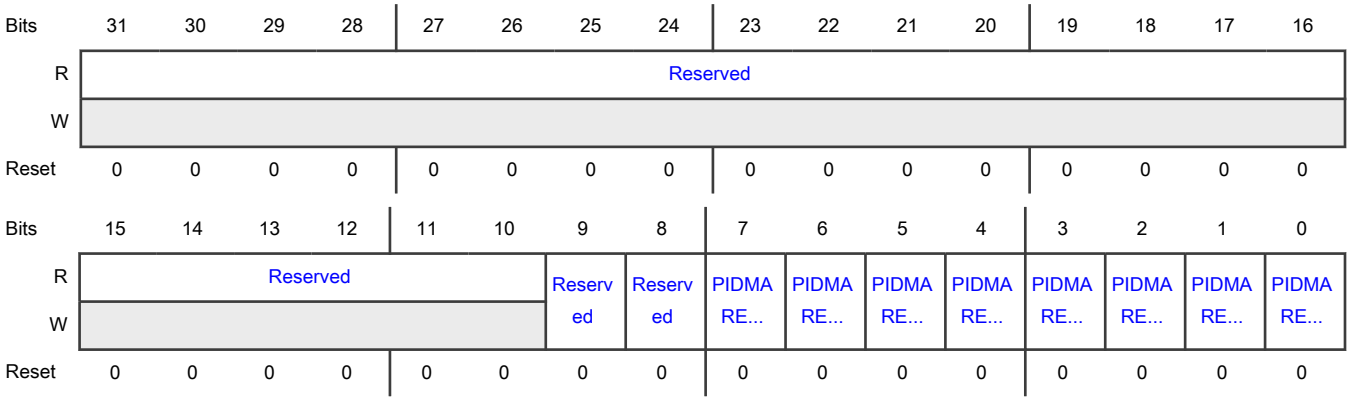
### Offset

| Register | Offset |
|---|---|
| DMAR0 | 44h |

### Function

Selects the precision inputs that trigger a DMA request after conversion is complete.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>— | Reserved |
| 8<br>— | Reserved |
| 7<br>PIDMAREN7 | Precision Input DMA Request Enable 7<br>Specifies whether a DMA request is triggered after conversion is complete for precision input 7.<br>0b - Not triggered<br>1b - Triggered |
| 6<br>PIDMAREN6 | Precision Input DMA Request Enable 6<br>Specifies whether a DMA request is triggered after conversion is complete for precision input 6.<br>0b - Not triggered<br>1b - Triggered |
| 5<br>PIDMAREN5 | Precision Input DMA Request Enable 5<br>Specifies whether a DMA request is triggered after conversion is complete for precision input 5.<br>0b - Not triggered<br>1b - Triggered |
| 4<br>PIDMAREN4 | Precision Input DMA Request Enable 4<br>Specifies whether a DMA request is triggered after conversion is complete for precision input 4.<br>0b - Not triggered |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Triggered |
| 3<br>PIDMAREN3 | Precision Input DMA Request Enable 3<br>Specifies whether a DMA request is triggered after conversion is complete for precision input 3.<br>0b - Not triggered<br>1b - Triggered |
| 2<br>PIDMAREN2 | Precision Input DMA Request Enable 2<br>Specifies whether a DMA request is triggered after conversion is complete for precision input 2.<br>0b - Not triggered<br>1b - Triggered |
| 1<br>PIDMAREN1 | Precision Input DMA Request Enable 1<br>Specifies whether a DMA request is triggered after conversion is complete for precision input 1.<br>0b - Not triggered<br>1b - Triggered |
| 0<br>PIDMAREN0 | Precision Input DMA Request Enable 0<br>Specifies whether a DMA request is triggered after conversion is complete for precision input 0.<br>0b - Not triggered<br>1b - Triggered |

### 57.6.2.16 DMA Request Enable For Standard Inputs (DMAR1)

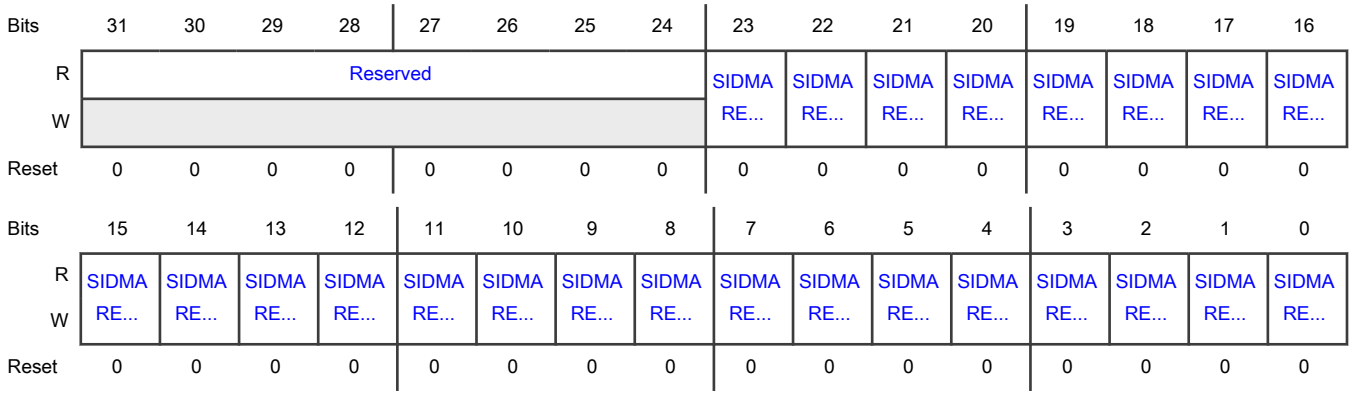
**Offset**

| Register | Offset |
|---|---|
| DMAR1 | 48h |

**Function**

Selects the standard inputs that trigger a DMA request after conversion is complete.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>SIDMARENn | Standard Input DMA Request Enable n<br>Specifies whether a DMA request is triggered after conversion is complete for standard input *n*.<br>0b - DMA request is not triggered<br>1b - DMA is request triggered |

## 57.6.2.17 DMA Request Enable For External Inputs (DMAR2)

### Offset

| Register | Offset |
|---|---|
| DMAR2 | 4Ch |

### Function

Selects the external inputs that trigger a DMA request after conversion is complete.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| ADC_0 | DMAR2 | — |
| ADC_1 | DMAR2 | — |
| ADC_2 | — | DMAR2 |

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... | EIDMARE... |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>EIDMARENn | External Input DMA Request Enable n<br>Specifies whether a DMA request is triggered after external input *n* is converted.<br>0b - DMA request is not triggered<br>1b - DMA request is triggered |

## 57.6.2.18 Analog Watchdog Threshold Values (THRHLR0 - THRHLR3)

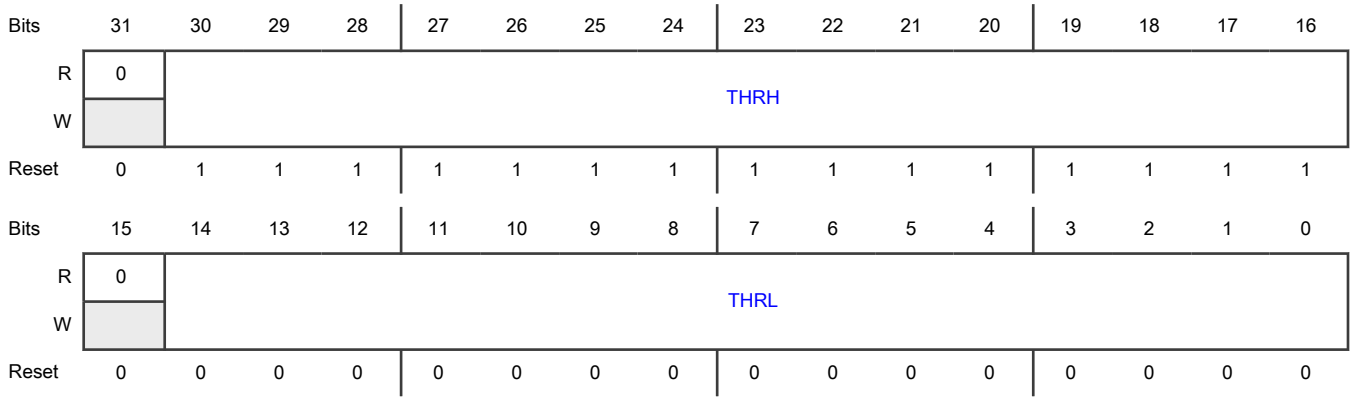
### Offset

| Register | Offset |
|---|---|
| THRHLR0 | 60h |
| THRHLR1 | 64h |
| THRHLR2 | 68h |
| THRHLR3 | 6Ch |

### Function

Specifies limits for the valid operating range of a monitored input.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-16<br>THRH | High Threshold Value<br>If enabled, flags an interrupt when the converted data of the input is higher than this value. |
| 15<br>— | Reserved |
| 14-0<br>THRL | Low Threshold Value<br>If enabled, flags an interrupt when the converted data of the input is lower than this value. |

## 57.6.2.19 Presampling Control (PSCR)

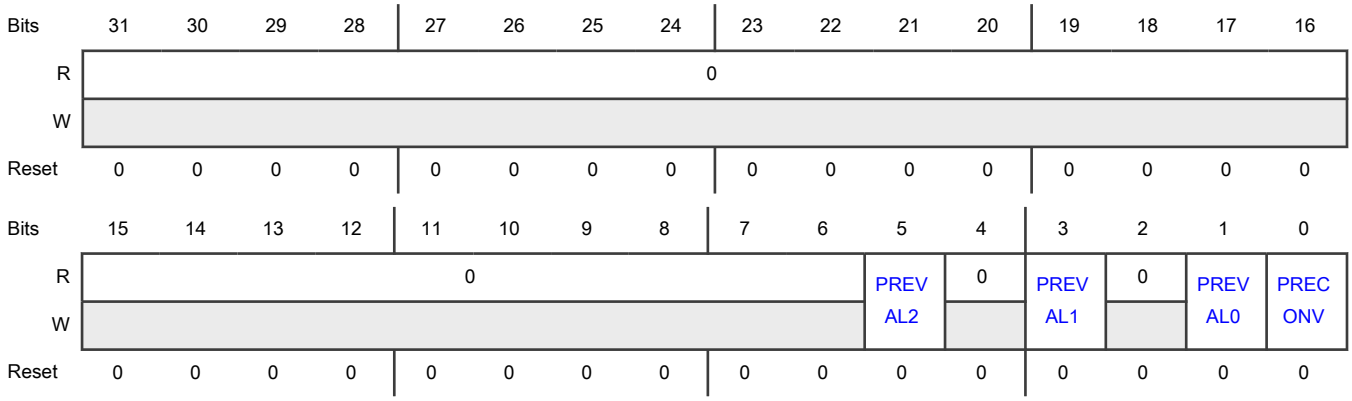
### Offset

| Register | Offset |
|---|---|
| PSCR | 80h |

### Function

Configures ADC to presample an internal voltage before the actual input is converted.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved |
| 5<br>PREVAL2 | Presampling Voltage Select For External Inputs<br>Selects the internal voltage that is presampled for external inputs.<br>**NOTE**<br>This field is not supported in every instance. The following table includes only supported registers.<br>0b - VREFL<br>1b - VREFH |
| 4<br>— | Reserved |
| 3<br>PREVAL1 | Presampling Voltage Select For Standard Inputs<br>Selects the internal voltage that is presampled for standard inputs and for the temperature sensor.<br>0b - VREFL<br>1b - VREFH |
| 2 | Reserved |

| Instance | Field supported in | Field not supported in |
|---|---|---|
| ADC_0 | PSCR | — |
| ADC_1 | PSCR | — |
| ADC_2 | — | PSCR |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 1<br>PREVAL0 | Presampling Voltage Select For Precision Inputs<br>Selects the internal voltage that is presampled for precision inputs.<br>0b - VREFL<br>1b - VREFH |
| 0<br>PRECONV | Convert Presampled Value<br>Specifies whether presampling is followed by the comparison. If enabled, presampling is followed by conversion and the result is written to the conversion data register of the selected input.<br>0b - No conversion after presampling<br>1b - Presampling is followed by conversion |

### 57.6.2.20 Presampling Enable For Precision Inputs (PSR0)

#### Offset

| Register | Offset |
|---|---|
| PSR0 | 84h |

#### Function
Enables presampling for each precision input.

#### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | Reserved | Reserved | PRES7 | PRES6 | PRES5 | PRES4 | PRES3 | PRES2 | PRES1 | PRES0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>— | Reserved |
| 8<br>— | Reserved |
| 7<br>PRES7 | Presampling Enable n<br>Enables presampling for precision input *n*.<br>0b - Disable<br>1b - Enable |
| 6<br>PRES6 | Presampling Enable n<br>Enables presampling for precision input *n*.<br>0b - Disable<br>1b - Enable |
| 5<br>PRES5 | Presampling Enable n<br>Enables presampling for precision input *n*.<br>0b - Disable<br>1b - Enable |
| 4<br>PRES4 | Presampling Enable n<br>Enables presampling for precision input *n*.<br>0b - Disable<br>1b - Enable |
| 3<br>PRES3 | Presampling Enable n<br>Enables presampling for precision input *n*.<br>0b - Disable<br>1b - Enable |
| 2<br>PRES2 | Presampling Enable n<br>Enables presampling for precision input *n*.<br>0b - Disable<br>1b - Enable |
| 1 | Presampling Enable n |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| PRES1 | Enables presampling for precision input *n*.<br>0b - Disable<br>1b - Enable |
| 0<br>PRES0 | Presampling Enable n<br>Enables presampling for precision input *n*.<br>0b - Disable<br>1b - Enable |

### 57.6.2.21 Presampling Enable For Standard Inputs (PSR1)

#### Offset

| Register | Offset |
|---|---|
| PSR1 | 88h |

#### Function

Enables presampling for each standard input.

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0 | Presampling Enable n |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| PRESn | Enables presampling for standard input *n*.<br>0b - Disable<br>1b - Enable |

### 57.6.2.22 Presampling Enable For External Inputs (PSR2)

#### Offset

| Register | Offset |
|---|---|
| PSR2 | 8Ch |

#### Function

Enables presampling for each external input.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| ADC_0 | PSR2 | — |
| ADC_1 | PSR2 | — |
| ADC_2 | — | PSR2 |

#### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>PRESn | Presampling Enable n<br>Enables presampling for external input *n*.<br>0b - Disable<br>1b - Enable |

## 57.6.2.23 Conversion Timing For Precision Inputs (CTR0)

### Offset

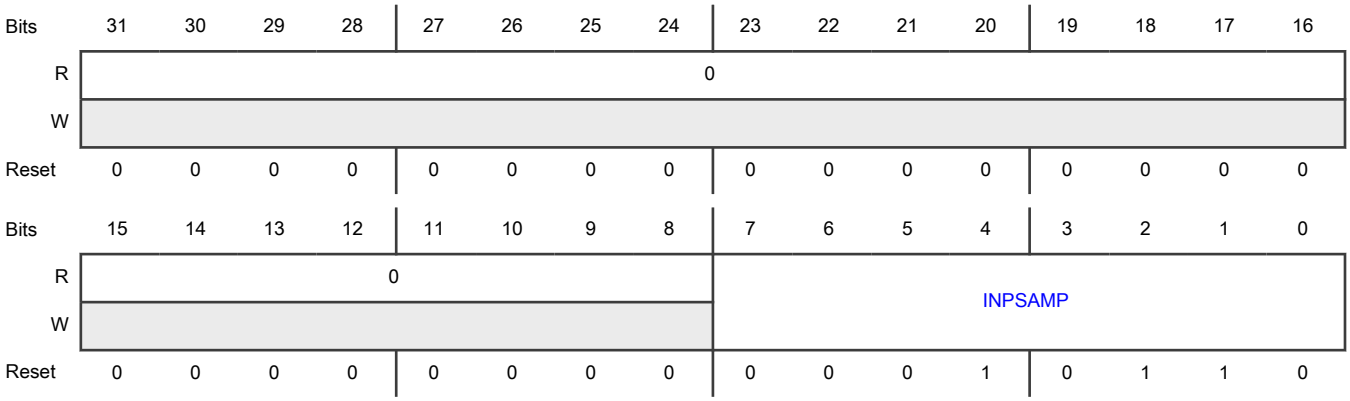
| Register | Offset |
|---|---|
| CTR0 | 94h |

### Function

Specifies duration, in terms of the number of conversion clock cycles, of the sampling of precision inputs.

The conversion clock frequency depends on the configuration of MCR[ADCLKSEL].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>INPSAMP | Input Sample Cycles<br>Specifies the sample duration in terms of conversion clock cycles. The minimum value is 8. Specifying a lower value automatically forces a value of 8. |

### 57.6.2.24 Conversion Timing For Standard Inputs (CTR1)

#### Offset

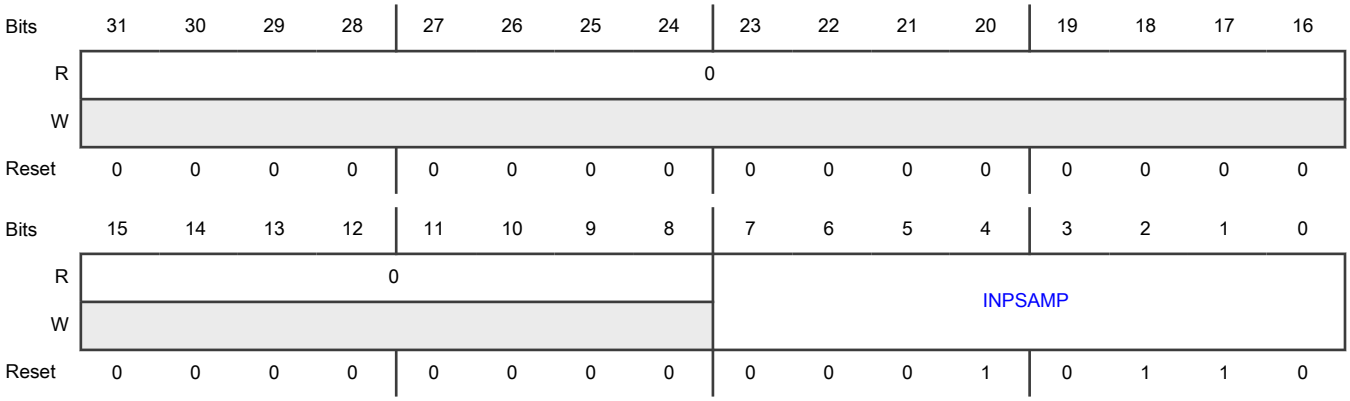
| Register | Offset |
|---|---|
| CTR1 | 98h |

#### Function

Specifies the duration, in terms of the number of conversion clock cycles, of the sampling of standard inputs or the temperature sensor.

The conversion clock frequency depends on the configuration of MCR[ADCLKSEL].

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>INPSAMP | Input Sample Cycles<br>Specifies the sample duration in terms of conversion clock cycles. The minimum value is 8. Specifying a lower value automatically forces a value of 8. |

### 57.6.2.25 Conversion Timing For External Inputs (CTR2)

#### Offset

| Register | Offset |
|---|---|
| CTR2 | 9Ch |

#### Function

Specifies the number of conversion clock cycles when the external inputs are sampled.

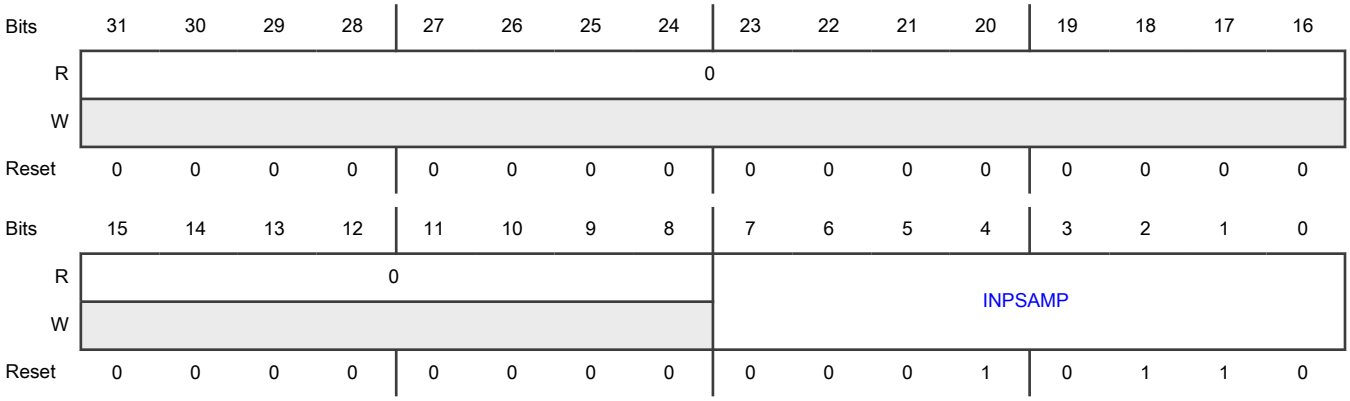
The conversion clock frequency depends on the configuration of MCR[ADCLKSEL].

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| ADC_0 | CTR2 | — |
| ADC_1 | CTR2 | — |
| ADC_2 | — | CTR2 |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>INPSAMP | Input Sample Cycles<br>Specifies the sample duration in terms of conversion clock cycles. The minimum acceptable value is 8. Specifying a lower value automatically forces a value of 8. |

## 57.6.2.26 Normal Conversion Enable For Precision Inputs (NCMR0)

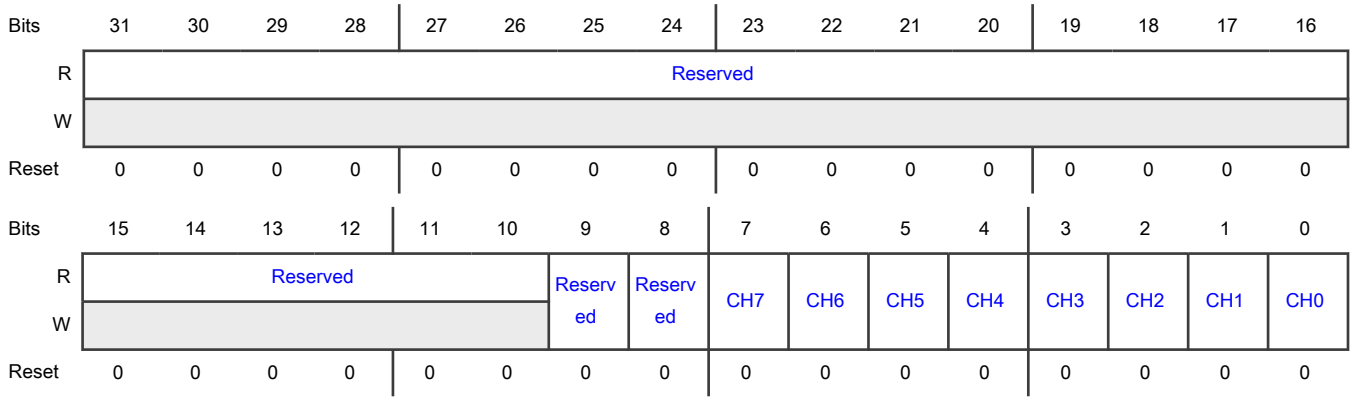
### Offset

| Register | Offset |
|---|---|
| NCMR0 | A4h |

### Function

Selects the precision inputs to be converted during a normal conversion.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>— | Reserved |
| 8<br>— | Reserved |
| 7<br>CH7 | Precision Input To Be Converted<br>Selects precision input *n* for conversion.<br>0b - Input is not selected<br>1b - Input is selected |
| 6<br>CH6 | Precision Input To Be Converted<br>Selects precision input *n* for conversion.<br>0b - Input is not selected<br>1b - Input is selected |
| 5<br>CH5 | Precision Input To Be Converted<br>Selects precision input *n* for conversion.<br>0b - Input is not selected<br>1b - Input is selected |
| 4<br>CH4 | Precision Input To Be Converted<br>Selects precision input *n* for conversion.<br>0b - Input is not selected |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Input is selected |
| 3<br>CH3 | Precision Input To Be Converted<br>Selects precision input *n* for conversion.<br>0b - Input is not selected<br>1b - Input is selected |
| 2<br>CH2 | Precision Input To Be Converted<br>Selects precision input *n* for conversion.<br>0b - Input is not selected<br>1b - Input is selected |
| 1<br>CH1 | Precision Input To Be Converted<br>Selects precision input *n* for conversion.<br>0b - Input is not selected<br>1b - Input is selected |
| 0<br>CH0 | Precision Input To Be Converted<br>Selects precision input *n* for conversion.<br>0b - Input is not selected<br>1b - Input is selected |

### 57.6.2.27 Normal Conversion Enable For Standard Inputs (NCMR1)

**Offset**

| Register | Offset |
|---|---|
| NCMR1 | A8h |

**Function**

Selects the standard inputs to be converted during a normal conversion.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | CH55 | CH54 | CH53 | CH52 | CH51 | CH50 | CH49 | CH48 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CH47 | CH46 | CH45 | CH44 | CH43 | CH42 | CH41 | CH40 | CH39 | CH38 | CH37 | CH36 | CH35 | CH34 | CH33 | CH32 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>CHn | Standard Input To Be Converted<br>Selects standard input *n* for conversion.<br>0b - Input *n* is not selected<br>1b - Input *n* is selected |

## 57.6.2.28 Normal Conversion Enable For External Inputs (NCMR2)

### Offset

| Register | Offset |
|---|---|
| NCMR2 | ACh |

### Function

Selects the external inputs to be converted during a normal conversion.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| ADC_0 | NCMR2 | — |
| ADC_1 | NCMR2 | — |
| ADC_2 | — | NCMR2 |

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CH95 | CH94 | CH93 | CH92 | CH91 | CH90 | CH89 | CH88 | CH87 | CH86 | CH85 | CH84 | CH83 | CH82 | CH81 | CH80 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CH79 | CH78 | CH77 | CH76 | CH75 | CH74 | CH73 | CH72 | CH71 | CH70 | CH69 | CH68 | CH67 | CH66 | CH65 | CH64 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>CHn | External Input To Be Converted<br>Selects external input *n* for conversion.<br>0b - Input *n* is not selected<br>1b - Input *n* is selected |

## 57.6.2.29 Injected Conversion Enable For Precision Inputs (JCMR0)

### Offset

| Register | Offset |
|---|---|
| JCMR0 | B4h |

### Function

Selects the precision inputs to be converted during an injected conversion.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | Reserved | Reserved | CH7 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1 | CH0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>— | Reserved |
| 8<br>— | Reserved |
| 7<br>CH7 | Precision Input To Be Converted<br>Selects precision input 7 for conversion.<br>0b - Input 7 is not selected<br>1b - Input 7 is selected |
| 6<br>CH6 | Precision Input To Be Converted<br>Selects precision input 6 for conversion.<br>0b - Input 6 is not selected<br>1b - Input 6 is selected |
| 5<br>CH5 | Precision Input To Be Converted<br>Selects precision input 5 for conversion.<br>0b - Input 5 is not selected<br>1b - Input 5 is selected |
| 4<br>CH4 | Precision Input To Be Converted<br>Selects precision input 4 for conversion.<br>0b - Input 4 is not selected<br>1b - Input 4 is selected |
| 3<br>CH3 | Precision Input To Be Converted<br>Selects precision input 3 for conversion.<br>0b - Input 3 is not selected<br>1b - Input 3 is selected |
| 2<br>CH2 | Precision Input To Be Converted<br>Selects precision input 2 for conversion.<br>0b - Input 2 is not selected<br>1b - Input 2 is selected |
| 1 | Precision Input To Be Converted |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| CH1 | Selects precision input 1 for conversion.<br>0b - Input 1 is not selected<br>1b - Input 1 is selected |
| 0<br>CH0 | Precision Input To Be Converted<br>Selects precision input 0 for conversion.<br>0b - Input 0 is not selected<br>1b - Input 0 is selected |

### 57.6.2.30 Injected Conversion Enable For Standard Inputs (JCMR1)

#### Offset

| Register | Offset |
|---|---|
| JCMR1 | B8h |

#### Function

Selects the standard inputs to be converted during an injected conversion.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | CH55 | CH54 | CH53 | CH52 | CH51 | CH50 | CH49 | CH48 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CH47 | CH46 | CH45 | CH44 | CH43 | CH42 | CH41 | CH40 | CH39 | CH38 | CH37 | CH36 | CH35 | CH34 | CH33 | CH32 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0 | Standard Input To Be Converted |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| CHn | Selects standard input *n* for conversion.<br>0b - Input *n* is not selected<br>1b - Input *n* is selected |

### 57.6.2.31 Injected Conversion Enable For External Inputs (JCMR2)

#### Offset

| Register | Offset |
|---|---|
| JCMR2 | BCh |

#### Function

Selects the external inputs to be converted during an injected conversion.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| ADC_0 | JCMR2 | — |
| ADC_1 | JCMR2 | — |
| ADC_2 | — | JCMR2 |

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CH95 | CH94 | CH93 | CH92 | CH91 | CH90 | CH89 | CH88 | CH87 | CH86 | CH85 | CH84 | CH83 | CH82 | CH81 | CH80 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CH79 | CH78 | CH77 | CH76 | CH75 | CH74 | CH73 | CH72 | CH71 | CH70 | CH69 | CH68 | CH67 | CH66 | CH65 | CH64 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>CHn | External Input To Be Converted<br>Selects external input *n* for conversion.<br>0b - Input *n* is not selected<br>1b - Input *n* is selected |

## 57.6.2.32 Delay Start Of Data Conversion (DSDR)

### Offset

| Register | Offset |
|---|---|
| DSDR | C4h |

### Function

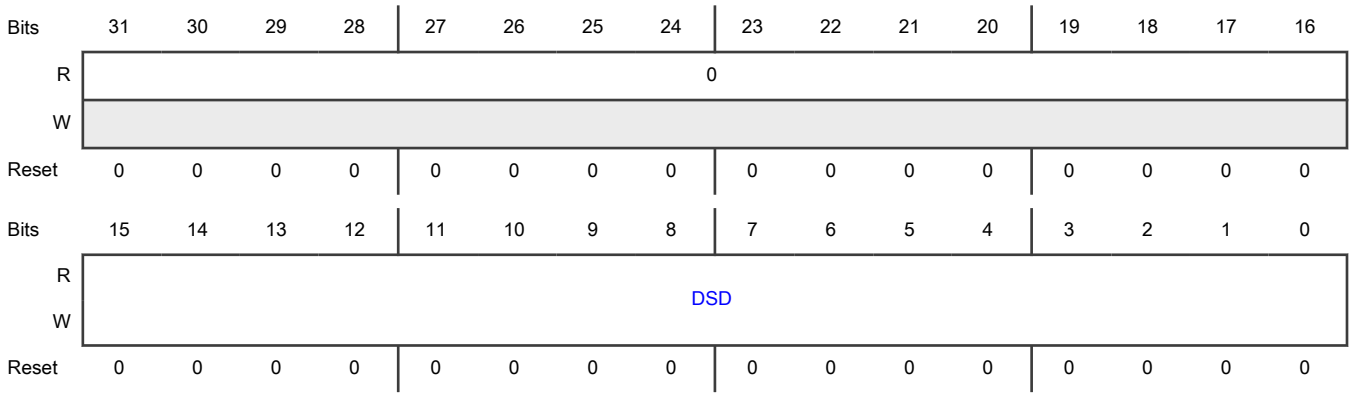
Delays the start of the conversion when another input channel is selected. The necessary delay depends on the device characteristics of, for example, the input channel multiplexer.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| ADC_0 | DSDR | — |
| ADC_1 | DSDR | — |
| ADC_2 | — | DSDR |

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>DSD | Delay<br>Specifies the delay in terms of the number of module clock cycles. |

## 57.6.2.33 Power Down Exit Delay (PDEDR)

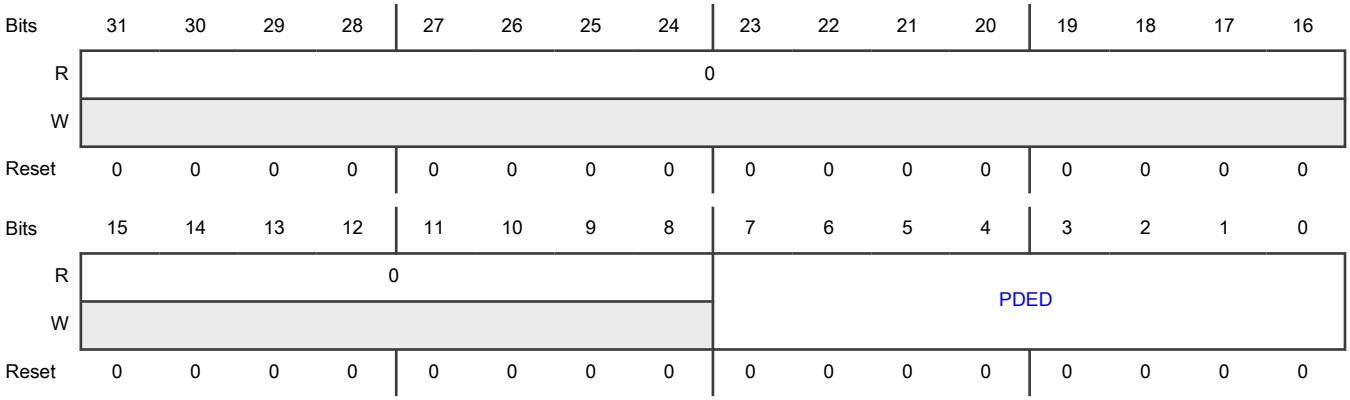
### Offset

| Register | Offset |
|---|---|
| PDEDR | C8h |

### Function

Delays the transition out of Power Down into Idle state to wait for necessary settling times. See the device data sheet for the minimal time between the power-down exit request and when ADC can start a conversion.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>PDED | Delay<br>Specifies a delay in terms of the number of conversion clock cycles. |

### 57.6.2.34 Precision Input n Conversion Data (PCDR0 - PCDR7)

#### Offset

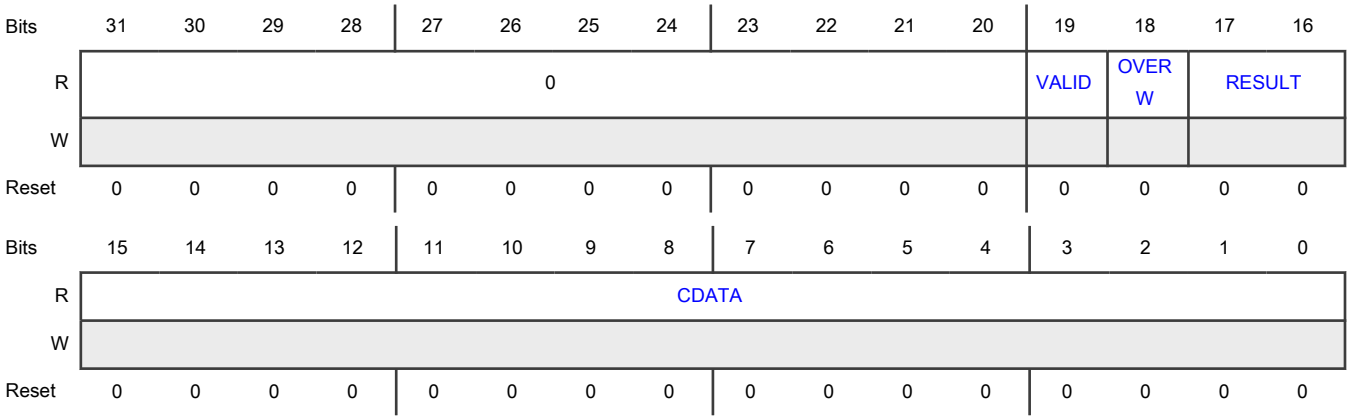
For n = 0 to 7:

| Register | Offset |
|---|---|
| PCDRn | 100h + (n × 4h) |

#### Function

Contains conversion data from precision input *n*.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19<br>VALID | Conversion Data Available<br>Indicates whether new conversion data is available. This field is automatically reset to 0 when the data is read.<br>0b - No unread conversion data<br>1b - Unread conversion data is available |
| 18<br>OVERW | Overwrite Status Flag<br>Indicates whether the previous conversion data was overwritten without having been read, in which case the overwritten data is lost.<br>The ability to overwrite conversion data is controlled by MCR[OWREN].<br>0b - No unread data is overwritten<br>1b - Unread data is overwritten |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 17-16<br>RESULT | Conversion Data Type<br>Indicates the type of trigger that started the conversion for this conversion data.<br>00b - Normal trigger<br>01b - Injected trigger<br>10b - BCTU trigger |
| 15-0<br>CDATA | Conversion Data<br>Contains conversion data from precision input *n*, determined by the successive approximation algorithm. The conversion data is always 15 bits wide, regardless of the conversion resolution selected (CALBISTREG[RESN]).<br>Depending on the value of MCR[WLSIDE], the conversion data can be in bits [14:0] (MCR[WLSIDE] = 0), or in bits [15:1] (MCR[WLSIDE] = 1). |

### 57.6.2.35 Standard Input n Conversion Data (ICDR0 - ICDR23)

#### Offset

For a = 0 to 23:

| Register | Offset |
|---|---|
| ICDRa | 180h + (a × 4h) |

#### Function

Contains conversion data from standard input *n*.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | VALID | OVERW | RESULT | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

**Fields**

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19<br>VALID | Conversion Data Available<br>Indicates whether new conversion data is available. This field is automatically reset to 0 when the data is read.<br>0b - No unread conversion data<br>1b - Unread conversion data is available |
| 18<br>OVERW | Overwrite Status Flag<br>Indicates whether the previous conversion data was overwritten without having been read, in which case the overwritten data is lost.<br>The ability to overwrite conversion data is controlled by MCR[OWREN].<br>0b - No unread data is overwritten<br>1b - Unread data is overwritten |
| 17-16<br>RESULT | Conversion Data Type<br>Indicates the type of trigger that started the conversion for this conversion data.<br>00b - Normal trigger<br>01b - Injected trigger<br>10b - BCTU trigger |
| 15-0<br>CDATA | Conversion Data<br>Contains conversion data from standard input *n*, determined by the SAR algorithm. The conversion data bit count is dependent on the conversion resolution selected (CALBISTREG[RESN]).<br>Depending on the value of MCR[WLSIDE], the conversion data MSB bits start from 14 (MCR[WLSIDE] = 0), or 15 (MCR[WLSIDE] = 1). |

### 57.6.2.36 External Input n Conversion Data (ECDR0 - ECDR31)

**Offset**

For a = 0 to 31:

| Register | Offset |
|---|---|
| ECDRa | 200h + (a × 4h) |

**Function**

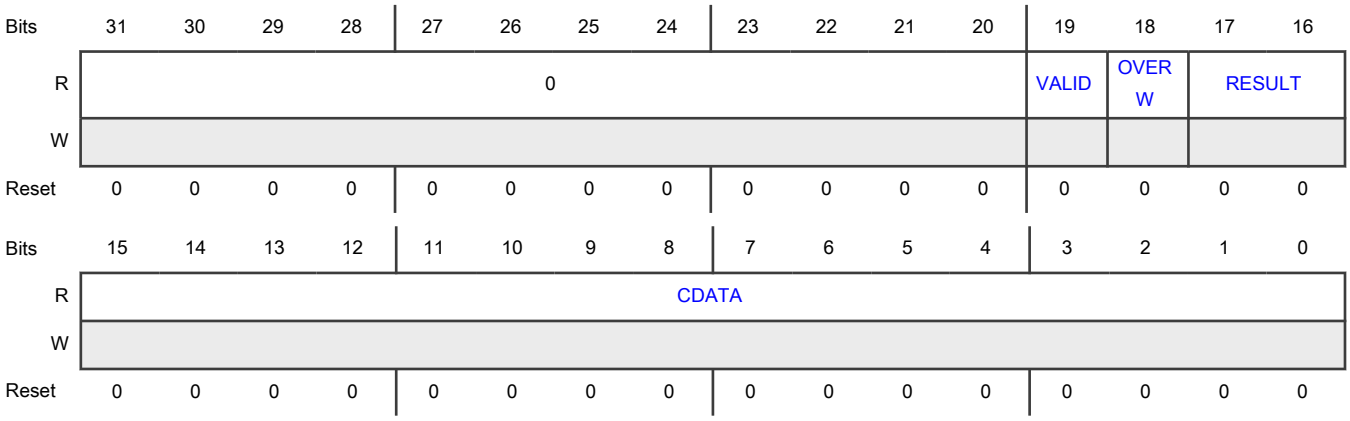
Contains conversion data from external input *n*.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| ADC_0 | ECDR0–ECDR31 | — |
| ADC_1 | ECDR0–ECDR31 | — |
| ADC_2 | — | ECDR0–ECDR31 |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19<br>VALID | Conversion Data Available<br>Indicates whether new conversion data is available. This field is automatically reset to 0 when the data is read.<br>0b - No unread conversion data<br>1b - Unread conversion data is available |
| 18<br>OVERW | Overwrite Status Flag<br>Indicates whether the previous conversion data was overwritten without having been read, in which case the overwritten data is lost.<br>The ability to overwrite conversion data is controlled by MCR[OWREN].<br>0b - No unread data is overwritten<br>1b - Unread data is overwritten |
| 17-16<br>RESULT | Conversion Data Type<br>Indicates the type of trigger that started the conversion for this conversion data. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 00b - Normal trigger<br>01b - Injected trigger<br>10b - BCTU trigger |
| 15-0<br>CDATA | Conversion Data<br>Contains conversion data from external input *n*, determined by the SAR algorithm. The conversion data is always 15 bits wide, regardless of the conversion resolution selected (CALBISTREG[RESN]).<br>Depending on the value of MCR[WLSIDE], the conversion data can be in bits [14:0] (MCR[WLSIDE] = 0), or in bits [15:1] (MCR[WLSIDE] = MCR[WLSIDE]). |

### 57.6.2.37 Channel Analog Watchdog Select For Precision Inputs (CWSELRPI0)

#### Offset

| Register | Offset |
|---|---|
| CWSELRPI0 | 2B0h |

#### Function

Selects the analog watchdog threshold register (THRHLR) that provides limits to monitor precision inputs.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | WSEL_SI0_7 | | 0 | | WSEL_SI0_6 | | 0 | | WSEL_SI0_5 | | 0 | | WSEL_SI0_4 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | WSEL_SI0_3 | | 0 | | WSEL_SI0_2 | | 0 | | WSEL_SI0_1 | | 0 | | WSEL_SI0_0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-28 | Analog Watchdog Selection |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| WSEL_SI0_7 | Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 27-26<br>— | Reserved |
| 25-24<br>WSEL_SI0_6 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 23-22<br>— | Reserved |
| 21-20<br>WSEL_SI0_5 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 19-18<br>— | Reserved |
| 17-16<br>WSEL_SI0_4 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 15-14<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 13-12<br>WSEL_SI0_3 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 11-10<br>— | Reserved |
| 9-8<br>WSEL_SI0_2 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 7-6<br>— | Reserved |
| 5-4<br>WSEL_SI0_1 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 3-2<br>— | Reserved |
| 1-0<br>WSEL_SI0_0 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |

### 57.6.2.38 Channel Analog Watchdog Select For Precision Inputs (CWSELRPI1)

#### Offset

| Register | Offset |
|---|---|
| CWSELRPI1 | 2B4h |

#### Function
Selects the analog watchdog threshold register (THRHLR) that provides limits to monitor precision inputs.

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-4<br>— | Reserved |
| 3-0<br>— | Reserved |

### 57.6.2.39 Channel Analog Watchdog Select For Standard Inputs (CWSELRSI0)

#### Offset

| Register | Offset |
|---|---|
| CWSELRSI0 | 2C0h |

#### Function
Selects the analog watchdog threshold register (THRHLR) that provides limits to monitor the standard inputs.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | WSEL_SI7_0 | | 0 | | WSEL_SI6_0 | | 0 | | WSEL_SI5_0 | | 0 | | WSEL_SI4_0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | WSEL_SI3_0 | | 0 | | WSEL_SI2_0 | | 0 | | WSEL_SI1_0 | | 0 | | WSEL_SI0_0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-28<br>WSEL_SI7_0 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 27-26<br>— | Reserved |
| 25-24<br>WSEL_SI6_0 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 23-22<br>— | Reserved |
| 21-20<br>WSEL_SI5_0 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 19-18<br>— | Reserved |
| 17-16<br>WSEL_SI4_0 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 15-14<br>— | Reserved |
| 13-12<br>WSEL_SI3_0 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 11-10<br>— | Reserved |
| 9-8<br>WSEL_SI2_0 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 7-6<br>— | Reserved |
| 5-4<br>WSEL_SI1_0 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 3-2<br>— | Reserved |
| 1-0<br>WSEL_SI0_0 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |

### 57.6.2.40 Channel Analog Watchdog Select For Standard Inputs (CWSELRSI1)

#### Offset

| Register | Offset |
|---|---|
| CWSELRSI1 | 2C4h |

#### Function

Selects the analog watchdog threshold register (THRHLR) that provides limits to monitor the standard inputs.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | WSEL_SI7_1 | | 0 | | WSEL_SI6_1 | | 0 | | WSEL_SI5_1 | | 0 | | WSEL_SI4_1 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | WSEL_SI3_1 | | 0 | | WSEL_SI2_1 | | 0 | | WSEL_SI1_1 | | 0 | | WSEL_SI0_1 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-28<br>WSEL_SI7_1 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 27-26<br>— | Reserved |
| 25-24<br>WSEL_SI6_1 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 23-22<br>— | Reserved |
| 21-20<br>WSEL_SI5_1 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 19-18<br>— | Reserved |
| 17-16<br>WSEL_SI4_1 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 11b - Analog watchdog THRHLR3 |
| 15-14<br>— | Reserved |
| 13-12<br>WSEL_SI3_1 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 11-10<br>— | Reserved |
| 9-8<br>WSEL_SI2_1 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 7-6<br>— | Reserved |
| 5-4<br>WSEL_SI1_1 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 3-2<br>— | Reserved |
| 1-0<br>WSEL_SI0_1 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |

### 57.6.2.41 Channel Analog Watchdog Select For Standard Inputs (CWSELRSI2)

#### Offset

| Register | Offset |
|---|---|
| CWSELRSI2 | 2C8h |

#### Function

Selects the analog watchdog threshold register (THRHLR) that provides limits to monitor the standard inputs.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | WSEL_SI7_2 | | 0 | | WSEL_SI6_2 | | 0 | | WSEL_SI5_2 | | 0 | | WSEL_SI4_2 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | WSEL_SI3_2 | | 0 | | WSEL_SI2_2 | | 0 | | WSEL_SI1_2 | | 0 | | WSEL_SI0_2 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-28<br>WSEL_SI7_2 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 11b - Analog watchdog THRHLR3 |
| 27-26<br>— | Reserved |
| 25-24<br>WSEL_SI6_2 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 23-22<br>— | Reserved |
| 21-20<br>WSEL_SI5_2 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 19-18<br>— | Reserved |
| 17-16<br>WSEL_SI4_2 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 15-14<br>— | Reserved |
| 13-12<br>WSEL_SI3_2 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 11-10<br>— | Reserved |
| 9-8<br>WSEL_SI2_2 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 7-6<br>— | Reserved |
| 5-4<br>WSEL_SI1_2 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 3-2<br>— | Reserved |
| 1-0<br>WSEL_SI0_2 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |

### 57.6.2.42 Channel Analog Watchdog Select For External inputs (CWSELREI0)

#### Offset

| Register | Offset |
|---|---|
| CWSELREI0 | 2D0h |

#### Function

Selects the analog watchdog threshold register (THRHLR) that provides limits to monitor external inputs.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| ADC_0 | CWSELREI0 | — |
| ADC_1 | CWSELREI0 | — |
| ADC_2 | — | CWSELREI0 |

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | WSEL_SI0_7 | | 0 | | WSEL_SI0_6 | | 0 | | WSEL_SI0_5 | | 0 | | WSEL_SI0_4 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | WSEL_SI0_3 | | 0 | | WSEL_SI0_2 | | 0 | | WSEL_SI0_1 | | 0 | | WSEL_SI0_0 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-28<br>WSEL_SI0_7 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 27-26<br>— | Reserved |
| 25-24<br>WSEL_SI0_6 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 23-22<br>— | Reserved |
| 21-20<br>WSEL_SI0_5 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 19-18<br>— | Reserved |
| 17-16<br>WSEL_SI0_4 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 15-14<br>— | Reserved |
| 13-12<br>WSEL_SI0_3 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 11-10<br>— | Reserved |
| 9-8<br>WSEL_SI0_2 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 7-6<br>— | Reserved |
| 5-4<br>WSEL_SI0_1 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 3-2<br>— | Reserved |
| 1-0<br>WSEL_SI0_0 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |

### 57.6.2.43 Channel Analog Watchdog Select For External inputs (CWSELREI1)

#### Offset

| Register | Offset |
|---|---|
| CWSELREI1 | 2D4h |

#### Function

Selects the analog watchdog threshold register (THRHLR) that provides limits to monitor external inputs.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| ADC_0 | CWSELREI1 | — |
| ADC_1 | CWSELREI1 | — |
| ADC_2 | — | CWSELREI1 |

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | WSEL_SI1_15 | | 0 | | WSEL_SI1_14 | | 0 | | WSEL_SI1_13 | | 0 | | WSEL_SI1_12 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | WSEL_SI1_11 | | 0 | | WSEL_SI1_10 | | 0 | | WSEL_SI1_9 | | 0 | | WSEL_SI1_8 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-28<br>WSEL_SI1_15 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 27-26<br>— | Reserved |
| 25-24<br>WSEL_SI1_14 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 23-22<br>— | Reserved |
| 21-20<br>WSEL_SI1_13 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 19-18<br>— | Reserved |
| 17-16<br>WSEL_SI1_12 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 15-14<br>— | Reserved |
| 13-12<br>WSEL_SI1_11 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 11-10<br>— | Reserved |
| 9-8<br>WSEL_SI1_10 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 7-6<br>— | Reserved |
| 5-4<br>WSEL_SI1_9 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 3-2<br>— | Reserved |
| 1-0<br>WSEL_SI1_8 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |

### 57.6.2.44 Channel Analog Watchdog Select For External inputs (CWSELREI2)

#### Offset

| Register | Offset |
|---|---|
| CWSELREI2 | 2D8h |

#### Function

Selects the analog watchdog threshold register (THRHLR) that provides limits to monitor external inputs.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| ADC_0 | CWSELREI2 | — |
| ADC_1 | CWSELREI2 | — |
| ADC_2 | — | CWSELREI2 |

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | WSEL_SI2_23 | | 0 | | WSEL_SI2_22 | | 0 | | WSEL_SI2_21 | | 0 | | WSEL_SI2_20 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | WSEL_SI2_19 | | 0 | | WSEL_SI2_18 | | 0 | | WSEL_SI2_17 | | 0 | | WSEL_SI2_16 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-28<br>WSEL_SI2_23 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 27-26<br>— | Reserved |
| 25-24<br>WSEL_SI2_22 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 23-22<br>— | Reserved |
| 21-20<br>WSEL_SI2_21 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 19-18<br>— | Reserved |
| 17-16<br>WSEL_SI2_20 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 15-14<br>— | Reserved |
| 13-12<br>WSEL_SI2_19 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 11-10<br>— | Reserved |
| 9-8<br>WSEL_SI2_18 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 7-6<br>— | Reserved |
| 5-4<br>WSEL_SI2_17 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 3-2<br>— | Reserved |
| 1-0<br>WSEL_SI2_16 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |

### 57.6.2.45 Channel Analog Watchdog Select For External inputs (CWSELREI3)

#### Offset

| Register | Offset |
|---|---|
| CWSELREI3 | 2DCh |

#### Function

Selects the analog watchdog threshold register (THRHLR) that provides limits to monitor external inputs.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| ADC_0 | CWSELREI3 | — |
| ADC_1 | CWSELREI3 | — |
| ADC_2 | — | CWSELREI3 |

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | WSEL_SI3_31 | | 0 | | WSEL_SI3_30 | | 0 | | WSEL_SI3_29 | | 0 | | WSEL_SI3_28 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | WSEL_SI3_27 | | 0 | | WSEL_SI3_26 | | 0 | | WSEL_SI3_25 | | 0 | | WSEL_SI3_24 | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-28<br>WSEL_SI3_31 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 27-26<br>— | Reserved |
| 25-24<br>WSEL_SI3_30 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 23-22<br>— | Reserved |
| 21-20<br>WSEL_SI3_29 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 19-18<br>— | Reserved |
| 17-16<br>WSEL_SI3_28 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 15-14<br>— | Reserved |
| 13-12<br>WSEL_SI3_27 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 11-10<br>— | Reserved |
| 9-8<br>WSEL_SI3_26 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 7-6<br>— | Reserved |
| 5-4<br>WSEL_SI3_25 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |
| 3-2<br>— | Reserved |
| 1-0<br>WSEL_SI3_24 | Analog Watchdog Selection<br>Selects the analog watchdog thresholds that the input is to be compared to.<br>00b - Analog watchdog THRHLR0<br>01b - Analog watchdog THRHLR1<br>10b - Analog watchdog THRHLR2<br>11b - Analog watchdog THRHLR3 |

### 57.6.2.46 Channel Watchdog Enable For Precision Inputs (CWENR0)

#### Offset

| Register | Offset |
|---|---|
| CWENR0 | 2E0h |

#### Function

Enables the analog watchdog per precision input.

#### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | Reserved | Reserved | CWEN7 | CWEN6 | CWEN5 | CWEN4 | CWEN3 | CWEN2 | CWEN1 | CWEN0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>— | Reserved |
| 8<br>— | Reserved |
| 7<br>CWEN7 | Channel Analog Watchdog Enable 7<br>Enables the analog watchdog for precision input 7.<br>When enabled, conversion data on the input is compared to the selected data watchdog threshold value.<br>0b - Disable<br>1b - Enable |
| 6<br>CWEN6 | Channel Analog Watchdog Enable 6<br>Enables the analog watchdog for precision input 6. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When enabled, conversion data on the input is compared to the selected data watchdog threshold value.<br>0b - Disable<br>1b - Enable |
| 5<br>CWEN5 | Channel Analog Watchdog Enable 5<br>Enables the analog watchdog for precision input 5.<br>When enabled, conversion data on the input is compared to the selected data watchdog threshold value.<br>0b - Disable<br>1b - Enable |
| 4<br>CWEN4 | Channel Analog Watchdog Enable 4<br>Enables the analog watchdog for precision input 4.<br>When enabled, conversion data on the input is compared to the selected data watchdog threshold value.<br>0b - Disable<br>1b - Enable |
| 3<br>CWEN3 | Channel Analog Watchdog Enable 3<br>Enables the analog watchdog for precision input 3.<br>When enabled, conversion data on the input is compared to the selected data watchdog threshold value.<br>0b - Disable<br>1b - Enable |
| 2<br>CWEN2 | Channel Analog Watchdog Enable 2<br>Enables the analog watchdog for precision input 2.<br>When enabled, conversion data on the input is compared to the selected data watchdog threshold value.<br>0b - Disable<br>1b - Enable |
| 1<br>CWEN1 | Channel Analog Watchdog Enable 1<br>Enables the analog watchdog for precision input 1.<br>When enabled, conversion data on the input is compared to the selected data watchdog threshold value.<br>0b - Disable<br>1b - Enable |
| 0<br>CWEN0 | Channel Analog Watchdog Enable 0<br>Enables the analog watchdog for precision input 0.<br>When enabled, conversion data on the input is compared to the selected data watchdog threshold value. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disable<br>1b - Enable |

### 57.6.2.47 Channel Watchdog Enable For Standard Inputs (CWENR1)

#### Offset

| Register | Offset |
|---|---|
| CWENR1 | 2E4h |

#### Function

Enables the analog watchdog for standard input *n*.

When enabled, conversion data on the input is compared to the selected data watchdog threshold value.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | CWEN 55 | CWEN 54 | CWEN 53 | CWEN 52 | CWEN 51 | CWEN 50 | CWEN 49 | CWEN 48 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CWEN 47 | CWEN 46 | CWEN 45 | CWEN 44 | CWEN 43 | CWEN 42 | CWEN 41 | CWEN 40 | CWEN 39 | CWEN 38 | CWEN 37 | CWEN 36 | CWEN 35 | CWEN 34 | CWEN 33 | CWEN 32 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>CWENn | Channel Analog Watchdog Enable For Standard Inputs<br>Enables the analog watchdog for standard input *n*.<br>0b - Disable<br>1b - Enable |

### 57.6.2.48 Channel Watchdog Enable For External Inputs (CWENR2)

#### Offset

| Register | Offset |
|---|---|
| CWENR2 | 2E8h |

#### Function

Enables the analog watchdog per external input.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| ADC_0 | CWENR2 | — |
| ADC_1 | CWENR2 | — |
| ADC_2 | — | CWENR2 |

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CWEN 95 | CWEN 94 | CWEN 93 | CWEN 92 | CWEN 91 | CWEN 90 | CWEN 89 | CWEN 88 | CWEN 87 | CWEN 86 | CWEN 85 | CWEN 84 | CWEN 83 | CWEN 82 | CWEN 81 | CWEN 80 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CWEN 79 | CWEN 78 | CWEN 77 | CWEN 76 | CWEN 75 | CWEN 74 | CWEN 73 | CWEN 72 | CWEN 71 | CWEN 70 | CWEN 69 | CWEN 68 | CWEN 67 | CWEN 66 | CWEN 65 | CWEN 64 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-0<br>CWENn | Channel Analog Watchdog Enable For External Inputs<br>Enables the analog watchdog for external input *n*.<br>0b - Disable<br>1b - Enable |

### 57.6.2.49 Analog Watchdog Out Of Range For Precision Inputs (AWORR0)

#### Offset

| Register | Offset |
|---|---|
| AWORR0 | 2F0h |

#### Function

Indicates the status of analog watchdog comparisons on precision inputs.

#### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | Reserved | Reserved | AWOR_CH7 | AWOR_CH6 | AWOR_CH5 | AWOR_CH4 | AWOR_CH3 | AWOR_CH2 | AWOR_CH1 | AWOR_CH0 |
| W | | | | | | | | | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>— | Reserved |
| 8<br>— | Reserved |
| 7<br>AWOR_CH7 | Analog Watchdog Out Of Range For Precision Inputs<br>Indicates whether a data conversion on precision input 7 is out of the limits defined by the selected analog watchdog threshold value.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion is within limits |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Conversion is not within limits<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 6<br>AWOR_CH6 | Analog Watchdog Out Of Range For Precision Inputs<br>Indicates whether a data conversion on precision input 6 is out of the limits defined by the selected analog watchdog threshold value.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion is within limits<br>1b - Conversion is not within limits<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 5<br>AWOR_CH5 | Analog Watchdog Out Of Range For Precision Inputs<br>Indicates whether a data conversion on precision input 5 is out of the limits defined by the selected analog watchdog threshold value.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion is within limits<br>1b - Conversion is not within limits<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 4<br>AWOR_CH4 | Analog Watchdog Out Of Range For Precision Inputs<br>Indicates whether a data conversion on precision input 4 is out of the limits defined by the selected analog watchdog threshold value.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Conversion is within limits<br>1b - Conversion is not within limits<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 3<br>AWOR_CH3 | Analog Watchdog Out Of Range For Precision Inputs<br>Indicates whether a data conversion on precision input 3 is out of the limits defined by the selected analog watchdog threshold value.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion is within limits<br>1b - Conversion is not within limits<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 2<br>AWOR_CH2 | Analog Watchdog Out Of Range For Precision Inputs<br>Indicates whether a data conversion on precision input 2 is out of the limits defined by the selected analog watchdog threshold value.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion is within limits<br>1b - Conversion is not within limits<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 1<br>AWOR_CH1 | Analog Watchdog Out Of Range For Precision Inputs<br>Indicates whether a data conversion on precision input 1 is out of the limits defined by the selected analog watchdog threshold value.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Conversion is within limits<br>1b - Conversion is not within limits<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 0<br>AWOR_CH0 | Analog Watchdog Out Of Range For Precision Inputs<br>Indicates whether a data conversion on precision input 0 is out of the limits defined by the selected analog watchdog threshold value.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion is within limits<br>1b - Conversion is not within limits<br>When writing<br>0b - No effect<br>1b - Clears flag |

### 57.6.2.50 Analog Watchdog Out Of Range For Standard Inputs (AWORR1)

**Offset**

| Register | Offset |
|---|---|
| AWORR1 | 2F4h |

**Function**
Indicates the status of analog watchdog comparisons on standard inputs.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... |
| W | | | | | | | | | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH9 | AWOR_CH8 | AWOR_CH7 | AWOR_CH6 | AWOR_CH5 | AWOR_CH4 | AWOR_CH3 | AWOR_CH2 | AWOR_CH1 | AWOR_CH0 |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-0<br>AWOR_CHn | Analog Watchdog Out Of Range For Standard Inputs<br>Indicates whether a data conversion on a standard input *n* is out of the limits defined by the selected analog watchdog threshold value.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion is within limits<br>1b - Conversion is not within limits<br>When writing<br>0b - No effect<br>1b - Clears flag |

## 57.6.2.51 Analog Watchdog Out Of Range For External Inputs (AWORR2)

### Offset

| Register | Offset |
|---|---|
| AWORR2 | 2F8h |

### Function

Indicates the status of analog watchdog comparisons on external inputs.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| ADC_0 | AWORR2 | — |
| ADC_1 | AWORR2 | — |
| ADC_2 | — | AWORR2 |

## Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH... | AWOR_CH9 | AWOR_CH8 | AWOR_CH7 | AWOR_CH6 | AWOR_CH5 | AWOR_CH4 | AWOR_CH3 | AWOR_CH2 | AWOR_CH1 | AWOR_CH0 |
| W | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## Fields

| Field | Function |
|---|---|
| 31-0<br>AWOR_CHn | Analog Watchdog Out Of Range For External Inputs<br>Indicates whether a data conversion on external input *n* is out of the limits defined by the selected analog watchdog threshold value.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Conversion is within limits<br>1b - Conversion is not within limits<br>When writing<br>0b - No effect<br>1b - Clears flag |

### 57.6.2.52 Self-Test Configuration 1 (STCR1)

#### Offset

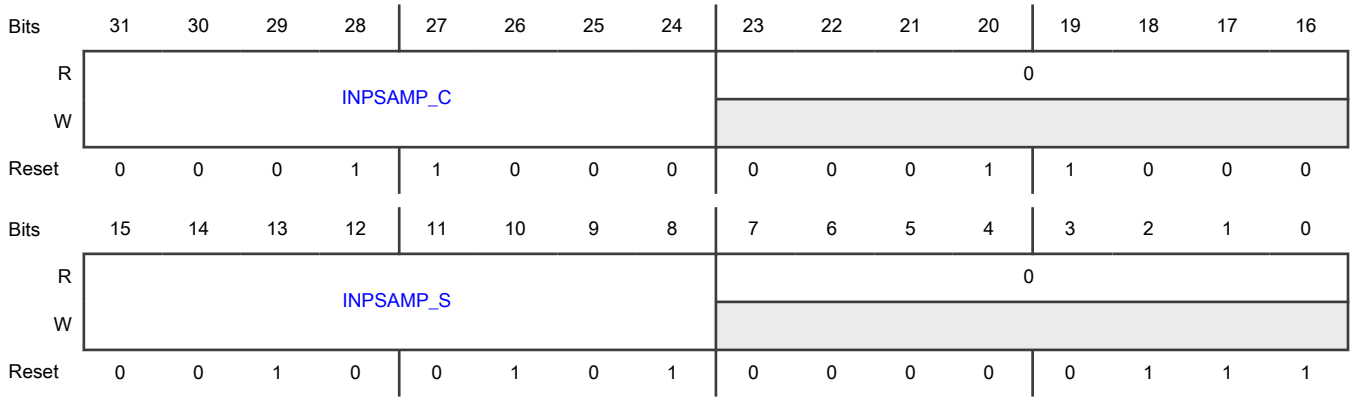
| Register | Offset |
|---|---|
| STCR1 | 340h |

#### Function

Specifies the input sampling duration in terms of conversion clock cycles.

Conversion clock frequency depends on the configuration of MCR[ADCLKSEL].

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-24<br>INPSAMP_C | Input Sampling Time Algorithm C<br>Specifies the sample duration for test conversions related to algorithm C. The minimum acceptable value is 8. Specifying a lower value automatically forces a value of 8. |
| 23-16<br>— | Reserved |
| 15-8<br>INPSAMP_S | Input Sampling Time Algorithm S<br>Specifies sample duration for test conversions related to algorithm S. The minimum value is 8. Specifying a lower value automatically forces a value of 8. |
| 7-0<br>— | Reserved |

### 57.6.2.53 Self-Test Configuration 2 (STCR2)

#### Offset

| Register | Offset |
|---|---|
| STCR2 | 344h |

#### Function
Configures ADC self-test functions.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | MSKWDSE... | SERR | MSKWDTE... | 0 | MSKST_E... | 0 | | | | MSKWDG_... | 0 | MSKWDG_... |
| W | | | | | | W1C | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MSKERR_C | 0 | MSKERR_... | MSKERR_... | MSKERR_... | 0 | | | EN | 0 | | FMA_WDS... | FMA_WDT... | FMA_C | 0 | FMA_S |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

#### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27<br>MSKWDSERR | Mask Interrupt Self-Test Watchdog Sequence Error<br>Generates an interrupt when the WDSERR flag transitions to 1.<br>0b - No interrupt is generated<br>1b - Interrupt is generated |
| 26<br>SERR | Self-Test Error Injection<br>Writes 1 to the STSR1[ERR_C], STSR1[ERR_S0], STSR1[ERR_S1], and STSR1[ERR_S2] status fields, causing an error. This field is reset to 0 immediately after writing.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Error can be injected<br>1b - Error is being injected |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When writing<br>0b - No effect<br>1b - Injects a self-test error |
| 25<br>MSKWDTERR | Mask Interrupt Self-Test Watchdog Timer Error<br>Generates an interrupt when the WDTERR flag is set.<br>0b - No interrupt is generated<br>1b - Interrupt is generated |
| 24<br>— | Reserved |
| 23<br>MSKST_EOC | Mask Interrupt Self-Test End Of Conversion<br>Generates an interrupt when the ST_EOC flag is set.<br>0b - No interrupt is generated<br>1b - Interrupt is generated |
| 22-19<br>— | Reserved |
| 18<br>MSKWDG_EOA_C | Mask Error Interrupt End Of Algorithm C<br>Generates an interrupt when the WDG_EOA_C flag is set.<br>0b - No interrupt is generated<br>1b - Interrupt is generated |
| 17<br>— | Reserved |
| 16<br>MSKWDG_EOA_S | Mask Error Interrupt End Of Algorithm S<br>Generates an interrupt when the WDG_EOA_S flag is set.<br>0b - No interrupt is generated<br>1b - Interrupt is generated |
| 15<br>MSKERR_C | Mask Error Interrupt Algorithm C<br>Generates an interrupt when the ERR_C flag is set.<br>0b - No interrupt is generated<br>1b - Interrupt is generated |
| 14 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 13<br>MSKERR_S2 | Mask Error Interrupt Algorithm S2<br>Generates an interrupt when the ERR_S2 flag is set.<br>0b - No interrupt is generated<br>1b - Interrupt is generated |
| 12<br>MSKERR_S1 | Mask Error Interrupt Algorithm S1<br>Generates an interrupt when the ERR_S1 flag is set.<br>0b - No interrupt is generated<br>1b - Interrupt is generated |
| 11<br>MSKERR_S0 | Mask Error Interrupt Algorithm S0<br>Generates an interrupt when the ERR_S0 flag is set.<br>0b - No interrupt is generated<br>1b - Interrupt is generated |
| 10-8<br>— | Reserved |
| 7<br>EN | Self-Test Enable<br>Enables ADC structural self-test. When BCTU is enabled, this field has no effect. This field must be 1 before starting normal conversion and must not be changed during conversion. This field must be reset to 0 only after end of conversion for the last self-test channel has been received.<br>0b - Disable<br>1b - Enable |
| 6-5<br>— | Reserved |
| 4<br>FMA_WDSERR | Fault Mapping Self-Test Watchdog Sequence Error<br>Specifies whether a self-test watchdog sequence error sets the flag on the critical or noncritical fault line.<br>0b - Noncritical fault line<br>1b - Critical fault line |
| 3<br>FMA_WDTERR | Fault Mapping Self-Test Watchdog Timer Error<br>Specifies whether a self-test watchdog timer error sets the flag on the critical or noncritical fault line.<br>0b - Noncritical fault line<br>1b - Critical fault line |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 2<br>FMA_C | Fault Mapping Algorithm C<br>Specifies whether a fault in algorithm C sets the flag on the critical or noncritical fault line.<br>0b - Noncritical fault line<br>1b - Critical fault line |
| 1<br>— | Reserved |
| 0<br>FMA_S | Fault Mapping Algorithm S<br>Specifies whether a fault in algorithm S sets the flag on the critical or noncritical fault line.<br>0b - Noncritical fault line<br>1b - Critical fault line |

### 57.6.2.54 Self-Test Configuration 3 (STCR3)

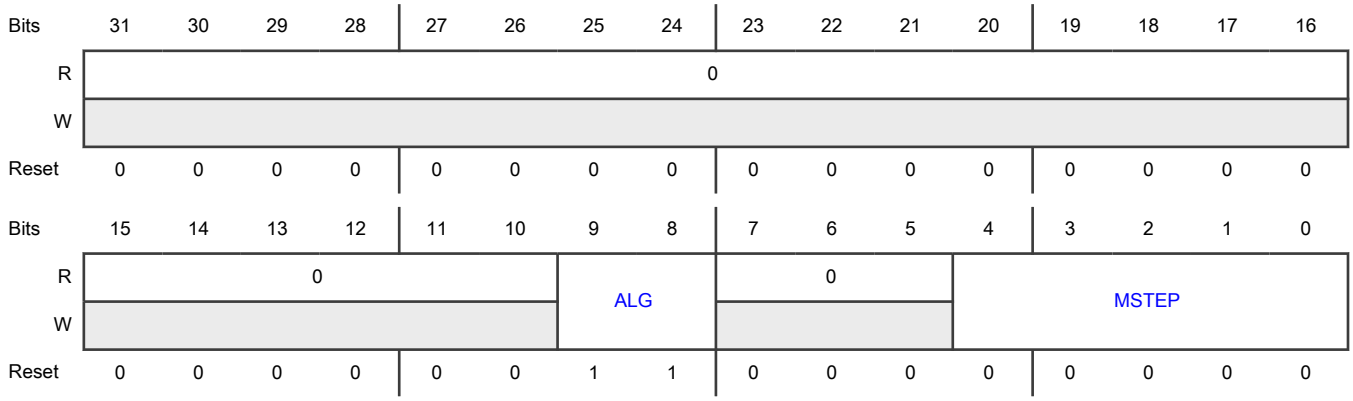
#### Offset

| Register | Offset |
|---|---|
| STCR3 | 348h |

#### Function

Configures ADC self-test functions. When BCTU is enabled, writes to the fields registers have no effect. This register must be programmed before starting a conversion and must not be changed when conversion is ongoing.

#### Diagram



**Fields**

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9-8<br>ALG | Algorithm Selection<br>One-Shot operation mode:<br>• 00 Algorithm S (single step = MSTEP)<br>• 01 Reserved<br>• 10 Algorithm C (single step = MSTEP)<br>• 11 Algorithm S (default)<br>(use for test/debug purposes)<br>Continuous conversion mode:<br>• 00 Algorithm S<br>• 01 Reserved<br>• 10 Algorithm C<br>• 11 Algorithm S + algorithm C (default) |
| 7-5<br>— | Reserved |
| 4-0<br>MSTEP | Algorithm Step<br>For One-Shot operation mode:<br>• MSTEP = 0 to 2 for algorithm S<br>• MSTEP = 0 to 11 for algorithm C<br>Unused codes are reserved and must not be used.<br>For Scan operation mode:<br>This field is not used in Scan operation mode because single-step execution (interleaved mode) is always performed.<br>This field must be programmed to zero in Scan operation mode. |

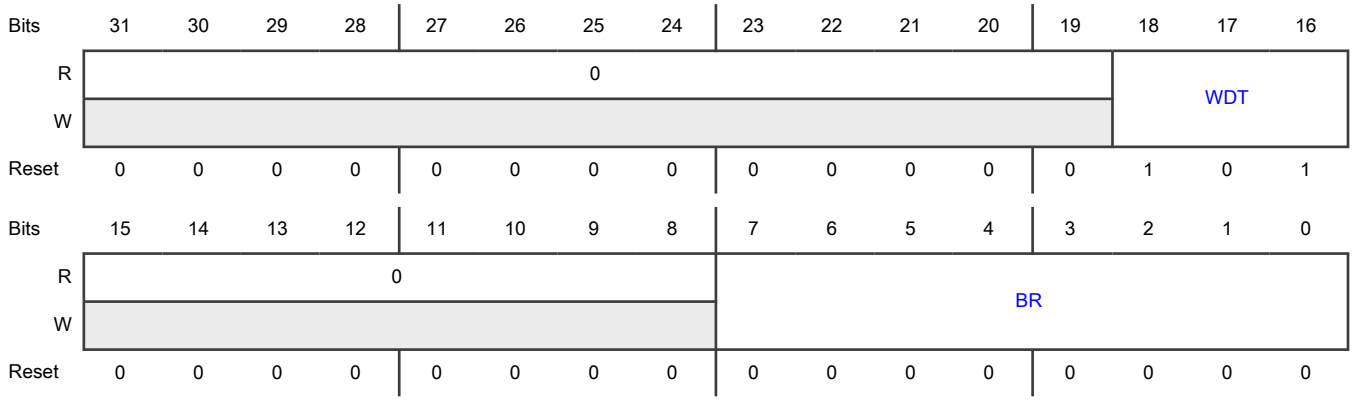
### 57.6.2.55 Self-Test Baud Rate (STBRR)

**Offset**

| Register | Offset |
|---|---|
| STBRR | 34Ch |

**Function**
Specifies when the self-test algorithm steps are executed and the maximum time allowed for self-test to finish.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18-16<br>WDT | Self-Test Watchdog Timer<br>Specifies the safe time period in terms of conversion clock cycles. The safe time period is the maximum duration until the running self-test is expected to finish.<br>000b - 8192 conversion clock cycles (~0.1 ms at 80 MHz)<br>001b - 39,936 conversion clock cycles (~0.5 ms at 80 MHz)<br>010b - 79,872 conversion clock cycles (~1 ms at 80 MHz)<br>011b - 159,744 conversion clock cycles (~2 ms at 80 MHz)<br>100b - 400,384 conversion clock cycles (~5 ms at 80 MHz)<br>101b - 799,744 conversion clock cycles (~10 ms at 80 MHz)<br>110b - 1,599,488 conversion clock cycles (~20 ms at 80 MHz)<br>111b - 3,999,744 conversion clock cycles (~50 ms at 80 MHz) |
| 15-8<br>— | Reserved |
| 7-0<br>BR | Baud Rate<br>Selects the number of conversions of the selected inputs, after which the next self-test algorithm step executes. This field must be programmed when STCR2[EN] is zero, that is, before enabling self-test.<br>0000_0000b - A step of the selected self-test algorithm is executed every time after the set of selected inputs has been converted.<br>1111_1111b - A step of the selected self-test algorithm is executed after the set of selected inputs has been converted 256 times. |

### 57.6.2.56 Self-Test Status 1 (STSR1)

#### Offset

| Register | Offset |
|---|---|
| STSR1 | 350h |

#### Function
Indicates self-test status.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | WDSERR | 0 | WDTERR | OVERWR | ST_EOC | 0 | | | | WDG_EOA... | 0 | WDG_EOA... |
| W | | | | | W1C | | W1C | W1C | W1C | | | | | W1C | | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ERR_C | 0 | ERR_S2 | ERR_S1 | ERR_S0 | 0 | STEP_C | | | | | | 0 | | | |
| W | W1C | | W1C | W1C | W1C | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27<br>WDSERR | Self-Test Watchdog Sequence Error<br>Indicates whether the selected self-test algorithm has executed in the correct sequence.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Algorithm executed in correct sequence<br>1b - Algorithm did not execute in correct sequence<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 26 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 25<br>WDTERR | Self-Test Watchdog Timer Error<br>Indicates whether the self-test algorithm has finished within the defined safe time period.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Algorithm finished within the safe time period (or safe time period not yet elapsed).<br>1b - Algorithm did not finish within safe time period.<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 24<br>OVERWR | Self-Test Error Status Overwrite<br>Indicates whether an error occurred when the respective error status flag (ERR_S0, ERR_S1, ERR_S2, or ERR_C) was set.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No self-test error status flag overwritten<br>1b - Self-test error status flag overwritten<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 23<br>ST_EOC | Self-Test End Of Conversion<br>Indicates whether a self-test conversion has completed.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not complete<br>1b - Complete<br>When writing<br>0b - No effect<br>1b - Clears flag |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 22-19<br>— | Reserved |
| 18<br>WDG_EOA_C | Self-Test Watchdog End Of Algorithm C<br>Indicates whether self-test algorithm C has completed.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not complete<br>1b - Complete<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 17<br>— | Reserved |
| 16<br>WDG_EOA_S | Self-Test Watchdog End Of Algorithm S<br>Indicates whether self-test algorithm S has completed.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not complete<br>1b - Complete<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 15<br>ERR_C | Error Algorithm C<br>Indicates whether an error occurred during execution of algorithm C.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No error<br>1b - Error<br>When writing |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No effect<br>1b - Clears flag |
| 14<br>— | Reserved |
| 13<br>ERR_S2 | Error Algorithm S Step 2<br>Indicates whether an error occurred during execution of step 2 of algorithm S.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No error<br>1b - Error<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 12<br>ERR_S1 | Error Algorithm S Step 1<br>Indicates whether an error occurred during execution of step 1 of algorithm S.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No error<br>1b - Error<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 11<br>ERR_S0 | Error Algorithm S Step 0<br>Indicates whether an error occurred during execution of step 0 of algorithm S.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No error<br>1b - Error<br>When writing |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No effect<br>1b - Clears flag |
| 10<br>— | Reserved |
| 9-5<br>STEP_C | Step Of Algorithm C<br>Indicates the step of self-test algorithm C in which an error occurred. Although this register field is read-only, no transfer error is generated when writing to it. |
| 4-0<br>— | Reserved |

### 57.6.2.57 Self-Test Status 2 (STSR2)

#### Offset

| Register | Offset |
|---|---|
| STSR2 | 354h |

#### Function

Contains the conversion result when an error in step 1 of algorithm S has occurred.

#### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DATA0 | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-16 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 15<br>— | Reserved |
| 14-0<br>DATA0 | Conversion Data ERR_S1<br>Contains conversion data from the moment when the error in step 1 of algorithm S has occurred. |

### 57.6.2.58 Self-Test Status 3 (STSR3)

#### Offset

| Register | Offset |
|---|---|
| STSR3 | 358h |

#### Function

Contains the conversion result when an error has occurred in step 2 or step 0 of algorithm S.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DATA1 | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DATA0 | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-16<br>DATA1 | Conversion Data ERR_S2<br>Contains conversion data from the moment when the error in step 2 of algorithm S has occurred. |
| 15 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 14-0<br>DATA0 | Conversion Data ERR_S0<br>Contains the conversion data from the moment when the error in step 0 of algorithm S has occurred. |

### 57.6.2.59 Self-Test Status 4 (STSR4)

#### Offset

| Register | Offset |
|---|---|
| STSR4 | 35Ch |

#### Function

Contains the conversion result when an error has occurred in algorithm C.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DATA1 | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

#### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-16<br>DATA1 | Conversion Data ERR_C<br>Conversion data from the moment when the error in algorithm C has occurred. |
| 15-0<br>— | Reserved |

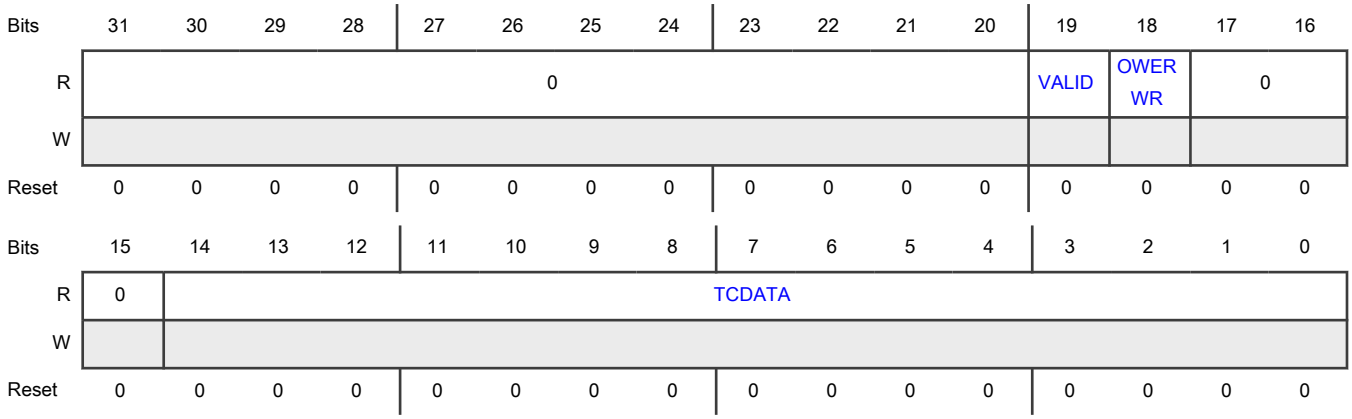
### 57.6.2.60 Self-Test Conversion Data 1 (STDR1)

#### Offset

| Register | Offset |
|---|---|
| STDR1 | 370h |

#### Function
Contains the result of the self-test conversion.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19<br>VALID | Valid Conversion Data<br>Indicates that conversion data is available in TCDATA. This field is automatically reset to 0 when the conversion data is read.<br>0b - No unread conversion data<br>1b - Unread conversion data is available |
| 18<br>OWERWR | Conversion Data Overwrite Status<br>Indicates whether the conversion data in TCDATA has been overwritten over previous data that had not been read.<br>0b - Current conversion data not overwritten<br>1b - Current conversion data was overwritten |
| 17-16<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 15<br>— | Reserved |
| 14-0<br>TCDATA | Test Channel Conversion Data<br>Contains the conversion data from self-test. If overwrite is enabled (MCR[OWREN]=1), the conversion data is from the last-executed self-test conversion. When overwrite is not enabled, conversion data is available (not overwritten) until it is read by software. The conversion data in this register is in two's complement format. |

### 57.6.2.61 Self-Test Analog Watchdog S0 (STAW0R)

#### Offset

| Register | Offset |
|---|---|
| STAW0R | 380h |

#### Function

Configures self-test for step 0 of algorithm S.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | AWDE | WDTE | THRH | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |

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

#### Fields

| Field | Function |
|---|---|
| 31<br>AWDE | Self-Test Watchdog Enable<br>Enables the self-test watchdog for step 0 of algorithm S.<br>0b - Disable<br>1b - Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 30<br>WDTE | Self-Test Watchdog Timer Enable<br>Enables the self-test watchdog timer for algorithm S.<br>The self-test watchdog timer verifies:<br>• Correct sequence of execution of the steps of the algorithm<br>• Execution of the algorithm within the safe time period as defined by STBRR[WDT]<br>This field must be 1 only in continuous conversion mode (MCR[MODE]=1).<br>The safe time period starts when this field is written with 1. If the safe time period has elapsed before algorithm S starts, the WDTERR flag is set. The safe time period is restarted each time algorithm S starts.<br>0b - Disable<br>1b - Enable |
| 29-16<br>THRH | Higher Threshold Value<br>Sets the ERR_S0 flag when a self-test conversion value is greater than this value. |
| 15<br>— | Reserved |
| 14-0<br>THRL | Lower Threshold Value<br>Sets the ERR_S0 flag when a self-test conversion value is less than this value. |

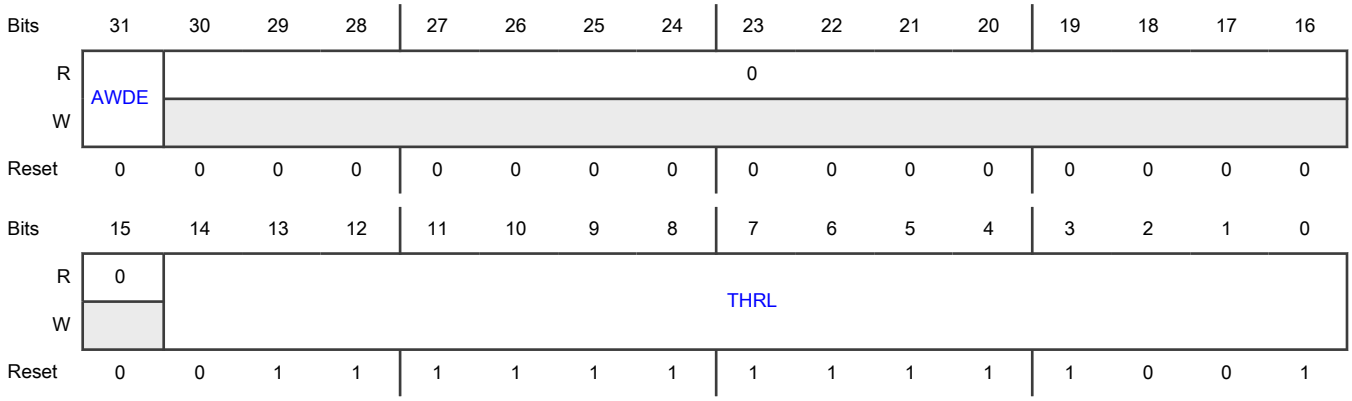
### 57.6.2.62 Self-Test Analog Watchdog S1 (STAW1R)

**Offset**

| Register | Offset |
|---|---|
| STAW1R | 388h |

**Function**
Configures self-test for step 1 of algorithm S.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>AWDE | Self-Test Watchdog Enable<br>Enables the self-test watchdog for step 1 of algorithm S.<br>0b - Disable<br>1b - Enable |
| 30-16<br>— | Reserved |
| 15<br>— | Reserved |
| 14-0<br>THRL | Lower Threshold Value<br>Sets the ERR_S1 flag when a self-test conversion value is less than this value. |

## 57.6.2.63 Self-Test Analog Watchdog S2 (STAW2R)

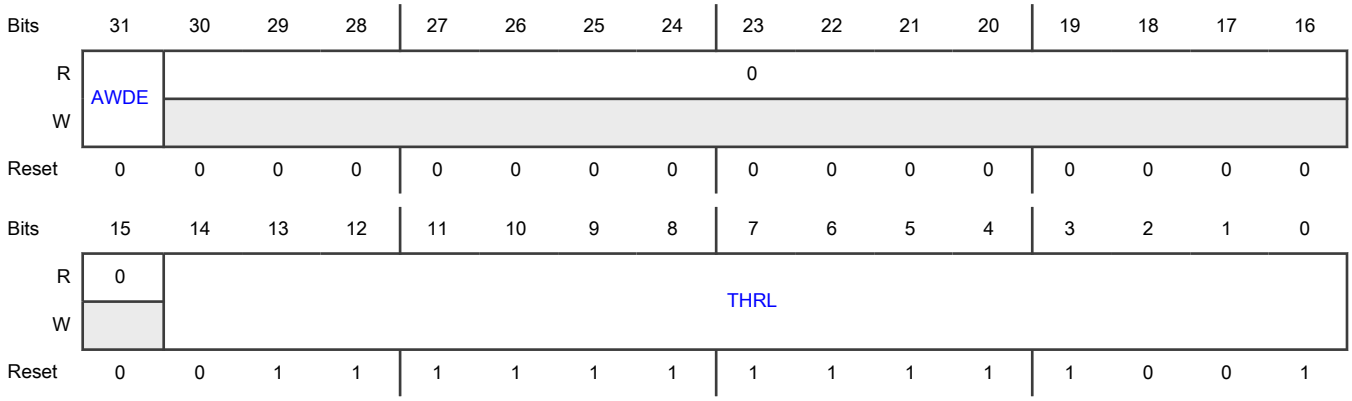
### Offset

| Register | Offset |
|---|---|
| STAW2R | 38Ch |

### Function

Configures self-test for step 2 of algorithm S.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>AWDE | Self-Test Watchdog Enable<br>Enables the self-test watchdog for step 2 of algorithm S.<br>0b - Disable<br>1b - Enable |
| 30-16<br>— | Reserved |
| 15<br>— | Reserved |
| 14-0<br>THRL | Lower Threshold Value<br>Sets the ERR_S2 flag when a self-test conversion value is less than this value. |

## 57.6.2.64 Self-Test Analog Watchdog C0 (STAW4R)

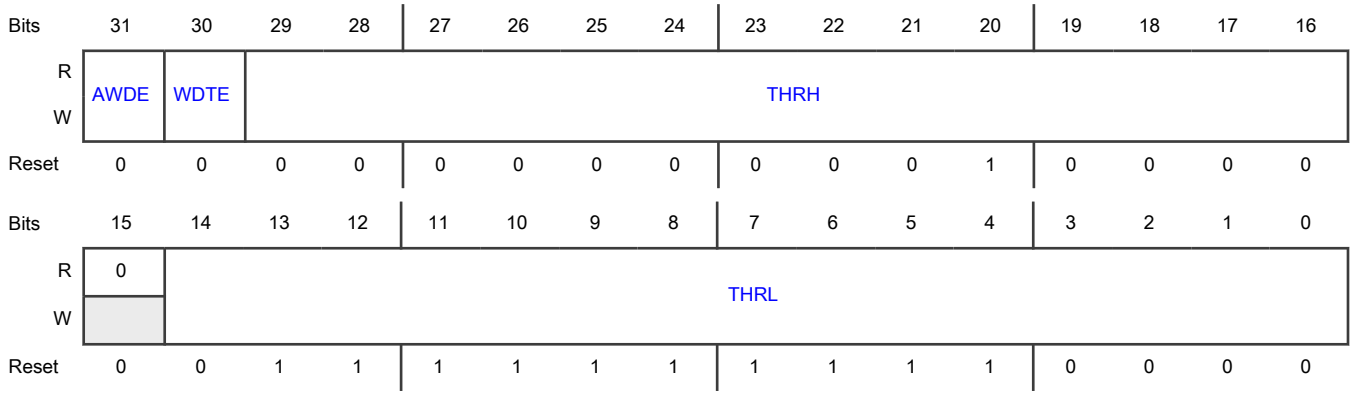
### Offset

| Register | Offset |
|---|---|
| STAW4R | 394h |

### Function

Configures self-test for step 0 of algorithm C.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>AWDE | Self-Test Watchdog Enable<br>Enables the self-test watchdog for algorithm C.<br>0b - Disable<br>1b - Enable |
| 30<br>WDTE | Self-Test Watchdog Timer Enable<br>Enables the self-test watchdog timer for algorithm C.<br>The self-test watchdog timer verifies:<br>• Correct sequence of execution of algorithm steps<br>• Execution of the algorithm within the safe time period as defined by STBRR[WDT]<br>This field must be 1 only in continuous conversion mode (MODE 1).<br>The safe time period starts when this field is written with 1. If the safe time period elapses before algorithm C starts, the WDTERR flag is set. The safe time period is restarted each time algorithm C starts.<br>0b - Disable<br>1b - Enable |
| 29-16<br>THRH | Higher Threshold Value<br>Sets the ERR_C flag when a self-test conversion value is greater than this value. This value must be in two's complement format and must be positive.<br>This value is valid only for step 0 of algorithm C (see STAW5R for the values used for the other steps of algorithm C). |
| 15<br>— | Reserved |
| 14-0<br>THRL | Lower Threshold Value |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Sets the ERR_C flag when a self-test conversion value is less than this value. This value must be in two's complement format and must be negative. (With the reset value of this bit field, the test always fails. You must set at least STAW4R[14]=1, in order to have a negative value in this bit field.)<br>This value is used only for step 0 of algorithm C (see STAW5R for the values used for the other steps of algorithm C). |

### 57.6.2.65 Self-Test Analog Watchdog C (STAW5R)

#### Offset

| Register | Offset |
|---|---|
| STAW5R | 398h |

#### Function

Configures self-test for steps 1–11 of algorithm C.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | THRH | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

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

#### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-16<br>THRH | Higher Threshold Value<br>Sets the ERR_C flag when a self-test conversion value is greater than this value. This value must be entered in two's complement format and must be positive.<br>This value is valid only for steps 1–11 of algorithm C (see STAW4R for the value used for step 0 of algorithm C). |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 15<br>— | Reserved |
| 14-0<br>THRL | Lower Threshold Value<br>Sets the ERR_C flag when a self-test conversion value is less than this value. This value must be in two's complement format and must be negative.<br>This value is used only for steps 1–11 of algorithm C (see STAW4R for the value used for step 0 of algorithm C). |

### 57.6.2.66 Analog Miscellaneous In/Out register (AMSIO)

#### Offset

| Register | Offset |
|---|---|
| AMSIO | 39Ch |

#### Function

Configures the SAR algorithm compare step. In case you want to do a high-speed calibration or a high-speed conversion see Table 269. All other values in this register must stay at their reset value and may not be written.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | Reserved | | 0 | Reserved | HSEN | | CMPCTRL0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

#### Fields

| Field | Function |
|---|---|
| 31-23<br>— | Reserved |
| 22-21 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 20<br>— | Reserved |
| 19<br>— | Reserved |
| 18-17<br>HSEN | High-Speed Enable<br>This setting must be adapted to the ADC clock frequency. See Table 269. |
| 16<br>CMPCTRL0 | Compare Control 0<br>This setting must be adapted to the ADC clock frequency. See Table 269. |
| 15-12<br>— | Reserved |
| 11-0<br>— | Reserved |

### 57.6.2.67 Control And Calibration Status (CALBISTREG)

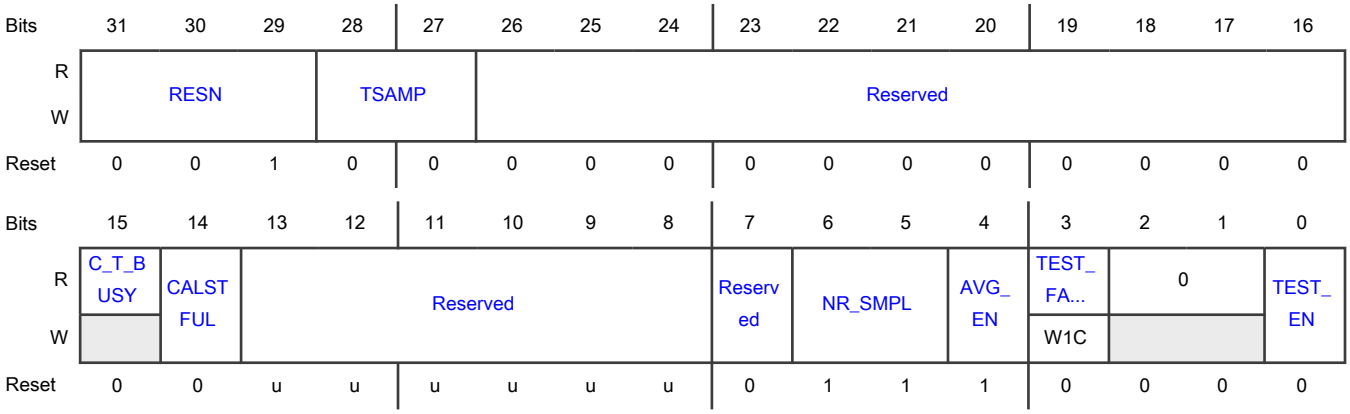
#### Offset

| Register | Offset |
|---|---|
| CALBISTREG | 3A0h |

#### Function

Controls several ADC functions for data conversion, calibration, and calibration status.

#### Diagram



### Fields

| Field | Function |
|---|---|
| 31-29<br>RESN | Conversion Resolution<br>Specifies the number of significant bits per conversion data (functional conversion only, not for calibration or self-test). Reducing this number speeds up the conversion because the SAR algorithm executes fewer steps. This field must be modified only when ADC is idle.<br>000b - 14-bit resolution<br>001b - 12-bit resolution<br>010b - 10-bit resolution<br>011b - 8-bit resolution |
| 28-27<br>TSAMP | Sample Period In Calibration<br>Specifies the number of conversion clock cycles during which the reference voltage is sampled.<br>00b - 22 conversion clock cycles<br>01b - 8 conversion clock cycles<br>10b - 16 conversion clock cycles<br>11b - 32 conversion clock cycles |
| 26-16<br>— | Reserved<br>The value of this field must not be modified. |
| 15<br>C_T_BUSY | Calibration Busy<br>Indicates whether calibration is running.<br>0b - Calibration can be started<br>1b - Calibration is in progress |
| 14<br>CALSTFUL | Calibration And Self-Test Full Range Comparison<br>Specifies the range of conversion data bits compared when ADC is calibrated or running self-test. Because the results from calibration and self-test are so low that the significant bits are in the lower 11 bits only, the comparison cycles for the upper 4 bits can be skipped to reduce the execution duration of the calibration and of the self-test.<br>0b - Lowest 11 bits are compared.<br>1b - All 15 bits are compared. |
| 13-8<br>— | Reserved<br>**CAUTION**<br>This field contains a value that ADC uses internally. Writing to this field can cause ADC to operate unreliably. |
| 7<br>— | Reserved<br>This field is reserved and always has the value 0. Only the reset value can be written to this field. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 6-5<br>NR_SMPL | Calibration Averaging Number<br>Specifies the number of data conversions per result (averaging) during calibration.<br>00b - 4 samples<br>01b - 8 samples<br>10b - 16 samples<br>11b - 32 samples |
| 4<br>AVG_EN | Calibration Averaging Enable<br>Enables averaging of the calibration results over several conversion iterations to improve accuracy. This field only affects the calibration process.<br>0b - Disable<br>1b - Enable |
| 3<br>TEST_FAIL | Calibration Status<br>Indicates calibration status. This field reads 0 when calibration is in progress. This field is reset to 0 when calibration starts.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Calibration finished successfully or has not been run since the last reset<br>1b - Calibration did not finish successfully<br>When writing<br>0b - No effect<br>1b - Clears flag |
| 2-1<br>— | Reserved |
| 0<br>TEST_EN | Calibration Enable<br>Calibrates ADC. This field automatically resets to 0 after the calibration is complete or is interrupted.<br>0b - Wait to start a calibration<br>1b - Start calibration |

### 57.6.2.68 Offset And Gain User (OFSGNUSR)

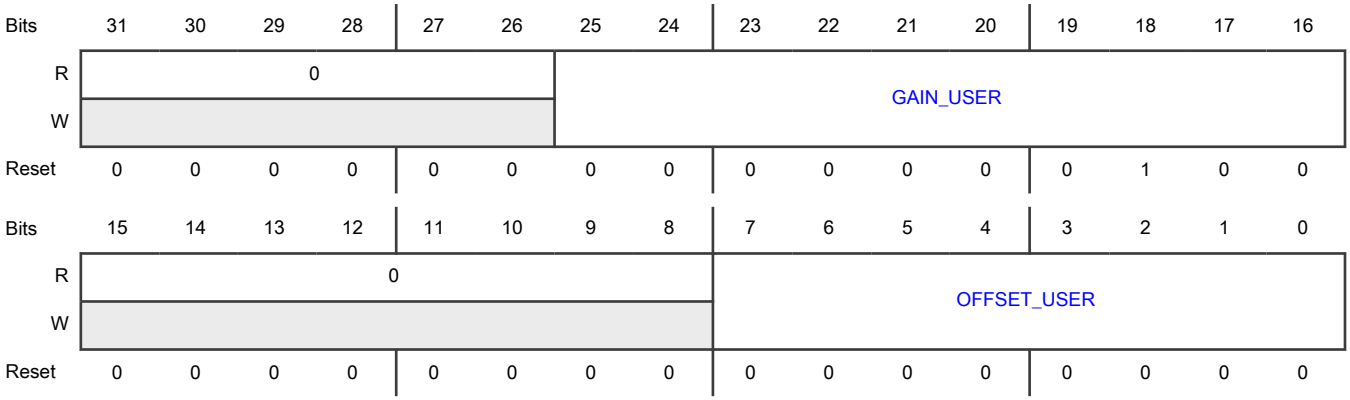
#### Offset

| Register | Offset |
|---|---|
| OFSGNUSR | 3A8h |

#### Function

Specifies the user-configured offset and gain values used by the SAR algorithm. The values must be written in two's complement format and can be either positive or negative.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25-16<br>GAIN_USER | Gain User<br>Adds extra gain to the gain calculated during calibration. The value must be in two's complement format. |
| 15-8<br>— | Reserved |
| 7-0<br>OFFSET_USER | Offset User<br>Adds extra offset to the offset calculated by calibration. The value is must be in two's complement format. |

### 57.6.2.69 Calibration Value 2 (CAL2)

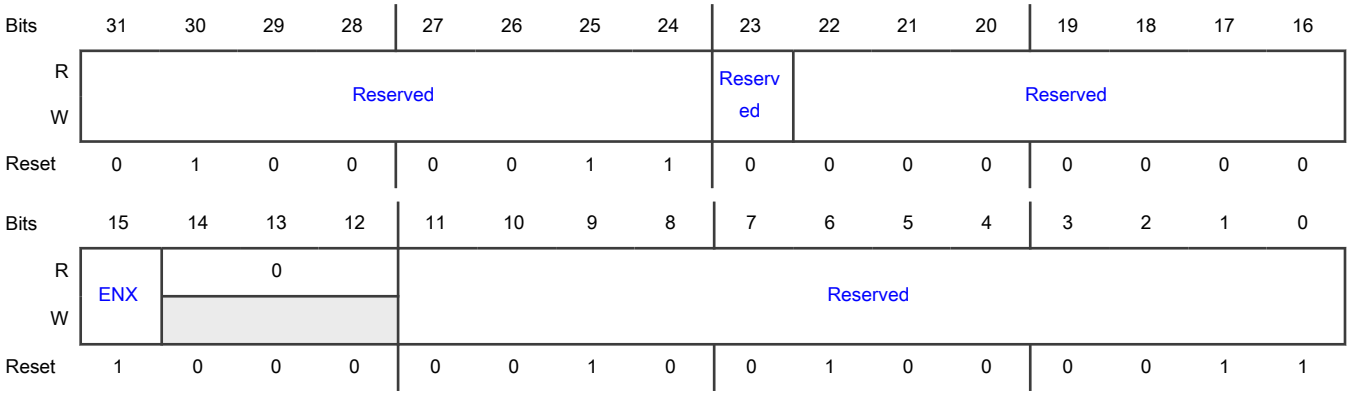
#### Offset

| Register | Offset |
|---|---|
| CAL2 | 3B4h |

### Function

Contains fields used during calibration.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>— | Reserved |
| 15<br>ENX | Enable X<br>Enables the inclusion of CLPX in error processing.<br>0b - Disable<br>1b - Enable |
| 14-12<br>— | Reserved |
| 11-0<br>— | Reserved |

## 57.7 Glossary

| | |
|---|---|
| **CDAC** | Capacitive digital to analog converter |
| **CT** | Compare phase time |
| **DMA** | Direct memory access |
| **DP** | Data processing time |

| | |
|---|---|
| **PST** | Presample phase time |
| **ST** | Sample phase time |
| **SAR** | Successive approximation |
| **TPT** | Trigger processing time |
| **VrefH** | Reference voltage high |
| **VrefL** | Reference voltage low |

# Chapter 58
# Low Power Comparator (LPCMP)

## 58.1 Chip-specific LPCMP information

### 58.1.1 Instantiation information

Table 275. LPCMP instances

| Chip | Instance | No. of external inputs |
|---|---|---|
| MCXE31B | LPCMP_0 | 8 |
| | LPCMP_1 | 4 |
| | LPCMP_2 | 4 |
| MCXE317 | LPCMP_0 | 8 |
| | LPCMP_1 | 4 |
| MCXE315/MCXE316 | LPCMP_0 | 8 |

### 58.1.2 LPCMP input output connections

See the IOMUX file attached to this document for pin/pad assignments corresponding to CMP pins.

The LPCMP channels can be used as a wakeup source in trigger mode to wakeup the device from standby mode.

### 58.1.3 LPCMP-DAC "vrefh0" and "vrefh1" references

The 8-bit DAC sub-block supports selection of "vrefh0" and "vrefh1" by CMPx_DCR[VRSEL]. For this device, the references are connected as follows:

- vrefh0 (External Reference): VDD_HV_A
- vrefh1 (Internal Reference): 1.2 V PMC bandgap reference

**NOTE**

1.2 V internal reference voltage is not available in Standby mode.

### 58.1.4 LPCMP window control

The window mode operation of all the comparator instances in the chip can be enabled/disabled by the TRGMUX. See the TRGMUX connectivity file attached to this document for details.

**NOTE**

Window signal must be of minimum 4 cycle pulse for window mode to function.

### 58.1.5 Comparator Trigger Mode

The comparator modules in the device support trigger mode operation as described in 'Trigger Mode' section. Device can operate in trigger mode in both standby and run mode to continuously scan the input channels. The main features of the device trigger mode operation are:

- Round robin clock: RTC_CLK
- Round robin trigger source: RTC_API

**NOTE**

It must be ensured that the RTC_CLK period is greater than the comparison time corresponding to the value of C0[PMODE]. It is also required to **not** select the internal reserved channels, if available on the package, for trigger mode operation by INPSEL and INNSEL. See the IOMUX file attached to this document for the pins available in various packages supported for the device.

**NOTE**

Only SIRC and SXOSC are supported as RTC_CLK for trigger mode operation. The LPCMP initialization delay is not supported by RRCR0[RR_INITMOD] with FXOSC or FIRC as RTC_CLK.

**NOTE**

In run mode, the generated trigger is delayed by 3 RTC_CLK cycles before being given to comparator trigger input.

### 58.1.6 Interaction with RTC API to cause wakeup

LPCMP can be used for waking up the chip from standby. For this, RTC-API and LPCMP must be configured before entering into standby mode as per below shown figure.

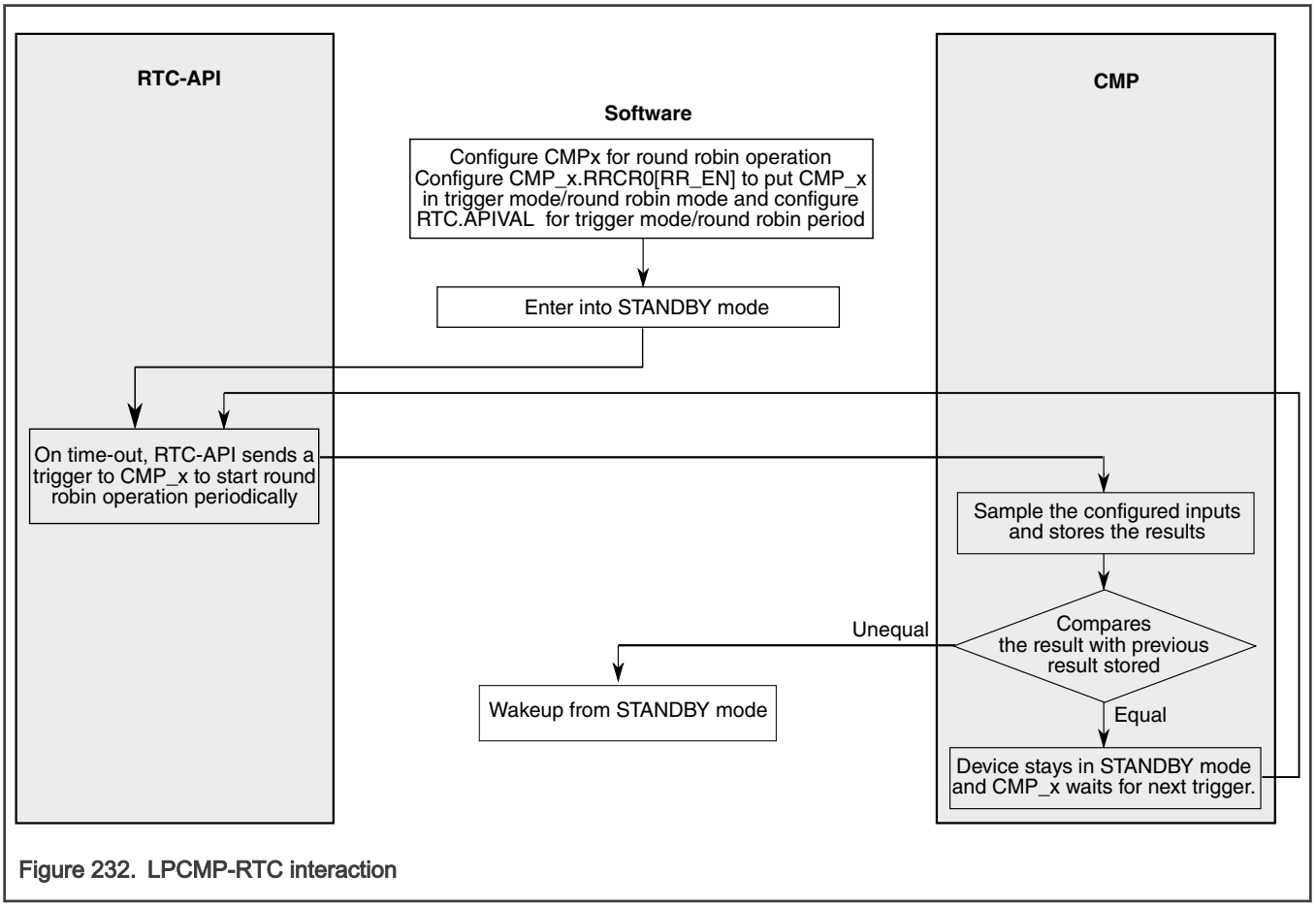



Figure 232. LPCMP-RTC interaction

In the trigger mode operation, only the continuous mode is supported. None of the window/filter/sample functions should be used. Refer to the "Functional Modes" section for details on comparator modes of operation.

**Register configurations before entering Standby mode for LPCMP trigger mode operation:**

1. Configure RTC.APIVAL to set the period of the round robin operation.
2. Execute standby mode entry.

## 58.2 Overview

The LPCMP module provides a circuit to compare two analog input voltages. It includes the following:

- A low power comparator (CMP)
- A DAC
- An analog mux (ANMUX)

See Block diagram for more information.

LPCMP can operate across the full range of the supply voltage, known as rail-to-rail operation.

DAC is a 256-tap resistor ladder network that provides a selectable voltage reference for applications requiring a voltage reference. DAC divides the supply reference $V_{in}$ into 256 voltage levels. An 8-bit digital signal input selects the output voltage level, which varies from $V_{in}$ to $V_{in}/256$.

You can select $V_{in}$ from the following voltage sources:

- VREFH0
- VREFH1

See the Chip-specific LPCMP information for more information on source of VREFH0 and VREFH1.

**NOTE**

The LPCMP's internal DAC output is available as an on-chip internal signal only and is not available for an external chip pin.

ANMUX allows you to select an analog input signal from among eight channel options. One channel option is the DAC output. Other chip resources are connected to the other channels. See the Chip-specific LPCMP information section for more information. ANMUX can operate across the full range of the supply voltage.

### 58.2.1 Block diagram

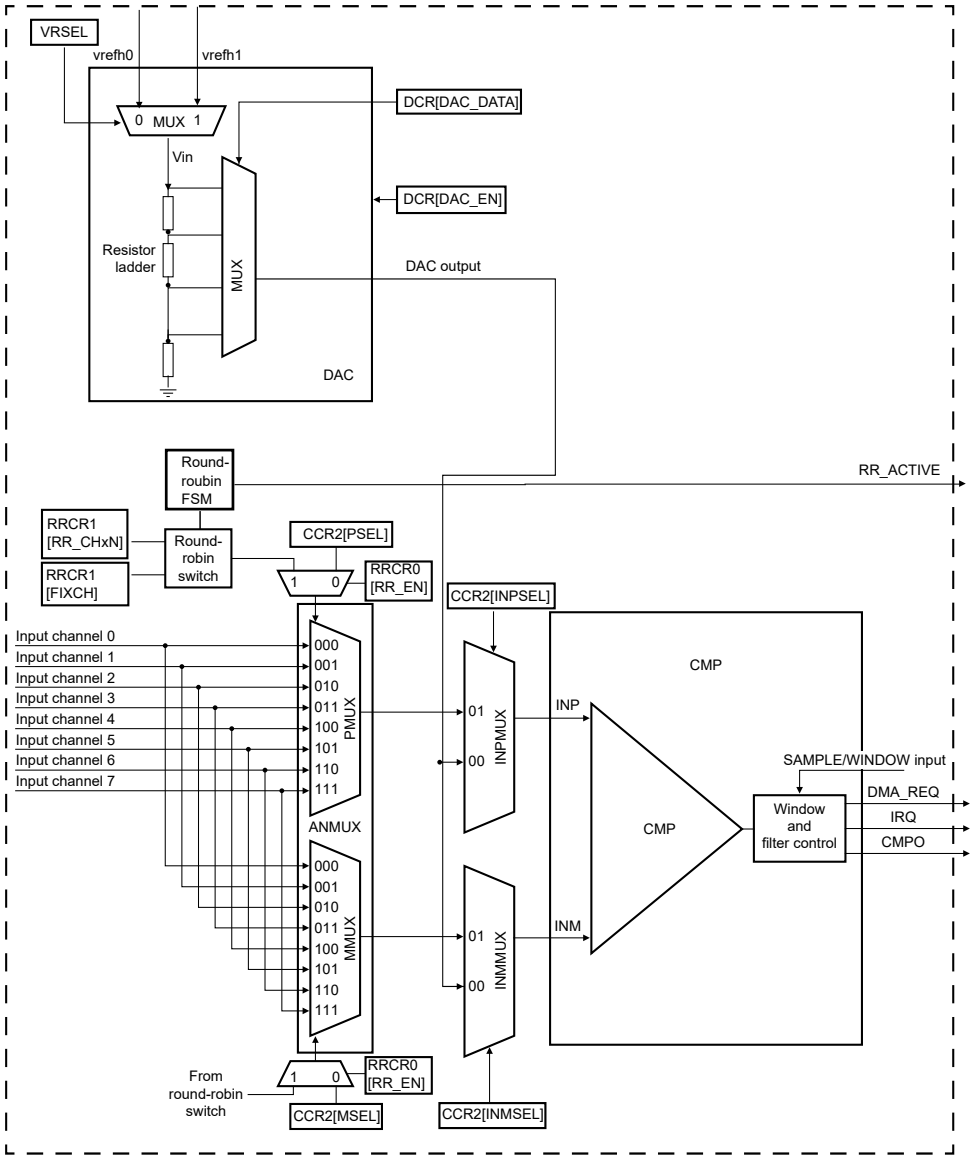



Figure 233. Block diagram

### 58.2.2 Features

The features of the LPCMP module include：

- Includes two 8-to-1 channel MUXes to select input signal from eight channels
- Supports multiple operation modes to produce a wide range of outputs such as:
  - Sampled
  - Windowed, which is ideal for certain PWM zero-crossing-detection applications
  - Digitally filtered
- Provides the following advance features for window and sample:

- Window and sample signals can be inverted.
- CMPO rising, falling or both edges closes the window.
- CMPO level can be defined when window is closed.

- Provides selectable performance levels:
  - Low-Power (speed) mode
  - High-Power (speed) mode
- Supports programmable hysteresis control
- Provides a selectable inversion on comparator output
- Uses an external hysteresis at the same time the output filter is used for internal functions
- Provides interrupt and DMA support
- Supports Round Robin Trigger mode
- Includes an 8-bit resolution DAC
- Provides a selectable supply reference source for DAC

## 58.3 Functional description

### 58.3.1 Functional block diagram

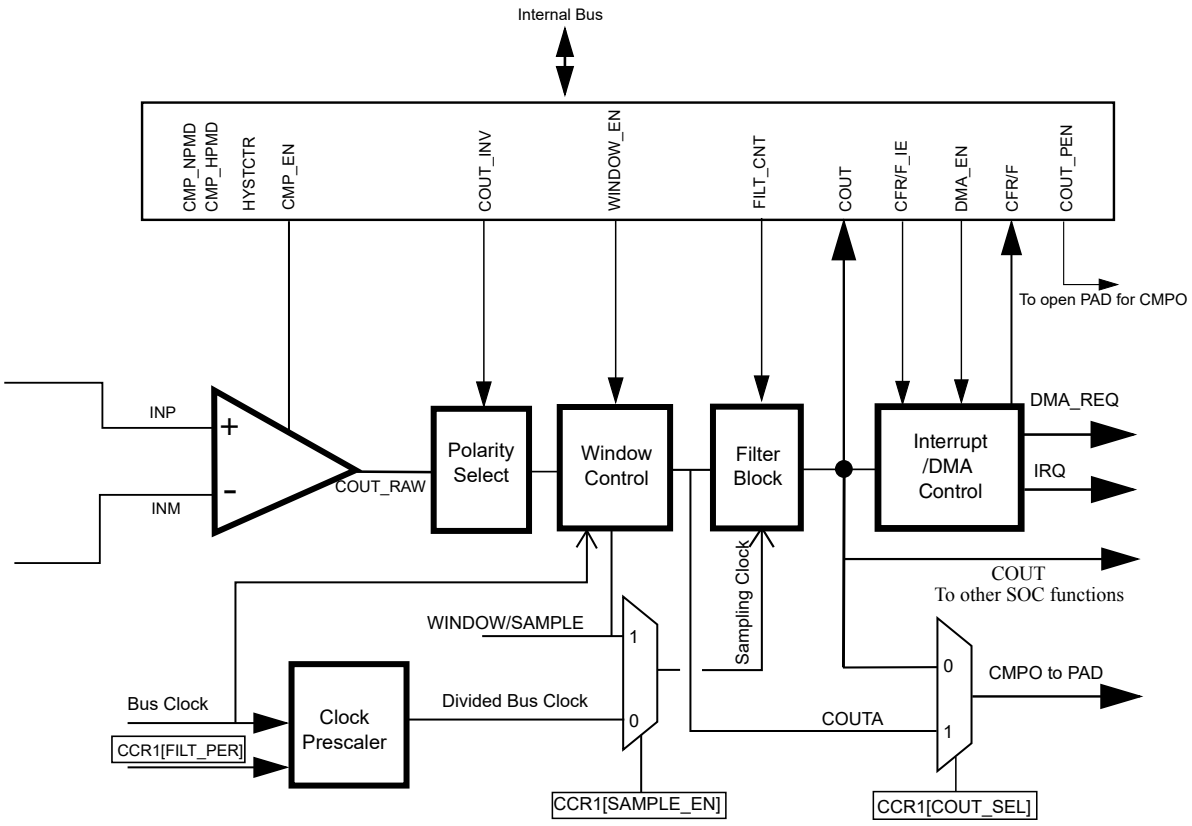



Figure 234. Functional block diagram

As shown in the block diagram, the functions are:

- Compared two analog input voltages applied to INP and INM, COUT_RAW is high when the INP input voltage is greater than the INM input voltage, and COUT_RAW is low when the INP input voltage is less than the INM input voltage.
- The COUT_RAW signal can be inverted by enabling CCR1[COUT_INV].
- The optionally inverted comparator output COUT_RAW is sampled on every bus clock when you enable the CCR1[WINDOW_EN] to generate COUTA. In this case, the comparator output is ignored during time periods when the input voltages are not valid. This is useful when you implement zero-crossing-detection for certain PWM applications.
- The window control block is bypassed when CCR1[WINDOW_EN] is disabled.
- The filter block acts as a simple sampler when CCR1[FILT_CNT] is set to 01h.
- The filter block acts as a filter based on multiple samples when CCR1[FILT_CNT] is set to be greater than 01h.
  - If CCR1[SAMPLE_EN] is set to 1, use the external SAMPLE input as the sampling clock.
  - If CCR1[SAMPLE_EN] is set to 0, use the divided bus clock as the sampling clock.
- Bypasses the filter block when it is not in use.

```
Bypass_Filter_Block = (FILT_CNT == 0x00) |  (~SAMPLE_EN & (FILT_PER == 0x00))
```

- Both COUTA and COUT can be configured as module output CMPO by configuring CCR1[COUT_SEL], and are used for different purposes within the system.
- The optionally filtered COUT can be read directly in CSR[COUT].
- The SAMPLE/WINDOW signal can be inverted by setting CCR1[WINDOW_INV].
- The SAMPLE/WINDOW signal can be closed by CMPO's falling edge and/or rising edge by setting CCR1[WINDOW_CLS] in Window mode.
- In Window mode, when window is closed, define the COUTA value as CCR1[COUTA_OW] by setting CCR1[COUTA_OWEN]. If CCR1[COUTA_OWEN] is not set, COUTA holds the last sampled value.

**NOTE**

See the chip configuration section for the source of SAMPLE/WINDOW input.

### 58.3.2 Round-robin trigger mode

You can enable Round-Robin Trigger mode by setting RRCR0[RR_EN] and CCR0[CMP_EN] to 1. A trigger event initiates a comparison sequence. The next trigger event should not occur before the current sequence completes.

RRCR1[FIXP] and RRCR1[FIXCH] select the reference channel for the plus side mux or the minus side mux. RRCR1[RR_CHnEN] selects active channels.

When a trigger comes, the analog comparator enables. After the comparison sequence completes, the analog comparator disables again. RRCR0[RR_INITMOD] controls the analog stabilization time.

**NOTE**

RR_INITMOD*round robin clock period must be longer than the initialization delay specified in the Comparator and 8-bit DAC electrical specifications section of LPCMP datasheet.

After the stabilization process completes, the round robin manner comparison sequence begins. Sample the comparison result for the selected active channel after RRCR0[RR_NSAM] defines the configurable number of operation clocks.

After all the active channels are sampled/compared, if the comparison result changes from its pre-programmed state, the corresponding flag in RRSR[RR_CHnF] is set. Write to RRCSR[RR_CHnOUT] to configure the pre-programmed state for each channel. Update RRCSR[RR_CHnOUT] to store the last comparison result for each channel. If any flag in RRSR[RR_CHnF] sets, CSR[RRF] also sets. If IER[RRF_IE] sets, an asynchronous interrupt asserts. Note that these flags do not support generating a DMA transfer event.

The following diagram shows the basic flow of this mode. In the diagram, RRCR1[RR_CH1EN], RRCR1[RR_CH3EN], and RRCR1[RR_CH4EN] are 1, so channels #1, #3, and #4 are selected for round-robin depending on their priority setting. RRCR0[RR_NSAM] sets to 2'b01, so you can sample one clock later the comparison result of the selected channel. After you compare the channel #4, the result is sampled, and round-robin ends. If any of the comparison results from channel #1, #3, or #4 changed from their programmed value (written to RRCSR[RR_CH1OUT], RRCSR[RR_CH3OUT], and RRCSR[RR_CH4OUT]), generates an interrupt. Software can then poll RRSR[RR_CHnF] to see which channel input(s) changed value.

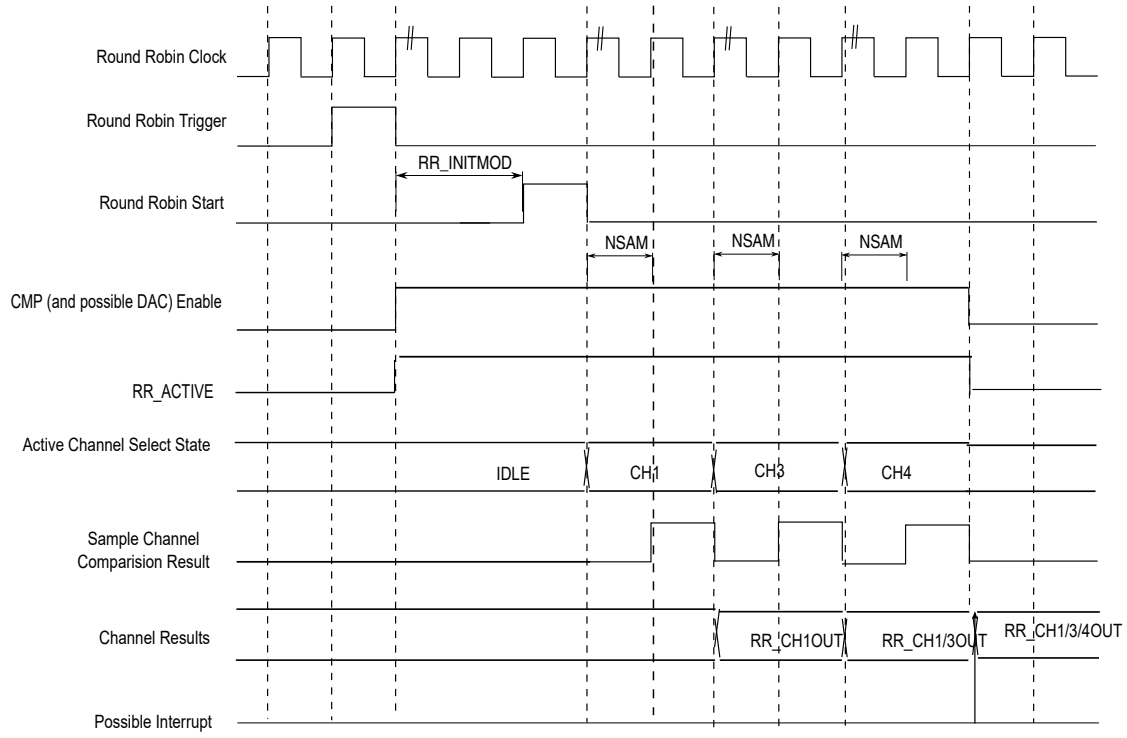



**Figure 235. Trigger mode**

The table below shows the channel decode in both Functional mode and Trigger mode. Other cases not in the table are illegal.

**Table 276. CMP channel decode in functional mode and round-robin trigger mode**

| Mode | RR_EN | PSEL[2:0] | MSEL[2:0] | INPSEL[1:0] | INMSEL[1:0] | FIXP | FIXCH[2:0] | RR_CHxN | INP | INM | CMP Behavior |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Functional mode | 0 | x[1] | 0 to 7 | 0 | 1 | x | x | x | DAC | Channel decoded from MSEL[2:0] | Channel 0 to 7 can be compared with DAC |

*Table continues on the next page...*

Table 276. CMP channel decode in functional mode and round-robin trigger mode (continued)

| Mode | RR_EN | PSEL[2:0] | MSEL[2:0] | INPSEL[1:0] | INMSEL[1:0] | FIXP | FIXCH[2:0] | RR_CHxN | INP | INM | CMP Behavior |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 to 7 | x | 1 | 0 | x | x | x | Channel decoded from PSEL[2:0] | DAC | Channel 0 to 7 can be compared with DAC |
| | | 0 to 7 | 0 to 7 | 1 | 1 | x | x | x | Channel decoded from PSEL[2:0] | Channel decoded from MSEL[2:0] | Channel 0 to 7 can be compared with channel 0 to 7[2] |
| Trigger mode | 1 | x | x | 0 | 1 | 0 | x | 0 to 7 | DAC | Channel sweep (RR_CHxN) | Channel 0 to 7 can be swept with DAC |
| | | x | x | 1 | 0 | 1 | x | 0 to 7 | Channel sweep (RR_CHxN) | DAC | Channel 0 to 7 can be swept with DAC |
| | | x | x | 1 | 1 | 0 | 0 to 7 | 0 to 7 | Channel fixed by FIXCH[2:0] | Channel sweep (RR_CHxN) | Channel 0 to 7 can be swept with a fixed channel(0 to 7)[3] |
| | | x | x | 1 | 1 | 1 | 0 to 7 | 0 to 7 | Channel sweep (RR_CHxN) | Channel fixed by FIXCH[2:0] | Channel 0 to 7 can be swept with a fixed channel(0 to 7)[3] |

1. "x" means "don't care"
2. PSEL should not be same as MSEL.
3. Channel in the sweep side should not be same as the fixed side.

### 58.3.3 Low-pass filter mode

The low-pass filter mode operates on an unfiltered, optionally inverted comparator output COUTA, and generates the filtered and synchronized output COUT. You can configure both COUTA and COUT as module outputs and use for different purposes within the system.

Synchronization and edge detection determine the bit values of status register. They also apply to COUT for all sampling and windowed modes. You can perform filtering using an internal timebase defined by CCR1[FILT_PER], or use an external sample input to determine sample time.

The need for digital filtering and the amount of filtering depends on your requirements. Filtering can become more useful in the absence of an external hysteresis circuit. Without external hysteresis, generate a high-frequency oscillations at COUTA when the selected INM and INP input voltages differ by less than the offset voltage of the differential comparator.

#### 58.3.3.1 Enabling low-pass filter mode

You can enable low-pass filter mode by setting the following:

- CCR1[FILT_CNT] > 01h
- CCR1[FILT_PER] to a nonzero value or writing 1 to CCR1[SAMPLE_EN].

If you use the divided bus clock to drive the low-pass filter, it samples COUTA every CCR1[FILT_PER] bus clock cycle.

If CCR1[SAMPLE_EN] is set to 1, the low-pass filter samples COUTA on each positive transition of the sample input. The output state of the filter changes when all the consecutive CCR1[FILT_CNT] samples agree that the output value has changed.

#### 58.3.3.2 Latency issues

Program the value of CCR1[FILT_PER] or sample period such that the sampling period is longer than the period of the expected noise, ensuring that a given noise spike corrupts only one sample. You must choose the value of CCR1[FILT_CNT] to reduce the probability of noisy samples causing an incorrect transition to recognize. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the power of CCR1[FILT_CNT].

You must trade off the values of CCR1[FILT_PER] or sample period and CCR1[FILT_CNT] against the need for minimal latency in recognizing actual comparator output transitions. The probability of detecting an actual output change within the nominal latency is the probability of a correct sample raised to the power of CCR1[FILT_CNT].

Table 278 summarizes maximum latency values for the various modes of operation in the absence of noise. Filtering latency restarts each time the noise masks an actual output transition.

### 58.3.4 Low power mode operation

Below table introduces the mode of operation of lower power.

**Table 277. Low power mode operation**

| Mode of operation | Description |
|---|---|
| STANDBY | LPCMP can operate only in Continuous mode or Round-robin trigger mode. |

### 58.3.5 Functional modes

You can combine the comparator window and filter features as shown in the following table.

**Table 278. Functional modes**

| Mode # | CMP_EN | WINDOW_EN | SAMPLE_EN | FILT_CNT | FILT_PER | Operation | Maximum latency[1] |
|---|---|---|---|---|---|---|---|
| 1 | 0 | X | X | X | X | See the Disabled mode (#1). | N/A |
| 2A | 1 | 0 | X | 0x00 | X | See the Continuous mode (#2A and #2B). | $T_{PD}$ |
| 2B | 1 | 0 | 0 | X | 0x00 | | |
| 3A | 1 | 0 | 1 | 0x01 | X | See the Sampled, non-filtered mode (#3A and #3B). | $T_{PD} + T_{SAMPLE} + 3T_{per}$ |
| 3B | 1 | 0 | 0 | 0x01 | > 0x00 | | $T_{PD} + (FILT_PER * T_{per}) + 3T_{per}$ |

*Table continues on the next page...*

Table 278. Functional modes (continued)

| Mode # | CMP_EN | WINDOW_EN | SAMPLE_EN | FILT_CNT | FILT_PER | Operation | Maximum latency[1] |
|---|---|---|---|---|---|---|---|
| 4A | 1 | 0 | 1 | > 0x01 | X | See the Sampled, filtered mode (#4A and #4B). | $T_{PD}$ + (FILT_CNT * $T_{SAMPLE}$) + $3T_{per}$ |
| 4B | 1 | 0 | 0 | > 0x01 | > 0x00 | | $T_{PD}$ + (FILT_CNT * FILT_PER x $T_{per}$) + $3T_{per}$ |
| 5A | 1 | 1 | 0 | 0x00 | X | See the Windowed mode (#5A and #5B). | $T_{PD}$ + $2T_{per}$ |
| 5B | 1 | 1 | 0 | X | 0x00 | | |
| 6 | 1 | 1 | 0 | 0x01 | > 0x00 | See the Windowed/ Resampled mode (#6). | $T_{PD}$ + (FILT_PER * $T_{per}$) + $3T_{per}$ |
| 7 | 1 | 1 | 0 | > 0x01 | > 0x00 | See the Windowed/ Filtered mode (#7). | $T_{PD}$ + (FILT_CNT * FILT_PER x $T_{per}$) + $3T_{per}$ |
| All other combinations of CMP_EN, WINDOW_EN, SAMPLE_EN, FILT_CNT, and FILT_PER are illegal. | | | | | | | |

1. $T_{PD}$ represents the intrinsic delay of the analog component plus the polarity select logic. $T_{SAMPLE}$ is the clock period of the external sample clock. $T_{per}$ is the period of the bus clock.

#### 58.3.5.1 Disabled mode (#1)

In this mode:

- The analog comparator is non-functional and consumes no power.
- CSR[COUT] and CMPO are the same as CCR1[COUT_INV].

### 58.3.5.2 Continuous mode (#2A and #2B)

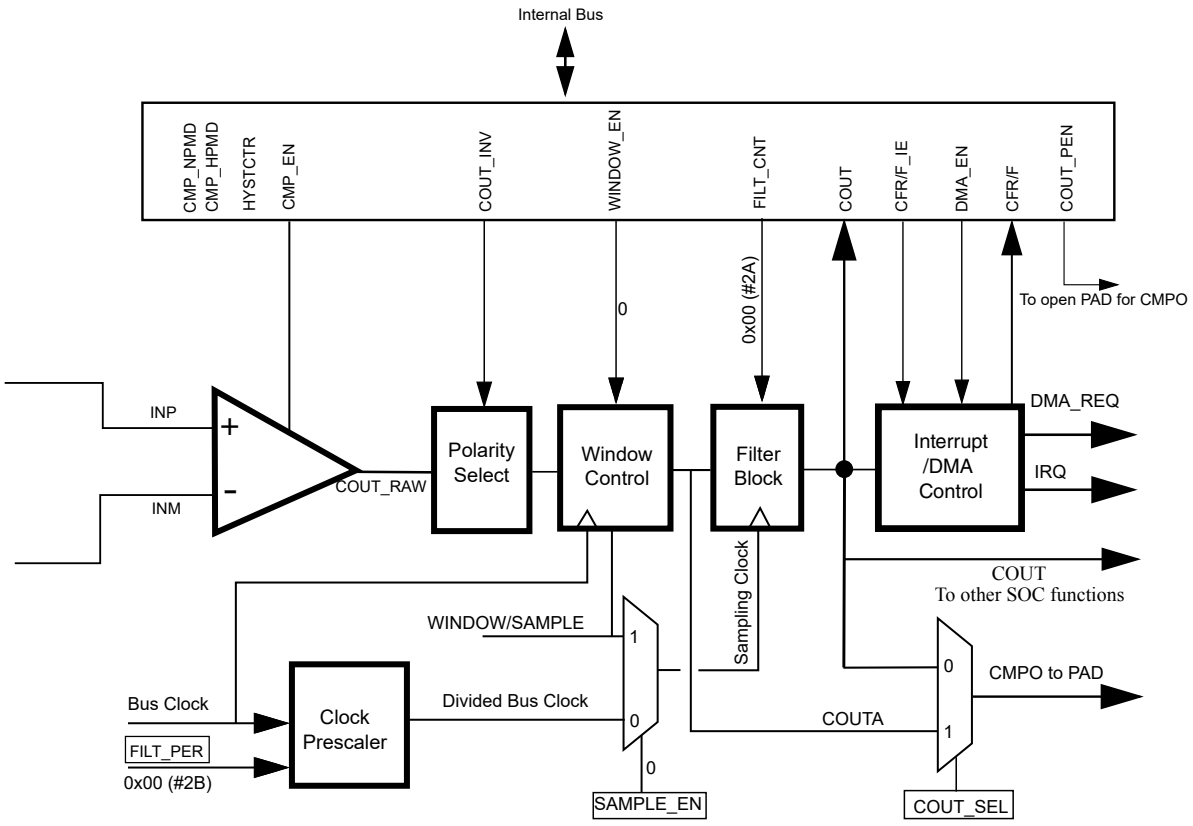



Figure 236. Comparator operation in continuous mode

COUT_RAW is optionally inverted in this mode but is not subject to external sampling or filtering. Both window control and filter blocks bypass completely, and CSR[COUT] updates continuously. The path from comparator input pins to output pins operates in a combinational (unclocked) mode. COUT and COUTA are identical in this mode.

For cases where a comparator drives a fault input, you must configure it to operate in Continuous mode so that an external fault can immediately pass to the target fault circuitry through the comparator.

### 58.3.5.3 Sampled, non-filtered mode (#3A and #3B)

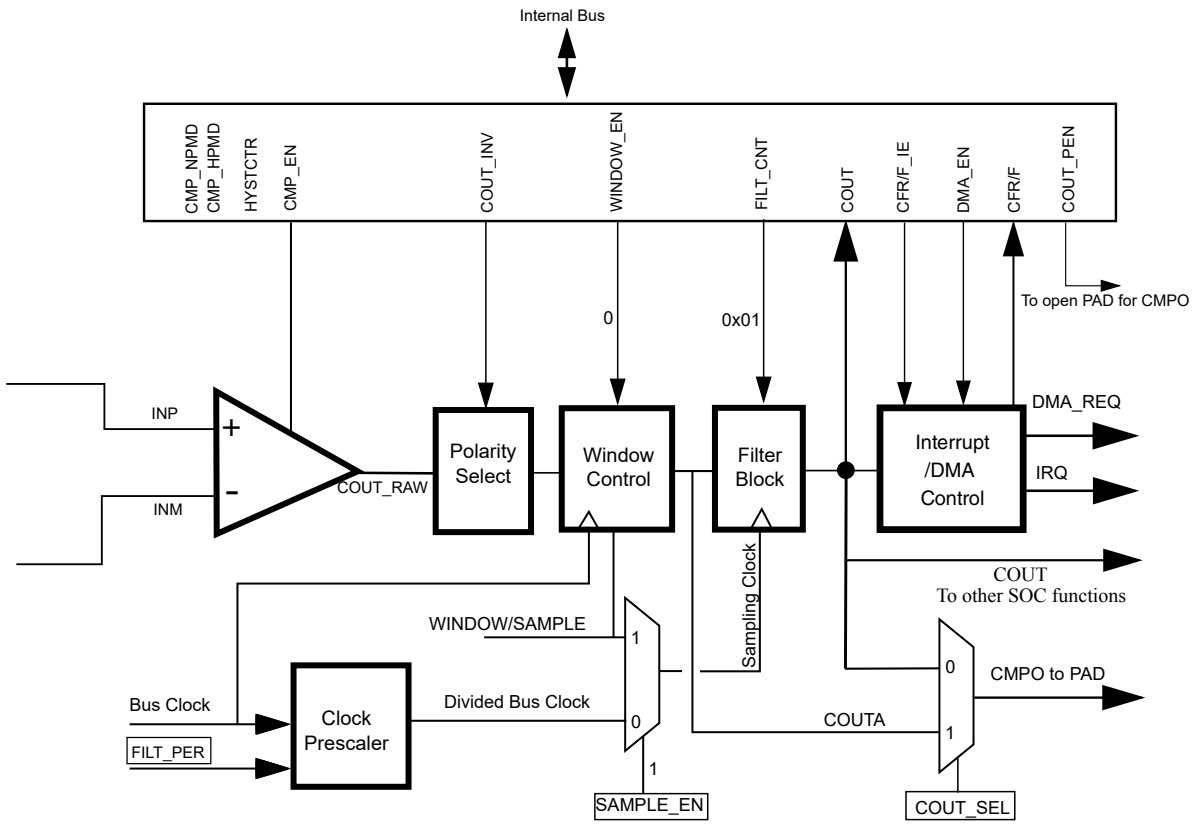



Figure 237. Sampled, non-filtered (#3A): sampling point externally driven

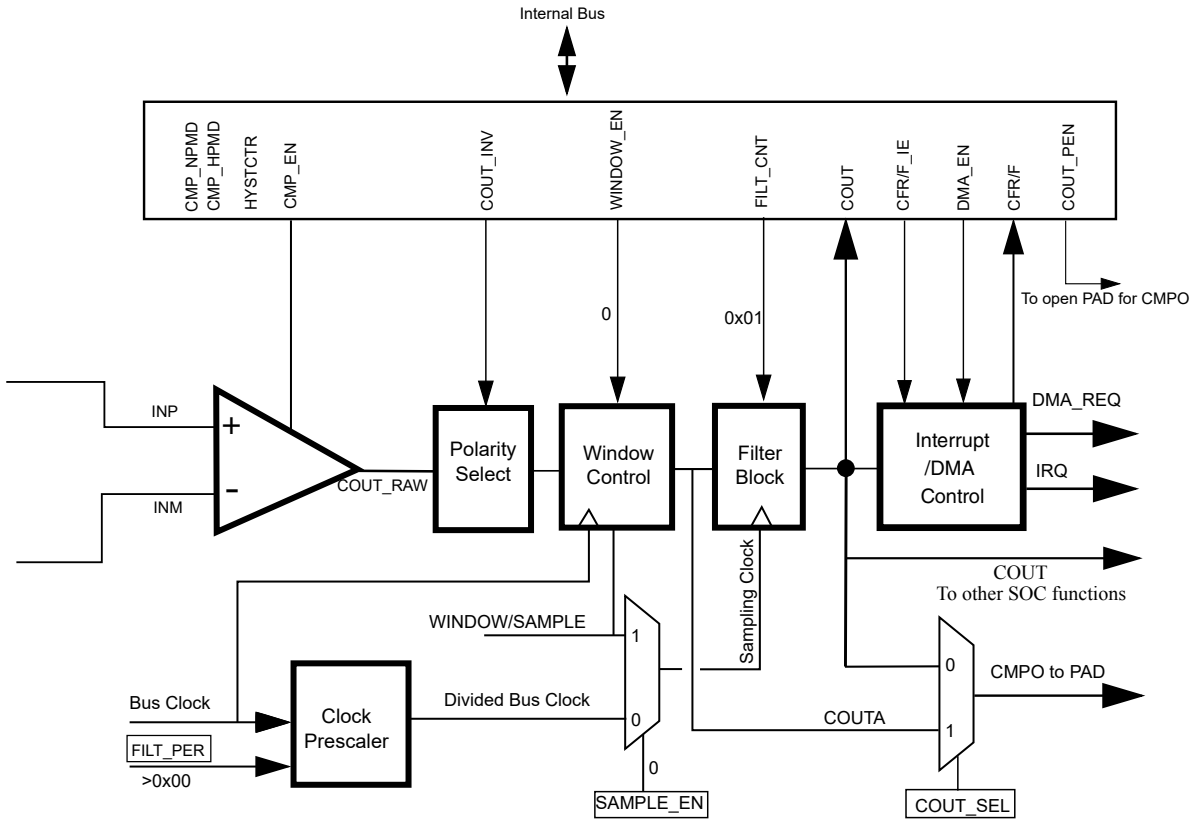



Figure 238. Sampled, Non-Filtered (#3B): sampling interval internally derived

In this mode, the path from analog inputs to COUTA is combinational (unclocked). Windowing control bypasses completely. You can sample COUTA whenever you detect a rising edge on the sampling clock.

The difference in two operation modes (#3A and #3B) of sampled, non-filtered mode is that how you drive the clock to the filter block. In #3A, the clock to filter block drives externally, and in #3B, the clock to filter block drives internally.

The filter block has no other function than sample or hold of the comparator output in this mode.

The following figure shows the comparator operation in this mode, assuming that the polarity select sets to a non-inverting state.

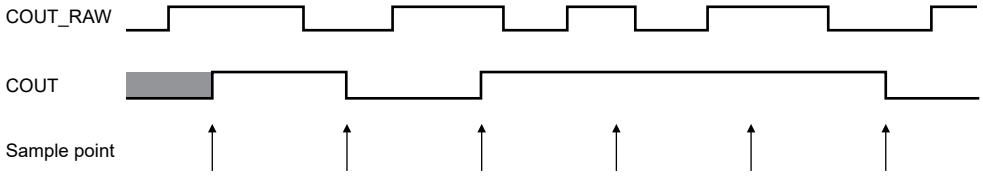



Figure 239. Sampled, Non-Filtered mode timing diagram

### 58.3.5.4 Sampled, filtered mode (#4A and #4B)

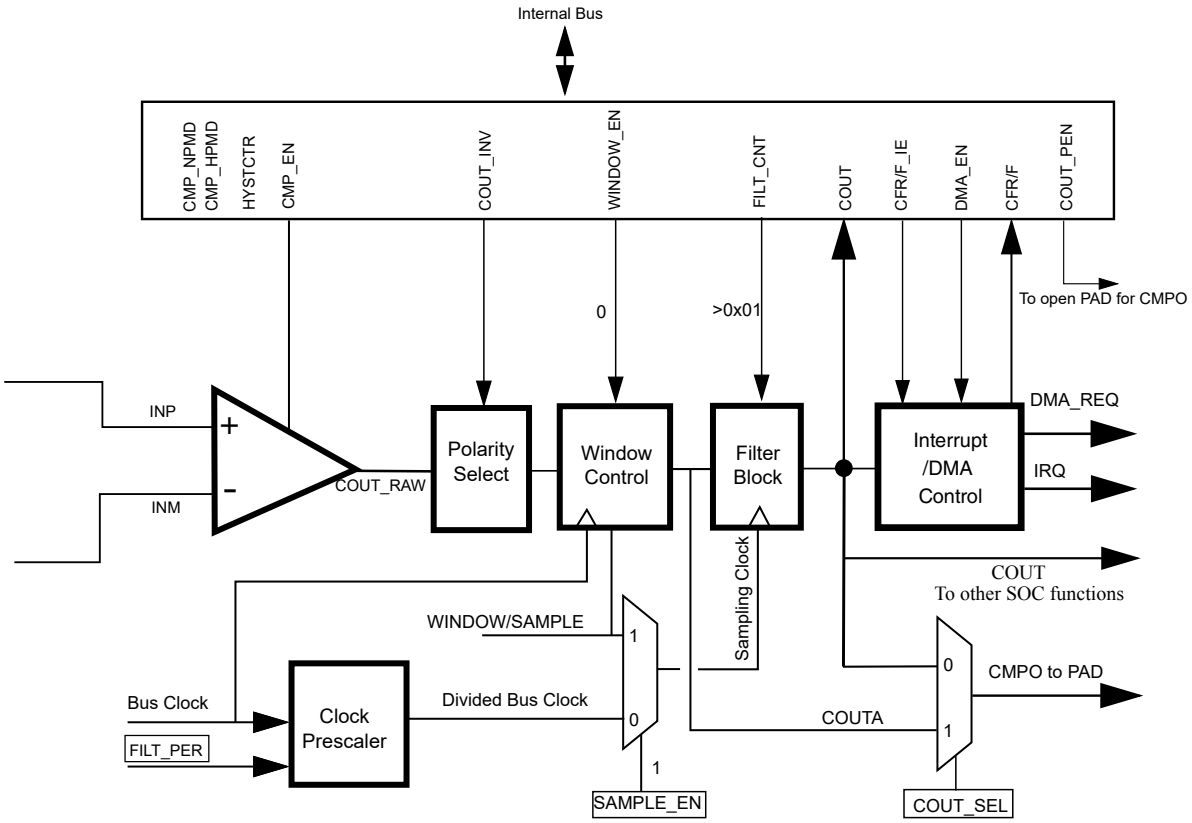



Figure 240. Sampled, filtered (#4A): sampling point externally driven

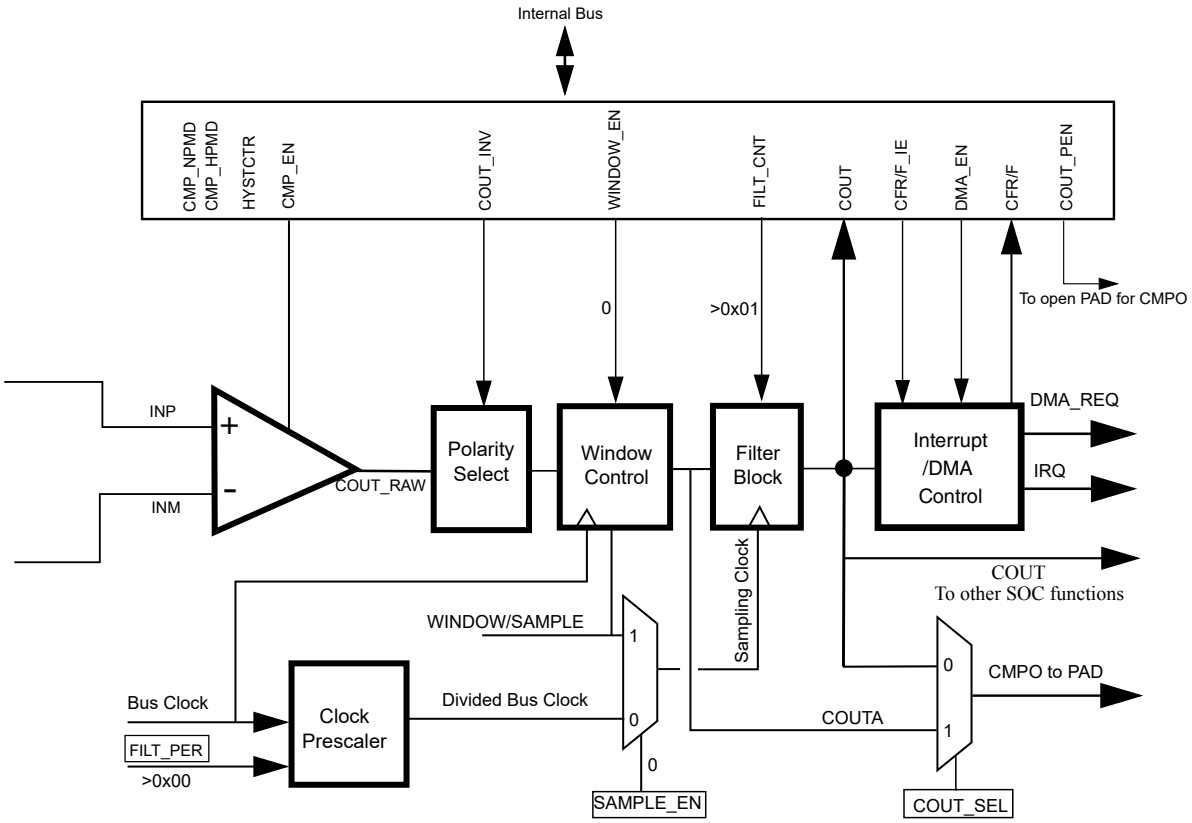



**Figure 241. Sampled, filtered (#4B): sampling point internally derived**

In this mode, the path from the analog inputs to COUTA is combinational(unclocked). Windowing control bypasses completely. You can sample COUTA whenever you detect a rising edge on the sampling clock.

The only difference in operation between sampled, non-filtered (#3A) mode and sampled, filtered (#4A) mode is that CCR1[FILT_CNT] is larger than 1, which activates filter operation.

The only difference in operation between sampled, non-filtered (#3B) mode and sampled, filtered (#4B) mode is that CCR1[FILT_CNT] is larger than 1, which activates filter operation.

### 58.3.5.5 Windowed mode (#5A and #5B)

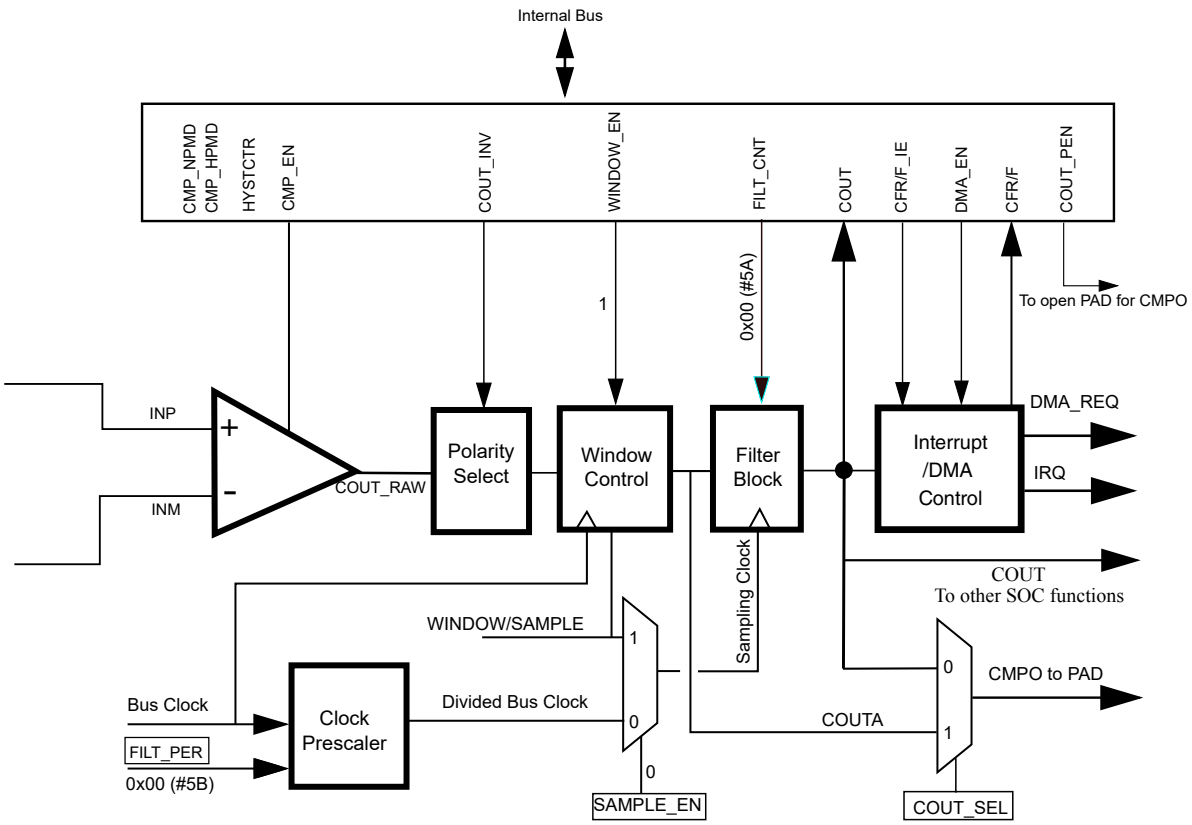



Figure 242. Windowed mode

The bus clock clocks COUTA whenever you enable the window in this mode. The last latched value holds after you disable the window and the filter block is bypassed.

The following figure shows the comparator operation in this mode, ignoring the latency of the analog comparator, polarity select, and window control block. The polarity select sets to a non-inverting state.

COUTA may lag the analog inputs by up to two functional clock cycles plus the combinational path delay through the comparator and polarity select logic in the actual operation.

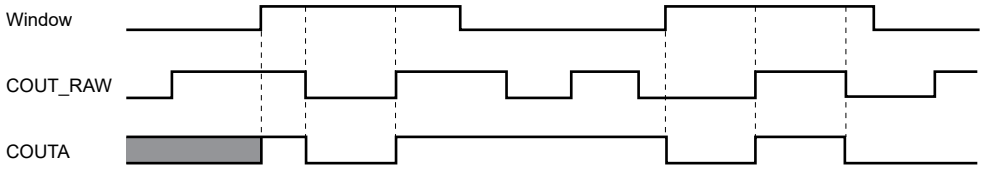



Figure 243. Windowed mode timing diagram

The following figure shows that if CCR1[COUTA_OWEN] becomes 1, you can define COUTA level as CCR1[COUTA_OW], after you closes the window.

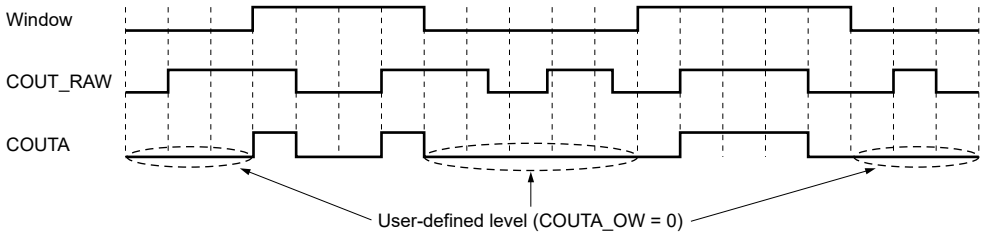



**Figure 244. Windowed mode timing diagram with user defined value 0 outside window**

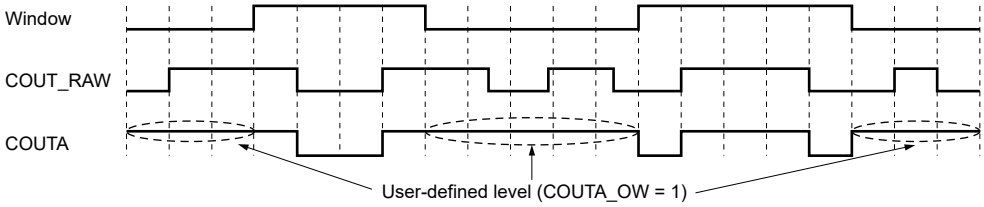



**Figure 245. Windowed mode timing diagram with user defined value 1 outside window**

**NOTE**

When the window is open, COUT_A will switch from COUTA_OW to COUT_RAW. When the window is closed, COUT_A will switch from COUT_RAW to COUTA_OW. This may generate unnecessary transition flags, for instance, CFR or CFF. User needs to choose COUTA_OW carefully according to the actual application, and select the appropriate flag CFR or CFF to generate interrupt.

If CCR1[WINDOW_CLS] becomes 1, you can define the CMPO event (rising edge, falling edge or both edges that CCR1[EVT_SEL] selects) to close the window. The external window signal has to go to zero and back to one to enable the internal window again. The following figure shows an example that CMPO rising edge closes the internal window.

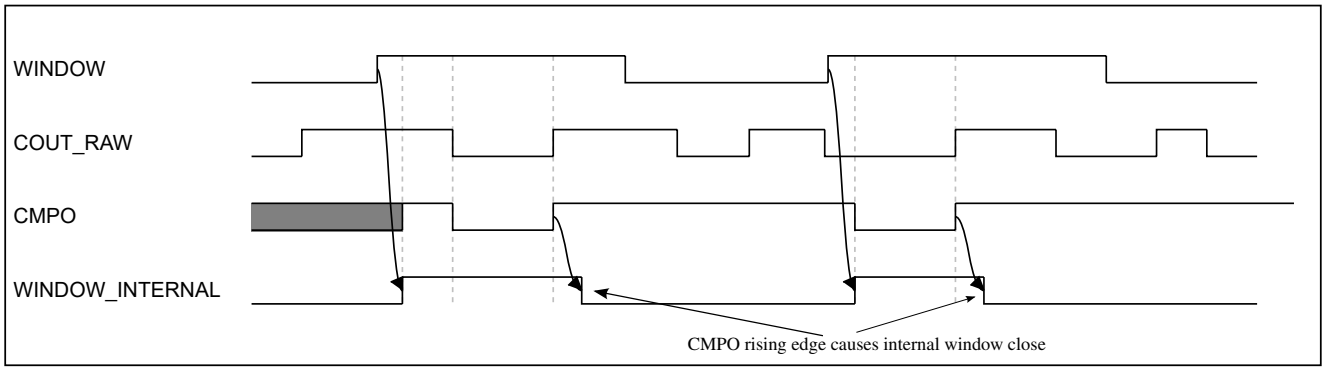



**Figure 246. Windowed mode timing diagram with CMPO rising edge close window**

The following figure shows that if CCR1[WINDOW_INV] becomes 1, you can invert the window signal before you use it.

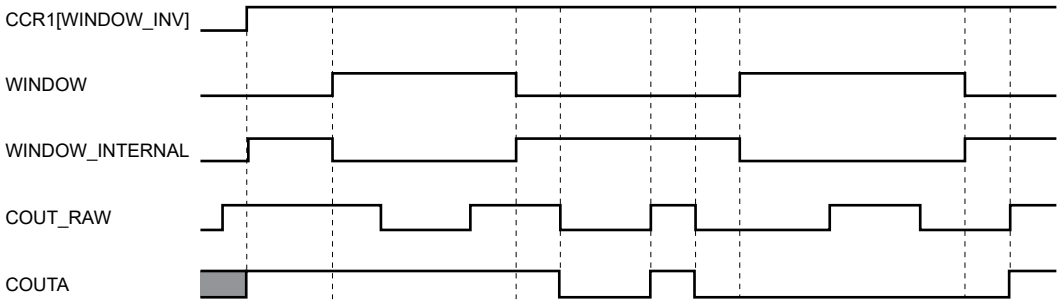



Figure 247. Windowed mode timing diagram with window signal inverted

### 58.3.5.6 Windowed/Resampled mode (#6)

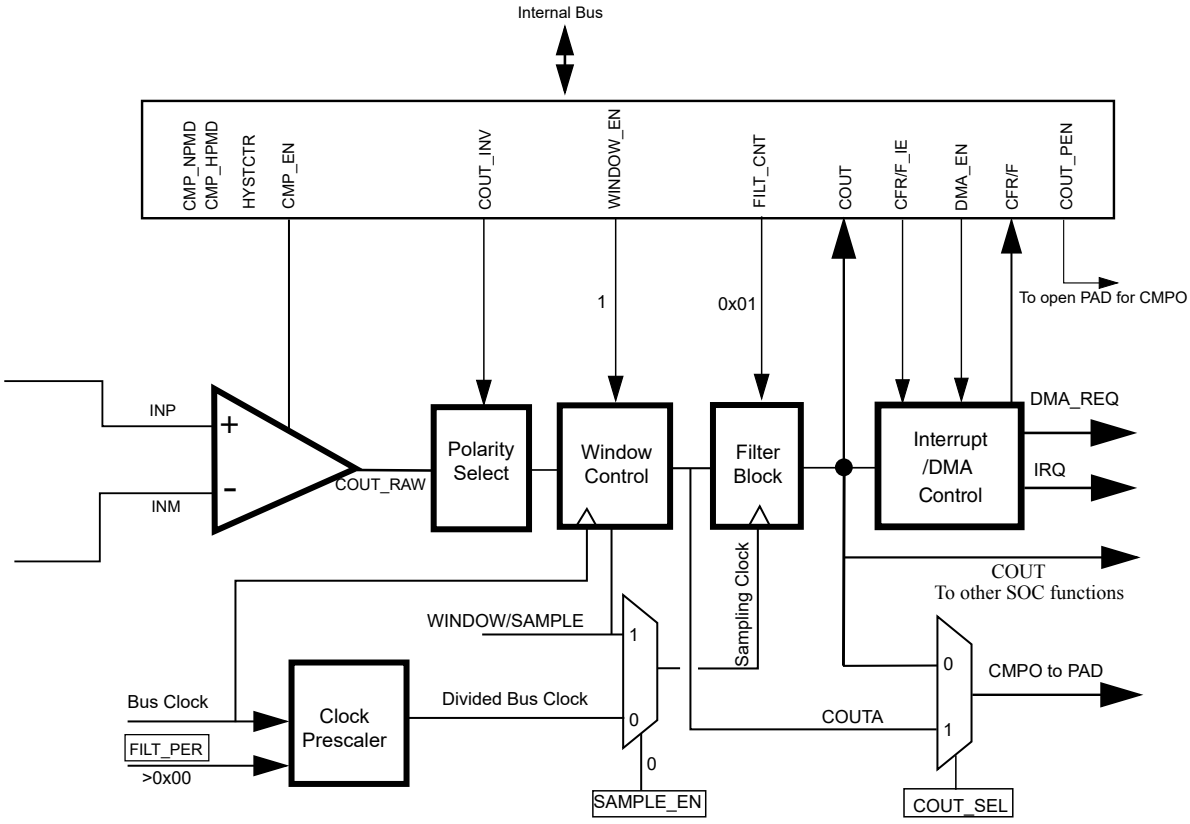



Figure 248. Windowed/Resampled mode

This mode of operation results in an unfiltered string of comparator samples where CCR1[FILT_PER] and the bus clock rate determines the interval between the samples. The following section shows that the configuration for this mode is virtually identical to that for the Windowed/Filtered mode. The only difference is that the value of CCR1[FILT_CNT] must be 1 in this mode.

The following figure uses the same input stimulus shown in Figure 243, and adds resampling of COUTA to generate COUT. The arrows in the figure indicate the time points at which the samples are taken. You can ignore prop delays and latency for clarity.

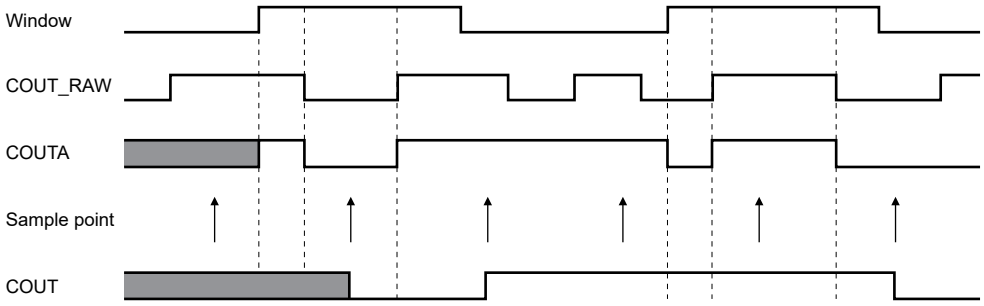



Figure 249. Windowed/Resampled mode operation

This example demonstrates the operation of the comparator in Windowed/Resampled mode, and does not reflect any specific application. Based on the sampling rate and window placement, COUT may not see zero-crossing events that the analog comparator detects. You must carefully consider the sampling period and/or window placement for a given application.

### 58.3.5.7 Windowed/Filtered mode (#7)

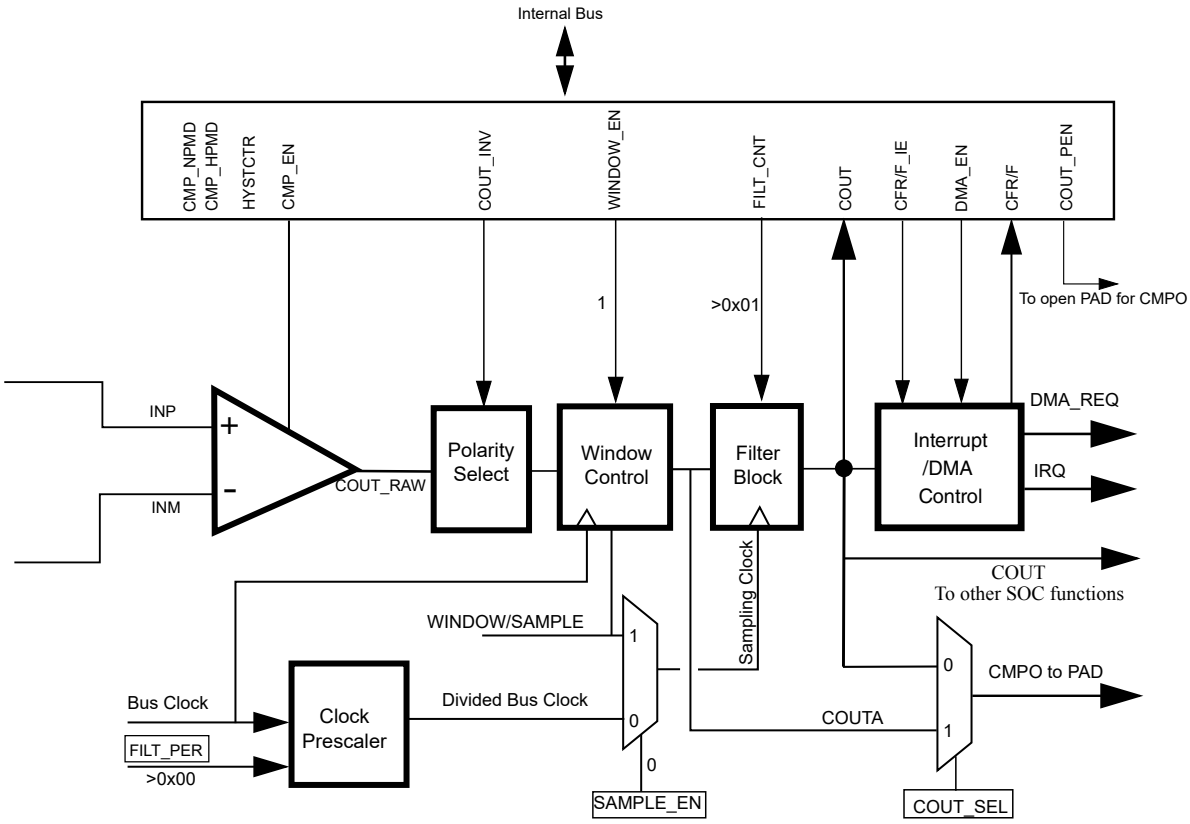



Figure 250. Windowed/Filtered mode

The only difference in operation between Windowed/Resampled mode (#6) and Windowed/Filtered mode (#7) is that CCR1[FILT_CNT] is >1, which activates filter operation.

This mode is the most complex mode of operation for the comparator block, as it utilizes both windowing and filtering features. It also has the highest latency of any of the modes. This is approximately: up to 2 peripheral clock synchronization in the window function + ((CCR1[FILT_CNT] x CCR1[FILT_PER]) + 1) x peripheral clock for the filter function.

### 58.3.6 DMA

After DMA is enabled by writing 1 to CCR1[DMA_EN] and interrupt is enabled by writing 1 to IER[CFR_IE], IER[CFF_IE], or both, the corresponding change on COUT forces a DMA transfer request rather than a CPU interrupt. After the DMA completes the transfer, it sends a transfer completing indicator signal that deasserts the DMA transfer request and clears the flags (both CSR[CFR] and CSR[CFF]) to allow a subsequent change on comparator output to occur and forces another DMA request.

### 58.3.7 Clocking

LPCMP requires the following clocks to operate:

**Table 279. LPCMP clocks**

| Type of clock | Description |
|---|---|
| Bus | Controls the access to LPCMP registers and window/ filter function. |
| Round-robin clock (RCLK) | Controls Round-robin trigger mode. |

### 58.3.8 Resets

The global chip reset signal resets LPCMP.

### 58.3.9 Interrupts

After the corresponding IER becomes 1, CSR[CFR], CSR[CFF], and CSR[RRF] can generate an interrupt, assuming that CCR1[DMA_EN] is not 1. You can clear either the flag or IER to deassert the interrupt.

## 58.4 External signal descriptions

Below table introduces external signals.

**Table 280. External signal descriptions**

| Signal | Description | I/O |
|---|---|---|
| CMPO | Filtered or unfiltered comparator output | O |
| Input_Analog_Channels | Analog input channels (see the chip-specific information for more on the connections). | I |
| VREFH_EXT | External reference voltage for the CMP-DAC (see the chip-specific information for more on the connections). | I |
| RR_ACTIVE | Round-robin trigger mode enabled. | O |

## 58.5 Initialization

You can enable LPCMP by writing 1 to CCR0[CMP_EN], and then configuring the control registers (CCR1, CCR2, DCR, and so on).

To disable LPCMP, write 0 to CCR0[CMP_EN]. Switching operation modes or changing control register fields on-the-fly (when CCR0[CMP_EN] is set to 1) may cause noise on the COUT or COUTA signals. To avoid unwanted signal noise, you must ensure to disable the module before switching modes or changing control fields.

The time required to stabilize COUT is the power-on delay of the comparators plus the largest propagation delay from a selected analog source through the analog comparator, windowing function, and filter (see the Comparator and 8-bit DAC electrical specifications section of LPCMP datasheet for more information on propagation delay and power-up delay). Table 278 specifies the delay that the windowing and filter function causes.

During operation, you must always consider the propagation delay of the selected data paths. It can take many bus clock cycles for COUT and CSR[CFR]/CSR[CFF] to reflect an input change or a configuration change to one of the components involved in the data path.

## 58.6 Application information

### 58.6.1 Round-robin trigger mode programing recommendation

Configure the Round-robin trigger mode as follows:

1. Configure the comparison cycles by RRCR0[RR_NSAM]. Note: It is a mandatory request that the round robin cycling period must set longer than the time that all the active channels complete the specified comparison cycles set by RRCR0[RR_NSAM].
2. Configure CMP initialization delay by RRCR0[RR_INITMOD]. Note: In programming RRCR0[RR_INITMOD], the RR_INITMOD x round robin clock period must be longer than the initialization delay, see the LPCMP datasheet for more information.
3. Configure RRCR1[FIXP] to select the fixed port of CMP and
    a. If you use one input channel to compare with other channels, configure RRCR1[FIXCH] to select the fixed channel.
    b. If you use DAC output to compare with input channels, configure CCR2[INPSEL] or CCR2[INMSEL] (according to RRCR1[FIXP] ) to select the DAC output.
4. Configure channels for comparison by RRCR1[RR_CHnEN].
5. Write RRCSR[RR_CHnOUT] to define the pre-set state of channel n.
6. Clear channel flags RRSR[RR_CHnF].
7. Enable round robin interrupt by IER[RRF_IE] (disable IER[CFR_IE] and IER[CFF_IE]).
8. Enable round-robin trigger mode by setting RRCR0[RR_EN] to 1.
9. Enable comparator by setting CCR0[CMP_EN] to 1.

### 58.6.2 Round-robin clock (RCLK) frequency requirement

(1) RCLK high frequency limit

RCLK high frequency limit depends on two facts:

1. The analog CMP and DAC initialization time (see the chip data sheet for more information on the initialization time.)
    - RRCR0[RR_INITMOD] provides a maximum 63 RCLK cycles for the analog CMP and DAC initialization.
    - RCLK must be slow to assure: 63 * ($1/f_{RCLK}$) > $T_{initialization}$, where $f_{RCLK}$ is in MHz, and $T_{initialization}$ is in microsecond.
    - so $f_{RCLK} < 63 / T_{initialization}$
    - Example: $T_{initialization}$ = 40 µs, then $f_{RCLK}$ should be smaller than 1.575 MHz.
2. The analog CMP propagation delay (see the Comparator and 8-bit DAC electrical specifications section of LPCMP datasheet for more information on the CMP propagation delay.)
    - RRCR0[RR_NSAM] provides a maximum 4 RCLK cycles for the analog CMP propagation delay.
    - RCLK must be slow to assure: 4 * ($1/f_{RCLK}$) > $T_{propagation}$, where $f_{RCLK}$ is in MHz, and $T_{propagation}$ is in microsecond.
    - $f_{RCLK} < 4 / T_{propagation}$
    - Example: $T_{propagation}$ = 0.1 µs, then $f_{RCLK}$ must be smaller than 40 MHz.

(2) RCLK low frequency limit

In theory, RCLK frequency has no low limit. But the lower the RCLK frequency, the longer the scan time. Therefore, the lower limit of the RCLK frequency depends on the system application.

## 58.7 LPCMP register descriptions

The memory map comprises of 32-bit aligned registers, which you can access via 8-, 16- or 32-bit reads and 32-bit write. Attempted accesses using unsupported write data sizes, writes to read-only resources, or to reserved spaces terminate with an error. Read access to reserved address generates a transfer error and the read data bus shows all 0s.

### 58.7.1 LPCMP memory map

LPCMP_0 base address: 4037_0000h

LPCMP_1 base address: 4037_4000h

LPCMP_2 base address: 404E_8000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID (VERID) | 32 | R | 0100_0001h |
| 4h | Parameter (PARAM) | 32 | R | 0000_0002h |
| 8h | Comparator Control Register 0 (CCR0) | 32 | RW | 0000_0002h |
| Ch | Comparator Control Register 1 (CCR1) | 32 | RW | 0000_0000h |
| 10h | Comparator Control Register 2 (CCR2) | 32 | RW | 0000_0000h |
| 18h | DAC Control (DCR) | 32 | RW | 0000_0000h |
| 1Ch | Interrupt Enable (IER) | 32 | RW | 0000_0000h |
| 20h | Comparator Status (CSR) | 32 | RW | 0000_0000h |
| 24h | Round Robin Control Register 0 (RRCR0) | 32 | RW | 0000_0000h |
| 28h | Round Robin Control Register 1 (RRCR1) | 32 | RW | 0000_0000h |
| 2Ch | Round Robin Control and Status (RRCSR) | 32 | RW | 0000_0000h |
| 30h | Round Robin Status (RRSR) | 32 | RW | 0000_0000h |

### 58.7.2 Version ID (VERID)

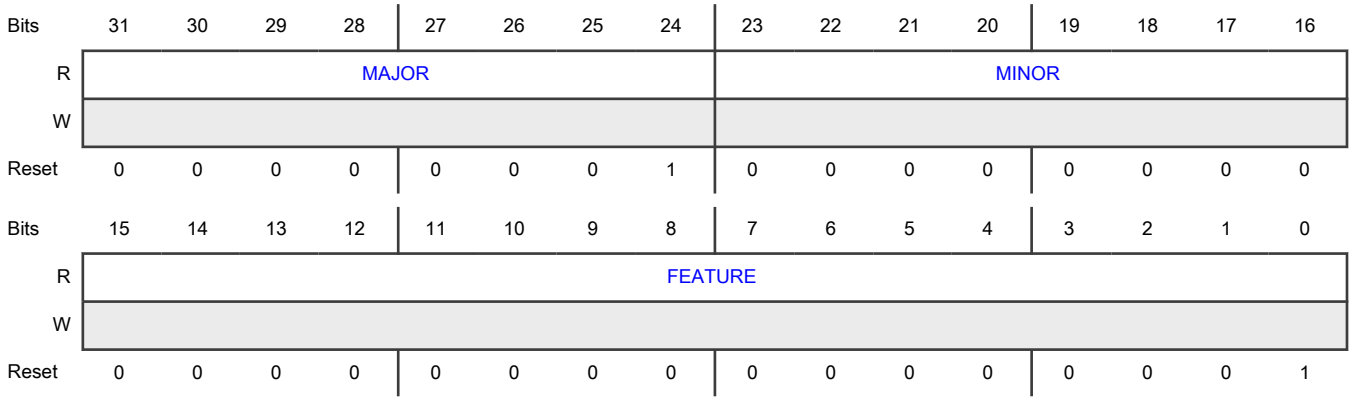
**Offset**

| Register | Offset |
|---|---|
| VERID | 0h |

**Function**

Contains version numbers for the module design and feature set.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>Returns the major version number for the module design. |
| 23-16<br>MINOR | Minor Version Number<br>Returns the minor version number for the module design. |
| 15-0<br>FEATURE | Feature Specification Number<br>Returns the feature set number.<br>0000_0000_0000_0001b - Round robin feature |

## 58.7.3 Parameter (PARAM)

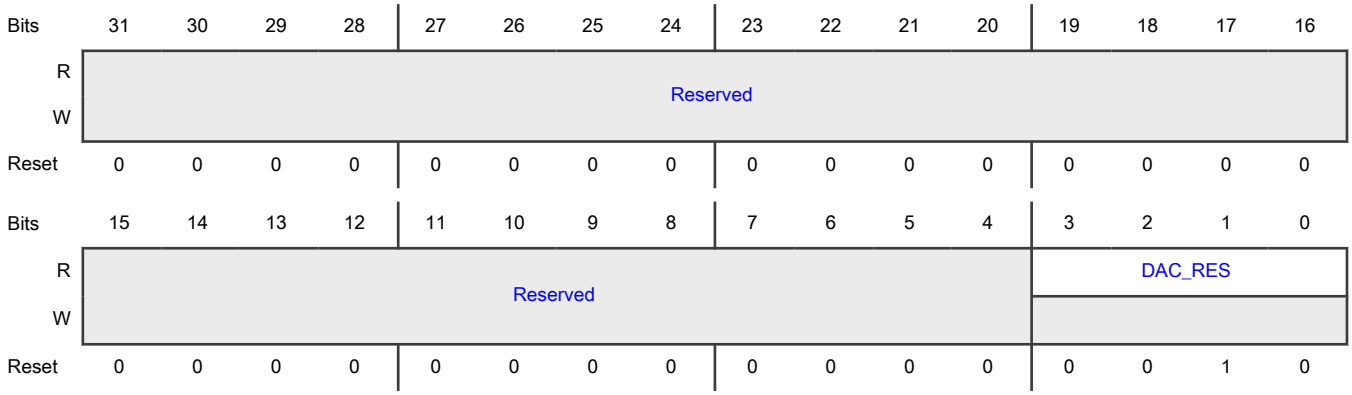
### Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

### Function

Contains parameter values that are implemented in the module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>DAC_RES | DAC Resolution<br>Indicates supported DAC resolutions.<br>**NOTE**<br>All other bit field values are reserved.<br>0000b - 4-bit DAC<br>0001b - 6-bit DAC<br>0010b - 8-bit DAC<br>0011b - 10-bit DAC<br>0100b - 12-bit DAC<br>0101b - 14-bit DAC<br>0110b - 16-bit DAC |

## 58.7.4 Comparator Control Register 0 (CCR0)

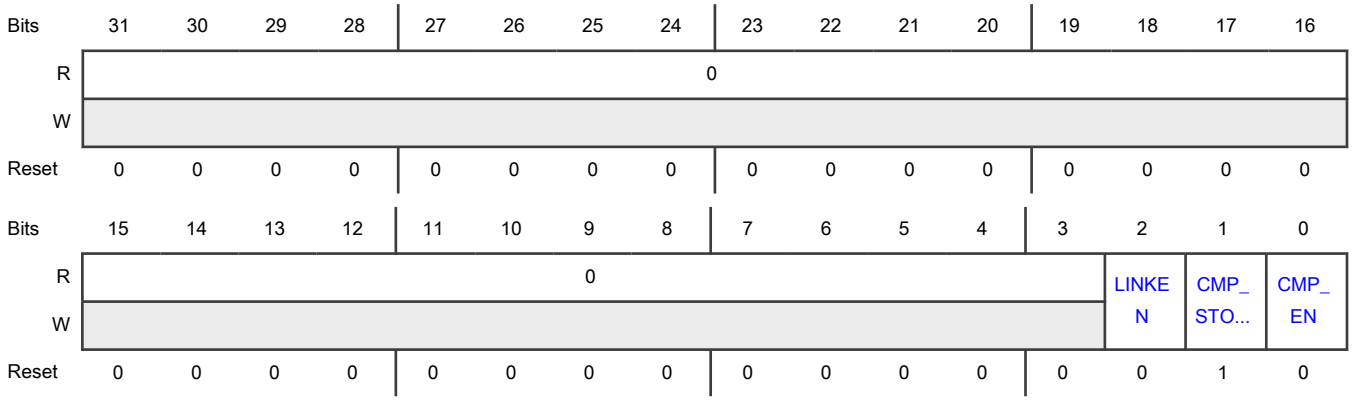
### Offset

| Register | Offset |
|---|---|
| CCR0 | 8h |

### Function

Contains configuration options for enabling the analog comparator and the DAC.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2<br>LINKEN | CMP-to-DAC Link Enable<br>Enables the CMP-to-DAC link.<br>0b - Disable: enabling or disabling the DAC is independent from enabling or disabling the CMP.<br>1b - Enable: enabling/disabling DAC is controlled by the CMP_EN bit instead of DCR[DAC_EN]. Also, when the CMP is auto-disabled because software selects the same signal for both the plus and minus comparator inputs, the DAC is disabled too. |
| 1<br>CMP_STOP_EN | Comparator STANDBY Mode Enable<br>Enables the analog comparator or the DAC when the module is in STANDBY mode.<br>**NOTE**<br>This field has no effect in Round-robin Trigger mode.<br>0b - Disables the analog comparator regardless of CMP_EN.<br>1b - Allows CMP_EN to enable the analog comparator. |
| 0<br>CMP_EN | Comparator Enable<br>Enables the analog comparator.<br>**NOTE**<br>When CCR0[LINKEN]=1, CMP_EN also controls the enabling/disabling of the DAC instead of DCR[DAC_EN].<br>0b - Disable (The analog logic remains off and consumes no power.)<br>1b - Enable |

## 58.7.5 Comparator Control Register 1 (CCR1)

### Offset

| Register | Offset |
|---|---|
| CCR1 | Ch |

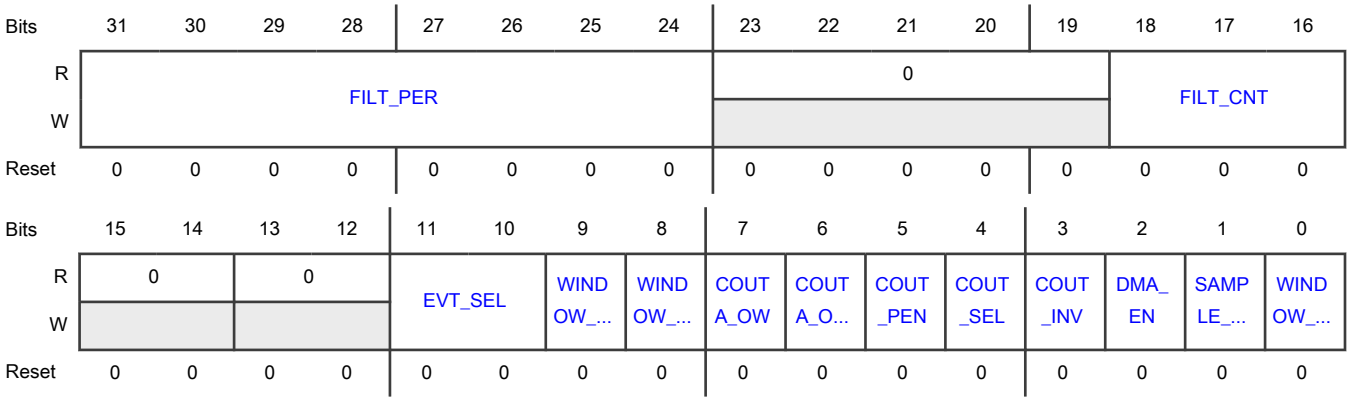
### Function

Contains configuration options for the comparator operation, such as enabling Sampling or Windowing mode.

**NOTE**

You cannot enable Sampling and Windowing modes both at the same time. Sampling mode takes precedence over Windowing mode. If you write 1 to both SAMPLE_EN and WINDOW_EN, only SAMPLE_EN becomes 1.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>FILT_PER | Filter Sample Period<br>Specifies the sampling period (in bus clock cycles) of the comparator output filter. Programming this field to 00h bypasses the filter. See Functional description for more information on filter programming and latency.<br>**NOTE**<br>FILT_PER has no effect in Sampling mode (CCR1[SAMPLE_EN] = 1). |
| 23-19<br>— | Reserved |
| 18-16<br>FILT_CNT | Filter Sample Count<br>Specifies the number of consecutive samples that must agree before the comparator output filter accepts the sample as a new valid output state. See Functional description for more information on filter programming and latency. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 000b - Filter is bypassed: COUT = COUTA<br>001b - 1 consecutive sample (Comparator output is simply sampled.)<br>010b - 2 consecutive samples<br>011b - 3 consecutive samples<br>100b - 4 consecutive samples<br>101b - 5 consecutive samples<br>110b - 6 consecutive samples<br>111b - 7 consecutive samples |
| 15-14<br>— | Reserved |
| 13-12<br>— | Reserved |
| 11-10<br>EVT_SEL | CMPO Event Select<br>Selects which CMPO signal edge (rising, falling, or both) defines a CMPO event.<br>**NOTE**<br>Valid only in Windowing mode.<br>00b - Rising edge<br>01b - Falling edge<br>1xb - Both edges |
| 9<br>WINDOW_CLS | CMPO Event Window Close<br>Enables a CMPO event (defined as a CMPO rising edge, falling edge, or both) to close an active window. See EVT_SEL to configure the CMPO event.<br>**NOTE**<br>The WINDOW signal has to go to zero and back to one again to re-activate the window.<br>Valid only in Windowing mode.<br>0b - CMPO event cannot close the window<br>1b - CMPO event can close the window |
| 8<br>WINDOW_INV | WINDOW/SAMPLE Signal Invert<br>Inverts the window/sample signal.<br>0b - Do not invert<br>1b - Invert |
| 7 | COUTA Output Level for Closed Window |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| COUTA_OW | Defines the COUTA signal value when the window is closed.<br>**NOTE**<br>Valid only in Windowing mode and when COUTA_OWEN=1.<br>0b - COUTA is 0<br>1b - COUTA is 1 |
| 6<br>COUTA_OWEN | COUTA_OW Enable<br>Enables the COUTA signal value to be defined by COUTA_OW when the window is closed.<br>**NOTE**<br>Valid only in Windowing mode.<br>0b - COUTA holds the last sampled value.<br>1b - Enables the COUTA signal value to be defined by COUTA_OW. |
| 5<br>COUT_PEN | Comparator Output Pin Enable<br>Enables the comparator output to become an available signal option for a selected package pin.<br>0b - Not available<br>1b - Available |
| 4<br>COUT_SEL | Comparator Output Select<br>Selects which comparator output option, COUT or COUTA, to use for CMPO.<br>0b - Use COUT (filtered)<br>1b - Use COUTA (unfiltered) |
| 3<br>COUT_INV | Comparator Invert<br>Selects the polarity of the analog comparator function, affecting the value driven to the COUT output (on both the chip pin and as CSR[COUT]) when CCR0[CMP_EN] is 0.<br>**NOTE**<br>COUT_INV has no effect in Trigger mode.<br>0b - Do not invert<br>1b - Invert |
| 2<br>DMA_EN | DMA Enable<br>Enables DMA transfers triggered from the LPCMP module. After this field and the corresponding interrupt enable field becomes 1, a DMA request is asserted when CFR or CFF becomes 1.<br>0b - Disable<br>1b - Enable |
| 1 | Sampling Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| SAMPLE_EN | Enables Sampling mode.<br>0b - Disable<br>1b - Enable |
| 0<br>WINDOW_EN | Windowing Enable<br>Enables Windowing mode.<br>**NOTE**<br>Valid only when SAMPLE_EN = 0.<br>0b - Disable<br>1b - Enable |

## 58.7.6 Comparator Control Register 2 (CCR2)

### Offset

| Register | Offset |
|---|---|
| CCR2 | 10h |

### Function

Contains the configuration options for the comparator operation, such as selecting the plus and minus comparator inputs and the hysteresis levels.

**NOTE**

When an inappropriate operation selects the same signal for both the plus and minus comparator inputs, the analog comparator automatically shuts down (regardless of CMP_EN) to prevent itself from becoming a noise generator.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | INMSEL | | 0 | | INPSEL | | 0 | | MSEL | | 0 | | PSEL | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | HYSTCTR | | 0 | OFFSET | 0 | CMP_HPMD |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-28<br>INMSEL | Input Minus Select<br>Selects the minus input of the comparator.<br>**NOTE**<br>These selections connect directly to the minus input of the comparator.<br>00b - IN0: from the 8-bit DAC output<br>01b - IN1: from the analog 8-1 mux<br>10b - Reserved<br>11b - Reserved |
| 27-26<br>— | Reserved |
| 25-24<br>INPSEL | Input Plus Select<br>Selects the plus input of the comparator.<br>**NOTE**<br>These selections connect directly to the plus input of the comparator.<br>00b - IN0: from the 8-bit DAC output<br>01b - IN1: from the analog 8-1 mux<br>10b - Reserved<br>11b - Reserved |
| 23<br>— | Reserved |
| 22-20<br>MSEL | Minus Input MUX Select<br>Selects the input used for the negative mux. See the chip-specific LPCMP information for more on connections.<br>**NOTE**<br>MSEL has no effect in Trigger mode.<br>000b - Input channel 0<br>001b - Input channel 1<br>010b - Input channel 2<br>011b - Input channel 3 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 100b - Input channel 4<br>101b - Input channel 5<br>110b - Input channel 6<br>111b - Input channel 7 |
| 19<br>— | Reserved |
| 18-16<br>PSEL | Plus Input MUX Select<br>Selects the input used for the positive mux. See the chip-specific LPCMP information for more on connections.<br>**NOTE**<br>PSEL has no effect in Trigger mode.<br>000b - Input channel 0<br>001b - Input channel 1<br>010b - Input channel 2<br>011b - Input channel 3<br>100b - Input channel 4<br>101b - Input channel 5<br>110b - Input channel 6<br>111b - Input channel 7 |
| 15-6<br>— | Reserved |
| 5-4<br>HYSTCTR | Comparator Hysteresis Control<br>Selects the level of internally generated hysteresis for the comparator output.<br>**NOTE**<br>This applies to the comparator hard block.<br>00b - Level 0: Analog comparator hysteresis 0 mV.<br>01b - Level 1: Analog comparator hysteresis 10 mV.<br>10b - Level 2: Analog comparator hysteresis 20 mV.<br>11b - Level 3: Analog comparator hysteresis 30 mV. |
| 3<br>— | Reserved |
| 2 | Comparator Offset Control |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| OFFSET | Selects the level of internally generated voltage offset for the comparator output. See the chip data sheet to get the specific values for each offset level.<br>0b - Level 0: The hysteresis selected by HYSTCTR is valid for both directions (rising and falling).<br>1b - Level 1: Hysteresis does not apply when INP (input-plus) crosses INM (input-minus) in the rising direction or when INM crosses INP in the falling direction. Hysteresis still applies for INP crossing INM in the falling direction. |
| 1<br>— | Reserved |
| 0<br>CMP_HPMD | CMP High Power Mode Select<br>Selects Low or High Power(Speed) mode for the comparator.<br>0b - Low power (speed) comparison mode<br>1b - High power (speed) comparison mode |

## 58.7.7 DAC Control (DCR)

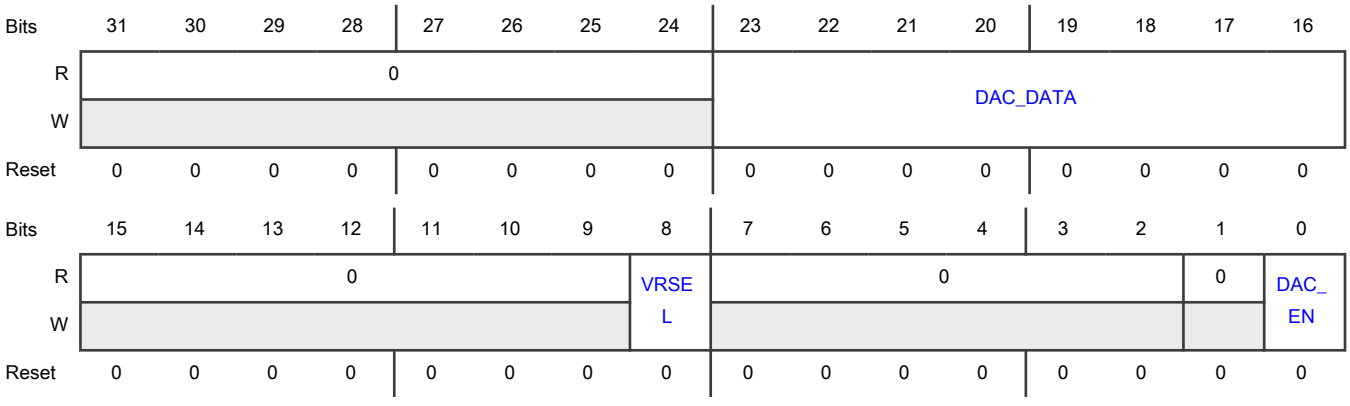
### Offset

| Register | Offset |
|---|---|
| DCR | 18h |

### Function

Contains the configuration options to enable the DAC.

### Diagram



**Fields**

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16<br>DAC_DATA | DAC Output Voltage Select<br>Selects the DAC output (DACO) voltage from one of 256 distinct levels by configuring the value of DAC_DATA. The DACO ranges from Vin/256 to Vin.<br>**NOTE**<br>DACO = (Vin/256) * (DAC_DATA + 1) |
| 15-9<br>— | Reserved |
| 8<br>VRSEL | DAC Reference High Voltage Source Select<br>Selects the high voltage reference source for the Vin supply of the DAC's resistor ladder network. See the chip-specific LPCMP information for the source of vrefh0 and vrefh1.<br>0b - VREFH0<br>1b - VREFH1 |
| 7-2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>DAC_EN | DAC Enable<br>Enables the DAC. When disabled, power-down the DAC to conserve power.<br>**NOTE**<br>You can control the link from the CMP enable to the DAC enable by setting up CCR0[LINKEN].<br>0b - Disable<br>1b - Enable |

## 58.7.8 Interrupt Enable (IER)

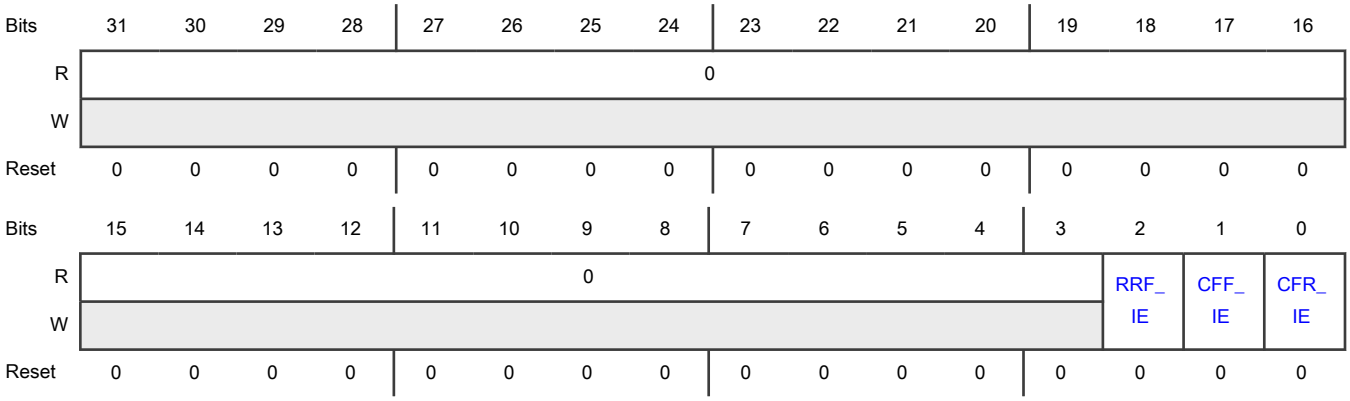
**Offset**

| Register | Offset |
|---|---|
| IER | 1Ch |

### Function

Provides enable fields for the comparator and round-robin flag interrupts.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2<br>RRF_IE | Round-Robin Flag Interrupt Enable<br>Enables or disables the round-robin flag interrupt.<br>0b - Disables the round-robin flag interrupt.<br>1b - Enables the round-robin flag interrupt when the comparison result changes for a given channel. |
| 1<br>CFF_IE | Comparator Flag Falling Interrupt Enable<br>Enables or disables the comparator flag falling interrupt.<br>0b - Disables the comparator flag falling interrupt.<br>1b - Enables the comparator flag falling interrupt when CFF is set. |
| 0<br>CFR_IE | Comparator Flag Rising Interrupt Enable<br>Enables or disables the comparator flag rising interrupt.<br>0b - Disables the comparator flag rising interrupt.<br>1b - Enables the comparator flag rising interrupt when CFR is set. |

## 58.7.9 Comparator Status (CSR)

### Offset

| Register | Offset |
|---|---|
| CSR | 20h |

### Function

Indicates comparator status, including COUT, CFF, CFR, and RRF.

**NOTE**

LPCMP may output a glitch and affect the value of CSR[CFF] and CSR[CFR] at the moment of enabling CMP. In order to ensure correctness, it is recommended to write one to clear (W1C) CSR[CFF] and CSR[CFR] before further configuring CMP.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | COUT | 0 | | | | | RRF | CFF | CFR |
| W | | | | | | | | | | | | | | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>COUT | Analog Comparator Output<br>Returns the current value of the analog comparator output when read. This field resets to 0 and reads as CCR1[COUT_INV] after the analog comparator module disables when CCR0[CMP_EN] = 0. Writing to this field is ignored. |
| 7-3<br>— | Reserved |
| 2<br>RRF | Round-Robin Flag<br>Detects when any channel's last comparison result is different from the pre-set value in Trigger mode. Write 1 to clear this field. This field clears when CCR0[CMP_EN] or RRCR0[RR_EN] is not 1.<br>0b - Not detected<br>1b - Detected |
| 1<br>CFF | Analog Comparator Flag Falling<br>Detects when a falling edge on COUT occurs. Write 1 to clear this field when CCR1[DMA_EN] is disabled. If CCR1[DMA_EN] is enabled, the flag automatically clears after DMA is done. This field clears when CCR0[CMP_EN] is not 1. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Not detected<br>1b - Detected |
| 0<br>CFR | Analog Comparator Flag Rising<br>Detects when a rising edge on COUT occurs. Write 1 to clear this field when CCR1[DMA_EN] is disabled. If CCR1[DMA_EN] is enabled, the flag automatically clears after DMA is done. This field clears when CCR0[CMP_EN] is not 1.<br>0b - Not detected<br>1b - Detected |

## 58.7.10 Round Robin Control Register 0 (RRCR0)

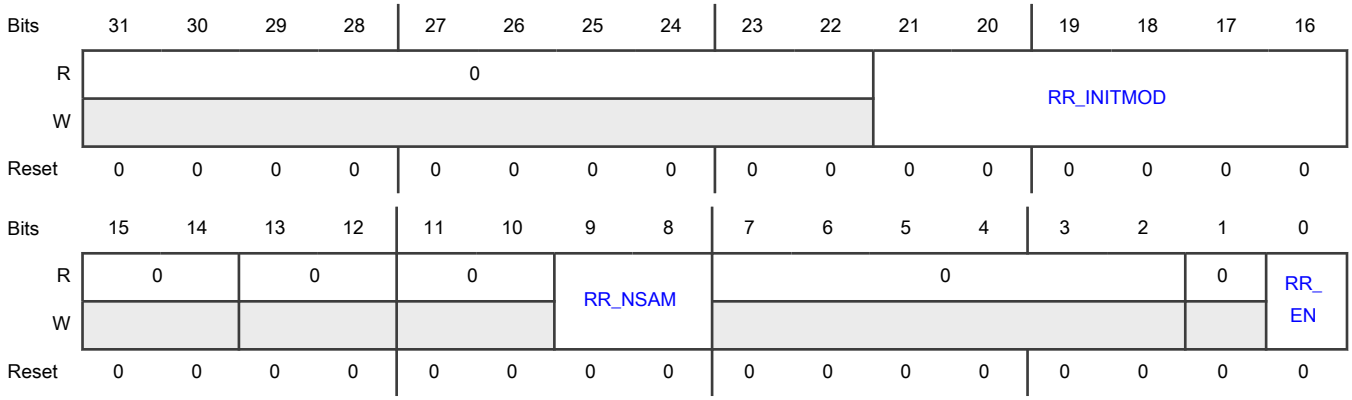
### Offset

| Register | Offset |
|---|---|
| RRCR0 | 24h |

### Function

Contains configuration options for the round-robin operation, such as enabling it and specifying the initialization delay.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-22<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 21-16<br>RR_INITMOD | Initialization Delay Modulus<br>Specifies the number of round-robin clock cycles that determines the comparator and DAC initialization delay specified in the chip datasheet. Calculate the initialization delay as RR_INITMOD * (round-robin clock period).<br>For example, if the initialization delay is 80us and the round-robin clock is 100kHz, program RR_INITMOD to be 80us/10us = 8.<br>00_0000b - 63 cycles (same as 111111b)<br>00_0001b-11_1111b - 1 to 63 cycles |
| 15-14<br>— | Reserved |
| 13-12<br>— | Reserved |
| 11-10<br>— | Reserved |
| 9-8<br>RR_NSAM | Number of Sample Clocks<br>Specifies the number of the round-robin clock cycles to wait after scanning the active channel before sampling the channel's comparison result. After the next cycle of the round-robin clock, the sampling takes place RR_NSAM clocks later.<br>00b - 0 clock<br>01b - 1 clock<br>10b - 2 clocks<br>11b - 3 clocks |
| 7-2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>RR_EN | Round-Robin Enable<br>Enables the round-robin operation.<br>0b - Disable<br>1b - Enable |

## 58.7.11 Round Robin Control Register 1 (RRCR1)

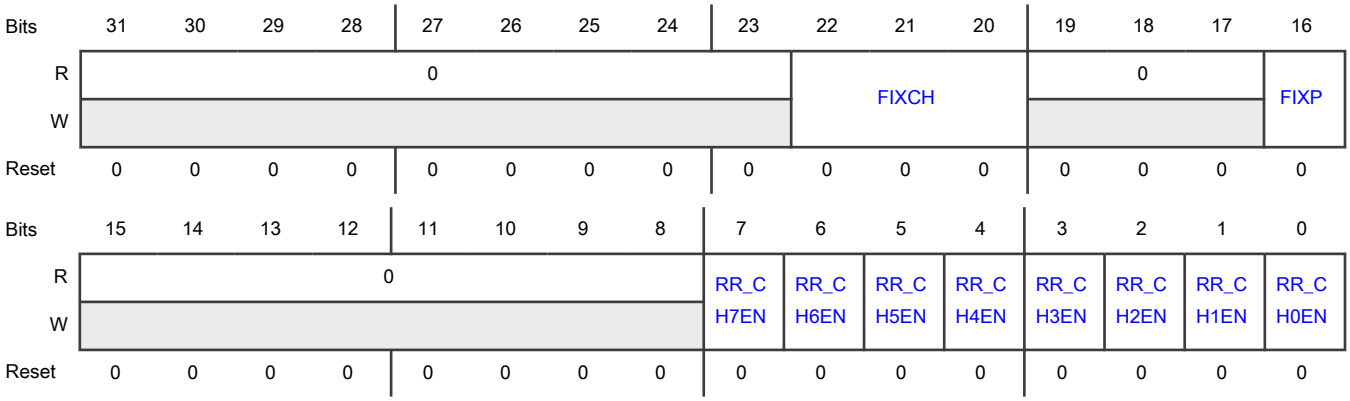
### Offset

| Register | Offset |
|---|---|
| RRCR1 | 28h |

### Function

Contains configuration options for the round-robin operation, such as enabling individual channels to participate.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-23<br>— | Reserved |
| 22-20<br>FIXCH | Fixed Channel Select<br>Selects which channel in the mux port to fix for a given round-robin trigger mode application.<br>000b - Channel 0<br>001b - Channel 1<br>010b - Channel 2<br>011b - Channel 3<br>100b - Channel 4<br>101b - Channel 5<br>110b - Channel 6<br>111b - Channel 7 |
| 19-17<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 16<br>FIXP | Fixed Port<br>Fixes an analog mux port (plus or minus) for round-robin trigger mode. The inputs to the non-fixed port sweep during each round.<br>0b - Fix the plus port. Sweep only the inputs to the minus port.<br>1b - Fix the minus port. Sweep only the inputs to the plus port. |
| 15-8<br>— | Reserved |
| 7-0<br>RR_CHnEN | Channel n Input Enable in Trigger Mode<br>Enables channel n of the non-fixed mux port to check its voltage value when in Trigger mode.<br>**NOTE**<br>RR_CHnEN has no effect when the same channel is selected as the reference voltage.<br>0b - Disable<br>1b - Enable |

## 58.7.12 Round Robin Control and Status (RRCSR)

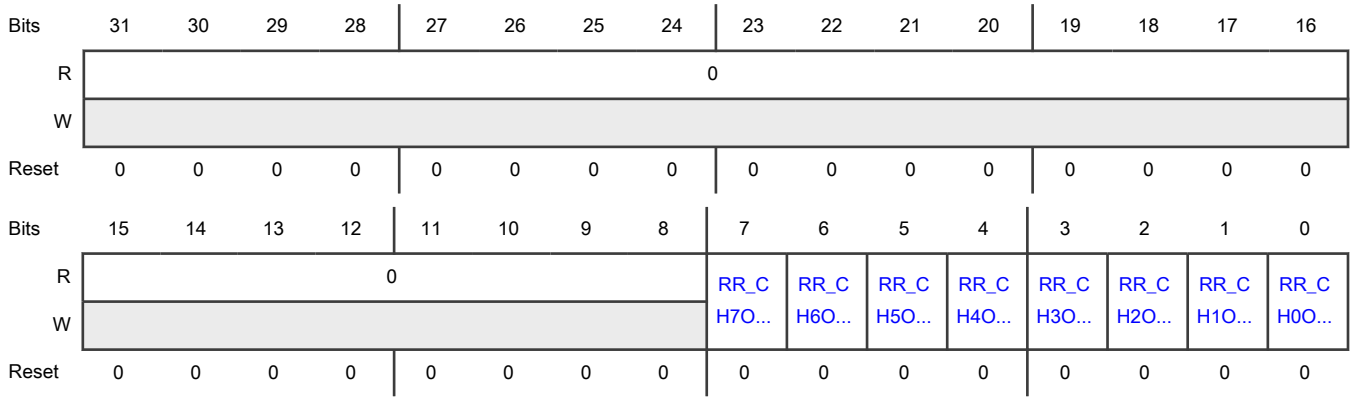
### Offset

| Register | Offset |
|---|---|
| RRCSR | 2Ch |

### Function

Contains the latest comparison results of the individual channels with the fixed mux port. It also allows you to define the pre-set state for each channel.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>RR_CHnOUT | Comparison Result for Channel n<br>Returns the latest comparison result for channel n when read and defines the pre-set state for channel n when written to. |

## 58.7.13 Round Robin Status (RRSR)

### Offset

| Register | Offset |
|---|---|
| RRSR | 30h |

### Function

Contains individual channel flags that indicates when a channel's last comparison result is different from its pre-set value.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | RR_CH7F | RR_CH6F | RR_CH5F | RR_CH4F | RR_CH3F | RR_CH2F | RR_CH1F | RR_CH0F |
| W | | | | | | | | | W1C | W1C | W1C | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>RR_CHnF | Channel n Input Changed Flag<br>Indicates when the corresponding channel's last comparison result is different from its pre-set value. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>To clear a flag, write a 1 to it.<br>0b - No different<br>1b - Different |

## 58.8 Glossary

| | |
|---|---|
| **CMP** | Comparator |
| **DAC** | Digital-to-analog convertor |
| **ANMUX** | Analog multiplexer |

# Chapter 59
# Logic Control Unit (LCU)

## 59.1 Chip-specific LCU information

### 59.1.1 LCU instances

This chip has two identical LCU instances, LCU_0 and LCU_1.

## 59.2 Overview

LCU selects multiple inputs from timers, Pulse Width Modulation (PWM) signals, and Input/Output (I/O) pads, and combines them using a programmable logic function to create output waveforms.

### 59.2.1 Block diagram

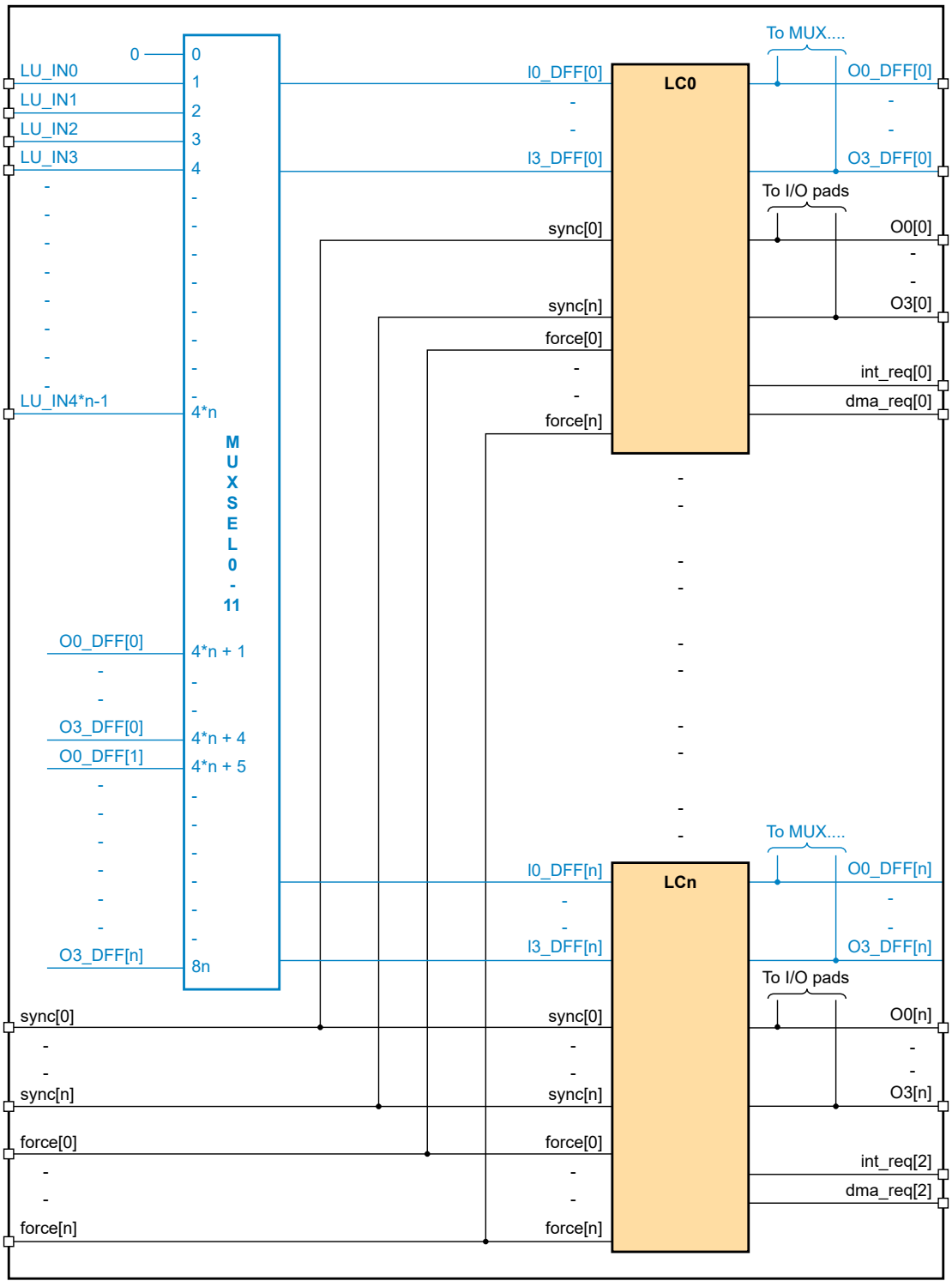



Figure 251. Block diagram

### 59.2.2 Features

- 3 LCs with programmable logic function for generating output results

- 12 inputs to and 12 outputs from LCs
- Muxing to map any input to any LC
- Independent filters for the rising and falling output states of each LC
- Software override logic for inputs using multiple modes
- 3 force inputs with programmable edge filtering to force output states using multiple modes
- 2 sync inputs to control transition timing
- Lock bit to block writes to CFG[WP]
- Interrupt request generation based on output change or force event
- DMA transfer request generation based on output change or force event

## 59.3 Functional description

### 59.3.1 LC operation

#### 59.3.1.1 Overview

An LC performs logic operations based on the lookup table principle, as described in Logic operations.

Each LC supports the following features:

- Independent filter thresholds for the rising and falling output states support different transition delays.
- Bypass of the output state filter by setting the filter threshold to zero. Bypassing the filter causes the asynchronous signal to propagate to the output of the LC.
- Force logic that instantly switches an LC output to the inactive state upon external fault signal assertion. The return from the inactive state to the output state is either:
  - Instantaneous: when LCU deasserts the force signal.
  - Instantaneous synchronized: when LCU deasserts the force signal and then the sync input asserts.
  - Manual: when you write 1 to the force status bit and then the LCU deasserts the force input.
  - Manual synchronized: when you write 1 to the force status bit and LCU deasserts the force input, and then the sync input asserts.
- Input software override logic to override external inputs by writing to the following registers:
  - LC*n* Sync Control (LC*n*_SCTRL[SW_MODE])
  - Software Override Value (SWVALUE[SWVALUE])
  - Software Override Enable (SWEN[SWEN])
- Generation of interrupt requests or DMA transfer requests.

#### 59.3.1.2 LC diagram

This figure illustrates an example LC implementation with:

- One LC
- Four force inputs
- Two sync inputs

For the number of each resource in this chip, see Features.

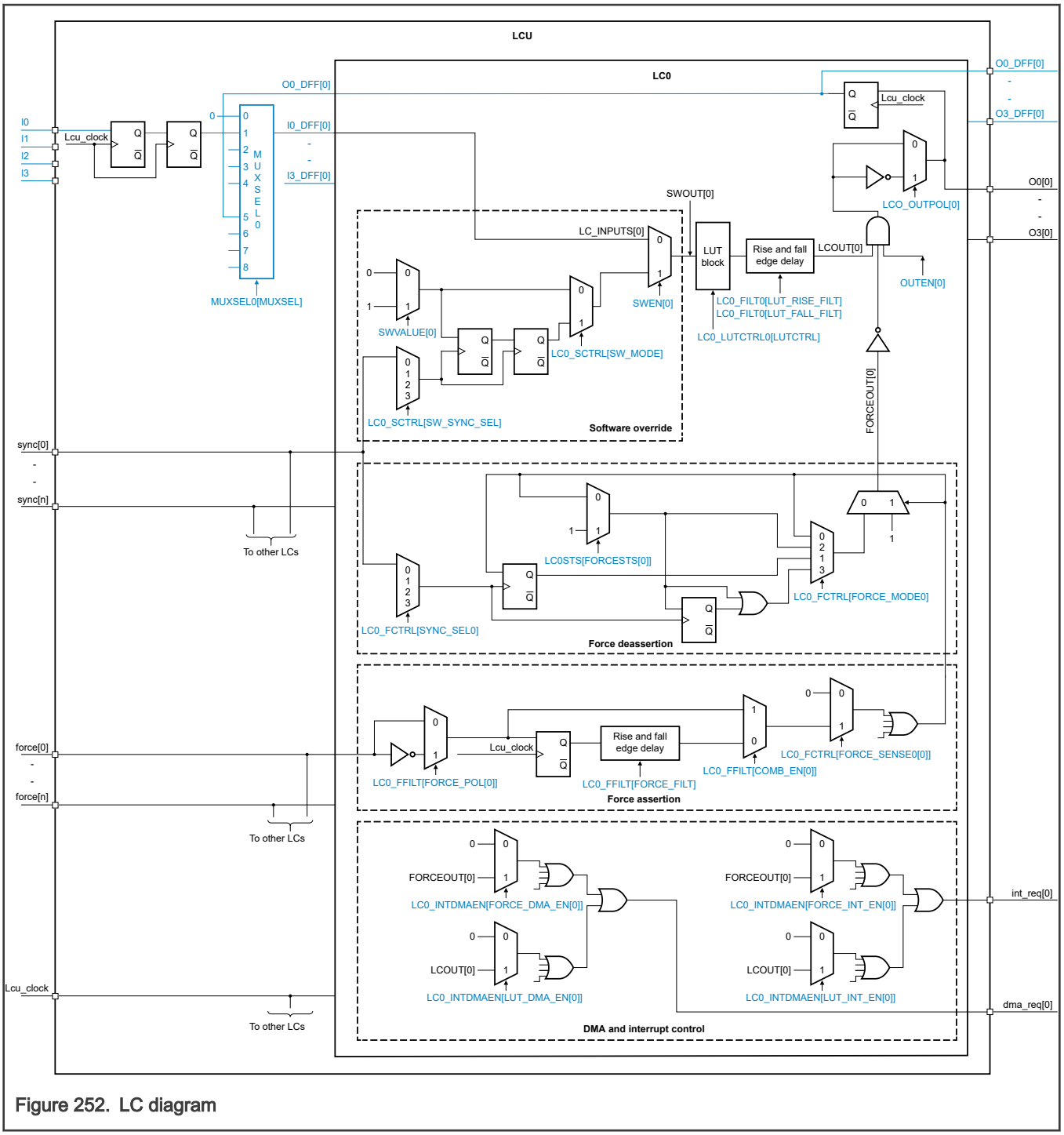



Figure 252. LC diagram

### 59.3.1.3 Logic operations

An LC reads its set of inputs (inputs 3, 2, 1, and 0) as a four-bit binary value, with Input 3 as the most-significant bit and Input 0 as the least-significant bit. For example, if:

Input 0 = 0

Input 1 = 1

Input 2 = 0

Input 3 = 1

then the combined input value is 1010b (10 decimal).

Each possible four-bit input value corresponds to a bit position in the LUT Control (LCn_LUTCTRLm[LUTCTRL]), as illustrated in LC LUT. For a given LC output, if the combined LC input value corresponds to a bit position in that output's LUTCTRL value that equals 1, then the LC asserts that output.

For example, if you write 35h to LUTCTRL (shown in the Example column of the LC LUT), then the combined input values 0 (0000b), 2 (0010b), 4 (0100b), and 5 (0101b) cause assertion of the output. All other input values cause deassertion of the output.

LCU processes each look-up table asynchronously.

#### 59.3.1.4 LC LUT

The LUT maps the combined LC input value to a bit position in LC*n*_LUTCTRL*m*.

**Table 281. LC LUT**

| Input 3 | Input 2 | Input 1 | Input 0 | LUTCTRL bit position | Example LUTCTRL value: 35h |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 2 | 1 |
| 0 | 0 | 1 | 1 | 3 | 0 |
| 0 | 1 | 0 | 0 | 4 | 1 |
| 0 | 1 | 0 | 1 | 5 | 1 |
| 0 | 1 | 1 | 0 | 6 | 0 |
| 0 | 1 | 1 | 1 | 7 | 0 |
| 1 | 0 | 0 | 0 | 8 | 0 |
| 1 | 0 | 0 | 1 | 9 | 0 |
| 1 | 0 | 1 | 0 | 10 | 0 |
| 1 | 0 | 1 | 1 | 11 | 0 |
| 1 | 1 | 0 | 0 | 12 | 0 |
| 1 | 1 | 0 | 1 | 13 | 0 |
| 1 | 1 | 1 | 0 | 14 | 0 |
| 1 | 1 | 1 | 1 | 15 | 0 |

#### 59.3.1.5 LC output filters

Each LC supports two optional digital filters, a Rise Filter (LC*n*_FLIT*m*[LUT_RISE_FILT]) and a Fall Filter (LC*n*_FLIT*m*[LUT_FALL_FILT]), to post-process the LUT output and delay the output signal change.

Each filter starts accumulating clock cycles on the occurrence of the associated LUT event. The Rise Filter counts from an output assertion, and the Fall Filter counts from an output deassertion. When the number of clock cycles reaches the filter threshold (matches the specified filter value), the output flips to the new state. For example, if you specify a Rise Filter value of 100h, and the LUT event triggers an output assertion, the output signal does not assert until the 256th clock cycle after the event.

A filter value of 0 bypasses the filter.

**NOTE**

LCU usecase (intercell/feedback loop path along with TRGMUX Path) does not work with default RISE/FALL filter time. You must configure RISE/FALL Filter delay at least for 1 cycle to avoid such glitches/failures.

### 59.3.2 Behavior in different chip modes of operation

Table 282. Modes of operation

| Operation mode | Status of outputs |
|---|---|
| Normal | • Controlled by LC*n*_LUTCTRL*m* and MUXSEL*n*<br>• Toggled based on the states of the selected inputs |
| Debug | Inactive or normal operation controlled by Debug Mode Enable (DBGEN) |

### 59.3.3 Clocking

This module has no clocking considerations.

### 59.3.4 Interrupts

Each LC has a single interrupt request output int_req[n]. Interrupt request enables bits for all the 4 outputs in each LC defined in LC*n*_INTDMAEN with LC*n*_INTDMAEN[FORCE_INT_EN] for interrupt request generation when a force event occurs and LC*n*_INTDMAEN[LUT_INT_EN] for interrupt request generation when a LUT event occurs.

When user enables interrupt for a force event on an output and the force event occurs on that output, LCU generates an interrupt request if LC*n*_STS[FORCESTS] and LC*n*_INTDMAEN[FORCE_INT_EN] for that output is 1. Similarly when user enables interrupt for a LUT event on an output and a LUT event occurs on that output, LCU generates an interrupt request if LC*n*_STS[LUT_STS] and LC*n*_INTDMAEN[LUT_INT_EN] for that output is 1.

### 59.3.5 DMA

Each LC has a single DMA request output dma_req[n]. DMA request enables bits for all the 4 outputs in each LC defined in LC*n*_INTDMAEN with LC*n*_INTDMAEN[FORCE_DMA_EN] for DMA request generation when a force event occurs and LC*n*_INTDMAEN[LUT_DMA_EN] for DMA request generation when a LUT event occurs.

When user enables DMA for a force event on an output and the force event occurs on that output, LCU generates a DMA request if LC*n*_STS[FORCESTS] and LC*n*_INTDMAEN[FORCE_DMA_EN] for that output is 1. Similarly when user enables DMA for a LUT event on an output and a LUT event occurs on that output, LCU generates a DMA request if LC*n*_STS[LUT_STS] and LC*n*_INTDMAEN[LUT_DMA_EN] for that output is 1.

## 59.4 External signals

This module has no external signals.

## 59.5 Initialization

This module does not require initialization.

## 59.6 Application information

### 59.6.1 Use-case examples

#### 59.6.1.1 Two-channel multiplexer

#### 59.6.1.1.1 Overview

A multiplexer (or mux) takes multiple inputs and, based on some control mechanism, passes only one of those inputs to the output. You can implement a two-channel multiplexer controlled by an external signal or a software override using three inputs and one output of a single LC, as illustrated in Logic diagram and Implementation.

This example implementation controls the multiplexer with the SEL input signal and a software override. The software override controls the multiplexer output regardless of the SEL signal state. To use software override, you must enable it for the corresponding output (SWEN[SWEN]). You assert software override by writing 1 to the corresponding bit of the Software Override Value (SWVALUE[SWVALUE]).

#### 59.6.1.1.2 Logic diagram

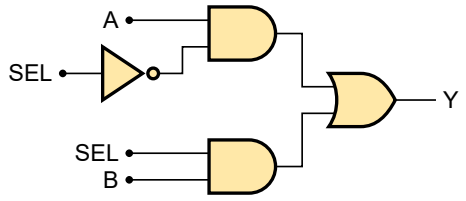



Figure 253. Logic diagram

#### 59.6.1.1.3 Implementation

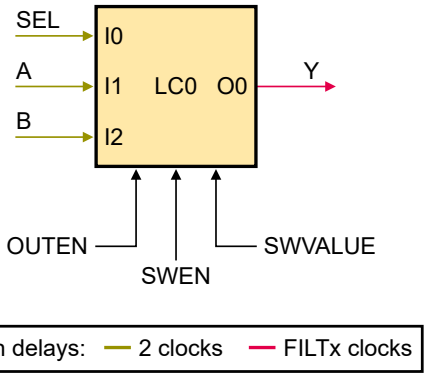



Figure 254. Implementation

#### 59.6.1.1.4 Connections and controls

Table 283. Connections and controls

| Name | Description |
|---|---|
| SEL | Input signal |
| A | |
| B | |
| OUTEN | Output enable |
| SWEN | Software override enable |
| SWVALUE | Software override |
| Y | Output signal |

#### 59.6.1.1.5 Truth table

**Table 284. Truth table**

| Inputs | | | Output |
|---|---|---|---|
| A | B | SEL | Y |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

#### 59.6.1.1.6 Register configuration

**Table 285. Register configuration**

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| LUTCTRL0 | — | — | — | — | E4E4h | — | — | — |
| LUTCTRL1 | — | — | — | — | — | 0h | — | — |
| LUTCTRL2 | — | — | — | — | — | — | 0h | — |
| LUTCTRL3 | — | — | — | — | — | — | — | 0h |
| FILT0 | — | — | — | — | 0h | — | — | — |
| FILT1 | — | — | — | — | — | 0h | — | — |
| FILT2 | — | — | — | — | — | — | 0h | — |
| FILT3 | — | — | — | — | — | — | — | 0h |
| INTDMAEN | — | — | — | — | 0h | | | |
| OUTPOL | — | — | — | — | 0h | | | |
| FFILT | — | — | — | — | 0h | | | |
| FCTRL | — | — | — | — | 0h | | | |
| SCTRL | — | — | — | — | 0h | | | |
| MUXSEL0 | — | — | — | 1h | — | — | — | — |
| MUXSEL1 | — | — | 2h | — | — | — | — | — |
| MUXSEL2 | — | 3h | — | — | — | — | — | — |
| MUXSEL3 | 0h | — | — | — | — | — | — | — |
| SWEN | — | — | — | — | Disable software override for LC0 input 0: 0h | | | |

*Table continues on the next page...*

Table 285. Register configuration (continued)

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| | | | | | Enable software override for LC0 input 0: 1h | | | |
| SWVALUE | — | — | — | — | Deassert software override for LC0 input 0: 0h<br>Assert software override for LC0 input 0: 1h | | | |
| OUTEN | — | — | — | — | 1h | | | |

#### 59.6.1.1.7 Waveforms

In this figure, the SWEN and SWVALUE signals represent the states of the software override control registers. Writes to SWEN and SWVALUE have immediate impact on LC outputs, whereas LC inputs require input synchronization to prevent metastability.

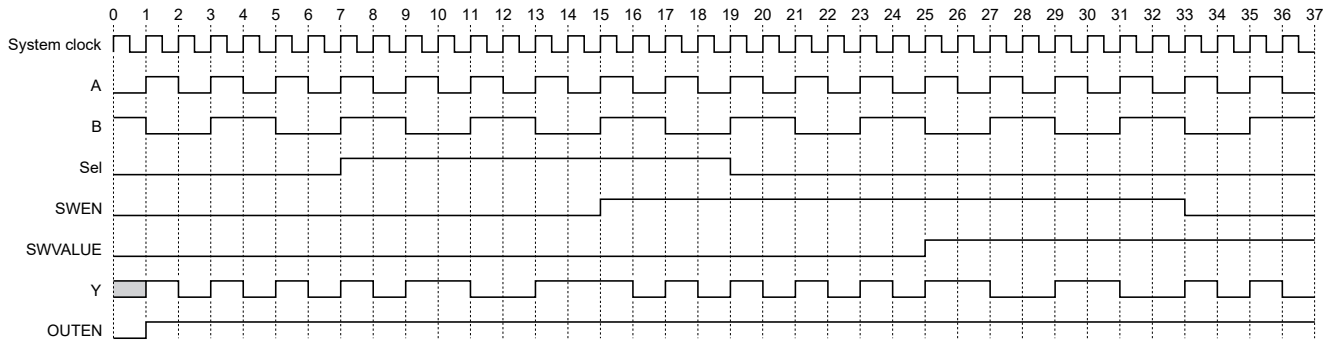



Figure 255. Waveforms

### 59.6.1.2 Binary to Gray code converter

#### 59.6.1.2.1 Overview

Gray code, also known as reflected binary code (RBC), is a non-weighted, minimum change code. In this code, any two adjacent code numbers differ by only one bit. This code solves the problem of physical switch transitions by changing only one switch at a time, so there is never any ambiguity of position.

You can implement binary to Gray code conversion using a single LC, as illustrated in Logic diagram and Implementation.

In this example, the LC also generates a DMA request on each rising edge of output 3, which is bit 0 of the Gray code (G0).

#### 59.6.1.2.2 Conversion table

Table 286. Conversion table

| Decimal | Binary | Gray |
|---|---|---|
| 0 | 0000 | 0000 |
| 1 | 0001 | 0001 |
| 2 | 0010 | 0011 |
| 3 | 0011 | 0010 |
| 4 | 0100 | 0110 |

*Table continues on the next page...*

Table 286. Conversion table (continued)

| Decimal | Binary | Gray |
|---|---|---|
| 5 | 0101 | 0111 |
| 6 | 0110 | 0101 |
| 7 | 0111 | 0100 |
| 8 | 1000 | 1100 |
| 9 | 1001 | 1101 |
| 10 | 1010 | 1111 |
| 11 | 1011 | 1110 |
| 12 | 1100 | 1010 |
| 13 | 1101 | 1011 |
| 14 | 1110 | 1001 |
| 15 | 1111 | 1000 |

#### 59.6.1.2.3 Logic diagram

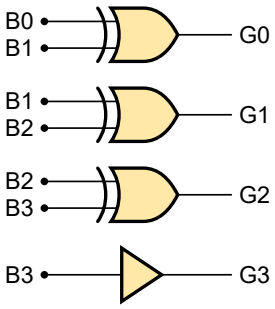



Figure 256. Logic diagram

#### 59.6.1.2.4 Implementation

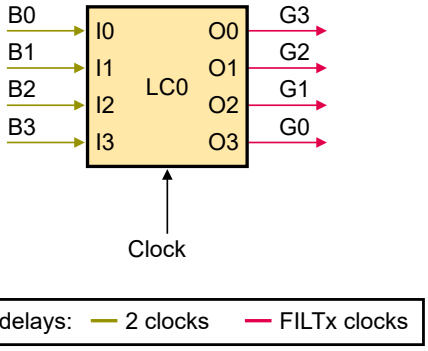



Figure 257. Implementation

#### 59.6.1.2.5 Connections and controls

Table 287. Connections and controls

| Name | Description |
|---|---|
| B0 | Bit 0 of the input binary code |
| B1 | Bit 1 of the input binary code |
| B2 | Bit 2 of the input binary code |
| B3 | Bit 3 of the input binary code |
| G0 | Bit 0 of the output Gray code |
| G1 | Bit 1 of the output Gray code |
| G2 | Bit 2 of the output Gray code |
| G3 | Bit 3 of the output Gray code |
| Clock | System clock input |

#### 59.6.1.2.6 Truth table

Table 288. Truth table

| Inputs | | | | Outputs | | | |
|---|---|---|---|---|---|---|---|
| B3 | B2 | B1 | B0 | G3 | G2 | G1 | G0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

#### 59.6.1.2.7 Register configuration

Table 289. Register configuration

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| LUTCTRL0 | — | — | — | — | FF00h | — | — | — |
| LUTCTRL1 | — | — | — | — | — | FF0h | — | — |
| LUTCTRL2 | — | — | — | — | — | — | 3C3Ch | — |
| LUTCTRL3 | — | — | — | — | — | — | — | 6666h |
| FILT0 | — | — | — | — | 0h | — | — | — |
| FILT1 | — | — | — | — | — | 0h | — | — |
| FILT2 | — | — | — | — | — | — | 0h | — |
| FILT3 | — | — | — | — | — | — | — | 0h |
| INTDMAEN | — | — | — | — | 800h | | | |
| OUTPOL | — | — | — | — | 0h | | | |
| FFILT | — | — | — | — | 0h | | | |
| FCTRL | — | — | — | — | 0h | | | |
| SCTRL | — | — | — | — | 0h | | | |
| MUXSEL0h | — | — | — | 01h | — | — | — | — |
| MUXSEL1 | — | — | 2h | — | — | — | — | — |
| MUXSEL2 | — | 3h | — | — | — | — | — | — |
| MUXSEL3 | 4h | — | — | — | — | — | — | — |
| SWEN | — | — | — | — | 0h | | | |
| SWVALUE | — | — | — | — | 0h | | | |
| OUTEN | — | — | — | — | Fh | | | |

#### 59.6.1.2.8 Waveforms

The LC outputs are delayed by two clock cycles because of the two-stage input synchronizers.

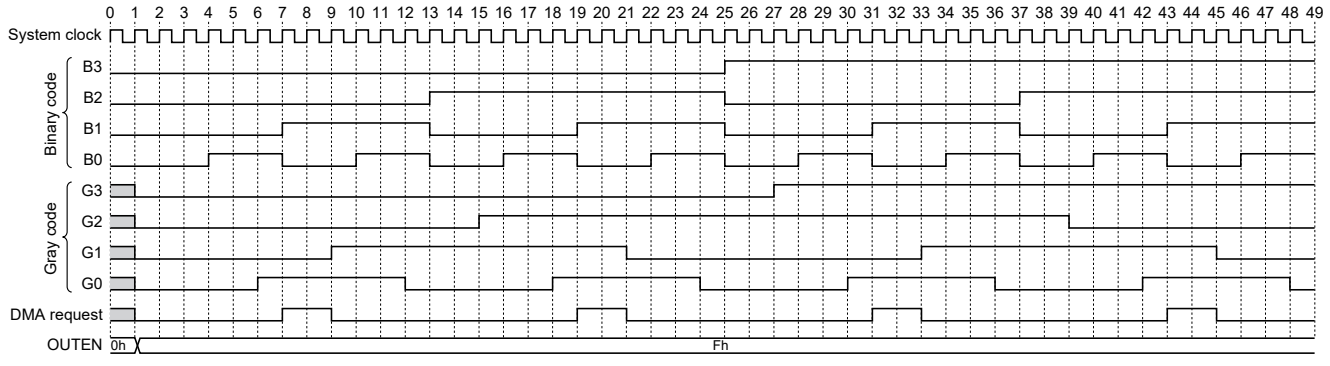



Figure 258. Waveforms

### 59.6.1.3 Manchester encoder and decoder

#### 59.6.1.3.1 Overview

Manchester encoding (also known as phase encoding) is a data-modulation technique for binary data transfer based one of these signals:

- Analog
- RF
- Optical
- High-speed-digital
- Long-distance-digital

The fundamental idea behind Manchester encoding is to use voltage transitions, instead of voltage levels, to represent 1s and 0s. It performs an exclusive OR of data and clock signals to encode and decode data. A high-to-low transition on the falling clock edge indicates 0. A low-to-high transition on the rising clock edge indicates 1.

#### 59.6.1.3.2 Logic diagram

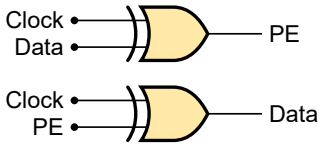



Figure 259. Logic diagram

#### 59.6.1.3.3 Implementation

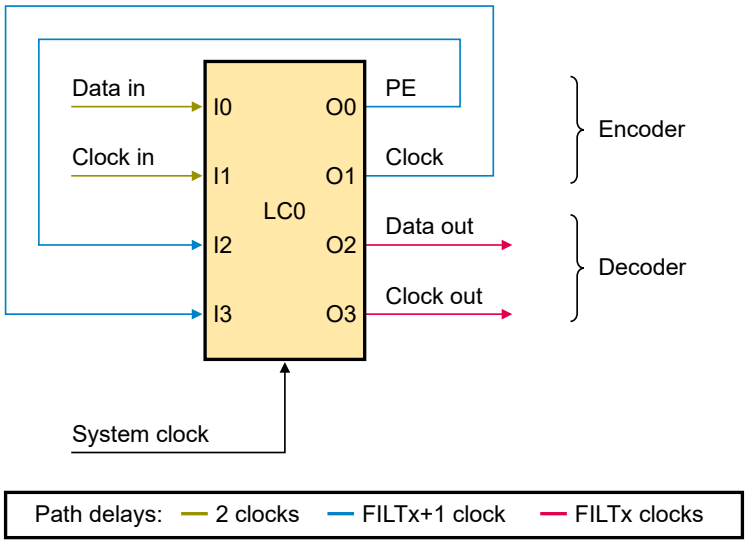



Figure 260. Implementation

#### 59.6.1.3.4 Connections and controls

Table 290. Connections and controls

| Name | Description |
|---|---|
| Data in | Input signal |
| Clock in | |
| PE | Manchester-encoded signal from LC |
| Clock | Auxiliary signal fed to the decoder to allow Manchester decoding |
| Data out | Output signal after Manchester decoding |
| System clock | Clock frequency supplied to LC |
| Clock out | Output signal |

#### 59.6.1.3.5 Truth table for encoder

Table 291. Truth table for encoder

| Inputs | | Output |
|---|---|---|
| Data in | Clock in | PE |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

#### 59.6.1.3.6 Truth table for decoder

Table 292. Truth table for decoder

| Inputs | | Output |
|---|---|---|
| Clock | PE | Data out |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

#### 59.6.1.3.7 Register configuration

Table 293. Register configuration

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| LUTCTRL0 | — | — | — | — | 6666h | — | — | — |
| LUTCTRL1 | — | — | — | — | — | AAAAh | — | — |
| LUTCTRL2 | — | — | — | — | — | — | FF0h | — |

*Table continues on the next page...*

Table 293. Register configuration (continued)

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| LUTCTRL3 | — | — | — | — | — | — | — | FF00h |
| FILT0 | — | — | — | — | 0h | — | — | — |
| FILT1 | — | — | — | — | — | 0h | — | — |
| FILT2 | — | — | — | — | — | — | 0h | — |
| FILT3 | — | — | — | — | — | — | — | 0h |
| INTDMAEN | — | — | — | — | 0h | | | |
| OUTPOL | — | — | — | — | 0h | | | |
| FFILT | — | — | — | — | 0h | | | |
| FCTRL | — | — | — | — | 0h | | | |
| SCTRL | — | — | — | — | 0h | | | |
| MUXSEL0 | — | — | — | 1h | — | — | — | — |
| MUXSEL1 | — | — | 2h | — | — | — | — | — |
| MUXSEL2 | — | 5h | — | — | — | — | — | — |
| MUXSEL3 | 6h | — | — | — | — | — | — | — |
| SWEN | — | — | — | — | 0h | | | |
| SWVALUE | — | — | — | — | 0h | | | |
| OUTEN | — | — | — | — | Fh | | | |

#### 59.6.1.3.8 Waveforms for encoder

This figure includes the Manchester-encoded waveform (labeled PE) of the input waveform (labeled "Data in"). The PE waveform is delayed by two clock cycles because of two-stage input synchronization.

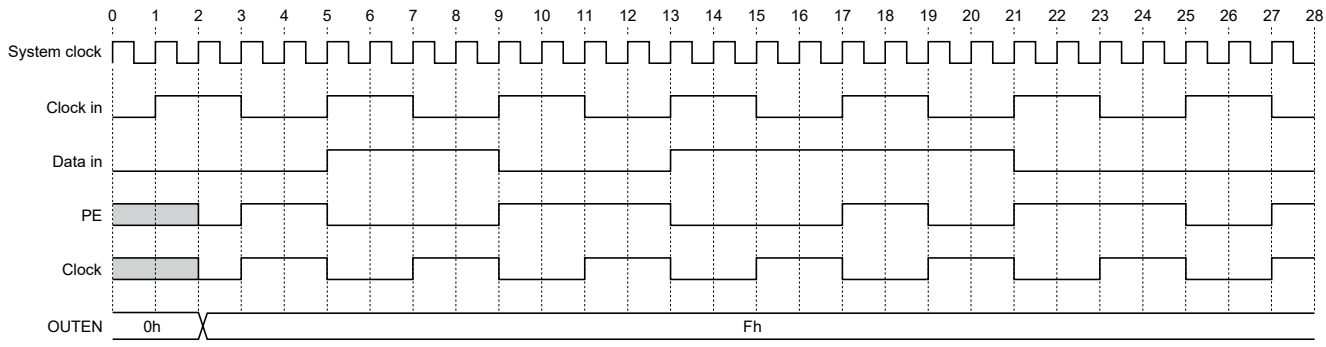



Figure 261. Waveforms for encoder

#### 59.6.1.3.9 Waveforms for decoder

The output waveform (labeled "Data out") is delayed by four clock cycles from the input waveform (labeled "Data in") because of two-stage input synchronization. In this example, the Manchester encoder also propagates the Clock waveform that the decoder uses to reconstruct the "Data in" waveform from the Manchester-decoded waveform (labeled PE).

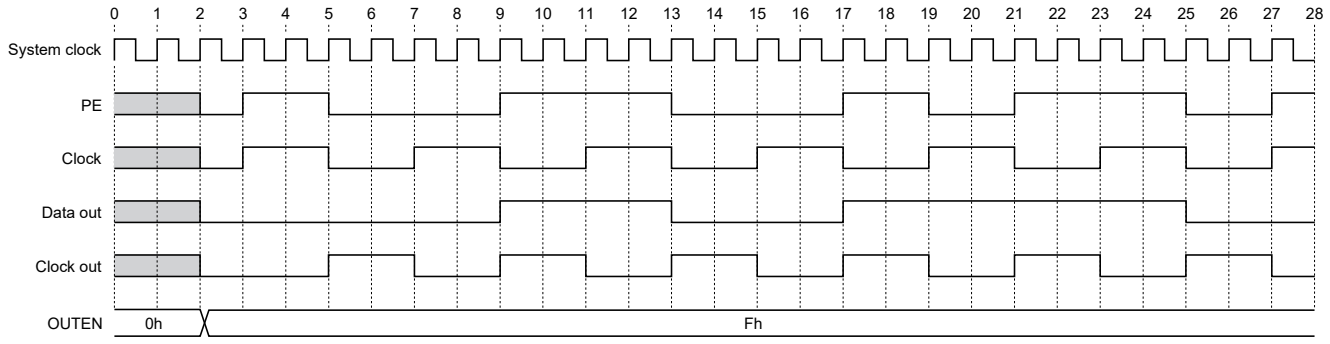



Figure 262. Waveforms for decoder

### 59.6.1.4 Adjustable PWM/FM generator

#### 59.6.1.4.1 Overview

In many cases, it is useful to route an LC output back to one of its inputs, as illustrated in Implementation. In addition, you can delay outputs with a digital filter with different time constants for delaying rising and falling edges. Combining these capabilities, you can use an LC to generate output waveforms with adjustable pulse width and frequency.

In addition to waveform generation, the software-override feature forces the generator output low or high by software. You control the override via Software Override Enable (SWEN) and Software Override Value (SWVALUE).

#### 59.6.1.4.2 Implementation

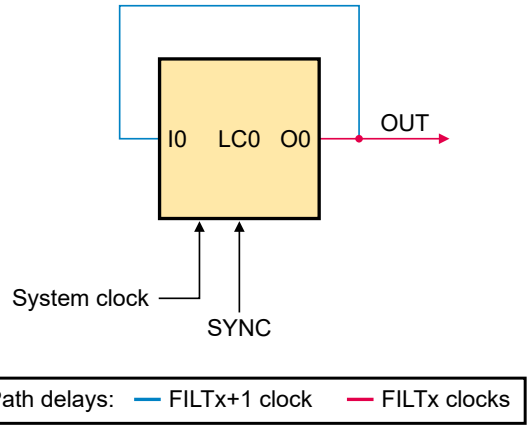



Figure 263. Implementation

#### 59.6.1.4.3 Connections and controls

Table 294. Connections and controls

| Name | Description |
|---|---|
| I0 | Delayed output routed back to LC input |
| O0 and OUT | Delayed output |
| System clock | Clock frequency supplied to LC |
| SYNC | Software override synchronization signal |

#### 59.6.1.4.4 Truth table

This example generates an adjustable PWM/FM waveform according to the following table. In the table:

- OUT is the LC output signal that is also routed to the LC input using an internal multiplexer.
- TdH is the digital filter value to delay the rising edge of the output signal.
- TdL is the digital filter value to delay the falling edge of the output signal.

**Table 295. Truth table**

| I0 | OUT |
|---|---|
| 0 | 1 delayed by TdH (see Output edge delays) |
| 1 | 0 delayed by TdL (see Output edge delays) |

#### 59.6.1.4.5 Output edge delays

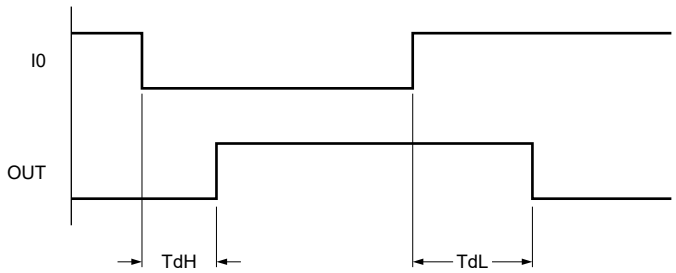



**Figure 264. Output edge delays**

#### 59.6.1.4.6 Register configuration

**Table 296. Register configuration**

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| LUTCTRL0 | — | — | — | — | 1h | — | — | — |
| LUTCTRL1 | — | — | — | — | — | 0h | — | — |
| LUTCTRL2 | — | — | — | — | — | — | 0h | — |
| LUTCTRL3 | — | — | — | — | — | — | — | 0h |
| FILT0 | — | — | — | — | 1_0001h | — | — | — |
| FILT1 | — | — | — | — | — | 0h | — | — |
| FILT2 | — | — | — | — | — | — | 0h | — |
| FILT3 | — | — | — | — | — | — | — | 0h |
| INTDMAEN | — | — | — | — | 0h | | | |
| OUTPOL | — | — | — | — | 0h | | | |
| FFILT | — | — | — | — | 0h | | | |
| FCTRL | — | — | — | — | 0h | | | |

*Table continues on the next page...*

Table 296. Register configuration (continued)

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| SCTRL | — | — | — | — | SWVALUE for input 0 changes immediately: 0h<br>SWVALUE for input 0 changes on rising edge of sync: 1h | | | |
| MUXSEL0 | — | — | — | 5h | — | — | — | — |
| MUXSEL1 | — | — | 0h | — | — | — | — | — |
| MUXSEL2 | — | 0h | — | — | — | — | — | — |
| MUXSEL3 | 0h | — | — | — | — | — | — | — |
| SWEN | — | — | — | — | Disable software override: 0h<br>Enable software override: 1h | | | |
| SWVALUE | — | — | — | — | Deassert software override: 0h<br>Assert software override: 1h | | | |
| OUTEN | — | — | — | — | 1h | | | |

#### 59.6.1.4.7 Waveforms for immediate software override

In this figure, the SWEN and SWVALUE signals represent the states of the software override control registers. Writes to Software Override Enable (SWEN) and Software Override Value (SWVALUE) have immediate impact on LC outputs, whereas LC inputs require input synchronization to prevent metastability.

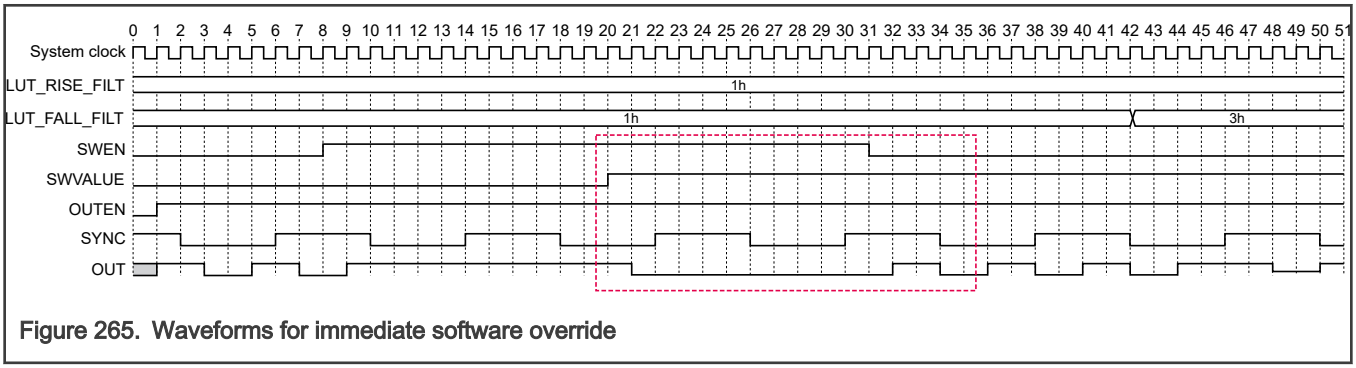



Figure 265. Waveforms for immediate software override

#### 59.6.1.4.8 Waveforms for synced software override

In this example, software override control is synced with the rising edges of the SYNC waveform.

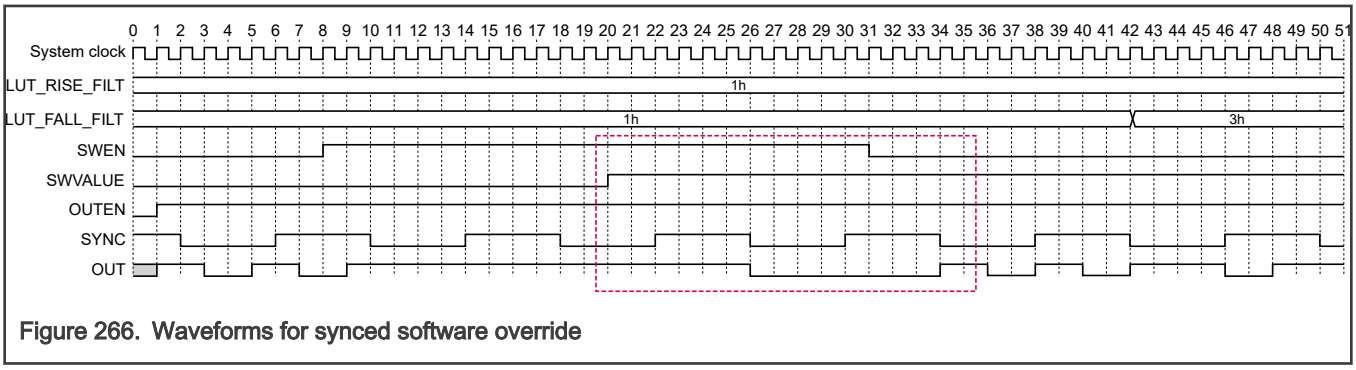



Figure 266. Waveforms for synced software override

### 59.6.1.5 SR latch

#### 59.6.1.5.1 Overview

An SR latch is the simplest bistable device and a fundamental component of data storage.

This example implementation uses one LC that includes an internal feedback path configured by the internal multiplexer, as illustrated in Implementation. Output Q responds to inputs S (set) and R (reset). It changes with a delay of two clock cycles—the time required for the S and R signal inputs to pass through the internal two-stage synchronizers. This example implementation bypasses the Q digital filter; therefore, the loop of Q back to the input experiences a delay of one clock cycle." See Waveforms.

#### 59.6.1.5.2 Implementation

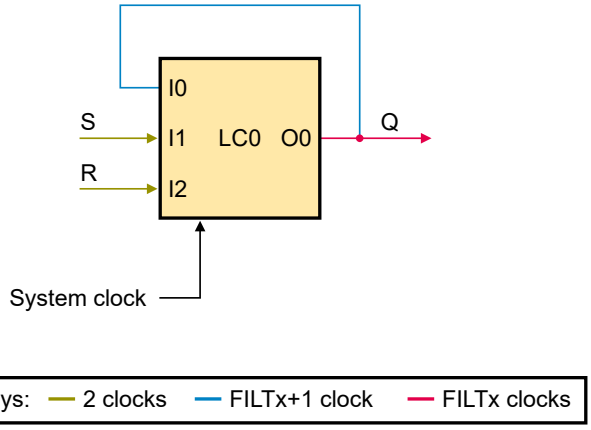



Figure 267. Implementation

#### 59.6.1.5.3 Connections and controls

Table 297. Connections and controls

| Name | Description |
|---|---|
| S | Set input |
| R | Reset input |
| Q | Both an output signal and an input that feeds back to LC0 |
| System clock | Clock frequency supplied to LC |

#### 59.6.1.5.4 Truth table

Table 298. Truth table

| Inputs | | | Output | State |
|---|---|---|---|---|
| R | S | Q | Q | |
| 0 | 0 | 0 | 0 | Latch |
| 0 | 0 | 1 | 1 | Latch |
| 0 | 1 | 0 | 1 | Set |

*Table continues on the next page...*

Table 298. Truth table (continued)

| Inputs | | | Output | State |
|---|---|---|---|---|
| R | S | Q | Q | |
| 0 | 1 | 1 | 1 | Set |
| 1 | 0 | 0 | 0 | Reset |
| 1 | 0 | 1 | 0 | Reset |
| 1 | 1 | 0 | 0 | Not allowed |
| 1 | 1 | 1 | 0 | |

#### 59.6.1.5.5 Register configuration

Table 299. Register configuration

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| LUTCTRL0 | — | — | — | — | Eh | — | — | — |
| LUTCTRL1 | — | — | — | — | — | 0h | — | — |
| LUTCTRL2 | — | — | — | — | — | — | 0h | — |
| LUTCTRL3 | — | — | — | — | — | — | — | 0h |
| FILT0 | — | — | — | — | 0h | — | — | — |
| FILT1 | — | — | — | — | — | 0h | — | — |
| FILT2 | — | — | — | — | — | — | 0h | — |
| FILT3 | — | — | — | — | — | — | — | 0h |
| INTDMAEN | — | — | — | — | 0h | | | |
| OUTPOL | — | — | — | — | 0h | | | |
| FFILT | — | — | — | — | 0h | | | |
| FCTRL | — | — | — | — | 0h | | | |
| SCTRL | — | — | — | — | 0h | | | |
| MUXSEL0 | — | — | — | 5h | — | — | — | — |
| MUXSEL1 | — | — | 2h | — | — | — | — | — |
| MUXSEL2 | — | 3h | — | — | — | — | — | — |
| MUXSEL3 | 0h | — | — | — | — | — | — | — |
| SWEN | — | — | — | — | 0h | | | |
| SWVALUE | — | — | — | — | 0h | | | |
| OUTEN | — | — | — | — | 1h | | | |

#### 59.6.1.5.6 Waveforms

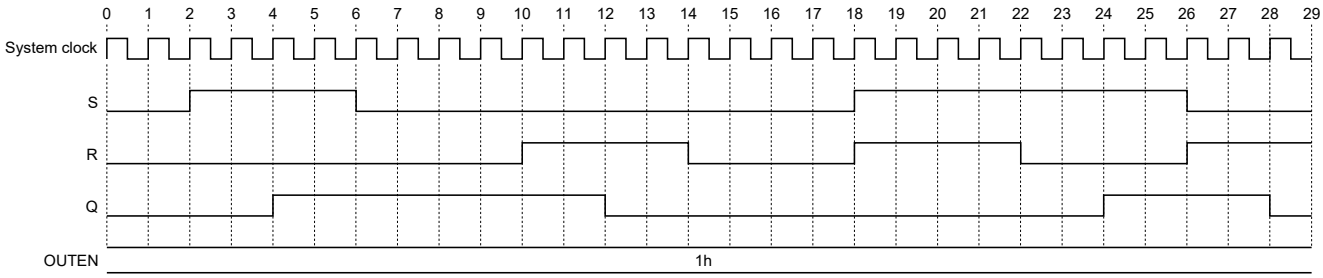



Figure 268. Waveforms

### 59.6.1.6 D flip-flop

#### 59.6.1.6.1 Overview

A D flip-flop stores and outputs whatever logic level is applied to its data terminal at the time its clock input goes high.

You can implement a D flip-flop element using an LC as illustrated in Implementation. On a clock rising edge, the LC either asserts or deasserts the Q output based on the D input.

The Q output and CLKO digital filters are bypassed. Therefore, the states of these variables are fed to inputs with a delay of one clock cycle. See Waveforms.

#### 59.6.1.6.2 Implementation

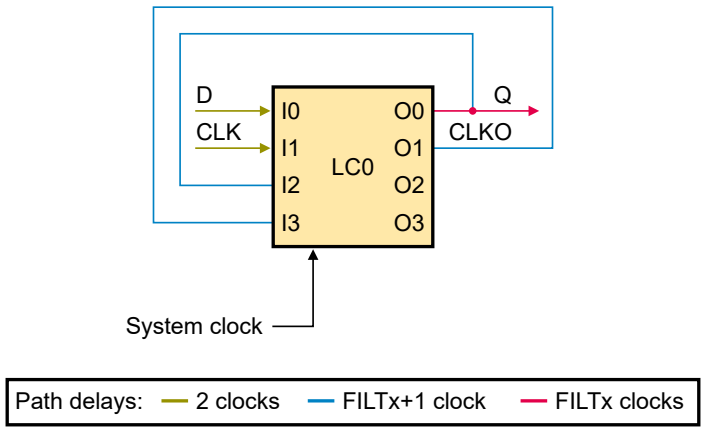



Figure 269. Implementation

#### 59.6.1.6.3 Connections and controls

Table 300. Connections and controls

| Name | Description |
|---|---|
| D | Input signal |
| CLK | Input clock |
| Q | Output signal; holds the logic state of the last Q output |
| CLKO | Holds the logic state of the last CLK input |

#### 59.6.1.6.4 Truth table

**Table 301. Truth table**

| Inputs | | | | Outputs | | State |
|---|---|---|---|---|---|---|
| CLKO | Q | CLK | D | Q | CLKO | |
| 0 | 0 | 0 | 0 | 0 | 0 | Hold |
| 0 | 0 | 0 | 1 | 0 | 0 | Hold |
| 0 | 0 | 1 | 0 | 0 | 1 | Reset |
| 0 | 0 | 1 | 1 | 1 | 1 | Set |
| 0 | 1 | 0 | 0 | 1 | 0 | Hold |
| 0 | 1 | 0 | 1 | 1 | 0 | Hold |
| 0 | 1 | 1 | 0 | 0 | 1 | Reset |
| 0 | 1 | 1 | 1 | 1 | 1 | Set |
| 1 | 0 | 0 | 0 | 0 | 0 | Hold |
| 1 | 0 | 0 | 1 | 0 | 0 | Hold |
| 1 | 0 | 1 | 0 | 0 | 1 | Hold |
| 1 | 0 | 1 | 1 | 0 | 1 | Hold |
| 1 | 1 | 0 | 0 | 1 | 0 | Hold |
| 1 | 1 | 0 | 1 | 1 | 0 | Hold |
| 1 | 1 | 1 | 0 | 1 | 1 | Hold |
| 1 | 1 | 1 | 1 | 1 | 1 | Hold |

#### 59.6.1.6.5 Register configuration

**Table 302. Register configuration**

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| LUTCTRL0 | — | — | — | — | F0B8h | — | — | — |
| LUTCTRL1 | — | — | — | — | — | CCCCh | — | — |
| LUTCTRL2 | — | — | — | — | — | — | 0h | — |
| LUTCTRL3 | — | — | — | — | — | — | — | 0h |
| FILT0 | — | — | — | — | 0h | — | — | — |
| FILT1 | — | — | — | — | — | 0h | — | — |
| FILT2 | — | — | — | — | — | — | 0h | — |
| FILT3 | — | — | — | — | — | — | — | 0h |
| INTDMAEN | — | — | — | — | 0h | | | |
| OUTPOL | — | — | — | — | 0h | | | |
| FFILT | — | — | — | — | 0h | | | |

*Table continues on the next page...*

Table 302. Register configuration (continued)

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| FCTRL | — | — | — | — | 0h | | | |
| SCTRL | — | — | — | — | 0h | | | |
| MUXSEL0 | — | — | — | 1h | — | — | — | — |
| MUXSEL1 | — | — | 2h | — | — | — | — | — |
| MUXSEL2 | — | 5h | — | — | — | — | — | — |
| MUXSEL3 | 6h | — | — | — | — | — | — | — |
| SWEN | — | — | — | — | 0h | | | |
| SWVALUE | — | — | — | — | 0h | | | |
| OUTEN | — | — | — | — | 3h | | | |

#### 59.6.1.6.6 Waveforms

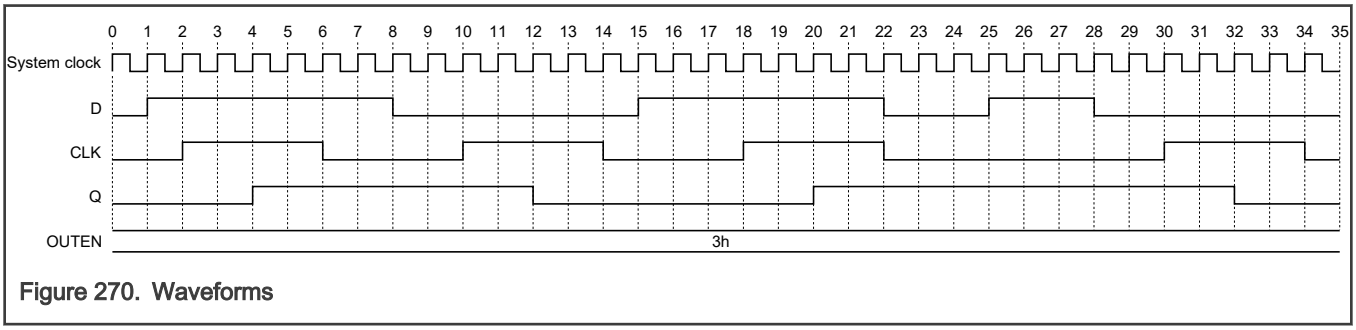



Figure 270. Waveforms

### 59.6.1.7 JK flip-flop

#### 59.6.1.7.1 Overview

A JK flip-flop is a gated SR latch (see SR latch) with an added clock input circuit that prevents the illegal or invalid output condition that can occur when both SR-latch inputs (S and R) are asserted.

Truth table for JK flip-flop presents the four possible input combinations for a JK flip-flop. All output changes occur at the clock rising edge.

You can create a JK flip-flop using two LCs, as illustrated in Implementation. The first LC detects rising edges of the CLK clock input (see Truth table for LC0). The second LC processes J and K inputs at every rising clock edge (see Truth table for LC1).

Output Q and CLKO digital filters are bypassed. Therefore, the states of these variables are fed to inputs with a delay of one clock cycle (see Waveforms).

#### 59.6.1.7.2 Implementation

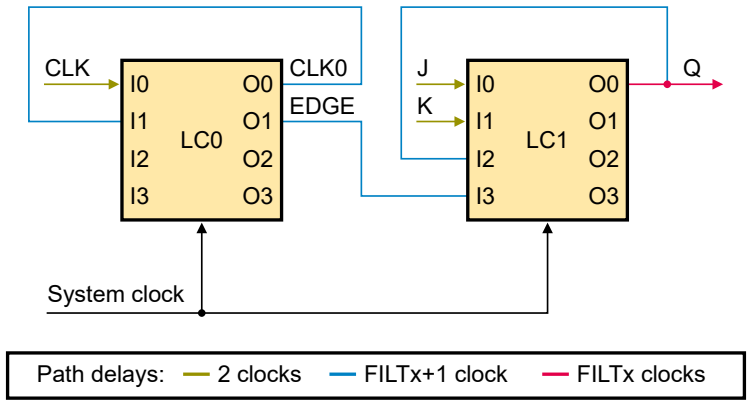



Figure 271. Implementation

#### 59.6.1.7.3 Truth table for JK flip-flop

Table 303. Truth table for JK flip-flop

| J | K | State |
|---|---|---|
| 0 | 0 | Hold |
| 1 | 0 | Set |
| 0 | 1 | Clear |
| 1 | 1 | Toggle |

#### 59.6.1.7.4 Truth table for LC0

Table 304. Truth table for LC0

| Inputs | | Outputs | | State |
|---|---|---|---|---|
| CLKO | CLK | CLKO | EDGE | |
| 0 | 0 | 0 | 0 | No edge |
| 0 | 1 | 1 | 1 | Edge detected |
| 1 | 0 | 0 | 0 | No edge |
| 1 | 1 | 1 | 0 | No edge |

#### 59.6.1.7.5 Truth table for LC1

Table 305. Truth table for LC1

| Inputs | | | | Output | State |
|---|---|---|---|---|---|
| EDGE | Q | J | K | Q | |
| 0 | 0 | X | X | 0 | Hold |
| 0 | 1 | X | X | 1 | Hold |

*Table continues on the next page...*

Table 305. Truth table for LC1 (continued)

| Inputs | | | | Output | State |
|---|---|---|---|---|---|
| EDGE | Q | J | K | Q | |
| 1 | 0 | 0 | 0 | 0 | Hold |
| 1 | 1 | 0 | 0 | 1 | Hold |
| 1 | X | 1 | 0 | 1 | Set |
| 1 | X | 0 | 1 | 0 | Clear |
| 1 | 0 | 1 | 1 | 1 | Toggle |
| 1 | 1 | 1 | 1 | 0 | Toggle |

#### 59.6.1.7.6 Register configuration for LC0

Table 306. Register configuration for LC0

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| LUTCTRL0 | — | — | — | — | Ah | — | — | — |
| LUTCTRL1 | — | — | — | — | — | 2h | — | — |
| LUTCTRL2 | — | — | — | — | — | — | 0h | — |
| LUTCTRL3 | — | — | — | — | — | — | — | 0h |
| FILT0 | — | — | — | — | 0h | — | — | — |
| FILT1 | — | — | — | — | — | 0h | — | — |
| FILT2 | — | — | — | — | — | — | 0h | — |
| FILT3 | — | — | — | — | — | — | — | 0h |
| INTDMAEN | — | — | — | — | 0h | | | |
| OUTPOL | — | — | — | — | 0h | | | |
| FFILT | — | — | — | — | 0h | | | |
| FCTRL | — | — | — | — | 0h | | | |
| SCTRL | — | — | — | — | 0h | | | |
| MUXSEL0 | — | — | — | 1h | — | — | — | — |
| MUXSEL1 | — | — | 9h | — | — | — | — | — |
| MUXSEL2 | — | 0h | — | — | — | — | — | — |
| MUXSEL3 | 0h | — | — | — | — | — | — | — |
| SWEN | — | — | — | — | 0h | | | |
| SWVALUE | — | — | — | — | 0h | | | |
| OUTEN | — | — | — | — | 3h | | | |

#### 59.6.1.7.7 Register configuration for LC1

Table 307. Register configuration for LC1

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| LUTCTRL0 | — | — | — | — | 3AF0h | — | — | — |
| LUTCTRL1 | — | — | — | — | — | 0h | — | — |
| LUTCTRL2 | — | — | — | — | — | — | 0h | — |
| LUTCTRL3 | — | — | — | — | — | — | — | 0h |
| FILT0 | — | — | — | — | 0h | — | — | — |
| FILT1 | — | — | — | — | — | 0h | — | — |
| FILT2 | — | — | — | — | — | — | 0h | — |
| FILT3 | — | — | — | — | — | — | — | 0h |
| INTDMAEN | — | — | — | — | 0h | | | |
| OUTPOL | — | — | — | — | 0h | | | |
| FFILT | — | — | — | — | 0h | | | |
| FCTRL | — | — | — | — | 0h | | | |
| SCTRL | — | — | — | — | 0h | | | |
| MUXSEL4 | — | — | — | 5h | — | — | — | — |
| MUXSEL5 | — | — | 6h | — | — | — | — | — |
| MUXSEL6 | — | Dh | — | — | — | — | — | — |
| MUXSEL7 | Ah | — | — | — | — | — | — | — |
| SWEN | — | — | — | — | 0h | | | |
| SWVALUE | — | — | — | — | 0h | | | |
| OUTEN | — | — | — | — | 10h | | | |

#### 59.6.1.7.8 Waveforms

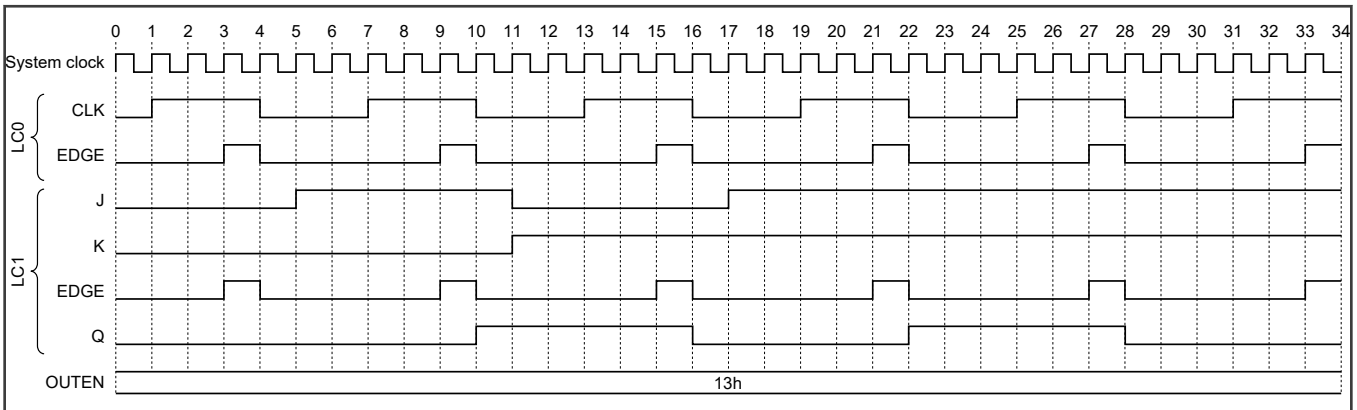



Figure 272. Waveforms

#### 59.6.1.8 Incremental encoder

##### 59.6.1.8.1 Overview

An incremental encoder provides A and B pulse outputs in the form of 90-degree shifted waveforms (see Outputs). The decoding circuit processes these waveforms to detect the rotor position and the direction of rotation.

This example implementation requires two LCs (see Implementation):

- LC0 provides pulse streams on clockwise (CW) or counterclockwise (CCW) output at every A and B signal edge. If the encoder rotation is CW (A signal leading), then pulses appear at the CW output. If the encoder rotation is CCW (B signal leading), then pulses appear at the CCW output. Two pulse counters accumulate these pulse streams to provide the actual absolute encoder position in the respective direction of rotation.
- LC1 then uses the LC0 outputs to detect the direction of the rotation.

##### 59.6.1.8.2 Outputs

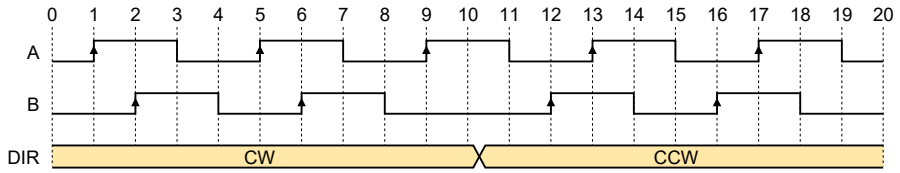



Figure 273. Outputs

##### 59.6.1.8.3 Implementation

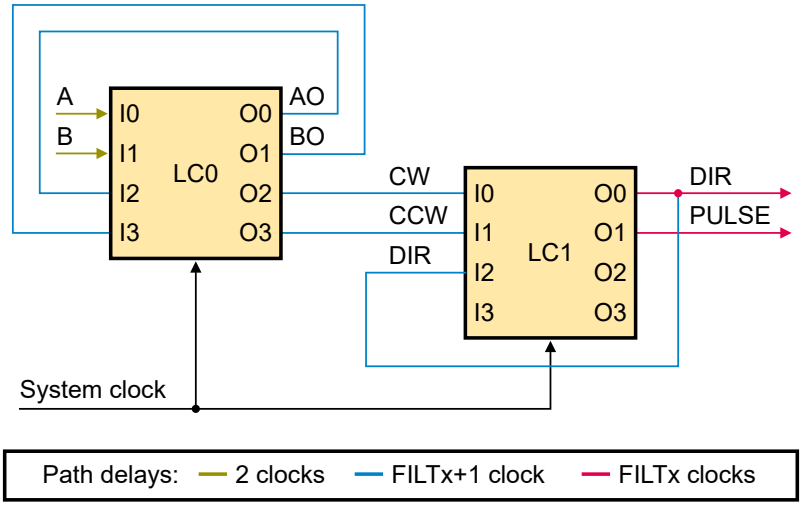



Figure 274. Implementation

#### 59.6.1.8.4 Truth table for pulse generation

**Table 308. Truth table for pulse generation**

| Inputs | | | | Outputs | | | |
|---|---|---|---|---|---|---|---|
| BO | AO | B | A | AO | BO | CW | CCW |
| 0 | 0 | 0 | 0 | 0 | 0 | — | — |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | — |
| 0 | 0 | 1 | 0 | 0 | 1 | — | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 | | |
| 0 | 1 | 0 | 0 | 0 | 0 | — | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | | |
| 0 | 1 | 1 | 0 | 0 | 1 | — | — |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | — |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | |
| 1 | 0 | 0 | 1 | 1 | 0 | — | — |
| 1 | 0 | 1 | 0 | 0 | 1 | — | — |
| 1 | 0 | 1 | 1 | 1 | 1 | — | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | | |
| 1 | 1 | 0 | 1 | 1 | 0 | — | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | — |
| 1 | 1 | 1 | 1 | 1 | 1 | — | — |

#### 59.6.1.8.5 Truth table for decoding direction of rotation

In this table:

- DIR = 0 represents a CCW direction.
- DIR = 1 represents a CW direction.

**Table 309. Truth table for decoding direction of rotation**

| Inputs | | | Output |
|---|---|---|---|
| DIR | CCW | CW | DIR |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

#### 59.6.1.8.6 Register configuration for LC0

Table 310. Register configuration for LC0

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| LUTCTRL0 | — | — | — | — | AAAAh | — | — | — |
| LUTCTRL1 | — | — | — | — | — | CCCCh | — | — |
| LUTCTRL2 | — | — | — | — | — | — | 4182h | — |
| LUTCTRL3 | — | — | — | — | — | — | — | 2814h |
| FILT0 | — | — | — | — | 1_0001h | — | — | — |
| FILT1 | — | — | — | — | — | 1_0001h | — | — |
| FILT2 | — | — | — | — | — | — | 0h | — |
| FILT3 | — | — | — | — | — | — | — | 0h |
| INTDMAEN | — | — | — | — | 0h | | | |
| OUTPOL | — | — | — | — | 0h | | | |
| FFILT | — | — | — | — | 0h | | | |
| FCTRL | — | — | — | — | 0h | | | |
| SCTRL | — | — | — | — | 0h | | | |
| MUXSEL0 | — | — | — | 1h | — | — | — | — |
| MUXSEL1 | — | — | 2h | — | — | — | — | — |
| MUXSEL2 | — | 9h | — | — | — | — | — | — |
| MUXSEL3 | Ah | — | — | — | — | — | — | — |
| SWEN | — | — | — | — | 0h | | | |
| SWVALUE | — | — | — | — | 0h | | | |
| OUTEN | — | — | — | — | Fh | | | |

#### 59.6.1.8.7 Register configuration for LC1

Table 311. Register configuration for LC1

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| LUTCTRL0 | — | — | — | — | BAh | — | — | — |
| LUTCTRL1 | — | — | — | — | — | 0h | — | — |
| LUTCTRL2 | — | — | — | — | — | — | 0h | — |
| LUTCTRL3 | — | — | — | — | — | — | — | 0h |
| FILT0 | — | — | — | — | 0h | — | — | — |
| FILT1 | — | — | — | — | — | 0h | — | — |
| FILT2 | — | — | — | — | — | — | 0h | — |
| FILT3 | — | — | — | — | — | — | — | 0h |

*Table continues on the next page...*

Table 311. Register configuration for LC1 (continued)

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| INTDMAEN | — | — | — | — | 0h | | | |
| OUTPOL | — | — | — | — | 0h | | | |
| FFILT | — | — | — | — | 0h | | | |
| FCTRL | — | — | — | — | 0h | | | |
| SCTRL | — | — | — | — | 0h | | | |
| MUXSEL4 | — | — | — | Bh | — | — | — | — |
| MUXSEL5 | — | — | Ch | — | — | — | — | — |
| MUXSEL6 | — | Dh | — | — | — | — | — | — |
| MUXSEL7 | 0h | — | — | — | — | — | — | — |
| SWEN | — | — | — | — | 0h | | | |
| SWVALUE | — | — | — | — | 0h | | | |
| OUTEN | — | — | — | — | 10h | | | |

#### 59.6.1.8.8 Waveforms

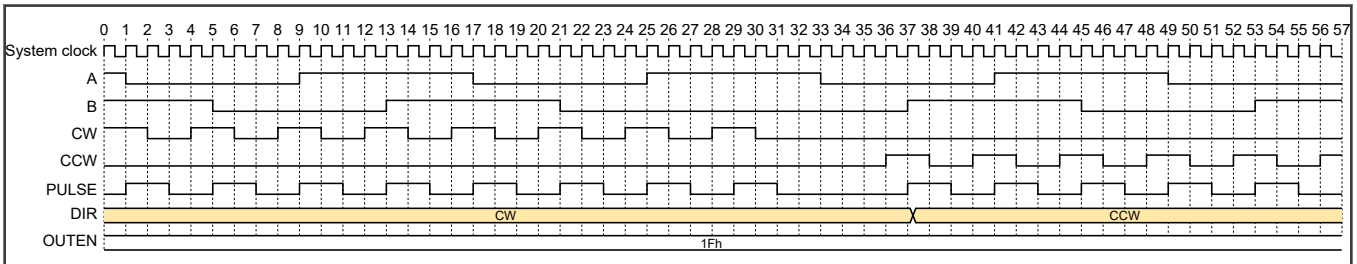



Figure 275. Waveforms

### 59.6.1.9 AC motor PWM controller

#### 59.6.1.9.1 Overview

Driving a three-phase AC motor requires dedicated PWMs to perform:

- Complementary signal generation
- Dead time insertion
- Fault-state assertion
- Manual or automatic fault recovery

LCU enhances basic timer functionality with the necessary complementary features available only on dedicated motor control PWMs.

Implementation for AC motor controller shows all input and output signals of LCU in a typical three-phase motor control application.

TdH is the time delay preceding each rising edge of the complementary PWM output signals. See Truth table for details.

Fault signals are routed from analog comparators or external protection circuits to protect power switches against damage caused by overcurrent. Any active fault must immediately change all PWM outputs to the inactive state. Force logic causes PWM outputs to transition to the inactive state upon an external fault event. See Truth table for details.

The dead time is one clock cycle. On a fault event, both PWM outputs transition to the inactive state (logic 0). When the fault event deasserts, both PWM signals start operating according to FORCE_MODE = 00 (cleared on force deassertion). See Waveforms for force cleared on force deassertion for details.

#### 59.6.1.9.2 Implementation for AC motor controller

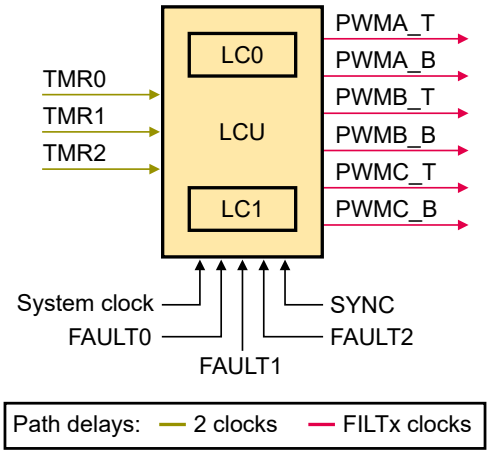



Figure 276. Implementation for AC motor controller

#### 59.6.1.9.3 Implementation for controlling one phase of AC motor controller

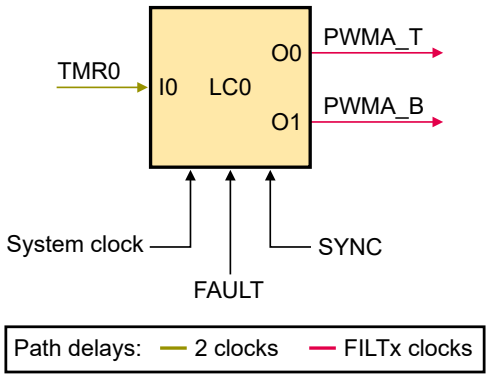



Figure 277. Implementation for controlling one phase of AC motor controller

#### 59.6.1.9.4 Connections and controls

Table 312. Connections and controls

| Name | Description |
|---|---|
| TMR*n* | Synchronized timer input signals |
| System clock | Clock supplied to the LC |
| FAULT*n* | Fault signal (see Overview) |

*Table continues on the next page...*

Table 312. Connections and controls (continued)

| Name | Description |
|---|---|
| SYNC | Sync signal for optional automatic fault recovery |
| PWM*x*_T | PWM output signal for controlling top power switch |
| PWM*x*_B | PWM output signal for controlling bottom power switch |

#### 59.6.1.9.5 Truth table

Table 313. Truth table

| Input | Outputs | |
|---|---|---|
| TMR0 | PWMA_T | PWMB_B |
| 0 | 0 | 1 delayed by TdH (see Output edge delays) |
| 1 | 1 delayed by TdH (see Output edge delays) | 0 |

#### 59.6.1.9.6 Register configuration

Table 314. Register configuration

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| LUTCTRL0 | — | — | — | — | 2h | — | — | — |
| LUTCTRL1 | — | — | — | — | — | 1h | — | — |
| LUTCTRL2 | — | — | — | — | — | — | 0h | — |
| LUTCTRL3 | — | — | — | — | — | — | — | 0h |
| FILT0 | — | — | — | — | 1_0000h | — | — | — |
| FILT1 | — | — | — | — | — | 1_0000h | — | — |
| FILT2 | — | — | — | — | — | — | 0h | — |
| FILT3 | — | — | — | — | — | — | — | 0h |
| INTDMAEN | — | — | — | — | 0h | | | |
| OUTPOL | — | — | — | — | 0h | | | |
| FFILT | — | — | — | — | 100_0000h | | | |
| FCTRL | — | — | — | — | 0h | | | |
| SCTRL | — | — | — | — | Software sync on rising edge for inputs 2 and 0, immediate for 3 and 1: 101h<br>Software sync on rising edge for all four inputs: 1111h | | | |
| MUXSEL0 | — | — | — | 01h | — | — | — | — |
| MUXSEL1 | — | — | 0h | — | — | — | — | — |
| MUXSEL2 | — | 0h | — | — | — | — | — | — |
| MUXSEL3 | 0h | — | — | — | — | — | — | — |

*Table continues on the next page...*

Table 314. Register configuration (continued)

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| SWEN | — | — | — | — | 0h | | | |
| SWVALUE | — | — | — | — | 0h | | | |
| OUTEN | — | — | — | — | 3h | | | |

#### 59.6.1.9.7 Waveforms for force cleared on force deassertion

This example shows two behaviors, depending on the Force Clearing mode (LC0_FCTRL[FORCE_MODE0]).

00b - Both PWM signals start normal operation when the fault event deasserts. See Waveforms for force cleared on force deassertion for details.

01b - Both PWM signals start normal operation on rising sync after the fault event deasserts. See Waveforms for force cleared on rising sync after force deassertion for details.

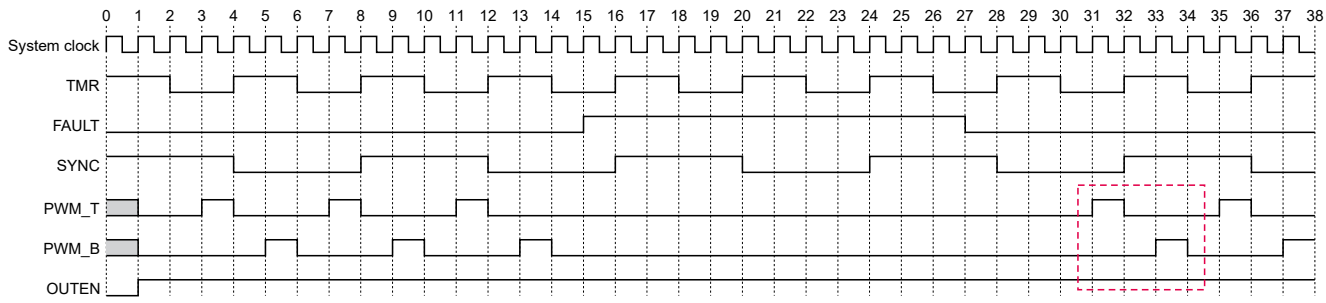



Figure 278. Waveforms for force cleared on force deassertion

#### 59.6.1.9.8 Waveforms for force cleared on rising sync after force deassertion

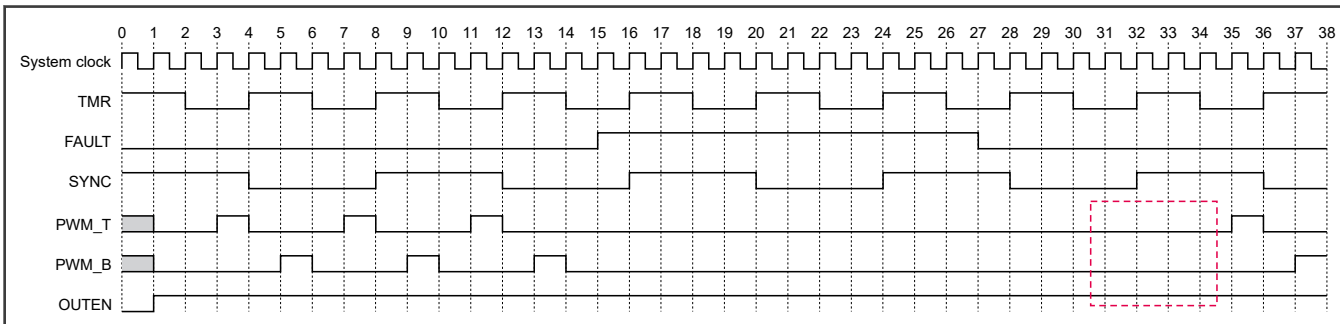



Figure 279. Waveforms for force cleared on rising sync after force deassertion

### 59.6.1.10 AC motor double-switching controller

#### 59.6.1.10.1 Overview

The simplest method of obtaining motor winding currents is to measure the current in each phase by placing a shunt resistor in two or three inverter legs (phases).

This example implementation of an AC motor drive for low-cost and high-volume applications uses a single shunt on the DC bus return path, as illustrated in AC motor power stage schematic. LCU supports double-switching by XORing two timer channel outputs. The implementation uses two LCs arranged to perform double switching, dead time generation, and the processing of three external fault signals.

#### 59.6.1.10.2 Double-switching

The double-switching technique requires:

- Six synchronized timer pulses
- Three fault signals
- Six PWM outputs for controlling top and bottom power switches
- A sync signal for automatic fault recovery

Waveforms for double-switching technique example applies the PWM switching cycle to an AC motor to reconstruct currents in three phases using a single DC bus shunt resistor (RSH). It applies the non-zero switching vectors 100, 110, and 101 to the power switches, which sample currents IA[0], -IC[0], -IC[1], and IA[1]. Because the durations (T1) of active vectors 110 and 101 are the same, it also applies the zero switching vector (000) with durations T2 and T3 in the middle of the switching cycle to allow phase current reconstruction. This technique can only be used by PWM modules targeted at motor control and power conversion applications.

#### 59.6.1.10.3 Fault inputs and deadtime insertion

Fault signals are routed from analog comparators or external protection circuits to protect power switches against damage caused by overcurrent. You must program the Combinational Force Path (LCn_FFILT[COMB_EN]) to enable an active fault input to immediately force all PWM outputs to the inactive state.

#### 59.6.1.10.4 Deadtime insertion

In Waveforms for generation of PWM complementary signals using an LC, the PWMA_T output is generated by XORing timer outputs TMR0 and TMR1. The PWMA_B output is complementary to PWMA_T. Digital filters insert a dead time of one system clock cycle to delay the rising edges of both PWM signals.

#### 59.6.1.10.5 Fault events

Waveforms for force cleared on force deassertion shows the reaction to the fault event. On a fault event, both the PWM outputs transition to the inactive state (logic 0). When fault event deasserts, both the PWM signals start operating according to FORCE_MODE = 00 (cleared on force deassertion).

Alternatively, after the fault event deasserts, fault clearing can be performed according to FORCE_MODE = 01 (cleared on rising sync force deassertion). See Waveforms for force cleared on rising sync after force deassertion for details.

### 59.6.1.10.6 AC motor power stage schematic

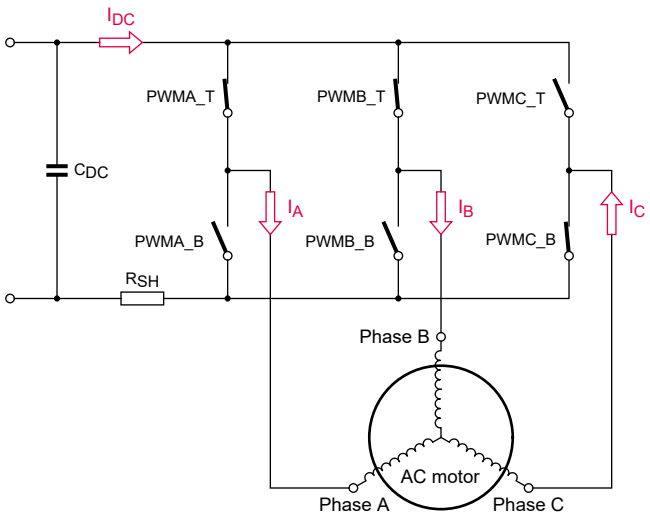



Figure 280. AC motor power stage schematic

### 59.6.1.10.7 Waveforms for double-switching technique example

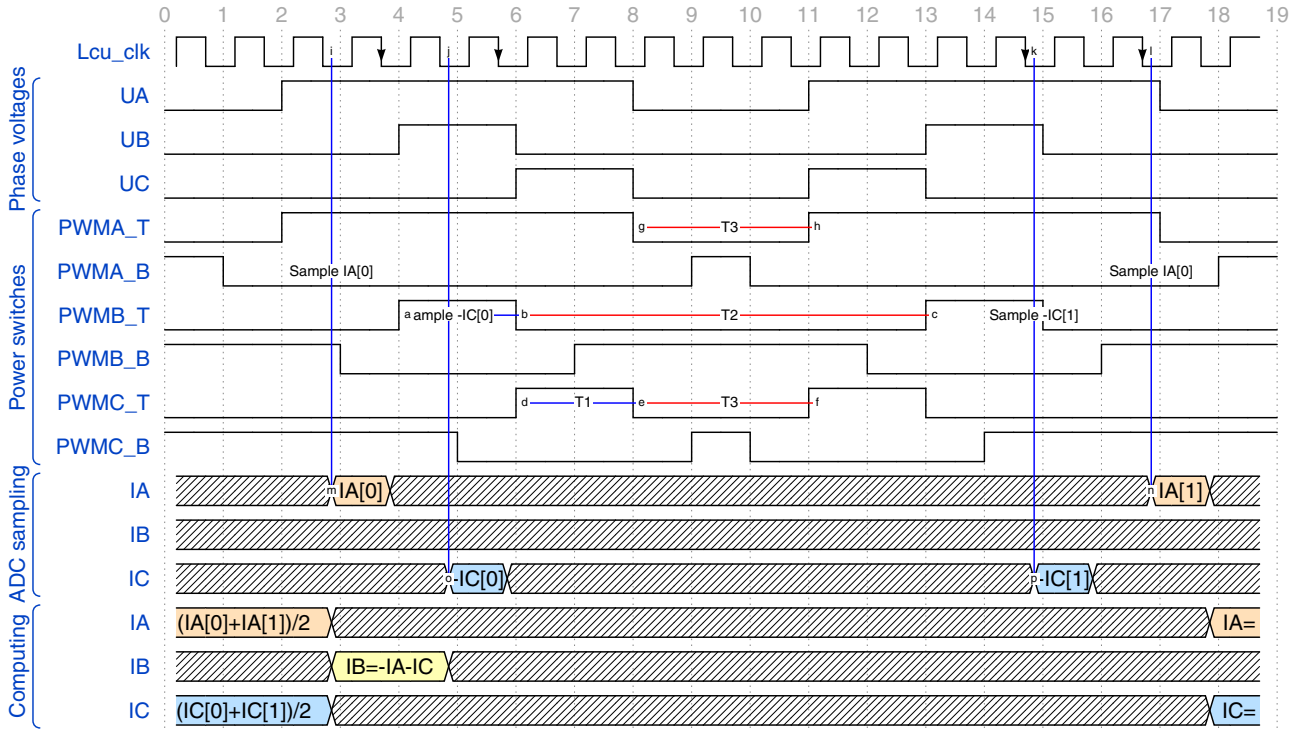



Figure 281. Waveforms for double-switching technique example

#### 59.6.1.10.8 Implementation for AC motor double-switching controller

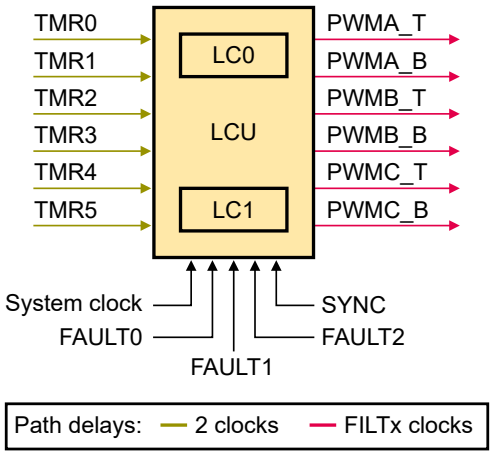



Figure 282. Implementation for AC motor double-switching controller

#### 59.6.1.10.9 Implementation for controlling one phase

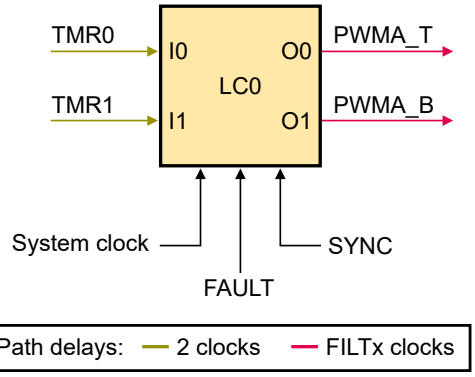



Figure 283. Implementation for controlling one phase

#### 59.6.1.10.10 Connections and controls

Table 315. Connections and controls

| Name | Description |
|---|---|
| TMR*n* | Synchronized timer input signals |
| System clock | Clock supplied to the LC |
| FAULT*n* | Fault signal (see Implementation for AC motor double-switching controller for more details) |
| SYNC | Sync signal for optional automatic fault recovery |
| PWM*x*_T | PWM output signal for controlling top power switch |
| PWM*x*_B | PWM output signal for controlling bottom power switch |

#### 59.6.1.10.11 Truth table

**Table 316. Truth table**

| Inputs | | Outputs | |
|---|---|---|---|
| TMR1 | TMR0 | PWMA_T | PWMB_B |
| 0 | 0 | 0 | 1 delayed by TdH (see Output edge delays) |
| 0 | 1 | 1 delayed by TdH (see Output edge delays) | 0 |
| 1 | 0 | 1 delayed by TdH (see Output edge delays) | 0 |
| 1 | 1 | 0 | 1 delayed by TdH (see Output edge delays) |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 |

#### 59.6.1.10.12 Register configuration

**Table 317. Register configuration**

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| LUTCTRL0 | — | — | — | — | 6h | — | — | — |
| LUTCTRL1 | — | — | — | — | — | 9h | — | — |
| LUTCTRL2 | — | — | — | — | — | — | 0h | — |
| LUTCTRL3 | — | — | — | — | — | — | — | 0h |
| FILT0 | — | — | — | — | 1_0000h | — | — | — |
| FILT1 | — | — | — | — | — | 1_0000h | — | — |
| FILT2 | — | — | — | — | — | — | 0h | — |
| FILT3 | — | — | — | — | — | — | — | 0h |
| INTDMAEN | — | — | — | — | 0h | | | |
| OUTPOL | — | — | — | — | 0h | | | |
| FFILT | — | — | — | — | 100_0000h | | | |
| FCTRL | — | — | — | — | Force inputs 2 and 0 affect output 0, but force inputs 3 and 1 do not affect output 0: 101h<br>All four inputs affect output 0: 1111h | | | |
| SCTRL | — | — | — | — | 0h | | | |
| MUXSEL0 | — | — | — | 1h | — | — | — | — |
| MUXSEL1 | — | — | 2h | — | — | — | — | — |
| MUXSEL2 | — | 0h | — | — | — | — | — | — |
| MUXSEL3 | 0h | — | — | — | — | — | — | — |

*Table continues on the next page...*

Table 317. Register configuration (continued)

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| SWEN | — | — | — | — | 0h | | | |
| SWVALUE | — | — | — | — | 0h | | | |
| OUTEN | — | — | — | — | 3h | | | |

### 59.6.1.10.13 Waveforms for generation of PWM complementary signals using an LC

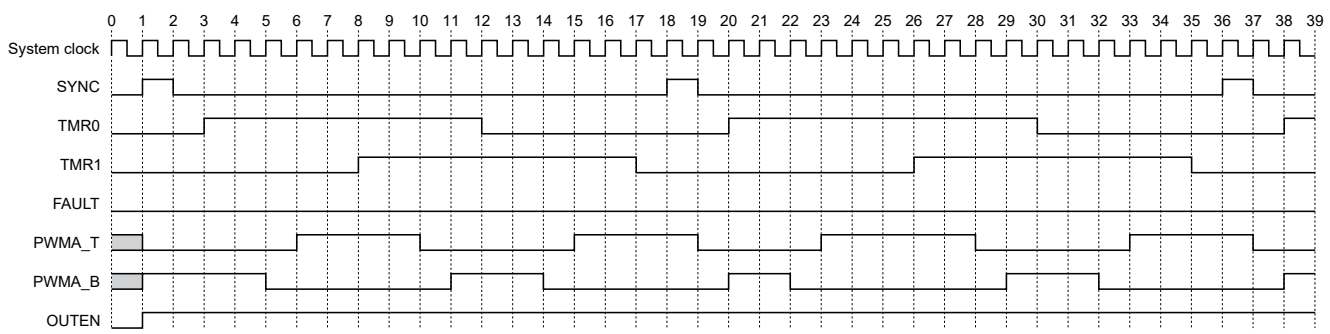



Figure 284. Waveforms for generation of PWM complementary signals using an LC

### 59.6.1.10.14 Waveforms for force cleared on force deassertion

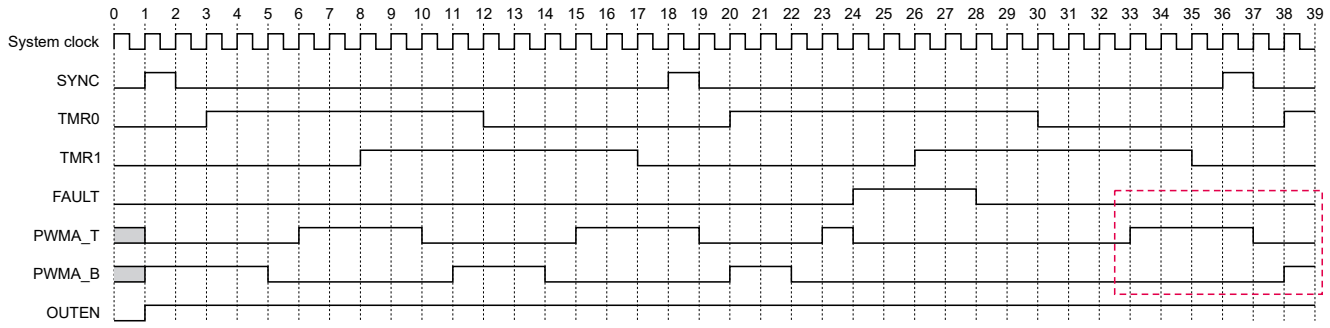



Figure 285. Waveforms for force cleared on force deassertion

### 59.6.1.10.15 Waveforms for force cleared on rising sync after force deassertion

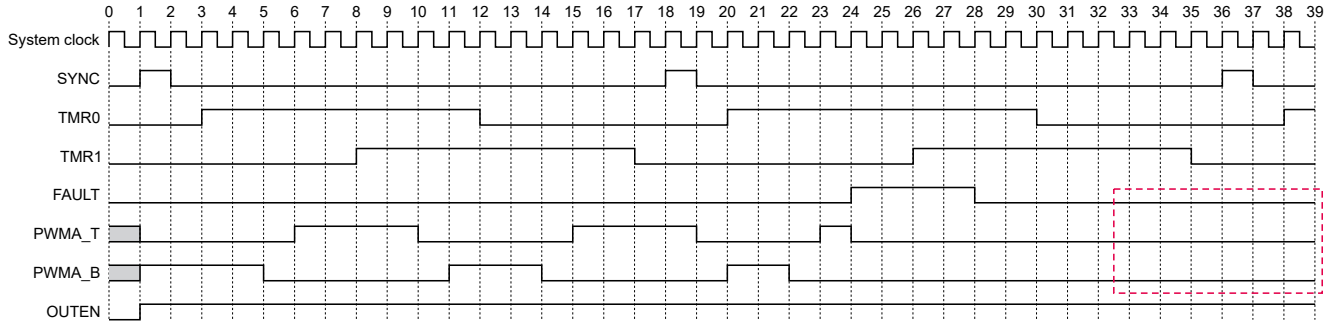



Figure 286. Waveforms for force cleared on rising sync after force deassertion

### 59.6.1.11 Brushless DC (BLDC) motor PWM controller

#### 59.6.1.11.1 Overview

You can use LCU to replace the mechanical commutator in the BLDC motor control application with Hall-sensor position measurement.

In this example implementation, LC0 receives the direction (DIR) and Hall A, Hall B, and Hall C sensor output states, and from those inputs generates the switching states S1, S2, and S3. LC1 and LC2 convert the switching states to PWM or on-off signals for controlling power switches, as illustrated in Waveforms for BLDC motor control technique.

BLDC motor power stage schematic and BLDC motor electrical arrangement illustrate a three-phase BLDC motor power stage with switching pulses and phase currents controlling motor rotation in the CCW direction. The state of the Hall sensors is 010. As the rotor shaft rotates, Hall sensors track the motor position. You accelerate or decelerate the motor by controlling the width of the PWM signal. You can include one or more faults (force inputs) to force power switches instantaneously into the inactive state if there is an overcurrent.

#### 59.6.1.11.2 Implementation

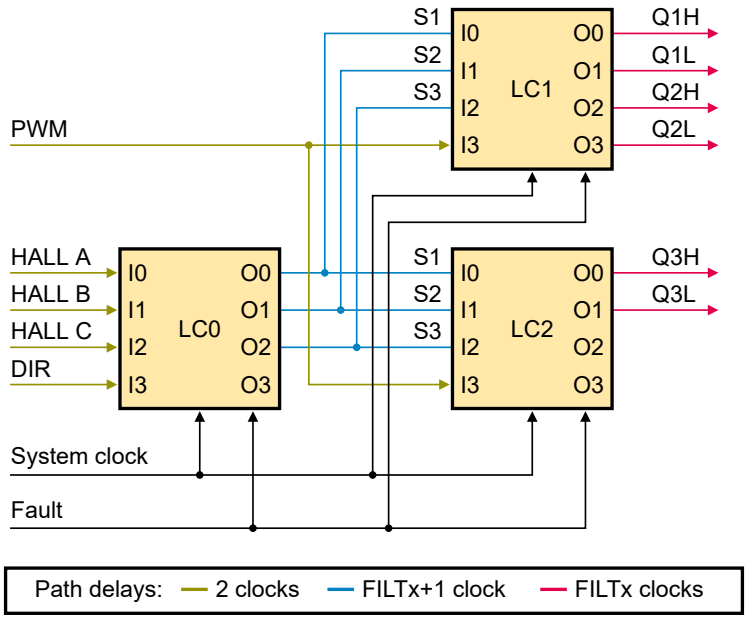



Figure 287. Implementation

### 59.6.1.11.3 BLDC motor power stage schematic

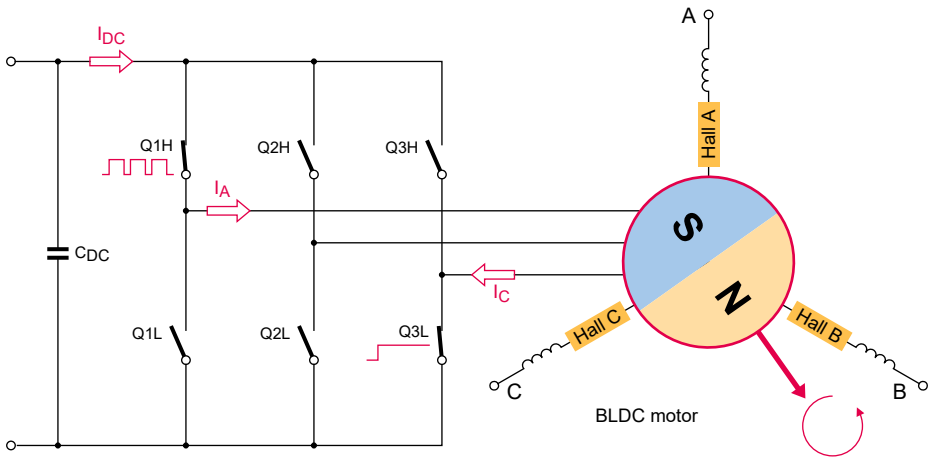



Figure 288. BLDC motor power stage schematic

### 59.6.1.11.4 BLDC motor electrical arrangement

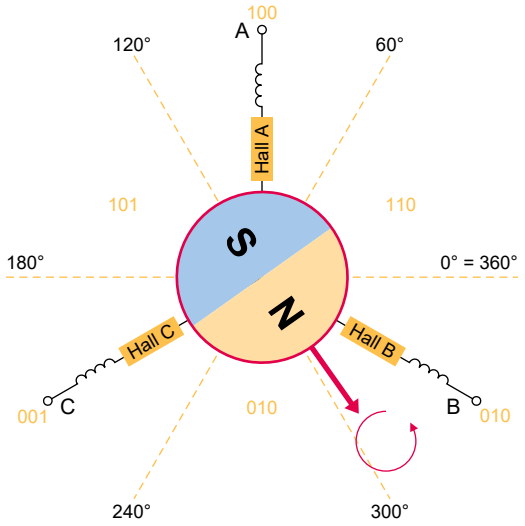



Figure 289. BLDC motor electrical arrangement

### 59.6.1.11.5 Waveforms for BLDC motor control technique

This figure illustrates the basic waveforms for controlling a BLDC motor with stator windings and Hall sensors arranged according to BLDC motor electrical arrangement. Configure the external hardware (motor or sensor) so that it can produce these waveforms; then you can configure LCU.

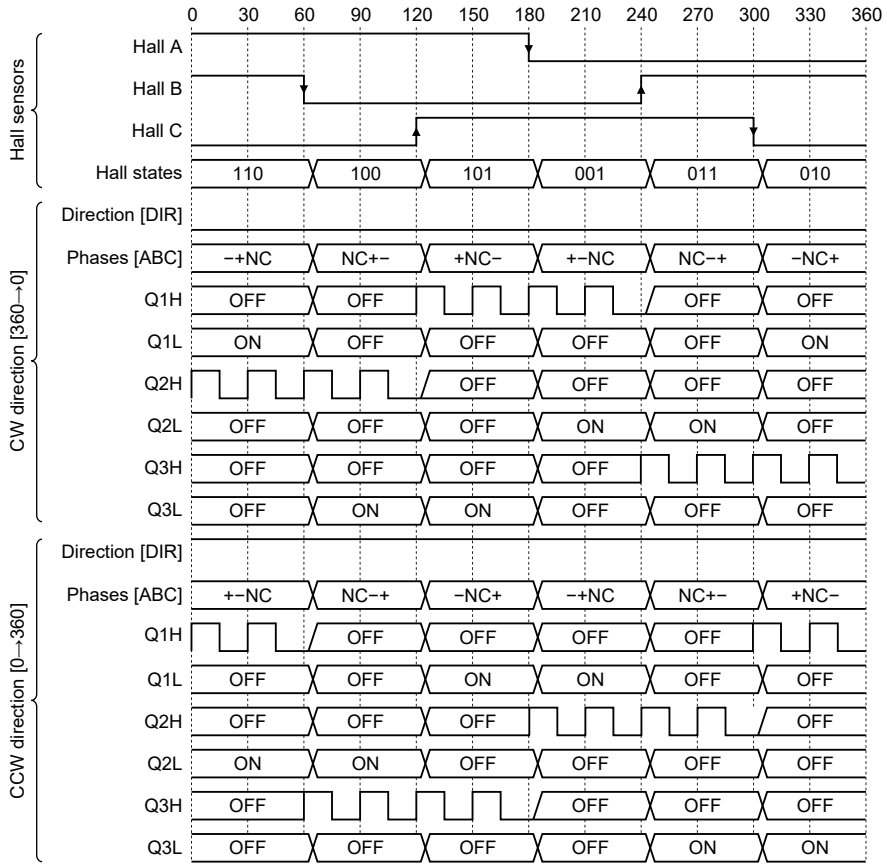



Figure 290. Waveforms for BLDC motor control technique

### 59.6.1.11.6 Connections and controls

Table 318. Connections and controls

| Input-output group | Name | Description |
|---|---|---|
| LC0 inputs | HALL A | Hall sensor A output state |
| | HALL B | Hall sensor B output state |
| | HALL C | Hall sensor C output state |
| | DIR | Direction of shaft rotation, as illustrated in Waveforms for reversing BLDC motor. |
| LC1 and LC2 inputs | S1 | Switching state 1 (output 0 from LC0) |
| | S2 | Switching state 2 (output 1 from LC0) |
| | S2 | Switching state 3 (output 2 from LC0) |
| | PWM | Output from PWM |
| LC0, LC1, and LC2 inputs | Clock | LCU clock input |
| | Fault | Optional fault input |

*Table continues on the next page...*

Table 318. Connections and controls (continued)

| Input-output group | Name | Description |
|---|---|---|
| LC1 outputs | Q1H | Motor control switch outputs, as illustrated in BLDC motor electrical arrangement |
| | Q1L | |
| | Q2H | |
| | Q2L | |
| LC2 outputs | Q3H | |
| | Q3L | |

### 59.6.1.11.7 Truth table for generating switching states

This table includes all Hall sensor states, even those that never occur because of sensor displacement by 120°.

Table 319. Truth table for generating switching states

| Inputs | | | | Outputs | | | Comments |
|---|---|---|---|---|---|---|---|
| DIR | Hall C | Hall B | Hall A | S1 | S2 | S3 | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | Not used |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | Hall states |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | Hall states |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | Hall states |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | Hall states |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | Hall states |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | Hall states |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | Not used |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | Not used |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | Complement of Hall states |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | Complement of Hall states |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 | Complement of Hall states |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | Complement of Hall states |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | Complement of Hall states |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | Complement of Hall states |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | Not used |

#### 59.6.1.11.8 Truth table for generating switching pulses for phases A and B

Table 320. Truth table for generating switching pulses for phases A and B

| Inputs | | | | Outputs | | | | Comments |
|---|---|---|---|---|---|---|---|---|
| PWM | S3 | S2 | S1 | Q1H | Q1L | Q2H | Q2L | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Not used |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | — |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | — |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | — |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | — |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | — |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | — |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | Not used |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Not used |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | — |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | — |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | — |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | — |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | — |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | — |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | Not used |

#### 59.6.1.11.9 Truth table for generating switching pulses for phase C

Table 321. Truth table for generating switching pulses for phase C

| Inputs | | | | Outputs | | Comments |
|---|---|---|---|---|---|---|
| PWM | S3 | S2 | S1 | Q3H | Q3L | |
| 0 | 0 | 0 | 0 | 0 | 0 | Not used |
| 0 | 0 | 0 | 1 | 0 | 1 | — |
| 0 | 0 | 1 | 0 | 0 | 0 | — |
| 0 | 0 | 1 | 1 | 0 | 0 | — |
| 0 | 1 | 0 | 0 | 0 | 0 | — |
| 0 | 1 | 0 | 1 | 0 | 1 | — |
| 0 | 1 | 1 | 0 | 0 | 0 | — |
| 0 | 1 | 1 | 1 | 0 | 0 | Not used |
| 1 | 0 | 0 | 0 | 0 | 0 | Not used |
| 1 | 0 | 0 | 1 | 0 | 1 | — |

*Table continues on the next page...*

Table 321. Truth table for generating switching pulses for phase C (continued)

| Inputs | | | | Outputs | | Comments |
|---|---|---|---|---|---|---|
| PWM | S3 | S2 | S1 | Q3H | Q3L | |
| 1 | 0 | 1 | 0 | 1 | 0 | — |
| 1 | 0 | 1 | 1 | 0 | 0 | — |
| 1 | 1 | 0 | 0 | 0 | 0 | — |
| 1 | 1 | 0 | 1 | 0 | 1 | — |
| 1 | 1 | 1 | 0 | 1 | 0 | — |
| 1 | 1 | 1 | 1 | 0 | 0 | Not used |

### 59.6.1.11.10 Configuration for generating switching states

Table 322. Configuration for generating switching states

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| LUTCTRL0 | — | — | — | — | 542Ah | — | — | — |
| LUTCTRL1 | — | — | — | — | — | 324Ch | — | — |
| LUTCTRL2 | — | — | — | — | — | — | E70h | — |
| LUTCTRL3 | — | — | — | — | — | — | — | 0h |
| FILT0 | — | — | — | — | 0h | — | — | — |
| FILT1 | — | — | — | — | — | 0h | — | — |
| FILT2 | — | — | — | — | — | — | 0h | — |
| FILT3 | — | — | — | — | — | — | — | 0h |
| INTDMAEN | — | — | — | — | 0h | | | |
| OUTPOL | — | — | — | — | 0h | | | |
| FFILT | — | — | — | — | 0h | | | |
| FCTRL | — | — | — | — | 0h | | | |
| SCTRL | — | — | — | — | 0h | | | |
| MUXSEL0 | — | — | — | 1h | — | — | — | — |
| MUXSEL1 | — | — | 2h | — | — | — | — | — |
| MUXSEL2 | — | 3h | — | — | — | — | — | — |
| MUXSEL3 | 4h | — | — | — | — | — | — | — |
| SWEN | — | — | — | — | 0h | | | |
| SWVALUE | — | — | — | — | 0h | | | |
| OUTEN | — | — | — | — | 7h | | | |

#### 59.6.1.11.11 Configuration for switching pulses for phases A and B

Table 323. Configuration for switching pulses for phases A and B

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| LUTCTRL0 | — | — | — | — | 3000h | — | — | — |
| LUTCTRL1 | — | — | — | — | — | C0Ch | — | — |
| LUTCTRL2 | — | — | — | — | — | — | A00h | — |
| LUTCTRL3 | — | — | — | — | — | — | — | 5050h |
| FILT0 | — | — | — | — | 0h | — | — | — |
| FILT1 | — | — | — | — | — | 0h | — | — |
| FILT2 | — | — | — | — | — | — | 0h | — |
| FILT3 | — | — | — | — | — | — | — | 0h |
| INTDMAEN | — | — | — | — | 0h | | | |
| OUTPOL | — | — | — | — | 0h | | | |
| FFILT | — | — | — | — | F00_0000h | | | |
| FCTRL | — | — | — | — | 101_0101h | | | |
| SCTRL | — | — | — | — | 0h | | | |
| MUXSEL4 | — | — | — | Dh | — | — | — | — |
| MUXSEL5 | — | — | Eh | — | — | — | — | — |
| MUXSEL6 | — | Fh | — | — | — | — | — | — |
| MUXSEL7 | 8h | — | — | — | — | — | — | — |
| SWEN | — | — | — | — | 0h | | | |
| SWVALUE | — | — | — | — | 0h | | | |
| OUTEN | — | — | — | — | F0h | | | |

#### 59.6.1.11.12 Configuration for switching pulses for phase C

Table 324. Configuration for switching pulses for phase C

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| LUTCTRL0 | — | — | — | — | 4400h | — | — | — |
| LUTCTRL1 | — | — | — | — | — | 2222h | — | — |
| LUTCTRL2 | — | — | — | — | — | — | 0h | — |
| LUTCTRL3 | — | — | — | — | — | — | — | 0h |
| FILT0 | — | — | — | — | 0h | — | — | — |
| FILT1 | — | — | — | — | — | 0h | — | — |
| FILT2 | — | — | — | — | — | — | 0h | — |
| FILT3 | — | — | — | — | — | — | — | 0h |

*Table continues on the next page...*

Table 324. Configuration for switching pulses for phase C (continued)

| Register | I3 | I2 | I1 | I0 | O0 | O1 | O2 | O3 |
|---|---|---|---|---|---|---|---|---|
| INTDMAEN | — | — | — | — | 0h | | | |
| OUTPOL | — | — | — | — | 0h | | | |
| FFILT | — | — | — | — | 0300_0000h | | | |
| FCTRL | — | — | — | — | 101h | | | |
| SCTRL | — | — | — | — | 0h | | | |
| MUXSEL8 | — | — | — | Dh | — | — | — | — |
| MUXSEL9 | — | — | Eh | — | — | — | — | — |
| MUXSEL10 | — | Fh | — | — | — | — | — | — |
| MUXSEL11 | 8h | — | — | — | — | — | — | — |
| SWEN | — | — | — | — | 0h | | | |
| SWVALUE | — | — | — | — | 0h | | | |
| OUTEN | — | — | — | — | 300h | | | |

### 59.6.1.11.13 Waveforms for reversing BLDC motor

This figure illustrates the waveforms for controlling a reversing BLDC motor. The DIR input indicates the direction of rotation:

- Logic 0 for CW rotation
- Logic 1 for CCW rotation

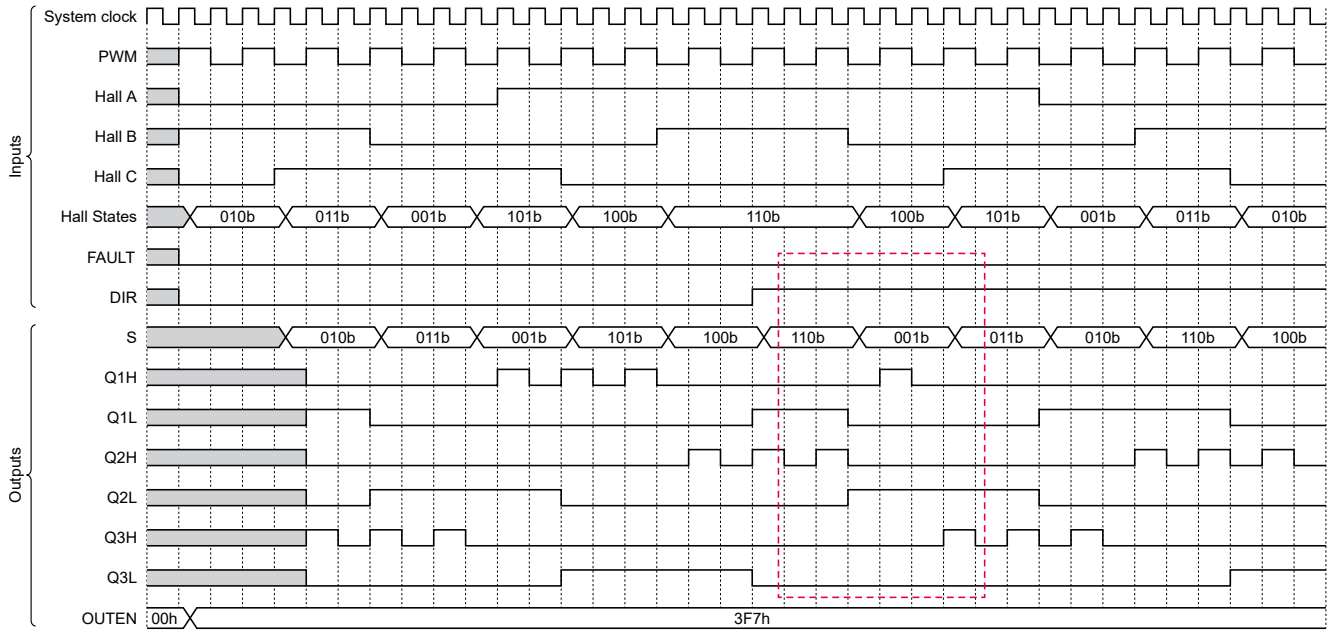



Figure 291. Waveforms for reversing BLDC motor

#### 59.6.1.11.14 Waveforms for reversing BLDC motor with fault assertion

This figure illustrates the waveforms for the reaction to an external fault in addition to reversing the BLDC motor. On the occurrence of an external fault event, the power switches transition to the inactive state (logic 0). Two clock cycles after external fault deassertion, the power switches start operating normally.

Switching state outputs S1, S2, and S3, generated by LC0, receive an additional delay of one clock cycle on their way to the inputs of other LCs (digital filters bypassed).

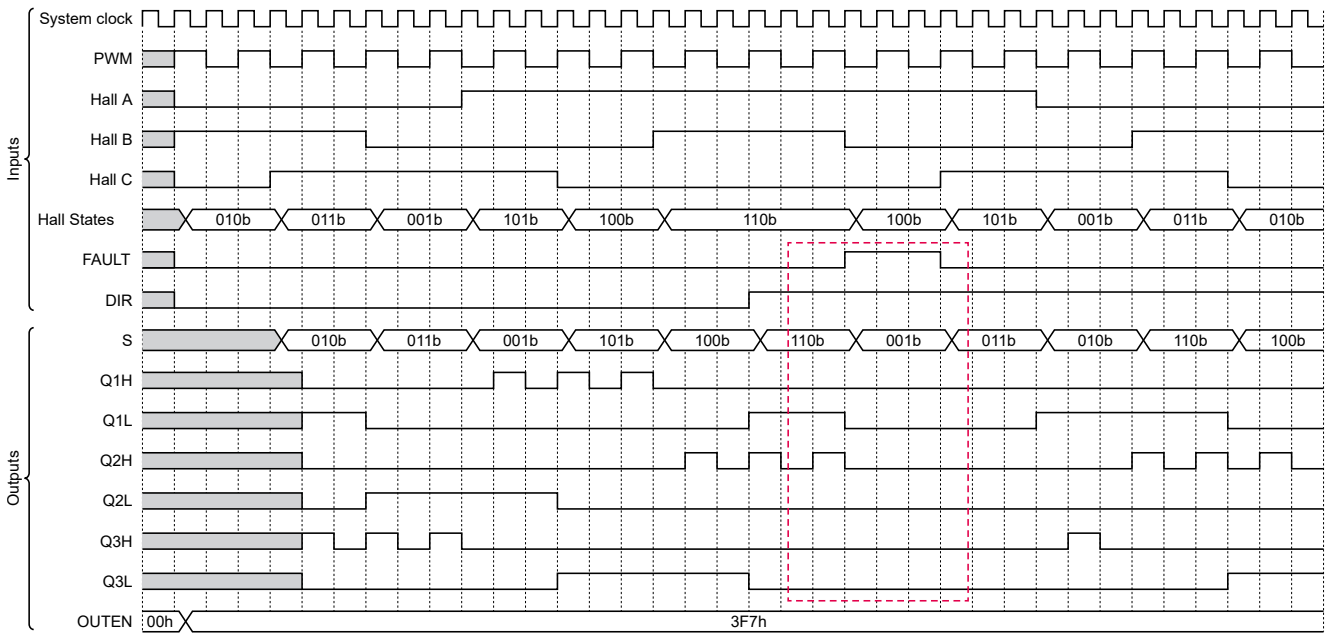



Figure 292. Waveforms for reversing BLDC motor with fault assertion

## 59.7 Mapping of 12-bit input, output, and state fields

This table maps inputs, outputs, or states to the bit positions in the following 12-bit fields:

- SWEN[SWEN]
- SWVALUE[SWVALUE]
- OUTEN[OUTEN]
- LCIN[LC_INPUTS]
- SWOUT[SWOUT]
- LCOUT[LCOUT]
- FORCEOUT[FORCEOUT]
- FORCESTS[FORCESTS]
- DBGEN[DBGEN]

| LC | Input, output, or state | Bit |
|---|---|---|
| 0 | 0 | 0 |
| | 1 | 1 |

*Table continues on the next page...*

*Table continued from the previous page...*

| LC | Input, output, or state | Bit |
|---|---|---|
| | 2 | 2 |
| | 3 | 3 |
| 1 | 0 | 4 |
| | 1 | 5 |
| | 2 | 6 |
| | 3 | 7 |
| 2 | 0 | 8 |
| | 1 | 9 |
| | 2 | 10 |
| | 3 | 11 |

## 59.8 LCU register descriptions

**NOTE**

Access to address offset 2A4h does not generate a transfer error.

### 59.8.1 LCU memory map

LCU_0 base address: 4009_8000h

LCU_1 base address: 4009_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | LC 0 Output 0 LUT Control (LC0_LUTCTRL0) | 32 | RW | 0000_0000h |
| 4h | LC 0 Output 1 LUT Control (LC0_LUTCTRL1) | 32 | RW | 0000_0000h |
| 8h | LC 0 Output 2 LUT Control (LC0_LUTCTRL2) | 32 | RW | 0000_0000h |
| Ch | LC 0 Output 3 LUT Control (LC0_LUTCTRL3) | 32 | RW | 0000_0000h |
| 10h | LC 0 Output 0 Filter (LC0_FILT0) | 32 | RW | 0000_0000h |
| 14h | LC 0 Output 1 Filter (LC0_FILT1) | 32 | RW | 0000_0000h |
| 18h | LC 0 Output 2 Filter (LC0_FILT2) | 32 | RW | 0000_0000h |
| 1Ch | LC 0 Output 3 Filter (LC0_FILT3) | 32 | RW | 0000_0000h |
| 20h | LC 0 Interrupt and DMA Enable (LC0_INTDMAEN) | 32 | RW | 0000_0000h |
| 24h | LC 0 Status (LC0_STS) | 32 | RW | 0000_0000h |
| 28h | LC 0 Output Polarity Control (LC0_OUTPOL) | 32 | RW | 0000_0000h |
| 2Ch | LC 0 Force Filter (LC0_FFILT) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 30h | LC 0 Force Control (LC0_FCTRL) | 32 | RW | 0000_0000h |
| 34h | LC 0 Sync Control (LC0_SCTRL) | 32 | RW | 0000_0000h |
| 40h | LC 1 Output 0 LUT Control (LC1_LUTCTRL0) | 32 | RW | 0000_0000h |
| 44h | LC 1 Output 1 LUT Control (LC1_LUTCTRL1) | 32 | RW | 0000_0000h |
| 48h | LC 1 Output 2 LUT Control (LC1_LUTCTRL2) | 32 | RW | 0000_0000h |
| 4Ch | LC 1 Output 3 LUT Control (LC1_LUTCTRL3) | 32 | RW | 0000_0000h |
| 50h | LC 1 Output 0 Filter (LC1_FILT0) | 32 | RW | 0000_0000h |
| 54h | LC 1 Output 1 Filter (LC1_FILT1) | 32 | RW | 0000_0000h |
| 58h | LC 1 Output 2 Filter (LC1_FILT2) | 32 | RW | 0000_0000h |
| 5Ch | LC 1 Output 3 Filter (LC1_FILT3) | 32 | RW | 0000_0000h |
| 60h | LC 1 Interrupt and DMA Enable (LC1_INTDMAEN) | 32 | RW | 0000_0000h |
| 64h | LC 1 Status (LC1_STS) | 32 | RW | 0000_0000h |
| 68h | LC 1 Output Polarity Control (LC1_OUTPOL) | 32 | RW | 0000_0000h |
| 6Ch | LC 1 Force Filter (LC1_FFILT) | 32 | RW | 0000_0000h |
| 70h | LC 1 Force Control (LC1_FCTRL) | 32 | RW | 0000_0000h |
| 74h | LC 1 Sync Control (LC1_SCTRL) | 32 | RW | 0000_0000h |
| 80h | LC 2 Output 0 LUT Control (LC2_LUTCTRL0) | 32 | RW | 0000_0000h |
| 84h | LC 2 Output 1 LUT Control (LC2_LUTCTRL1) | 32 | RW | 0000_0000h |
| 88h | LC 2 Output 2 LUT Control (LC2_LUTCTRL2) | 32 | RW | 0000_0000h |
| 8Ch | LC 2 Output 3 LUT Control (LC2_LUTCTRL3) | 32 | RW | 0000_0000h |
| 90h | LC 2 Output 0 Filter (LC2_FILT0) | 32 | RW | 0000_0000h |
| 94h | LC 2 Output 1 Filter (LC2_FILT1) | 32 | RW | 0000_0000h |
| 98h | LC 2 Output 2 Filter (LC2_FILT2) | 32 | RW | 0000_0000h |
| 9Ch | LC 2 Output 3 Filter (LC2_FILT3) | 32 | RW | 0000_0000h |
| A0h | LC 2 Interrupt and DMA Enable (LC2_INTDMAEN) | 32 | RW | 0000_0000h |
| A4h | LC 2 Status (LC2_STS) | 32 | RW | 0000_0000h |
| A8h | LC 2 Output Polarity Control (LC2_OUTPOL) | 32 | RW | 0000_0000h |
| ACh | LC 2 Force Filter (LC2_FFILT) | 32 | RW | 0000_0000h |
| B0h | LC 2 Force Control (LC2_FCTRL) | 32 | RW | 0000_0000h |
| B4h | LC 2 Sync Control (LC2_SCTRL) | 32 | RW | 0000_0000h |
| 200h - 22Ch | Mux Select (MUXSEL0 - MUXSEL11) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 280h | Configuration (CFG) | 32 | RW | 0303_0280h |
| 284h | Software Override Enable (SWEN) | 32 | RW | 0000_0000h |
| 288h | Software Override Value (SWVALUE) | 32 | RW | 0000_0000h |
| 28Ch | Output Enable (OUTEN) | 32 | RW | 0000_0000h |
| 290h | Logic Inputs (LCIN) | 32 | R | 0000_0000h |
| 294h | Overridden Inputs (SWOUT) | 32 | R | 0000_0000h |
| 298h | Logic Outputs (LCOUT) | 32 | R | 0000_0000h |
| 29Ch | Forced Outputs (FORCEOUT) | 32 | R | 0000_0000h |
| 2A0h | Force Status (FORCESTS) | 32 | RW | 0000_0000h |
| 2A8h | Debug Mode Enable (DBGEN) | 32 | RW | 0000_0000h |

## 59.8.2 LC n Output m LUT Control (LC0_LUTCTRL0 - LC2_LUTCTRL3)

### Offset

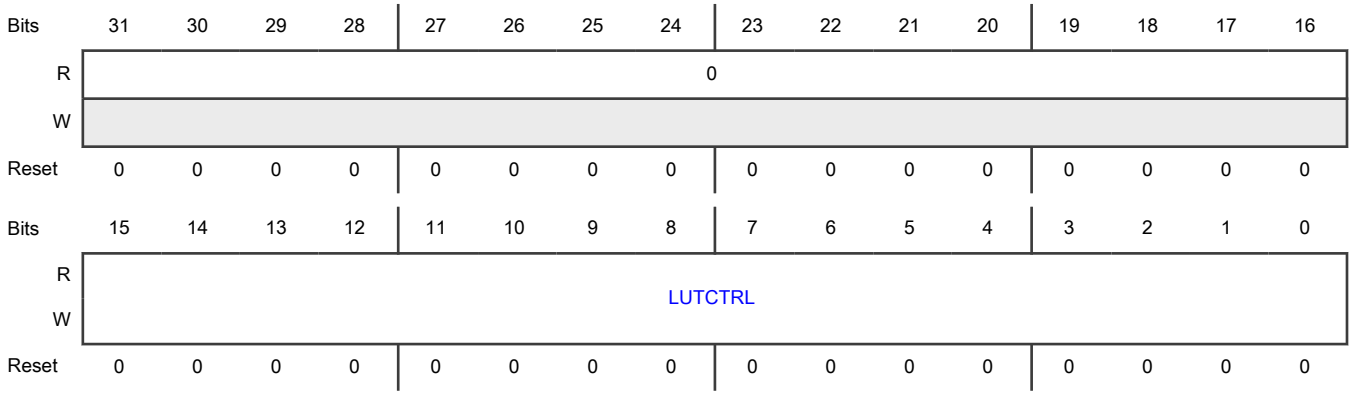
For n = 0 to 2; m = 0 to 3:

| Register | Offset |
|---|---|
| LCn_LUTCTRLm | 0h + (n × 40h) + (m × 4h) |

### Function

See LUTCTRL.

### Diagram



**Fields**

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>LUTCTRL | LUT Control<br>Specifies the LUT positions, based on the combined LC input value, that result in assertion of this output. For more information, see Logic operations. |

## 59.8.3 LC n Output m Filter (LC0_FILT0 - LC2_FILT3)

**Offset**

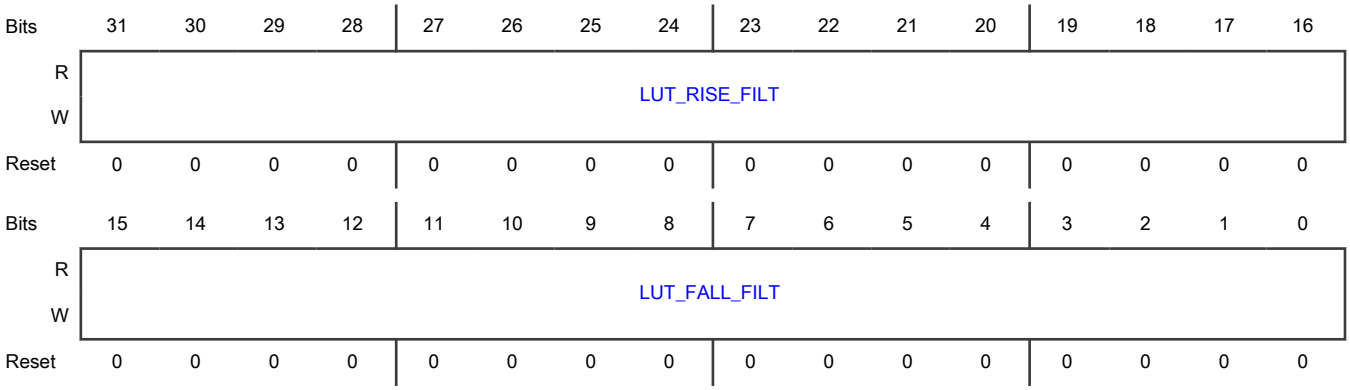
For n = 0 to 2; m = 0 to 3:

| Register | Offset |
|---|---|
| LCn_FILTm | 10h + (n × 40h) + (m × 4h) |

**Function**

Specifies the rising- and falling-edge thresholds for the LC output filters.

**Diagram**




**Fields**

| Field | Function |
|---|---|
| 31-16<br>LUT_RISE_FILT | Rise Filter<br>Specifies the number of consecutive clock cycles the filter output must be logic 1 before the output signal goes high.<br>0000_0000_0000_0000b - Bypass filter<br>All other values - Filter threshold |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 15-0<br>LUT_FALL_FILT | Fall Filter<br>Specifies the number of consecutive clock cycles the filter output must be logic 0 before the output signal goes low.<br>0000_0000_0000_0000b - Bypass filter<br>All other values - Filter threshold |

## 59.8.4 LC n Interrupt and DMA Enable (LC0_INTDMAEN - LC2_INTDMAEN)

### Offset

| Register | Offset |
|---|---|
| LC0_INTDMAEN | 20h |
| LC1_INTDMAEN | 60h |
| LC2_INTDMAEN | A0h |

### Function
Enables interrupt and DMA requests for LUT and force events.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24 | Force DMA Enable<br>Enables the generation of a DMA request when a force event occurs (LC*n*_STS[FORCESTS]). |

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>FORCE_DMA_EN</td><td>For each bit:<br>0b - Disable<br>1b - Enable<br>Mapping of bits:<table><tr><th>Output</th><th>Register bit</th></tr><tr><td>0</td><td>24</td></tr><tr><td>1</td><td>25</td></tr><tr><td>2</td><td>26</td></tr><tr><td>3</td><td>27</td></tr></table></td></tr>
<tr><td>23-20<br>—</td><td>Reserved</td></tr>
<tr><td>19-16<br>FORCE_INT_EN</td><td>Force Interrupt Enable<br>Enables the generation of an interrupt request when a force event occurs (LCn_STS[FORCESTS]).<br>For each bit:<br>0b - Disable<br>1b - Enable<br>Mapping of bits:<table><tr><th>Output</th><th>Register bit</th></tr><tr><td>0</td><td>16</td></tr><tr><td>1</td><td>17</td></tr><tr><td>2</td><td>18</td></tr><tr><td>3</td><td>19</td></tr></table></td></tr>
<tr><td>15-12<br>—</td><td>Reserved</td></tr>
<tr><td>11-8<br>LUT_DMA_EN</td><td>LUT DMA Enable<br>Enables the generation of a DMA request when an LUT event occurs (LCn_STS[LUT_STS]).<br>For each bit:<br>0b - Disable</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>1b - Enable<br>Mapping of bits:<table><tr><th>Output</th><th>Register bit</th></tr><tr><td>0</td><td>8</td></tr><tr><td>1</td><td>9</td></tr><tr><td>2</td><td>10</td></tr><tr><td>3</td><td>11</td></tr></table></td></tr>
<tr><td>7-4<br>—</td><td>Reserved</td></tr>
<tr><td>3-0<br>LUT_INT_EN</td><td>LUT Interrupt Enable<br>Enables the generation of an interrupt request when an LUT event occurs (LCn_STS[LUT_STS]).<br>For each bit:<br>0b - Disable<br>1b - Enable<br>Mapping of bits:<table><tr><th>Output</th><th>Register bit</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr><tr><td>2</td><td>2</td></tr><tr><td>3</td><td>3</td></tr></table></td></tr>
</table>

## 59.8.5 LC n Status (LC0_STS - LC2_STS)

### Offset

| Register | Offset |
|---|---|
| LC0_STS | 24h |
| LC1_STS | 64h |
| LC2_STS | A4h |

### Function

Indicates the occurrence of LUT and force events.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | FORCESTS | | | | 0 | | | | LUT_STS | | | |
| W | | | | | W1C | | | | | | | | W1C | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11-8<br>FORCESTS | Force Event<br>Indicates that a force event has occurred on the associated output.<br>For each bit:<br>0b - No event<br>1b - Event occurred<br>Mapping of bits: |

| Output | Register bit |
|---|---|
| 0 | 8 |
| 1 | 9 |
| 2 | 10 |
| 3 | 11 |

When you enable DMA for a force output (LC*n*_INTDMAEN[FORCE_DMA_EN]) and a force event occurs on that output, LCU generates a DMA request. The resulting DMA done signal causes LCU to change the bit to 0, or you can write 1 to the bit.

You can also change these bits to 0 by writing to the corresponding Force Status bits (FORCESTS[FORCESTS]).

The timing of status clearing depends on the selected Force Mode (LC*n*_FCTRL[FORCE_MODEm]).

| | |
|---|---|
| 7-4<br>— | Reserved |
| 3-0 | LUT Event |

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>LUT_STS</td><td>Indicates that an LUT event has occurred for the associated LC output.<br>For each bit:<br>0b - No event<br>1b - Event occurred<br>Mapping of bits:
<table>
<tr><th>Output</th><th>Register bit</th></tr>
<tr><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td></tr>
</table>
When you enable DMA for an LUT event on an output (LCn_INTDMAEN[LUT_DMA_EN]) and an LUT event occurs on that output, LCU generates a DMA request. The resulting DMA done signal causes LCU to change the bit to 0.<br>When you enable interrupt request for an output (LCn_INTDMAEN[LUT_INT_EN]) and an LUT event occurs on that output, LCU generates an interrupt request. Write 1 to the bit to change it to 0.</td></tr>
</table>

## 59.8.6 LC n Output Polarity Control (LC0_OUTPOL - LC2_OUTPOL)

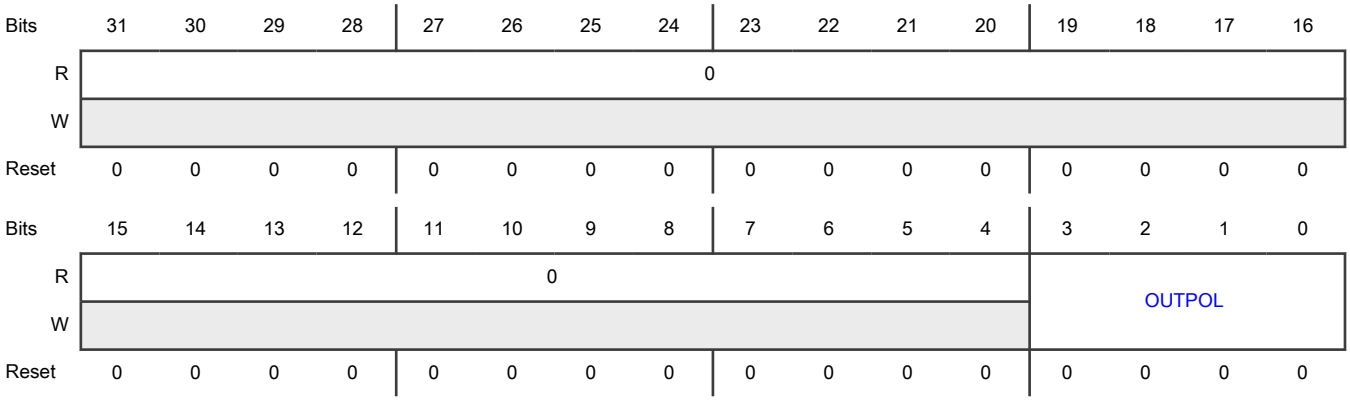
### Offset

| Register | Offset |
|---|---|
| LC0_OUTPOL | 28h |
| LC1_OUTPOL | 68h |
| LC2_OUTPOL | A8h |

### Function

Specifies the polarity of the LC outputs.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>OUTPOL | Output Polarity<br>Specifies the polarity of the outputs.<br>For each bit:<br>0b - Not inverted<br>1b - Inverted<br>Mapping of bits: |

| Output | Bit |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |

## 59.8.7 LC n Force Filter (LC0_FFILT - LC2_FFILT)

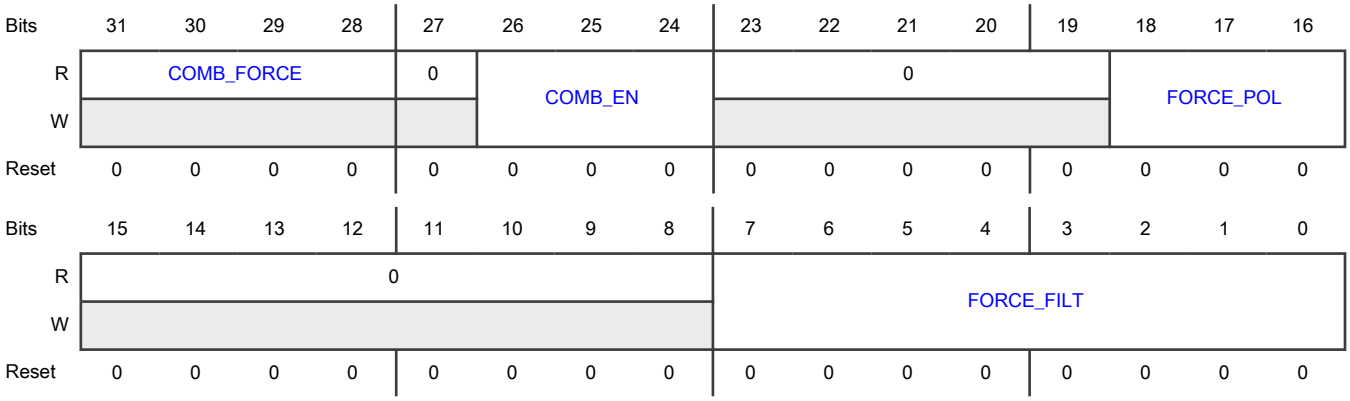
### Offset

| Register | Offset |
|---|---|
| LC0_FFILT | 2Ch |
| LC1_FFILT | 6Ch |
| LC2_FFILT | ACh |

### Function

Controls the force filter for this LC.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-28<br>COMB_FORCE | Combined Sensed Force Input<br>Indicates the combined value of force inputs to each output.<br>For each bit:<br>0b - Logic low<br>1b - Logic high<br>Mapping of bits: |

| Output | Bit |
|---|---|
| 0 | 28 |
| 1 | 29 |
| 2 | 30 |
| 3 | 31 |

| Field | Function |
|---|---|
| 27<br>— | Reserved |
| 26-24<br>COMB_EN | Combinational Force Path (CFP) Enable<br>Enables an active force input to combinationally affect the LC outputs. When CFP is not enabled, force inputs must be synchronized and then optionally filtered. When you enable CFP (write 1 to one of these bits), the corresponding force input bypasses synchronization and filtering and immediately affects the LC output.<br>The Force Filter (FORCE_FILT) is still in effect. Force inputs that do not meet the pulse width requirements for synchronization and filtering are not registered in Force Status (FORCESTS) and you do not need to write 1 to change the bit to 0. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | CFP provides a safety factor when the LC outputs drive a motor and immediate response is required. In some topologies, an oscillation can result if the assertion of the force input causes the LC output to turn off. This situation in turn leads to deassertion of the force input and the output turning back on. In general, you must disable CFP unless the LCU outputs drive a motor or similar safety-critical application, which requires an immediate response without synchronization delays. |
| 23-19<br>— | Reserved |
| 18-16<br>FORCE_POL | Force Input Polarity<br>Specifies the polarity of the force inputs to this LC.<br>For each bit:<br>0b - Not inverted<br>1b - Inverted<br>Mapping of bits: |

| Input | Bit |
|---|---|
| 0 | 16 |
| 1 | 17 |
| 2 | 18 |

| | |
|---|---|
| 15-8<br>— | Reserved |
| 7-0<br>FORCE_FILT | Force Filter<br>Specifies the count, in clock cycles, that a force input must remain at a given logic state before the filtered force input switches state.<br>0000_0000b - Bypass filter<br>All other values - Filter threshold |

## 59.8.8 LC n Force Control (LC0_FCTRL - LC2_FCTRL)

**Offset**

| Register | Offset |
|---|---|
| LC0_FCTRL | 30h |
| LC1_FCTRL | 70h |
| LC2_FCTRL | B0h |

### Function

Provides control of the force inputs.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SYNC_SEL3 | | FORCE_MODE3 | | FORCE_SENSE3 | | | | SYNC_SEL2 | | FORCE_MODE2 | | FORCE_SENSE2 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SYNC_SEL1 | | FORCE_MODE1 | | FORCE_SENSE1 | | | | SYNC_SEL0 | | FORCE_MODE0 | | FORCE_SENSE0 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-30<br>SYNC_SEL3 | Sync Select<br>Selects which sync input to use for output 3 of this LC.<br>00b - Sync input 0<br>01b - Sync input 1<br>10b - Reserved<br>11b - Reserved |
| 29-28<br>FORCE_MODE3 | Force Clearing Mode<br>Specifies the timing for clearing force events for output 3 in this LC.<br>00b - Deassertion. Cleared on deassertion of force inputs<br>01b - Rising sync after deassertion. Cleared on rising sync after deassertion of force inputs<br>10b - Writing 1 after deassertion. Cleared by writing 1 to the correct LCn_STS[FORCESTS] or FORCESTS[FORCESTS] bits after deassertion of force inputs<br>11b - Rising sync after writing 1 and deassertion. Cleared on rising sync after writing 1 to the correct LCn_STS[FORCESTS] or FORCESTS[FORCESTS] bits and deassertion of force inputs |
| 27-24<br>FORCE_SENSE3 | Force Input Sensitivity<br>Selects which force inputs affect output 3 of this LC.<br>For each bit:<br>0b - Does not affect<br>1b - Affects<br>Mapping of bits: |

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td><table>
<tr><th>Input</th><th>Bit</th></tr>
<tr><td>0</td><td>24</td></tr>
<tr><td>1</td><td>25</td></tr>
<tr><td>2</td><td>26</td></tr>
<tr><td>Reserved</td><td>27</td></tr>
</table>
Example: 011b selects force inputs 0 and 1, but not 2.</td></tr>
<tr><td>23-22<br>SYNC_SEL2</td><td>Sync Select<br>Selects which sync input to use for output 2 of this LC.<br>00b - Sync input 0<br>01b - Sync input 1<br>10b - Reserved<br>11b - Reserved</td></tr>
<tr><td>21-20<br>FORCE_MODE2</td><td>Force Clearing Mode<br>Specifies the timing for clearing force events for output 2 in this LC.<br>00b - Deassertion. Cleared on deassertion of force inputs<br>01b - Rising sync after deassertion. Cleared on rising sync after deassertion of force inputs<br>10b - Writing 1 after deassertion. Cleared by writing 1 to the correct LCn_STS[FORCESTS] or FORCESTS[FORCESTS] bits after deassertion of force inputs<br>11b - Rising sync after writing 1 and deassertion. Cleared on rising sync after writing 1 to the correct LCn_STS[FORCESTS] or FORCESTS[FORCESTS] bits and deassertion of force inputs</td></tr>
<tr><td>19-16<br>FORCE_SENSE2</td><td>Force Input Sensitivity<br>Selects which force inputs affect output 2 of this LC.<br>For each bit:<br>0b - Does not affect<br>1b - Affects<br>Mapping of bits:<table>
<tr><th>Input</th><th>Bit</th></tr>
<tr><td>0</td><td>16</td></tr>
<tr><td>1</td><td>17</td></tr>
<tr><td>2</td><td>18</td></tr>
</table></td></tr>
</table>

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td><table>
<tr><th>Input</th><th>Bit</th></tr>
<tr><td>Reserved</td><td>19</td></tr>
</table>
Example: 011b selects force inputs 0 and 1, but not 2.</td></tr>
<tr><td>15-14<br>SYNC_SEL1</td><td>Sync Select<br>Selects which sync input to use for output 1 of this LC.<br>00b - Sync input 0<br>01b - Sync input 1<br>10b - Reserved<br>11b - Reserved</td></tr>
<tr><td>13-12<br>FORCE_MODE1</td><td>Force Clearing Mode<br>Specifies the timing for clearing force events for output 1 in this LC.<br>00b - Deassertion. Cleared on deassertion of force inputs<br>01b - Rising sync after deassertion. Cleared on rising sync after deassertion of force inputs<br>10b - Writing 1 after deassertion. Cleared by writing 1 to the correct LCn_STS[FORCESTS] or FORCESTS[FORCESTS] bits after deassertion of force inputs<br>11b - Rising sync after writing 1 and deassertion. Cleared on rising sync after writing 1 to the correct LCn_STS[FORCESTS] or FORCESTS[FORCESTS] bits and deassertion of force inputs</td></tr>
<tr><td>11-8<br>FORCE_SENSE1</td><td>Force Input Sensitivity<br>Selects which force inputs affect output 1 of this LC.<br>For each bit:<br>0b - Does not affect<br>1b - Affects<br>Mapping of bits:<table>
<tr><th>Input</th><th>Bit</th></tr>
<tr><td>0</td><td>8</td></tr>
<tr><td>1</td><td>9</td></tr>
<tr><td>2</td><td>10</td></tr>
<tr><td>Reserved</td><td>11</td></tr>
</table>
Example: 011b selects force inputs 0 and 1, but not 2.</td></tr>
<tr><td>7-6</td><td>Sync Select</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>SYNC_SEL0</td><td>Selects which sync input to use for output 0 of this LC.<br>00b - Sync input 0<br>01b - Sync input 1<br>10b - Reserved<br>11b - Reserved</td></tr>
<tr><td>5-4<br>FORCE_MODE0</td><td>Force Clearing Mode<br>Specifies the timing for clearing force events for output 0 in this LC.<br>00b - Deassertion. Cleared on deassertion of force inputs<br>01b - Rising sync after deassertion. Cleared on rising sync after deassertion of force inputs<br>10b - Writing 1 after deassertion. Cleared by writing 1 to the correct LCn_STS[FORCESTS] or FORCESTS[FORCESTS] bits after deassertion of force inputs<br>11b - Rising sync after writing 1 and deassertion. Cleared on rising sync after writing 1 to the correct LCn_STS[FORCESTS] or FORCESTS[FORCESTS] bits and deassertion of force inputs</td></tr>
<tr><td>3-0<br>FORCE_SENSE0</td><td>Force Input Sensitivity<br>Selects which force inputs affect output 0 of this LC.<br>For each bit:<br>0b - Does not affect<br>1b - Affects<br>Mapping of bits:<br><table><tr><th>Input</th><th>Bit</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr><tr><td>2</td><td>2</td></tr><tr><td>Reserved</td><td>3</td></tr></table><br>Example: 011b selects force inputs 0 and 1, but not 2.</td></tr>
</table>

## 59.8.9 LC n Sync Control (LC0_SCTRL - LC2_SCTRL)

**Offset**

| Register | Offset |
|---|---|
| LC0_SCTRL | 34h |

*Table continues on the next page...*

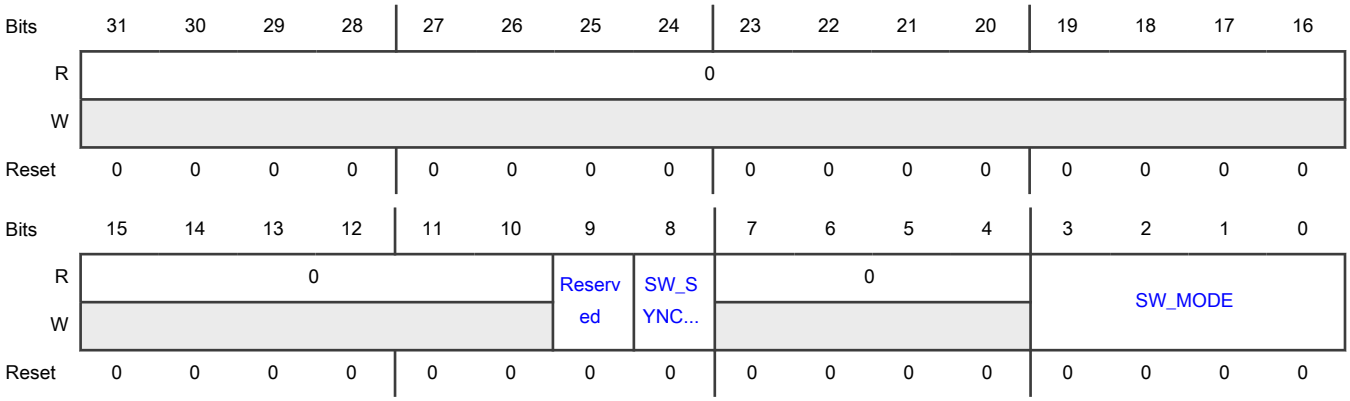
*Table continued from the previous page...*

| Register | Offset |
|---|---|
| LC1_SCTRL | 74h |
| LC2_SCTRL | B4h |

### Function

Controls the software sync behavior.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>— | Reserved |
| 8<br>SW_SYNC_SEL | Software Sync Select<br>Selects which sync input to use for software synced mode.<br>0b - Sync input 0<br>1b - Sync input 1 |
| 7-4<br>— | Reserved |
| 3-0<br>SW_MODE | Software Sync Mode<br>Specifies the software sync mode for the inputs to this LC. When Software Override is enabled (SWEN), these bits control whether Software Override Value (SWVALUE) changes occur immediately or on the rising edge of the selected sync pulse.<br>For each bit: |

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Immediate<br>1b - On rising edge of sync<br>Mapping of bits: |

| Input | Bit |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |

## 59.8.10 Mux Select (MUXSEL0 - MUXSEL11)

### Offset

For a = 0 to 11:

| Register | Offset |
|---|---|
| MUXSELa | 200h + (a × 4h) |

### Function

Selects the sources for inputs to the LCs.

**NOTE**

Access to Address space 200h + (48 to (124)) does not generate a transfer error.

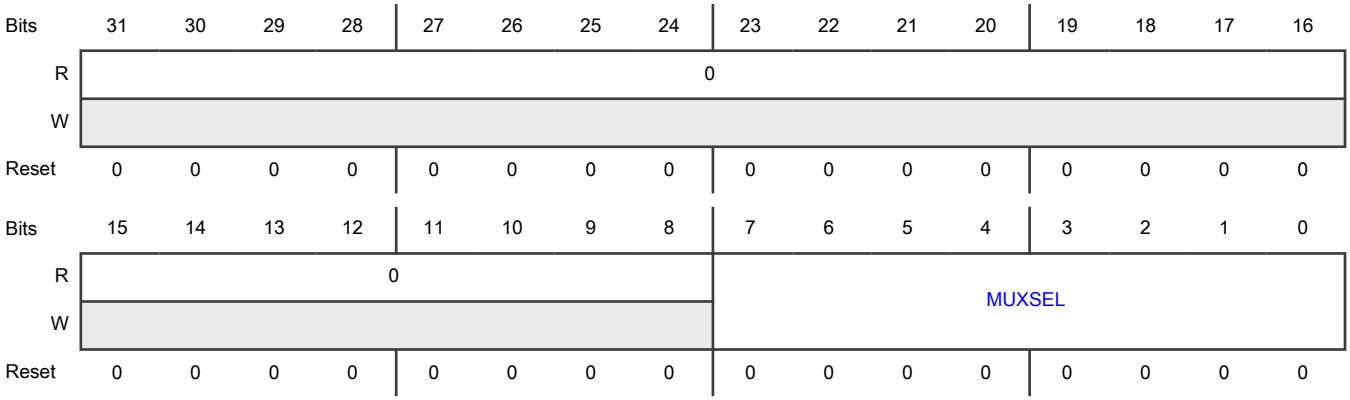
| Register | LC | Input |
|---|---|---|
| MUXSEL0 | 0 | 0 |
| MUXSEL1 | | 1 |
| MUXSEL2 | | 2 |
| MUXSEL3 | | 3 |
| MUXSEL4 | 1 | 0 |
| MUXSEL5 | | 1 |
| MUXSEL6 | | 2 |
| MUXSEL7 | | 3 |
| MUXSEL8 | 2 | 0 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Register | LC | Input |
|---|---|---|
| MUXSEL9 | | 1 |
| MUXSEL10 | | 2 |
| MUXSEL11 | | 3 |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>MUXSEL | Mux Select<br>Selects the source of the LC input.<br>All LU_IN inputs go through a two-stage synchronizer, but the LU_OUT signals fed back to this mux do not have this two-stage delay because they are already synchronous to LCU clock.<br>0000_0000b - Logic 0<br>0000_0001b-0000_1100b - LU_IN0 to LU_IN11<br>0000_1101b-0001_1000b - LU_OUT0 to LU_OUT11<br>All other values are reserved. |

## 59.8.11 Configuration (CFG)

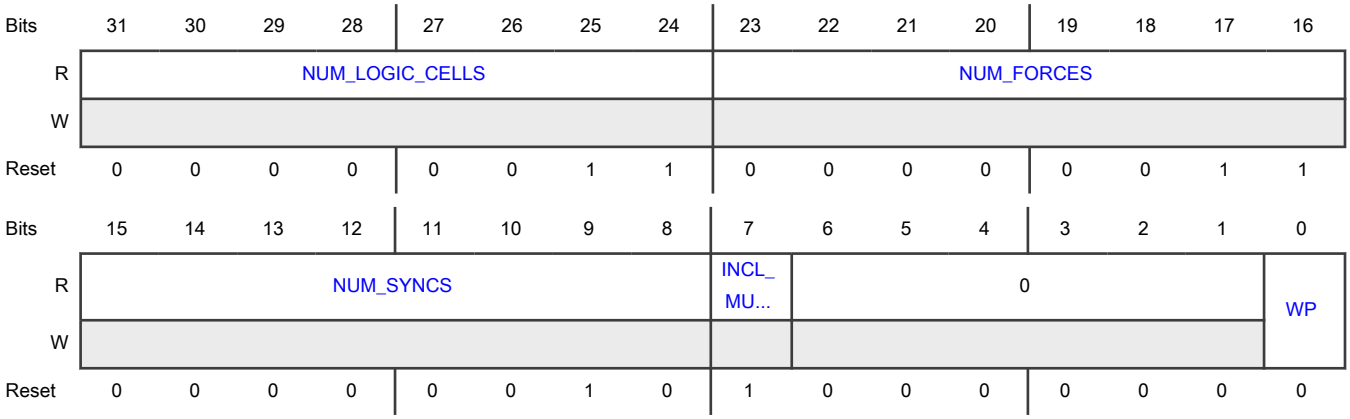
### Offset

| Register | Offset |
|---|---|
| CFG | 280h |

### Function

Indicates the LCU configuration, and provides write protection.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>NUM_LOGIC_CELLS | LCs<br>Indicates the number of LCs. |
| 23-16<br>NUM_FORCES | Force Inputs<br>Indicates the number of force inputs for each LC. |
| 15-8<br>NUM_SYNCS | Sync Inputs<br>Indicates the number of sync inputs for each LC. |
| 7<br>INCL_MUXES | Input Muxing<br>Indicates whether LCU supports input muxing.<br>0b - Not supported<br>1b - Supported |
| 6-1<br>— | Reserved |
| 0<br>WP | Write Protect<br>Turns on write protection for all LCU registers except SWVALUE, LC*n*_STS, and FORCESTS.<br>0b - No effect<br>1b - Turn on write protection |

## 59.8.12 Software Override Enable (SWEN)

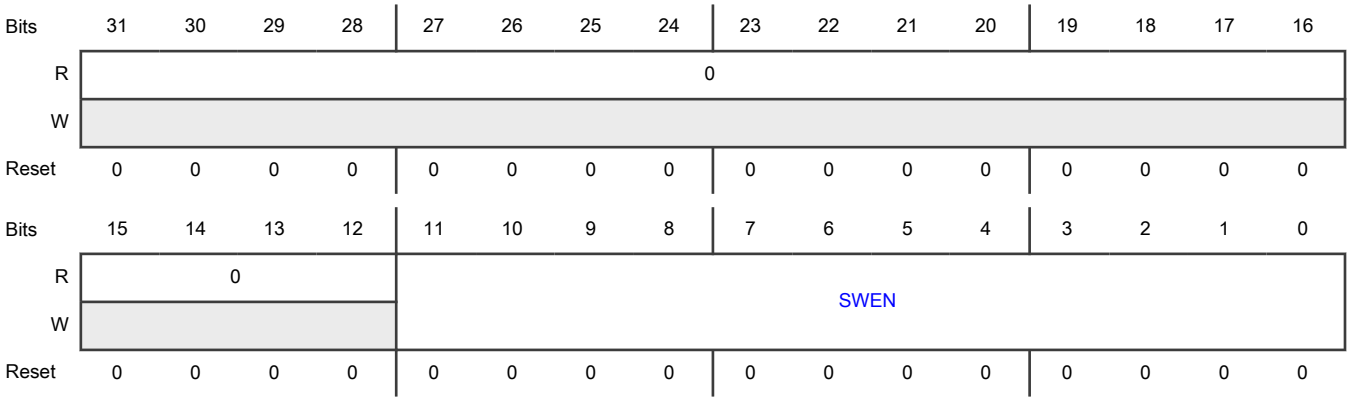
### Offset

| Register | Offset |
|---|---|
| SWEN | 284h |

### Function

Enables overrides for LC inputs.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11-0<br>SWEN | Software Override Enable<br>Enables software override of LC inputs.<br>For each bit:<br>0b - Disable<br>1b - Enable<br>Mapping of bits: See Mapping of 12-bit input, output, and state fields. |

## 59.8.13 Software Override Value (SWVALUE)

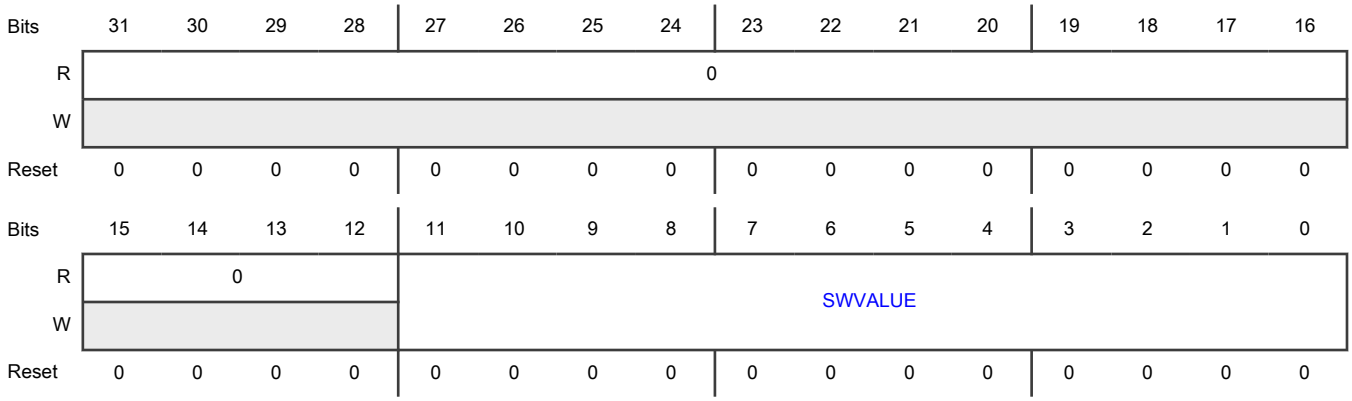
### Offset

| Register | Offset |
|---|---|
| SWVALUE | 288h |

### Function

Specifies the software override value for each LC input.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11-0<br>SWVALUE | Software Override Value<br>Specifies the software override value for each LC input.<br>For each bit:<br>0b - 0<br>1b - 1<br>Mapping of bits: See Mapping of 12-bit input, output, and state fields. |

## 59.8.14 Output Enable (OUTEN)

### Offset

| Register | Offset |
|---|---|
| OUTEN | 28Ch |

### Function

Enables LC outputs.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11-0<br>OUTEN | Output Enables<br>Enables LC outputs.<br>For each bit:<br>0b - Disable<br>1b - Enable<br>Mapping of bits: See Mapping of 12-bit input, output, and state fields. |

## 59.8.15 Logic Inputs (LCIN)

### Offset

| Register | Offset |
|---|---|
| LCIN | 290h |

### Function

Indicates states of LC inputs.

If you write to this register, LCU generates a bus transfer error.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11-0<br>LC_INPUTS | Logic Inputs<br>Indicates states of LC inputs.<br>For each bit:<br>0b - 0<br>1b - 1<br>Mapping of bits: See Mapping of 12-bit input, output, and state fields. |

## 59.8.16 Overridden Inputs (SWOUT)

### Offset

| Register | Offset |
|---|---|
| SWOUT | 294h |

### Function

Indicates states of LC inputs or states of software-overridden inputs, depending upon the state of the corresponding SWEN bit.

If you write to this register, LCU generates a bus transfer error.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11-0<br>SWOUT | Overridden Inputs<br>Indicates states of LC inputs or software-overridden inputs.<br>For each bit, when the corresponding SWEN bit is 0:<br>0b - LC input is 0<br>1b - LC input is 1<br>For each bit, when the corresponding SWEN bit is 1:<br>0b - Software-overridden input is 0<br>1b - Software-overridden input is 1<br>Mapping of bits: See Mapping of 12-bit input, output, and state fields. |

## 59.8.17 Logic Outputs (LCOUT)

### Offset

| Register | Offset |
|---|---|
| LCOUT | 298h |

### Function

Indicates states of LC outputs.

If you write to this register, LCU generates a bus transfer error.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11-0<br>LCOUT | Logic Outputs<br>Indicates states of LC outputs.<br>For each bit:<br>0b - 0<br>1b - 1<br>Mapping of bits: See Mapping of 12-bit input, output, and state fields. |

## 59.8.18 Forced Outputs (FORCEOUT)

### Offset

| Register | Offset |
|---|---|
| FORCEOUT | 29Ch |

### Function

Indicates the current state of the force outputs for all LCU logic outputs. Writing to this register generates a bus transfer error.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11-0<br>FORCEOUT | Forced Outputs<br>Indicates the current state of force outputs for the logic outputs.<br>For each bit:<br>0b - Not asserted<br>1b - Asserted<br>Mapping of bits: See Mapping of 12-bit input, output, and state fields. |

## 59.8.19 Force Status (FORCESTS)

### Offset

| Register | Offset |
|---|---|
| FORCESTS | 2A0h |

### Function

See FORCESTS.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | FORCESTS | | | | | | | | | | | |
| W | | | | | W1C | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11-0<br>FORCESTS | Force Status<br>Indicates the current force states of all LCs, mirrored from LC*n*_STS[FORCESTS]. Change these bits to 0 by writing 1 to them or to the corresponding bit in the specific LC*n*_STS register. This field allows you to simultaneously change FORCESTS bits to 0 across multiple LCs.<br>For each bit:<br>0b - Not in force state<br>1b - Read: In force state — Write: Clear force state bit<br>Mapping of bits: See Mapping of 12-bit input, output, and state fields. |

## 59.8.20 Debug Mode Enable (DBGEN)

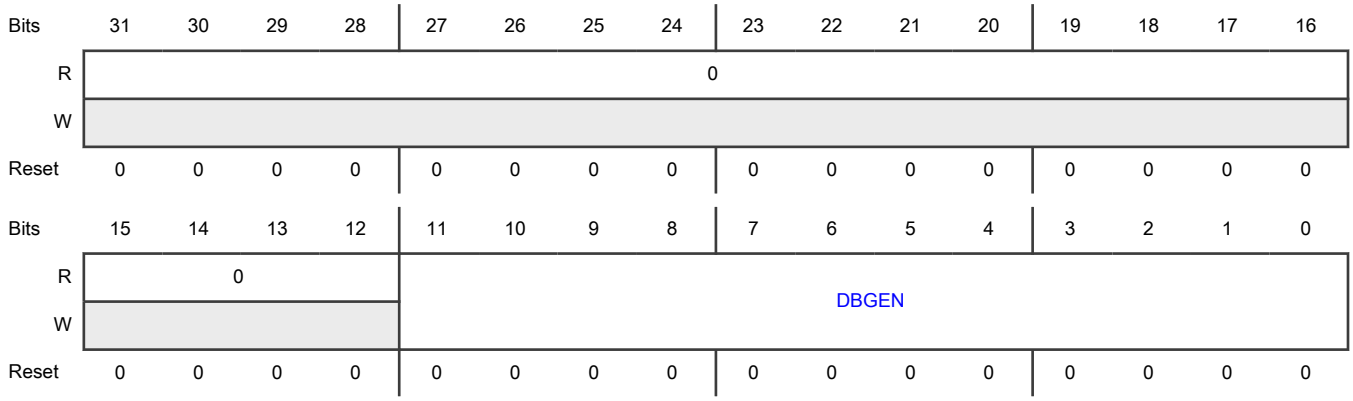
### Offset

| Register | Offset |
|---|---|
| DBGEN | 2A8h |

### Function

Enables outputs to continue operation when the chip is in Debug mode.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved |
| 11-0<br>DBGEN | Debug Mode Enable<br>Enables outputs to continue operation in Debug mode.<br>For each bit:<br>0b - Debug mode inactive<br>1b - Debug mode active<br>Mapping of bits: See Mapping of 12-bit input, output, and state fields. |

# Chapter 60
# Enhanced Modular IO Subsystem (eMIOS)

## 60.1 Chip-specific eMIOS information

### 60.1.1 eMIOS configuration

This chip has up to 3 instances of eMIOS that are each configured as shown in the following tables.

**Table 325. eMIOS instances**

| Instance | MCXE31B | MCXE315/MCXE316/ MCXE317 |
|---|---|---|
| eMIOS_0 | Yes | Yes |
| eMIOS_1 | Yes | Yes |
| eMIOS_2 | Yes | No |

**Table 326. eMIOS configuration**

| Parameter | eMIOS_0 | eMIOS_1 | eMIOS_2 |
|---|---|---|---|
| Timer width | 16 bits | 16 bits | 16 bits |
| Number of channels | 24 | 24 | 24 |
| Channels configuration | See EMIOS channel configuration figure below | | |
| Local Channel prescaler width | 4 | 4 | 4 |
| Number of global counter buses | 2 | 2 | 2 |
| Number of global prescaler | 1 | 1 | 1 |
| Global channel prescaler width | 8 | 8 | 8 |

**NOTE**

eMIOS is clocked by CORE_CLK.

### 60.1.2 Channel types

The eMIOS contain 4 different types of channel, which are listed below.

**Table 327. eMIOS channel types**

| Mode Description | Name | Ch TypeX | Ch TypeY | Ch TypeG | Ch TypeH |
|---|---|---|---|---|---|
| General Purpose Input / Output | GPIO | X | X | X | X |
| Single Action Input Capture | SAIC | X | X | X | X |
| Single Action Output Compare | SAOC | X | X | X | X |

*Table continues on the next page...*

Table 327. eMIOS channel types (continued)

| Mode Description | Name | Ch TypeX | Ch TypeY | Ch TypeG | Ch TypeH |
|---|---|---|---|---|---|
| Modulus Counter | MC | X | | | |
| Modulus Counter Buffered (Up / Down) | MCB | X | | X | |
| Input Pulse Width Measurement | IPWM | | | X | X |
| Input Period Measurement | IPM | | | X | X |
| Double Action Output Compare | DAOC | | | X | X |
| Output Pulse Width and Frequency Modulation Buffered | OPWFMB | X | | X | |
| Center aligned Output PWM Buffered with dead time | OPWMCB | | | X | |
| Output Pulse Width Modulation Buffered | OPWMB | X | X | X | X |
| Output Pulse Width Modulation Trigger | OPWMT | X | X | X | X |
| Pulse Edge Counting | PEC | | | X | |

**NOTE**

ChType X and G has internal counters.

Table 328. eMIOS channel configuration

| Channel number | eMIOS0 | eMIOS1 | eMIOS2 |
|---|---|---|---|
| CH0 | X | X | X |
| CH8 | X | X | X |
| CH16 | X | X | X |
| CH22 | X | X | X |
| CH23 | X | X | X |
| CH1 | G | H | H |
| CH2 | G | H | H |
| CH3 | G | H | H |

*Table continues on the next page...*

Table 328. eMIOS channel configuration (continued)

| | | | |
|---|---|---|---|
| CH4 | G | H | H |
| CH5 | G | H | H |
| CH6 | G | H | H |
| CH7 | G | H | H |
| CH9 | H | H | H |
| CH10 | H | H | H |
| CH11 | H | H | H |
| CH12 | H | H | H |
| CH13 | H | H | H |
| CH14 | H | H | H |
| CH15 | H | H | H |
| CH17 | Y | Y | Y |
| CH18 | Y | Y | Y |
| CH19 | Y | Y | Y |
| CH20 | Y | Y | Y |
| CH21 | Y | Y | Y |

### 60.1.3 Channel configuration

The eMIOS channel configuration with respect to Type and Counter_bus is shown below.

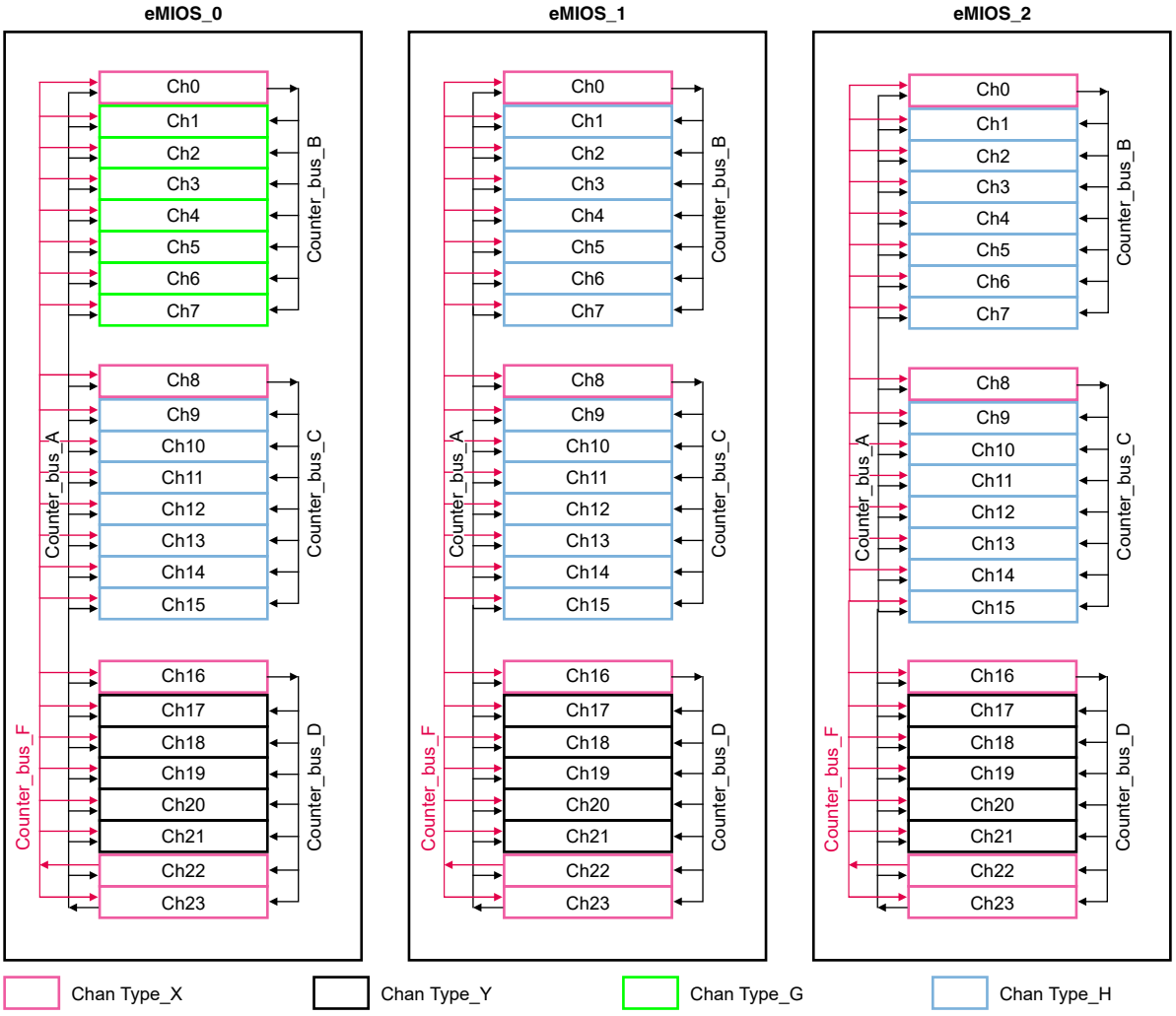



Figure 293. EMIOS channel types

## 60.1.4 eMIOS disabling

eMIOS can disable it's output using it's disable inputs. Disable inputs[3:0] are driven from the flag out bits[11:8].

| Disable inputs | |
|---|---|
| eMIOS0 Output disable input[3:0] | trgmux_output[31:28] |
| eMIOS1 Output disable input[3:0] | trgmux_output[31:28] |
| eMIOS2 Output disable input[3:0] | trgmux_output[31:28] |

For details, see the TRGMUX connectivity file attached to this document.

## 60.1.5 BCTU Interface

Each eMIOS channel, with the exception of the main counterbus channel (ch[23]) is capable of triggering the BCTU. See BCTU chapter for mapping. BCTU allows triggering from as many eMIOS modes as possible.

## 60.2 Introduction

eMIOS provides independent channels, UCs, that you configure to generate or measure time events for different functions in different chip applications.

eMIOS distributes these channels across a number of global and local counter buses. Each local bus is dedicated to a group of eight contiguous channels. Each channel can generate its own timebase, and each counter bus has its timebase provided by a dedicated channel. For a list of counter buses with their assigned channels and timebase sources, see Counter buses, channels, and timebase sources.

## 60.3 Block diagram

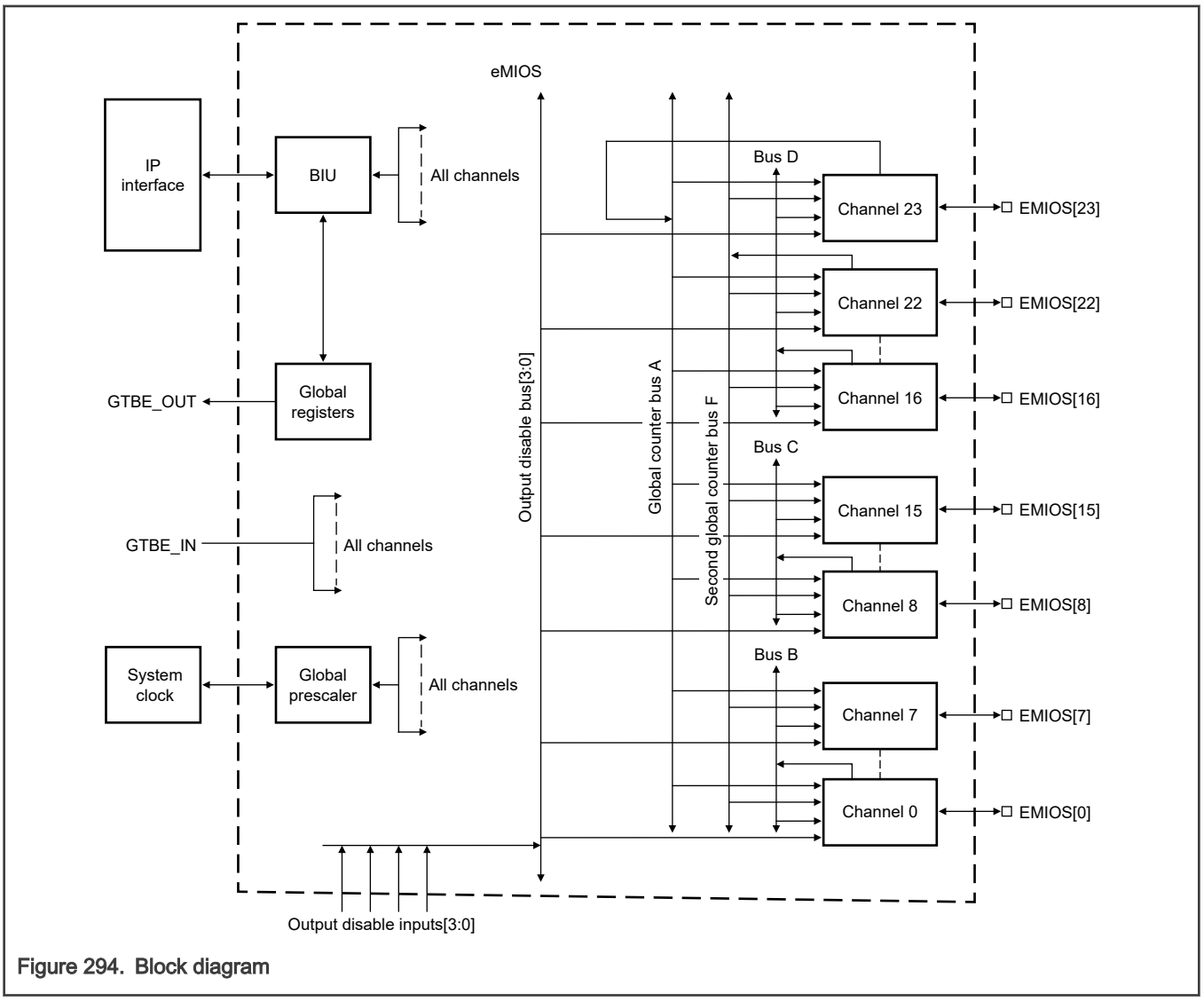



Figure 294. Block diagram

## 60.4 Features

- 24 UCs distributed across local counter buses as shown in Counter buses, channels, and timebase sources
- Global counter bus A driven by UC23
- Global counter bus F driven by UC22
- Synchronized timebases shared through the counter buses

- Dedicated timebase for each channel, distinct from the counter buses
- Global clock prescaler (GCP)
- One CP per channel
- Dedicated set of control and status registers for each UC
- 16-bit–wide data registers
- Shadow flag register (GFLAG) to access all channel flags with one read access
- Ability to freeze the UC state for debug purposes
- Motor control capability

## 60.5 Functional description

### 60.5.1 Reset

eMIOS resets asynchronously. Reset affects all registers.

On reset, UCs enter GPIO input mode.

### 60.5.2 Counter buses, channels, and timebase sources

**Table 329. Counter buses, channels, and timebase sources**

| Counter bus | Channels | Timebase channel |
|---|---|---|
| Global bus A | All UCs | 23 |
| Local bus B | • UC0<br>• UC1<br>• UC2<br>• UC3<br>• UC4<br>• UC5<br>• UC6<br>• UC7 | 0 |
| Local bus C | • UC8<br>• UC9<br>• UC10<br>• UC11<br>• UC12<br>• UC13<br>• UC14<br>• UC15 | 8 |
| Local bus D | • UC16 | 16 |

*Table continues on the next page...*

**Table 329. Counter buses, channels, and timebase sources (continued)**

| Counter bus | Channels | Timebase channel |
|---|---|---|
| | • UC17<br>• UC18<br>• UC19<br>• UC10<br>• UC21<br>• UC22<br>• UC23 | |
| Global bus F | All UCs | 22 |

## 60.5.3 Unified channels (UC)

### 60.5.3.1 Overview

Each UC contains the following:

- Two double-buffered data registers, A*n* and B*n*, that allow up to two events—input capture, output compare, or both—to occur before software intervention is needed
- Two comparators, A and B, that indicate when the selected counter bus is equal to the value in A*n* and B*n*
- An internal counter (CNT*n*[C]) that runs in all modes except GPIO.

  You can use this counter as a local timebase or to count input events. You can use it as a timebase if CNT*n*[C] is not used in the current mode.
- An output flip-flop that holds the logic level to be applied to the output pin
- A status register, UC Status n (S0 - S23), that flags input capture and match events, indicates flag overruns and overflows, and shows the input and output pin states
- A control register, UC Control n (C0 - C23), that controls UC operation

You control the following operational characteristics for each UC:

- The logic (C*n*[MODE]) that specifies the behavior of the UC (see UC modes)
- The counter bus (C*n*[BSL]) that provides the timebase for all timing functions
- The CP (C*n*[UCPRE] and C2_*n*[UCEXTPRE])
- The minimum input pulse width (C*n*[IF]) for the input filter that determines valid pin transitions
- Which input edges to detect (C*n*[EDSEL])
- Which of four output disable input signals (C*n*[ODISSL]) to use for disabling outputs
- For all output modes except GPIO, whether to disable the output flip-flop (C*n*[ODIS])

UC block diagram illustrates the basic UC architecture.

For a list of counter buses and their assigned channels and timebase sources, see Counter buses, channels, and timebase sources.

### 60.5.3.2 UC block diagram

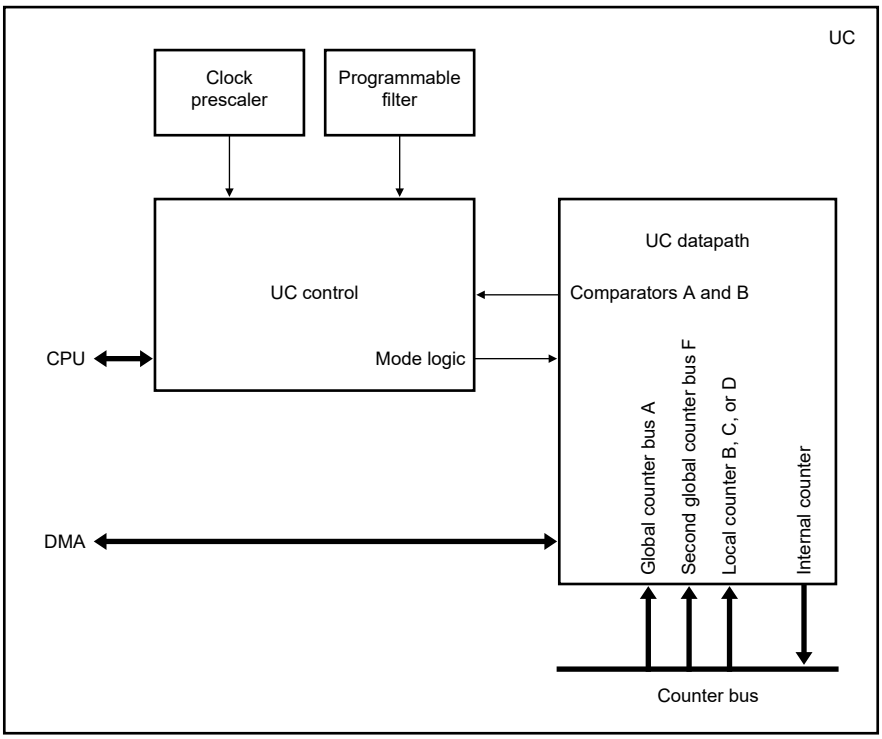



Figure 295. UC block diagram

### 60.5.3.3 Buffered modes

To provide smooth waveform generation even when A*n*[A] and B*n*[B] are changing, the following modes double buffer the A*n* and B*n* registers:

- Modulus Counter Buffered (MCB) mode
- Output Pulse Width and Frequency Modulation Buffered (OPWFMB) mode
- Output PWM Buffered (OPWMB) mode

These modes appear in separate sections because there are several basic differences in UC operation between them and their corresponding nonbuffered modes.

### 60.5.3.4 UC control and datapath

As illustrated in UC control and datapath diagram, the UC contains control and datapath sub-blocks.

The control sub-block generates signals to control the multiplexers in the datapath sub-block. eMIOS implements each UC mode in dedicated logic independent from the other modes. The UC modes share a set of registers allowing the storing of sequential events.

The datapath block contains the channel A and B registers, the internal timebase, and the comparators. Multiplexers receive inputs from the control sub-block to configure the datapath for the specific channel modes. This configuration consists of selecting the input of comparators and data for the register inputs. The outputs of the A and B comparators connect to the control block.

### 60.5.3.5 UC control and datapath diagram

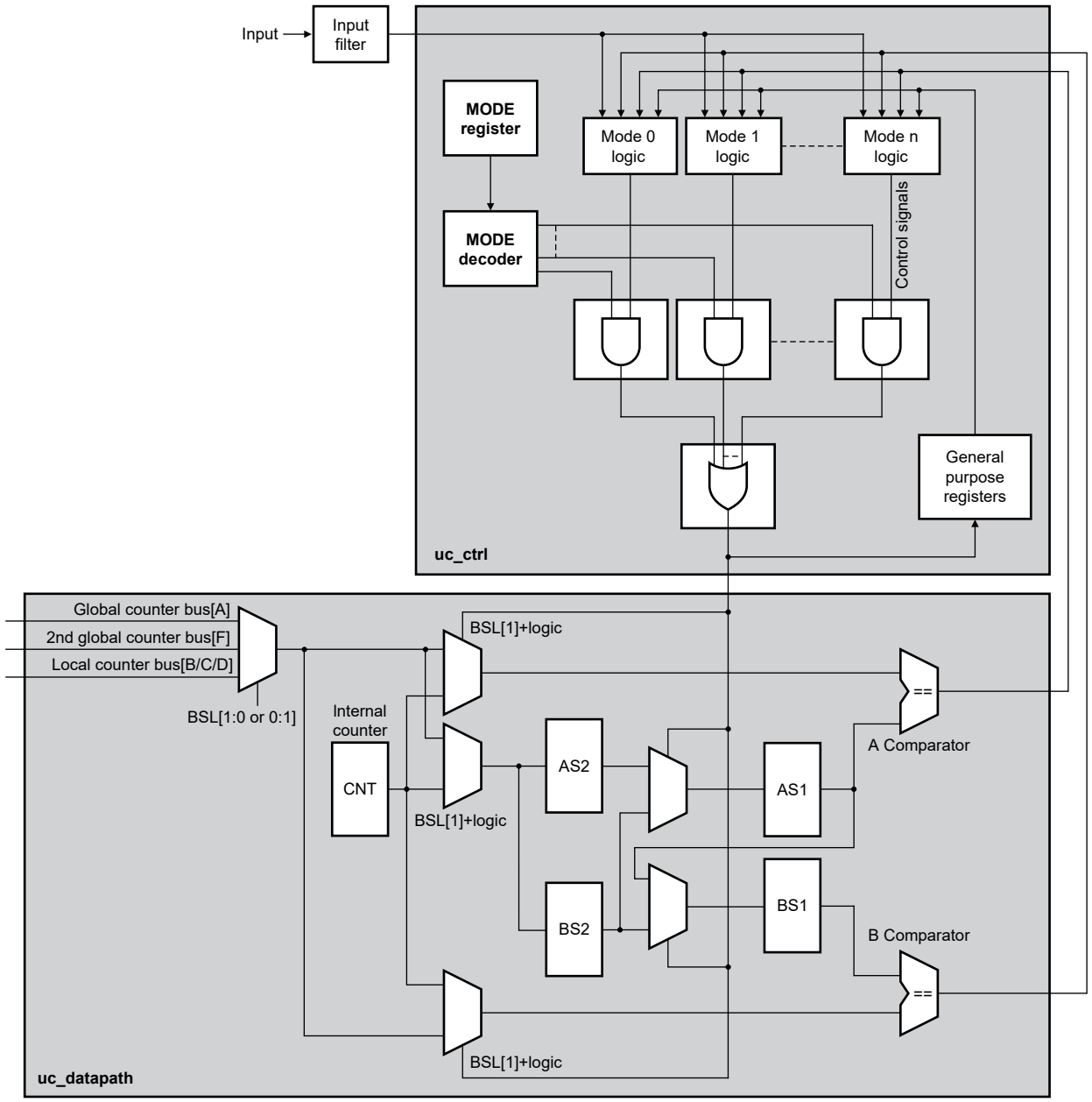



Figure 296. UC control and datapath diagram

### 60.5.3.6 UC modes

You can configure UCs to operate in the following modes:

- General-Purpose Input and Output (GPIO) mode
- Single Action Input Capture (SAIC) mode
- Single Action Output Capture (SAOC) mode
- Input Pulse Width Measurement (IPWM) mode
- Input Period Measurement (IPM) Mode

- Double Action Output Compare (DAOC) mode
- Pulse Edge Counting (PEC) mode
- Modulus Counter (MC) mode
- Modulus Counter Buffered (MCB) mode
- Output Pulse Width and Frequency Modulation Buffered (OPWFMB) mode
- Center Aligned Output PWM with Dead Time Insertion Buffered (OPWMCB) mode
- Output PWM Buffered (OPWMB) mode
- Output PWM with Trigger (OPWMT) mode

Each channel supports a specific subset of these modes. See the chip-configuration information to see which modes each channel supports.

### 60.5.3.7 General-Purpose Input and Output (GPIO) mode

#### 60.5.3.7.1 Overview

This mode disables all UC input-capture and output-compare functions, and resets and disables the internal counter (C*n*[C]). All control fields remain accessible.

To change the UC from one operation mode to another, you must first change to GPIO mode and then change to the desired final mode. To prepare the UC for a new operation mode, you must configure AS1, BS1, AS2, and BS2 before exiting GPIO mode and entering the desired final mode (see AS1, AS2, BS1, and BS2 shadow registers). Specifically:

- Writing to A*n*[A] stores the same value in AS1 and AS2.
- Writing to B*n*[B] stores the same value in BS1 and BS2.
- Writing to ALTA*n*[ALTA] stores a value only in AS2.

As its name implies, GPIO mode supports both input and output submodes. See Differences for input and output submodes.

#### 60.5.3.7.2 Differences for input and output submodes

**Table 330. Differences for input and output submodes**

| Submode | To enter submode, write this value to C*n*[MODE] | Characteristics |
|---|---|---|
| Input | 0 | • You control the generation of input-capture flags with C*n*[EDPOL] and C*n*[EDSEL].<br>• You can determine the input pin status by reading S*n*[UCIN]. |
| Output | 1 | • The UC performs as a single output port pin.<br>• The value of C*n*[EDPOL] permanently drives the output flip-flop. |

### 60.5.3.8 Single Action Input Capture (SAIC) mode

#### 60.5.3.8.1 Overview

In this mode, when a triggering event occurs on the input pin, eMIOS:

- Captures the selected timebase into AS2
- Changes S*n*[FLAG] to 1 to indicate that an input capture has occurred

A*n*[A] returns the value of AS2. As soon as the UC enters SAIC mode after exiting GPIO mode, the channel is ready to capture events.

eMIOS captures the events as soon as they occur. Therefore, reading A*n*[A] always returns the value of the latest captured event. Subsequent capture events occur regardless of whether you read the previous event from A*n*[A]. Each new capture event sets the input capture flag (writes 1 to S*n*[FLAG]).

The input capture is triggered by an edge transition on the input pin as configured by C*n*[EDPOL] and C*n*[EDSEL].

SAIC with rising edge triggering and SAIC with both edges triggering show how to use the UC for input capture.

#### 60.5.3.8.2 SAIC submodes and MODE field values

**Table 331. SAIC submodes and MODE field values**

| Capture behavior | C*n*[MODE] (binary) |
|---|---|
| The UC only captures the timestamp. | 000_0010 |
| • The UC captures a timestamp.<br>• AS2 bit 31 indicates the input signal level. | 100_0010 |

#### 60.5.3.8.3 Effect of C*n*[EDPOL] on edge capturing

**Table 332. Effect of C*n*[EDPOL] on edge capturing**

| C*n*[EDPOL] | Input signal transition | Resulting value of AS2 bit 31 |
|---|---|---|
| 0 | 0→1 | 0 |
| | 1→0 | 1 |
| 1 | 0→1 | 1 |
| | 1→0 | 0 |

#### 60.5.3.8.4 SAIC with rising edge triggering

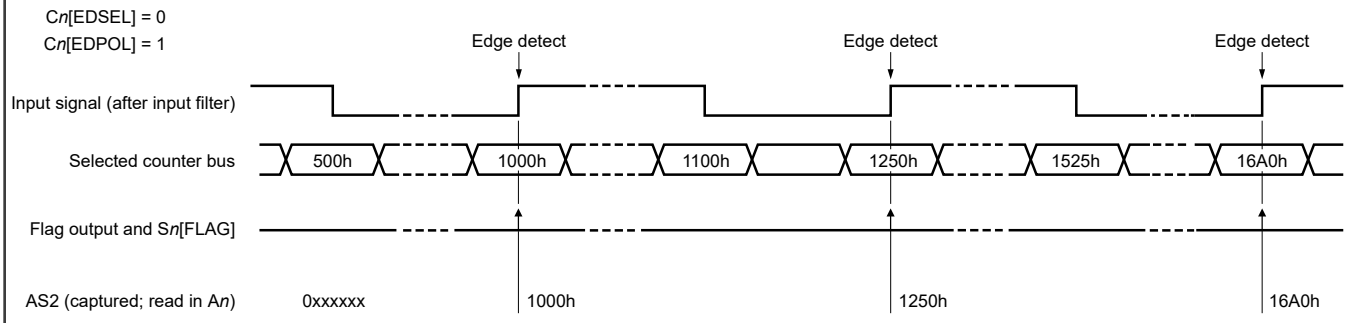



**Figure 297. SAIC with rising edge triggering**

#### 60.5.3.8.5 SAIC with both edges triggering

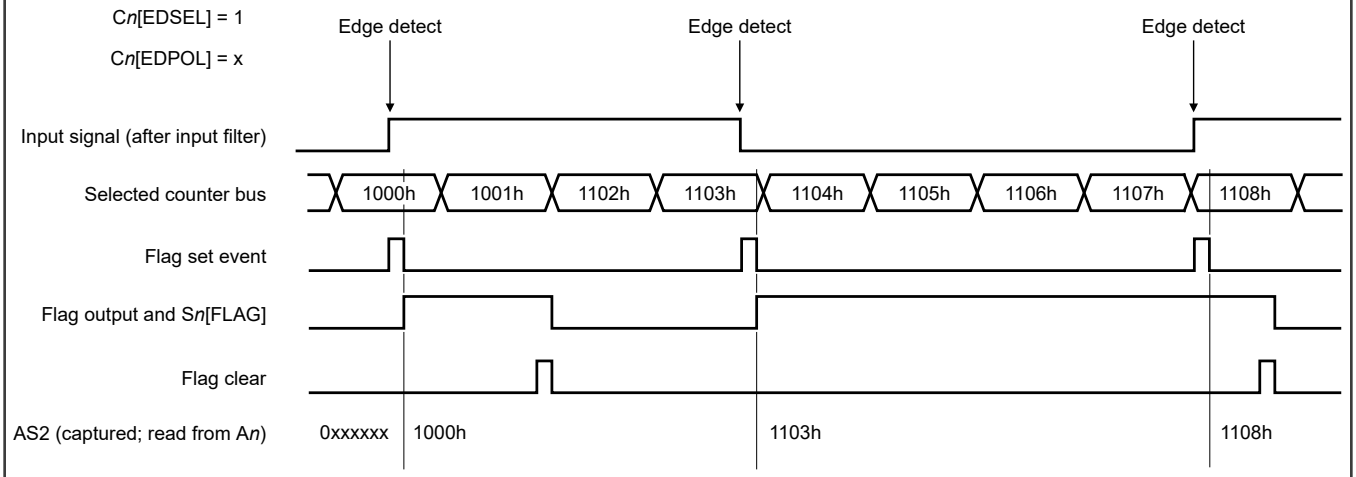



**Figure 298. SAIC with both edges triggering**

### 60.5.3.9 Single Action Output Capture (SAOC) mode

#### 60.5.3.9.1 Overview

In this mode, eMIOS loads a match value into AS2 and then immediately transfers it to AS1 for comparison with the selected timebase. When an output-compare match occurs, eMIOS:

- Either toggles the output flip-flop or transfers C*n*[EDPOL] to the output flip-flop, as determined by C*n*[EDSEL].
- Sets the input capture flag (changes S*n*[FLAG] to 1).

Along with the match, S*n*[FLAG] becomes 1 to indicate that the output-compare match has occurred. Writing to A*n*[A] stores the value in AS2, and reading A*n*[A] returns the AS1 value.

The channel internal counter in SAOC mode is free-running. It starts counting as soon as the UC enters SAOC mode.

You can force an output-compare match by writing 1 to C*n*[FORCMA]. A forced output-compare match does not set the input capture flag (S*n*[FLAG]).

When the UC enters SAOC mode after exiting GPIO mode, the output flip-flop value becomes the complement of C*n*[EDPOL].

You select the internal or external counter bus with C*n*[BSL].

SAOC with C*n*[EDPOL] transferred to the output flip-flop shows how to perform a single output compare and transfer C*n*[EDPOL] to the output flip-flop.

SAOC toggling the output flip-flop shows how to perform a single output compare and toggle the output flip-flop at each match.

#### 60.5.3.9.2 Preload the desired match value

eMIOS enables matches immediately after the UC enters SAOC mode. Therefore, you must write the desired match value to AS1 before the UC enters SAOC mode. You can update AS1 at any time. This modifies the match value reflected in the channel-generated output signal. Subsequent capture events occur regardless of whether you write to A*n*[A] again. Each new capture event sets the input capture flag (S*n*[FLAG]). See SAOC with flag behavior.

### 60.5.3.9.3 SAOC with C*n*[EDPOL] transferred to the output flip-flop

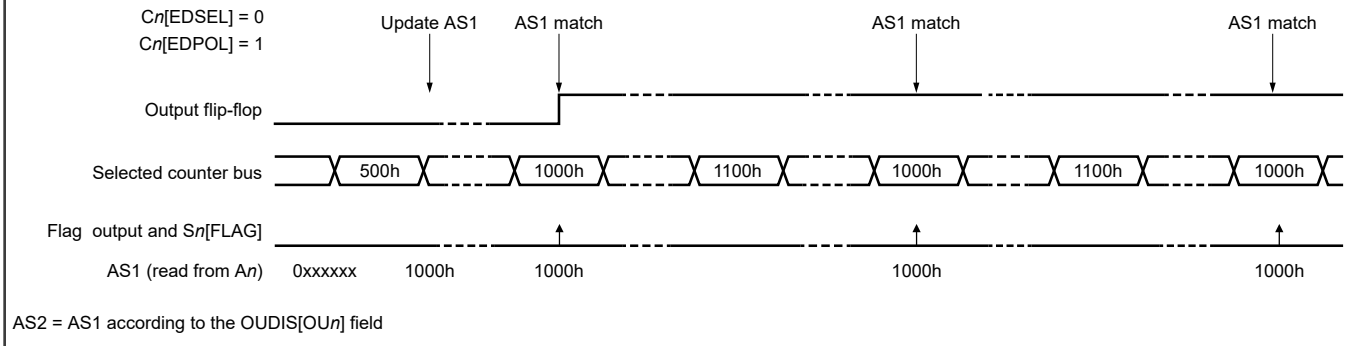



**Figure 299. SAOC with C*n*[EDPOL] transferred to the output flip-flop**

### 60.5.3.9.4 SAOC toggling the output flip-flop

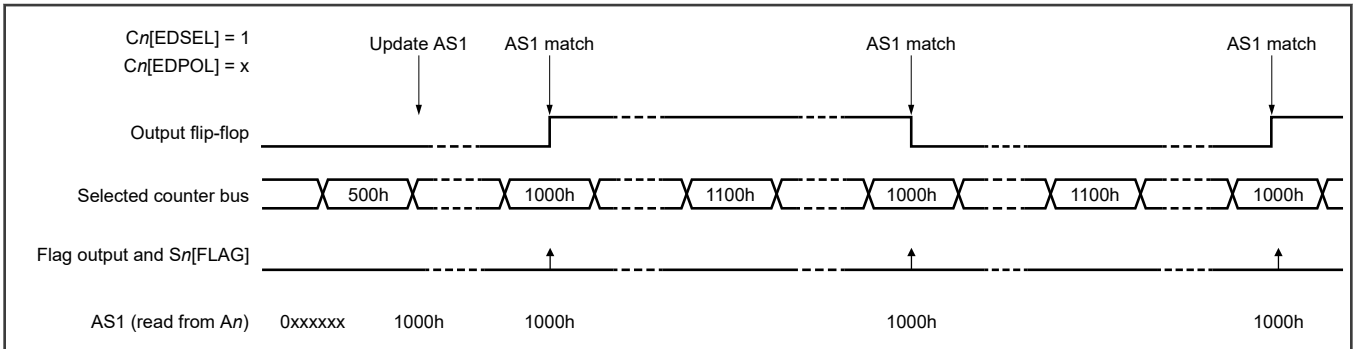



**Figure 300. SAOC toggling the output flip-flop**

### 60.5.3.9.5 SAOC with flag behavior

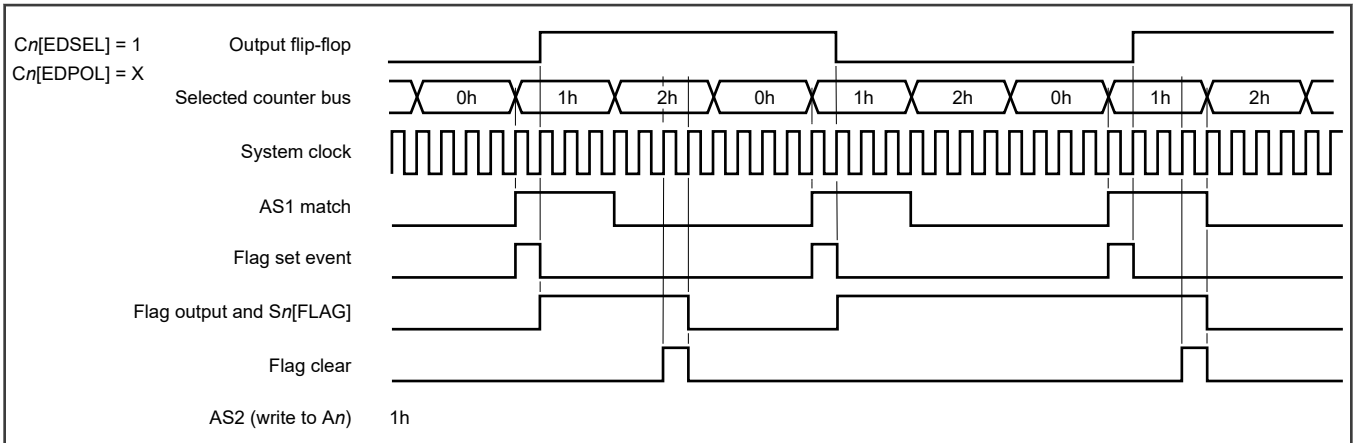



**Figure 301. SAOC with flag behavior**

### 60.5.3.10 Input Pulse Width Measurement (IPWM) mode

#### 60.5.3.10.1 Overview

This mode allows you to measure the width of a positive or negative pulse. Select whether to measure from the rising or falling edge with Edge Polarity (Cn[EDPOL]).

The first leading edge:

- Triggers the first input capture, which latches the count value of the selected timebase into BS2 (see AS1, AS2, BS1, and BS2 shadow registers)
- Does not set the input-capture flag (Sn[FLAG])
- Enables AS2 capture

The next trailing edge:

- Latches the selected timebase into AS2
- Sets the input-capture flag (Sn[FLAG])
- Transfers BS2 to BS1 and AS1

eMIOS captures successive values on consecutive edges of opposite polarity. If you initiate subsequent input capture events while the input-capture flag (Sn[FLAG]) is set, eMIOS updates AS2, BS1, and AS1 with the latest captured values and keeps the flag set.

#### 60.5.3.10.2 Coherency

Reading A*n*[A] updates BS1 with AS1. It also disables transfers from BS2 and BS1 until the next read of B*n*[B].

Reading Bn updates BS1 with AS1. It also re-enables transfers from BS2 to BS1, which takes effect at the next trailing edge capture.

Because transfers from BS2 to AS1 are never blocked, to guarantee data coherency you must read An, then read Bn.

If you do not require coherent data, you should read Bn before you read An, although doing so requires a second read of Bn to unblock BS1 updates.

#### 60.5.3.10.3 Timing example: measure pulse width

This figure shows how to measure the input pulse width by subtracting BS1 from AS2.

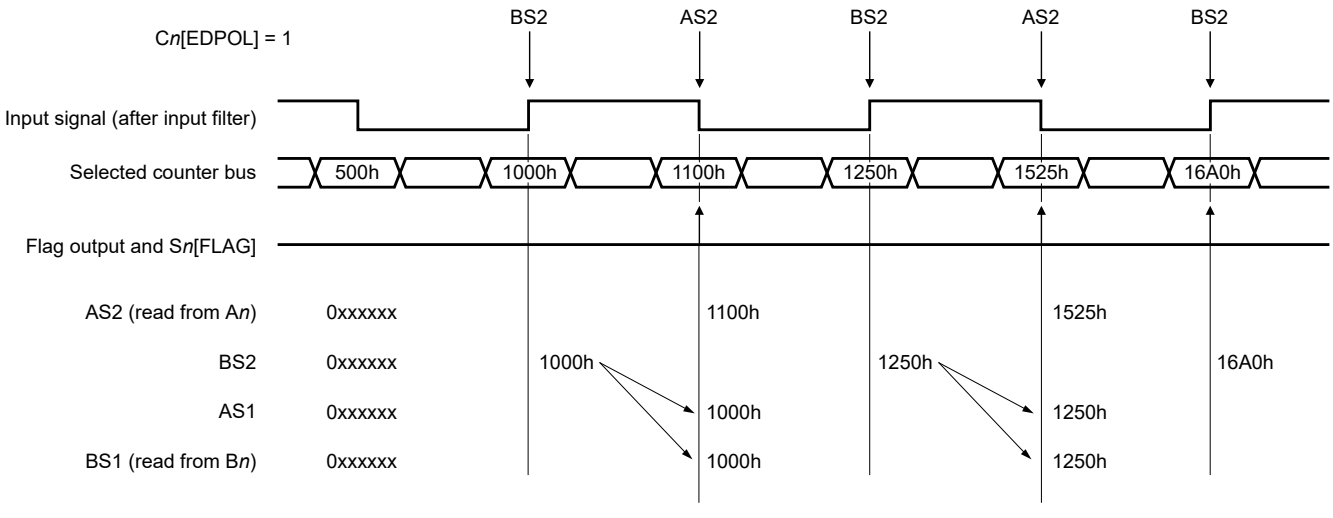



**Figure 302. Timing example: measure pulse width**

#### 60.5.3.10.4 Timing example: AS1 and BS1 updates on channel reads

This example illustrates AS1 and BS1 updates when you read A*n*[A] and B*n*[B]. AS1 always has coherent data related to AS2 (see Coherency).

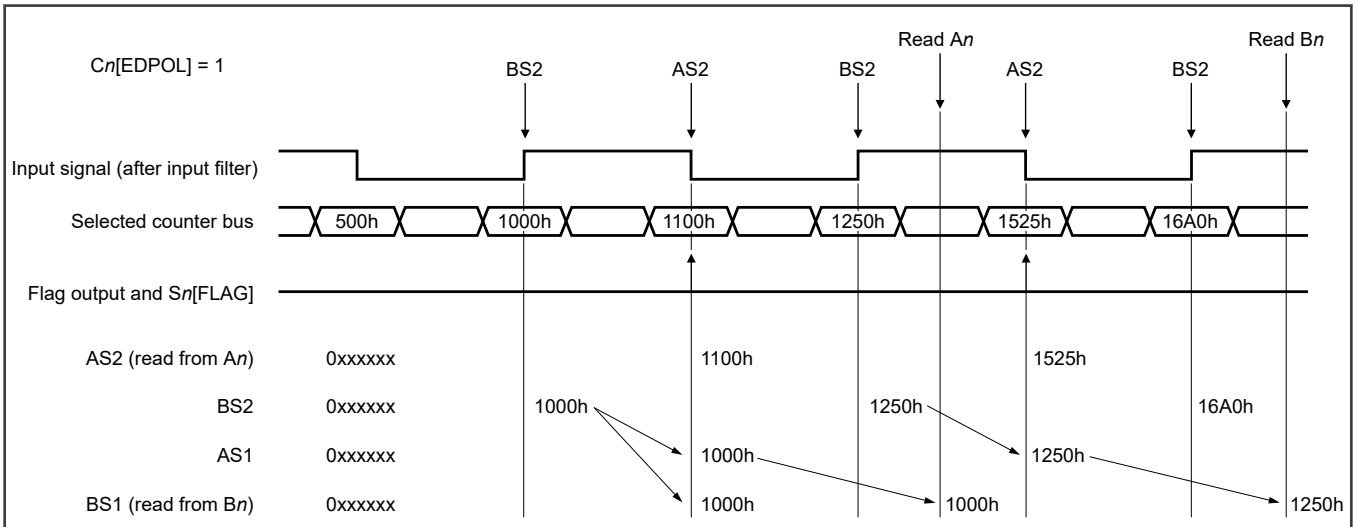



**Figure 303. Timing example: BS1 and AS1 updates on channel reads**

### 60.5.3.11 Input Period Measurement (IPM) Mode

#### 60.5.3.11.1 Overview

This mode allows you to measure the period of an input signal. Select whether to measure from the rising or falling edges with Edge Polarity (C*n*[EDPOL]).

The first edge of selected polarity:

- Latches the selected timebase into AS2 and BS2 (see AS1, AS2, BS1, and BS2 shadow registers)
- Transfers BS2 to BS1
- Does not set the input-capture flag (S*n*[FLAG])

The second edge of the same polarity:

- Latches the counter bus value into AS2 and BS2
- Transfers BS2 to BS1 and AS1
- Sets the input-capture flag (S*n*[FLAG])

eMIOS repeats this process for each subsequent capture.

eMIOS captures successive values on two consecutive edges of the same polarity. The measured input signal must have a period of at least four system clock cycles to be properly captured by the synchronization logic at the channel input, even if the input filter is in bypass mode.

#### 60.5.3.11.2 Coherency

Reading A*n*[A] updates BS1 with AS1, which provides coherent data in AS2 and BS1 (see Timing example: AS1 and BS1 updates on channel reads). It also disables transfers from BS2 and BS1 until the next read of B*n*[B].

Reading B*n*[B] updates BS1 with AS1. It also re-enables transfers from BS2 to BS1, which takes effect at the next edge capture.

#### 60.5.3.11.3 Timing example: measure input period

This example illustrates how to measure the input period by subtracting BS1 from AS2.

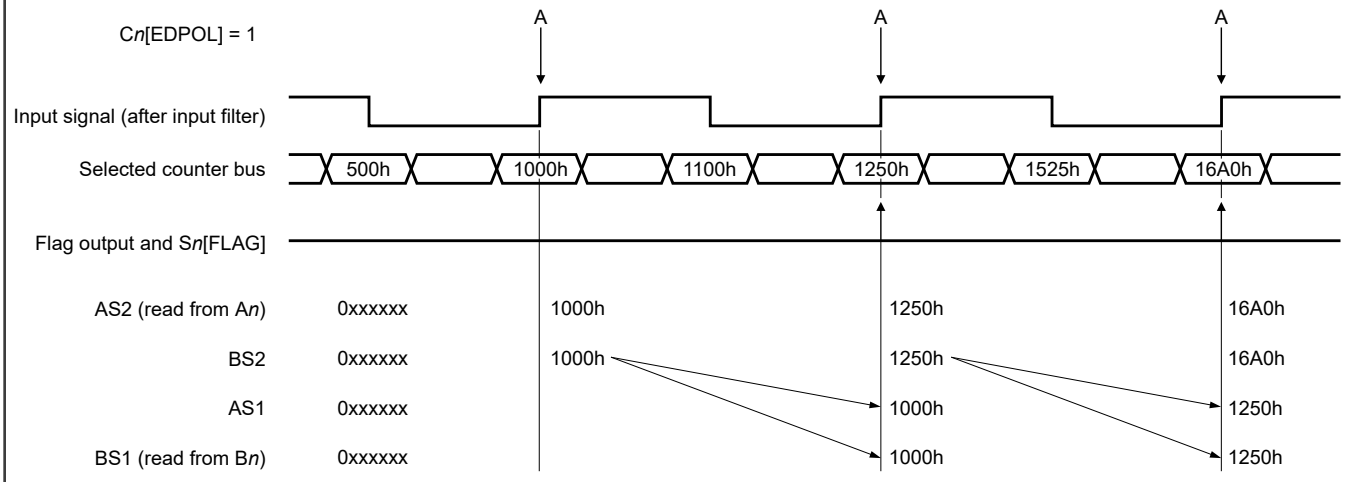



**Figure 304. Timing example: measure input period**

#### 60.5.3.11.4 Timing example: AS1 and BS1 updates on channel reads

This diagram illustrates AS1 and BS1 updates when you read A*n*[A] and B*n*[B].

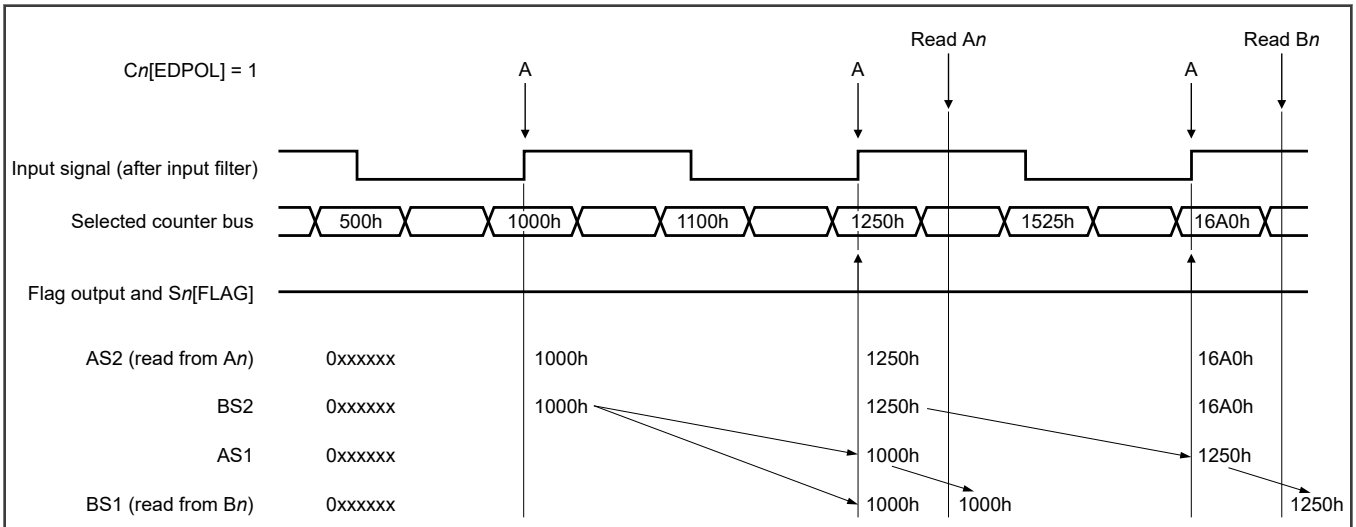



**Figure 305. Timing example: AS1 and BS1 updates on channel reads**

### 60.5.3.12 Double Action Output Compare (DAOC) mode

#### 60.5.3.12.1 Overview

In this mode, matches on comparators A and B cause the UC to generate the leading and trailing edges of the variable pulse width output. See DAOC submodes and MODE field values. There is no restriction on the order in which A and B matches occur.

When the UC enters DAOC mode from GPIO mode, eMIOS disables both comparators and drives the complement of Edge Polarity (C*n*[EDPOL]) onto the output flip-flop.

#### 60.5.3.12.2 DAOC submodes and MODE field values

Table 333. DAOC submodes and MODE field values

| When set flag event occurs | C*n*[MODE] (binary) |
|---|---|
| Only on B matches | 000_0110 |
| On A or B matches | 000_0111 |

#### 60.5.3.12.3 Data transfer and compare

If you enable output (write 0 to OUDIS[OU*n*]), on the next clock cycle the UC transfers AS2 and BS2 to AS1 and BS1 respectively (see Timing example: leading dead time insertion). If you disable output (write 1 to OUDIS[OU*n*]), no transfer occurs.

eMIOS enables and disables the A and B comparators independently. It enables comparator A only after the transfer to AS1 completes, and disables comparator A on the next A match. Similarly, eMIOS enables comparator B only after the transfer to BS1 completes, and disables comparator B on the next B match.

When a match occurs on comparator A, the output flip-flop acquires the value of Edge Polarity (C*n*[EDPOL]). When a match occurs on comparator B, the output flip-flop acquires the complement of Edge Polarity (C*n*[EDPOL]).

#### 60.5.3.12.4 Differences between flag on A match and flag on A and B matches

In the flag on B match submode (see DAOC submodes and MODE field values), the UC sets the input-capture flag (S*n*[FLAG]) only for matches on the B comparator. In the flag on both A and B matches submode, it sets the flag for matches on either comparator.

If subsequent enabled output compares occur on AS1 or BS1, the UC continues generating pulses regardless of the flag state.

If you load both AS1 and BS1 with the same value, the B match takes precedence and the UC drives the complement of Edge Polarity (C*n*[EDPOL]) onto the output flip-flop.

#### 60.5.3.12.5 Forced operation by Cn[FORCMA] and Cn[FORCMB]

Force Match A (C*n*[FORCMA]) and Force Match B (C*n*[FORCMB]) allow you to force the output flip-flop to the level corresponding to the comparator, Edge Polarity (C*n*[EDPOL]) for an A match, or the complement of Edge Polarity for a B match. The force-match operations do not set the input-capture flag (S*n*[FLAG]).

#### 60.5.3.12.6 Timing example: flag on both matches

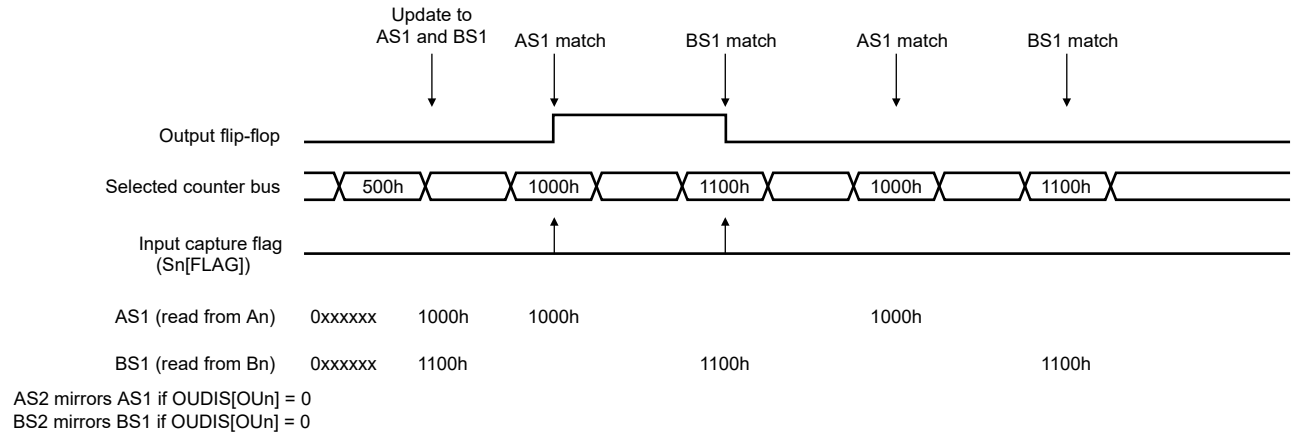



Figure 306. Timing example: flag on both matches

### 60.5.3.12.7 Timing example: flag on second match

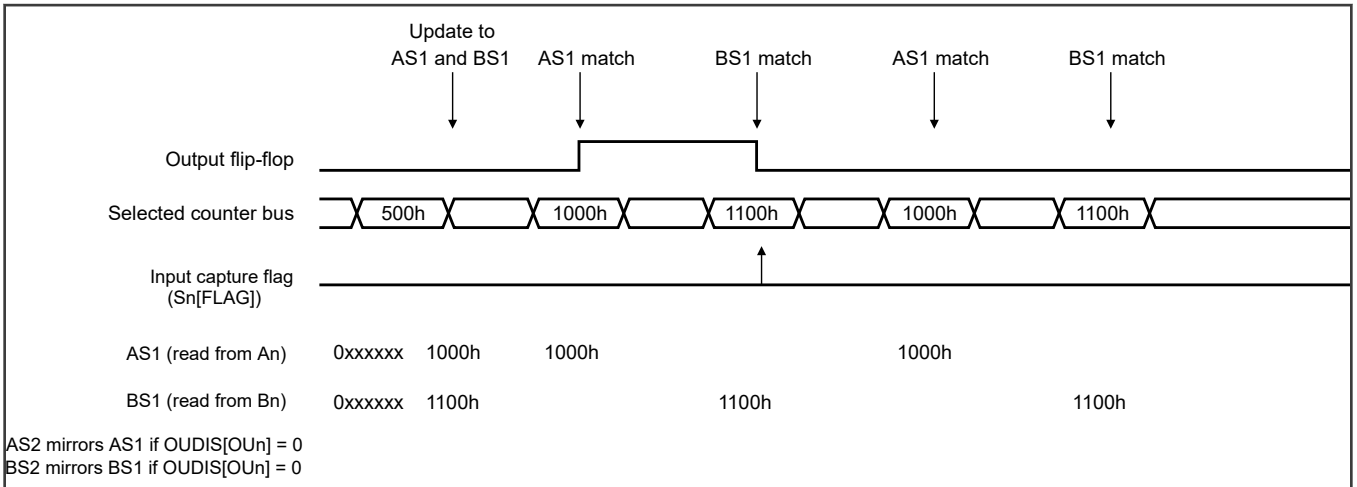



Figure 307. Timing example: flag on second match

### 60.5.3.12.8 Timing example: transfer disabling

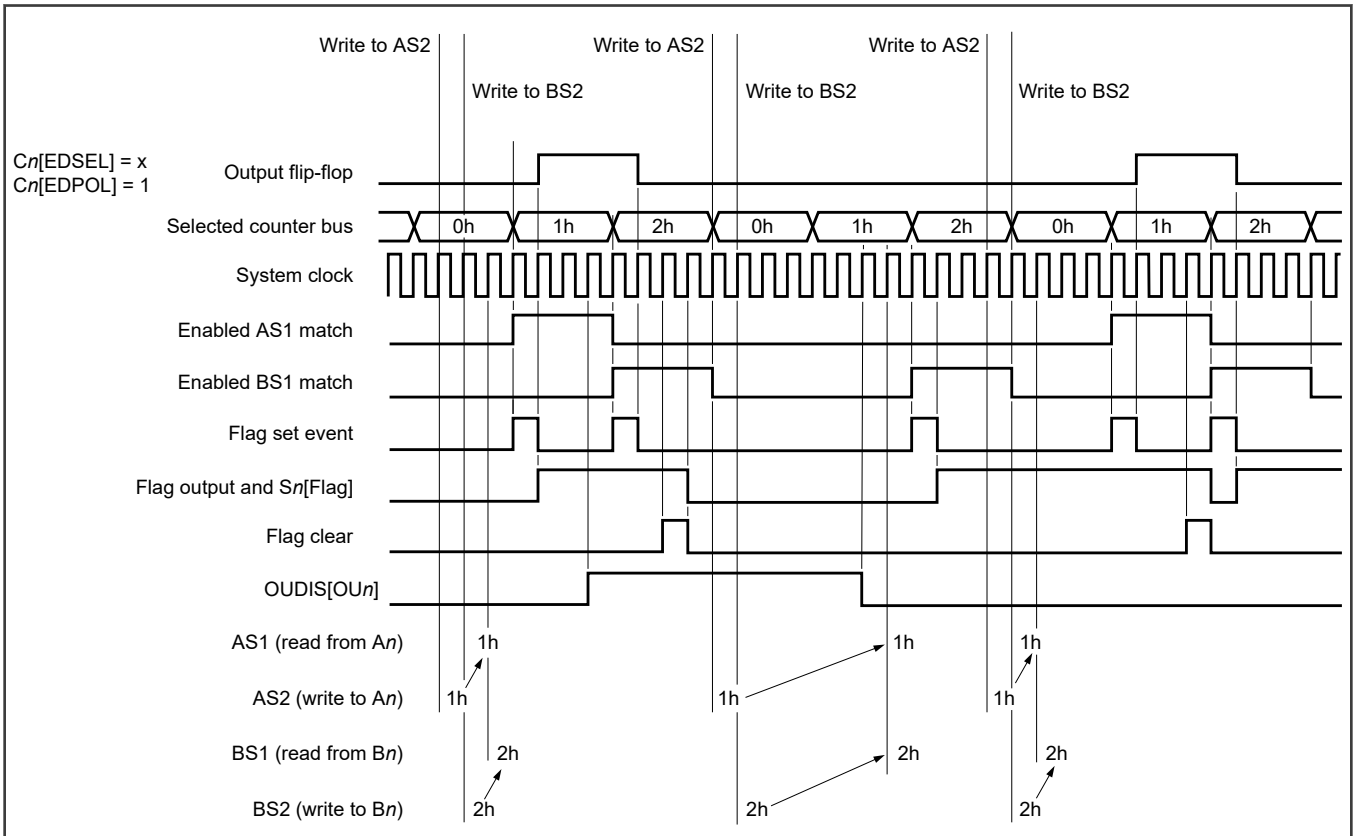



Figure 308. Timing example: transfer disabling

### 60.5.3.13 Pulse Edge Counting (PEC) mode

#### 60.5.3.13.1 Overview

This mode counts the number of pulses or edges detected on the input for a desired time window. C*n*[EDSEL] and C*n*[EDPOL] determine which edges the UC counts.

Specify the window start time in AS1 and the end time in BS1. After you write to AS1, when the selected timebase matches comparator A, eMIOS resets the internal counter and starts counting input events. When the timebase matches comparator B, eMIOS:

- Disables the internal counter.
- Transfers the counter's content to AS2.
- Sets the input-capture flag (changes S*n*[FLAG] to 1).

You obtain the number of detected pulses by reading C*n*[C] or AS2.

Timing example continuous and Timing example single-shot show how to use the UC in continuous or single-shot operation.

#### 60.5.3.13.2 PEC submodes and MODE field values

**Table 334. PEC submodes and MODE field values**

| Operation | C*n*[MODE] (binary) |
|---|---|
| Continuous; the next match between comparator A and the selected timebase resets the internal counter and enables counting again. To guarantee coherent measurements when you read C*n*[C] after S*n*[FLAG] becomes 1, you must check if the timebase value is out of the time interval defined by AS1 and BS1. Alternately, AS2 (available in ALTA*n*[ALTA]) always holds the latest available measurement, providing coherent data at any time after the first set-flag event. | 000_1010 |
| Single-shot; The next match between comparator A and the selected timebase has no effect until you write to A*n*[A]. eMIOS also transfers the C*n*[C] value to AS2 when a match in the B comparator occurs. | 000_1011 |

#### 60.5.3.13.3 Timing example continuous

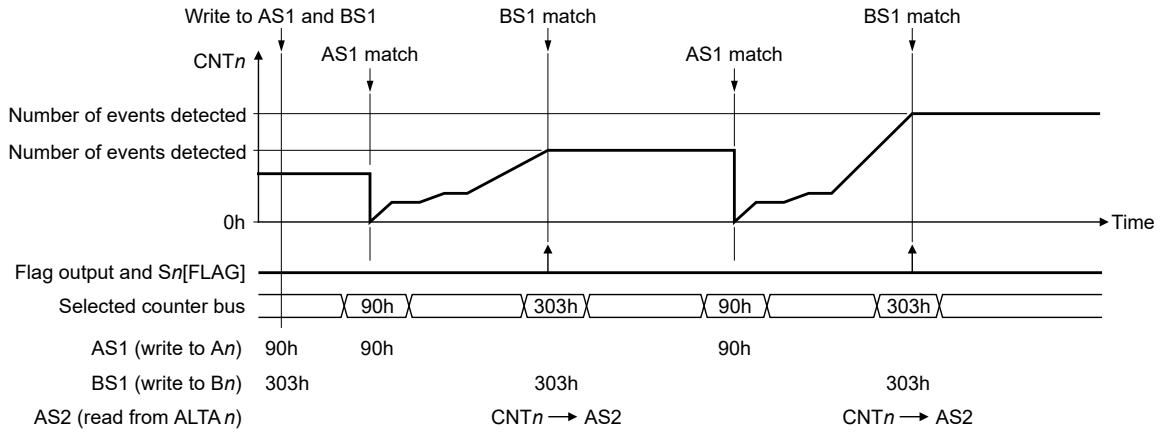



**Figure 309. Timing example continuous**

#### 60.5.3.13.4 Timing example single-shot

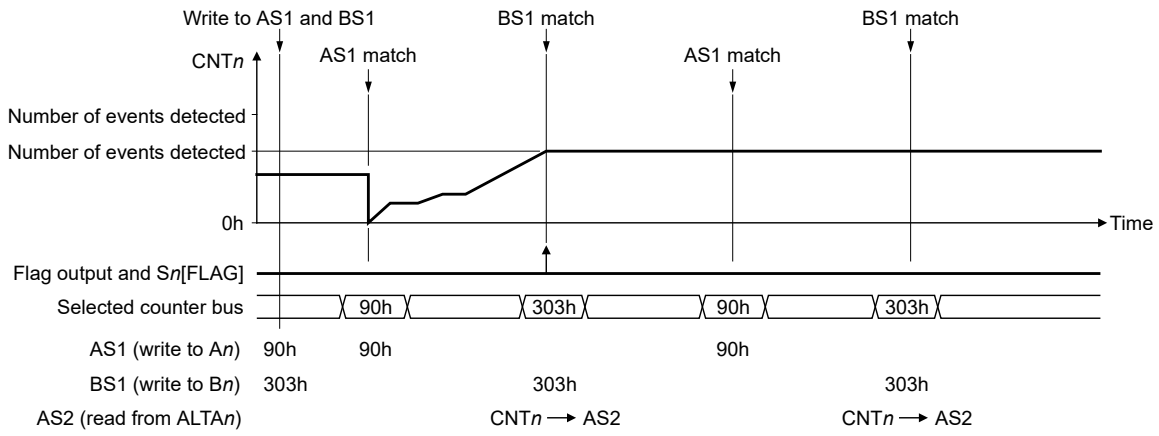



**Figure 310. Timing example single-shot**

### 60.5.3.14 Modulus Counter (MC) mode

#### 60.5.3.14.1 Overview

In this mode, the UC produces a timebase for a counter bus or as a general purpose timer. The internal counter (C*n*[C]) counts up from the current value until it matches AS1.

Selecting external clock source (see MC submodes and MODE field values) causes the UC to uses the input signal pin as the source. You select the triggering polarity edge using Edge Polarity (C*n*[EDPOL]) and Edge Select (C*n*[EDSEL]).

You must write only non-zero values to A*n*[A]. Writing to A*n*[A] or the internal counter may cause a match to be missed and the counter to roll over and resume operation in the next cycle. The channel writes 0 to BS1, which is not accessible by the MCU. You can read and write BS2, but it is not used in this mode.

#### 60.5.3.14.2 MC submodes and MODE field values

**Table 335. MC submodes and MODE field values**

| Submode or function | C*n*[MODE] (binary) |
|---|---|
| Internal clock source | 001_0pp0[1] |
| External clock source | 001_0pp1[1] |
| Internal counter reset on match start | 001_0p0p[1] |
| Internal counter reset on match end | 001_0p1p[1] |
| Up Count | 001_00pp[1] |
| Up Count-Down Count | 001_01pp[1] |

1. *p* = Adjust parameters for submodes and options. See Unified channels (UC).

### 60.5.3.14.3 Up Count submode

**Table 336. Up Count submode operation**

| Counter reset timing[1] | C*n*[MODE] | | |
|---|---|---|---|
| | Bit 1 | External clock (Bit 0 = 1) | Internal clock (Bit 0 = 0) |
| Reset on match start | 0 | When the AS1 match occurs, the UC:<br>• Writes 0 to the internal counter.<br>• Sets the input-capture flag (S*n*[FLAG]).<br>• Increments the internal counter at the next input event, resulting in a short zero count. | When the AS1 match occurs, the UC:<br>• Writes 0 to the internal counter.<br>• Sets the input-capture flag (S*n*[FLAG]).<br>• Maintains the internal counter at 0 on the next prescaler tick after the match.<br>• Resumes counting on the next prescaler tick. |
| | | See Timebase with the fastest prescaler ratio and Timebase generation with clear on match start and internal clock. | |
| Clear on match end | 1 | When the next input event after the AS1 match occurs, the UC:<br>• Writes 0 to the internal counter.<br>• Sets the input-capture flag (S*n*[FLAG]). | When the next internal counter tick after the AS1 match occurs, the UC:<br>• Writes 0 to the internal counter.<br>• Sets the input-capture flag (S*n*[FLAG]). |
| | | See Timebase with the fastest prescaler ratio and Timebase generation with clear on match end. | |

1. If you select internal clock source and a prescaler value of 1 for the internal counter, the behavior is the same for both submodes.

See Timing example: Up Count submode.

### 60.5.3.14.4 Timing example: Up Count submode

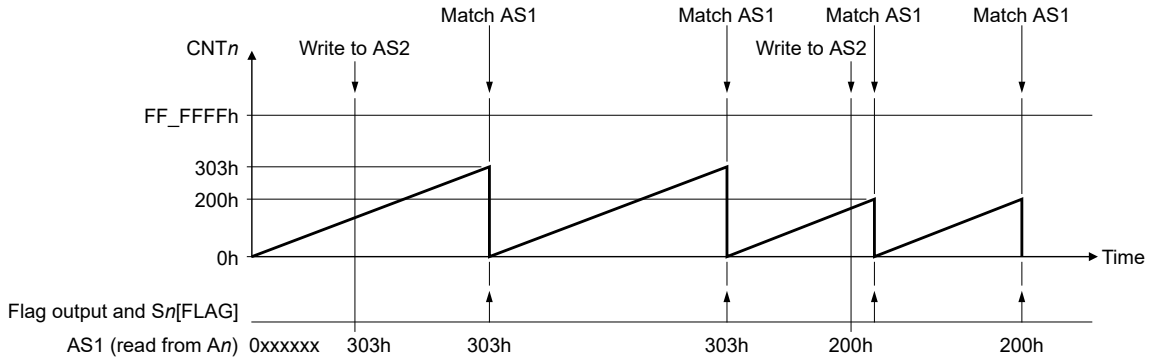



**Figure 311. Timing example: Up Count submode**

### 60.5.3.14.5 Up Count-Down Count submode operation

In Up Count-Down Count submode:

- When the internal counter matches AS1, the counter begins decrementing and the UC sets the input-capture flag (S*n*[FLAG]).
- When the internal counter matches BS1, the counter begins incrementing and, if in clear on match end operation (C*n*[1] = 1), the UC sets the input-capture flag (S*n*[FLAG]).

See Timing example: Up Count-Down Count submode.

### 60.5.3.14.6 Timing example: Up Count-Down Count submode

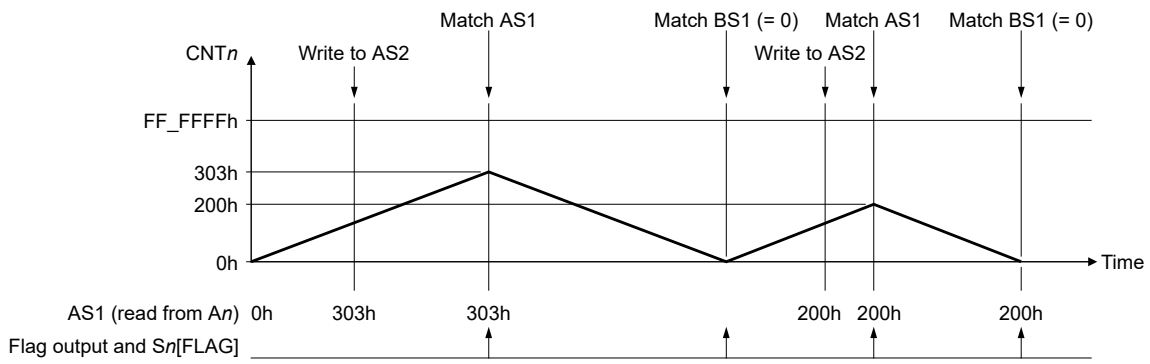



**Figure 312. Timing example: Up Count-Down Count submode**

### 60.5.3.14.7 Up Count with delayed reload operation

When the internal counter matches AS1 in the Up Count submode or BS1 in the Up Count-Down Count submode, the UC asserts the counter bus reload signal. You can read the counter bus reload signal in the following modes:

- Output PWM Buffered (OPWMB) mode
- Output Pulse Width and Frequency Modulation Buffered (OPWFMB) mode
- Center Aligned Output PWM with Dead Time Insertion Buffered (OPWMCB) mode

You can use the Reload Signal Output Delay Interval (C2*n*[UCRELDEL_INT]) to cause the UC to assert the reload signal only after a specified number of match events (from 2 to 32).

See Timing example: Up Count mode with delayed reload.

#### 60.5.3.14.8 Timing example: Up Count mode with delayed reload

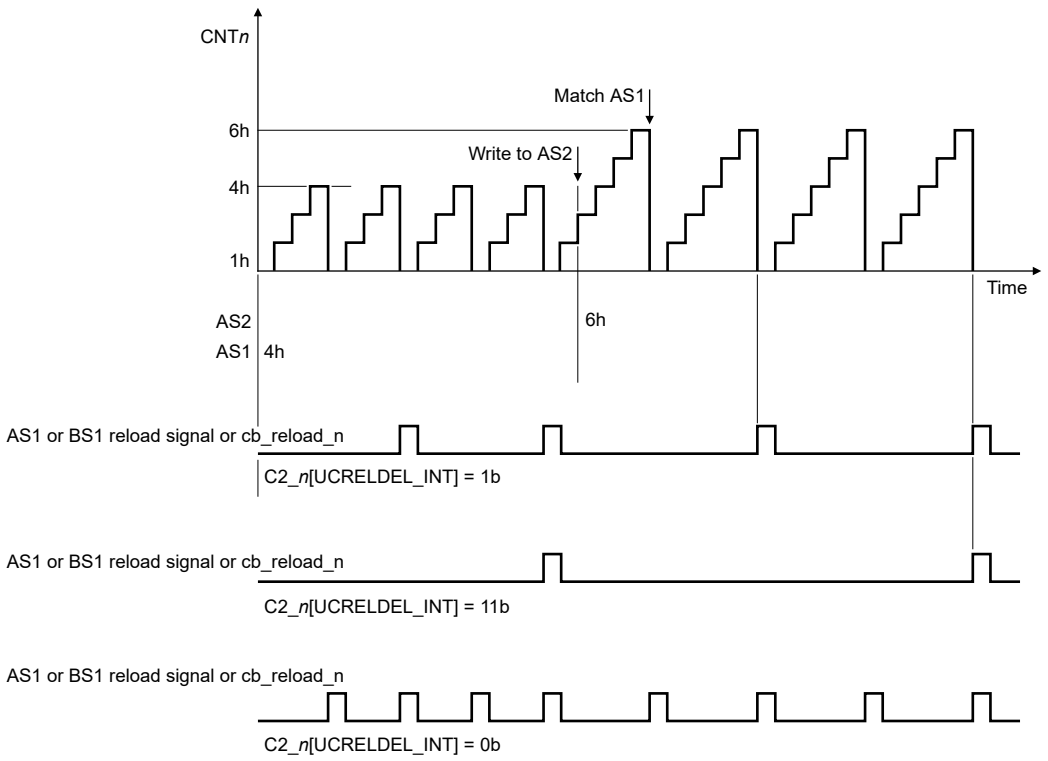



**Figure 313. Timing example: Up Count mode with delayed reload**

### 60.5.3.15 Modulus Counter Buffered (MCB) mode

#### 60.5.3.15.1 Overview

In this mode, the UC generates a timebase that can be shared with other channels through the internal counter buses. AS1 is double-buffered, so you can change AS2 at any time with smooth transitions. The UC updates AS1 at the counter period boundary, which is when the internal counter (CNT*n*[C]) transitions to 1h.

Selecting external clock (see MCB submodes and MODE field values) causes the UC to use the input signal as the source. You select the triggering edge with Edge Polarity (C*n*[EDPOL]) and Edge Select (C*n*[EDSEL]).

You must ensure the internal counter is in the range from 1h to the AS1 value; otherwise, the A*n*[A] match does not occur and this causes the internal counter to wrap at the maximum counter value (FFFFh for a 16-bit counter). After a counter wrap occurs, the UC resets the counter to 1h and resumes normal operation.

**NOTE**

Do not write 1h to the internal counter when the UC is in Freeze state. Doing so can cause the match to be missed and the counter to overflow.

#### 60.5.3.15.2 Changing to MCB mode

To avoid causing an unexpected interrupt or DMA request, you must use the following procedure to change to MCB mode:

1. Put the UC in GPIO mode (write 0b or 1b to C*n*[MODE]).
2. Disable the UC input capture flag (write 0 to C*n*[FEN]).
3. Write the correct values for MCB mode to A*n* and B*n*.

4. Put the UC in MCB mode (write the appropriate value to C*n*[MODE], as described in MCB submodes and MODE field values).
5. Clear the UC input capture flag (write 1 to S*n*[FLAG]).
6. Enable the UC input capture flag (write 1 to C*n*[FEN]).

### 60.5.3.15.3 MCB submodes and MODE field values

**Table 337. MCB submodes and MODE field values**

| Submode or function | C*n*[MODE] (binary) |
|---|---|
| Internal clock source | 101_0pp0b[1] |
| External clock source | 101_0pp1b[1] |
| Up Count | 101_000p[1] |
| Reserved | 101_001p[1] |
| Up Count-Down Count | 101_01pp[1] |
| Flag set only on match start | 101_010p[1] |
| Flags also set at counter period boundary | 101_011p[1] |

1. *p* = Adjust parameters for submodes and options. See Unified channels (UC).

### 60.5.3.15.4 Up Count submode

In the Up Count submode:

- The internal counter (CNT*n*[C]) starts counting up (incrementing) from its current value.
- When the internal counter matches AS1 and a clock tick occurs (either prescaled clock or input pin event), the UC resets the internal counter to 1h.
- The UC sets the input-capture flag (S*n*[FLAG]) one system clock cycle after the match occurs.

You must write only values greater than 1h to AS1.

The boundary between counter period *n* and period *n*+1 occurs when the UC changes the internal counter from the AS1 value in period *n* to 1h in period *n*+1. AS1 defines the counter period.

The UC asserts the AS1 load signal (which has a duration of one system clock cycle) at the last system clock cycle of a counter period. This causes the UC to update AS1 with AS2 at the period boundary, at the same time that it resets the counter to 1h. Therefore, any value you write to AS2 during period *n* updates AS1 at the next period boundary and the UC uses it in period *n*+1. See Timing example: Up Count submode.

You can use Output Update Disable (OUDIS) to delay the update of AS1 for synchronization.

The UC sets the input-capture flag (S*n*[FLAG]) at the period boundary. See Timing example: Up Count submode flag assertion.

### 60.5.3.15.5 Timing example: Up Count submode

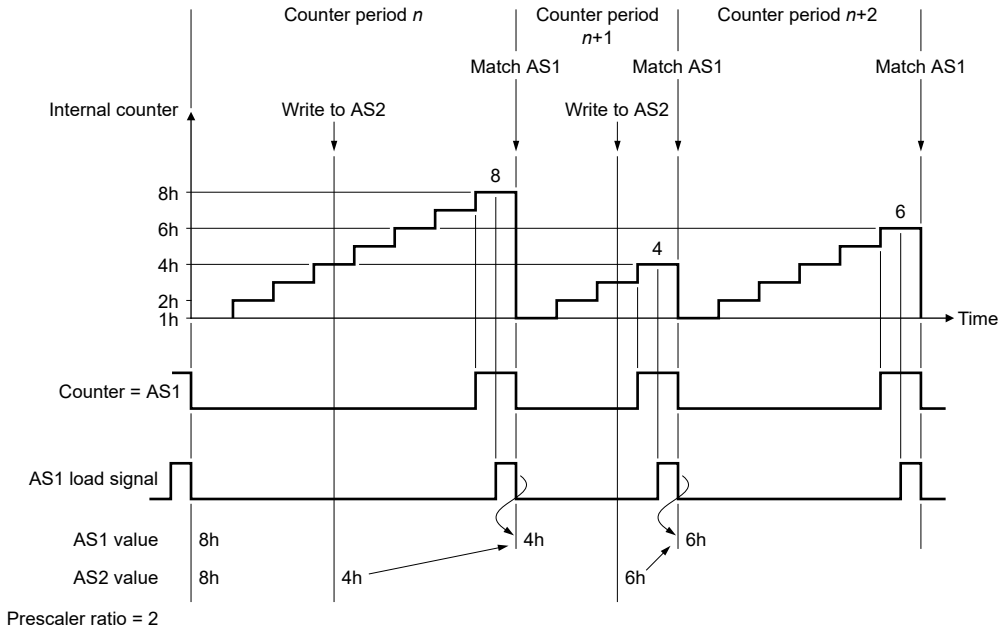



Figure 314. Timing example: Up Count submode

### 60.5.3.15.6 Timing example: Up Count submode flag assertion

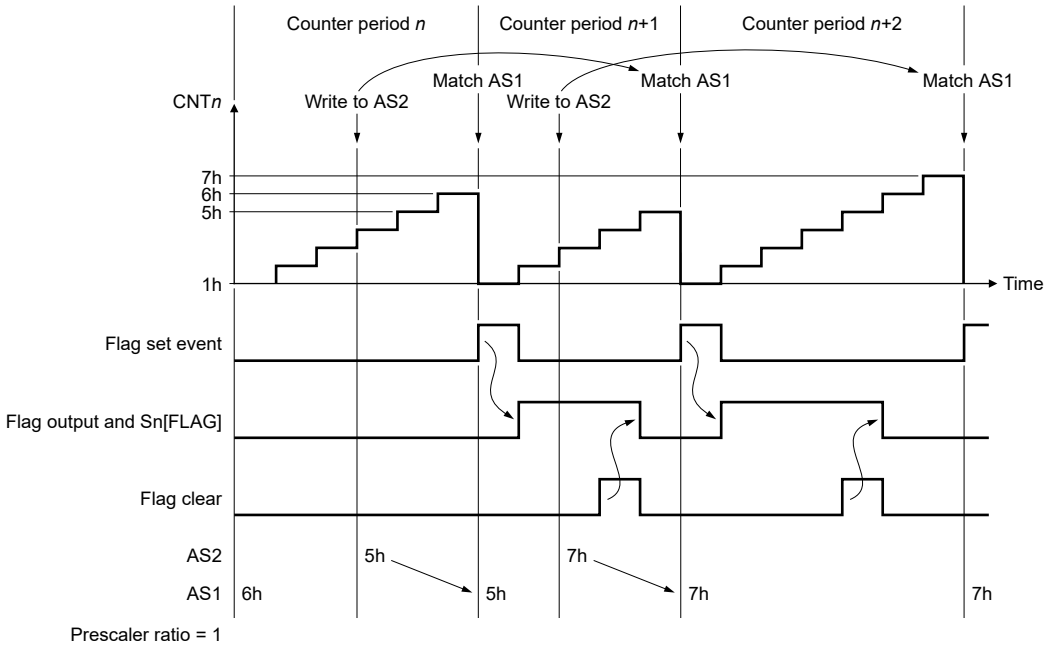



Figure 315. Timing example: Up Count submode flag assertion

### 60.5.3.15.7 Up Count-Down Count submode

In Up Count-Down Count submode:

- The counter period is: (2 x AS1) – 2. The counter changes direction at AS1 match and counts down (decrements) until it reaches 1h, then counts up (increments) again.
- BS1 generates a match to start the internal counter incrementing, and BS1 cannot be changed in this submode.

The UC updates AS1 at the counter period boundary. If you write AS2 (write A*n*[A]) during period *n*, the UC uses this new value in period *n*+1 for AS1 match. You can disable updates of AS1 (OUDIS[OU*n*]). See Timing example: Up Count-Down Count submode.

The UC sets the input-capture flag (S*n*[FLAG]) at AS1 match start, and may also do so at the period boundary. See Overview and Timing example: Up Count-Down Count submode flag assertion.

### 60.5.3.15.8 Timing example: Up Count-Down Count submode

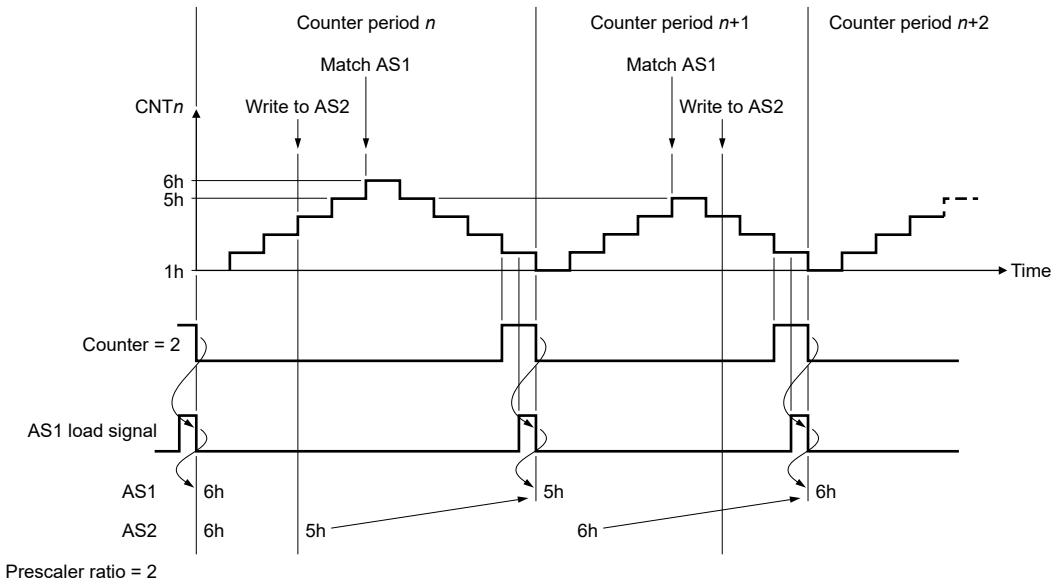



Figure 316. Timing example: Up Count-Down Count submode

### 60.5.3.15.9 Timing example: Up Count-Down Count submode flag assertion

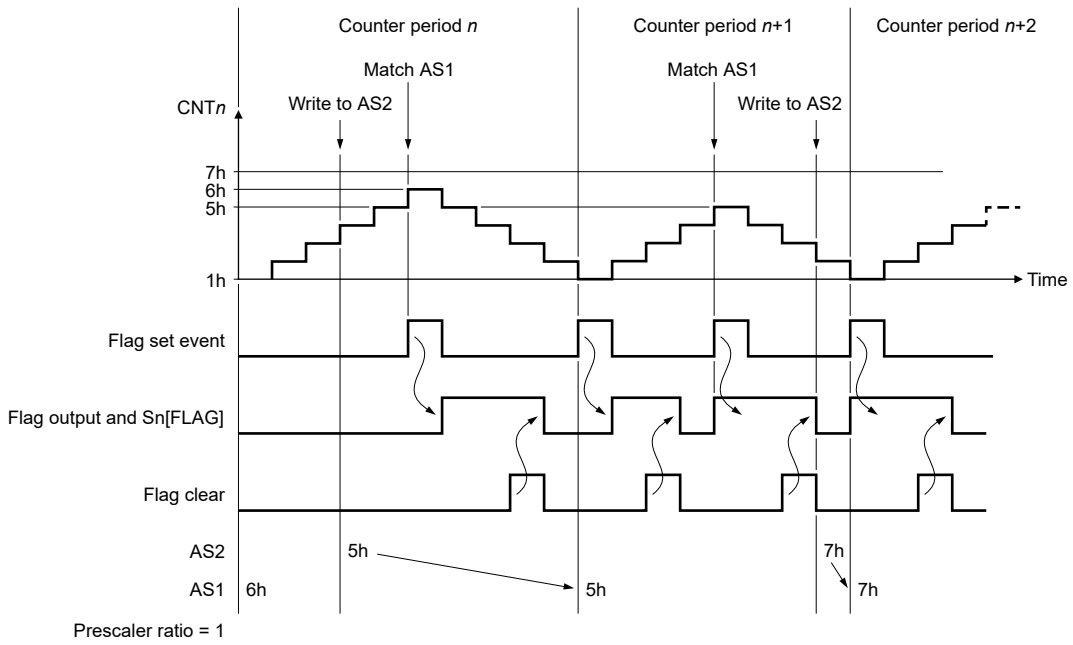



**Figure 317. Timing example: Up Count-Down Count submode flag assertion**

### 60.5.3.15.10 Up Count mode with delayed reload operation

When the internal counter matches AS1 in the Up Count submode or 1h in the Up Count-Down Count submode (see MCB submodes and MODE field values), the UC asserts the counter bus reload signal. You can read the counter bus reload signal in the following modes:

- Output PWM Buffered (OPWMB) mode
- Output Pulse Width and Frequency Modulation Buffered (OPWFMB) mode
- Center Aligned Output PWM with Dead Time Insertion Buffered (OPWMCB) mode

You can use the Reload Signal Output Delay Interval (C2[UCRELDEL_INT]) to cause the UC to assert the reload signal only after a specified number of match events (from 2 to 32).

See Timing example: Up Count mode with delayed reload.

#### 60.5.3.15.11 Timing example: Up Count mode with delayed reload

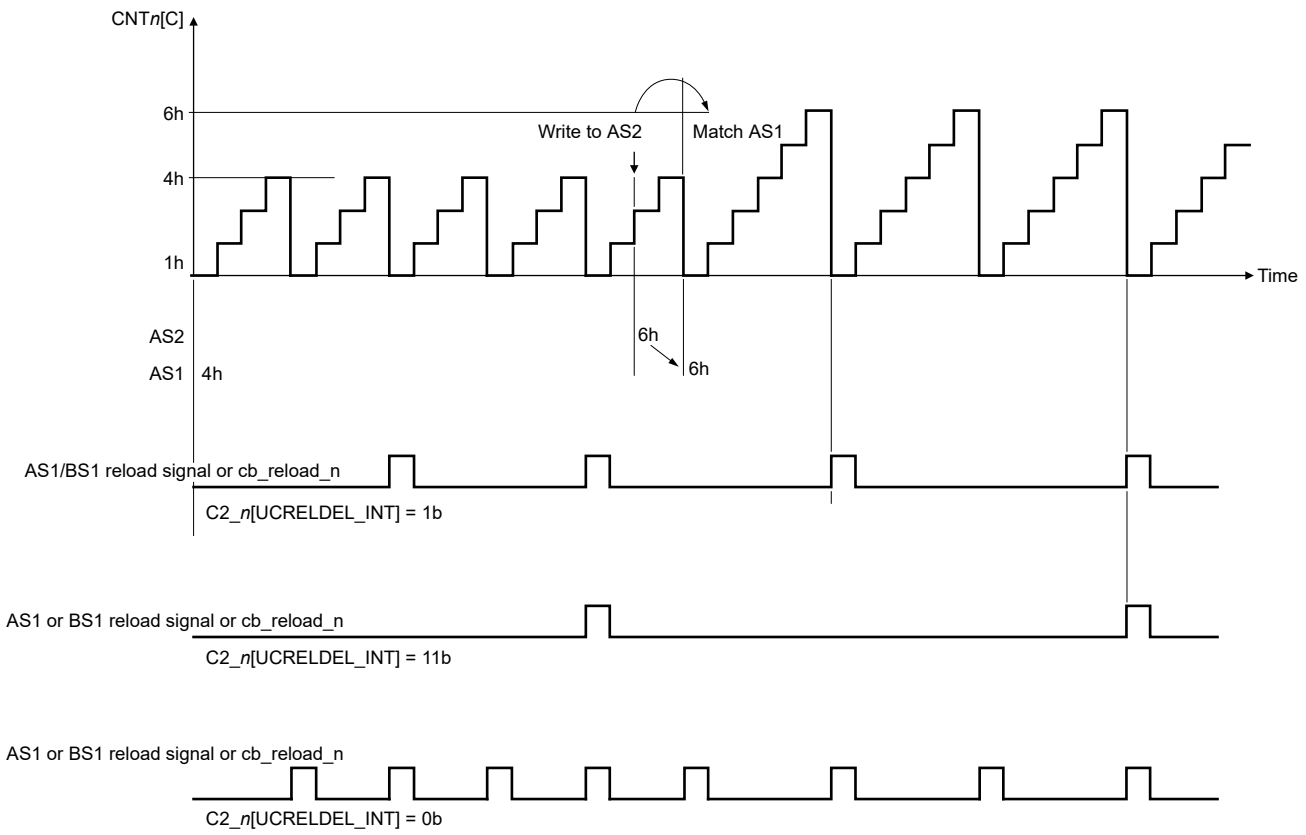



**Figure 318. Timing example: Up Count mode with delayed reload**

### 60.5.3.16 Output Pulse Width and Frequency Modulation Buffered (OPWFMB) mode

#### 60.5.3.16.1 Overview

In this mode, the UC produces an output signal that has:

- A duty cycle determined by AS1.
- A period of BS1.

The double-buffered AS1 and BS1 registers produce smooth duty-cycle and period transitions when you change AS1 and BS1 during runtime.

See Table 344 for the values that you must write to C*n*[MODE] to enter this mode.

To provide smooth and consistent channel operation:

- AS1 and BS1 are both double-buffered to allow smooth signal generation when changing the register values.
- There is a delay from the AS1 match to the output pin transition (see AS1 and BS1 match timing).
- The internal counter ranges from 1h to BS1. When you write to BS1, you must write only values greater than 1h. Writing 0h or 1h leads to unpredictable results.

When a match on comparator A occurs, the UC drives C*n*[EDPOL] onto the output flip-flop. When a match on comparator B occurs, the UC drives the complement of C*n*[EDPOL] onto the output flip-flop. A BS1 match also causes the internal counter to transition to 1h, thus restarting the counter cycle.

The UC automatically selects the internal channel counter as the timebase when you select this mode. The mode supports duty cycles ranging from 0% to 100%.

#### 60.5.3.16.2 OPWFMB submodes and MODE field values

**Table 338. OPWFMB submodes and MODE field values**

| Match behavior | C*n*[MODE] (binary) |
|---|---|
| When BS1 matches the selected counter | 101_1000 |
| When AS1 or BS1 match the selected counter | 101_1010 |

#### 60.5.3.16.3 Avoid counter wrapping

When the UC enters OPWFMB mode, the output flip-flop acquires the value of C*n*[EDPOL]. If the UC transitions from GPIO to OPWFMB mode when the internal counter value is not within the range 1h–BS1, then the B match never occurs. Because the match never occurs, the internal counter wraps at the maximum counter value—FFFFh for a 16-bit counter. When the counter wrap occurs, the counter returns to 1h and normal OPWFMB operation resumes. Therefore, to avoid this counter wrapping, ensure that the counter value is within the range 1h–BS1 before the UC enters OPWFMB mode.

#### 60.5.3.16.4 AS1 and BS1 match timing

As illustrated in Timing example: AS1 and BS1 match, the output pin transition occurs when the AS1 or BS1 match signal deasserts, as indicated by the AS1 match negative-edge detection signal. If you write 4h to AS1, the output pin transitions 4 counter increments plus one system clock cycle after the counter period starts.

#### 60.5.3.16.5 Timing example: AS1 and BS1 match

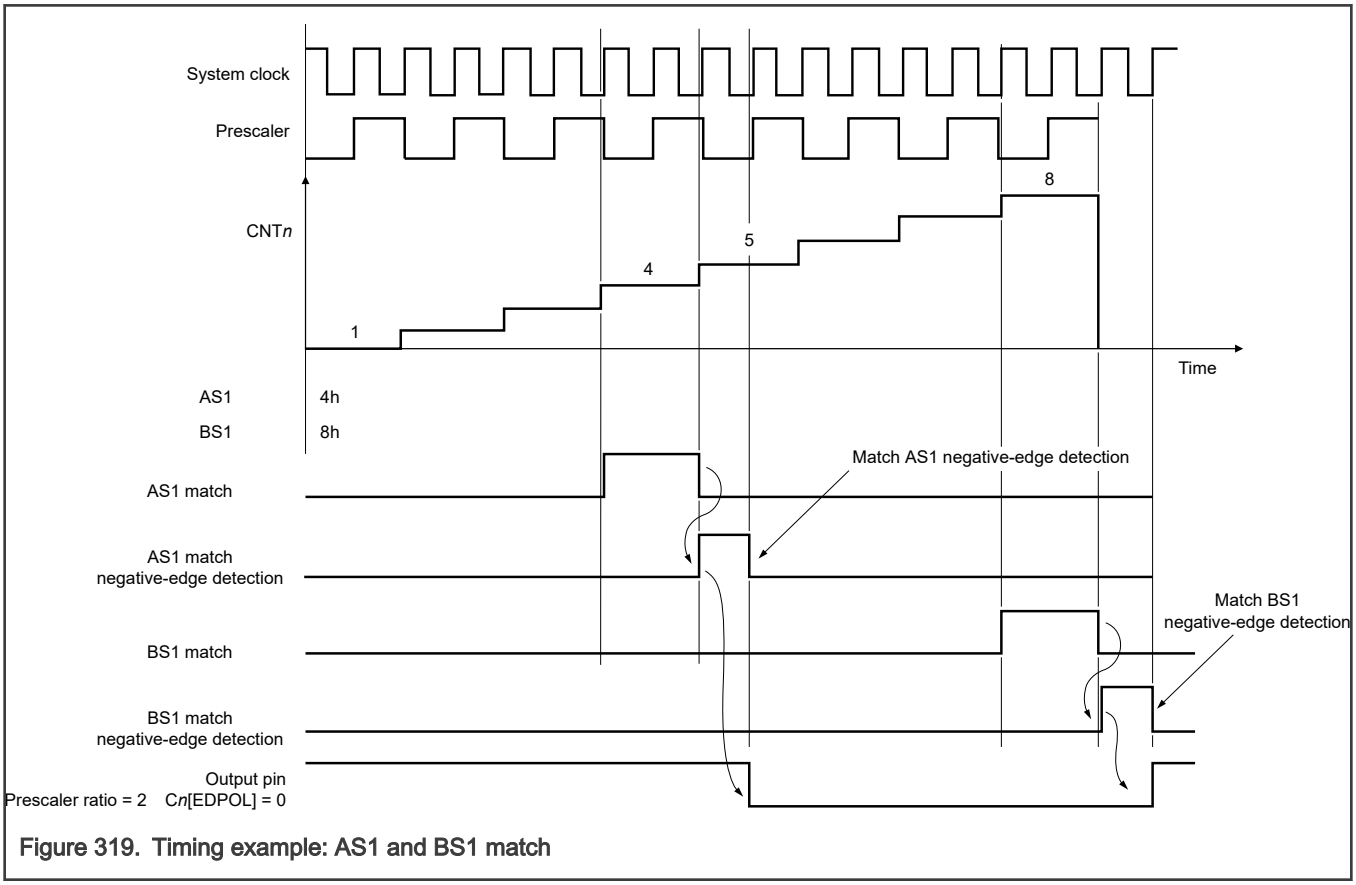



**Figure 319. Timing example: AS1 and BS1 match**

### 60.5.3.16.6 0% duty cycle

Timing example: AS1 = 0 illustrates the generated output signal when AS1 = 0, which results in a 0% duty cycle.

Because the internal counter never reaches zero in this situation, the UC infers a match as if AS1 = 1h. However, it is the positive edge of the match signal that triggers the output pin transition instead of the negative edge, which is what happens when AS1 = 1h. The AS1 positive-edge match signal from period *n*+1 occurs at the same time as the BS1 negative-edge match signal from period *n*. This timing allows the AS1 positive-edge match to mask the BS1 negative-edge match when they occur at the same time. As a result, no transition occurs on the output flip-flop and the UC generates a 0% duty cycle.

### 60.5.3.16.7 Timing example: AS1 = 0

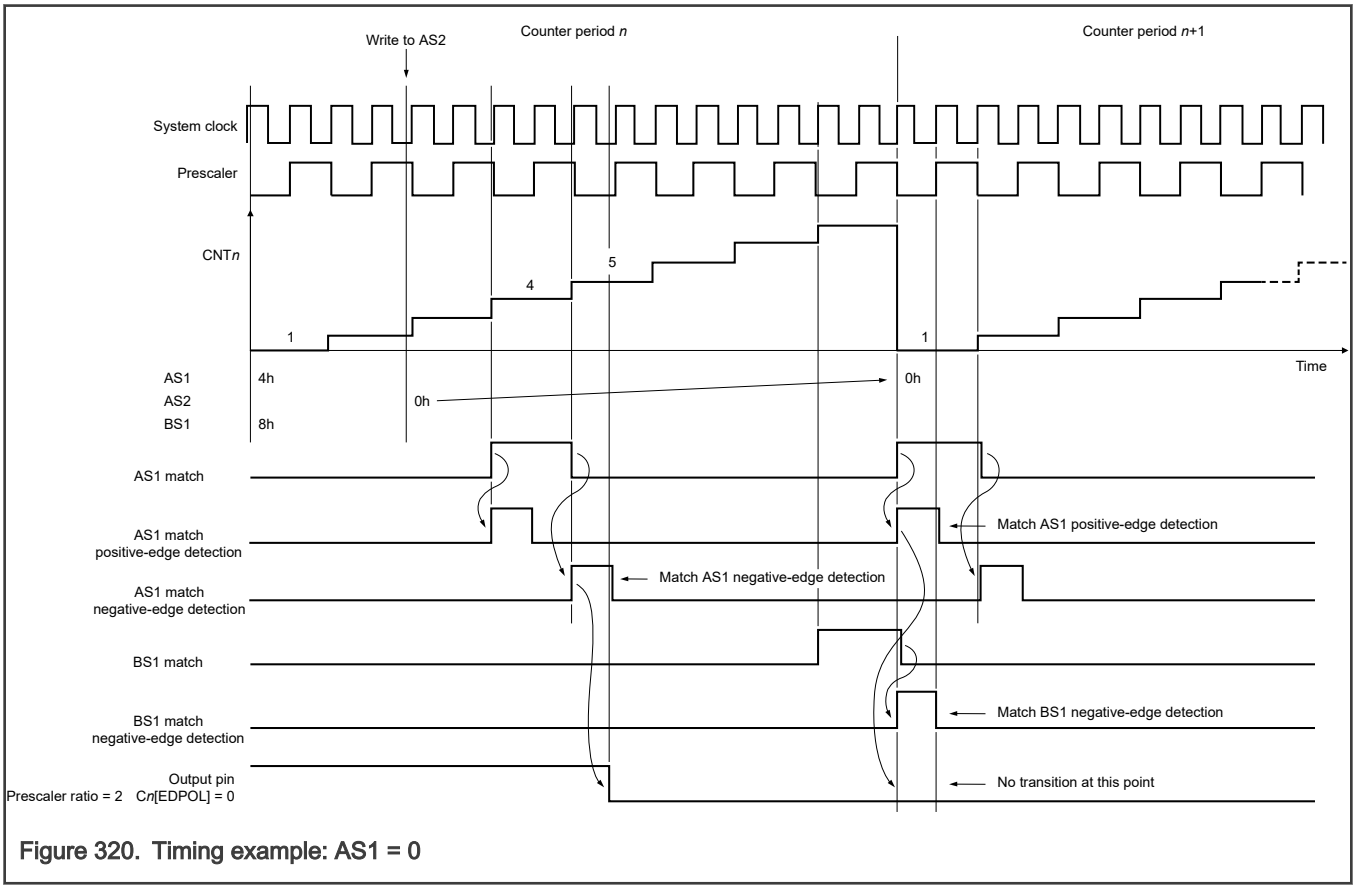



Figure 320. Timing example: AS1 = 0

### 60.5.3.16.8 Loading AS1 and BS1

As illustrated in Timing example: AS1 and BS1 update and FLAGs, AS1 and BS1 both use the same signal, generated at the last system clock cycle of a counter period. Therefore, eMIOS updates AS1 and BS1 with AS2 and BS2 values, respectively, at the same time that it changes the internal counter (CNT*n*[C]) to 1h. This event is the counter period boundary. The load signal pulse lasts for one system clock cycle. If you write AS2 and BS2 within counter period *n*, their values are available in AS1 and BS1, respectively, at the first clock cycle of counter period *n*+1. The UC then uses these new values for matches at counter period *n*+1. You can use Output Update Disable (OUDIS) to delay updating AS1 and BS1 for synchronization purposes.

### 60.5.3.16.9 Timing example: AS1 and BS1 update and FLAGs

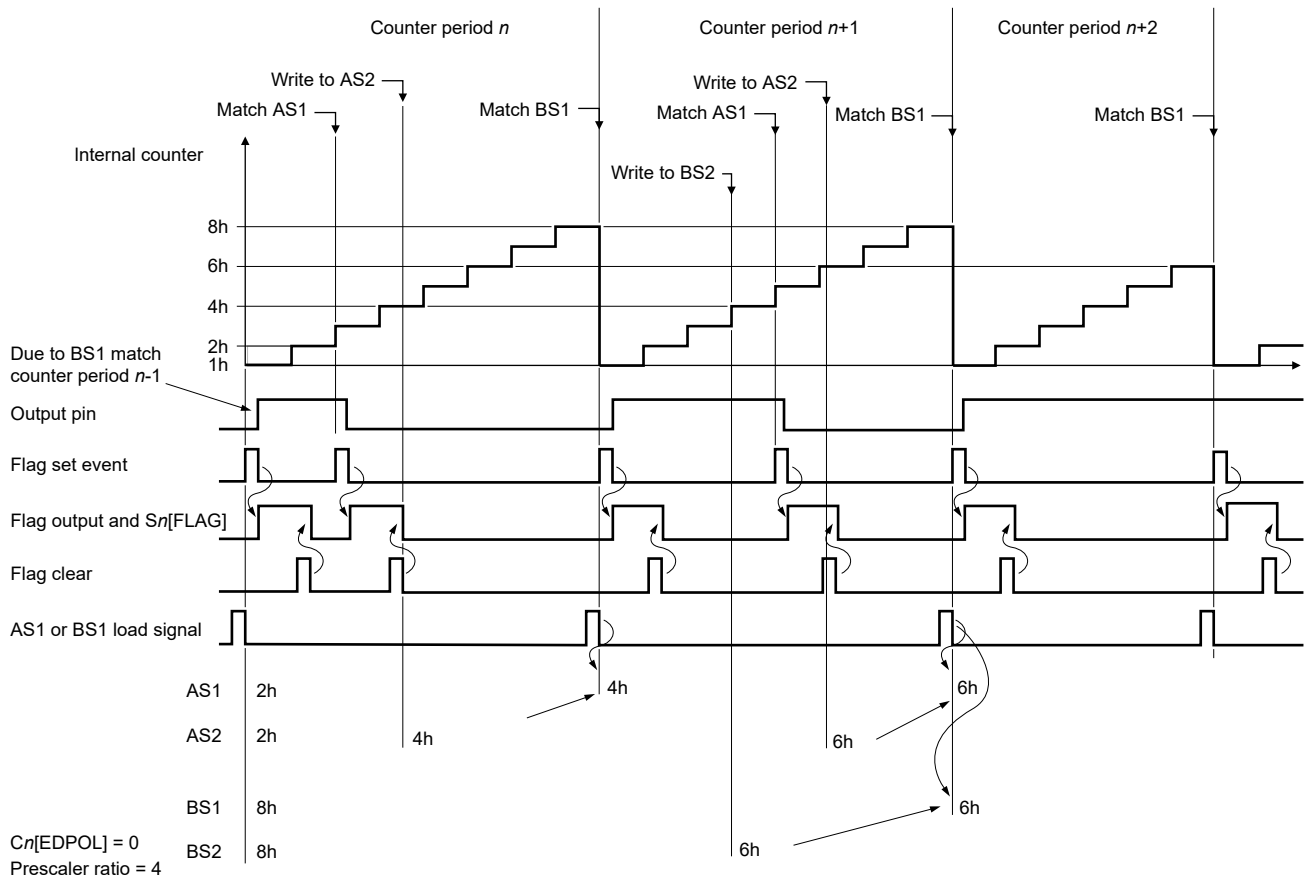



Figure 321. Timing example: AS1 and BS1 update and FLAGs

### 60.5.3.16.10 Flag generation

Timing example: AS1 and BS1 update and FLAGs assumes that both the channel and global prescalers are 1h (divide ratio of 2), causing the internal counter to transition every four system clock cycles. You can configure flags to be generated only on BS1 matches when bit 1 of C*n*[MODE] is 0, or on both AS1 and BS1 matches when that bit is 1. Because the BS1 flag occurs at the period boundary, you can use this flag to indicate that the AS2 or BS2 data written in period *n* has been loaded to AS1 or BS1, respectively, thereby generating matches in period *n*+1. Flag operation is synchronous, which means the flag sets one system clock cycle after the set-flag event.

### 60.5.3.16.11 Output disable

As illustrated in Timing example: active output disabled, the output-disable feature in OPWFMB mode forces the channel output flip-flop to the value of C*n*[EDPOL]. This functionality supports applications that use active-high signals and a high-to-low transition at an AS1 match. In this case, you must write 0 to C*n*[EDPOL]. You also must configure the internal counter to transition on every system clock cycle (write 0 to both GCP and CP).

The output-disable feature operates synchronously, meaning that the assertion of the Output Disable input signal causes the channel output flip-flop to transition to C*n*[EDPOL] at the next system clock cycle. If the Output Disable input pin deasserts, the output-signal transition occurs at the next AS1 or BS1 match.

Timing example: active output disabled assumes that the Output Disable input is enabled (using C*n*[ODIS]) and selected (using C*n*[ODISSL]). See UC Control n (C0 - C23) for a detailed description of the ODIS and ODISSL fields that enable and select the Output Disable inputs, respectively.

### 60.5.3.16.12 Timing example: active output disabled

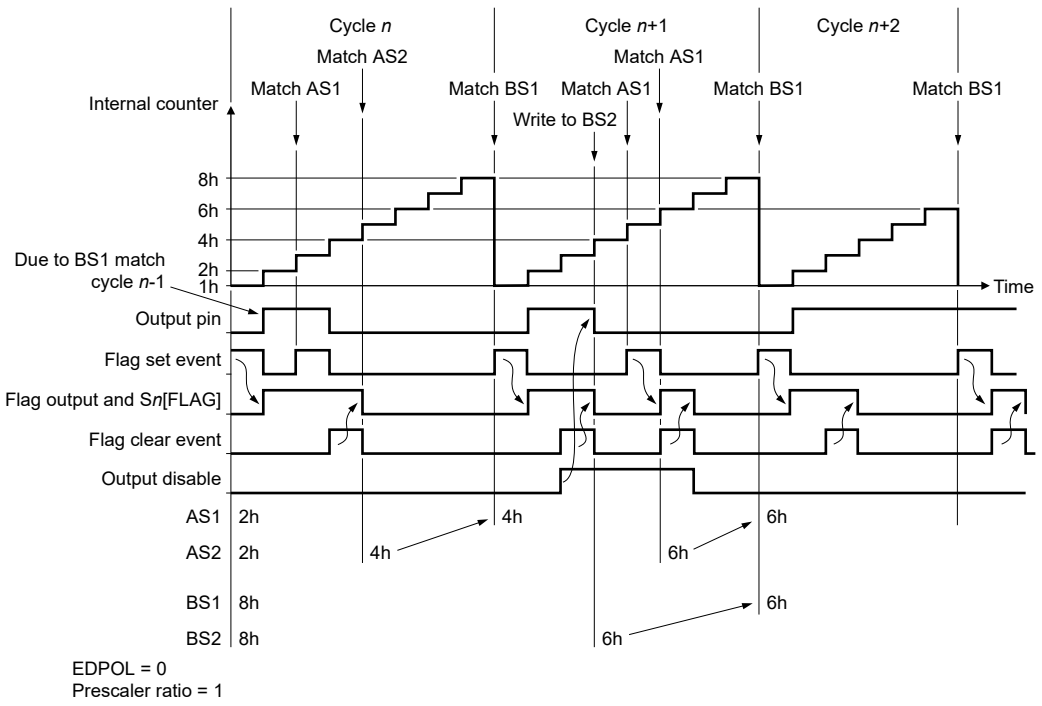



**Figure 322. Timing example: active output disabled**

### 60.5.3.16.13 Force match

Force Match A (C*n*[FORCMA]) and Force Match B (C*n*[FORCMB]) allow you to force the output flip-flop to the level corresponding to a match on comparators A or B, respectively. Similarly to a BS1 match, Force Match B changes the internal counter to 1h. The force-match operations do not set the input-capture flag (S*n*[FLAG]).

### 60.5.3.16.14 100% and 0% duty cycles

Timing example: from 100% to 0% duty cycle illustrates the generation of 100% and 0% duty-cycle signals. Initially, AS1 = 8h and BS1 = 8h. In this case, a BS1 match has precedence over an AS1 match—therefore, the output flip-flop acquires the complement of C*n*[EDPOL]. This configuration corresponds to a 100% duty-cycle signal. The same output signal can be generated for any AS1 value greater than or equal to BS1.

If AS1 is 0, the UC generates a 0% duty-cycle signal. In this case, the BS1 = 8h match from period 8 occurs at the same time as the AS1 = 0h match from period 9. See Timing example: AS1 = 0 for AS1 and BS1 match generation. In this case, an AS1 match has precedence over a BS1 match, and the output signal transitions to C*n*[EDPOL].

#### 60.5.3.16.15 Timing example: from 100% to 0% duty cycle

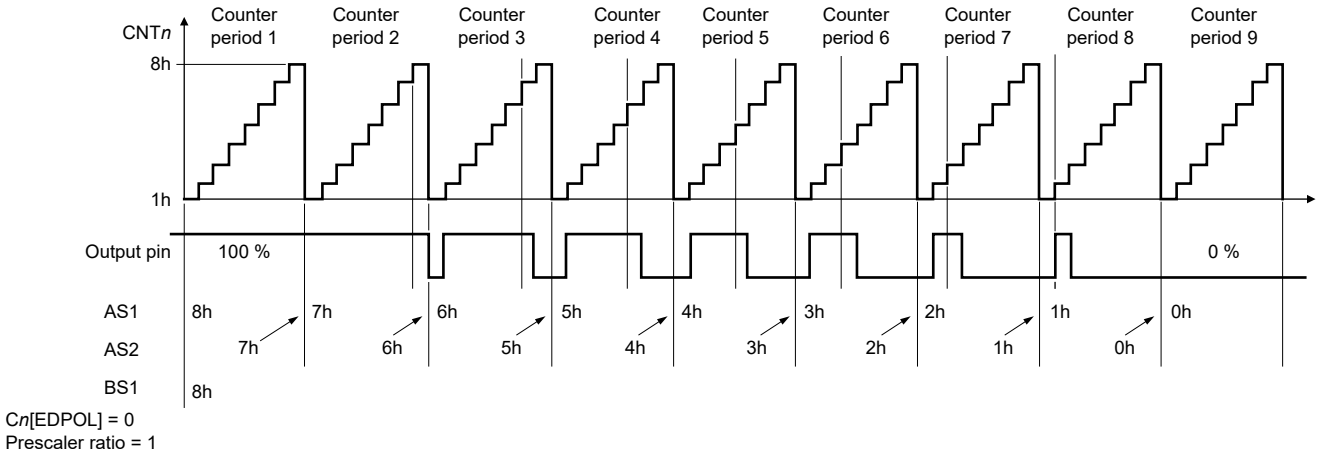



**Figure 323. Timing example: from 100% to 0% duty cycle**

### 60.5.3.17 Center Aligned Output PWM with Dead Time Insertion Buffered (OPWMCB) mode

#### 60.5.3.17.1 Overview

In this mode, the UC generates a center-aligned PWM pulse with dead time insertion in the leading or trailing edge. See OPWMCB functions and MODE field values. AS1 and BS1 are double-buffered to allow smooth output signal generation when you change the AS2 or BS2 values.

When the UC enters OPWMCB mode from GPIO mode:

- The UC drives the complement of Edge Polarity (C*n*[EDPOL]) on the output flip-flop.
- Select the timebase using Bus Select (C*n*[BSL]). The timebase you select must be running in Up Count-Down Count mode. See Timing example: Up Count submode flag assertion. You must start the MCB timebase after the UC enters OPWMCB mode to avoid missing a match on the first duty cycle.
- Store the ideal duty cycle for the PWM signal in AS1 (write to A*n*[A]), which the UC compares with the selected timebase.
- Store the dead time value in BS1 (write to B*n*[B]), which the UC compares with the internal counter (CNT*n*[C]).

The internal counter may use the internal prescaler ratio, while the selected timebase may use a different prescaler ratio. The UC always resets the internal counter to 1h when an AS1 match occurs.

As in OPWMB and OPWFMB modes, the UC generates the match signal which compares the selected timebase with AS1 or BS1. When the UC deasserts the channel's combinational comparator output signal, it uses the match signal to assert or negate the channel's output flip-flop. See Timing example: AS1 and BS1 match for an illustration of the delay from match to output flip-flop transition in OPWFMB mode. The operation of OPWMCB mode is similar to OPWFMB mode with respect to match behavior and output pin transition.

See Entering OPWMCB mode.

#### 60.5.3.17.2 OPWMCB functions and MODE field values

**Table 339. OPWMCB functions and mode field values**

| Function | C*n*[MODE] (binary) |
|---|---|
| Insert trailing edge dead time with duty cycle determined by BS1 + AS1 | 101_11p0[1] |

*Table continues on the next page...*

Table 339. OPWMCB functions and mode field values (continued)

| Function | C*n*[MODE] (binary) |
|---|---|
| Insert leading edge dead time with duty cycle determined by BS1 – AS1 | 101_11p1[1] |
| Assert the input capture flag (Sn[FLAG]) on the trailing edge of the output PWM signal | 101_110p[1] |
| Assert the input capture flag (Sn[FLAG]) on both edges of the output PWM signal | 101_111p[1] |

1. *p* = Adjust parameters for submodes and options. See Unified channels (UC).

#### 60.5.3.17.3 Entering OPWMCB mode

To enter OPWMCB mode (this procedure assumes the channel is initially in GPIO mode):

1. Disable the GCP (write 0 to MCR[GPREN]).
2. For the timebase (MCB) channel:
   a. Disable the CP (write 0 to C*n*[UCPREN]).
   b. Initialize the internal counter (write 1h to CNT*n*[C]).
   c. Initialize the A data (write appropriate value to A*n*[A]).
   d. Configure the UC for MCB Up Count-Down Count mode (write 101_01*xy*b to C*n*[MODE], where *x* = 0 for flag set on match start or *x* = 1 for flag set at period boundary, and *y* = 0 for internal clock source or *y* = 1 for external clock source).
   e. Select the CP ratio (C*n*[UCPRE]).
   f. Enable the CP (write 1 to C*n*[UCPREN]).
3. For the OPWMCB channel:
   a. Disable the CP (write 0 to C*n*[UCPREN]).
   b. Initialize the A data (write appropriate value to A*n*[A]).
   c. Initialize the B data (write appropriate value to B*n*[B]).
   d. Select the timebase input (C*n*[BSL]).
   e. Configure the UC for MCB Up Count-Down Count mode (write 101_01*xy*b to C*n*[MODE], where *x* = 0 for flag set on match start or *x* = 1 for flag set at period boundary, and *y* = 0 for internal clock source or *y* = 1 for external clock source).
   f. Select the CP ratio (C*n*[UCPRE]).
   g. Enable the CP (write 1 to C*n*[UCPREN]).
4. Enable the GPC (write 1 to MCR[GPREN]).

#### 60.5.3.17.4 Timing example: AS1 and BS1 load

When the selected counter bus transitions from 2h to 1h:

- The UC loads AS1 and BS1 as shown in the following figure.
- This event defines the counter period boundary.

The UC loads the values you write to AS2 or BS2 within period *n* into AS1 or BS1, respectively, and uses them to generate matches in period *n*+1.

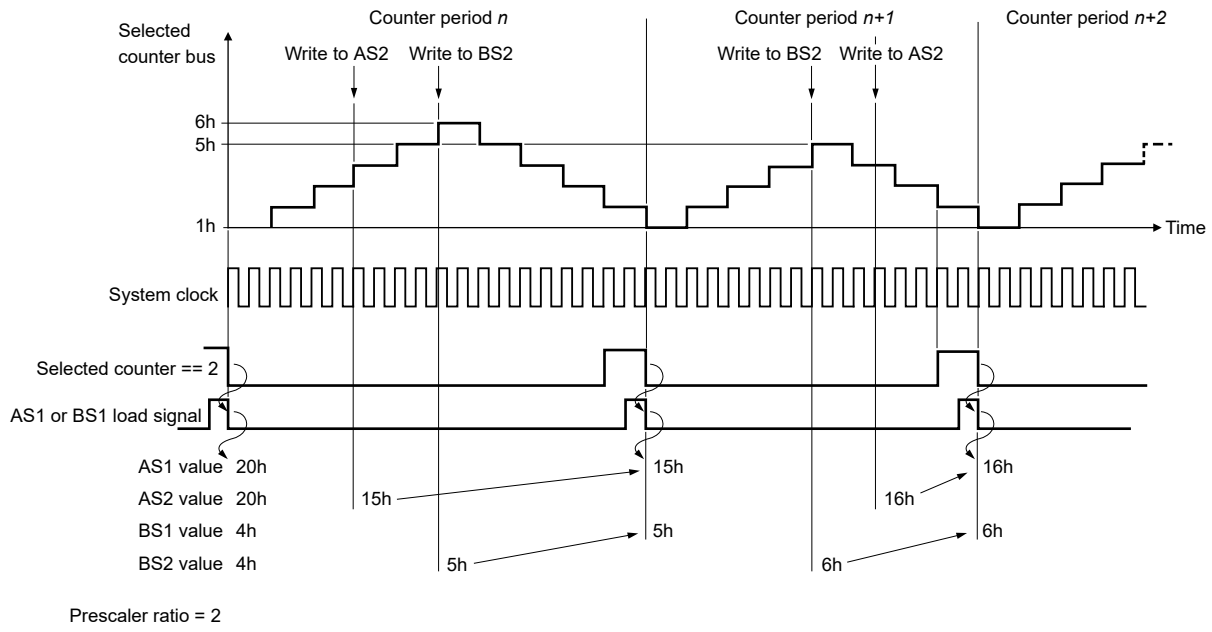



**Figure 324. Timing example: AS1 and BS1 load**

### 60.5.3.17.5 Operation sequence with leading edge dead time insertion

When operating with leading edge dead time insertion:

1. When the first AS1 match occurs, the UC writes 1h to the internal counter.
2. When a match between BS1 and the internal timebase occurs, the UC drives the value of Edge Polarity (C*n*[EDPOL]) on the output flip-flop.
3. When the next match between AS1 and the selected timebase occurs, the UC drives the complement of Edge Polarity (C*n*[EDPOL]) on the output flip-flop.

The UC repeats the above sequence continuously.

You must not allow the internal counter to reach 0h as the result of a rollover. To avoid this, you must not write a value greater than

2 x ((external Count Up limit) – A*n*[A])

into B*n*[B].

See Timing example: leading dead time insertion.

### 60.5.3.17.6 Timing example: leading dead time insertion

This example illustrates two cycles of a center-aligned PWM signal. AS1 and BS1 values change within the same period, which allows you to change the duty cycle and dead time values at the same time.

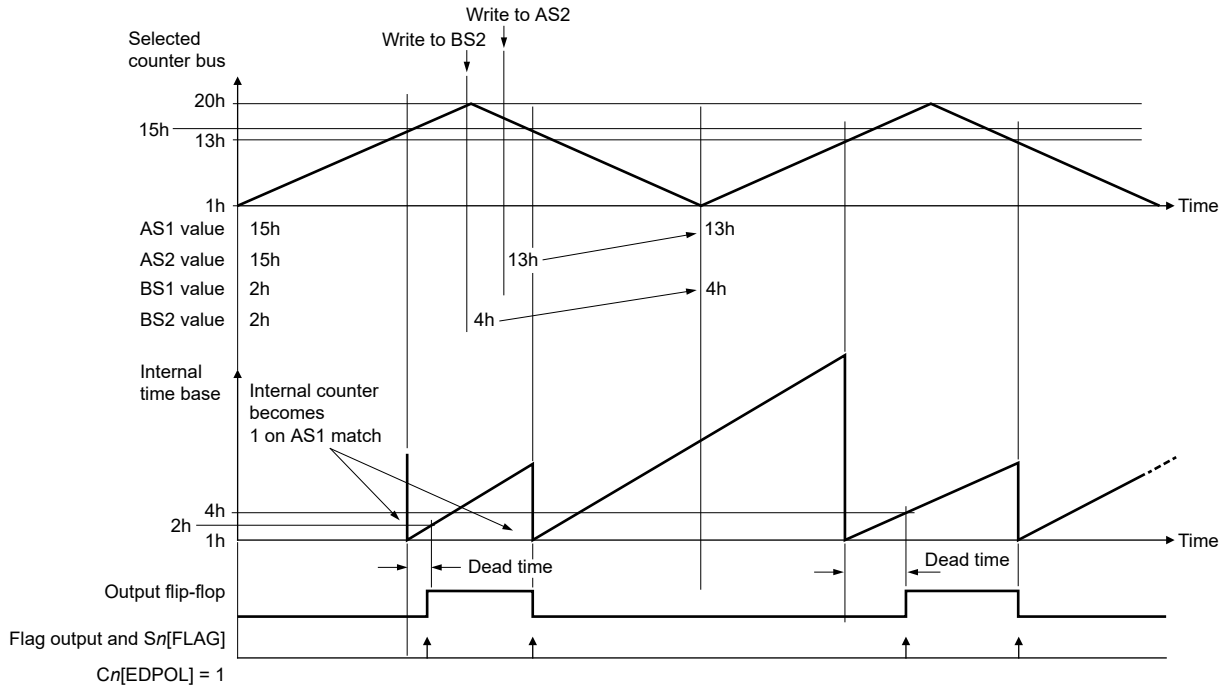



**Figure 325. Timing example: leading dead time insertion**

### 60.5.3.17.7 Operation sequence with trailing edge dead time insertion

When operating with trailing edge dead time insertion:

1. When the first match between AS1 and the selected timebase occurs, the UC:
   a. Drives the value of Edge Polarity (C*n*[EDPOL]) on the output flip-flop.
   b. Writes the internal counter (CNT*n*[C]) to 1h.
2. When the second match between AS1 and the selected timebase occurs, the UC:
   a. Writes the internal counter to 1h.
   b. Enables BS1 matches.
3. When a match between BS1 and the selected timebase occurs, the UC drives the complement of Edge Polarity (C*n*[EDPOL]) on the output flip-flop.

The UC repeats this sequence continuously.

### 60.5.3.17.8 Timing example: trailing dead time insertion

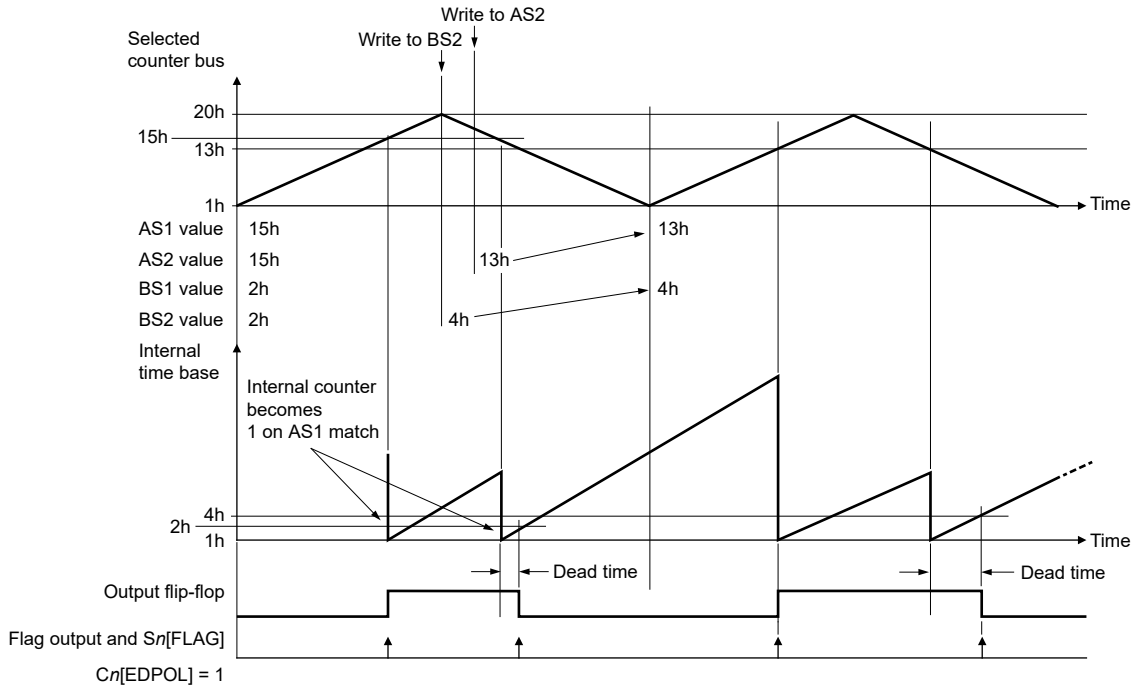



**Figure 326. Timing example: trailing dead time insertion**

### 60.5.3.17.9 Force match

Force Match A (C*n*[FORCMA]) and Force Match B (C*n*[FORCMB]) allow you to force the output flip-flop to the level corresponding to a match on comparators A or B, respectively. If subsequent matches occur on comparators A and B, the UC continues to generate PWM pulses regardless of the input capture flag (S*n*[FLAG]). The force-match operations do not set the input-capture flag (S*n*[FLAG]).

When you select leading edge dead time insertion:

- Force Match A drives the complement of Edge Polarity (C*n*[EDPOL]) on the output flip-flop.
- Force Match B drives the value of Edge Polarity (C*n*[EDPOL]) on the output flip-flop.

When you select trailing edge dead time insertion:

- Force Match A drives the value of Edge Polarity (C*n*[EDPOL]) on the output flip-flop.
- Force Match B drives the complement of Edge Polarity (C*n*[EDPOL]) on the output flip-flop.

Force Match operation does not write 1h to the internal timebase.

Force Match operation does not set the input-capture flag.

When you assert Force Match A and Force Match B at the same time, the UC drives the complement of Edge Polarity (C*n*[EDPOL]) on the output flip-flop. This is the equivalent of saying that Force Match A has precedence over Force Match B when you select leading edge dead time insertion, and Force Match B has precedence over Force Match A when you select trailing edge dead time insertion.

Force Match A and Force Match B have the same output transition behavior in both Freeze state and unfrozen state.

### 60.5.3.17.10 Duty cycle

You generate a duty cycle from 0% to 100% by writing appropriate values to AS1 and BS1 relative to the period of the external timebase.

To generate a 100% duty cycle waveform:

- Select trailing edge dead time insertion (see OPWMCB functions and MODE field values).
- Write 1h to Edge Polarity (C*n*[EDPOL]).
- The BS1 match must occur on or after the counter period boundary (external counter = 1).

You can also generate a 100% duty cycle waveform by writing 1h to AS1. However, if you do this before the UC enters OPWMCB mode, a 100% duty cycle may not occur in the first PWM period depending upon the pin state at mode entry.

To generate a 0% duty cycle waveform:

- AS1 must be greater than the maximum value of the selected counter period.
- The pin must start the current period with the complement value of Edge Polarity (C*n*[EDPOL]). If starting at 100% duty cycle, you can change Edge Polarity (C*n*[EDPOL]) to its complement by forcing the pin (see Force match).

You must write only non-zero values to AS1. If you write the maximum value of the external counter given by

(external counter period)/2

to AS1, the UC constantly drives the value of Edge Polarity (C*n*[EDPOL]) on the output flip-flop.

UC logic prevents matches from one period propagating to the next period. In trailing edge dead time insertion, the UC masks out BS1 matches that occur late in period *n*, so matches in period *n*+1 are unaffected and the output flip-flop does not transition.

### 60.5.3.17.11 Timing example: duty cycle

To generate a 100% duty cycle output waveform, write 4h to AS1 and 3h to BS1, as illustrated in this diagram. In this case, the trailing edge is at the boundary of counter period *n*+1, which actually belongs to period *n*+2 and therefore does not cause the output flip-flip to transition.

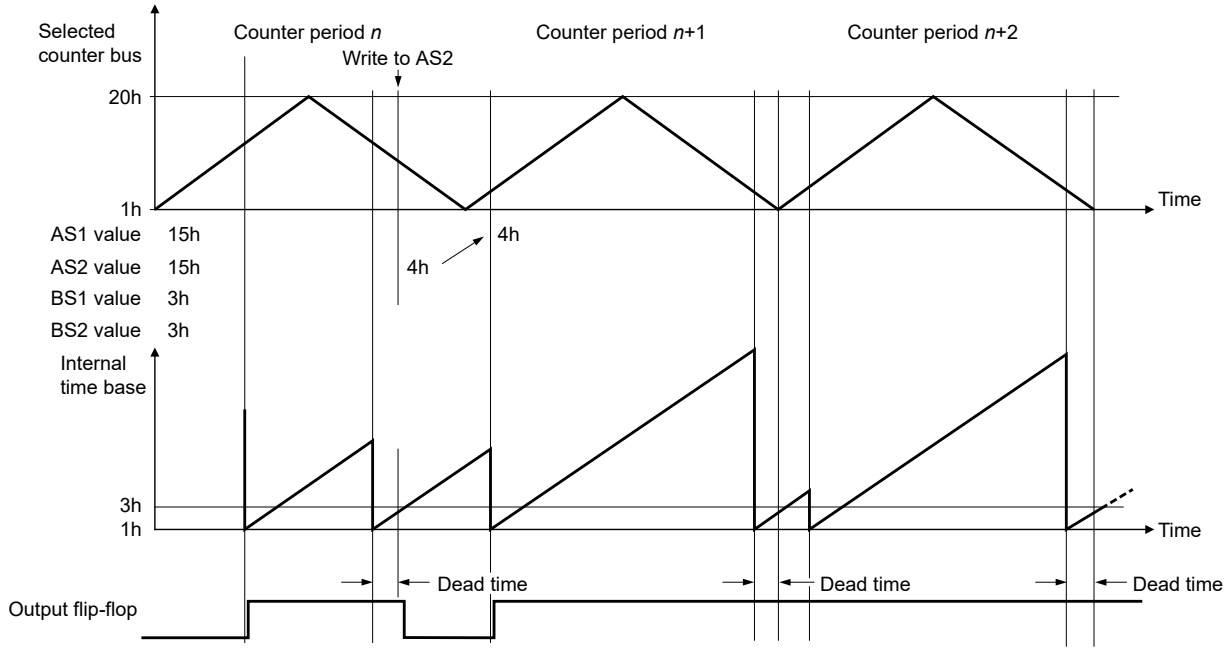



Figure 327. Timing example: duty cycle

### 60.5.3.17.12 Output disable

You can use Output Update Disable (write 0 to OUDIS[OU*n*]) to disable AS1 and BS1 updates. This allows you to update AS1 and BS1 in the same counter period and synchronize the loading of these registers with the loading of AS1 or BS1 in other channels.

Asserting Output Update Disable drives the complement of Edge Polarity (C*n*[EDPOL]) on output flip-flop. The internal channel matches continue to occur, and therefore the UC continues to set flags.

When you deassert Output Update Disable, the output flip-flop is again controlled by AS1 and BS1 matches. The UC transitions the output flip-flop only on system clock edges.

### 60.5.3.18 Output PWM Buffered (OPWMB) mode

#### 60.5.3.18.1 Overview

In this mode, the UC produces PWM pulses with programmable leading and trailing edge placement. You must select an external counter driven in MCB Up mode from one of the counter buses. AS1 and BS1 define the first and second edges, respectively. C*n*[EDPOL] defines the output signal polarity. If C*n*[EDPOL] = 0, a negative edge occurs when AS1 matches the selected counter bus and a positive edge occurs when BS1 matches the selected counter bus.

See OPWMB submodes and MODE field values for the values that you must write to C*n*[MODE] to enter this mode.

AS1 and BS1 are double-buffered and updated from AS2 and BS2, respectively, at the cycle boundary. The load operation is similar to the one that occurs in OPWFMB mode. See Timing example: AS1 and BS1 update and FLAGs for more information about AS1 and BS1 updates.

When the UC enters OPWMB mode, the UC drives C*n*[EDPOL] on the output flip-flop.

These rules apply to OPWMB mode:

- If the AS1 and BS1 matches occur at the same time within the same counter period, the BS1 match has precedence over the AS1 match.
- An AS1 = 0 match in period *n* has precedence over a BS1 match in period *n*-1.
- The UC masks out AS1 matches if they occur after a BS1 match within the same period.
- The UC loads any value that you write to AS2 or BS2 in period *n* into AS1 and BS1 at the following period boundary (assuming that the corresponding OU*n* field of Output Update Disable (OUDIS) is 0). Therefore, the UC uses the new values for AS1 and BS1 matches in period *n*+1.

Timing example: 100% to 0% duty cycle illustrates a waveform changing from 100% to 0% duty cycle. In this example, BS1 is programmed to the same value as the period of the external selected timebase. If you write a value smaller than 8h to BS1, you cannot achieve a 0% duty cycle by only changing AS1. Because BS1 matches have precedence over AS1 matches, the flag output transitions to the complement of C*n*[EDPOL] at a BS1 match. For another example, if you write 9h to BS1, a BS1 match does not occur, and the UC generates a 0% duty-cycle signal.

#### 60.5.3.18.2 OPWMB submodes and MODE field values

**Table 340. OPWMB submodes and MODE field values**

| When FLAG generation occurs | C*n*[MODE] (binary) |
|---|---|
| At a BS1 match | 110_0000 |
| At an AS1 or BS1 match | 110_0010 |

#### 60.5.3.18.3 Force match

Force Match A (C*n*[FORCMA]) and Force Match B (C*n*[FORCMB]) allow you to force the output flip-flop to the level corresponding to a match on comparators A or B, respectively. If subsequent matches occur on comparators A and B, the UC continues to generate PWM pulses regardless of the input capture flag (S*n*[FLAG]). The force-match operations do not set the input-capture flag (S*n*[FLAG]).

### 60.5.3.18.4 Matches and flag operation

Timing example: matches and flags illustrates the operation of OPWMB mode with an emphasis on AS1 and BS1 matches and the transition of the channel output pin. The output pin transitions are based on the negative edges of the AS1 and BS1 match signals. AS1 becomes zero in period *n*+1. In this case, the UC uses the match positive edge instead of the negative edge to transition the output flip-flop.

### 60.5.3.18.5 Timing example: matches and flags

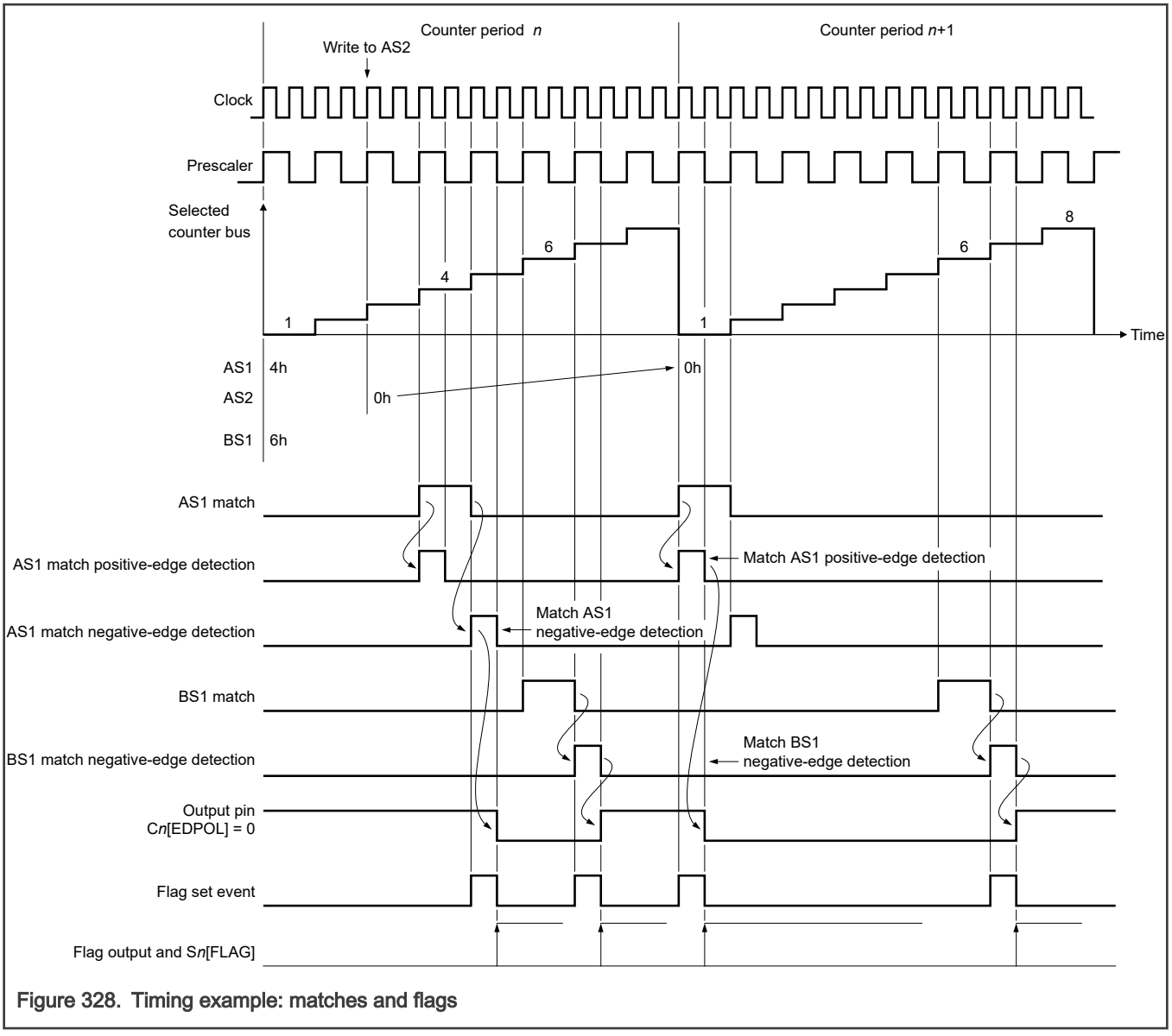



**Figure 328. Timing example: matches and flags**

### 60.5.3.18.6 0% duty cycle

Timing example: 0% duty cycle illustrates the channel operation for a 0% duty cycle. The AS1 match positive-edge signal occurs at the same time as the BS1 = 8h negative-edge signal. In this case, the AS1 match has precedence over the BS1 match, causing the output pin to remain at the value of C*n*[EDPOL] and generating a 0% duty-cycle signal.

### 60.5.3.18.7 Timing example: 0% duty cycle

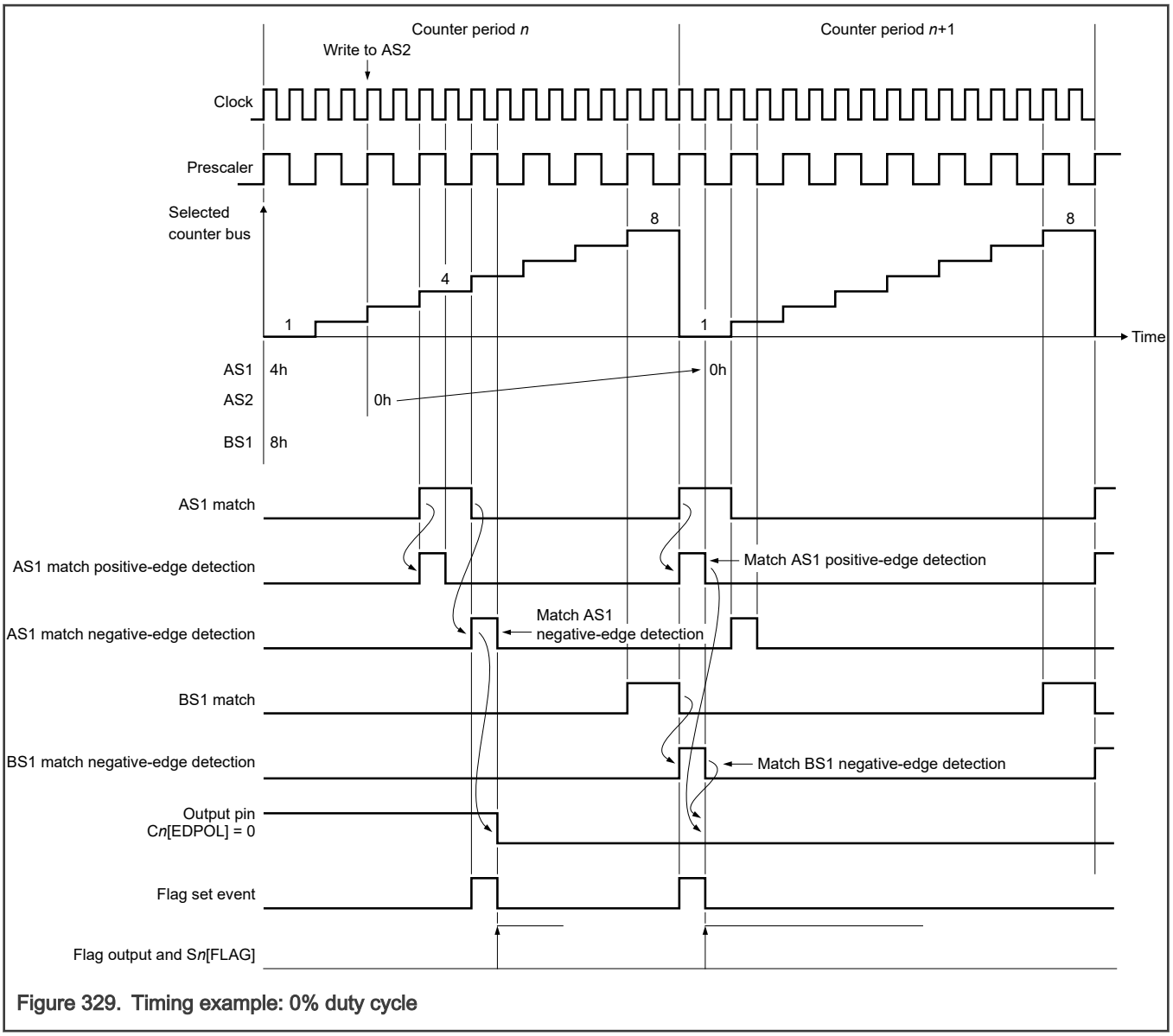



**Figure 329. Timing example: 0% duty cycle**

### 60.5.3.18.8 Active output disabled

Timing example: active output disabled illustrates the operation of the OPWMB mode with an emphasis on the assertion of the output-disable signal. The output disable forces a transition in the output pin to the value of C*n*[EDPOL]. The deassertion of the output-disable signal allows the output pin to transition at the following AS1 or BS1 match. The output disable does not modify the behavior of S*n*[FLAG]. A delay of one system clock cycle exists between the assertion of the output-disable signal and the transition of the output pin to C*n*[EDPOL].

### 60.5.3.18.9 Timing example: active output disabled

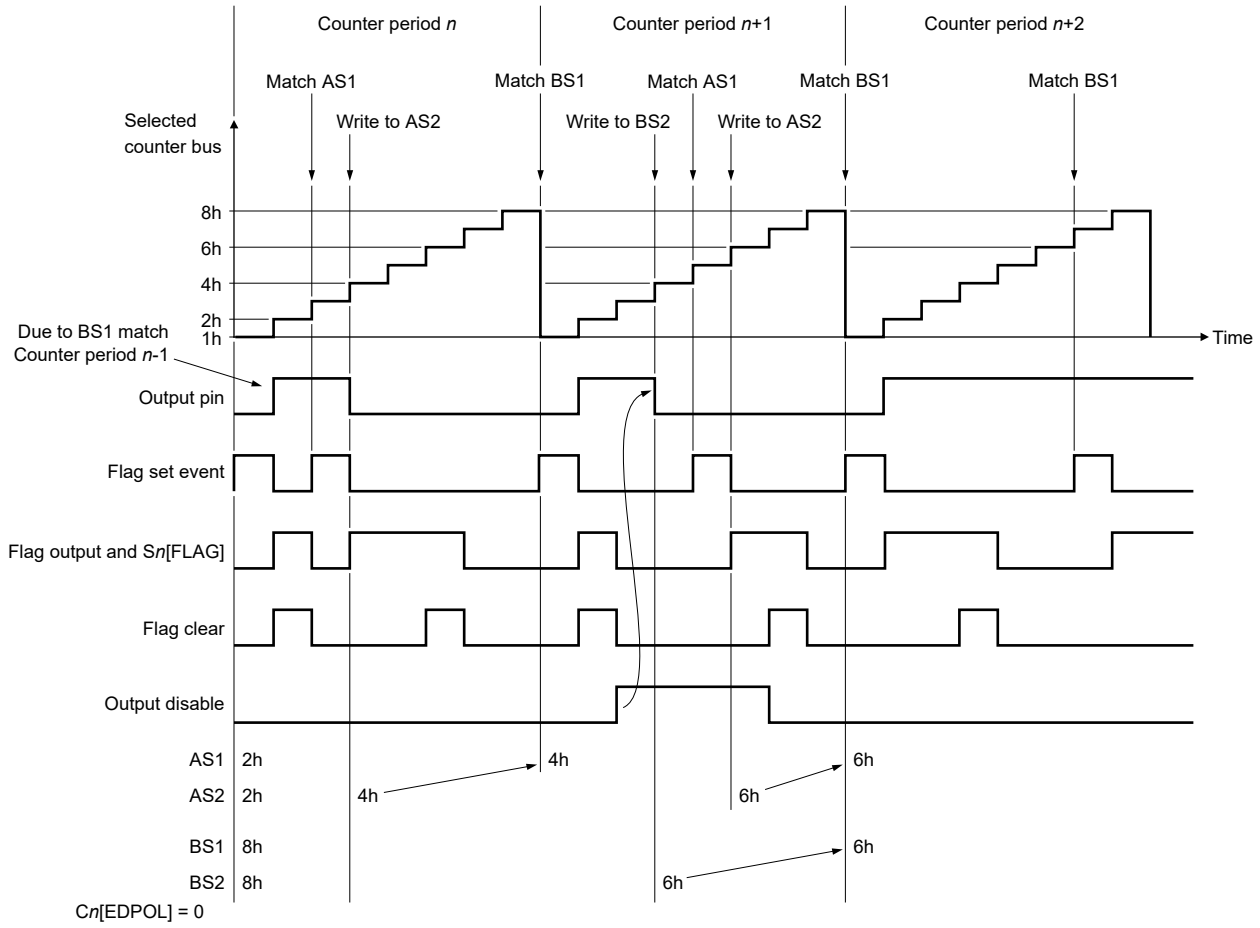



Figure 330. Timing example: active output disabled

### 60.5.3.18.10 Changing from 100% to 0% duty cycle

Timing example: 100% to 0% duty cycle illustrates a waveform changing from 100% to 0% duty cycle. In this example, BS1 has the same value as the period of the external selected timebase. If you write a value smaller than 8h to BS1, you cannot achieve a 0% duty cycle by only changing AS1. Because BS1 matches have precedence over AS1 matches, the output pin transitions to the complement of C*n*[EDPOL] at a BS1 match. For another example, if you write 9h to BS1, a BS1 match does not occur, and the UC generates a 0% duty-cycle signal.

#### 60.5.3.18.11 Timing example: 100% to 0% duty cycle

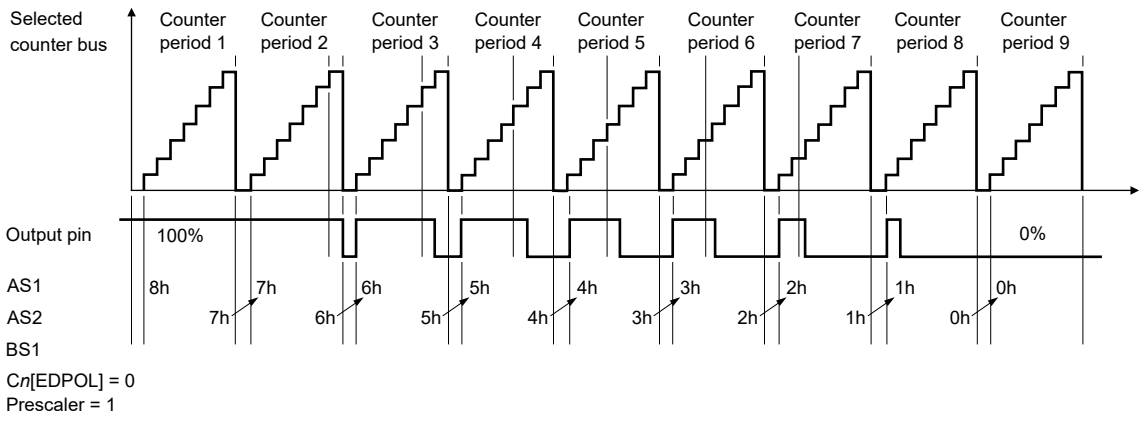



**Figure 331. Timing example: 100% to 0% duty cycle**

### 60.5.3.19 Output PWM with Trigger (OPWMT) mode

#### 60.5.3.19.1 Overview

In this mode, the UC generates PWM pulses where the period does not change while the UC outputs the signal, but the duty cycle changes. When the duty cycle changes, it must not create glitches. The mode supports multiple UCs executing in the same mode and sharing a common timebase. Each UC can have a fixed PWM leading-edge position with respect to the other UCs. It can also generate a trigger signal at any point in the period. eMIOS can output this trigger signal to initiate activity in other parts of the chip—for example, to start ADC conversions.

You must select an external counter, driven in either MC Up or MCB Up mode (see Modulus Counter Buffered (MCB) mode), from one of the counter buses.

AS1 in OPWMT mode, AS2 in OPWMT mode, and BS1 and BS2 in OPWMT mode describe the configuration and behavior of the individual shadow registers in this mode.

To account for the shift in the AS1-defined leading edge of the waveform, the BS1-defined trailing edge may roll over into the next period. This means that a match against BS1 does not have to be qualified by a match in AS1. This also means that if you incorrectly write a value less than AS1 to BS1, the output persists over a period boundary until the UC encounters the BS1 value.

This mode provides a buffered update of the trailing edge by updating BS1 with BS2 contents only at a match of AS1.

The positive edges of the AS1, BS1, and AS2 match signals drive the output pin and flag transitions. See Timing example: matches and flags for details on positive-edge matches.

The UC sets the input capture flag (S*n*[FLAG]) only at an AS2 match. An AS1, BS1, or BS2 match has no effect on the flag.

When the UC enters OPWMT mode, it drives the complement of C*n*[EDPOL] on the output flip-flop.

Timing example: OPWMT illustrates the UC running in OPWMT mode with trigger-event generation and duty-cycle update after the next-period update.

### 60.5.3.19.2 Timing example: OPWMT

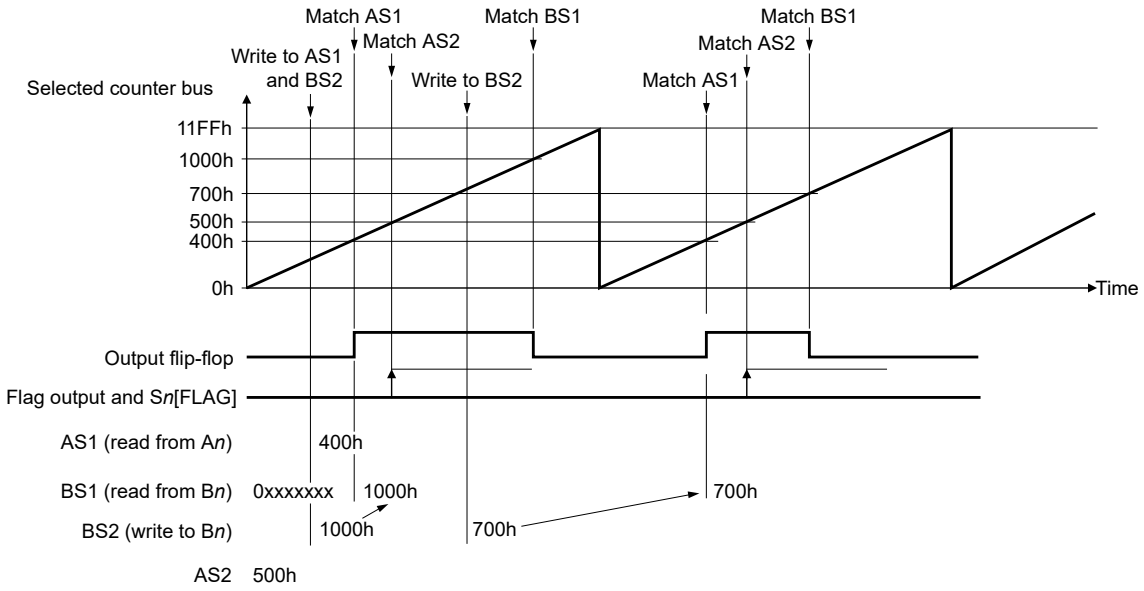



**Figure 332. Timing example: OPWMT**

### 60.5.3.19.3 AS1 in OPWMT mode

AS1 defines the leading edge of the PWM pulse output, and therefore the beginning of the PWM period. Using this fact, you can ensure that, when using a shared timebase, the leading edges of multiple UCs in OPWMT mode occur at specific times with respect to the other UCs. You can also introduce a fixed offset for each UC, which is particularly useful in the generation of lighting control signals. For this application, to reduce or eliminate noise generation, you can shift the PWM pulse of each UC with respect to the selected timebase by adjusting the AS1 value for each channel. The value you write to AS1 must be within the range of the selected timebase. Shadow registers (AS1, AS2, BS1, and BS2) loaded with 0h do not produce matches if the channel driving the timebase is in MCB mode.

AS1 is not buffered, because the shift of a PWM channel must not change while the UC generates the PWM signal. If you modify AS1, it updates immediately and one PWM pulse could be lost.

The UC compares AS1 to the selected timebase. When a match on AS1 occurs, the UC drives C*n*[EDPOL] on the output flip-flop. When a match occurs on comparator B, the UC drives the complement of C*n*[EDPOL] on the output flip-flop.

### 60.5.3.19.4 AS2 in OPWMT mode

AS2 defines the generation of a trigger event within the PWM period. The value you write to AS2 must be within the range of the selected timebase—otherwise the UC does not generate a trigger. A match on the comparator asserts the input-capture flag, but the match has no effect on the PWM output signal generation. The typical setup to obtain a trigger with a flag is to enable DMA and drive the DMA acknowledgement or DMA completion input high.

AS2 is not buffered, and therefore the UC updates it immediately. If the UC is running when you make a change, this could cause either of the following:

- The loss of one trigger event
- The generation of two trigger events within the same period

You access AS2 by reading or writing ALTA*n*[ALTA].

### 60.5.3.19.5 BS1 and BS2 in OPWMT mode

You access BS1 and BS2 using B*n*[B]:

- When you read B*n*[B], you read BS1.
- When you write to B*n*[B], you write to BS2.

BS1 defines the trailing edge of the PWM pulse output, and therefore the duty cycle of the PWM signal. To synchronize the BS1 update with the PWM signal and to ensure a correct output pulse generation, the BS2→BS1 transfer occurs at every AS1 match. This behavior is the same as in OPWM mode with next-period update.

Output Update Disable (OUDIS) affects transfers between BS2 and BS1 only.

#### 60.5.3.19.6 Force match

At any time, Force Match A (C*n*[FORCMA]) and Force Match B (C*n*[FORCMB]) allow you to force the output flip-flop to the level corresponding to a match on AS1 or BS1, respectively. Similarly to a BS1 match, Force Match B changes the internal counter to 1h. The force-match operations do not set the input-capture flag (S*n*[FLAG]). Any match resulting from writing 1 to FORCMA or FORCMB has priority over any simultaneous match regarding output pin transitions. BS2→BS1 transfer at an A match is not inhibited by writing 1 simultaneously to both FORCMA or FORCMB. If you write 1 to both FORCMA and FORCMB simultaneously, the output pin acquires the complement of C*n*[EDPOL], as if AS1 and BS1 had the same value. FORCMA assertion causes the UC to process the BS2→BS1 transfer as a regular A match comparison and not due to FORCMA assertion, regardless of FORCMB assertion. If subsequent matches occur on comparators AS1 and B, the UC continues to generate PWM pulses, regardless of the state of S*n*[FLAG].

#### 60.5.3.19.7 Duty cycle

To achieve 0% duty cycle, you must write the same value to AS1 and BS2. When a simultaneous match on comparators A and B occurs, the UC drives the complement of C*n*[EDPOL] on the output flip-flop at every period.

To achieve 100% duty cycle, you must write a value to BS1 that is greater than the maximum value of the selected timebase. The maximum counter value for the timebase is FF_FFFEh for a 24-bit counter and FFFEh for a 16-bit counter. When a match on comparator AS1 occurs, the UC drives C*n*[EDPOL] on the output flip-flop at every period. The match at comparator A still triggers the BS2→BS1 transfer.

Timing example: 0% duty cycle illustrates the UC running in OPWMT mode with trigger-event generation and a 0% duty cycle.

Timing example: 100% duty cycle illustrates the UC running in OPWMT mode with trigger-event generation and a 100% duty cycle.

### 60.5.3.19.8 Timing example: 0% duty cycle

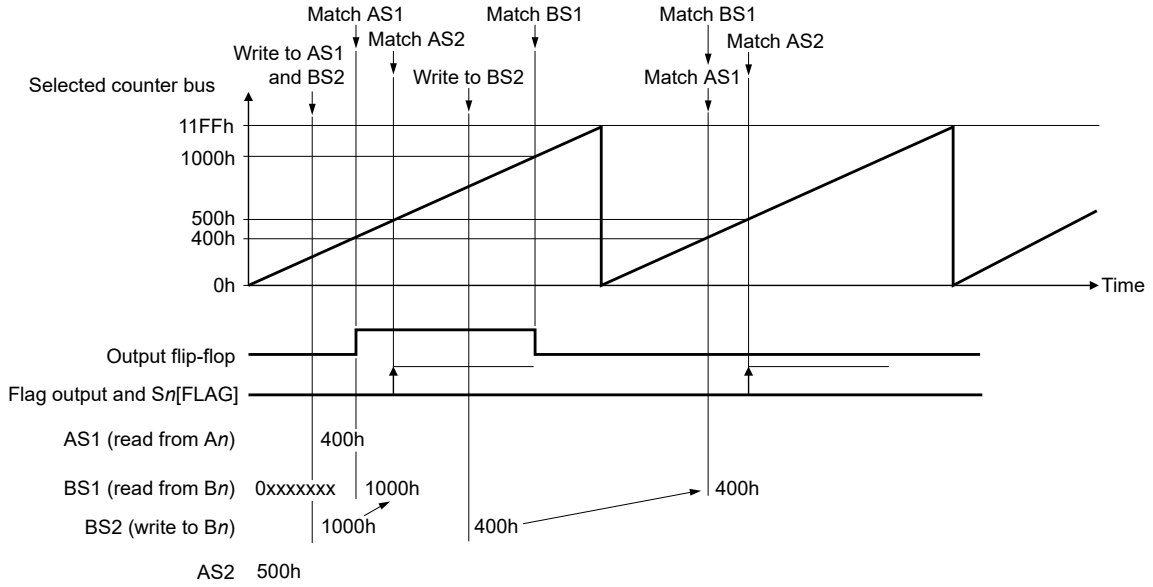



**Figure 333. Timing example: 0% duty cycle**

### 60.5.3.19.9 Timing example: 100% duty cycle

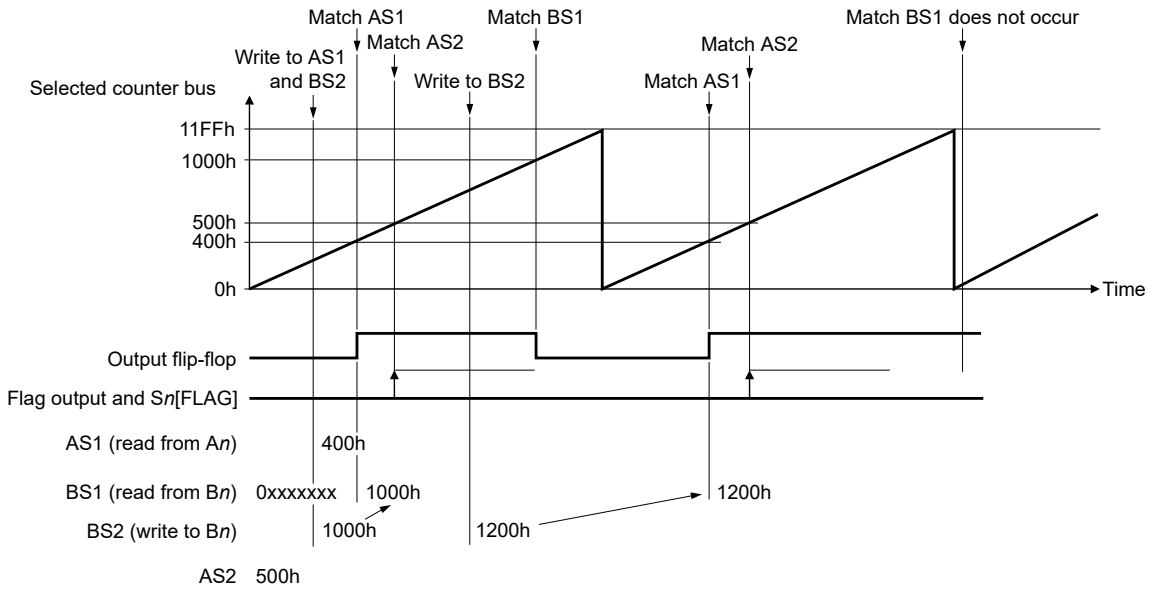



**Figure 334. Timing example: 100% duty cycle**

### 60.5.3.19.10 Output disable

As with other UC modes, OPWMT mode implements the output-disable function. When C*n*[ODIS] = 1, assertion of the output-disable input causes the UC to drive C*n*[EDPOL] on the output. Other than the fixed output, the UC continues to operate normally. When the output-disable signal deasserts, the UC returns to full normal operation.

### 60.5.3.20 Programmable input filter (PIF)

#### 60.5.3.20.1 Overview

The PIF specifies the minimum input pulse width, in clock cycles, that can pass through the filter. It supports bypass operation (C*n*[IF]), with the input signal synchronized before arriving at the digital filter, or from 2 to 16 clock cycles in powers of 2.

#### 60.5.3.20.2 Counter

The PIF is a 5-bit programmable up counter. The selected clock source increments the up counter after the number of clock cycles specified by the input filter (C*n*[IF]0.

The system clock synchronizes the input signal. When this signal changes state, the counter begins incrementing. As long as the new signal state is stable, the counter continues incrementing. If the counter overflows, eMIOS validates the new signal value. In this case, it transmits the value as a pulse edge to the edge detector. If the opposite edge appears on the signal before validation (overflow), eMIOS resets the counter. At the next transition, the counter starts incrementing again. A glitch does not occur on the edge detector for any pulse shorter than the full range of the masked counter.

Neither Freeze state nor deasserted GTBE disables the filter.

#### 60.5.3.20.3 UC PIF logic

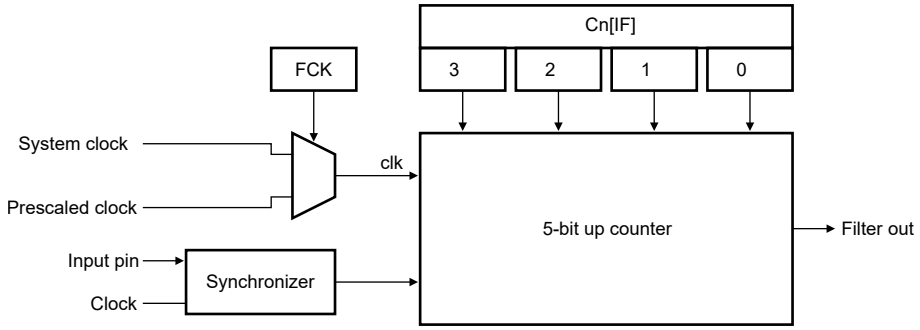



Figure 335. UC PIF logic

#### 60.5.3.20.4 Timing example: UC PIF at 4 cycles

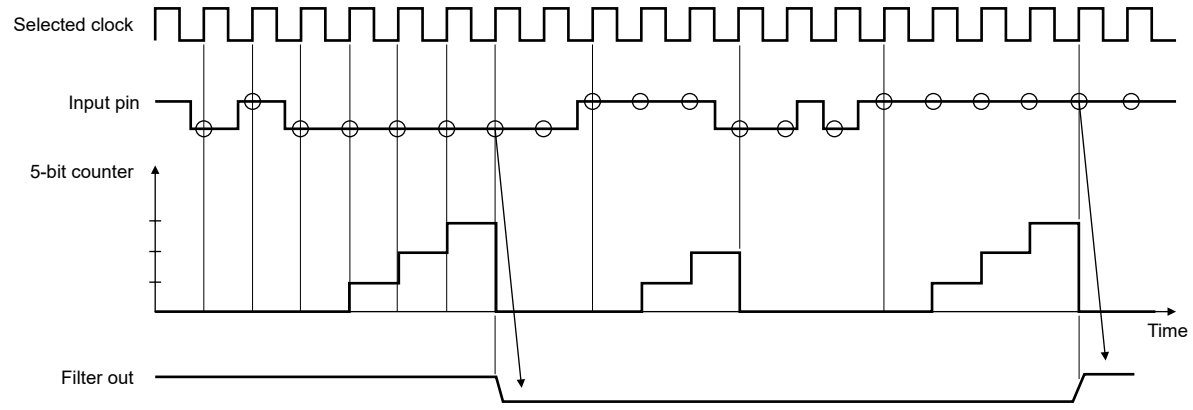



Figure 336. Timing example: UC PIF at 4 cycles

#### 60.5.3.21 UC clock prescaler (CP)

eMIOS divides the GCP output signal by the UC CP (C*n*[UCPRE] + 1) to generate a clock enable for the UC internal counter. See Global clock prescaler (GCP). You enable the UC CP by writing 1 to C*n*[UCPREN]. If you disable the CP (write 0 to the same field), eMIOS stops the UC internal counter.

To ensure safe operation and avoid glitches, you must perform the following steps to change the UC CP:

1. Disable the GCP (write 0 to MCR[GPREN]).
2. Disable the UC CP (write 0 to C*n*[UCPREN]).
3. Specify the new prescaler value (write CP – 1 to C*n*[GPRE].
4. Enable the UC CP (write 1 to C*n*[UCPREN]).
5. Enable the GCP (write 1 to MCR[GPREN]).

### 60.5.4 Peripheral bus interface unit (BIU)

The BIU provides the interface between the IIB and the peripheral bus, which allows communication among all submodules and the IP interface. The BIU supports 8-, 16-, and 32-bit accesses performed over a 32-bit data bus in a single clock cycle.

### 60.5.5 Global clock prescaler (GCP)

eMIOS divides the system clock by the GCP (MCR[GPRE] + 1) and routes the resulting prescaled clock output to the channel CPs. You enable the GCP by writing 1 to MCR[GPREN]. If you disable the GCP (write 0 to the same field), eMIOS clears the prescaler counter and stops the prescaler clock.

To ensure safe operation and avoid glitches, you must perform the following steps to change the GCP:

1. Disable the GCP (write 0 to MCR[GPREN]).
2. Specify the new prescaler value (write GCP – 1 to MCR[GPRE].
3. Enable the GCP (write 1 to MCR[GPREN]).

### 60.5.6 Freeze and chip Debug mode

#### 60.5.6.1 Enter and exit Freeze state

When the chip is in Debug mode (the chip Debug signal is asserted), you can freeze individual UCs. For the effects of Freeze state on eMIOS functions, see Freeze effects.

To put a channel in Freeze state, you must meet all of the following conditions:

- The chip must assert Debug.
- Prepare eMIOS for the Freeze state (write 1 to MCR[FRZ]).
- Enable Freeze for each channel you want to freeze:
  - Write 1 to C*n*[FREN].

A channel exits Freeze state when any of the following conditions occur:

- The chip deasserts Debug.
- You take eMIOS out of the Freeze state (write 0 to MCR[FRZ]).
- You disable Freeze for the channel:
  - write 0 to C*n*[FREN].

#### 60.5.6.2 Freeze effects

| Function | Effects of freeze |
|---|---|
| GCP | No effect |
| UC | • Halts counter, capture, and compare operation.<br>• Freezes UC in its current state.<br>• Makes all registers accessible.<br>• In output modes, force match remains operational, allowing software to force the output to the desired level.<br>• In input modes, the UC ignores input events.<br>• When eMIOS exits Freeze state then UC operations resume, but they may be inconsistent until the UC reenters GPIO mode. |
| STAC | No effect |
| BIU | No effect |

## 60.6 Initialization information

When initializing eMIOS, take the following considerations into account.

For eMIOS behavior on reset, see Reset.

Before changing the UC operating mode, you must:

1. Put the UC in GPIO mode (write 0b or 1b to C*n*[MODE]).
2. Write the correct values for the next operating mode to A*n* and B*n*.
3. Put the UC in the new operating mode (write the appropriate value to C*n*[MODE]).

If you change a UC from one mode to another without performing this procedure, the first operation cycle of the selected timebase can be random; that is, matches can occur in random time because you did not update A*n* and B*n* with correct values before the timebase matches the previous contents of either register.

**NOTE**

To to avoid causing an unexpected interrupt or DMA request when changing from GPIO to MCB mode, you must use the procedure described in Changing to MCB mode.

When you enable interrupts, any interrupt service routine must clear the flag before exiting.

## 60.7 Application information

### 60.7.1 Overview

All output operation modes can generate correlated output signals. You can disable updates of the output signals for each UC (OUDIS).

To guarantee that incrementing of the internal counters of correlated channels occurs in the same clock cycle, you must configure the internal CPs before enabling the GCP. If you configure the internal CPs after enabling the GCP, the internal counters may increment in the same ratio, but at different clock cycles.

You should drive the Output Disable Input signals with the flag signals of some UCs running in SAIC mode. When an output disable condition happens, the interrupt service routine must process the output channels before processing the channels running SAIC. This sequence avoids glitches in the output pins.

### 60.7.2 Timebase generation

In the MC and OPWFM modes with internal clock sources, you can modify the internal counter rate by configuring the CP ratio. Timebase with the fastest prescaler ratio illustrates a timebase with a prescaler ratio of one.

**NOTE**

The buffered modes, MCB and OPWFMB, behave differently.

If the prescaler ratio is greater than one or you select an external clock, the internal counter behaves in one of four different ways depending on the channel mode:

- In MC mode with Clear on Match Start and External Clock: Timebase generation with clear on match start and external clock.
- In MC mode with Clear on Match Start and Internal Clock source: Timebase generation with clear on match start and internal clock.
- In MC mode with Clear on Match End: Timebase generation with clear on match end.
- In OPWFM mode: Timebase generation with clear on match start and internal clock. The internal counter clears at the start of the match signal, skips the next prescaled clock edge, and then increments in the subsequent prescaled clock edge.

### 60.7.3 Timebase with the fastest prescaler ratio

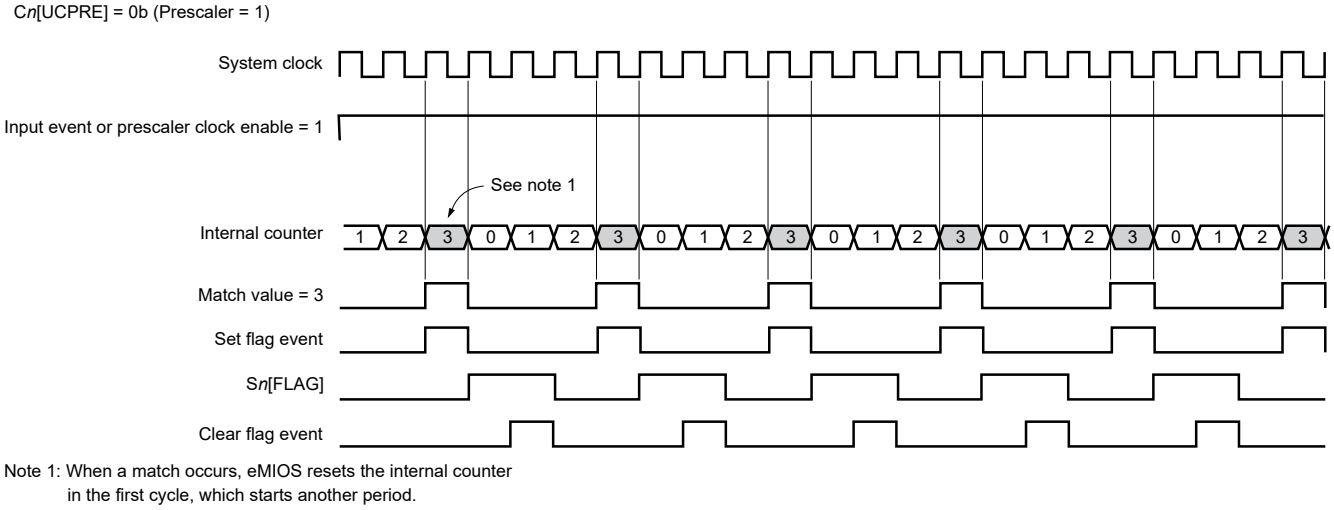
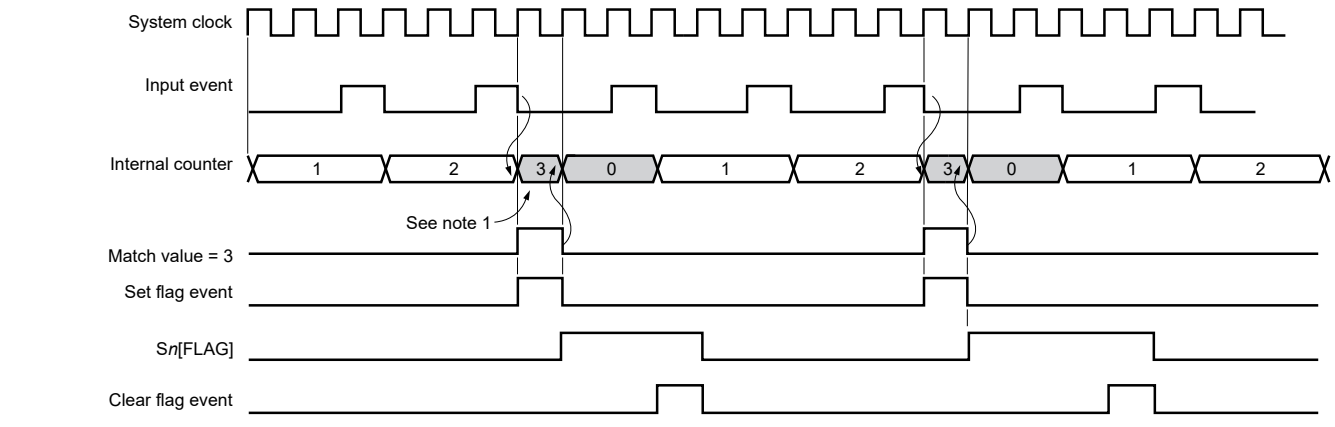



**Figure 337. Timebase with fastest prescaler ratio**

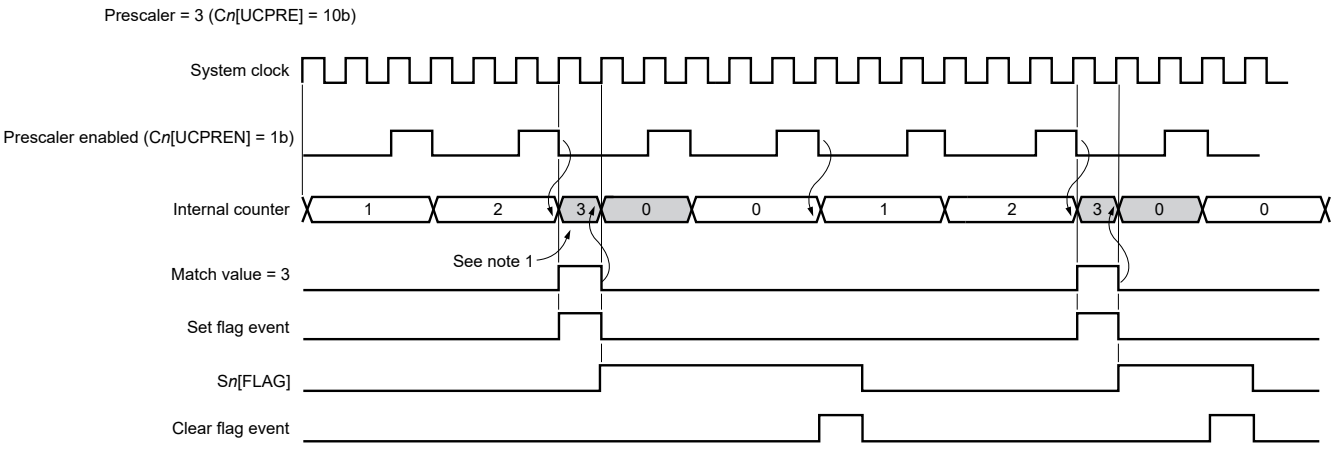
## 60.7.4 Timebase generation with clear on match start and external clock




Note 1: When a match occurs, eMIOS resets the internal counter in the first clock cycle.
The counter starts counting at the next edge of the prescaler clock enable.

**Figure 338. Timebase generation with clear on match start and external clock**

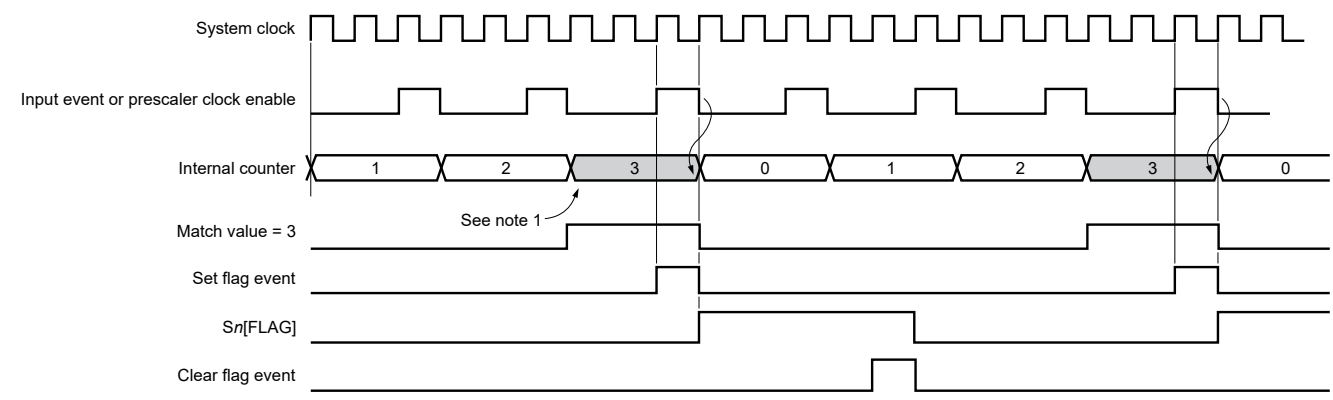
## 60.7.5 Timebase generation with clear on match start and internal clock




Note 1: When a match occurs, eMIOS clears the internal counter in the first clock cycle.
The counter starts counting again only after the second edge of the prescaled clock.

**Figure 339. Timebase generation with clear on match start and internal clock**

### 60.7.6 Timebase generation with clear on match end




Note 1: The match occurs only when the input event or prescaler clock enable is active.
When the match occurs, eMIOS resets the internal counter immediately.

**Figure 340. Timebase generation with clear on match end**

### 60.7.7 Coherent accesses

Reading A*n*[A] again in the same period as the last read of B*n*[B] results in incoherent results if the last read of B*n*[B] occurred after a disabled BS2→BS1 transfer.

### 60.7.8 Initializing channels and modes

Perform the following basic steps for output mode startup. This procedure assumes the channels are initially in GPIO mode:

1. Disable the GPC (write 0 to MCR[GPREN]).
2. For each timebase UC:
    a. Disable the CP (write 0 to C*n*[UCPREN]).
    b. Write an initial value to the internal counter (CNT*n*[C]).
    c. Write initial values to A*n*[A] and B*n*[B].
    d. Configure the UC for MC or MCB Up mode (write the appropriate value to C*n*[MODE]).
    e. Select the CP ratio (C*n*[UCPRE]).
    f. Enable the CP (write 1 to C*n*[UCPREN]).
3. For each output UC:
    a. Disable the CP (write 0 to C*n*[UCPREN]).
    b. Write initial values to A*n*[A] and B*n*[B].
    c. Select the timebase input (C*n*[BSL]).
    d. Specify the output mode (write the appropriate value to C*n*[MODE]).
    e. Select the same CP ratio as the timebase UC (C*n*[UCPRE]).
    f. Enable the CP (write 1 to C*n*[UCPREN]).
4. Enable the GPC (write 1 to MCR[GPREN]).
5. Enable GBTE (write 1 to MCR[GBTE]).

The timebase channel and the output channel may be the same for some applications such as in OPWFM or OPWFMB mode, or whenever the output channel drives the timebase itself.

You may configure flags at any time.

## 60.8 Memory maps and registers

### 60.8.1 Overall address map organization

All address locations not explicitly mentioned in this table are reserved. When you access an absent register, absent channel, or reserved address space, the transfer terminates with an error.

**Table 341. Overall address map organization**

| Base address (hex) | Description |
|---|---|
| 0 | Module Configuration (MCR) |
| 4 | Global Flag (GFLAG) |
| 8 | Output Update Disable (OUDIS) |
| C | Disable Channel (UCDIS) |
| 20 | Channels 0–7 |
| 120 | Channels 8–15 |
| 220 | Channels 16–23 |

### 60.8.2 UC memory overview

**Table 342. UC memory overview**

| Offset from UC *n* base address (hex) | Description |
|---|---|
| 0 | UC A n (A0 - A23) |
| 4 | UC B n (B0 - B23) |
| 8 | UC Counter n (CNT0 - CNT23) |
| C | UC Control n (C0 - C23) |
| 10 | UC Status n (S0 - S23) |
| 14 | Alternate Address n (ALTA0 - ALTA23) |
| 18 | UC Control 2 n (C2_0 - C2_23) |
| 1C–1F | Reserved |

### 60.8.3 AS1, AS2, BS1, and BS2 shadow registers

When you use eMIOS UC modes, you rely heavily on four shadow registers:

- AS1
- AS2
- BS1
- BS2

You use A*n*[A], B*n*[B], and ALTA*n*[ALTA] to access these shadow registers:

- Each instance of A*n*[A] has a separate associated instance of AS1 and AS2. In some UC modes, you also use ALTA*n*[ALTA] to access AS2.
- Each instance of B*n*[B] has a separate associated instance of BS1 and BS2.

See Relationship between A*n*, B*n*, ALTA*n*, and shadow registers.

Each UC mode uses the AS1-AS2 and BS1-BS2 shadow register pairs for various match and capture functions, as described in Unified channels (UC).

A reset clears all shadow registers.

Assignment of values for A*n*[A], B*n*[B], and ALTA*n*[ALTA] describes how you write to and read from the shadow registers for all operation modes. Values not listed are reserved. For more information, see Unified channels (UC).

## 60.8.4 Relationship between A*n*, B*n*, ALTA*n*, and shadow registers

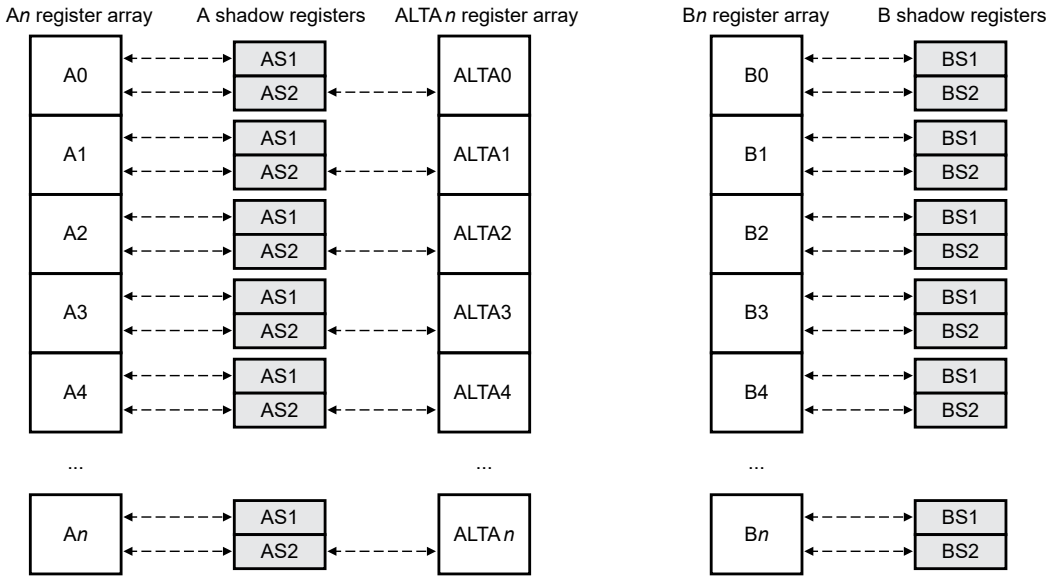



**Figure 341. Relationship between A*n*, B*n*, ALTA*n*, and shadow registers**

## 60.8.5 Assignment of values for A*n*[A], B*n*[B], and ALTA*n*[ALTA]

This table describes how you write to and read from the following registers for all operation modes:

- UC A n (A0 - A23)
- UC B n (B0 - B23)
- Alternate Address n (ALTA0 - ALTA23)

Values not listed are reserved. For more information, see Unified channels (UC).

**Table 343. Assignment of values for A*n*[A], B*n*[B], and ALTA*n*[ALTA]**

| Operation mode | Register access | | | | | |
|---|---|---|---|---|---|---|
| | A*n* | | B*n* | | ALTA*n* | |
| | Write | Read | Write | Read | Write | Read |
| GPIO | AS1 and AS2[1] | AS1 | BS1 and BS2[2] | BS1 | AS2 | AS2 |

*Table continues on the next page...*

Table 343. Assignment of values for A*n*[A], B*n*[B], and ALTA*n*[ALTA] (continued)

| Operation mode | Register access | | | | | |
|---|---|---|---|---|---|---|
| | A*n* | | B*n* | | ALTA*n* | |
| | Write | Read | Write | Read | Write | Read |
| SAIC[3] | — | AS2 | BS2 | BS2 | — | — |
| SAOC[3] | AS2 | AS1 | BS2 | BS2 | — | — |
| IPWM | — | AS2 | — | BS1 | — | — |
| IPM | — | AS2 | — | BS1 | — | — |
| DAOC | AS2 | AS1 | BS2 | BS1 | — | — |
| PEA | AS1 | AS2 | — | BS1 | — | — |
| PEC[3] | AS1 | AS1 | BS1 | BS1 | — | AS2 |
| WPTA | AS1 | AS1 | BS1 | BS1 | — | AS2 |
| MC[3] | AS2 | AS1 | BS2 | BS2 | — | — |
| OPWFM | AS2 | AS1 | BS2 | BS1 | — | — |
| OPWMC | AS2 | AS1 | BS2 | BS1 | — | — |
| OPWM | AS2 | AS1 | BS2 | BS1 | — | — |
| OPWMT | AS1 | AS1 | BS2 | BS1 | AS2 | AS2 |
| MCB[3] | AS2 | AS1 | BS2 | BS1 | — | — |
| OPWFMB | AS2 | AS1 | BS2 | BS1 | — | — |
| OPWMCB | AS2 | AS1 | BS2 | BS1 | — | — |
| OPWMB | AS2 | AS1 | BS2 | BS1 | — | — |

1. This operation writes the same value to AS1 and AS2.
2. This operation writes the same value to BS1 and BS2.
3. These modes do not require UC B n (B0 - B23), but you can still use it to access BS2.

## 60.8.6 eMIOS register descriptions

All control registers are 32 bits wide. Data fields are 16 bits wide. There are 24 UCs, as mapped in Counter buses, channels, and timebase sources.

### 60.8.6.1 eMIOS memory map

eMIOS_0 base address: 4008_8000h

eMIOS_1 base address: 4008_C000h

eMIOS_2 base address: 4009_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Module Configuration (MCR) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 4h | Global Flag (GFLAG) | 32 | R | 0000_0000h |
| 8h | Output Update Disable (OUDIS) | 32 | RW | 0000_0000h |
| Ch | Disable Channel (UCDIS) | 32 | RW | 0000_0000h |
| 20h | UC A 0 (A0) | 32 | RW | 0000_0000h |
| 24h | UC B 0 (B0) | 32 | RW | 0000_0000h |
| 28h | UC Counter 0 (CNT0) | 32 | RW | 0000_0000h |
| 2Ch | UC Control 0 (C0) | 32 | RW | 0000_0000h |
| 30h | UC Status 0 (S0) | 32 | RW | See section |
| 34h | Alternate Address 0 (ALTA0) | 32 | RW | 0000_0000h |
| 38h | UC Control 2 0 (C2_0) | 32 | RW | 0000_0000h |
| 40h | UC A 1 (A1) | 32 | RW | 0000_0000h |
| 44h | UC B 1 (B1) | 32 | RW | 0000_0000h |
| 48h | UC Counter 1 (CNT1) | 32 | RW | 0000_0000h |
| 4Ch | UC Control 1 (C1) | 32 | RW | 0000_0000h |
| 50h | UC Status 1 (S1) | 32 | RW | See section |
| 54h | Alternate Address 1 (ALTA1) | 32 | RW | 0000_0000h |
| 58h | UC Control 2 1 (C2_1) | 32 | RW | 0000_0000h |
| 60h | UC A 2 (A2) | 32 | RW | 0000_0000h |
| 64h | UC B 2 (B2) | 32 | RW | 0000_0000h |
| 68h | UC Counter 2 (CNT2) | 32 | RW | 0000_0000h |
| 6Ch | UC Control 2 (C2) | 32 | RW | 0000_0000h |
| 70h | UC Status 2 (S2) | 32 | RW | See section |
| 74h | Alternate Address 2 (ALTA2) | 32 | RW | 0000_0000h |
| 78h | UC Control 2 2 (C2_2) | 32 | RW | 0000_0000h |
| 80h | UC A 3 (A3) | 32 | RW | 0000_0000h |
| 84h | UC B 3 (B3) | 32 | RW | 0000_0000h |
| 88h | UC Counter 3 (CNT3) | 32 | RW | 0000_0000h |
| 8Ch | UC Control 3 (C3) | 32 | RW | 0000_0000h |
| 90h | UC Status 3 (S3) | 32 | RW | See section |
| 94h | Alternate Address 3 (ALTA3) | 32 | RW | 0000_0000h |
| 98h | UC Control 2 3 (C2_3) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| A0h | UC A 4 (A4) | 32 | RW | 0000_0000h |
| A4h | UC B 4 (B4) | 32 | RW | 0000_0000h |
| A8h | UC Counter 4 (CNT4) | 32 | RW | 0000_0000h |
| ACh | UC Control 4 (C4) | 32 | RW | 0000_0000h |
| B0h | UC Status 4 (S4) | 32 | RW | See section |
| B4h | Alternate Address 4 (ALTA4) | 32 | RW | 0000_0000h |
| B8h | UC Control 2 4 (C2_4) | 32 | RW | 0000_0000h |
| C0h | UC A 5 (A5) | 32 | RW | 0000_0000h |
| C4h | UC B 5 (B5) | 32 | RW | 0000_0000h |
| C8h | UC Counter 5 (CNT5) | 32 | RW | 0000_0000h |
| CCh | UC Control 5 (C5) | 32 | RW | 0000_0000h |
| D0h | UC Status 5 (S5) | 32 | RW | See section |
| D4h | Alternate Address 5 (ALTA5) | 32 | RW | 0000_0000h |
| D8h | UC Control 2 5 (C2_5) | 32 | RW | 0000_0000h |
| E0h | UC A 6 (A6) | 32 | RW | 0000_0000h |
| E4h | UC B 6 (B6) | 32 | RW | 0000_0000h |
| E8h | UC Counter 6 (CNT6) | 32 | RW | 0000_0000h |
| ECh | UC Control 6 (C6) | 32 | RW | 0000_0000h |
| F0h | UC Status 6 (S6) | 32 | RW | See section |
| F4h | Alternate Address 6 (ALTA6) | 32 | RW | 0000_0000h |
| F8h | UC Control 2 6 (C2_6) | 32 | RW | 0000_0000h |
| 100h | UC A 7 (A7) | 32 | RW | 0000_0000h |
| 104h | UC B 7 (B7) | 32 | RW | 0000_0000h |
| 108h | UC Counter 7 (CNT7) | 32 | RW | 0000_0000h |
| 10Ch | UC Control 7 (C7) | 32 | RW | 0000_0000h |
| 110h | UC Status 7 (S7) | 32 | RW | See section |
| 114h | Alternate Address 7 (ALTA7) | 32 | RW | 0000_0000h |
| 118h | UC Control 2 7 (C2_7) | 32 | RW | 0000_0000h |
| 120h | UC A 8 (A8) | 32 | RW | 0000_0000h |
| 124h | UC B 8 (B8) | 32 | RW | 0000_0000h |
| 128h | UC Counter 8 (CNT8) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 12Ch | UC Control 8 (C8) | 32 | RW | 0000_0000h |
| 130h | UC Status 8 (S8) | 32 | RW | See section |
| 134h | Alternate Address 8 (ALTA8) | 32 | RW | 0000_0000h |
| 138h | UC Control 2 8 (C2_8) | 32 | RW | 0000_0000h |
| 140h | UC A 9 (A9) | 32 | RW | 0000_0000h |
| 144h | UC B 9 (B9) | 32 | RW | 0000_0000h |
| 14Ch | UC Control 9 (C9) | 32 | RW | 0000_0000h |
| 150h | UC Status 9 (S9) | 32 | RW | See section |
| 154h | Alternate Address 9 (ALTA9) | 32 | RW | 0000_0000h |
| 158h | UC Control 2 9 (C2_9) | 32 | RW | 0000_0000h |
| 160h | UC A 10 (A10) | 32 | RW | 0000_0000h |
| 164h | UC B 10 (B10) | 32 | RW | 0000_0000h |
| 16Ch | UC Control 10 (C10) | 32 | RW | 0000_0000h |
| 170h | UC Status 10 (S10) | 32 | RW | See section |
| 174h | Alternate Address 10 (ALTA10) | 32 | RW | 0000_0000h |
| 178h | UC Control 2 10 (C2_10) | 32 | RW | 0000_0000h |
| 180h | UC A 11 (A11) | 32 | RW | 0000_0000h |
| 184h | UC B 11 (B11) | 32 | RW | 0000_0000h |
| 18Ch | UC Control 11 (C11) | 32 | RW | 0000_0000h |
| 190h | UC Status 11 (S11) | 32 | RW | See section |
| 194h | Alternate Address 11 (ALTA11) | 32 | RW | 0000_0000h |
| 198h | UC Control 2 11 (C2_11) | 32 | RW | 0000_0000h |
| 1A0h | UC A 12 (A12) | 32 | RW | 0000_0000h |
| 1A4h | UC B 12 (B12) | 32 | RW | 0000_0000h |
| 1ACh | UC Control 12 (C12) | 32 | RW | 0000_0000h |
| 1B0h | UC Status 12 (S12) | 32 | RW | See section |
| 1B4h | Alternate Address 12 (ALTA12) | 32 | RW | 0000_0000h |
| 1B8h | UC Control 2 12 (C2_12) | 32 | RW | 0000_0000h |
| 1C0h | UC A 13 (A13) | 32 | RW | 0000_0000h |
| 1C4h | UC B 13 (B13) | 32 | RW | 0000_0000h |
| 1CCh | UC Control 13 (C13) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1D0h | UC Status 13 (S13) | 32 | RW | See section |
| 1D4h | Alternate Address 13 (ALTA13) | 32 | RW | 0000_0000h |
| 1D8h | UC Control 2 13 (C2_13) | 32 | RW | 0000_0000h |
| 1E0h | UC A 14 (A14) | 32 | RW | 0000_0000h |
| 1E4h | UC B 14 (B14) | 32 | RW | 0000_0000h |
| 1ECh | UC Control 14 (C14) | 32 | RW | 0000_0000h |
| 1F0h | UC Status 14 (S14) | 32 | RW | See section |
| 1F4h | Alternate Address 14 (ALTA14) | 32 | RW | 0000_0000h |
| 1F8h | UC Control 2 14 (C2_14) | 32 | RW | 0000_0000h |
| 200h | UC A 15 (A15) | 32 | RW | 0000_0000h |
| 204h | UC B 15 (B15) | 32 | RW | 0000_0000h |
| 20Ch | UC Control 15 (C15) | 32 | RW | 0000_0000h |
| 210h | UC Status 15 (S15) | 32 | RW | See section |
| 214h | Alternate Address 15 (ALTA15) | 32 | RW | 0000_0000h |
| 218h | UC Control 2 15 (C2_15) | 32 | RW | 0000_0000h |
| 220h | UC A 16 (A16) | 32 | RW | 0000_0000h |
| 224h | UC B 16 (B16) | 32 | RW | 0000_0000h |
| 228h | UC Counter 16 (CNT16) | 32 | RW | 0000_0000h |
| 22Ch | UC Control 16 (C16) | 32 | RW | 0000_0000h |
| 230h | UC Status 16 (S16) | 32 | RW | See section |
| 234h | Alternate Address 16 (ALTA16) | 32 | RW | 0000_0000h |
| 238h | UC Control 2 16 (C2_16) | 32 | RW | 0000_0000h |
| 240h | UC A 17 (A17) | 32 | RW | 0000_0000h |
| 244h | UC B 17 (B17) | 32 | RW | 0000_0000h |
| 24Ch | UC Control 17 (C17) | 32 | RW | 0000_0000h |
| 250h | UC Status 17 (S17) | 32 | RW | See section |
| 254h | Alternate Address 17 (ALTA17) | 32 | RW | 0000_0000h |
| 258h | UC Control 2 17 (C2_17) | 32 | RW | 0000_0000h |
| 260h | UC A 18 (A18) | 32 | RW | 0000_0000h |
| 264h | UC B 18 (B18) | 32 | RW | 0000_0000h |
| 26Ch | UC Control 18 (C18) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 270h | UC Status 18 (S18) | 32 | RW | See section |
| 274h | Alternate Address 18 (ALTA18) | 32 | RW | 0000_0000h |
| 278h | UC Control 2 18 (C2_18) | 32 | RW | 0000_0000h |
| 280h | UC A 19 (A19) | 32 | RW | 0000_0000h |
| 284h | UC B 19 (B19) | 32 | RW | 0000_0000h |
| 28Ch | UC Control 19 (C19) | 32 | RW | 0000_0000h |
| 290h | UC Status 19 (S19) | 32 | RW | See section |
| 294h | Alternate Address 19 (ALTA19) | 32 | RW | 0000_0000h |
| 298h | UC Control 2 19 (C2_19) | 32 | RW | 0000_0000h |
| 2A0h | UC A 20 (A20) | 32 | RW | 0000_0000h |
| 2A4h | UC B 20 (B20) | 32 | RW | 0000_0000h |
| 2ACh | UC Control 20 (C20) | 32 | RW | 0000_0000h |
| 2B0h | UC Status 20 (S20) | 32 | RW | See section |
| 2B4h | Alternate Address 20 (ALTA20) | 32 | RW | 0000_0000h |
| 2B8h | UC Control 2 20 (C2_20) | 32 | RW | 0000_0000h |
| 2C0h | UC A 21 (A21) | 32 | RW | 0000_0000h |
| 2C4h | UC B 21 (B21) | 32 | RW | 0000_0000h |
| 2CCh | UC Control 21 (C21) | 32 | RW | 0000_0000h |
| 2D0h | UC Status 21 (S21) | 32 | RW | See section |
| 2D4h | Alternate Address 21 (ALTA21) | 32 | RW | 0000_0000h |
| 2D8h | UC Control 2 21 (C2_21) | 32 | RW | 0000_0000h |
| 2E0h | UC A 22 (A22) | 32 | RW | 0000_0000h |
| 2E4h | UC B 22 (B22) | 32 | RW | 0000_0000h |
| 2E8h | UC Counter 22 (CNT22) | 32 | RW | 0000_0000h |
| 2ECh | UC Control 22 (C22) | 32 | RW | 0000_0000h |
| 2F0h | UC Status 22 (S22) | 32 | RW | See section |
| 2F4h | Alternate Address 22 (ALTA22) | 32 | RW | 0000_0000h |
| 2F8h | UC Control 2 22 (C2_22) | 32 | RW | 0000_0000h |
| 300h | UC A 23 (A23) | 32 | RW | 0000_0000h |
| 304h | UC B 23 (B23) | 32 | RW | 0000_0000h |
| 308h | UC Counter 23 (CNT23) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 30Ch | UC Control 23 (C23) | 32 | RW | 0000_0000h |
| 310h | UC Status 23 (S23) | 32 | RW | See section |
| 314h | Alternate Address 23 (ALTA23) | 32 | RW | 0000_0000h |
| 318h | UC Control 2 23 (C2_23) | 32 | RW | 0000_0000h |

### 60.8.6.2 Module Configuration (MCR)

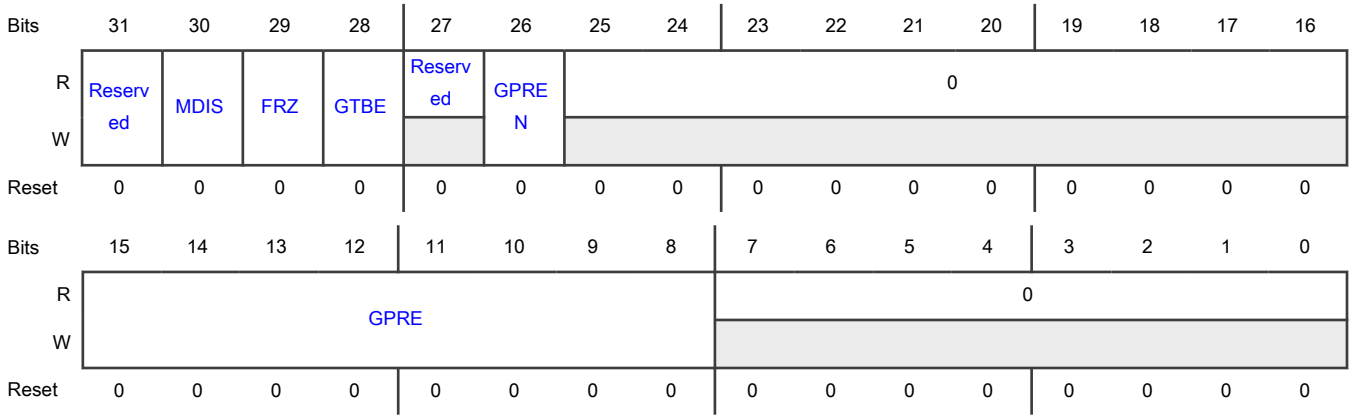
#### Offset

| Register | Offset |
|---|---|
| MCR | 0h |

#### Function

Configures eMIOS operation.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved<br>Write only the reset value to this field. |
| 30<br>MDIS | Module Disable<br>Disables eMIOS by stopping the clock. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When you disable eMIOS, its clock stops, it consumes less power, and you can access only the following registers:<br>• Module Configuration (MCR)<br>• Output Update Disable (OUDIS)<br>• Disable Channel (UCDIS)<br>0b - Enable<br>1b - Disable |
| 29<br>FRZ | Freeze<br>Prepares eMIOS to enter Freeze state (see Freeze and chip Debug mode). eMIOS does not enter Freeze state until both of the following conditions occur:<br>• The chip asserts Debug mode.<br>• You enable Freeze for each UC (write 1 to C*n*[FREN]).<br>A channel remains frozen until one or more of the following occur:<br>• You force eMIOS to exit the Freeze state (write 0 to FRZ).<br>• The chip deasserts Debug mode.<br>• You disable Freeze for that UC (write 0 to C*n*[FREN]).<br>0b - Exit Freeze state<br>1b - Enter Freeze state |
| 28<br>GTBE | Global Timebase Enable<br>Asserts the GTBE_OUT signal. If GTBE_OUT is connected to the GTBE_IN input of one or more eMIOS instances, asserting the signal turns on the internal counters of those eMIOS instances simultaneously.<br>**NOTE**<br>The GTBE_IN input enables (asserted) or disables (deasserted) the internal counters.<br>0b - Deassert GTBE_OUT<br>1b - Assert GTBE_OUT |
| 27<br>— | Reserved |
| 26<br>GPREN | Global Prescaler Enable<br>Enables the global prescaler counter. Disabling the global prescaler clears the prescaler counter and stops the prescaler clock.<br>0b - Disable<br>1b - Enable |
| 25-16 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 15-8<br>GPRE | Global Prescaler<br>Specifies the global clock prescaler. Write the desired clock divide ratio minus 1. For example, for a divide ratio of 8, write 111b. The prescaler can range from 1 (0b) to 256 (11111111b). |
| 7-0<br>— | Reserved |

### 60.8.6.3 Global Flag (GFLAG)

#### Offset

| Register | Offset |
|---|---|
| GFLAG | 4h |

#### Function

Groups the flag fields from all channels into one register to improve interrupt handling.

For UCs, each field mirrors the corresponding channel flag (S*n*[FLAG]).

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | F23 | F22 | F21 | F20 | F19 | F18 | F17 | F16 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | F15 | F14 | F13 | F12 | F11 | F10 | F9 | F8 | F7 | F6 | F5 | F4 | F3 | F2 | F1 | F0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 29<br>— | Reserved |
| 28<br>— | Reserved |
| 27<br>— | Reserved |
| 26<br>— | Reserved |
| 25<br>— | Reserved |
| 24<br>— | Reserved |
| 23<br>F23 | Mirror of UC 23 FLAG |
| 22<br>F22 | Mirror of UC 22 FLAG |
| 21<br>F21 | Mirror of UC 21 FLAG |
| 20<br>F20 | Mirror of UC 20 FLAG |
| 19<br>F19 | Mirror of UC 19 FLAG |
| 18<br>F18 | Mirror of UC 18 FLAG |
| 17<br>F17 | Mirror of UC 17 FLAG |
| 16<br>F16 | Mirror of UC 16 FLAG |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 15<br>F15 | Mirror of UC 15 FLAG |
| 14<br>F14 | Mirror of UC 14 FLAG |
| 13<br>F13 | Mirror of UC 13 FLAG |
| 12<br>F12 | Mirror of UC 12 FLAG |
| 11<br>F11 | Mirror of UC 11 FLAG |
| 10<br>F10 | Mirror of UC 10 FLAG |
| 9<br>F9 | Mirror of UC 9 FLAG |
| 8<br>F8 | Mirror of UC 8 FLAG |
| 7<br>F7 | Mirror of UC 7 FLAG |
| 6<br>F6 | Mirror of UC 6 FLAG |
| 5<br>F5 | Mirror of UC 5 FLAG |
| 4<br>F4 | Mirror of UC 4 FLAG |
| 3<br>F3 | Mirror of UC 3 FLAG |
| 2<br>F2 | Mirror of UC 2 FLAG |
| 1 | Mirror of UC 1 FLAG |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| F1 | |
| 0<br>F0 | Mirror of UC 0 FLAG |

### 60.8.6.4 Output Update Disable (OUDIS)

#### Offset

| Register | Offset |
|---|---|
| OUDIS | 8h |

#### Function

Disables transfers from AS2 to AS1 and BS2 to BS1 on a per-channel basis. This applies to any channel operating in MC, MCB, or any output mode that writes to AS2 and BS2.

When a field is 0, transfers on that channel occur immediately or in the next period, depending on the operation mode. Unless that mode's description states otherwise, the transfer occurs immediately.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | OU23 | OU22 | OU21 | OU20 | OU19 | OU18 | OU17 | OU16 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | OU15 | OU14 | OU13 | OU12 | OU11 | OU10 | OU9 | OU8 | OU7 | OU6 | OU5 | OU4 | OU3 | OU2 | OU1 | OU0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 29<br>— | Reserved |
| 28<br>— | Reserved |
| 27<br>— | Reserved |
| 26<br>— | Reserved |
| 25<br>— | Reserved |
| 24<br>— | Reserved |
| 23<br>OU23 | Channel 23 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 22<br>OU22 | Channel 22 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 21<br>OU21 | Channel 21 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 20<br>OU20 | Channel 20 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 19<br>OU19 | Channel 19 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 18<br>OU18 | Channel 18 Output Update Disable<br>0b - Enable<br>1b - Disable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 17<br>OU17 | Channel 17 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 16<br>OU16 | Channel 16 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 15<br>OU15 | Channel 15 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 14<br>OU14 | Channel 14 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 13<br>OU13 | Channel 13 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 12<br>OU12 | Channel 12 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 11<br>OU11 | Channel 11 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 10<br>OU10 | Channel 10 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 9<br>OU9 | Channel 9 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 8<br>OU8 | Channel 8 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 7<br>OU7 | Channel 7 Output Update Disable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Enable<br>1b - Disable |
| 6<br>OU6 | Channel 6 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 5<br>OU5 | Channel 5 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 4<br>OU4 | Channel 4 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 3<br>OU3 | Channel 3 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 2<br>OU2 | Channel 2 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 1<br>OU1 | Channel 1 Output Update Disable<br>0b - Enable<br>1b - Disable |
| 0<br>OU0 | Channel 0 Output Update Disable<br>0b - Enable<br>1b - Disable |

#### 60.8.6.5 Disable Channel (UCDIS)

**Offset**

| Register | Offset |
|---|---|
| UCDIS | Ch |

**Function**

Allows you to disable a UC by stopping its clock. Each field controls the clock for the corresponding UC.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>— | Reserved |
| 29<br>— | Reserved |
| 28<br>— | Reserved |
| 27<br>— | Reserved |
| 26<br>— | Reserved |
| 25<br>— | Reserved |
| 24<br>— | Reserved |
| 23<br>UCDIS23 | Disable UC 23<br>0b - Enable<br>1b - Disable |
| 22 | Disable UC 22 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| UCDIS22 | 0b - Enable<br>1b - Disable |
| 21<br>UCDIS21 | Disable UC 21<br>0b - Enable<br>1b - Disable |
| 20<br>UCDIS20 | Disable UC 20<br>0b - Enable<br>1b - Disable |
| 19<br>UCDIS19 | Disable UC 19<br>0b - Enable<br>1b - Disable |
| 18<br>UCDIS18 | Disable UC 18<br>0b - Enable<br>1b - Disable |
| 17<br>UCDIS17 | Disable UC 17<br>0b - Enable<br>1b - Disable |
| 16<br>UCDIS16 | Disable UC 16<br>0b - Enable<br>1b - Disable |
| 15<br>UCDIS15 | Disable UC 15<br>0b - Enable<br>1b - Disable |
| 14<br>UCDIS14 | Disable UC 14<br>0b - Enable<br>1b - Disable |
| 13<br>UCDIS13 | Disable UC 13<br>0b - Enable<br>1b - Disable |
| 12<br>UCDIS12 | Disable UC 12<br>0b - Enable<br>1b - Disable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 11<br>UCDIS11 | Disable UC 11<br>0b - Enable<br>1b - Disable |
| 10<br>UCDIS10 | Disable UC 10<br>0b - Enable<br>1b - Disable |
| 9<br>UCDIS9 | Disable UC 9<br>0b - Enable<br>1b - Disable |
| 8<br>UCDIS8 | Disable UC 8<br>0b - Enable<br>1b - Disable |
| 7<br>UCDIS7 | Disable UC 7<br>0b - Enable<br>1b - Disable |
| 6<br>UCDIS6 | Disable UC 6<br>0b - Enable<br>1b - Disable |
| 5<br>UCDIS5 | Disable UC 5<br>0b - Enable<br>1b - Disable |
| 4<br>UCDIS4 | Disable UC 4<br>0b - Enable<br>1b - Disable |
| 3<br>UCDIS3 | Disable UC 3<br>0b - Enable<br>1b - Disable |
| 2<br>UCDIS2 | Disable UC 2<br>0b - Enable<br>1b - Disable |
| 1<br>UCDIS1 | Disable UC 1 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Enable<br>1b - Disable |
| 0<br>UCDIS0 | Disable UC 0<br>0b - Enable<br>1b - Disable |

### 60.8.6.6 UC A n (A0 - A23)

#### Offset

For n = 0 to 23:

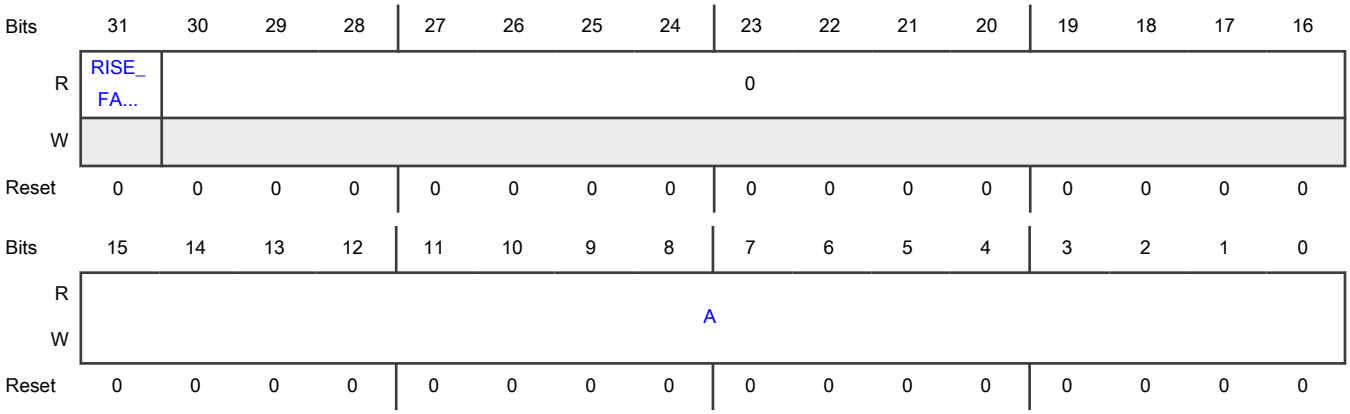
| Register | Offset |
|---|---|
| An | 20h + (n × 20h) |

#### Function

Accesses the AS1 and AS2 shadow registers for each UC, depending on the selected UC mode. See AS1, AS2, BS1, and BS2 shadow registers.

For information about how you write to and read from A*n*, B*n*, and ALTA*n* for all UC modes, see Assignment of values for A*n*[A], B*n*[B], and ALTA*n*[ALTA].

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31<br>RISE_FALL | Rising and falling edge detection<br>Indicates rising and falling edge in SAIC sub-mode. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 30-16<br>— | Reserved |
| 15-0<br>A | A<br>See the register description. |

### 60.8.6.7 UC B n (B0 - B23)

#### Offset

For n = 0 to 23:

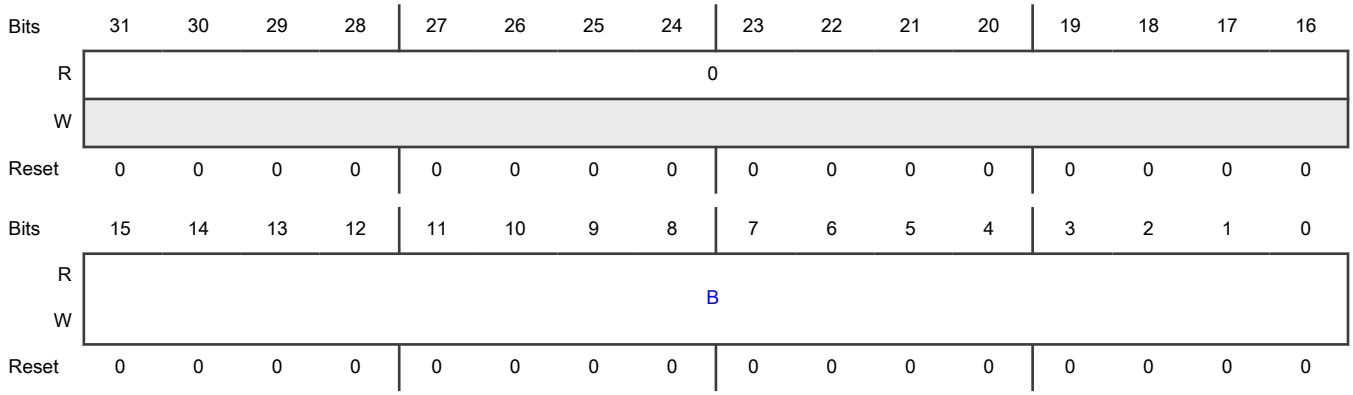
| Register | Offset |
|---|---|
| Bn | 24h + (n × 20h) |

#### Function

Accesses the BS1 and BS2 shadow registers for each UC, depending on the selected UC mode. See AS1, AS2, BS1, and BS2 shadow registers.

For information about how you write to and read from A*n*, B*n*, and ALTA*n* for all UC modes, see Assignment of values for A*n*[A], B*n*[B], and ALTA*n*[ALTA].

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0 | B |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| B | See the register description. |

### 60.8.6.8 UC Counter n (CNT0 - CNT23)

#### Offset

| Register | Offset |
|---|---|
| CNT0 | 28h |
| CNT1 | 48h |
| CNT2 | 68h |
| CNT3 | 88h |
| CNT4 | A8h |
| CNT5 | C8h |
| CNT6 | E8h |
| CNT7 | 108h |
| CNT8 | 128h |
| CNT16 | 228h |
| CNT22 | 2E8h |
| CNT23 | 308h |

#### Function

Indicates the value of the UC internal counter.

When eMIOS is in GPIO mode or the UC is frozen, this register is read-write. For all other modes, this register is read-only. When entering some UC modes, eMIOS automatically clears this register. See Unified channels (UC) for details.

The following modes use the UC counter:

- Output Pulse Width and Frequency Modulation Buffered (OPWFMB) mode
- Center Aligned Output PWM with Dead Time Insertion Buffered (OPWMCB) mode
- Pulse Edge Counting (PEC) mode
- Modulus Counter (MC) mode
- Modulus Counter Buffered (MCB) mode

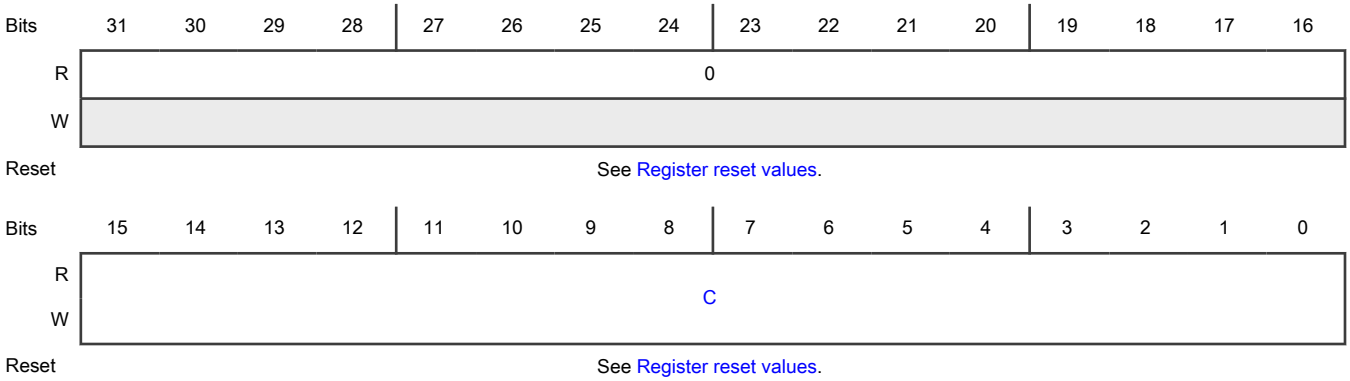
All other modes not in this list do not use the internal counter.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| eMIOS_0 | CNT0–CNT8<br>CNT16<br>CNT22–CNT23 | — |
| eMIOS_1 | CNT0<br>CNT8<br>CNT16<br>CNT22–CNT23 | CNT1–CNT7 |
| eMIOS_2 | CNT0<br>CNT8<br>CNT16<br>CNT22–CNT23 | CNT1–CNT7 |

### Diagram




### Register reset values

| Register | Reset value |
|---|---|
| CNT0 | eMIOS_0–eMIOS_2: 0000_0000h |
| CNT1–CNT7 | 0000_0000h |
| CNT8–CNT23 | eMIOS_0–eMIOS_2: 0000_0000h |
| CNT9–CNT15 | — |
| CNT17–CNT21 | — |

### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 15-0<br>C | Internal Counter Value<br>Indicates the value of the internal counter. |

### 60.8.6.9 UC Control n (C0 - C23)

#### Offset

For n = 0 to 23:

| Register | Offset |
|---|---|
| Cn | 2Ch + (n × 20h) |

#### Function

Controls UC operation.

**Table 344. MODE field values**

| UC mode | MODE (binary) |
|---|---|
| GPIO (input) | 000_0000 |
| GPIO (output) | 000_0001 |
| SAIC | 000_0010 |
| SAIC with edge capturing | 100_0010 |
| SAOC | 000_0011 |
| IPWM | 000_0100 |
| IPM | 000_0101 |
| DAOC (with FLAG = 1 on B match) | 000_0110 |
| DAOC (with FLAG = 1 on A and B match) | 000_0111 |
| PEC (continuous) | 000_1010 |
| PEC (single shot) | 000_1011 |
| Reserved | 000_1111 |
| MC (up counter with clear on match start) | 001_000p[1] |
| MC (up counter with clear on match end) | 001_001p[1] |
| MC (up/down counter) | 001_01pp[1] |
| Reserved | 010_0100–010_0101 |
| OPWMT | 010_0110 |
| Reserved | 010_0111–100_1111 |

*Table continues on the next page...*

Table 344. MODE field values (continued)

| UC mode | MODE (binary) |
|---|---|
| MCB (up counter) | 101_000p[1] |
| Reserved | 101_001p |
| MCB (up or down counter) | 101_01pp[1] |
| OPWFMB | 101_10p0[1] |
| Reserved | 101_10p1 |
| OPWMCB (with trailing edge dead time) | 101_11p0[1] |
| OPWMCB (with leading edge dead time) | 101_11p1[1] |
| OPWMB | 110_00p0[1] |
| Reserved | 110_0001–111_1111 |

1. p = Adjust parameters for the mode of operation. See Unified channels (UC) for details.

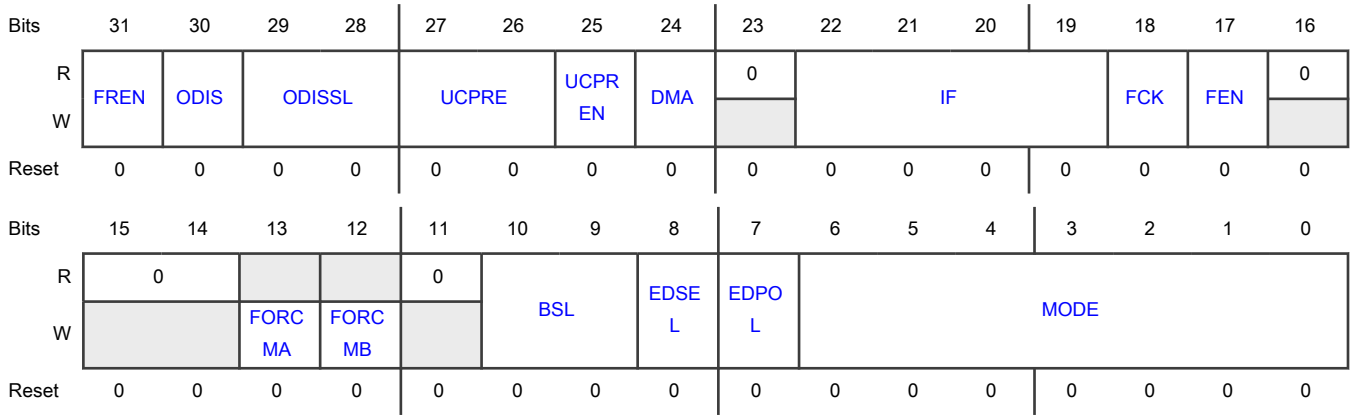
Table 345. Edge Polarity (EDPOL) settings

| For these modes: | EDPOL does this: | and EDPOL values mean: |
|---|---|---|
| Input | Selects which edge triggers the internal counter, an input capture, or the input capture flag. If EDPOL does not appear in the UC mode description, it has no effect on UC operation. | 0b - Trigger on a falling edge<br>1b - Trigger on a rising edge |
| Output | Selects the logic level on the output pin. | 0b - A match on comparator A deasserts the output flip-flop, while a match on comparator B sets it<br>1b - A match on comparator A asserts the output flip-flop, while a match on comparator B clears it |

Table 346. Edge Select (EDSEL) settings

| For these modes: | EDSEL does this: | and EDSEL values mean: |
|---|---|---|
| Input | Selects whether the internal counter is triggered by both edges of a pulse or by a single edge as defined by EDPOL. If EDSEL does not appear in the UC mode description, it has no effect on UC operation. | 0b - Single-edge triggering on edge specified in EDPOL<br>1b - Both-edges triggering |
| GPIO in | Selects whether an edge asserts the input capture flag (S*n*[FLAG]). | 0b - Sets S*n*[FLAG] on the edge specified in EDPOL<br>1b - Does not set S*n*[FLAG] |
| SAOC | Selects the behavior of the output flip-flop at each match. | 0b - Drive the EDPOL value on the output flip-flop<br>1b - Toggle output flip-flop |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>FREN | Freeze Enable<br>Enables putting the UC in Freeze state. See Freeze and chip Debug mode.<br>0b - Disable<br>1b - Enable |
| 30<br>ODIS | Output Disable<br>Disables the output pin when running any of the output modes except GPIO.<br>When you assert ODIS:<br>• If the selected Output Disable Input signal asserts, the output pin reflects:<br>— For OPWFMB and OPWMB modes, EDPOL (see Output Pulse Width and Frequency Modulation Buffered (OPWFMB) mode and Output PWM Buffered (OPWMB) mode).<br>— For all other modes, the complement of EDPOL, but the UC continues to operate normally—it continues to generate flags and matches.<br>• If the selected Output Disable Input signal deasserts, the output pin operates normally.<br>0b - Enables output pin (the output pin operates normally)<br>1b - Disables output pin |
| 29-28<br>ODISSL | Output Disable Select<br>Selects one of four output disable input signals.<br>00b - 0<br>01b - 1<br>10b - 2<br>11b - 3 |
| 27-26 | Prescaler<br>Selects the clock divider value for the UC internal prescaler. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| UCPRE | **NOTE**<br>The two least-significant bits of C2_*n*[UCEXTPRE] mirror UCPRE. Any write to UCPRE also affects UCEXTPRE.<br>00b - 1<br>01b - 2<br>10b - 3<br>11b - 4 |
| 25<br>UCPREN | Prescaler Enable<br>Enables the prescaler counter.<br>0b - Disable (no clock)<br>1b - Enable |
| 24<br>DMA | Direct Memory Access<br>Selects whether the input capture flag (S*n*[FLAG]) or overrun (S*n*[OVR]) triggers an interrupt or DMA request.<br>0b - Interrupt request<br>1b - DMA request |
| 23<br>— | Reserved |
| 22-19<br>IF | Input Filter<br>Selects the minimum input pulse width, in filter clock cycles, that can pass through the input filter. This field does not apply to output modes.<br>The filter latency—the difference in time between the input and the response—is three clock edges.<br>0000b - Bypassed; the input signal is synchronized before arriving at the digital filter<br>0001b - 2 cycles<br>0010b - 4 cycles<br>0100b - 8 cycles<br>1000b - 16 cycles<br>All other values are reserved. |
| 18<br>FCK | Filter Clock Select<br>Selects the clock source for the programmable input filter.<br>0b - Prescaled clock<br>1b - eMIOS module clock |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 17<br>FEN | Flag Enable<br>Enables the input capture flag (S*n*[FLAG]) to generate an interrupt request or a DMA request, as selected in DMA.<br>0b - Disable<br>1b - Enable |
| 16-14<br>— | Reserved |
| 13<br>FORCMA | Force Match A<br>For output modes, asserting Force Match A causes the comparator to report a successful comparison on comparator A regardless of whether a comparison actually occurred. It does not set the input capture flag (S*n*[FLAG]). This field always reads 0. This bit is meaningful for every output operation mode that uses comparator A. Otherwise it has no effect.<br>0b - No effect<br>1b - Force a match at comparator A |
| 12<br>FORCMB | Force Match B<br>For output modes, asserting Force Match B causes the comparator to report a successful comparison on comparator B regardless of whether a comparison actually occurred. It does not set the input capture flag (S*n*[FLAG] does not become 1). This field always reads 0.<br>This field applies only if the UC is in one of the following output modes.<br>• Single Action Output Capture (SAOC) mode<br>• Double Action Output Compare (DAOC) mode<br>• Output Pulse Width and Frequency Modulation Buffered (OPWFMB) mode<br>• Center Aligned Output PWM with Dead Time Insertion Buffered (OPWMCB) mode<br>• Output PWM Buffered (OPWMB) mode<br>• Output PWM with Trigger (OPWMT) mode<br>This field has no effect in all other modes.<br>0b - No effect<br>1b - Force a match at comparator B |
| 11<br>— | Reserved |
| 10-9<br>BSL | Bus Select<br>Selects one of the counter buses or the internal counter for the UC.<br>Not all chips support all counter buses. See the chip-specific eMIOS information for details.<br>00b - |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • Channels 0-22: counter bus A<br>• Channel 23: reserved<br>01b -<br>• Channels 1-7: counter bus B<br>• Channels 9-15: counter bus C<br>• Channels 17-23: counter bus D<br>• Channels 25-31: counter bus E<br>• Channels 0, 8, 16, 24: reserved<br>10b -<br>• Channels 0-21, 23: counter bus F<br>• Channel 22: reserved<br>11b - Internal counter for all channels |
| 8<br>EDSEL | Edge Selection<br>Controls several functions as shown in Table 346. |
| 7<br>EDPOL | Edge Polarity<br>Controls several functions as shown in Table 345. |
| 6-0<br>MODE | Mode Selection<br>Selects the UC mode as shown in Table 344.<br>**NOTE**<br>If you write a reserved value to this field, the results are unpredictable.<br>When you change the value of MODE, you must first enter GPIO mode to reset the UC's internal functions. Failure to do this can result in invalid and unexpected output compare or input capture results or incorrectly set input capture flags (S*n*[FLAG]). |

### 60.8.6.10 UC Status n (S0 - S23)

#### Offset

For n = 0 to 23:

| Register | Offset |
|---|---|
| Sn | 30h + (n × 20h) |

#### Function

Indicates the status of the UC input-output signals and the overflow condition of the internal counter.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | OVR | 0 | | | | | | | | | | | | | | |
| W | W1C | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | OVFL | 0 | | | | | | | | | | | | UCIN | UCOUT | FLAG |
| W | W1C | | | | | | | | | | | | | | | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | u[1] | 0 | 0 |

1. Reset value is undefined.

### Fields

| Field | Function |
|---|---|
| 31<br>OVR | Overrun<br>Indicates that FLAG generation occurred when FLAG was already 1.<br>Return OVR to 0 by writing 1 to either OVR or FLAG.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No overrun<br>1b - Overrun<br>When writing<br>0b - No overrun<br>1b - Return OVR to 0 |
| 30-16<br>— | Reserved |
| 15<br>OVFL | Overflow<br>Indicates that an overflow has occurred in the internal counter.<br>**NOTE**<br>This field is not supported in every instance. The following table includes only supported registers. |

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>eMIOS_0</td><td>S0–S8<br>S16<br>S22–S23</td><td>S9–S15<br>S17–S21</td></tr>
<tr><td>eMIOS_1</td><td>S0<br>S8<br>S16<br>S22–S23</td><td>S1–S7<br>S9–S15<br>S17–S21</td></tr>
<tr><td>eMIOS_2</td><td>S0<br>S8<br>S16<br>S22–S23</td><td>S1–S7<br>S9–S15<br>S17–S21</td></tr>
</table>
NOTE<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No overflow<br>1b - Overflow<br>When writing<br>0b - No overflow<br>1b - Return OVFL to 0</td></tr>
<tr><td>14-3<br>—</td><td>Reserved</td></tr>
<tr><td>2<br>UCIN</td><td>UC Input Pin<br>Indicates the input pin state after the input pin has been filtered and synchronized.<br>0b - Negated<br>1b - Asserted</td></tr>
<tr><td>1<br>UCOUT</td><td>UC Output Pin<br>Indicates the output pin state.<br>0b - Negated<br>1b - Asserted</td></tr>
<tr><td>0<br>FLAG</td><td>Flag<br>Indicates that an input capture or a match event in the comparators has occurred.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No event<br>1b - Event has occurred<br>When writing<br>0b - No event<br>1b - Clear the flag (FLAG returns to 0) |

### 60.8.6.11 Alternate Address n (ALTA0 - ALTA23)

#### Offset

For n = 0 to 23:

| Register | Offset |
|---|---|
| ALTAn | 34h + (n × 20h) |

#### Function

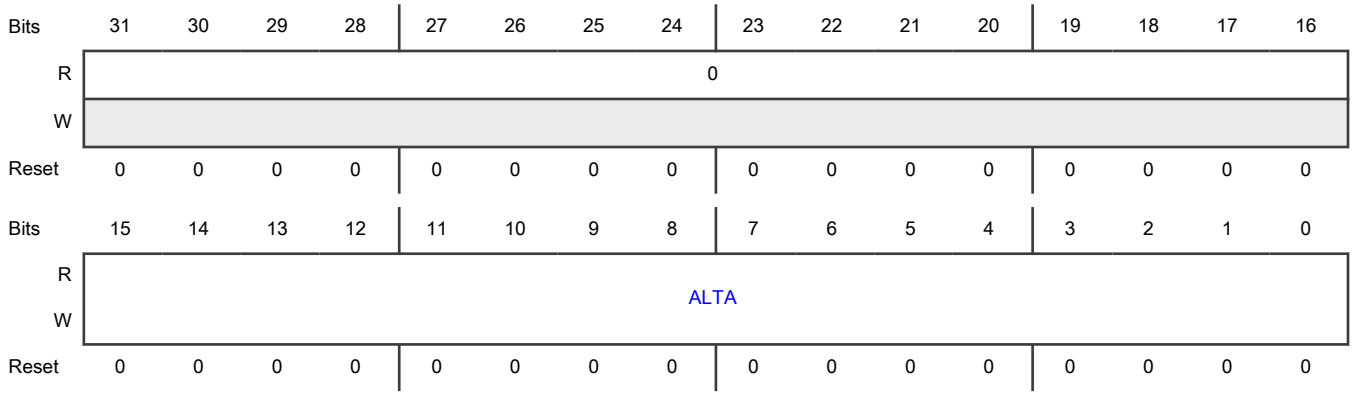
Provides an alternate address to access AS2 shadow registers in these restricted modes:

- General-Purpose Input and Output (GPIO) mode
- Pulse Edge Counting (PEC) mode
- Output PWM with Trigger (OPWMT) mode

In these modes, you can use A*n*[A] together with ALTA*n*[ALTA] to access both AS1 and AS2. Assignment of values for A*n*[A], B*n*[B], and ALTA*n*[ALTA] summarizes the ALTA*n* writing and reading accesses for all ALTA*n* operation modes.

In modes that do not require this register, any write access is ignored and any read access returns unspecified data.

#### Diagram



### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>ALTA | Alternate Address<br>See the register description. |

## 60.8.6.12 UC Control 2 n (C2_0 - C2_23)

### Offset

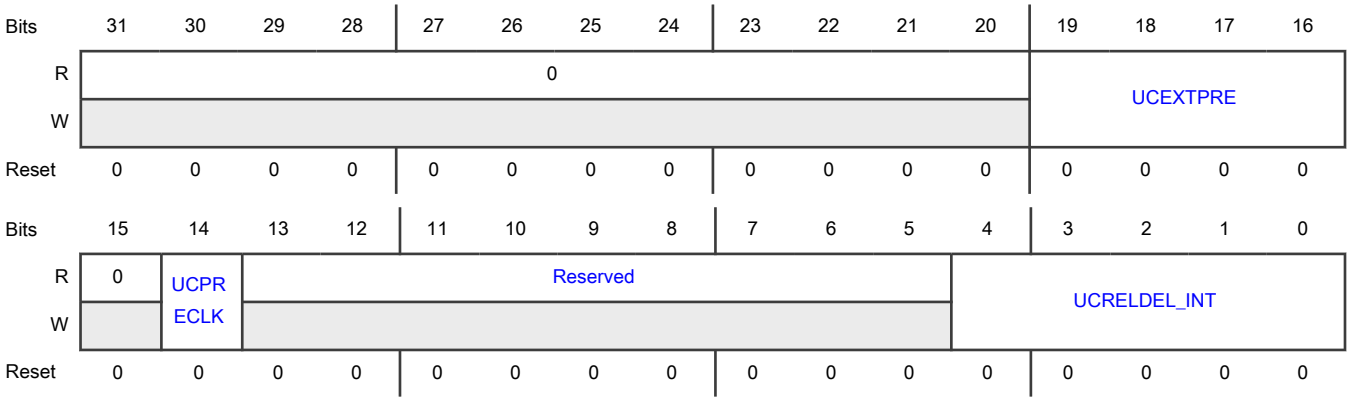
For n = 0 to 23:

| Register | Offset |
|---|---|
| C2_n | 38h + (n × 20h) |

### Function

Provides additional controls for the UC.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19-16<br>UCEXTPRE | Extended Prescaler<br>Specifies the clock divider for the internal UC prescaler. Write the desired divider value minus 1. For example, for a divider of 1, write 0. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>The two least-significant bits of UCEXTPRE mirror C*n*[UCPRE]. Any write operation to UCEXTPRE also affects UCPRE. |
| 15<br>— | Reserved |
| 14<br>UCPRECLK | Prescaler Clock Source<br>Selects the clock source for the internal prescaler.<br>0b - Prescaled clock<br>1b - eMIOS module clock |
| 13-5<br>— | Reserved |
| 4-0<br>UCRELDEL_INT | Reload Signal Output Delay Interval<br>Specifies the delay interval, in counter bus reload events, between each assertion of AS1-BS1 reload in MC and MCB modes.<br>00000b - Every event<br>00001b - Every 2nd event<br>00010b - Every 3rd event<br>. . .<br>11111b - Every 32nd event |

## 60.9 Glossary

| | |
|---|---|
| **BIU** | IP bus interface unit |
| **CP** | Clock prescaler |
| **GCP** | Global clock prescaler |
| **IIB** | Internal interface bus |
| **PIF** | Programmable input filter |
| **UC** | Unified channel |

# Chapter 61
# Body Cross-triggering Unit (BCTU)

## 61.1 Chip-specific BCTU information

### 61.1.1 BCTU configuration

The following table lists the BCTU configuration.

**Table 347. BCTU configuration**

| Feature | Configuration |
|---|---|
| FIFO1 | 16 words |
| FIFO2 | 8 words |

### 61.1.2 Trigger sources and instances

This chip has 1 instance of BCTU module.

**Table 348. BCTU instance**

| Instance | MCXE315/MCXE316/MCXE317/MCXE31B |
|---|---|
| BCTU | Yes |

The following table lists the BCTU trigger sources.

**Table 349. BCTU Trigger sources**

| BCTU Trigger Number | Module | Source | Applicability |
|---|---|---|---|
| 0 | eMIOS_0 | Channel_0 | All |
| 1 | eMIOS_0 | Channel_1 | All |
| 2 | eMIOS_0 | Channel_2 | All |
| 3 | eMIOS_0 | Channel_3 | All |
| 4 | eMIOS_0 | Channel_4 | All |
| 5 | eMIOS_0 | Channel_5 | All |
| 6 | eMIOS_0 | Channel_6 | All |
| 7 | eMIOS_0 | Channel_7 | All |
| 8 | eMIOS_0 | Channel_8 | All |
| 9 | eMIOS_0 | Channel_9 | All |
| 10 | eMIOS_0 | Channel_10 | All |
| 11 | eMIOS_0 | Channel_11 | All |
| 12 | eMIOS_0 | Channel_12 | All |
| 13 | eMIOS_0 | Channel_13 | All |
| 14 | eMIOS_0 | Channel_14 | All |

*Table continues on the next page...*

Table 349. BCTU Trigger sources (continued)

| BCTU Trigger Number | Module | Source | Applicability |
|---|---|---|---|
| 15 | eMIOS_0 | Channel_15 | All |
| 16 | eMIOS_0 | Channel_16 | All |
| 17 | eMIOS_0 | Channel_17 | All |
| 18 | eMIOS_0 | Channel_18 | All |
| 19 | eMIOS_0 | Channel_19 | All |
| 20 | eMIOS_0 | Channel_20 | All |
| 21 | eMIOS_0 | Channel_21 | All |
| 22 | eMIOS_0 | Channel_22 | All |
| 23 | Any of the TRGMUX inp | BCTU_Trg23 | All |
| 24 | eMIOS_1 | Channel_0 | All |
| 25 | eMIOS_1 | Channel_1 | All |
| 26 | eMIOS_1 | Channel_2 | All |
| 27 | eMIOS_1 | Channel_3 | All |
| 28 | eMIOS_1 | Channel_4 | All |
| 29 | eMIOS_1 | Channel_5 | All |
| 30 | eMIOS_1 | Channel_6 | All |
| 31 | eMIOS_1 | Channel_7 | All |
| 32 | eMIOS_1 | Channel_8 | All |
| 33 | eMIOS_1 | Channel_9 | All |
| 34 | eMIOS_1 | Channel_10 | All |
| 35 | eMIOS_1 | Channel_11 | All |
| 36 | eMIOS_1 | Channel_12 | All |
| 37 | eMIOS_1 | Channel_13 | All |
| 38 | eMIOS_1 | Channel_14 | All |
| 39 | eMIOS_1 | Channel_15 | All |
| 40 | eMIOS_1 | Channel_16 | All |
| 41 | eMIOS_1 | Channel_17 | All |
| 42 | eMIOS_1 | Channel_18 | All |
| 43 | eMIOS_1 | Channel_19 | All |
| 44 | eMIOS_1 | Channel_20 | All |
| 45 | eMIOS_1 | Channel_21 | All |
| 46 | eMIOS_1 | Channel_22 | All |

*Table continues on the next page...*

Table 349. BCTU Trigger sources (continued)

| BCTU Trigger Number | Module | Source | Applicability |
|---|---|---|---|
| 47 | Any of the TRGMUX inp | BCTU_Trg47 | All |
| 48 | eMIOS_2 | Channel_0 | MCXE31B |
| 49 | eMIOS_2 | Channel_1 | MCXE31B |
| 50 | eMIOS_2 | Channel_2 | MCXE31B |
| 51 | eMIOS_2 | Channel_3 | MCXE31B |
| 52 | eMIOS_2 | Channel_4 | MCXE31B |
| 53 | eMIOS_2 | Channel_5 | MCXE31B |
| 54 | eMIOS_2 | Channel_6 | MCXE31B |
| 55 | eMIOS_2 | Channel_7 | MCXE31B |
| 56 | eMIOS_2 | Channel_8 | MCXE31B |
| 57 | eMIOS_2 | Channel_9 | MCXE31B |
| 58 | eMIOS_2 | Channel_10 | MCXE31B |
| 59 | eMIOS_2 | Channel_11 | MCXE31B |
| 60 | eMIOS_2 | Channel_12 | MCXE31B |
| 61 | eMIOS_2 | Channel_13 | MCXE31B |
| 62 | eMIOS_2 | Channel_14 | MCXE31B |
| 63 | eMIOS_2 | Channel_15 | MCXE31B |
| 64 | eMIOS_2 | Channel_16 | MCXE31B |
| 65 | eMIOS_2 | Channel_17 | MCXE31B |
| 66 | eMIOS_2 | Channel_18 | MCXE31B |
| 67 | eMIOS_2 | Channel_19 | MCXE31B |
| 68 | eMIOS_2 | Channel_20 | MCXE31B |
| 69 | eMIOS_2 | Channel_21 | MCXE31B |
| 70 | eMIOS_2 | Channel_22 | MCXE31B |
| 71 | Any of the TRGMUX inp | BCTU_Trg71 | MCXE31B |

**NOTE**

You must clear triggers to BCTU after TRGMUX programming using TRGCFG_n[TRG_FLAG] bit.

## 61.2 Overview

BCTU accepts ADC conversion-request trigger inputs and routes those requests to one or more ADCs.

BCTU accepts the following trigger sources:

- Hardware triggers (outputs from on-chip timer modules)
- Software triggers through the SFTRGRn register

BCTU maps each trigger input to a Trigger Configuration (TRGCFG_0 - TRGCFG_71) register, which specifies the target ADC channel and other control data for the requested conversion.

You can configure a list of ADC channels, the CL, that processes a sequence of conversions in response to a single trigger input. See the Conversion list (CL) section.

BCTU stores each ADC conversion result for CPU access or routing to memory through DMA.

If the same ADC is targeted by multiple triggers, priority logic selects the next trigger to process from all active trigger inputs. See Priority of simultaneous triggers.

BCTU input trigger signals are level-sensitive; thus, if a logical level 1 is detected, an event is captured by the BCTU.

### 61.2.1 Block diagram

The following figure shows the top-level block diagram of BCTU.

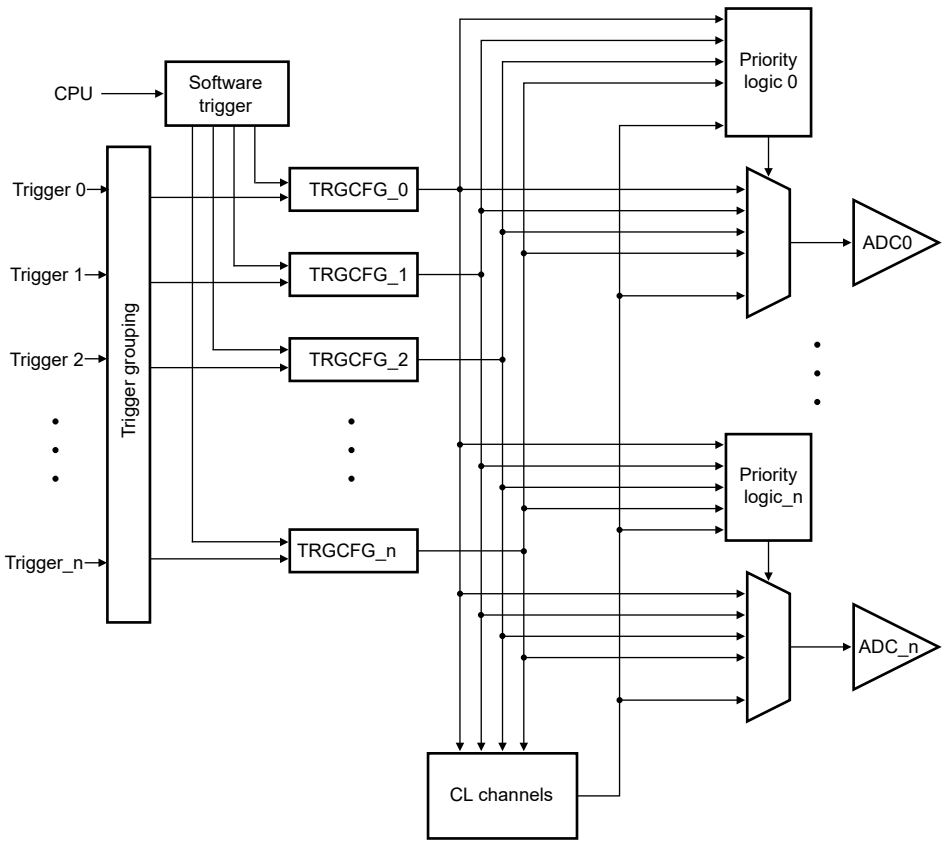



Figure 342. BCTU block diagram

### 61.2.2 Features

BCTU supports:

- 72 Trigger inputs, configured with the Trigger Configuration (TRGCFG_0 - TRGCFG_71) registers
- Trigger outputs to 3 ADCs
- BCTU to ADC0 and ADC1
- One priority selection logic block per ADC
- Selection multiplexers to select one trigger at a time for each ADC

- A CL of 32 ADC channels
- A trigger grouping block to latch the hardware triggers. BCTU clears these triggers when the ADC sends an acknowledgement for conversion completion.

### 61.2.3 Interaction with other peripherals

BCTU works together with timer and ADC modules on the chip. Use the trigger configuration registers (TRGCFG_*n*) to map BCTU triggers to ADC inputs. When a timer output asserts a BCTU trigger, BCTU passes the trigger to an ADC input and stores the trigger assertion until the corresponding ADC begins the requested conversion. When the ADC completes the conversion, BCTU asserts a trigger clear signal to clear both the internally stored trigger and the external trigger source if they are still asserted. Implementation illustrates the modules that interact with BCTU.

**NOTE**

A timer trigger can remain active until the corresponding ADC initiates conversion. Therefore, another match may occur in the timer before the ADC conversion begins. BCTU does not detect this trigger overrun. Your application must look for and respond to such conditions.

### 61.2.4 Implementation

The following figure shows the implementation structure of the BCTU.

**NOTE**

For more details of the actual trigger sources implemented in this device, see the chip-specific information section.

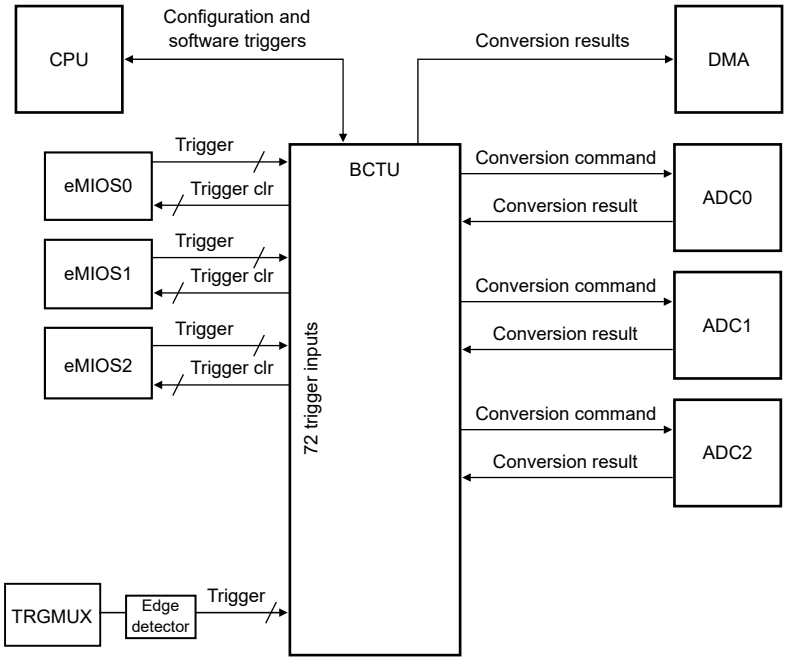



Figure 343. Implementation

## 61.3 Functional description

### 61.3.1 Triggers

#### 61.3.1.1 Overview

BCTU prioritizes ADC trigger inputs based on the input number: the lower the number, the higher the priority associated with it. Therefore, trigger 0 is the highest priority trigger and trigger 71 is the lowest priority. The trigger input priority scheme determines which trigger takes precedence when more than one trigger occurs at the same time, or when the triggered ADC is busy with a conversion. Therefore, there are two possible scenarios for the trigger priority selection:

- When there are no ongoing conversions
- When a triggered ADC is executing a conversion

BCTU supports two types of trigger sources:

- Hardware triggers from timer modules
- Software triggers (SFTRGR*n*)

#### 61.3.1.2 Hardware triggers

To request an ADC conversion through the BCTU, a timer module such as eMIOS asserts its output signal, which is connected to a BCTU trigger input. When BCTU detects the asserted trigger, it routes that request to the ADCs selected in the corresponding Trigger Configuration (TRGCFG_*n*[ADC_SEL*m*]). The trigger remains asserted until the requested conversion begins, at which time BCTU clears the trigger. If the timer asserts the trigger input when the trigger is already set, BCTU ignores the new trigger. If BCTU receives a new trigger input at the same time it is clearing the trigger, then the trigger remains asserted and BCTU initiates a new conversion.

If you configure the trigger for CL, trigger assertion begins a sequence of conversions controlled by the CL Channel Address (LISTCHR_0 - LISTCHR_15) registers. In this event, BCTU clears the trigger when the last conversion in the CL begins.

**NOTE**

Any new trigger input participates in the trigger priority selection scheme and therefore may not trigger the next conversion.

#### 61.3.1.3 Software triggers

You assert a software trigger by writing 1 to the corresponding field of the appropriate software trigger (SFTRGR*n*) register. Similar to the hardware triggers, BCTU changes the software trigger field to 0 when the requested conversion begins. If you write 1 to the software trigger field while the conversion is still in progress, BCTU ignores the request. If you write 1 to the field at the same time BCTU is attempting to clear the trigger (change the field to 0), the trigger remains asserted and BCTU triggers a new conversion following the trigger priority criteria.

#### 61.3.1.4 Configure all triggers

Global trigger enable (MCR[GTRGEN]) allows you to configure the BCTU trigger inputs and then activate them all at one time.

To configure and then activate all BCTU triggers:

1. Clear any currently active and asserted trigger inputs (write 0 to TRGFG_*n*[TRG_FLAG]), which also causes BCTU to assert trigger clear to the trigger sources.
2. Disable global triggers (MCR[GTRGEN]). This action deactivates all trigger inputs and prevents BCTU from generating any ADC conversion requests.
3. Configure the individual trigger inputs (see Configure individual triggers).
4. Enable global triggers (MCR[GTRGEN]). This action activates all individually enabled triggers at the same time.

#### 61.3.1.5 Configure individual triggers

Using Trigger Configuration (TRGCFG_0 - TRGCFG_71), for each trigger:

- Enable the trigger (TRIGEN).

- Specify the target ADC (ADC_SEL*n*).
- Specify the ADC channel (CHANNEL_VALUE_OR_LADDR).
- Specify whether the trigger initiates a single conversion or a CL (TRS).
- For a CL trigger, specify a pointer to the CL element (TRGCFG_x[CHANNEL_VALUE_OR_LADDR]).

#### 61.3.1.6 Reconfigure triggers on-the-fly

**NOTE**

Reconfigure BCTU triggers only when the individual trigger is not active (TRG_FLAG). If TRG_FLAG is 1, the requested conversion is still in progress and it is not safe to modify the trigger configuration.

To reconfigure a trigger on-the-fly (while BCTU is in full operation):

1. Disable the trigger (TRIGEN) to prevent it from requesting a new conversion.
2. Check the current state of the trigger (TRG_FLAG) to be sure no conversion or CL is in progress.
3. When any in-progress conversion or CL completes (TRG_FLAG == 0), clear the trigger source by writing 1 to TRG_FLAG. This action ensures that only trigger inputs that assert after you reconfigure and re-enable the trigger are able to request a new conversion or CL.
4. Reconfigure the trigger (see Configure individual triggers).
5. Enable the trigger (TRIGEN).

#### 61.3.1.7 Enable software triggers

To allow software triggers (SFTRGRn) to request conversions, you must enable global trigger enable (MCR[GTRGEN]). If you disable global trigger enable, software triggers do not initiate a conversion but asserted triggers remain asserted. However, when you enable global trigger enable, any asserted software triggers initiate conversions following the priority scheme of lowest to highest trigger number.

Software triggers operate regardless of individual trigger enable (TRGCFG_x[TRIGEN]). That is, writing 1 to any Software Trigger (SFTRGRn) initiates a conversion (following the priority scheme) regardless of the value of trigger enable. This software trigger independence supports the elimination of conflict between software and hardware triggers.

#### 61.3.1.8 Trigger timing

There are four clock cycles from assertion of a BCTU trigger input to assertion of the trigger output to the ADC. For single conversions, BCTU asserts trigger clear back to the trigger source simultaneous to asserting the trigger to the ADC.

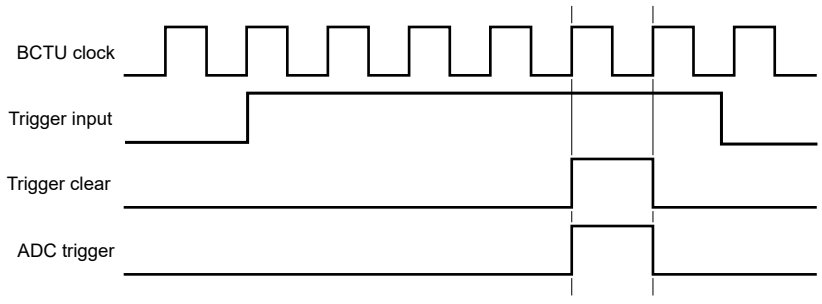



Figure 344. Trigger timing

#### 61.3.1.9 Priority of simultaneous triggers

When more than one trigger input asserts a conversion request to the same ADC at the same time—that is, in the same BCTU clock cycle—BCTU employs a priority scheme to determine which asserted trigger to process next. BCTU prioritizes trigger inputs based on the trigger number. The lowest number, trigger 0, has the highest priority, and the highest number, trigger 71, has the lowest priority.

If two triggers assert in the same clock cycle, BCTU gives priority to the lower numbered trigger. For example, if trigger 0 and trigger 1 both assert in the same cycle, BCTU selects trigger 0 for trigger generation. Trigger 1 remains active (pending) until the conversion requested by trigger 0 completes. See Simultaneous trigger priority management.

If trigger 0 asserts again before the next trigger selection opportunity at the end of the current conversion, then BCTU selects it again and trigger 1 remains asserted (pending) during the next ADC conversion time frame.

As illustrated in Trigger timing, BCTU requires four clock cycles to transmit a conversion request to an ADC if that ADC is not already executing a conversion. However, if the ADC is busy at the time of trigger assertion, BCTU requires only three clock cycles to generate the conversion request for a subsequent conversion.

Triggers that assert in the same clock cycle but target different ADCs are not in conflict, and BCTU processes both triggers at the same time.

#### 61.3.1.10 Simultaneous trigger priority management

The following figure shows the simultaneous triggers for priority management.

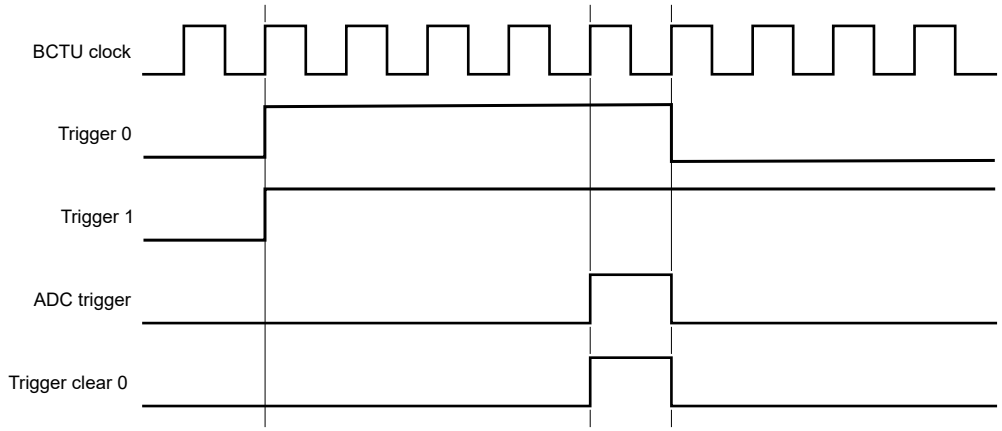



**Figure 345. Simultaneous trigger priority management**

#### 61.3.1.11 Trigger management when ADC is busy

When a trigger input asserts a conversion request to an ADC that is already processing a conversion requested by a different trigger, the new trigger remains active (pending) until the conversion in progress completes. This behavior is the same as if the two triggers had been simultaneous (occurred in the same clock cycle). When the in-progress conversion completes, the pending trigger then competes for priority with any other active trigger for that ADC.

#### 61.3.1.12 Trigger clearing mechanism

There are two sources of ADC conversion triggers: hardware trigger, from the trigger inputs and software triggers, from SFTRGRn register.

A logical level 1 is detected in the input and captured into an input trigger register. This register remains set until the related conversion starts. At this time, the input trigger register is cleared. If another logical level 1 is detected while the trigger register is already set, the new level is not considered by the BCTU hardware. If a new logical level 1 occurs at the same time the trigger register is being cleared, then the register remains set and a new conversion is triggered by this input.

**NOTE**

The new triggered conversion will participate in the trigger priority selection thus may not trigger the next conversion.

The software trigger behaves similarly to the hardware trigger. The difference is that the trigger register is set by a CPU write operation. The trigger register is cleared by the BCTU hardware when the related conversion starts. If the CPU tries to set the trigger register by writing to the SFTRGR register corresponding bit, while the register is already set, the new attempt to set the register is not considered by the BCTU hardware. If a set occurs at the same time the hardware is attempting to clear the same trigger register, then the set has precedence and the register remains set. A new conversion will be triggered following the trigger priority criteria implemented in the hardware.

In case a LIST of conversions is triggered, instead of a single conversion, the trigger register is cleared at the start of the last conversion in the LIST.

## 61.3.2 Conversion list (CL)

### 61.3.2.1 CL overview

A CL executes a sequence of conversions on one or more target ADCs. See CL architecture and operation. It also stores information from the last conversion executed in a sequence except for the target ADC, which is stored in ADC select n (TRGCFG_*m*[ADC_SEL*n*]).

If the channel or CL address (TRGCFG_x[CHANNEL_VALUE_OR_LADDR]) points to a CL element containing one or more entries:

1. The trigger initiates a sequence of conversions starting with the target CL element.
2. The sequence executes until the LAST channel (LISTCHR_*n*[LAST_y] is 1), and stops after that channel is converted.

When you configure the CL element for next channel wait for trigger (write 1 to LISTCHR_*n*[NEXT_CH_WAIT_ON_TRIG_y]):

1. BCTU clears the trigger source.
2. CL execution pauses after the current conversion is completed.
3. CL execution resumes when the same trigger asserts again.

BCTU determines trigger priority before executing any CL. See Priority of simultaneous triggers.

### 61.3.2.2 CL architecture and operation

The following figure shows the CL architecture and operation.

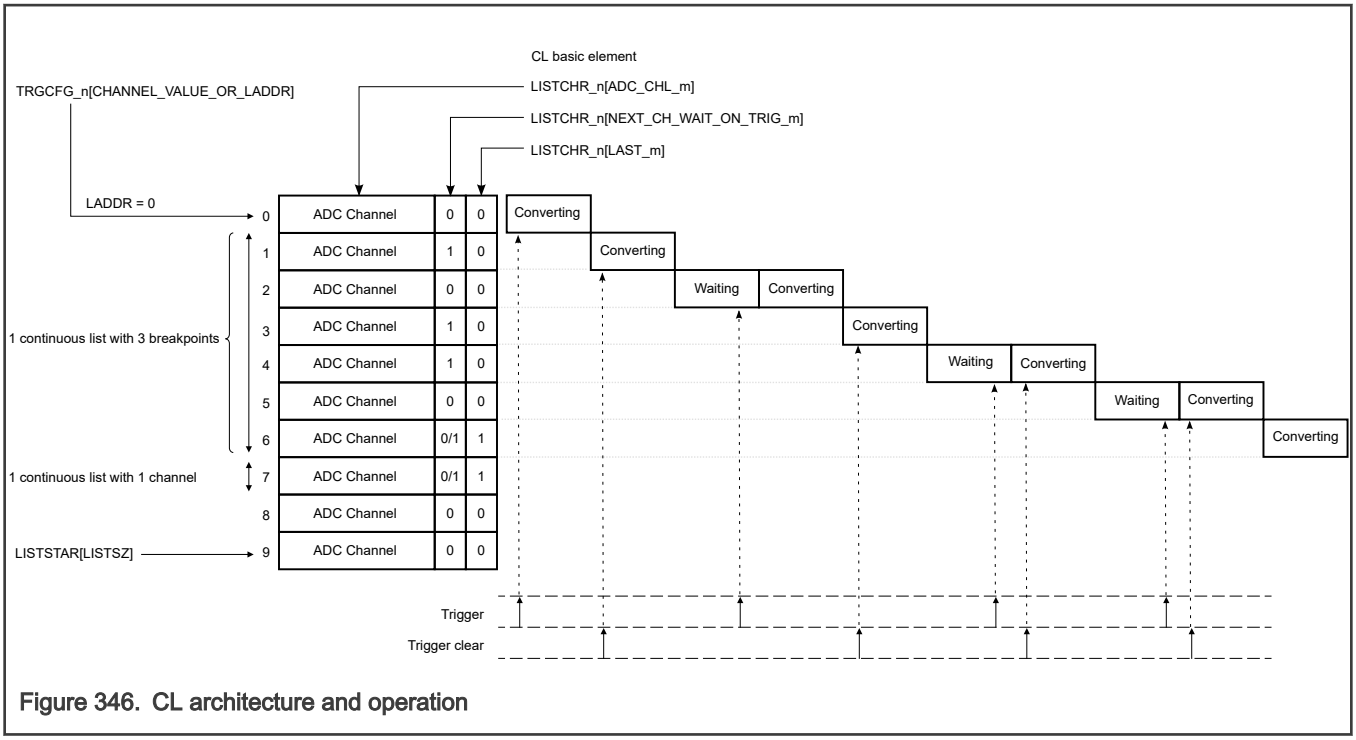



Figure 346. CL architecture and operation

### 61.3.2.3 CL channel rules

CL conversion obeys the following rules:

- The sequence of conversions executes until the Last Channel (LISTCHR_*n*[LAST_y] is 1).
- A CL conversion sequence precludes other conversions on the same ADC. If the CL is waiting (LISTCHR_*n*[NEXT_CH_WAIT_ON_TRIG_y] is 1), BCTU cannot route a non-CL conversion to that ADC (although that ADC remains available for normal and injected conversions).
- Two or more triggers for different ADCs may point to the same CL. This CL can drive parallel execution on the selected ADCs.
- If the Channel or CL Address (TRGCFG_x[CHANNEL_VALUE_OR_LADDR]) does not point to a valid list address, BCTU uses address 0 by default.
- When the CL pointer reaches the last available list address (LISTSTAR[LISTSZ] minus 1), the pointer wraps to list address 0.
- You can select more than one ADC per trigger using the ADC Select n (TRGCFG_x[ADC_SELn]) fields, but selecting more than one ADC is valid only for multiple parallel CL operation. If you select more than one ADC for a single-conversion trigger, BCTU ignores the trigger.

## 61.3.3 ADC conversion results access

### 61.3.3.1 Overview

The value of DATA_DEST determines access to ADC results.

- DATA_DEST = 000: ADC results are available in the ADC data registers.
- DATA_DEST = 001: ADC results are available in FIFO 1 Result Data (FIFO Result Data (FIFO1DR - FIFO2DR)). See Route conversion data to FIFO.
- DATA_DEST = 010: ADC results are available in FIFO 2 Result Data (FIFO Result Data (FIFO1DR - FIFO2DR)). See Route conversion data to FIFO.

You can access ADC results using flags and interrupt requests (see Access on interrupt request) or DMA (see Access on DMA request). Interrupt requests are based on Module Status (MSR) flags. ADC*n*DR provides the following information, with one register used for each ADC:

- The conversion results
- The channel that was used for the conversion
- The trigger input that initiated the conversion
- Whether a CL element generated the conversion
- Whether the data was the last conversion of a CL

#### 61.3.3.2 Access on interrupt request

If interrupts are enabled for ADC*n* Result Data (MCR[IEN*n*] is 1) and DMAs are disabled for ADC*n* Result Data (MCR[DMA*n*] is 0), BCTU generates an interrupt request for each available conversion data in ADC*n*DR. Because BCTU executes conversions on different ADCs in parallel, conversion results may be available on multiple ADCs at the same time. New Data (MSR[NDATAn] is 1) indicates that new conversion data is available in the corresponding ADC Data Result (ADC*n*DR). To clear the New Data flag, write 1 to MSR[NDATAn_CLR].

#### 61.3.3.3 Access on DMA request

If you enable DMA requests (MCR[DMA*n*]):

- You must select one channel for DMA per ADC.
- New conversion result data (ADC*n*DR) generates a DMA request.
- The DMA controller deasserts the DMA request and writes 1 to MSR[NDATAn_CLR] to clear the flag.

When an ADC generates new data (MSR[NDATAn]), BCTU processes the DMA request. See Clear a DMA request.

#### 61.3.3.4 Clear an interrupt request

BCTU clears an interrupt request when:

- The number of valid entries is less than or equal to the FIFOn watermark level (FIFOWM[WM_FIFO*n*]).
- You write 1 to FIFO watermark interrupt status (FIFOERR[WM_INT_FIFO*n*]).

#### 61.3.3.5 Clear a DMA request

BCTU clears a DMA request when:

- The number of valid entries is less than or equal to the FIFOn watermark level (FIFOWM[WM_FIFO*n*]).
- The DMA controller sends acknowledgment to BCTU.

#### 61.3.3.6 Clear an error-generated interrupt request

Clear an interrupt generated because of an error condition by writing 1 to the appropriate error flag:

- FIFOERR[OVR_ERR_FIFO1]
- FIFOERR[UNDR_ERR_FIFO1]
- FIFOERR[OVR_ERR_FIFO2]
- FIFOERR[UNDR_ERR_FIFO2]

#### 61.3.3.7 Conversion data overrun

If new ADC conversion data becomes available before you read the previous data (ADC*n*DR), then BCTU indicates a data overrun (MSR[DATAOVRn] is 1).

If you enable interrupts for the ADC (MCR[IEN*n*]), a data overrun also generates an interrupt request, regardless of the DMA request configuration.

To clear a data overrun (MSR[DATAOVRn]), write 1 to data overrun clear (MSR[DATAOVRn_CLR]). See Clear an error-generated interrupt request.

#### 61.3.3.8 Route conversion data to FIFO

BCTU allows you to route multiple ADC conversion results to one of 2 internal data FIFOs. Any trigger source can route results to an available FIFO based on the trigger index. Each FIFO has a dedicated counter (FIFOCNTR[CNTR_FIFO*n*]) to indicate the number of entries in the FIFO.

Each FIFO also has its own result data register (FIFO*n*DR) that contains the following information about the last conversion result routed to the FIFO.

- ADC result data
- Trigger source index
- ADC number
- ADC channel number

#### 61.3.3.9 Read ADC result from FIFO

Each FIFO has DMA and interrupt interfaces. The DMA interface reads data. The interrupt interface indicates watermark and error conditions:

- An overrun error (FIFOERR[OVR_ERR_FIFO*n*]) occurs when you attempt to write data to an already full FIFO.
- An underrun error (FIFOERR[UNDR_ERR_FIFO*n*]) occurs when you attempt to read data from an empty FIFO.
- BCTU asserts a watermark interrupt request if the number of valid FIFO entries exceeds the watermark level (FIFOWM[WM_FIFO*n*]).

You can read a FIFO:

- by enabling DMA requests (FIFOCR[DMA_EN_FIFO*n*])
- by polling the FIFO full status (FIFOSR[FULL_FIFO*n*])

See Configuration of DMA and interrupt behavior.

#### 61.3.3.10 Configuration of DMA and interrupt behavior

**Table 350. Configuration of DMA and interrupt behavior**

| FIFOCR[DMA_EN_FIFO*n*] | FIFOCR[IEN_FIFO*n*] | Condition |
|---|---|---|
| 0 | 0 | No DMA request; no interrupt request |
| 0 | 1 | Interrupt request when FIFO exceeds watermark level; interrupt request on error |
| 1 | 0 | DMA request when FIFO exceeds watermark level |
| 1 | 1 | DMA request when FIFO exceeds watermark level; interrupt request on error |

## 61.3.4 Multiple parallel conversions (MPC)

### 61.3.4.1 Overview

The MPC mechanism allows you to execute a series of simultaneous ADC conversions by selecting more than one ADC for any CL-configured trigger.

### 61.3.4.2 Configure MPC

To configure a CL to perform MPC:

1. Select CL for the trigger resolution (write 1 to TRGCFG_*m*[TRS]).
2. Select more than one ADC for the trigger (TRGCFG_*m*[ADC_SEL*n*]).
3. Specify the CL address for the first CL element (TRGCFG_*m*[CHANNEL_VALUE_OR_LADDR]).

### 61.3.4.3 CL address selection for MPC

When the trigger input for the CL asserts, BCTU uses the channel or list address (CHANNEL_VALUE_OR_LADDR) as an index into the CL for the lowest numbered ADC selected (ADC_SEL*n*), and each subsequent element of the CL for each remaining selected ADC. The set of selected ADC channels perform their conversions in parallel. When that set of conversions is complete, the next set of CL elements determine the next set of ADC channels to perform parallel conversions, as illustrated in:

- Example: MPC with two ADCs
- Example: MPC with three ADCs

### 61.3.4.4 Example: MPC with two ADCs

The following figure shows the example of MPC with two ADCs.

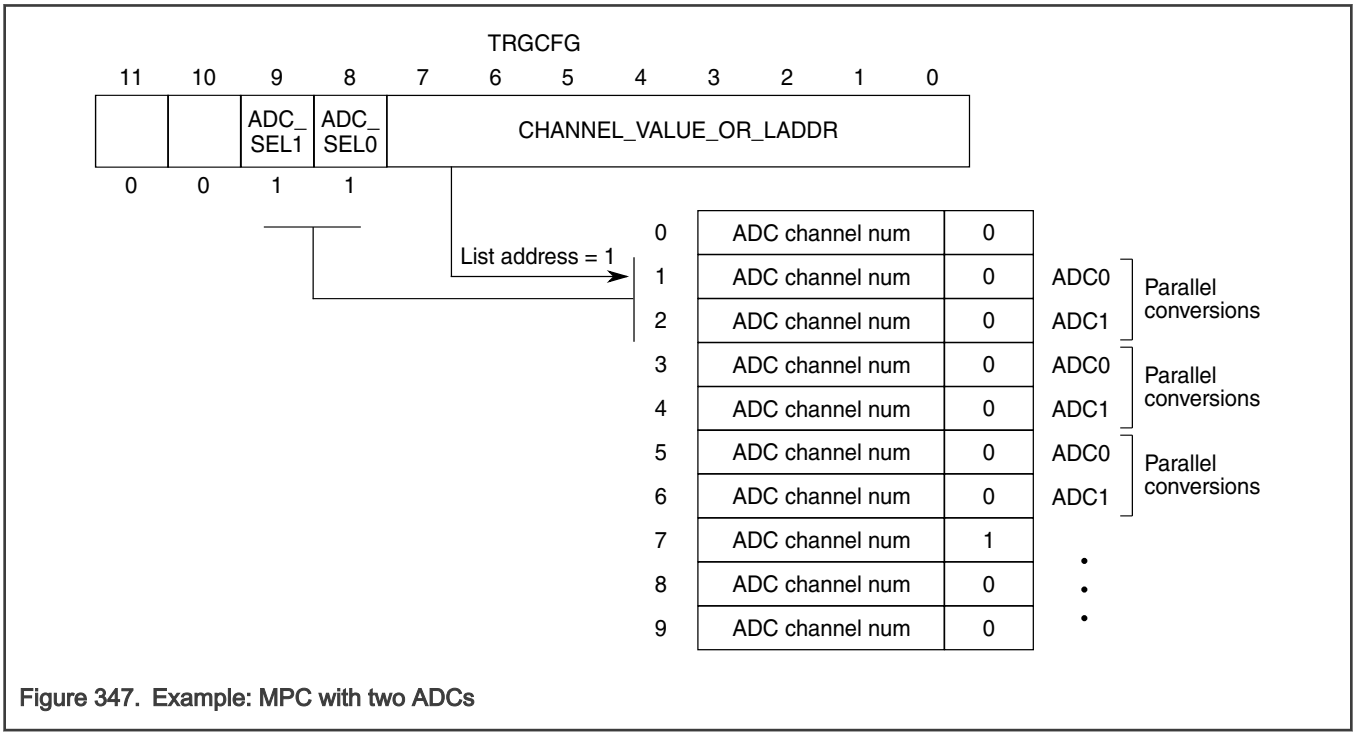



**Figure 347. Example: MPC with two ADCs**

### 61.3.4.5 Example: MPC with three ADCs

The following figure shows the example of MPC with three ADCs.

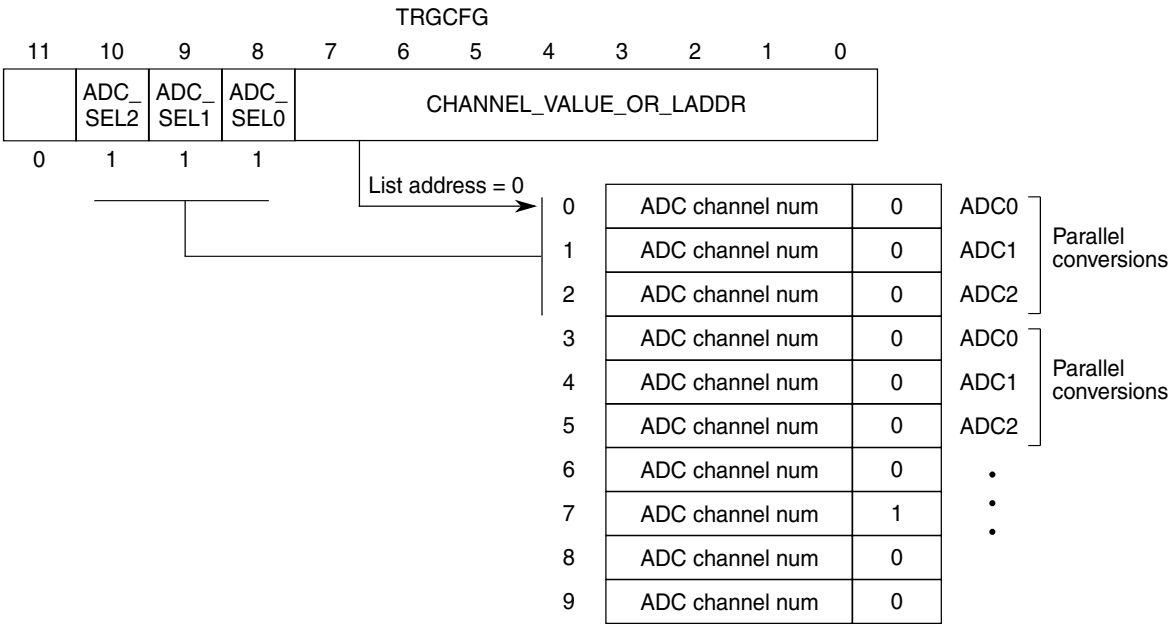



Figure 348. Example: MPC with three ADCs

#### 61.3.4.6 MPC priority rules

MPC follows the same priority rules as for single conversions, which is based on the trigger input number: the lower the trigger, the higher the priority. However, MPC cannot start until all ADCs selected for the CL are available; that is, they are not currently engaged in conversion. For example, if a trigger input asserts for a CL using ADC0 and ADC1, BCTU does not initiate the CL until ADC0 and ADC1 are both available.

Also, if ADC0 is available but there is a higher priority trigger for ADC1, the MPC does not start. If trigger inputs assert for two MPCs that have at least one ADC in common, the lower trigger number has higher priority. A priority conflict occurs only when separate requested CLs have at least one ADC in common. If the MPCs use different ADCs, there is no conflict and the MPCs execute at the same time.

### 61.3.5 Modes

#### 61.3.5.1 Module Disable (Low Power state)

Module disable (MCR[MDIS]) stops the BCTU clock, resulting in reduced power consumption.

When you disable BCTU (write 1 to MDIS), it:

- Waits for the completion of any in-progress conversions before stopping the clock.
- Ignores all trigger inputs.
- Does not trigger any ADC conversion requests.
- Maintains any asserted trigger inputs, which you can clear if necessary.

When you re-enable BCTU (write 0 to MDIS) and global triggers are enabled (MCR[GTRGEN]), BCTU processes all active asserted trigger inputs and sends conversion requests to ADCs according to the priority scheme.

### 61.3.6 Clocking

This module has two clock ports.

1. ipg_clk_bctu - is the main clock of complete BCTU IP.

2. ipg_clk_s - is the gated IPG clock to be used in BCTU registers only.

### 61.3.7 Software reset

You can reinitialize all BCTU registers to their reset values with software reset (MCR[Software_Reset]).

Software reset does not clear any active trigger inputs, but it does put the ADC interface into the Reset state. Therefore, to avoid data incoherency, you must terminate any BCTU-triggered conversions or wait until they are complete before asserting software reset.

### 61.3.8 Interrupts and DMA requests

**Table 351. Interrupts and DMA requests**

| Event | Enable interrupt | Enable DMA request | Description |
|---|---|---|---|
| Trigger Flag | MCR[TRGEN] | — | Interrupt request on assertion of trigger flag (MSR[TRGF]) |
| ADC Data Ready | MCR[IEN*n*] | MCR[DMA*n*] | Interrupt or DMA request when ADC result data is available (ADC*n*DR) |
| CL Last | MCR[LIST_IEN] | — | Interrupt request when the last conversion of any CL executes |
| FIFO Watermark | FIFOCR[IEN_FIFO*n*] | FIFOCR[DMA_EN_FIFO*n*] | Interrupt or DMA request when the number of valid FIFO entries exceeds the FIFO |

## 61.4 BCTU register descriptions

All BCTU registers are 32 bits wide. 8-bit write operations are allowed. The BCTU module does not include any coherence mechanism, so application software must ensure coherency.

**NOTE**

- You must always write 0 to reserved writable registers and fields. Read accesses of reserved registers and fields do not return meaningful data.
- For registers with write protection, if protection is active and you attempt to write to those registers, register content is unaffected and BCTU issues a transfer error.

### 61.4.1 BCTU memory map

BCTU base address: 4008_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Module Configuration (MCR) | 32 | RW | 0000_0000h |
| 8h | Module Status (MSR) | 32 | RW | 0000_0000h |
| 18h - 134h | Trigger Configuration (TRGCFG_0 - TRGCFG_71) | 32 | RW | 0000_0000h |
| 228h | Write Protection (WRPROT) | 32 | RW | 0000_0000h |
| 22Ch | Software Trigger 1 (SFTRGR1) | 32 | RW | 0000_0000h |
| 230h | Software Trigger 2 (SFTRGR2) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 234h | Software Trigger 3 (SFTRGR3) | 32 | RW | 0000_0000h |
| 23Ch - 244h | ADCn Result Data (ADC0DR - ADC2DR) | 32 | R | 0000_0000h |
| 24Ch | CL Size Status (LISTSTAR) | 32 | R | 0000_0020h |
| 250h - 28Ch | CL Channel Address (LISTCHR_0 - LISTCHR_15) | 32 | RW | 0000_0000h |
| 450h - 454h | FIFO Result Data (FIFO1DR - FIFO2DR) | 32 | R | 0000_0000h |
| 460h | FIFO Control (FIFOCR) | 32 | RW | 0000_0000h |
| 464h | FIFO Watermark Configuration (FIFOWM) | 32 | RW | 0000_0000h |
| 468h | FIFO Error/Status (FIFOERR) | 32 | RW | 0000_0000h |
| 46Ch | FIFO Status (FIFOSR) | 32 | R | 0000_0000h |
| 470h | FIFO Counter (FIFOCNTR) | 32 | R | 0000_0000h |

### 61.4.2 Module Configuration (MCR)

#### Offset

| Register | Offset |
|---|---|
| MCR | 0h |

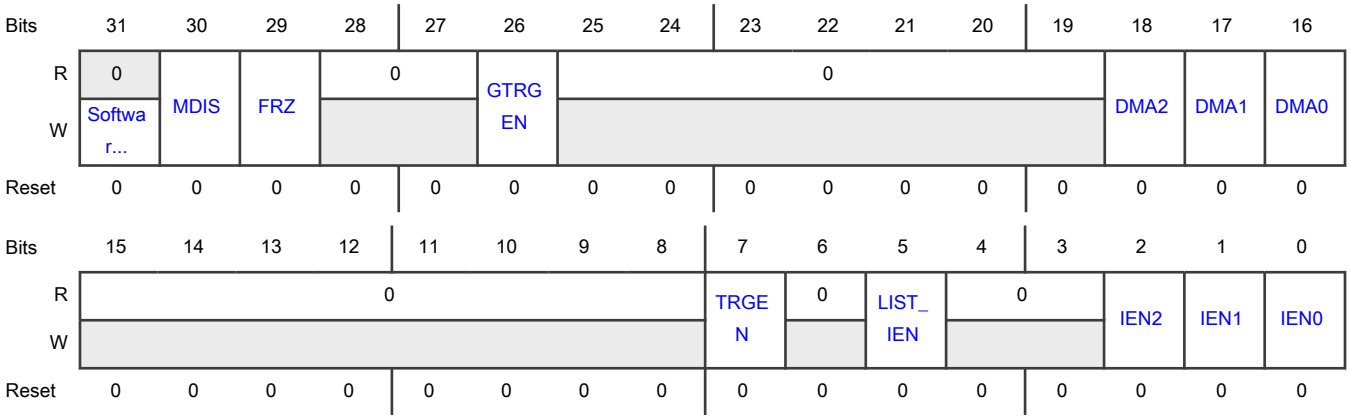
#### Function

Configures the following aspects of BCTU behavior:

- Low power
- Debug
- Trigger control
- DMA control
- Interrupt control

Access: User read/write

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>Software_Reset | Software Reset<br>Asserts soft reset, which initializes all BCTU registers and internal state machines to their reset values.<br>0b - Deasserts<br>1b - Asserts |
| 30<br>MDIS | Module Disable<br>Disables BCTU by putting it in Low Power state (see Module Disable (Low Power state)).<br>0b - Enable (normal operation)<br>1b - Disable (low-power operation) |
| 29<br>FRZ | Debug Freeze<br>Disables all hardware trigger inputs but leaves the software trigger enabled. Debug Freeze isolates BCTU from external triggers and allows you to manually trigger a conversion and read the conversion result.<br>0b - Disables<br>1b - Enables |
| 28-27<br>— | Reserved |
| 26<br>GTRGEN | Global Trigger Enable<br>Enables all individually enabled trigger inputs at the same time. For any given input trigger to be active, you must both enable the individual trigger (TRGCFG_*n*[TRIGEN]) and Global Trigger Enable.<br>See Configure all triggers.<br>0b - Disable<br>1b - Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 25-19<br>— | Reserved |
| 18-16<br>DMAn | Enable ADCnDR DMA<br>Enables DMA operation when new data is available in ADC*n*DR.<br>0b - Disable<br>1b - Enable |
| 15-8<br>— | Reserved |
| 7<br>TRGEN | Trigger Interrupt Request Enable<br>Enables an interrupt request on a Trigger Flag (MSR[TRGF]).<br>0b - Disable<br>1b - Enable |
| 6<br>— | Reserved |
| 5<br>LIST_IEN | CL Interrupt Enable<br>Enables an interrupt request for the last conversion in a CL (both ADC*n*DR[LIST] and ADC*n*DR[LAST] are 1).<br>0b - Disable<br>1b - Enable |
| 4-3<br>— | Reserved |
| 2-0<br>IENn | Interrupt Enable For ADCnDR<br>Enables an interrupt request when BCTU writes a new conversion result to ADC Data (ADC*n*DR). Also enables an interrupt request on an overrun in ADC*n*DR, which indicates the previous data has been overwritten by a new conversion result. See also MSR[NDATA*n*] and MSR[DATAOVR*n*]].<br>Enabling ADC*n*DR DMA (DMA*n* field) disables the interrupt for new data in ADC*n*DR, but the interrupt for ADC*n*DR data overrun remains enabled.<br>0b - Disable<br>1b - Enable |

## 61.4.3 Module Status (MSR)

### Offset

| Register | Offset |
|---|---|
| MSR | 8h |

### Function

Indicates the status of various BCTU functions, including:

- Whether conversion data is available for each ADC
- CL operation, such as the last conversion command executed in a CL
- Whether BCTU triggered any of the ADCs

All flag fields become 0 upon assertion of a low-power (software) or clock-stop (hardware) request. You can turn off individual flags by writing 1 to the field.

Access: User read/write

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | TRGF_CLR | | | | | LIST2_L... | LIST1_L... | LIST0_L... | | DATAOVR... | DATAOVR... | DATAOVR... | | NDATA2_... | NDATA1_... | NDATA0_... |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TRGF | 0 | | | | LIST2_L... | LIST1_L... | LIST0_L... | 0 | DATAOVR2 | DATAOVR1 | DATAOVR0 | 0 | NDATA2 | NDATA1 | NDATA0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>TRGF_CLR | TRGF Clear<br>Changes Trigger Flag (TRGF) to 0.<br>0b - No action<br>1b - Changes to 0 |
| 30-27<br>— | Reserved |
| 26-24 | CL n Last Clear |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| LISTn_Last_CLR | Changes CL *n* Last (LIST*n*_Last) to 0.<br>0b - No action<br>1b - Changes to 0 |
| 23<br>— | Reserved |
| 22-20<br>DATAOVRn_CLR | DATAOVRn Clear<br>Changes Data Overrun *n* (DATAOVR*n*) to 0.<br>0b - No action<br>1b - Changes to 0 |
| 19<br>— | Reserved |
| 18-16<br>NDATAn_CLR | New Data Clear<br>Changes New Data *n* (NDATA*n*) to 0.<br>0b - No action<br>1b - Changes to 0 |
| 15<br>TRGF | Trigger Flag<br>Indicates at least one ADC was triggered.<br>0b - No ADC triggered<br>1b - An ADC was triggered |
| 14-11<br>— | Reserved |
| 10-8<br>LISTn_Last | CL n Last Conversion<br>Indicates that ADCn has executed the last conversion in a CL.<br>0b - Last conversion not complete<br>1b - Last conversion complete |
| 7<br>— | Reserved |
| 6-4<br>DATAOVRn | Data Overrun n<br>Indicates that the data in ADCn has been overwritten with new data.<br>0b - Data not overwritten<br>1b - Data overwritten |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 3<br>— | Reserved |
| 2-0<br>NDATAn | New Data n<br>Indicates that new conversion data is available for ADCn.<br>0b - Not available<br>1b - Available |

### 61.4.4 Trigger Configuration (TRGCFG_0 - TRGCFG_71)

#### Offset

For a = 0 to 71:

| Register | Offset |
|---|---|
| TRGCFG_a | 18h + (a × 4h) |

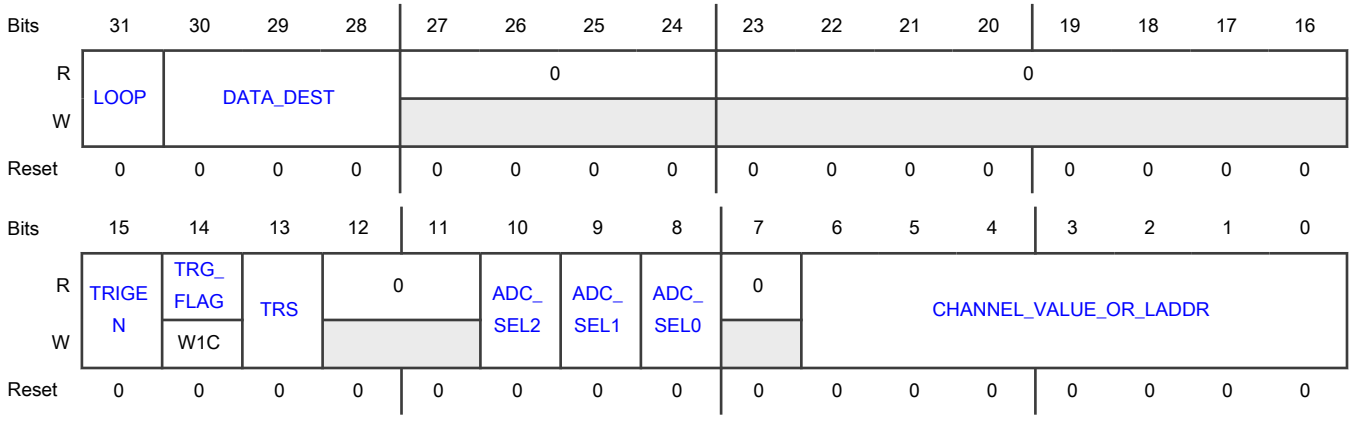
#### Function

Configures the corresponding trigger:

- Specifies the target ADC and ADC channel to use for conversion.
- Specifies whether to use a CL or a single ADC conversion command.

Access: User read/write

#### Diagram



**Fields**

| Field | Function |
|---|---|
| 31<br>LOOP | Loop<br>Specifies that the current trigger executes in a loop; that is, the triggered conversion repeats until you disable Loop. Each loop-triggered conversion wins the priority evaluation among all pending triggers, and executes only if it is the highest-priority triggered conversion at the moment the conversion starts.<br>0b - Disable<br>1b - Enable |
| 30-28<br>DATA_DEST | Data Destination<br>Specifies the destination for storing the conversion results. You can route each ADC's conversion result to either of the following:<br>• ADC-specific data registers<br>• One of the available FIFOs<br>BCTU routes conversion results to different destinations based on the trigger index that requested the conversion. You cannot duplicate the same data in more than one destination.<br>000b - ADC-specific data registers<br>001b - FIFO1<br>010b - FIFO2<br>011b - Reserved<br>All other values are reserved. |
| 27-24<br>— | Reserved |
| 23-16<br>— | Reserved |
| 15<br>TRIGEN | Trigger Enable<br>Enables hardware input triggers to initiate ADC conversions. This field does not affect software triggers.<br>0b - Disable<br>1b - Enable |
| 14<br>TRG_FLAG | Trigger Flag<br>Indicates the trigger input is asserted and an ADC conversion is in progress. When the conversion completes, BCTU changes this field to 0. Writing 1 to this field causes the input trigger event to clear when the in-progress conversion completes, and prevents this trigger from initiating a new ADC conversion until the input trigger is asserted again.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No ongoing ADC conversion initiated<br>1b - Ongoing ADC conversion initiated<br>When writing<br>0b - No action<br>1b - Changes to 0 |
| 13<br>TRS | Trigger Resolution<br>Selects single conversion or CL conversions.<br>0b - Single conversion<br>1b - CL conversions |
| 12-11<br>— | Reserved |
| 10-8<br>ADC_SELn | ADC Select n<br>Selects ADC *n* for conversion.<br>0b - Deselects<br>1b - Selects |
| 7<br>— | Reserved |
| 6-0<br>CHANNEL_VALUE_OR_LADDR | Channel or CL Address<br>Depending on the value of TRS:<br>• 0: Specifies the ADC channel for conversion<br>• 1: Specifies the address of the first CL position to execute |

## 61.4.5 Write Protection (WRPROT)

### Offset

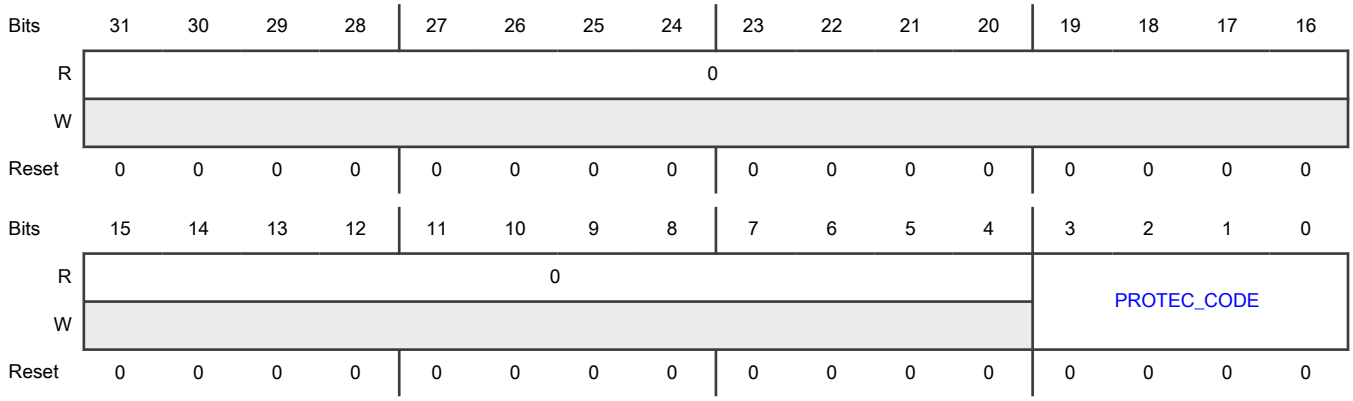
| Register | Offset |
|---|---|
| WRPROT | 228h |

### Function

Enables write protection at reset exit. To allow writes to protected registers, you must write a special code into PROTEC_CODE. There are two codes to remove protection: one that permanently disables the protection; one that disables protection during one write cycle to a protected register, after which the protection code returns to the initial (protected) value of 0000b.

Access: User read/write

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>PROTEC_CODE | Protection Code<br>Controls the protection of write-protected registers.<br>**NOTE**<br>You can write this field only in Normal mode (MCR[MDIS] is 0).<br>0000b - Enable protection<br>1001b - Disable protection for one write cycle<br>1010b - Disable protection permanently |

## 61.4.6 Software Trigger 1 (SFTRGR1)

### Offset

| Register | Offset |
|---|---|
| SFTRGR1 | 22Ch |

### Function

Allows you to trigger conversions on the associated triggers.

To use a software trigger, you must first disable the corresponding hardware trigger (TRGCFG_x[TRIGEN]).

BCTU writes 0 to a software trigger field on completion of the triggered conversion.

Access: User read/write; this register is write protected by Write Protection (WRPROT). If protection is enabled, writes to this register generate a transfer error and do not modify its content.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SFTRG31 | SFTRG30 | SFTRG29 | SFTRG28 | SFTRG27 | SFTRG26 | SFTRG25 | SFTRG24 | SFTRG23 | SFTRG22 | SFTRG21 | SFTRG20 | SFTRG19 | SFTRG18 | SFTRG17 | SFTRG16 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SFTRG15 | SFTRG14 | SFTRG13 | SFTRG12 | SFTRG11 | SFTRG10 | SFTRG9 | SFTRG8 | SFTRG7 | SFTRG6 | SFTRG5 | SFTRG4 | SFTRG3 | SFTRG2 | SFTRG1 | SFTRG0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>SFTRGm | Software Trigger m<br>Starts an ADC conversion.<br>0b - No effect<br>1b - Trigger conversion |

## 61.4.7 Software Trigger 2 (SFTRGR2)

### Offset

| Register | Offset |
|---|---|
| SFTRGR2 | 230h |

### Function

Allows you to trigger conversions on the associated triggers.

To use a software trigger, you must first disable the corresponding hardware trigger (TRGCFG_*n*[TRIGEN]).

BCTU writes 0 to a software trigger field on completion of the triggered conversion.

Access: User read/write; this register is write protected by Write Protection (WRPROT). If protection is enabled, writes to this register generate a transfer error and do not modify its content.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SFTRG63 | SFTRG62 | SFTRG61 | SFTRG60 | SFTRG59 | SFTRG58 | SFTRG57 | SFTRG56 | SFTRG55 | SFTRG54 | SFTRG53 | SFTRG52 | SFTRG51 | SFTRG50 | SFTRG49 | SFTRG48 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SFTRG47 | SFTRG46 | SFTRG45 | SFTRG44 | SFTRG43 | SFTRG42 | SFTRG41 | SFTRG40 | SFTRG39 | SFTRG38 | SFTRG37 | SFTRG36 | SFTRG35 | SFTRG34 | SFTRG33 | SFTRG32 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>SFTRGm | Software Trigger m<br>Starts an ADC conversion.<br>0b - No effect<br>1b - Trigger conversion |

## 61.4.8 Software Trigger 3 (SFTRGR3)

### Offset

| Register | Offset |
|---|---|
| SFTRGR3 | 234h |

### Function

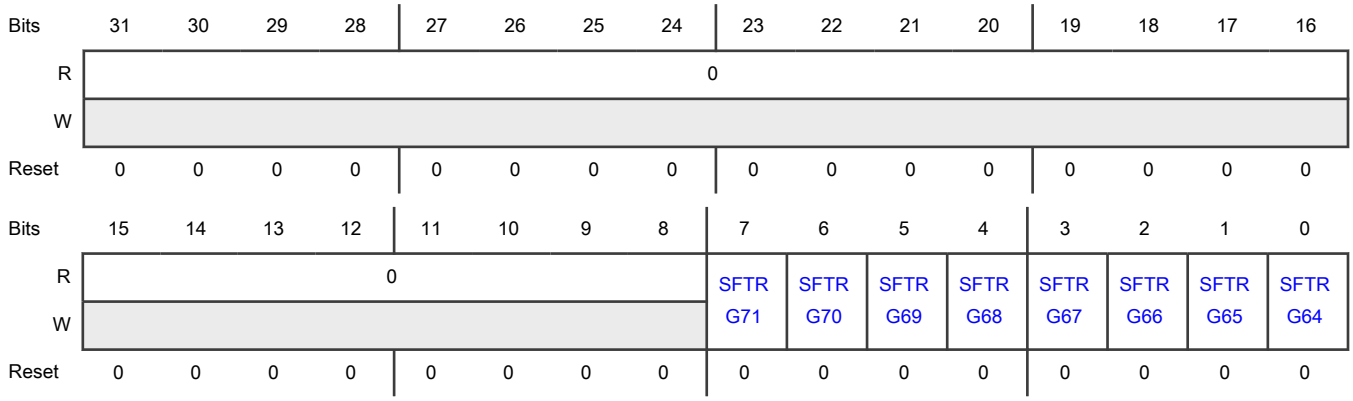
Allows you to trigger conversions on the associated triggers.

To use a software trigger, you must first disable the corresponding hardware trigger (TRGCFG_*n*[TRIGEN]).

BCTU writes 0 to a software trigger field on completion of the triggered conversion.

Access: User read/write; this register is write protected by Write Protection (WRPROT). If protection is enabled, writes to this register generate a transfer error and do not modify its content.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>SFTRGm | Software Trigger m<br>Starts an ADC conversion.<br>0b - No effect<br>1b - Trigger conversion |

## 61.4.9 ADCn Result Data (ADC0DR - ADC2DR)

### Offset

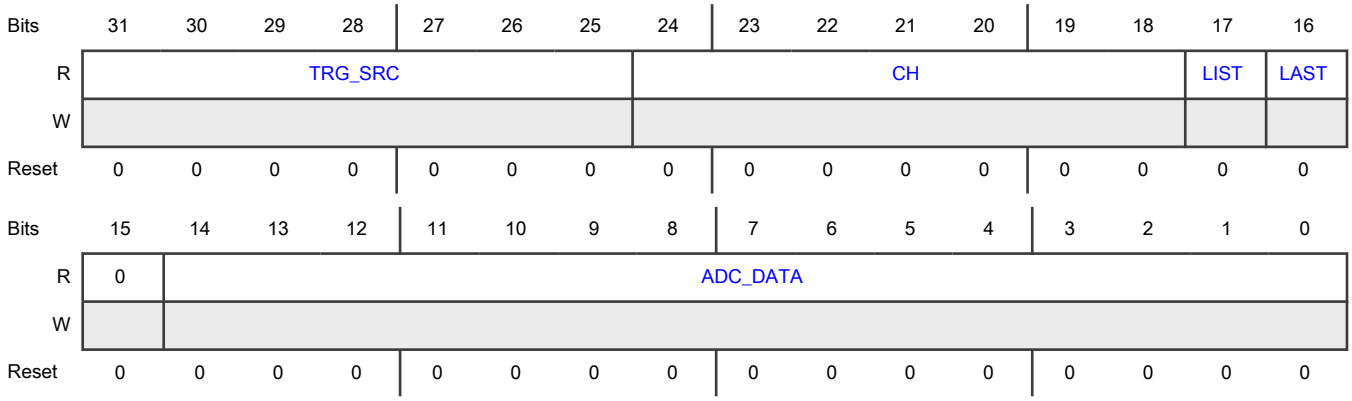
| Register | Offset |
|---|---|
| ADC0DR | 23Ch |
| ADC1DR | 240h |
| ADC2DR | 244h |

### Function

Provides ADC conversion result data. Also indicates the ADC channel used in that conversion, the trigger source that initiated it, and whether conversion was initiated from a CL.

Access: User read

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-25<br>TRG_SRC | Trigger Source<br>Contains the trigger number used to trigger the conversion. |
| 24-18<br>CH | Channel<br>Contains the ADC channel used for the conversion. |
| 17<br>LIST | List<br>Indicates whether the conversion was part of a CL or a single conversion trigger.<br>0b - Single conversion<br>1b - CL |
| 16<br>LAST | Last<br>For CL conversions, indicates this conversion result is the last conversion of the CL.<br>0b - Not the last conversion of a CL, or not a CL conversion<br>1b - Last conversion of a CL |
| 15<br>— | Reserved |
| 14-0<br>ADC_DATA | ADC Data<br>Contains the data from the conversion. |

## 61.4.10 CL Size Status (LISTSTAR)

### Offset

| Register | Offset |
|---|---|
| LISTSTAR | 24Ch |

### Function

Indicates the size of the CL. Each element of the CL contains the channel number used in one conversion.

Access: User read

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>LISTSZ | CL Size<br>Indicates the size of the CL in number of elements. |

## 61.4.11 CL Channel Address (LISTCHR_0 - LISTCHR_15)

### Offset

For m = 0 to 15:

| Register | Offset |
|---|---|
| LISTCHR_m | 250h + (m × 4h) |

### Function

Configures these attributes for two CL elements:

- ADC channel number to use for conversion
- Whether the element is the last element of a CL

Access: User read/write

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LAST_y | NEXT_CH... | 0 | | 0 | | | | 0 | ADC_CH_y | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LAST_y_... | NEXT_CH... | 0 | | 0 | | | | 0 | ADC_CHL_y_plus_1 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>LAST_y | Last Channel<br>Specifies this element as the last channel in the CL.<br>0b - Not last<br>1b - Last channel in CL |
| 30<br>NEXT_CH_WAIT_ON_TRIG_y | Next Channel Wait For Trigger<br>Instructs CL execution to stop after executing the current ADC channel. Execution begins again when the same trigger, which started the CL, reoccurs.<br>**NOTE**<br>If this is the last channel in the CL (LAST_y), you must write 0 to this field.<br>0b - CL executes continuously<br>1b - CL stops executing |
| 29-28<br>— | Reserved |
| 27-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>ADC_CH_y | ADC Channel Selection<br>Specifies the ADC channel to use for the *y* element of the CL, where *y* equals (2 x *m*). |
| 15<br>LAST_y_plus_1 | Last Channel Plus 1<br>Specifies this element as the next-to-last channel in the CL. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Not next-to-last<br>1b - Next-to-last channel in CL |
| 14<br>NEXT_CH_WAIT_ON_TRIG_y_plus_1 | Next Channel Wait For Trigger Plus 1<br>Instructs CL execution to stop after executing the current ADC channel. Execution begins again when the same trigger, which started the CL, reoccurs.<br>**NOTE**<br>If this is the last channel in the CL (LAST_y_plus_1), you must write 0 to this field.<br>0b - CL executes continuously<br>1b - CL stops executing |
| 13-12<br>— | Reserved |
| 11-8<br>— | Reserved |
| 7<br>— | Reserved |
| 6-0<br>ADC_CHL_y_plus_1 | ADC Channel Selection Plus 1<br>Specifies the ADC channel to use for the (*y* + 1) element of the CL, where *y* equals (2 x *m*). |

## 61.4.12 FIFO Result Data (FIFO1DR - FIFO2DR)

### Offset

| Register | Offset |
|---|---|
| FIFO1DR | 450h |
| FIFO2DR | 454h |

### Function

Contains the ADC conversion result and information about the conversion:

- ADC channel
- Trigger source
- ADC number

Access: User read

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TRG_SRC | | | | | | | | CH | | | | | | ADC_NUM | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31-25<br>TRG_SRC | Trigger Source<br>Indicates the trigger number used to trigger the conversion. |
| 24-18<br>CH | Channel<br>Indicates the ADC channel used for the conversion. |
| 17-16<br>ADC_NUM | ADC Number<br>Indicates the ADC used for conversion.<br>00b - ADC 0<br>01b - ADC 1<br>10b - ADC 2<br>11b - ADC 3 |
| 15<br>— | Reserved |
| 14-0<br>ADC_DATA | ADC Data<br>Contains the conversion data. |

## 61.4.13 FIFO Control (FIFOCR)

### Offset

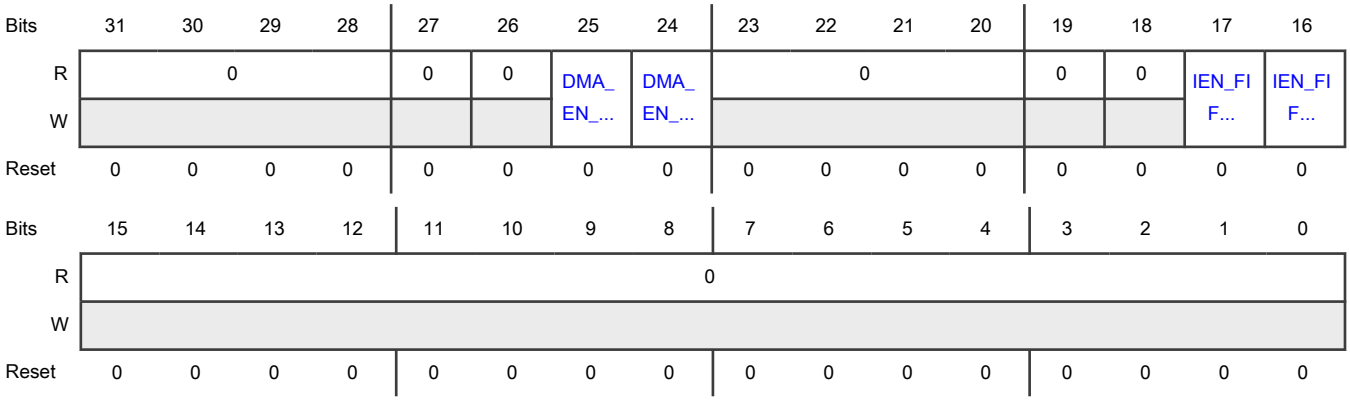
| Register | Offset |
|---|---|
| FIFOCR | 460h |

### Function

Provides control of FIFO-specific DMA, interrupts, and error interrupts.

Access: User read/write

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27<br>— | Reserved |
| 26<br>— | Reserved |
| 25-24<br>DMA_EN_FIFOn | FIFOn DMA Enable<br>Enables a DMA request when the number of valid FIFO entries is greater than the watermark level for that FIFO.<br>0b - Disable<br>1b - Enable |
| 23-20<br>— | Reserved |
| 19<br>— | Reserved |
| 18<br>— | Reserved |
| 17-16<br>IEN_FIFOn | FIFOn Interrupt Enable<br>Enables an interrupt request if the number of valid FIFO entries is greater than the watermark level for the FIFO. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disable<br>1b - Enable |
| 15-0<br>— | Reserved |

## 61.4.14 FIFO Watermark Configuration (FIFOWM)

### Offset

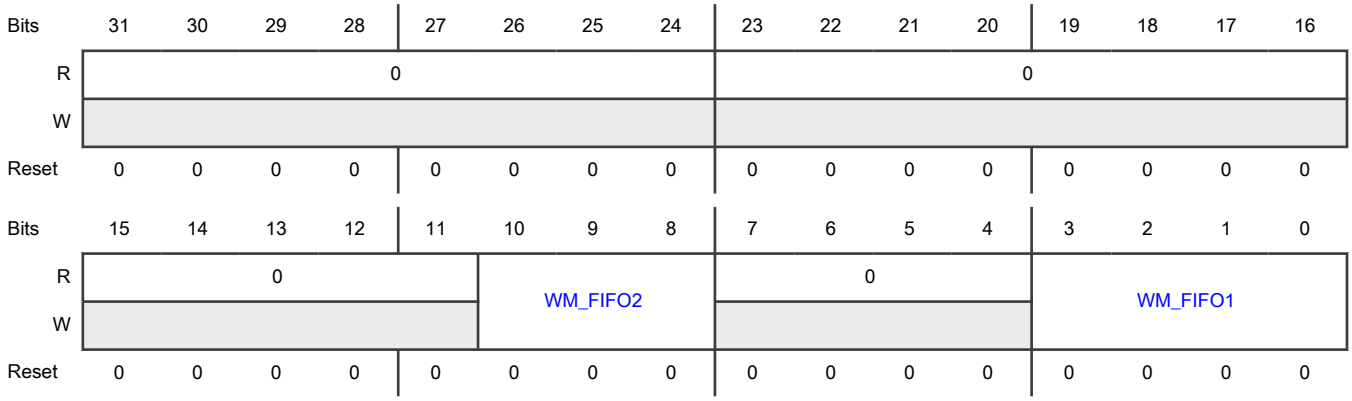
| Register | Offset |
|---|---|
| FIFOWM | 464h |

### Function

Specifies the FIFO watermark levels. If the number of active FIFO entries exceeds the watermark level, a DMA or interrupt request can be generated. See FIFO Control (FIFOCR).

Access: User read/write

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 15-11<br>— | Reserved |
| 10-8<br>WM_FIFO2 | FIFO2 Watermark Level<br>Specifies the watermark level for FIFO2. |
| 7-4<br>— | Reserved |
| 3-0<br>WM_FIFO1 | FIFO1 Watermark Level<br>Specifies the watermark level for FIFO1. |

## 61.4.15 FIFO Error/Status (FIFOERR)

### Offset

| Register | Offset |
|---|---|
| FIFOERR | 468h |

### Function

Allows you to clear FIFO error and status flags. Each FIFO has its own error and status flags that can be configured to generate an interrupt. FIFO Status (FIFOSR) maintains the status of these flags. BCTU writes a 1 to these flags, and you must then clear them using FIFOERR.

Access: User read/write

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | 0 | | UNDR_ER... | OVR_ERR... | UNDR_ER... | OVR_ERR... | 0 | | | | 0 | | WM_INT_... | WM_INT_... |
| W | | | | | W1C | W1C | W1C | W1C | | | | | | | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-28<br>— | Reserved |
| 27<br>UNDR_ERR_FIFO2 | Underrun Error Flag<br>Indicates you have attempted to read from an empty FIFO.<br>0b - No underrun<br>1b - Underrun |
| 26<br>OVR_ERR_FIFO2 | Overrun Error Flag<br>Indicates you have attempted to write to a full FIFO.<br>0b - No overrun<br>1b - Overrun |
| 25<br>UNDR_ERR_FIFO1 | Underrun Error Flag<br>Indicates you have attempted to read from an empty FIFO.<br>0b - No underrun<br>1b - Underrun |
| 24<br>OVR_ERR_FIFO1 | Overrun Error Flag<br>Indicates you have attempted to write to a full FIFO.<br>0b - No overrun<br>1b - Overrun |
| 23-20<br>— | Reserved |
| 19-18<br>— | Reserved |
| 17-16<br>WM_INT_FIFOn | FIFO Watermark Interrupt Status<br>Indicates the number of active entries in FIFOn exceeds the watermark level.<br>0b - Does not exceed watermark<br>1b - Exceeds watermark |
| 15-0<br>— | Reserved |

## 61.4.16 FIFO Status (FIFOSR)

### Offset

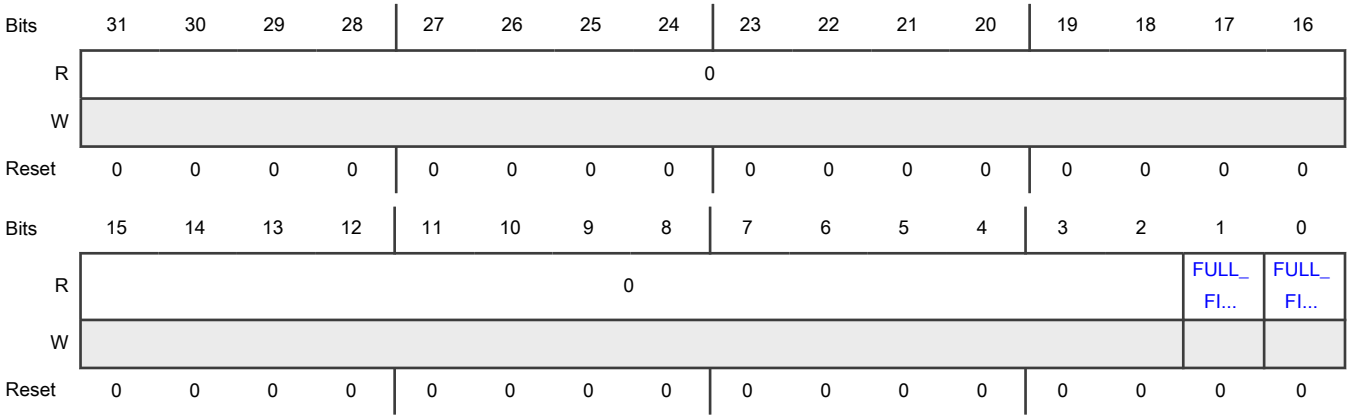
| Register | Offset |
|---|---|
| FIFOSR | 46Ch |

### Function

Contains the FIFO-full flags.

Access: User read

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1-0<br>FULL_FIFOn | FIFO Full<br>Indicates the FIFO is full.<br>0b - Not full<br>1b - Full |

## 61.4.17 FIFO Counter (FIFOCNTR)

### Offset

| Register | Offset |
|---|---|
| FIFOCNTR | 470h |

### Function

Indicates the number of active entries in each FIFO.

Access: User read

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | CNTR_FIFO2 | | | | 0 | | | | CNTR_FIFO1 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16<br>— | Reserved |
| 15-12<br>— | Reserved |
| 11-8<br>CNTR_FIFO2 | FIFO2 Counter<br>Indicates the number of active entries in FIFO2. See the chip-specific BCTU information for BCTU configuration. |
| 7-5<br>— | Reserved |
| 4-0<br>CNTR_FIFO1 | FIFO1 Counter<br>Indicates the number of active entries in FIFO1. See the chip-specific BCTU information for BCTU configuration. |

## 61.5 Glossary

**CL** Conversion list. The set of ADC channels, configured through the CL Channel Address (LISTCHR_0 - LISTCHR_15) registers, used for performing multiple sequential conversions initiated by a single trigger.

**MPC** Multiple parallel conversions

**PSI** Parallel side interface

**RWB** Read/write bus

**WDATA** Write Data bus

# Chapter 62
# Trigger MUX (TRGMUX)

## 62.1 Chip-specific TRGMUX information

### 62.1.1 TRGMUX input output configuration and instances

This chip has 1 instance of TRGMUX module.

The device supports the triggering scheme between peripherals. For the supported trigger sources and destination, see the TRGMUX connectivity file attached to this document.

This device has 16 pads (SIUL2) mapped from TRGMUX inputs and TRGMUX outputs are mapped to the eMIOS channels, hence two timers channels can use a single pin of the device to do input capture.

**While using TRGMUX, below points need to be taken care:**

- User must ensure the minimum pulse of 100 ns on SIUL2 pads when using them as trigger source on TRGMUX.
- Pulses which are visible on pads depends on the pad type. Different pad supports different frequencies. For more details on pad types and their respective bandwidth, see section "IO signal table" in "Signal Multiplexing" chapter.
- End of conversion (EOC) signals of ADC modules are mapped on TRGMUX inputs as trigger sources. The EOC signal is asserted after ADC conversion regardless whether conversion is signaled by polling flags, interrupt or DMA transfer. The signal shall not be used to start injected conversion on same ADC channel as it will overwrite current result register.
- The minimum pulse length requirement is 1.5X of destination clock, so that it gets properly sampled at the destination IP connected to TRGMUX outputs, otherwise there are chances of missing the triggers generated from source. For example, the trigger generated from PAD for ADC conversion should be kept high/low for at least 1.5X of ADC clock so that it gets sampled in ADC clock domain. To ensure this, pulse strechers have been placed before some of the hardware modules mapped on TRGMUX outputs.
- Some PADS are being shared by both ADC and TRGMUX. It is recommended that trigger initiated from such PADS should not be used to trigger a conversion on the ADC channel mapped on same PAD. Failing to do this will cause congestion on same PAD.
- Out of all the pads mapped on TRGMUX, first four pads have glitch filters. For details, see the TRGMUX connectivity file attached to this document. The trigger pulse width should honour the pulse width requirement as per Glitch Filter specifications. The same signal can be observed at output pin if the pulse width of the input data signal is more than 400ns and no output signal should be observed if the pulse width of the input data signal is less than 20ns. A signal with a pulse width that is between 20ns and 400 ns should not be applied as the behavior is not guaranteed.
- Trigger outputs are grouped peripheral-wise and have a common lock bit based on TRGMUX REGx. For instance, normal_trigger, injected_trigger and external_sync of ADC_0 are grouped onto TRGMUX_REG0 and have a common lock bit.

### 62.1.2 Pulse strechers in TRGMUX

TRGMUX has some hardware modules on its input side running at faster clock than some of the IPs present on output side. For instance, eMIOS reload outputs running at 160 MHz can trigger LPI2C trigger input clocked at 40 MHz, in that case there is a high chance that trigger from eMIOS will be missed. Following Pulse strechers are added for the IPs on output side of TRGMUX.

Table 352. Pulse strechers in TRGMUX

| Evaluation Parameter | ADC_0/1/2 external trigger to sync the start pulse | BCTU trigger 23/47/71 | FlexIO trigger input_0/1/2/3[1] | LPI2C_0 Trigger input | LPSPI_0/1/2 Trigger input | CM7_0/1_RXEV |
|---|---|---|---|---|---|---|
| **IP expects** | Synchronized single cycle pulse | Synchronized pulse | IP requirement is to have 2 cycle pulse of flexio_clk | IP requirement is to have 2 cycle pulse of lpi2c_clk | IP requirement is to have 2 cycle pulse of lpspi_clk | Single cycle pulse |
| **Frequency** | CORE_CLK | CORE_CLK | CORE_CLK | AIPS_SLOW_CLK | LPSPI_0 - PLAT_AIPS_CLK<br>LPSPI_1/2/3 - AIPS_SLOW_CLK | CORE_CLK |
| **Inside IP** | Used as combo signal | Flopped inside IP and clear after ADC conversion completed | Synchronized inside IP | Synchronized inside IP | Synchronized inside IP | - |
| **At SoC** | Since it an ASYNC signal and IP required synchronized single cycle pulse, so a pulse stretcher is added to convert pulse from any frequency domain into a signle cycle pulse of CORE_CLK | Since slow IP such as LPCMP or from PAD can also trigger BCTU and BCTU wants that trigger should clear after the ADC conversion. So a pulse stretcher is added which convert any size of pulse to a single cycle pulse of BCTU clock (CORE_CLK) | Pulse stretcher is added to convert any pulse into a two cyle pulse of FLEXIO_CLK | Pulse stretcher is added to convert any pulse into a two cyle pulse of LPI2C_CLK | Pulse stretcher is added to convert any pulse into a two cyle pulse of LPSPI_CLK | Pulse stretcher is added to convert any pulse into a single cyle pulse of CORE_CLK |
| **Output slot at which pulse stretcher is added** | 2, 6, 10 | 24, 25, 26 | 64, 65, 66, 67 | 84 | 88, 92, 96 | 156, 157 |

1. These pulse stretchers are available in MCXE31B variants only.

**NOTE**

The trigger outputs which have pulse stretcher before them, there should be atleast a gap of 5 cycle of destination clock for back to back trigger.

## 62.2 Overview

TRGMUX allows you to configure the trigger inputs for various peripherals.

### 62.2.1 Block diagram

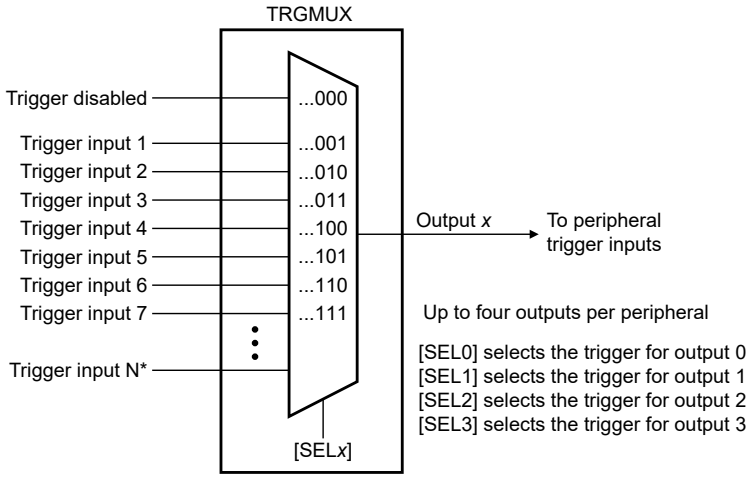



Figure 349. Block diagram

**NOTE**

Up to 255 trigger inputs may be available for SEL0, SEL1, and SEL2. For SEL3, up to 127 trigger inputs may be available. When the number of trigger inputs is 255, SEL3 is not available and becomes reserved. See the chip-specific TRGMUX information for details about trigger input and output configuration on this chip.

### 62.2.2 Features

- Configurable trigger sources for peripherals
- Dedicated TRGMUX register for each peripheral

## 62.3 Functional description

### 62.3.1 Clocking

This module has no clocking considerations.

### 62.3.2 Interrupts

This module has no interrupts.

## 62.4 External signals

This module has no external signals.

## 62.5 Initialization

This module does not require initialization.

## 62.6 TRGMUX register descriptions

### 62.6.1 TRGMUX memory map

You can only write to TRGMUX registers in Supervisor mode.

**Table 353. Select bit fields**

| Field | Description |
|---|---|
| SEL*x* | Specifies the MUX select for the peripheral trigger inputs. Use this field to select the trigger sources for peripheral modules.<br>0h - LOGIC 0 (VSS)<br>1h - LOGIC 1 (VDD)<br>2h - ADC12_0_EOC<br>3h - ADC12_1_EOC<br>4h - ADC12_2_EOC<br>5h - LPCMP_0_COUT output<br>6h - LPCMP_1_COUT output<br>7h - LPCMP_2_COUT output<br>8h - eDMA_eDMA_0 DONE<br>9h - eDMA_eDMA_1 DONE<br>Ah - eDMA_eDMA_16 DONE<br>Bh - eDMA_eDMA_17 DONE<br>Ch - eMIOS_0_RELOAD_OUT_CH[23]<br>Dh - eMIOS_0_RELOAD_OUT_CH[22]<br>Eh - eMIOS_0_RELOAD_OUT_CH[8]<br>Fh - eMIOS_0_RELOAD_OUT_CH[0]<br>10h - eMIOS_0_IPP_DO_eMIOS_CH[0]<br>11h - eMIOS_0_IPP_DO_eMIOS_CH[1]<br>12h - eMIOS_0_IPP_DO_eMIOS_CH[2]<br>13h - eMIOS_0_IPP_DO_eMIOS_CH[3]<br>14h - eMIOS_0_IPP_DO_eMIOS_CH[4]<br>15h - eMIOS_0_IPP_DO_eMIOS_CH[5]<br>16h - eMIOS_0_IPP_DO_eMIOS_CH[6]<br>17h - eMIOS_0_IPP_DO_eMIOS_CH[7]<br>18h - eMIOS_0_IPP_DO_eMIOS_CH[8]<br>19h - eMIOS_0_IPP_DO_eMIOS_CH[9]<br>1Ah - eMIOS_0_IPP_DO_eMIOS_CH[10]<br>1Bh - eMIOS_0_IPP_DO_eMIOS_CH[11]<br>1Ch - eMIOS_0_IPP_DO_eMIOS_CH[12]<br>1Dh - eMIOS_0_IPP_DO_eMIOS_CH[13] |

*Table continues on the next page...*

**Table 353. Select bit fields**

| Field | Description |
|---|---|
| | 1Eh - eMIOS_0_IPP_DO_eMIOS_CH[14]<br>1Fh - eMIOS_0_IPP_DO_eMIOS_CH[15]<br>20h - eMIOS_0_IPP_DO_eMIOS_CH[22]<br>21h - eMIOS_0_IPP_DO_eMIOS_CH[23]<br>22h - eMIOS_1_RELOAD_OUT_CH[23]<br>23h - eMIOS_1_RELOAD_OUT_CH[22]<br>24h - eMIOS_1_RELOAD_OUT_CH[8]<br>25h - eMIOS_1_RELOAD_OUT_CH[0]<br>26h - eMIOS_1_IPP_DO_eMIOS_CH[0]<br>27h - eMIOS_1_IPP_DO_eMIOS_CH[1]<br>28h - eMIOS_1_IPP_DO_eMIOS_CH[2]<br>29h - eMIOS_1_IPP_DO_eMIOS_CH[3]<br>2Ah - eMIOS_1_IPP_DO_eMIOS_CH[4]<br>2Bh - eMIOS_1_IPP_DO_eMIOS_CH[5]<br>2Ch - eMIOS_1_IPP_DO_eMIOS_CH[6]<br>2Dh - eMIOS_1_IPP_DO_eMIOS_CH[7]<br>2Eh - eMIOS_1_IPP_DO_eMIOS_CH[8]<br>2Fh - eMIOS_1_IPP_DO_eMIOS_CH[9]<br>30h - eMIOS_1_IPP_DO_eMIOS_CH[10]<br>31h - eMIOS_1_IPP_DO_eMIOS_CH[11]<br>32h - eMIOS_1_IPP_DO_eMIOS_CH[12]<br>33h - eMIOS_1_IPP_DO_eMIOS_CH[13]<br>34h - eMIOS_1_IPP_DO_eMIOS_CH[14]<br>35h - eMIOS_1_IPP_DO_eMIOS_CH[15]<br>36h - eMIOS_1_IPP_DO_eMIOS_CH[22]<br>37h - eMIOS_1_IPP_DO_eMIOS_CH[23]<br>38h - FlexIO_External Output Trigger 0<br>39h - FlexIO_External Output Trigger 1<br>3Ah - FlexIO_External Output Trigger 2<br>3Bh - FlexIO_External Output Trigger 3<br>3Ch - SIUL_TRGMUX_IN0<br>3Dh - SIUL_TRGMUX_IN1<br>3Eh - SIUL_TRGMUX_IN2<br>3Fh - SIUL_TRGMUX_IN3 |

*Table continues on the next page...*

Table 353. Select bit fields

| Field | Description |
|---|---|
| | 40h - SIUL_TRGMUX_IN4<br>41h - SIUL_TRGMUX_IN5<br>42h - SIUL_TRGMUX_IN6<br>43h - SIUL_TRGMUX_IN7<br>44h - SIUL_TRGMUX_IN8<br>45h - SIUL_TRGMUX_IN9<br>46h - SIUL_TRGMUX_IN10<br>47h - SIUL_TRGMUX_IN11<br>48h - SIUL_TRGMUX_IN12<br>49h - SIUL_TRGMUX_IN13<br>4Ah - SIUL_TRGMUX_IN14<br>4Bh - SIUL_TRGMUX_IN15<br>4Ch - LPI2C_0_Master trigger output<br>4Dh - LPI2C_0_Slave trigger output<br>4Eh - LPSPI_0_End of frame trigger<br>4Fh - LPSPI_0_Receive data trigger<br>50h - LPSPI_1_End of frame trigger<br>51h - LPSPI_1_Receive data trigger<br>52h - LPSPI_2_End of frame trigger<br>53h - LPSPI_2_Receive data trigger<br>54h - LPUART_0_trg_txword<br>55h - LPUART_0_trg_rxword<br>56h - LPUART_0_trg_rxidle<br>57h - LPUART_1_trg_txword<br>58h - LPUART_1_trg_rxword<br>59h - LPUART_1_trg_rxidle<br>5Ah - LPUART_2_trg_txword<br>5Bh - LPUART_2_trg_rxword<br>5Ch - LPUART_2_trg_rxidle<br>5Dh - LCU_0_LC0_out_i0<br>5Eh - LCU_0_LC0_out_i1<br>5Fh - LCU_0_LC0_out_i2<br>60h - LCU_0_LC0_out_i3<br>61h - LCU_0_LC1_out_i0 |

*Table continues on the next page...*

Table 353. Select bit fields

| Field | Description |
|---|---|
| | 62h - LCU_0_LC1_out_i1<br>63h - LCU_0_LC1_out_i2<br>64h - LCU_0_LC1_out_i3<br>65h - LCU_0_LC2_out_i0<br>66h - LCU_0_LC2_out_i1<br>67h - LCU_0_LC2_out_i2<br>68h - LCU_0_LC2_out_i3<br>69h - LCU_1_LC0_out_i0<br>6Ah - LCU_1_LC0_out_i1<br>6Bh - LCU_1_LC0_out_i2<br>6Ch - LCU_1_LC0_out_i3<br>6Dh - LCU_1_LC1_out_i0<br>6Eh - LCU_1_LC1_out_i1<br>6Fh - LCU_1_LC1_out_i2<br>70h - LCU_1_LC1_out_i3<br>71h - LCU_1_LC2_out_i0<br>72h - LCU_1_LC2_out_i1<br>73h - LCU_1_LC2_out_i2<br>74h - LCU_1_LC2_out_i3<br>75h - PIT0_PIT0 CH0<br>76h - PIT0_PIT0 CH1<br>77h - PIT0_PIT0 CH2<br>78h - PIT0_PIT0 CH3<br>79h - PIT0_PIT0 CH4 RTI<br>7Ah - PIT1_PIT1 CH0<br>7Bh - PIT1_PIT1 CH1<br>7Ch - PIT1_PIT1 CH2<br>7Dh - PIT1_PIT1 CH3<br>7Eh - CM7_0_TXEV<br>7Fh - CM7_1_TXEV |

TRGMUX base address: 4008_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | TRGMUX ADC12_0 (ADC12_0) | 32 | RW | 0000_0000h |
| 4h | TRGMUX ADC12_1 (ADC12_1) | 32 | RW | 0000_0000h |
| 8h | TRGMUX ADC12_2 (ADC12_2) | 32 | RW | 0000_0000h |
| Ch | TRGMUX LPCMP_0 (LPCMP_0) | 32 | RW | 0000_0000h |
| 10h | TRGMUX LPCMP_1 (LPCMP_1) | 32 | RW | 0000_0000h |
| 14h | TRGMUX LPCMP_2 (LPCMP_2) | 32 | RW | 0000_0000h |
| 18h | TRGMUX BCTU (BCTU) | 32 | RW | 0000_0000h |
| 1Ch | TRGMUX eMIOS012_ODIS (eMIOS012_ODIS) | 32 | RW | 0000_0000h |
| 20h | TRGMUX eMIOS0_0 (eMIOS0_0) | 32 | RW | 0000_0000h |
| 24h | TRGMUX eMIOS0_1 (eMIOS0_1) | 32 | RW | 0000_0000h |
| 28h | TRGMUX eMIOS0_2 (eMIOS0_2) | 32 | RW | 0000_0000h |
| 2Ch | TRGMUX eMIOS0_3 (eMIOS0_3) | 32 | RW | 0000_0000h |
| 30h | TRGMUX eMIOS1_0 (eMIOS1_0) | 32 | RW | 0000_0000h |
| 34h | TRGMUX eMIOS1_1 (eMIOS1_1) | 32 | RW | 0000_0000h |
| 38h | TRGMUX eMIOS1_2 (eMIOS1_2) | 32 | RW | 0000_0000h |
| 3Ch | TRGMUX eMIOS1_3 (eMIOS1_3) | 32 | RW | 0000_0000h |
| 40h | TRGMUX FlexIO (FlexIO) | 32 | RW | 0000_0000h |
| 44h | TRGMUX SIUL_OUT0 (SIUL_OUT0) | 32 | RW | 0000_0000h |
| 48h | TRGMUX SIUL_OUT1 (SIUL_OUT1) | 32 | RW | 0000_0000h |
| 4Ch | TRGMUX SIUL_OUT2 (SIUL_OUT2) | 32 | RW | 0000_0000h |
| 50h | TRGMUX SIUL_OUT3 (SIUL_OUT3) | 32 | RW | 0000_0000h |
| 54h | TRGMUX LPI2C_0 (LPI2C_0) | 32 | RW | 0000_0000h |
| 58h | TRGMUX LPSPI_0 (LPSPI_0) | 32 | RW | 0000_0000h |
| 5Ch | TRGMUX LPSPI_1 (LPSPI_1) | 32 | RW | 0000_0000h |
| 60h | TRGMUX LPSPI_2 (LPSPI_2) | 32 | RW | 0000_0000h |
| 64h | TRGMUX LPUART_0 (LPUART_0) | 32 | RW | 0000_0000h |
| 68h | TRGMUX LPUART_1 (LPUART_1) | 32 | RW | 0000_0000h |
| 6Ch | TRGMUX LPUART_2 (LPUART_2) | 32 | RW | 0000_0000h |
| 70h | TRGMUX LPUART_3 (LPUART_3) | 32 | RW | 0000_0000h |
| 74h | TRGMUX LCU0_SYNC (LCU0_SYNC) | 32 | RW | 0000_0000h |
| 78h | TRGMUX LCU0_FORCE (LCU0_FORCE) | 32 | RW | 0000_0000h |
| 7Ch | TRGMUX LCU0_0 (LCU0_0) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 80h | TRGMUX LCU0_1 (LCU0_1) | 32 | RW | 0000_0000h |
| 84h | TRGMUX LCU0_2 (LCU0_2) | 32 | RW | 0000_0000h |
| 88h | TRGMUX LCU1_SYNC (LCU1_SYNC) | 32 | RW | 0000_0000h |
| 8Ch | TRGMUX LCU1_FORCE (LCU1_FORCE) | 32 | RW | 0000_0000h |
| 90h | TRGMUX LCU1_0 (LCU1_0) | 32 | RW | 0000_0000h |
| 94h | TRGMUX LCU1_1 (LCU1_1) | 32 | RW | 0000_0000h |
| 98h | TRGMUX LCU1_2 (LCU1_2) | 32 | RW | 0000_0000h |
| 9Ch | TRGMUX CM7_RXEV (CM7_RXEV) | 32 | RW | 0000_0000h |

## 62.6.2 TRGMUX ADC12_0 (ADC12_0)

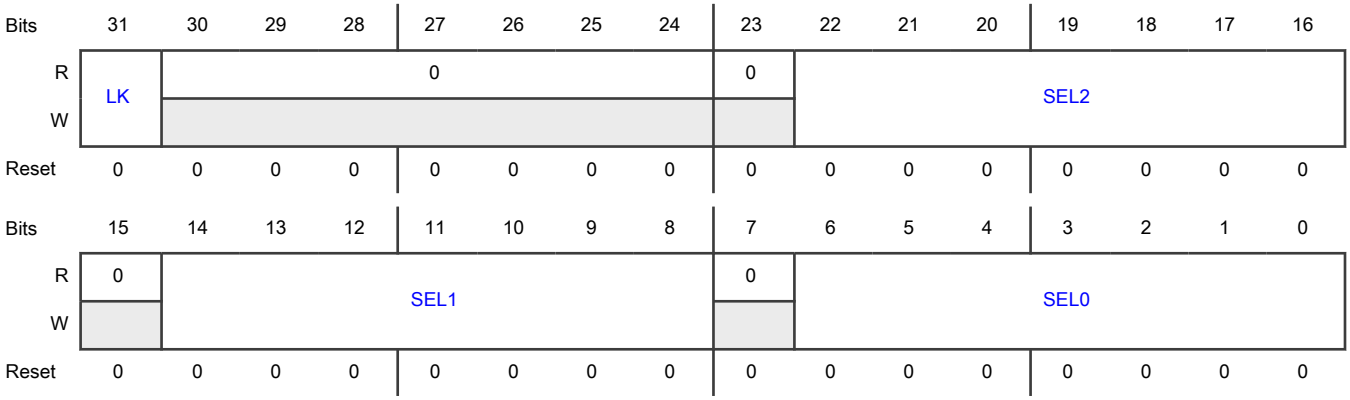
### Offset

| Register | Offset |
|---|---|
| ADC12_0 | 0h |

### Function

Configures the ADC12_0 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31 | TRGMUX Register Lock |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| LK | Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

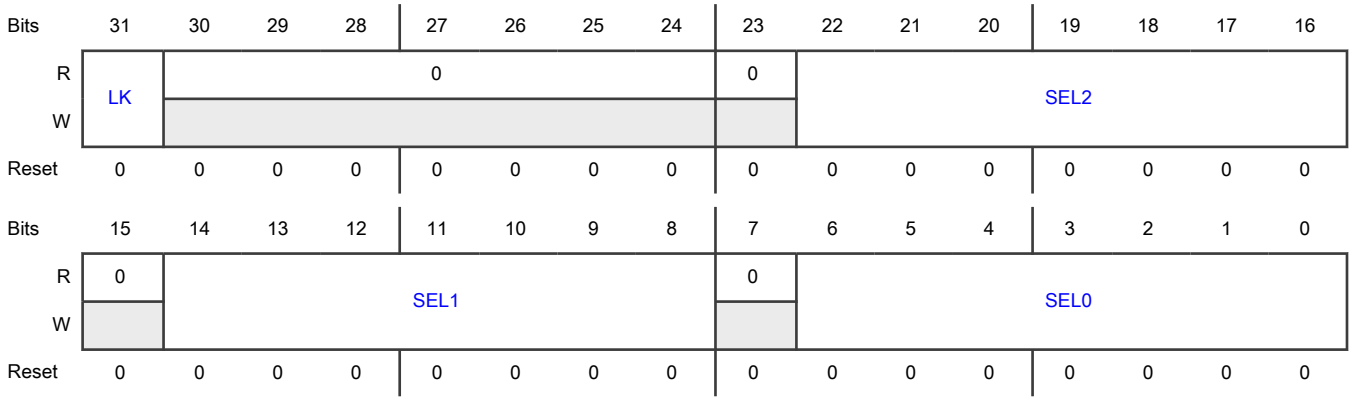
### 62.6.3 TRGMUX ADC12_1 (ADC12_1)

**Offset**

| Register | Offset |
|---|---|
| ADC12_1 | 4h |

**Function**
Configures the ADC12_1 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

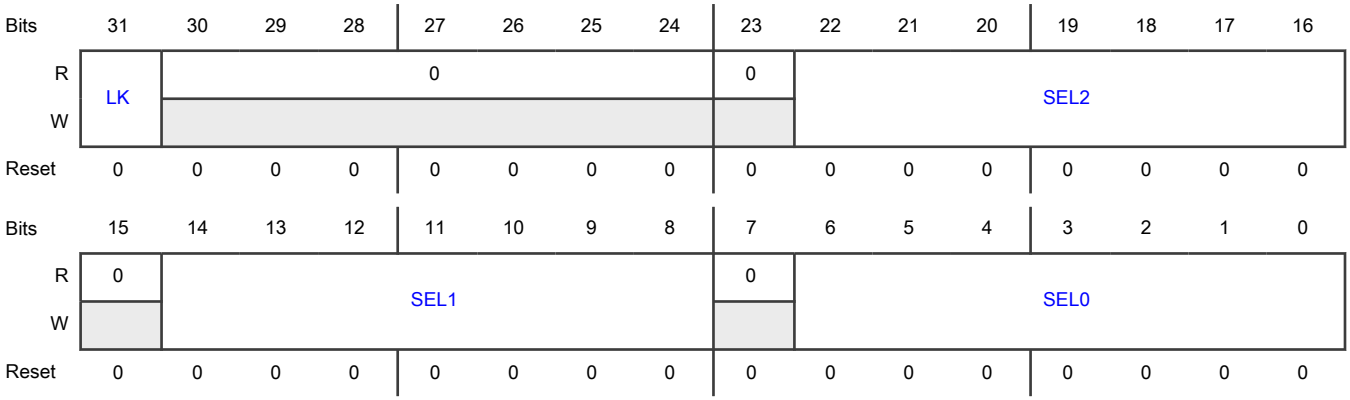
## 62.6.4 TRGMUX ADC12_2 (ADC12_2)

### Offset

| Register | Offset |
|---|---|
| ADC12_2 | 8h |

### Function
Configures the ADC12_2 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.5 TRGMUX LPCMP_0 (LPCMP_0)

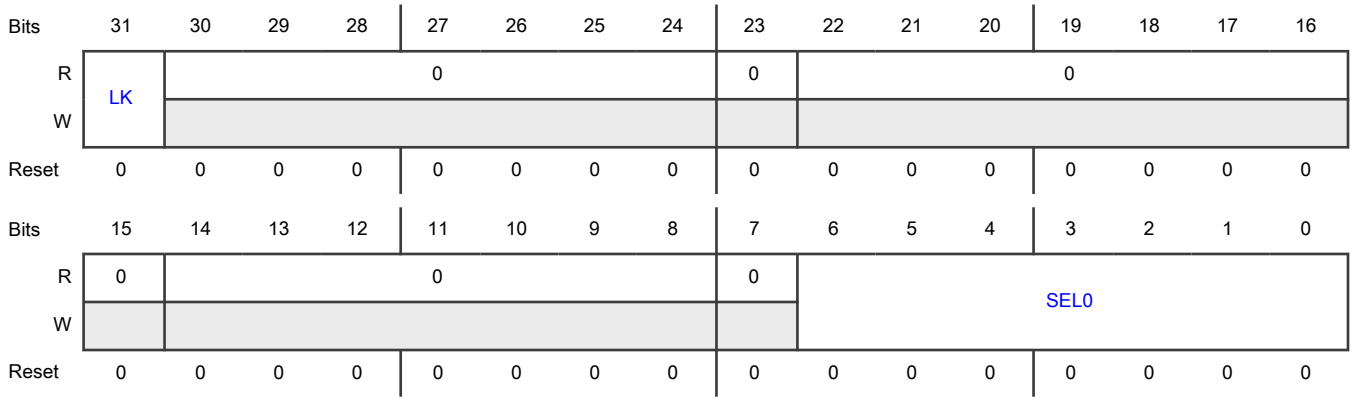
### Offset

| Register | Offset |
|---|---|
| LPCMP_0 | Ch |

### Function

Configures the LPCMP_0 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>— | Reserved |
| 15<br>— | Reserved |
| 14-8<br>— | Reserved |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.6 TRGMUX LPCMP_1 (LPCMP_1)

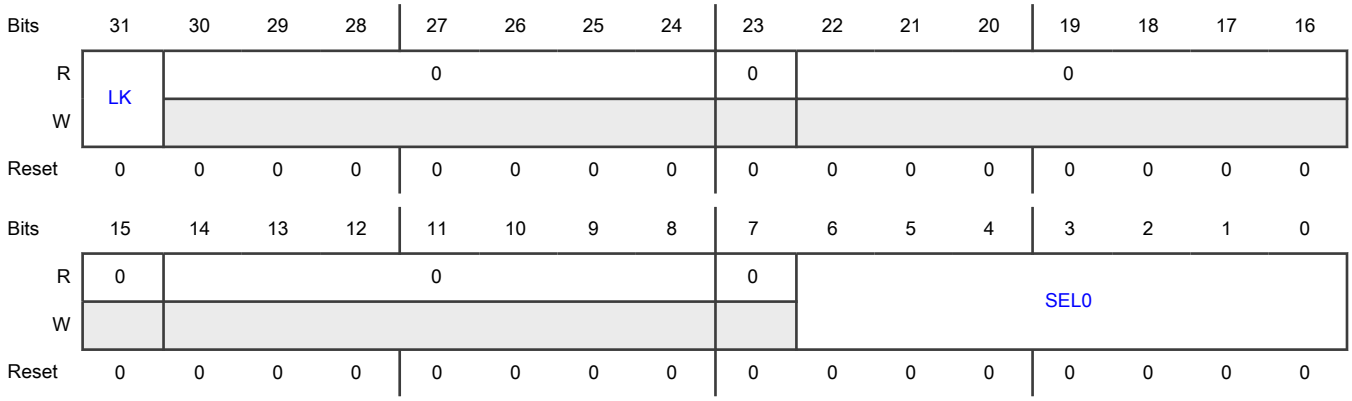
### Offset

| Register | Offset |
|---|---|
| LPCMP_1 | 10h |

### Function

Configures the LPCMP_1 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>— | Reserved |
| 15<br>— | Reserved |
| 14-8<br>— | Reserved |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

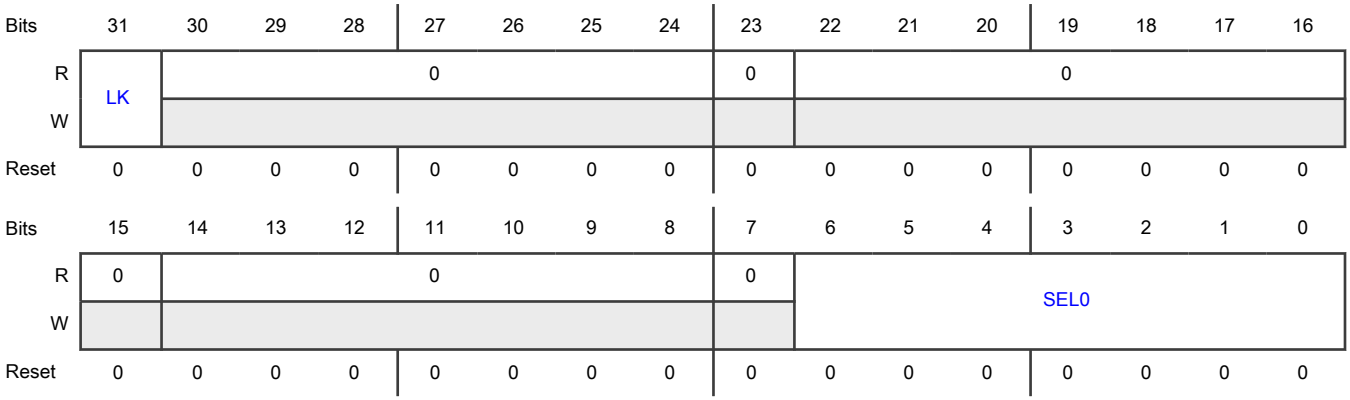
## 62.6.7 TRGMUX LPCMP_2 (LPCMP_2)

### Offset

| Register | Offset |
|---|---|
| LPCMP_2 | 14h |

### Function
Configures the LPCMP_2 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>— | Reserved |
| 15<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 14-8<br>— | Reserved |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.8 TRGMUX BCTU (BCTU)

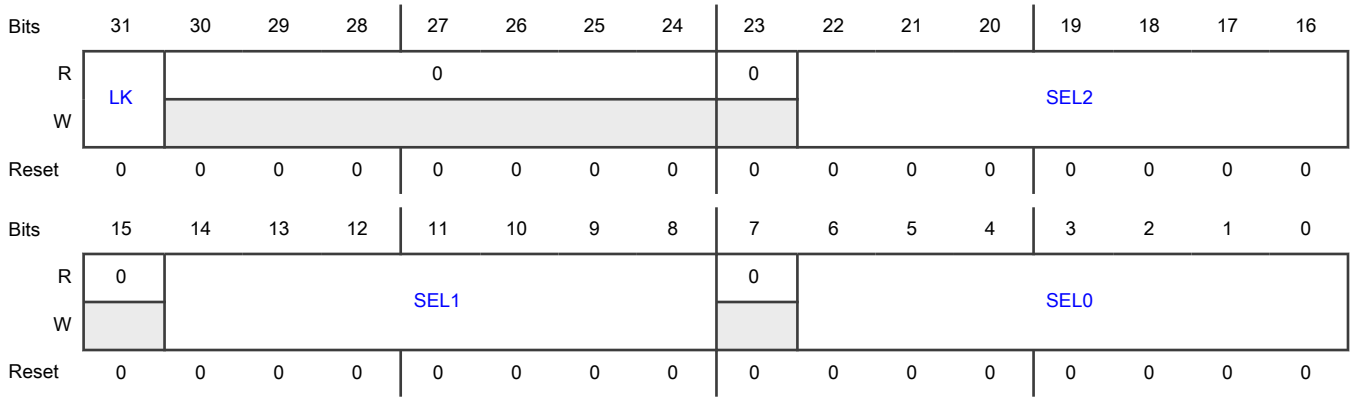
### Offset

| Register | Offset |
|---|---|
| BCTU | 18h |

### Function

Configures the BCTU module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

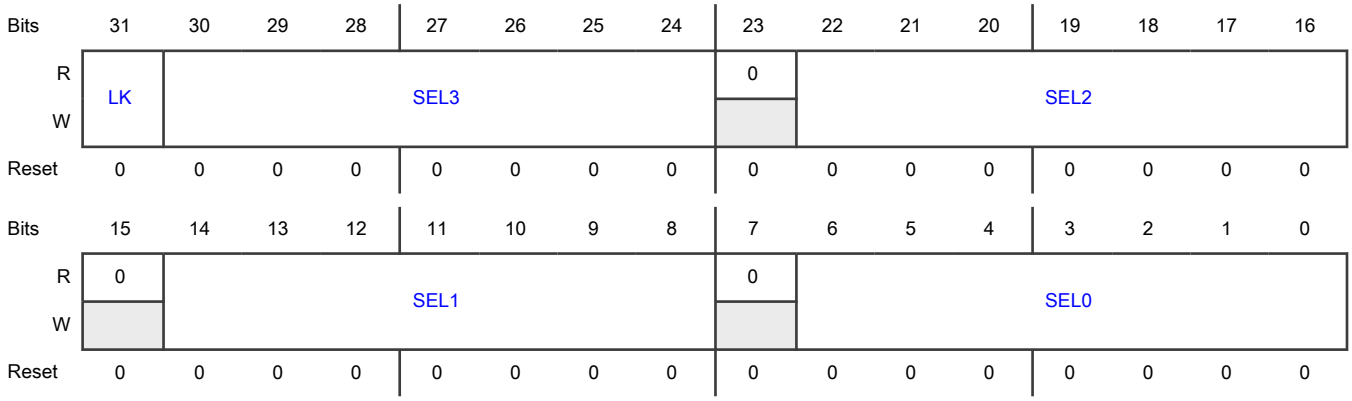
### 62.6.9 TRGMUX eMIOS012_ODIS (eMIOS012_ODIS)

**Offset**

| Register | Offset |
|---|---|
| eMIOS012_ODIS | 1Ch |

**Function**
Configures the eMIOS012_ODIS module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>SEL3 | TRGMUX Source Select 3<br>Specifies the source select for output 3. See Table 353 for field values. |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

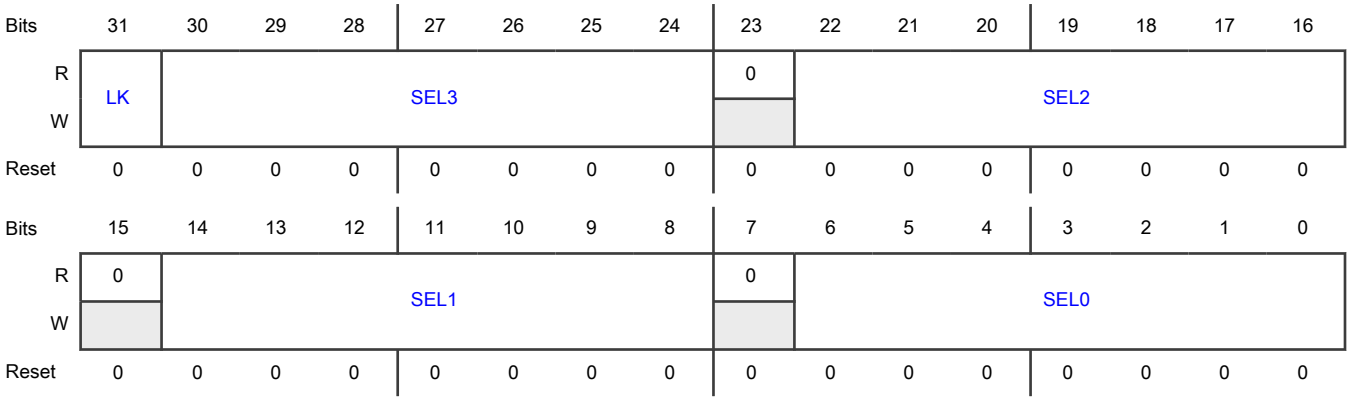
## 62.6.10 TRGMUX eMIOS0_0 (eMIOS0_0)

### Offset

| Register | Offset |
|---|---|
| eMIOS0_0 | 20h |

### Function
Configures the eMIOS0_0 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>SEL3 | TRGMUX Source Select 3<br>Specifies the source select for output 3. See Table 353 for field values. |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.11 TRGMUX eMIOS0_1 (eMIOS0_1)

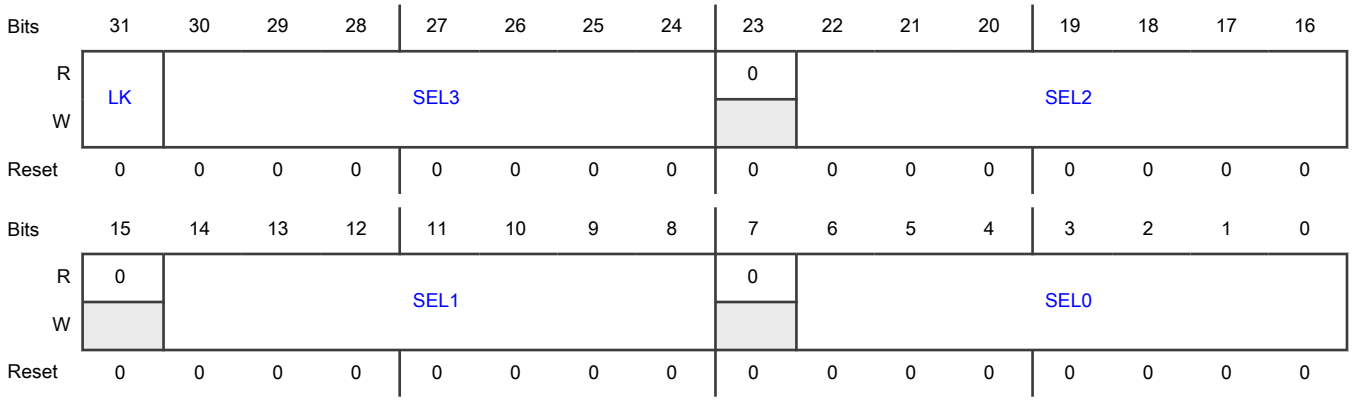
### Offset

| Register | Offset |
|---|---|
| eMIOS0_1 | 24h |

### Function

Configures the eMIOS0_1 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Register is not writable until the next system reset |
| 30-24<br>SEL3 | TRGMUX Source Select 3<br>Specifies the source select for output 3. See Table 353 for field values. |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.12 TRGMUX eMIOS0_2 (eMIOS0_2)

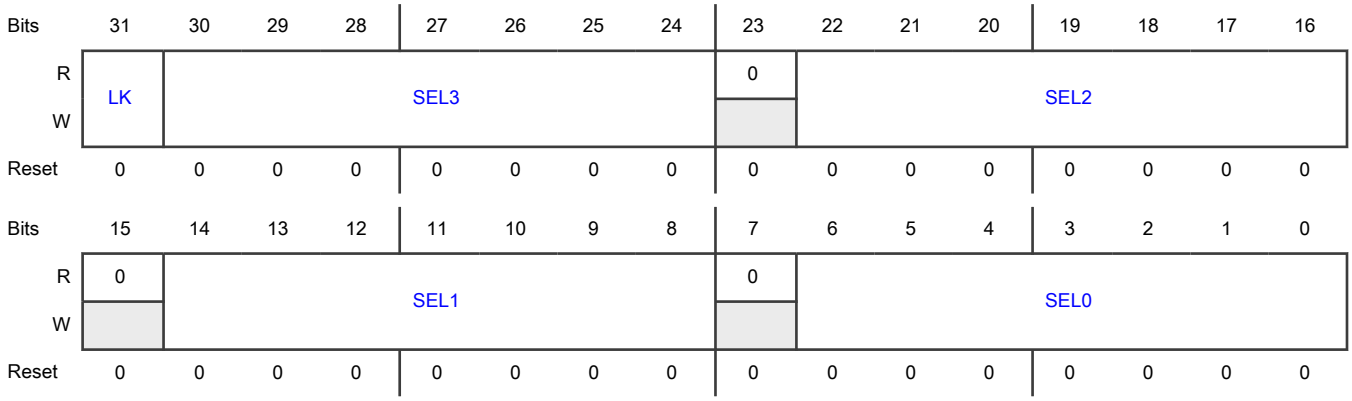
### Offset

| Register | Offset |
|---|---|
| eMIOS0_2 | 28h |

### Function

Configures the eMIOS0_2 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>SEL3 | TRGMUX Source Select 3<br>Specifies the source select for output 3. See Table 353 for field values. |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

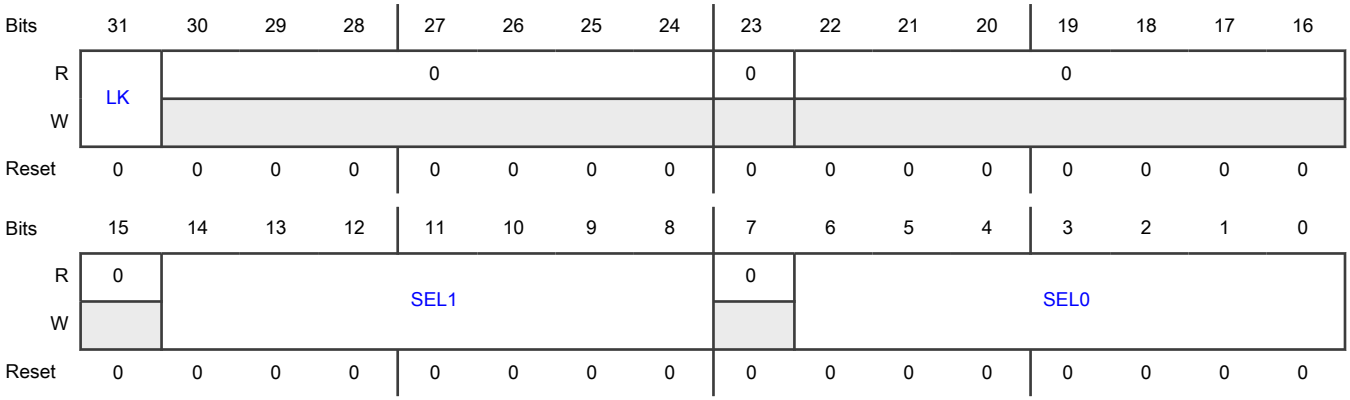
## 62.6.13 TRGMUX eMIOS0_3 (eMIOS0_3)

### Offset

| Register | Offset |
|---|---|
| eMIOS0_3 | 2Ch |

### Function
Configures the eMIOS0_3 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>— | Reserved |
| 15<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.14 TRGMUX eMIOS1_0 (eMIOS1_0)

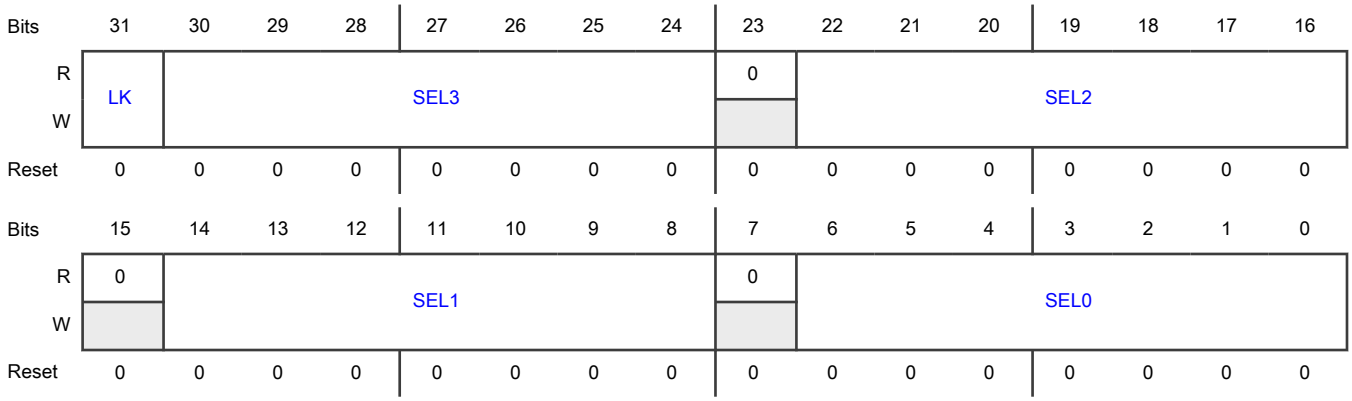
### Offset

| Register | Offset |
|---|---|
| eMIOS1_0 | 30h |

### Function

Configures the eMIOS1_0 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Register is not writable until the next system reset |
| 30-24<br>SEL3 | TRGMUX Source Select 3<br>Specifies the source select for output 3. See Table 353 for field values. |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.15 TRGMUX eMIOS1_1 (eMIOS1_1)

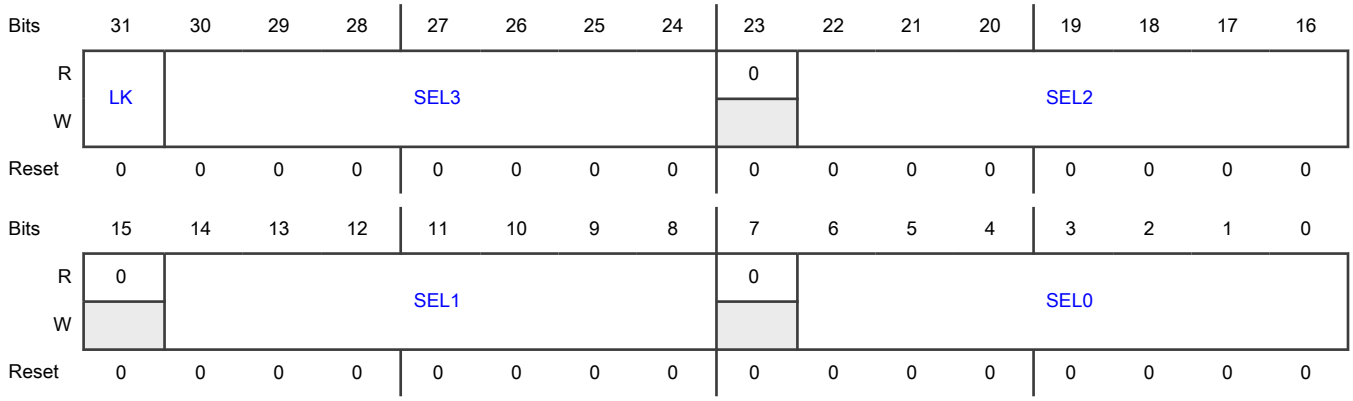
### Offset

| Register | Offset |
|---|---|
| eMIOS1_1 | 34h |

### Function

Configures the eMIOS1_1 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>SEL3 | TRGMUX Source Select 3<br>Specifies the source select for output 3. See Table 353 for field values. |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

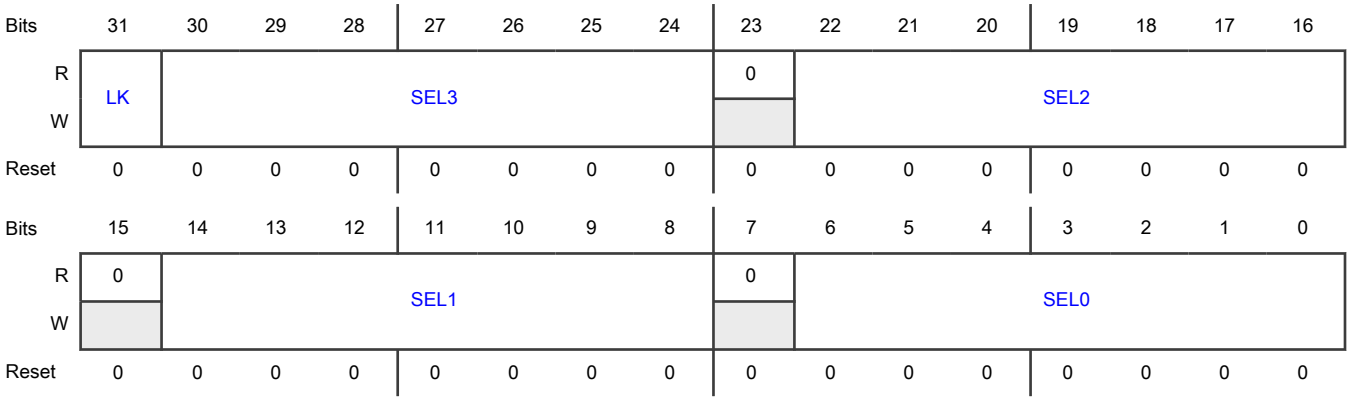
## 62.6.16 TRGMUX eMIOS1_2 (eMIOS1_2)

### Offset

| Register | Offset |
|---|---|
| eMIOS1_2 | 38h |

### Function
Configures the eMIOS1_2 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>SEL3 | TRGMUX Source Select 3<br>Specifies the source select for output 3. See Table 353 for field values. |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

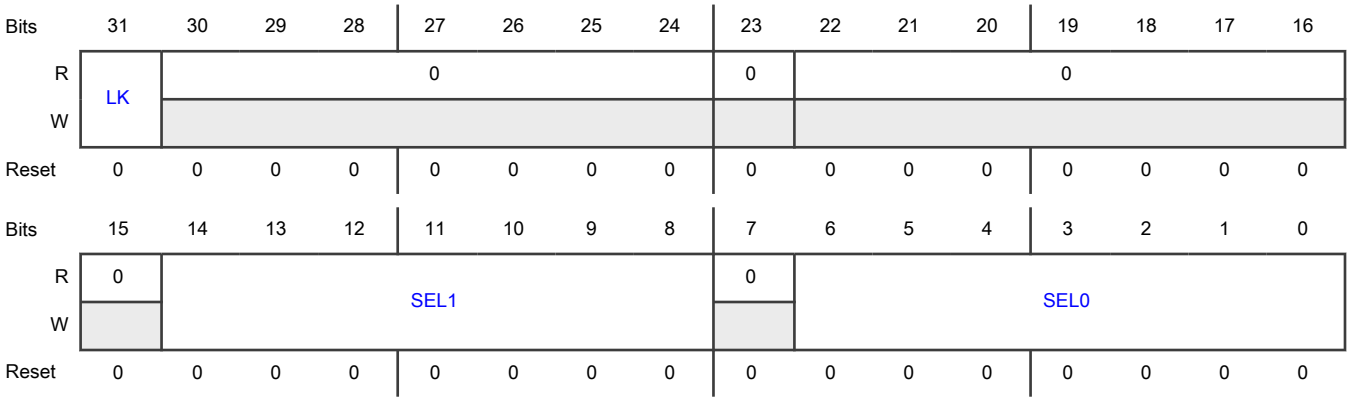
## 62.6.17 TRGMUX eMIOS1_3 (eMIOS1_3)

### Offset

| Register | Offset |
|---|---|
| eMIOS1_3 | 3Ch |

### Function
Configures the eMIOS1_3 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>— | Reserved |
| 15<br>— | Reserved |
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.18 TRGMUX FlexIO (FlexIO)

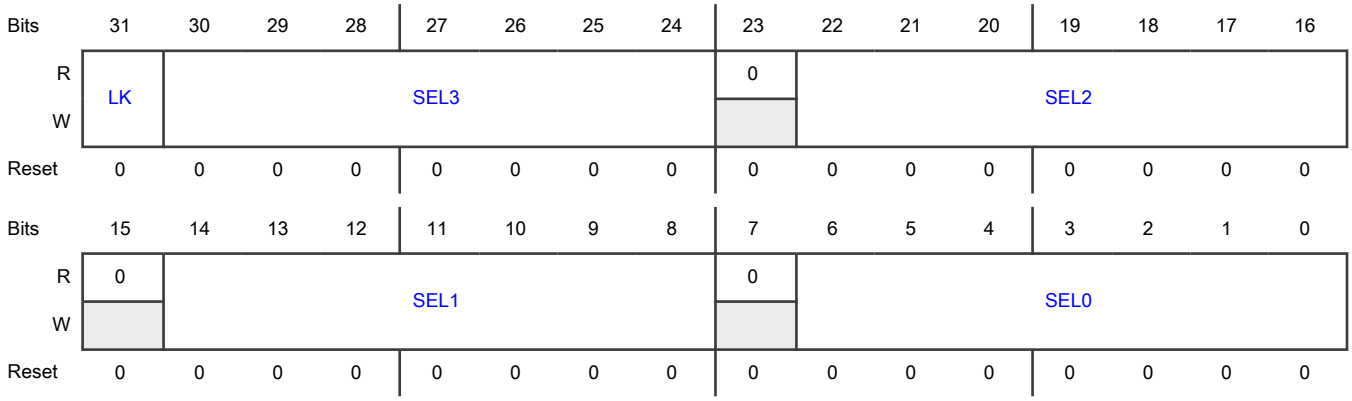
### Offset

| Register | Offset |
|---|---|
| FlexIO | 40h |

### Function

Configures the FlexIO module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>SEL3 | TRGMUX Source Select 3<br>Specifies the source select for output 3. See Table 353 for field values. |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.19 TRGMUX SIUL_OUT0 (SIUL_OUT0)

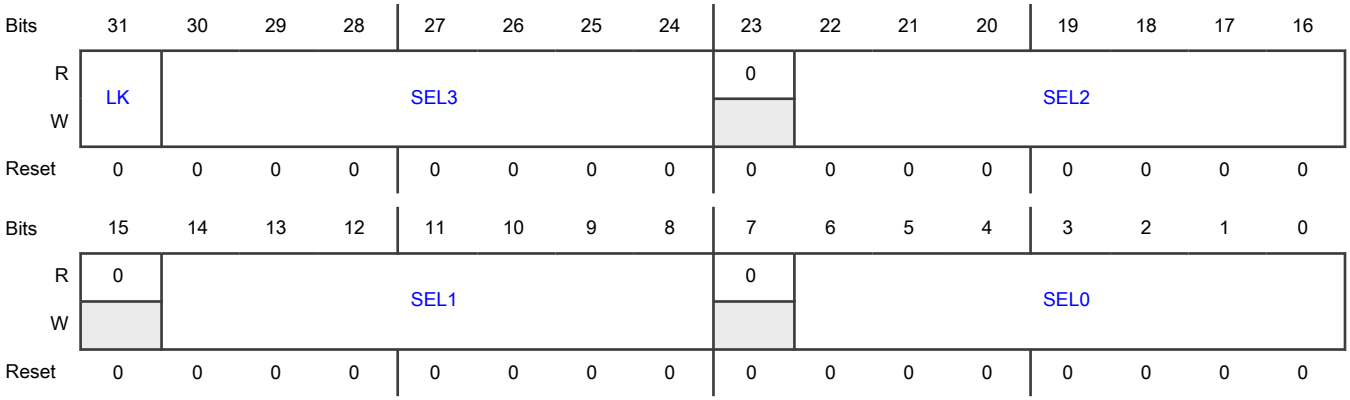
### Offset

| Register | Offset |
|---|---|
| SIUL_OUT0 | 44h |

### Function

Configures the SIUL_OUT0 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>SEL3 | TRGMUX Source Select 3<br>Specifies the source select for output 3. See Table 353 for field values. |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

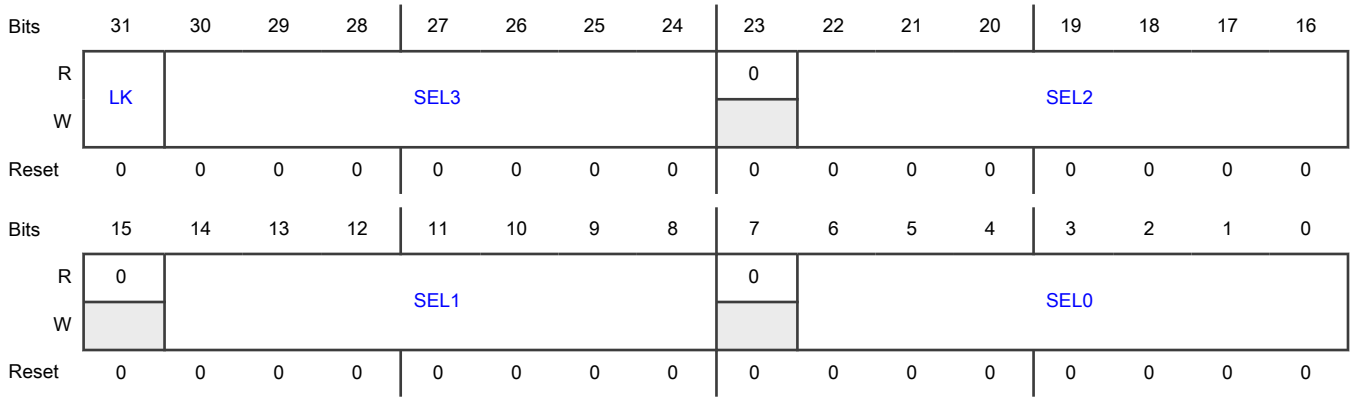
## 62.6.20 TRGMUX SIUL_OUT1 (SIUL_OUT1)

### Offset

| Register | Offset |
|---|---|
| SIUL_OUT1 | 48h |

### Function
Configures the SIUL_OUT1 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Register is not writable until the next system reset |
| 30-24<br>SEL3 | TRGMUX Source Select 3<br>Specifies the source select for output 3. See Table 353 for field values. |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

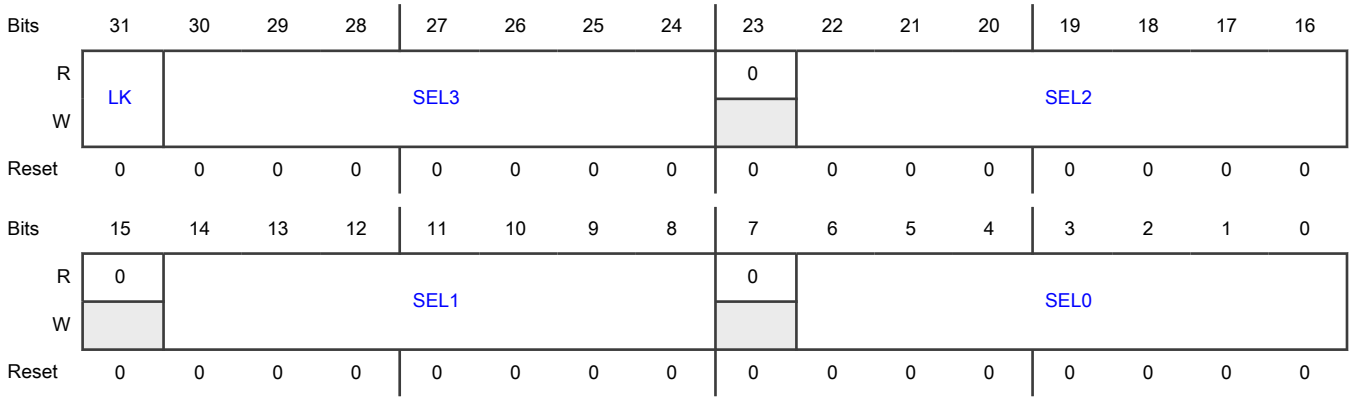
## 62.6.21 TRGMUX SIUL_OUT2 (SIUL_OUT2)

**Offset**

| Register | Offset |
|---|---|
| SIUL_OUT2 | 4Ch |

**Function**
Configures the SIUL_OUT2 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>SEL3 | TRGMUX Source Select 3<br>Specifies the source select for output 3. See Table 353 for field values. |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.22 TRGMUX SIUL_OUT3 (SIUL_OUT3)

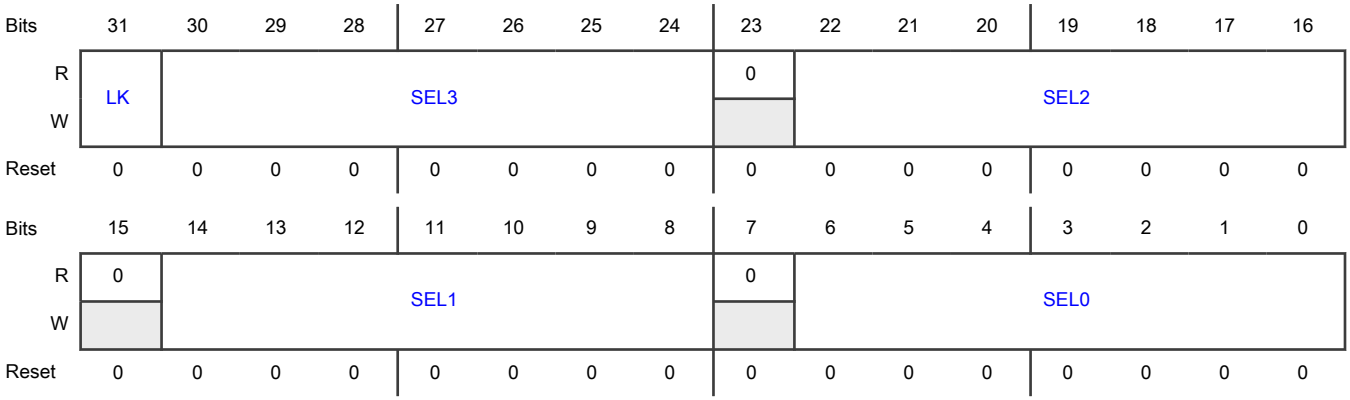
### Offset

| Register | Offset |
|---|---|
| SIUL_OUT3 | 50h |

### Function

Configures the SIUL_OUT3 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>SEL3 | TRGMUX Source Select 3<br>Specifies the source select for output 3. See Table 353 for field values. |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

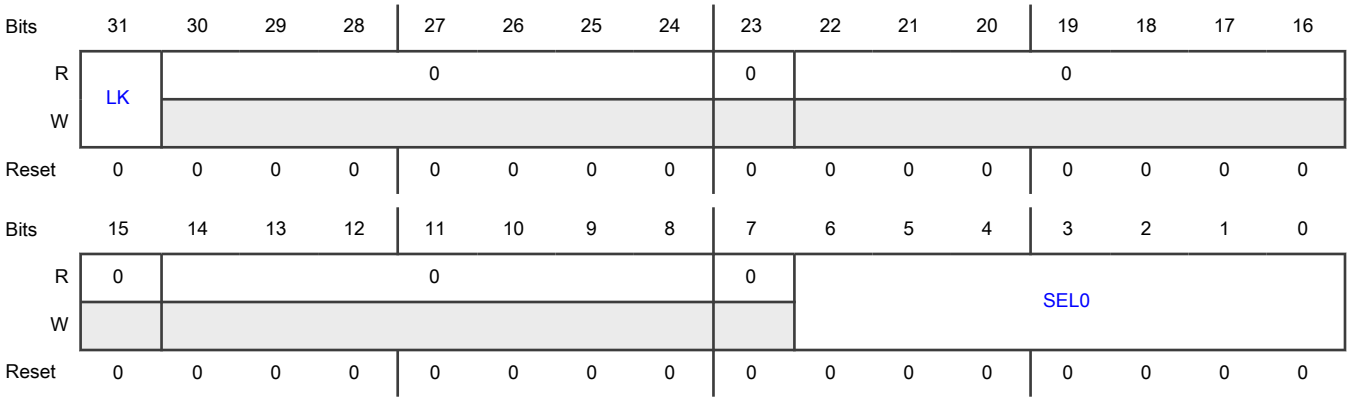
## 62.6.23 TRGMUX LPI2C_0 (LPI2C_0)

### Offset

| Register | Offset |
|---|---|
| LPI2C_0 | 54h |

### Function
Configures the LPI2C_0 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>— | Reserved |
| 15<br>— | Reserved |
| 14-8<br>— | Reserved |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.24 TRGMUX LPSPI_0 (LPSPI_0)

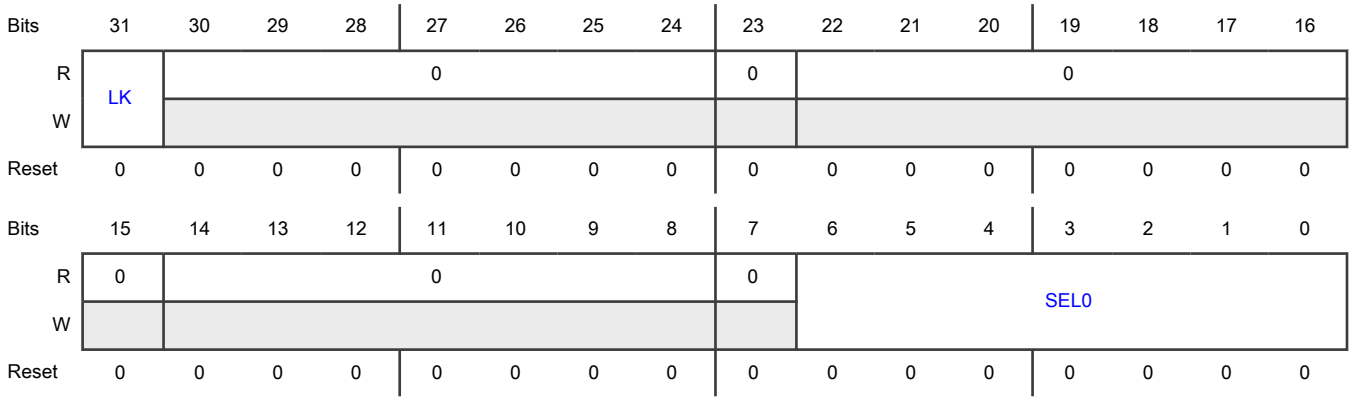
### Offset

| Register | Offset |
|---|---|
| LPSPI_0 | 58h |

### Function

Configures the LPSPI_0 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>— | Reserved |
| 15<br>— | Reserved |
| 14-8<br>— | Reserved |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

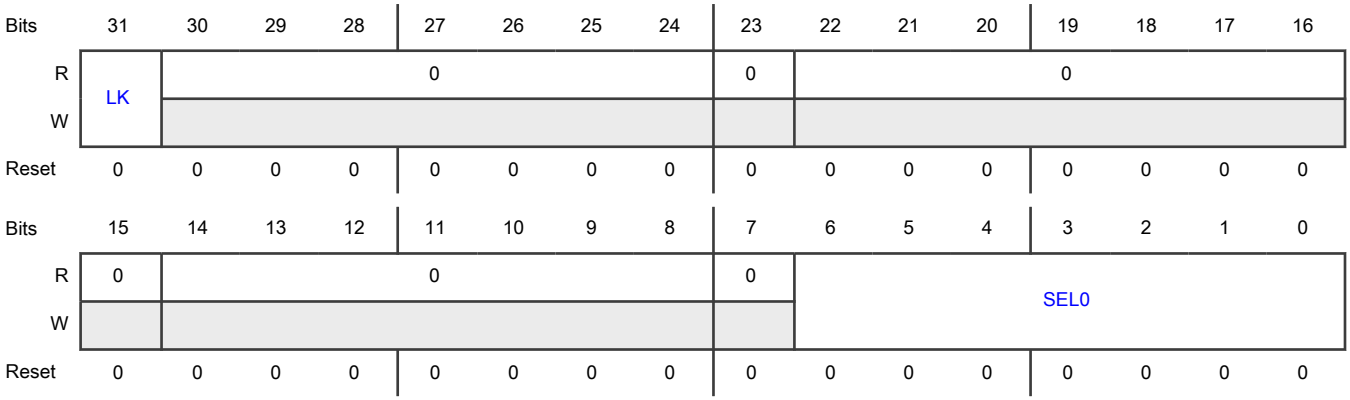
## 62.6.25 TRGMUX LPSPI_1 (LPSPI_1)

### Offset

| Register | Offset |
|---|---|
| LPSPI_1 | 5Ch |

### Function
Configures the LPSPI_1 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>— | Reserved |
| 15<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 14-8<br>— | Reserved |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.26 TRGMUX LPSPI_2 (LPSPI_2)

### Offset

| Register | Offset |
|---|---|
| LPSPI_2 | 60h |

### Function
Configures the LPSPI_2 module.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | SEL0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>— | Reserved |
| 15<br>— | Reserved |
| 14-8<br>— | Reserved |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.27 TRGMUX LPUART_0 (LPUART_0)

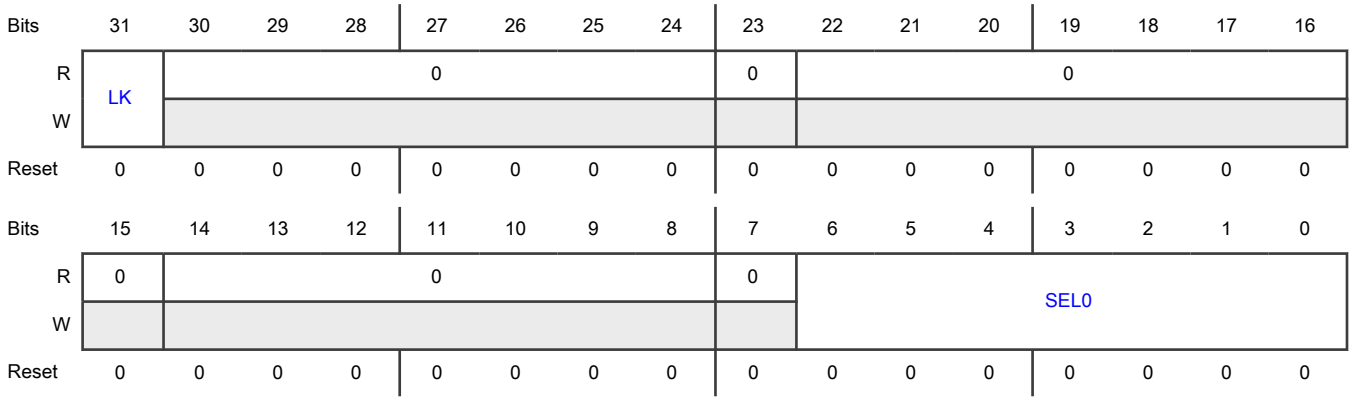
### Offset

| Register | Offset |
|---|---|
| LPUART_0 | 64h |

### Function

Configures the LPUART_0 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>— | Reserved |
| 15<br>— | Reserved |
| 14-8<br>— | Reserved |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.28 TRGMUX LPUART_1 (LPUART_1)

### Offset

| Register | Offset |
|---|---|
| LPUART_1 | 68h |

### Function
Configures the LPUART_1 module.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | | | | | | | 0 | SEL0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>— | Reserved |
| 15<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 14-8<br>— | Reserved |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.29 TRGMUX LPUART_2 (LPUART_2)

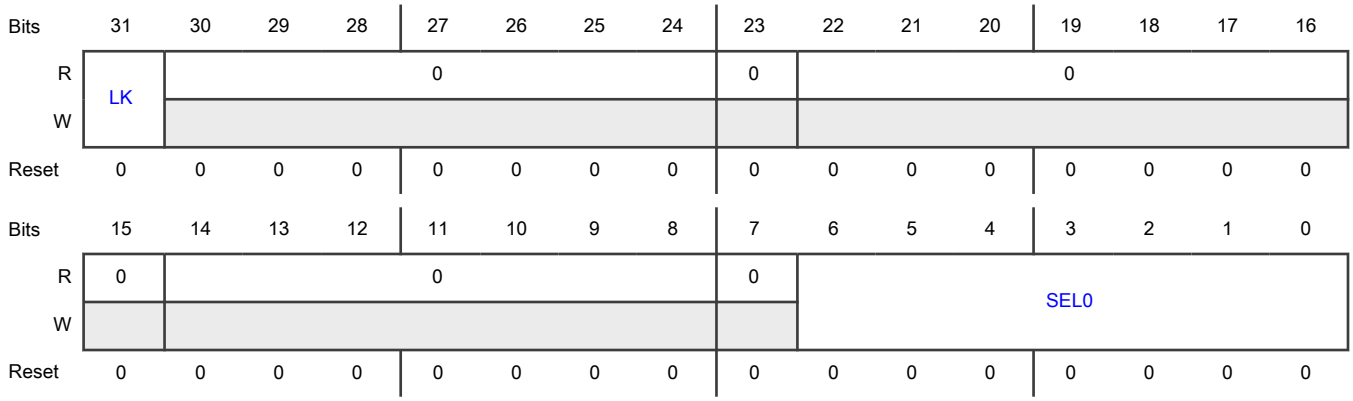
### Offset

| Register | Offset |
|---|---|
| LPUART_2 | 6Ch |

### Function

Configures the LPUART_2 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>— | Reserved |
| 15<br>— | Reserved |
| 14-8<br>— | Reserved |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.30 TRGMUX LPUART_3 (LPUART_3)

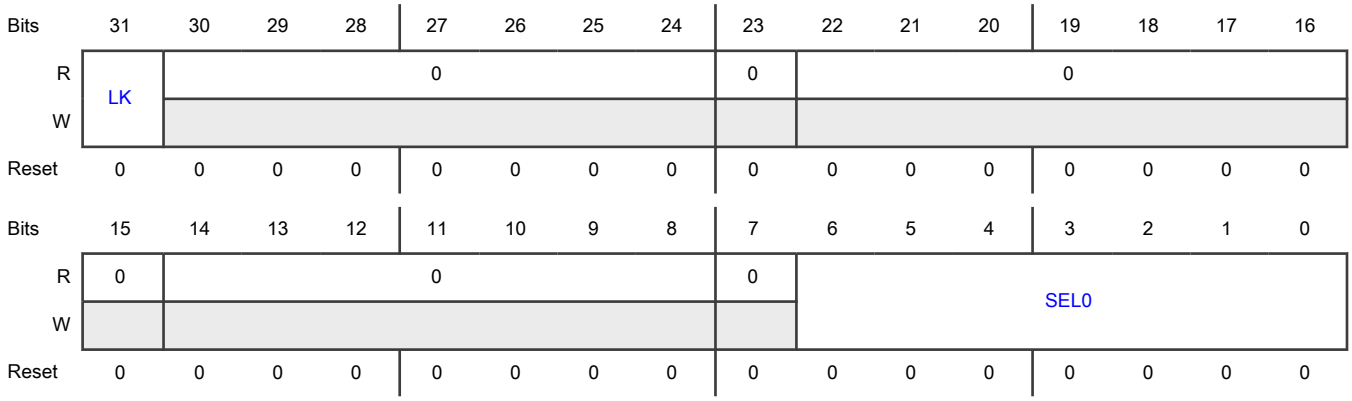
### Offset

| Register | Offset |
|---|---|
| LPUART_3 | 70h |

### Function

Configures the LPUART_3 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>— | Reserved |
| 15<br>— | Reserved |
| 14-8<br>— | Reserved |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

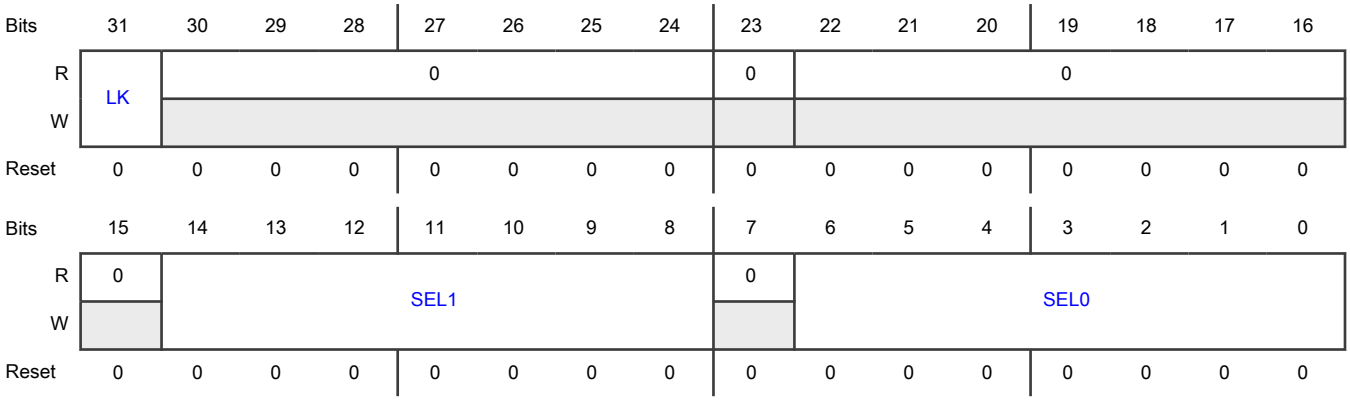
## 62.6.31 TRGMUX LCU0_SYNC (LCU0_SYNC)

### Offset

| Register | Offset |
|---|---|
| LCU0_SYNC | 74h |

### Function
Configures the LCU0_SYNC module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>— | Reserved |
| 15<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.32 TRGMUX LCU0_FORCE (LCU0_FORCE)

### Offset

| Register | Offset |
|---|---|
| LCU0_FORCE | 78h |

### Function
Configures the LCU0_FORCE module.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LK | 0 | | | | | | | 0 | SEL2 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | SEL1 | | | | | | | 0 | SEL0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.33 TRGMUX LCU0_0 (LCU0_0)

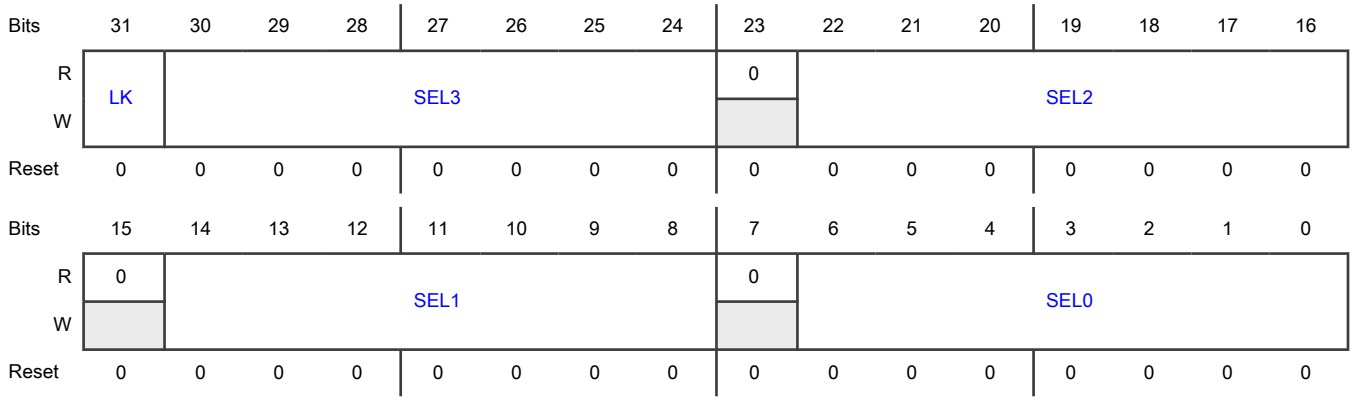
### Offset

| Register | Offset |
|---|---|
| LCU0_0 | 7Ch |

### Function

Configures the LCU0_0 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>SEL3 | TRGMUX Source Select 3<br>Specifies the source select for output 3. See Table 353 for field values. |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

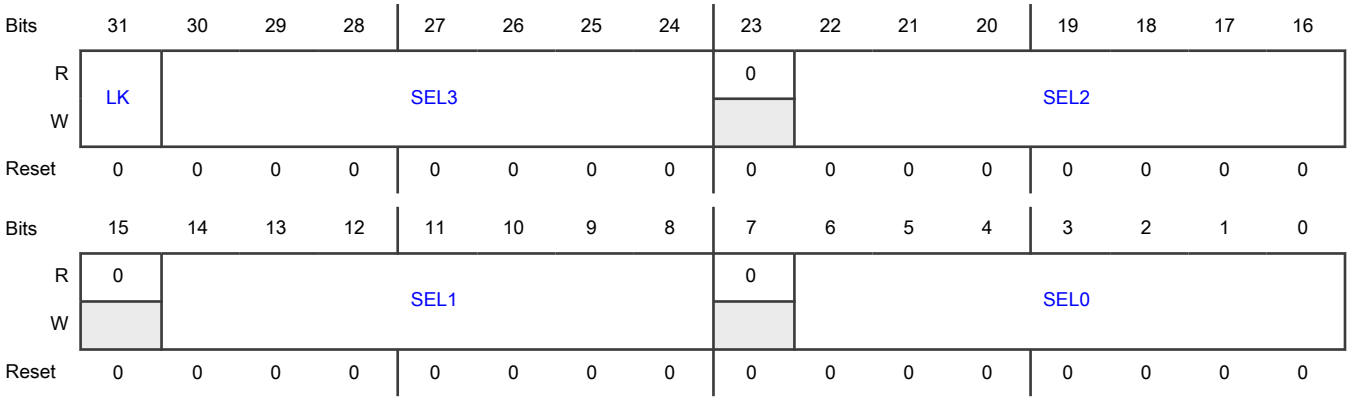
## 62.6.34 TRGMUX LCU0_1 (LCU0_1)

### Offset

| Register | Offset |
|---|---|
| LCU0_1 | 80h |

### Function
Configures the LCU0_1 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>SEL3 | TRGMUX Source Select 3<br>Specifies the source select for output 3. See Table 353 for field values. |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

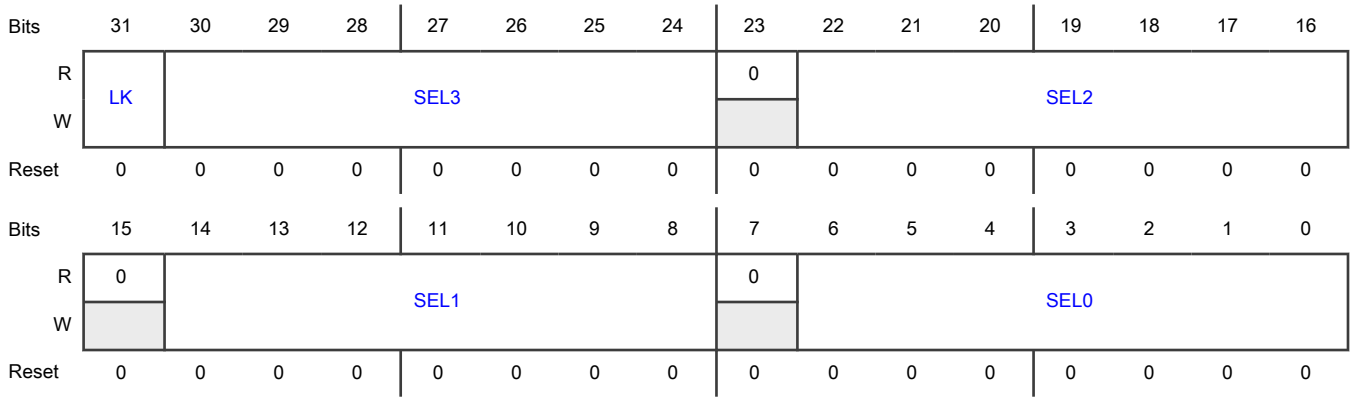
## 62.6.35 TRGMUX LCU0_2 (LCU0_2)

### Offset

| Register | Offset |
|---|---|
| LCU0_2 | 84h |

### Function
Configures the LCU0_2 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Register is not writable until the next system reset |
| 30-24<br>SEL3 | TRGMUX Source Select 3<br>Specifies the source select for output 3. See Table 353 for field values. |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.36 TRGMUX LCU1_SYNC (LCU1_SYNC)

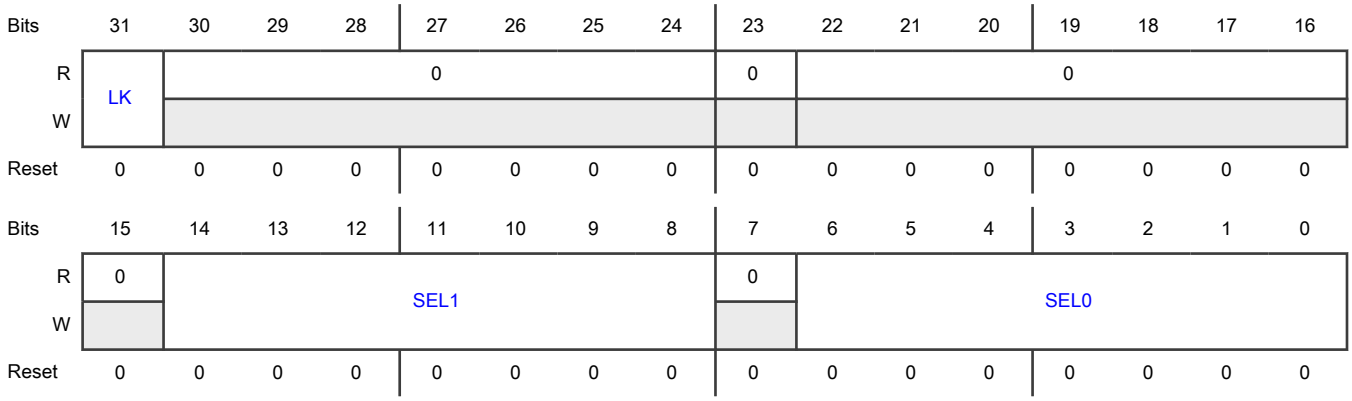
### Offset

| Register | Offset |
|---|---|
| LCU1_SYNC | 88h |

### Function

Configures the LCU1_SYNC module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>— | Reserved |
| 15<br>— | Reserved |
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.37 TRGMUX LCU1_FORCE (LCU1_FORCE)

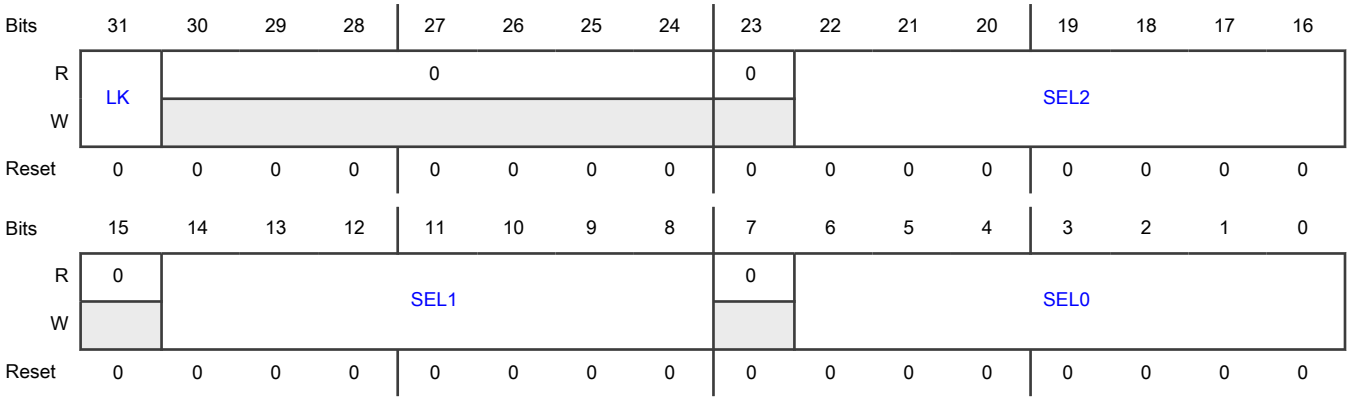
### Offset

| Register | Offset |
|---|---|
| LCU1_FORCE | 8Ch |

### Function

Configures the LCU1_FORCE module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

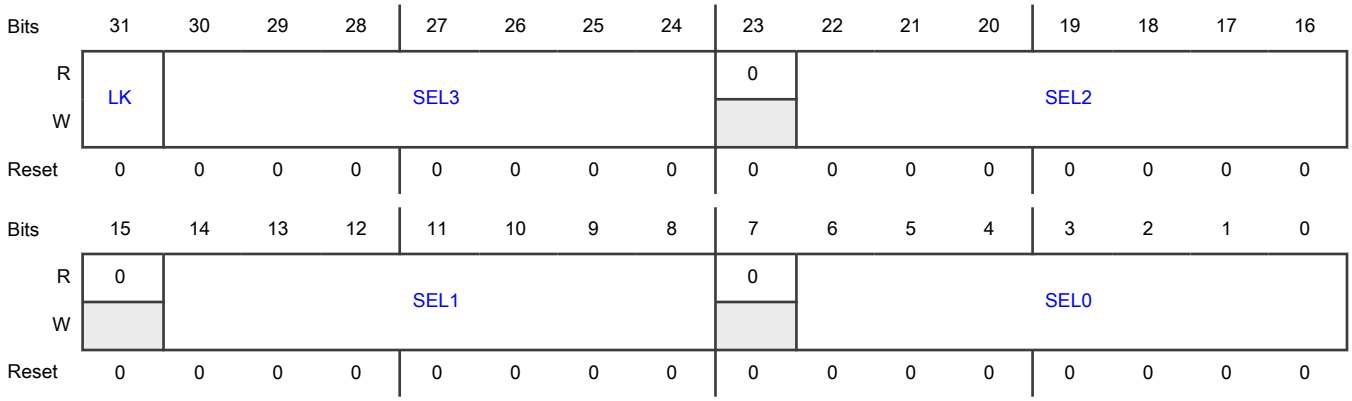
## 62.6.38 TRGMUX LCU1_0 (LCU1_0)

### Offset

| Register | Offset |
|---|---|
| LCU1_0 | 90h |

### Function
Configures the LCU1_0 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Register is not writable until the next system reset |
| 30-24<br>SEL3 | TRGMUX Source Select 3<br>Specifies the source select for output 3. See Table 353 for field values. |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.39 TRGMUX LCU1_1 (LCU1_1)

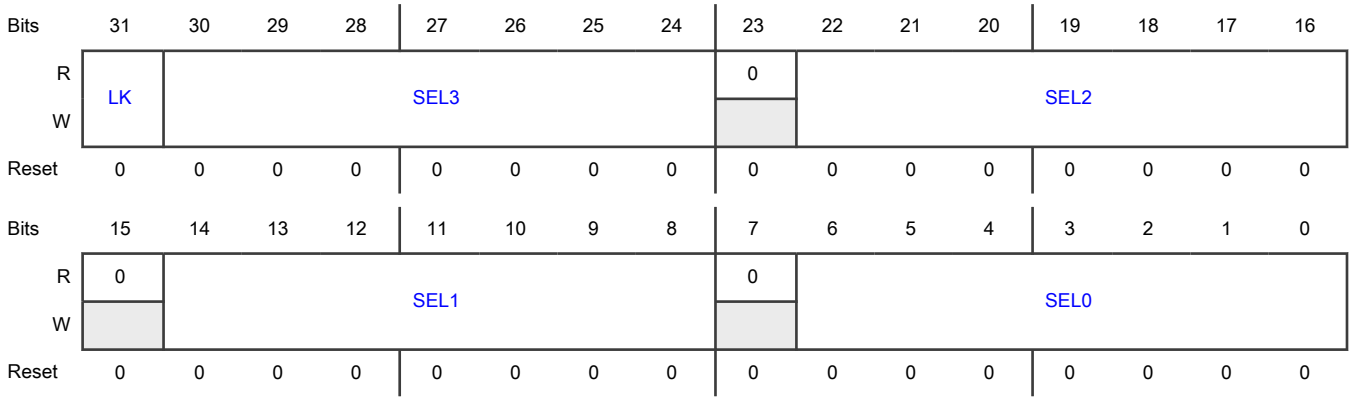
### Offset

| Register | Offset |
|---|---|
| LCU1_1 | 94h |

### Function

Configures the LCU1_1 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>SEL3 | TRGMUX Source Select 3<br>Specifies the source select for output 3. See Table 353 for field values. |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

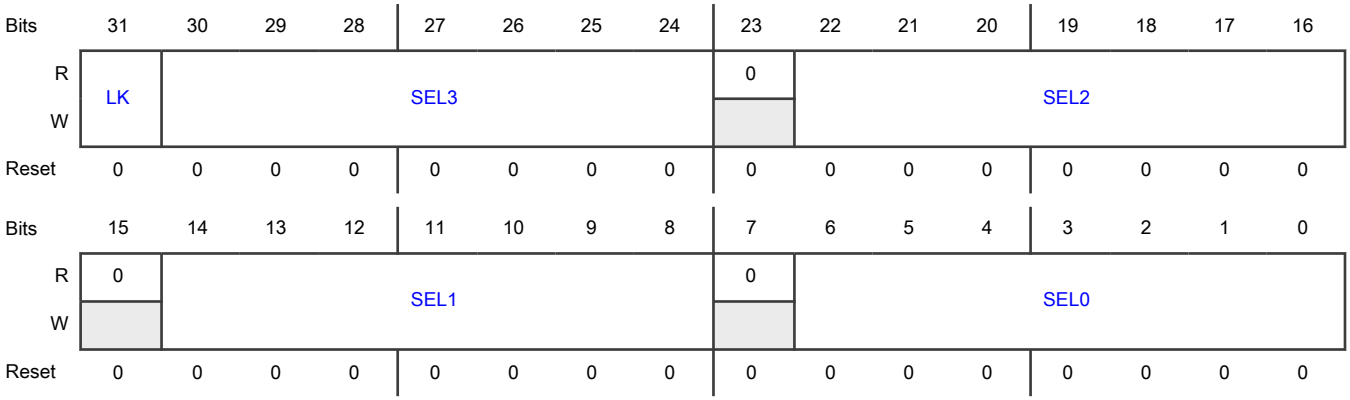
## 62.6.40 TRGMUX LCU1_2 (LCU1_2)

### Offset

| Register | Offset |
|---|---|
| LCU1_2 | 98h |

### Function
Configures the LCU1_2 module.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable<br>1b - Register is not writable until the next system reset |
| 30-24<br>SEL3 | TRGMUX Source Select 3<br>Specifies the source select for output 3. See Table 353 for field values. |
| 23<br>— | Reserved |
| 22-16<br>SEL2 | TRGMUX Source Select 2<br>Specifies the source select for output 2. See Table 353 for field values. |
| 15<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

## 62.6.41 TRGMUX CM7_RXEV (CM7_RXEV)

### Offset

| Register | Offset |
|---|---|
| CM7_RXEV | 9Ch |

### Function
Configures the CM7_RXEV module.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | SEL1 | | | | | | | 0 | SEL0 | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>LK | TRGMUX Register Lock<br>Disables writing to the register. You can write to this field once after system reset. When this field is 1, you cannot write to SEL*x* until the next system reset. This field becomes 0 after system reset.<br>0b - Register is writable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Register is not writable until the next system reset |
| 30-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22-16<br>— | Reserved |
| 15<br>— | Reserved |
| 14-8<br>SEL1 | TRGMUX Source Select 1<br>Specifies the source select for output 1. See Table 353 for field values. |
| 7<br>— | Reserved |
| 6-0<br>SEL0 | TRGMUX Source Select 0<br>Specifies the source select for output 0. See Table 353 for field values. |

# Chapter 63
# Software Watchdog Timer (SWT)

## 63.1 Chip-specific SWT information

### 63.1.1 SWT instances and configuration

This chip supports up to three instances of SWT, one for each Cortex-M7 core.

**Table 354. SWT configuration**

| Instance | Core | Reset type | Clock Source | SWT_MIN_TO (100µs) | SWT_TO_RST (25ms) |
|---|---|---|---|---|---|
| SWT_0[1] | Cortex-M7_0 | Functional Reset - RAM retention | SIRC (32K) | 3 | 320h |

1. SWT_0 supports operation in STANDBY mode.

**NOTE**

When using SWT as a wakeup source from standby, software should ensure proper value of timeout such that SWT timeout doesn't get expired again (after wakeup till standby exit is completed). The timeout should be programmed to a value greater than the time till standby exit is complete (sum of standby exit time and the time till software disables padkeeping).

**NOTE**

The "Reset only the SWT" feature using SWT_RRR[RRF] is supported only when SWT reset reaction is demoted to interrupt within MC_RGM.

#### Protect SWT from service by non-core masters

You must program protection for the SWT registers so that only core masters can service SWT.

When an SWT timeout takes place, the logic generating a reset to the system runs off the slow internal RC oscillator (32KHz). It could therefore take the SWT almost 31.25µs to reset the system, after timeout, leaving more time for runaway code to execute. The register interface clock of SWT is running from AIPS_SLOW_CLK, see Clocking chapter for more details. The register interface clock is responsible for generating the abort for invalid access while the counter clock is responsible for generating the SWT reset request. Since there is a synchronization happening between the register interface clock and counter clock domains before generating the SWT reset request, thus the generation of the reset request can take a number of register interface clock cycles after an invalid access is done.

#### Clock source

SWT is clocked by SIRC only.

## 63.2 Overview

The Software Watchdog Timer (SWT) is a 32-bit window watchdog timer that enables the system to recover from situations such as:

- Software trapped in a loop
- A bus transaction failing to terminate

In regular operation, SWT requires periodic execution of a watchdog servicing operation. The servicing operation resets the timer to a specified timeout period. If this servicing action does not occur before the timer expires, SWT generates an interrupt or a hardware reset request.

You can configure SWT to generate a reset request or an interrupt on the initial timeout. SWT always generates a reset request on a second consecutive timeout.

**NOTE**

Terminology in this chapter has been updated as follows:

**Table 355. Updated terms**

| Updated terms | Deprecated term |
|---|---|
| Initiator | Master |

### 63.2.1 Block diagram

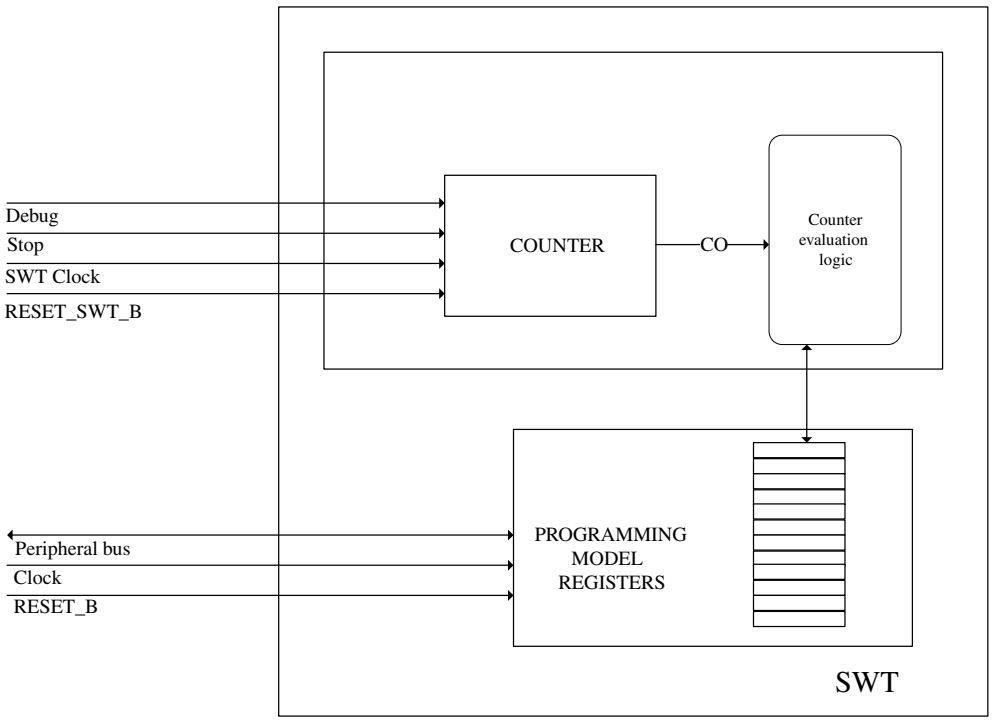



**Figure 350. SWT block diagram**

### 63.2.2 Features

SWT has the following features:

- 32-bit countdown timer
- Selection of regular or window servicing
- Selection of reset request or interrupt on an initial timeout
- Selection of fixed or keyed service sequence
- Initiator access protection
- Hard and soft configuration lock bits

## 63.3 Functional description

### 63.3.1 Behavior in different chip and core modes

SWT supports the core modes of operation as follows:

| Core mode | SWT behavior |
|---|---|
| Normal | When SWT is enabled (CR[WEN] = 1), the SWT timer counts down continuously. |
| Debug | If CR[FRZ] = 1, SWT stops the timer; otherwise, the timer continues to run. |
| Stop or Standby | If CR[STP] = 1, SWT stops the timer; otherwise, the timer continues to run. |

### 63.3.2 Register access latency

Accesses to SWT registers occur with no peripheral bus wait states. The peripheral bus bridge may add 1 or more system wait states. However, due to synchronization logic in the SWT design, recognition of the service sequence or configuration changes may require up to three system clock cycles plus seven counter clock cycles.

### 63.3.3 Service key generation

SWT generates service keys with the pseudorandom key generator defined by the following equation.

$$SK_{n+1} = (17 \times SK_n + 3) \bmod 2^{16}$$

**Figure 351. Pseudorandom key generator**

This algorithm generates a sequence of $2^{16}$ different key values before repeating. In keyed service sequence mode, each time you write a valid key to SR[WSC], SWT updates SK[SK] with the next key.

For example, if the previous service key (SK[SK]) is 100h, then the next service sequence keys are 1103h followed by 2136h.

### 63.3.4 Clocking

The SWT has two clock domains, one for the IPS interface and the other for the counter and reset request logic. The two clocks can be asynchronous to each other and have independent resets.

See the chip-specific SWT information for the clock source of the counter and reset logic.

See the chip-specific SWT information to find the clock source that drives the countdown timer.

### 63.3.5 Reset

The following table describes the SWT resets:

**Table 356. SWT resets**

| Reset | Description |
|---|---|
| RESET_B | It is used to reset the programming model registers. |
| RESET_SWT_B | It is used to reset the functional domain of the SWT. |

### 63.3.6 Interrupts

The SWT generates one interrupt:

- Timeout interrupt: it is asserted when the watchdog timer count exceeds the timeout period in the TO register.

### 63.3.7 Reset only the SWT

When a timeout triggers a reset request and the application does not require a system reset, you can make the SWT instance reset itself rather than the whole system.

To make a timeout reset request trigger only an SWT reset (rather than the whole system), write 1 to the Reset Request Flag (RRR[RRF]).

This action causes SWT to:

- Clear the reset request.
- If an interrupt request also occurred, clear the interrupt request (IR[TIF]).
- Reset and restart the SWT.

For systems with multiple SWT instances, this feature allows you to reset an individual SWT instance, avoiding the widespread impact of a system reset.

### 63.3.8 Test SWT operation

When you disable SWT, it loads the current countdown timer into CO[CNT]. When you enable SWT, it clears this register. You can use CO[CNT] to perform a software self-test of SWT.

To test SWT operation:

1. Enable the SWT (CR[WEN] = 1).
2. Do not service SWT for a fixed period of time that is less than the timeout value.
3. Disable the SWT (CR[WEN] = 0).
4. Read the value in CO[CNT] to determine whether the internal countdown timer is working properly.

**NOTE**

The value shown in CO[CNT] can lag behind the actual internal timer up to six system clock cycles plus eight counter clock cycles.

## 63.4 External signals

This module has no external signals.

## 63.5 Initialization

### 63.5.1 Initialize the SWT

To initialize the SWT, you must do the following:

- Set the timeout period.
- Control timeout behavior. There are two options:
  - Select reset request on timeout, or
  - Select interrupt on initial timeout.

You must initialize all registers before enabling SWT (CR[WEN]). You can initialize the registers in any sequence.

#### 63.5.1.1 Set the timeout period

When you enable SWT, it loads the greater of the specified watchdog timeout period or the minimum timeout period into an internal 32-bit countdown timer every time you perform a valid service operation.

To set the watchdog timeout period:

- Write the desired timeout period to TO[WTO].

The minimum timeout period is 3 clock cycles.

#### 63.5.1.2 Control timeout behavior

SWT can respond to a timeout in one of two ways:

- Generate an immediate reset request on any timeout.
- Generate an interrupt and load the countdown timer on an initial timeout, then generate a reset request on a subsequent timeout.

##### 63.5.1.2.1 Select reset request on timeout

To configure SWT to generate an immediate reset request on any timeout:

- Write 0 to CR[ITR].

##### 63.5.1.2.2 Select interrupt on initial timeout

To configure SWT to generate an interrupt on an initial timeout:

- Write 1 to CR[ITR].

In this configuration, on the initial timeout:

- SWT loads the countdown timer with the timeout period (TO[WTO]).
- SWT indicates the interrupt with the timeout interrupt flag (IR[TIF]).
- If a second consecutive timeout occurs before writing the service sequence for the first timeout, SWT generates a reset request.

You clear the interrupt flag by writing 1 to IR[TIF].

#### 63.5.1.3 Configure locking and unlocking

You can lock the SWT configuration with either a hard lock or a soft lock. When either lock is in effect, CR, TO, WN, and SK are read-only.

##### 63.5.1.3.1 Enable the hard lock

To enable the hard lock:

- Write 1 to CR[HLK].

Hard lock is disabled only by a reset.

##### 63.5.1.3.2 Enable the soft lock

To enable the soft lock:

- Write 1 to CR[SLK].

##### 63.5.1.3.3 Unlock the soft lock

To unlock the soft lock:

1. Write C520h to SR[WSC].
2. Write D928h to SR[WSC].

This unlock sequence:

- Ignores service sequence writes.
- Recognizes the unlock sequence regardless of previous writes.

- Recognizes the unlock sequence regardless of the time between writes.
- Does not require CR[WEN] to be 1.

You can write this unlock sequence at any time.

**NOTE**

It is possible for a keyed service sequence to unlock soft lock. See Avoid soft unlock for the procedure to handle this situation.

#### 63.5.1.3.4 Avoid soft unlock

When SWT operates in keyed sequence service, the sequence of service keys generated by the pseudorandom generator includes the unlock keys for soft lock. If one or more service routines use both unlock keys in the proper order, 0xC520 followed at any future time by 0xD928, SWT unlocks soft lock (CR[SLK] = 0). The unlock sequence logic ignores any service keys other than the unlock keys, so the unlock keys don't have to be inserted consecutively. If the second unlock key is also the second key of a service operation, unlock occurs before the service operation completes.

To avoid unlock:

If the service routine writes a key value of C520h to SR[WSC] as a key of the service operation, then do the following:

1. Complete the service operation.
2. Complete the unlock sequence by writing the second unlock key, D928h, to SR[WSC].
3. Reinitiate the soft lock (CR[SLK] = 1).

### 63.5.1.4 Select behavior on an invalid access

SWT can respond to an invalid access one of two ways:

- Generate a bus error.
- Generate a bus error and, if SWT is enabled, a reset request.

To select how SWT responds to an invalid access:

- Write the appropriate value to CR[RIA].

## 63.5.2 Initiate service operations

When enabled, SWT requires periodic execution of a servicing operation.

SWT can operate in one of the following service sequence modes:

- Fixed service sequence
- Keyed service sequence

### 63.5.2.1 Initiate a fixed service sequence

To initiate the fixed service sequence:

1. Select the fixed servicing mode (write 00b to CR[SMD]).
2. Write A602h to SR[WSC].
3. Write B480h to SR[WSC].

There is no timing requirement between the two writes. The service sequence logic ignores unlock-sequence writes.

### 63.5.2.2 Initiate a keyed service sequence

To initiate the keyed service sequence:

1. Select the keyed servicing mode (write 01b to CR[SMD]).
2. Read the initial service key from SK[SK].
3. Calculate the next service key using the provided equation.
4. Write that service key to SR[WSC].
5. Repeat steps 2 through 4 one time.

See Service key generation for information about service keys.

#### 63.5.2.3 Select window service mode

In window service mode, you must write the service sequence only when the watchdog timer is less than the window start value (WN[WST]). If you write the service sequence outside of this window, the access is invalid and generates a bus error or reset request depending on the Reset on Invalid Access setting (CR[RIA]).

For example, if the timeout period is 5000 and the window start value is 1000, you must write the service sequence during the last 20% of the timeout period.

Synchronization logic in SWT causes a slight delay in the opening of the window. This delay can be up to three system clock cycles plus four counter clock cycles.

To select window service mode:

- Write 1 to (CR[WND]).

## 63.6 SWT register descriptions

The SWT programming model allows only 32-bit (word) accesses.

Any of the following attempted accesses are invalid:

- Non-32-bit accesses
- Writes to read-only registers
- Writes of incorrect values to SR when SWT is enabled
- Accesses to reserved addresses
- Accesses by initiators without permission

You control how SWT responds to an invalid access with CR[RIA].

### 63.6.1 SWT memory map

SWT_0 base address: 4027_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Control (CR) | 32 | RW | FF00_010Ah |
| 4h | Interrupt (IR) | 32 | RW | 0000_0000h |
| 8h | Timeout (TO) | 32 | RW | 0000_0320h |
| Ch | Window (WN) | 32 | RW | 0000_0000h |
| 10h | Service (SR) | 32 | RW | 0000_0000h |
| 14h | Counter Output (CO) | 32 | R | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 18h | Service Key (SK) | 32 | RW | 0000_0000h |
| 1Ch | Event Request (RRR) | 32 | RW | 0000_0000h |

## 63.6.2 Control (CR)

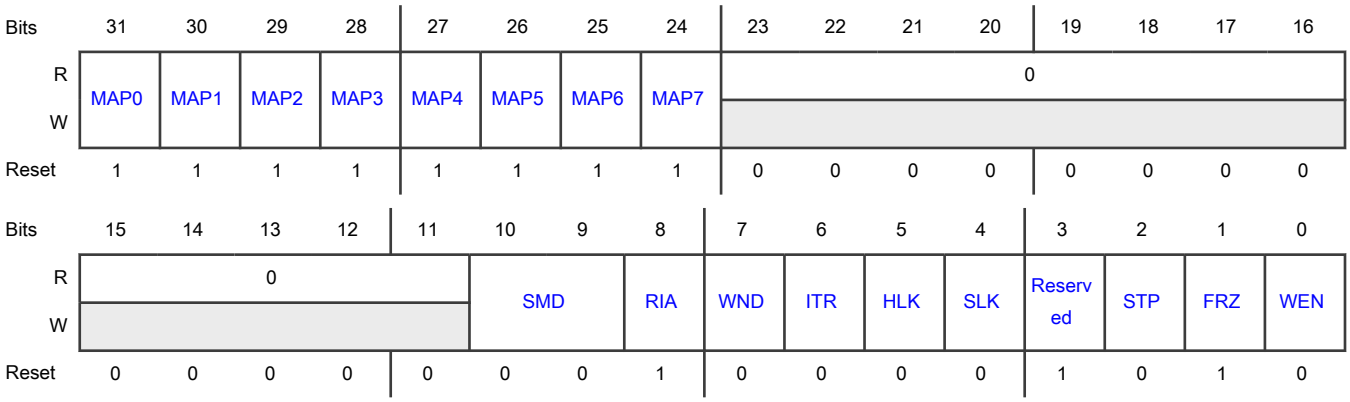
### Offset

| Register | Offset |
|---|---|
| CR | 0h |

### Function

Contains fields for configuring and controlling SWT. The register is read-only if either hard lock or soft lock is enabled (either HLK or SLK is 1).

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>MAP0 | Initiator Access Protection 0<br>The platform bus initiator assignments are chip-specific.<br>The number of this field corresponds to the XRDC DIDs of the bus initiator. MAP0 corresponds to XRDC DIDs 0 and 8.<br>0b - Access disabled<br>1b - Access enabled |
| 30 | Initiator Access Protection 1 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| MAP1 | See the description for MAP0. MAP1 corresponds to XRDC DIDs 1 and 9.<br>0b - Access disabled<br>1b - Access enabled |
| 29<br>MAP2 | Initiator Access Protection 2<br>See the description for MAP0. MAP2 corresponds to XRDC DIDs 2 and 10.<br>0b - Access disabled<br>1b - Access enabled |
| 28<br>MAP3 | Initiator Access Protection 3<br>See the description for MAP0. MAP3 corresponds to XRDC DIDs 3 and 11.<br>0b - Access disabled<br>1b - Access enabled |
| 27<br>MAP4 | Initiator Access Protection 4<br>See the description for MAP0. MAP4 corresponds to XRDC DIDs 4 and 12.<br>0b - Access disabled<br>1b - Access enabled |
| 26<br>MAP5 | Initiator Access Protection 5<br>See the description for MAP0. MAP5 corresponds to XRDC DIDs 5 and 13.<br>0b - Access disabled<br>1b - Access enabled |
| 25<br>MAP6 | Initiator Access Protection 6<br>See the description for MAP0. MAP6 corresponds to XRDC DIDs 6 and 14.<br>0b - Access disabled<br>1b - Access enabled |
| 24<br>MAP7 | Initiator Access Protection 7<br>See the description for MAP0. MAP7 corresponds to XRDC DIDs 7 and 15.<br>0b - Access disabled<br>1b - Access enabled |
| 23-11<br>— | Reserved |
| 10-9<br>SMD | Service Mode |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 00b - Fixed Service Sequence. To service the watchdog, write the fixed sequence 0xA602, 0xB480 to SR.<br>01b - Keyed Service Sequence. To service the watchdog, write two pseudorandom key values to SR.<br>10b - Reserved. Do not use this value. Writing this value can cause the watchdog to not be serviced.<br>11b - Reserved. Do not use this value. Writing this value can cause the watchdog not to be serviced. |
| 8<br>RIA | Reset on Invalid Access<br>Controls how SWT responds to an invalid access.<br>**NOTE**<br>For a description of how this chip implements SWT reset requests resulting from SWT invalid accesses, see the chip-specific SWT information.<br>0b - Generate a bus error<br>1b - Generate a bus error and reset request. If SWT is enabled (WEN = 1), generate a bus error and a reset request; otherwise, generate a bus error. |
| 7<br>WND | Window Mode<br>Specify regular or window mode operation.<br>0b - Regular mode. Can execute service sequence at any time<br>1b - Window mode. Can execute service sequence only when the timeout counter is less than the value in WN |
| 6<br>ITR | Interrupt Then Reset Request<br>**NOTE**<br>For a description of how this chip implements SWT reset requests and interrupt requests resulting from SWT timeouts, see the chip-specific SWT information.<br>0b - Generate a reset request on a timeout<br>1b - Generate an interrupt on an initial timeout; generate a reset request on a second consecutive timeout |
| 5<br>HLK | Hard Lock<br>Indicates that the hard lock is enabled. You cannot directly write 0 to this field. This field becomes 0 only after a reset.<br>0b - CR, TO, WN, and SK are read/write registers if SLK is also 0<br>1b - CR, TO, WN, and SK are read-only registers |
| 4<br>SLK | Soft Lock |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Indicates that the soft lock is enabled. You cannot directly write 0 to this field. Clear this field by writing the unlock sequence to the service register.<br>0b - CR, TO, WN, and SK are read/write registers if HLK is also 0<br>1b - CR, TO, WN, and SK are read-only registers |
| 3<br>— | Reserved |
| 2<br>STP | Stop Mode Control<br>Controls the watchdog timer when the chip enters Stop or Standby mode.<br>0b - Timer continues<br>1b - Timer stops |
| 1<br>FRZ | Debug Mode Control<br>Controls the watchdog timer when the core enters Debug mode.<br>0b - Timer continues<br>1b - Timer stops |
| 0<br>WEN | Watchdog Enable<br>Enables or disables SWT.<br>The reset value is 0. Therefore, after reset, you must enable SWT to start the countdown timer.<br>0b - Disabled<br>1b - Enabled |

### 63.6.3 Interrupt (IR)

#### Offset

| Register | Offset |
|---|---|
| IR | 4h |

#### Function

The timeout interrupt flag.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | TIF |
| W | | | | | | | | | | | | | | | | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>TIF | Timeout Interrupt Flag<br>Write 1 to this field to clear the flag and interrupt. Writing a 0 has no effect.<br>0b - No interrupt request<br>1b - Interrupt request due to an initial timeout |

## 63.6.4 Timeout (TO)

### Offset

| Register | Offset |
|---|---|
| TO | 8h |

### Function

Contains the 32-bit timeout period. The register is read-only if either hard lock or soft lock is enabled (CR[HLK] or CR[SLK] is 1).

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WTO | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WTO | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>WTO | Watchdog Timeout<br>Watchdog timeout period in clock cycles. When software writes a service sequence or enables SWT, SWT loads the internal 32-bit countdown timer with this value or 3h, whichever is greater. |

## 63.6.5 Window (WN)

### Offset

| Register | Offset |
|---|---|
| WN | Ch |

### Function

Contains the 32-bit window start value. SWT clears this register on reset. The register is read-only if either hard lock or soft lock is enabled (CR[HLK] or CR[SLK] is 1).

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WST | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | WST | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>WST | Window Start Value<br>When you enable window mode (CR[WND]), you can write the service sequence only when the internal timer is less than this value. |

## 63.6.6 Service (SR)

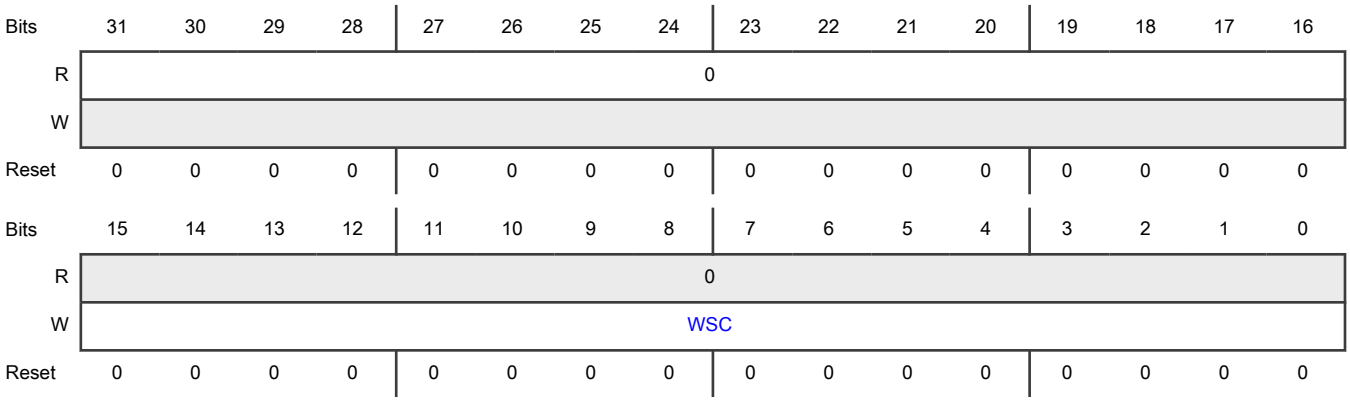
### Offset

| Register | Offset |
|---|---|
| SR | 10h |

### Function

Initiates the service operation and resets the watchdog timer.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>WSC | Watchdog Service Code<br>Use this field to service the watchdog and to unlock the Soft Lock (CR[SLK]).<br>To service the watchdog: If SWT is in keyed service mode (CR[SMD]), write two pseudorandom key values to WSC (see Service key generation for details). Otherwise, write the following values to WSC, in the order shown: |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1. A602h<br>2. B480h<br>To unlock the Soft Lock (CR[SLK]), write the following values to WSC, in the order shown:<br>1. C520h<br>2. D928h<br>When read, WSC always returns zero. |

## 63.6.7 Counter Output (CO)

### Offset

| Register | Offset |
|---|---|
| CO | 14h |

### Function

Shows the value of the internal timer when SWT is disabled.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>CNT | Watchdog Count<br>When SWT is disabled (CR[WEN] is 0), CNT shows the value of the internal timer. When SWT is enabled (CR[WEN] is 1), it writes 0 to CNT. Values in this field can lag behind the internal timer value up to six system clock cycles plus eight counter clock cycles. Therefore, the CNT value that is read immediately after disabling SWT may be higher than the actual value of the internal timer. |

## 63.6.8 Service Key (SK)

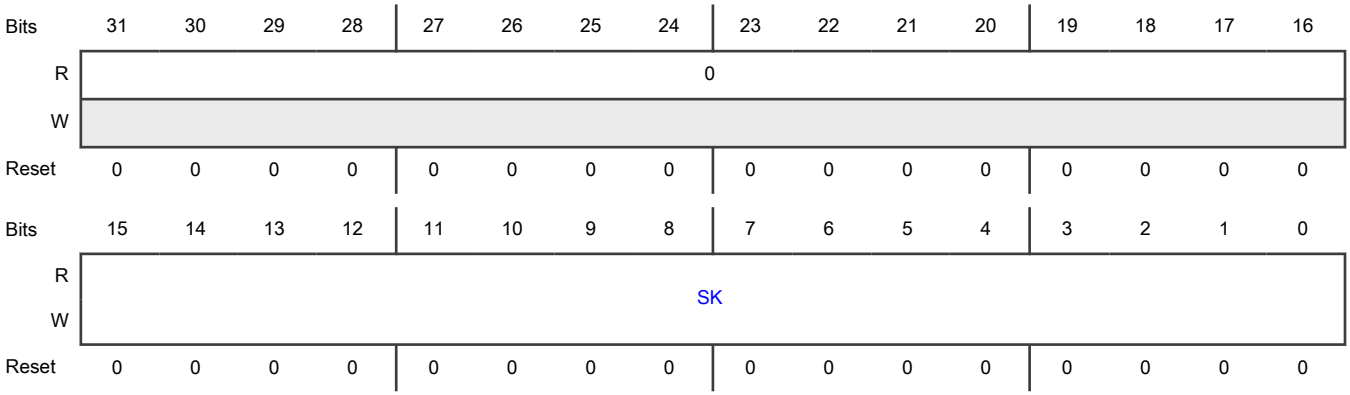
### Offset

| Register | Offset |
|---|---|
| SK | 18h |

### Function

Holds the previous (or initial) service key value. This register is read-only if either hard lock or soft lock is enabled (CR[HLK] or CR[SLK] is 1).

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>SK | Service Key<br>Holds the previous (or initial) service key value used in Initiate a keyed service sequence. If CR[SMD] is 01b, the next key value to write to SR is (17 * SK + 3) mod $2^{16}$. |

## 63.6.9 Event Request (RRR)

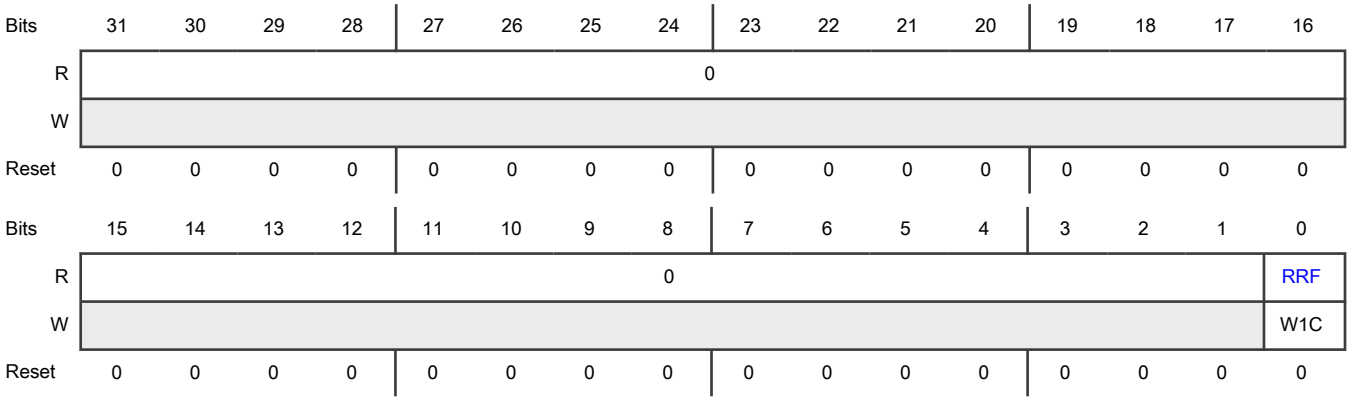
### Offset

| Register | Offset |
|---|---|
| RRR | 1Ch |

### Function

Contains the timeout reset request flag. See the chip-specific information for the specific event associated with this flag.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>RRF | Reset Request Flag<br>Write 1 to clear the flag and request. Writing 0 has no effect.<br>0b - No reset request<br>1b - Any reset request initiated |

# Chapter 64
# System Timer Module (STM)

## 64.1 Chip-specific STM information

### 64.1.1 STM instances and configuration

This chip has up to three instances of STM, one for each Cortex-M7 core. The STM counter increments at the STM module clock frequency divided by a pre-scaled value.

Table 357. STM instances

| Instance | MCXE31B | MCXE315/MCXE316/MCXE317 |
|---|---|---|
| STM_0 | Yes | Yes |
| STM_1 | Yes | No |

## 64.2 Overview

STM supports commonly required system and application software timing functions. STM includes a 32-bit count-up timer and four 32-bit compare channels with a separate interrupt source for each channel. The timer is driven by the STM module clock divided by an 8-bit prescale value (1 to 256).

### 64.2.1 Block diagram

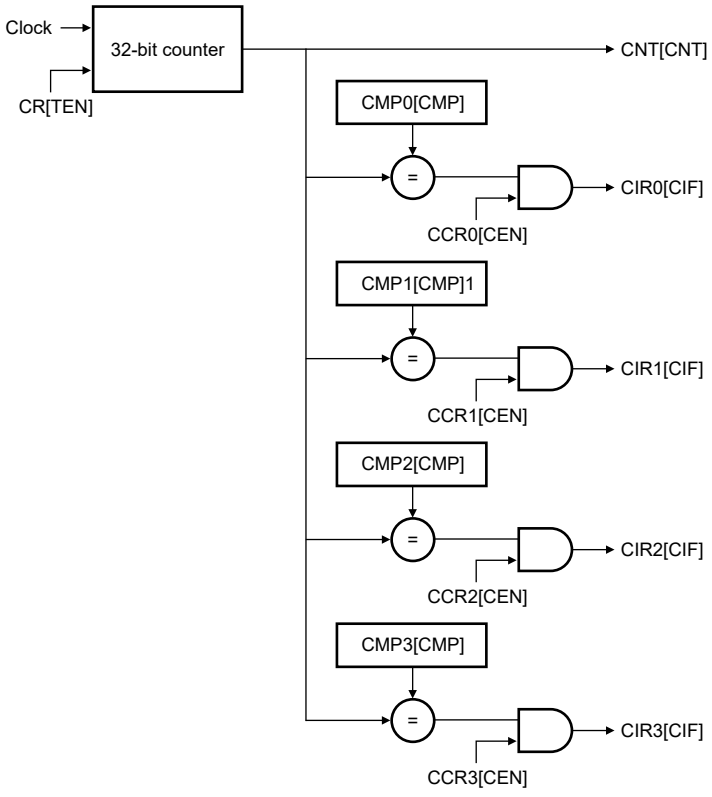



Figure 352. Block diagram

### 64.2.2 Features

STM has the following features:

- One 32-bit count-up timer with an 8-bit prescaler
- Four 32-bit compare channels
- An independent interrupt source for each channel
- Ability to stop the timer in Debug mode

## 64.3 Functional description

### 64.3.1 Count-up timer

STM has one 32-bit count-up timer that serves as the time base for four compare channels. When enabled, the counter increments at the module clock frequency divided by a prescaler value in the range from 1 to 256. When enabled in Normal mode, the timer increments continuously. The counter rolls over at FFFF_FFFFh to 0000_0000h with no restrictions at this boundary.

### 64.3.2 Compare channels

STM has four identical compare channels. Each channel includes a channel control register (CCR*n*), a channel interrupt register (CCR*n*), and a channel compare register (CMP*n*). When the channel is enabled and its channel compare value matches the timer count, STM sets the channel interrupt flag and generates an IRQ on that channel.

### 64.3.3 Behavior in different chip modes

STM supports the chip modes of operation as follows:

| Chip mode | STM behavior |
|---|---|
| Normal | When the timer is enabled (CR[TEN] = 1), the timer counts up continuously. |
| Debug | If CR[FRZ] = 1, STM stops the timer. Otherwise, when the timer is enabled (CR[TEN] = 1), the timer counts up continuously. |

### 64.3.4 Clocking

This module has no clocking considerations.

### 64.3.5 Interrupts

STM can generate a channel interrupt. For information, see:

- Compare channels
- Respond to compare channel events
- Channel Interrupt (CIR0 - CIR3)

## 64.4 External signals

This module has no external signals.

## 64.5 Initialization

This module does not require initialization.

## 64.6 Application information

### 64.6.1 Configure the timer

1. Set the initial timer count (CNT[CNT]).
2. Specify STM behavior in chip Debug mode (CNT[FRZ]).
3. Set the counter prescaler (CR[CPS]).
4. Start the timer (CR[TEN]).

### 64.6.2 Configure the compare channels

For each compare channel:

1. Set the channel compare value (CMP*n*[CMP]).
2. Enable the compare channel (CCR*n*[CEN]).

### 64.6.3 Respond to compare channel events

For each compare channel:

1. Check the channel interrupt flag (CIR*n*[CIF]).
2. If the channel interrupt flag is set, respond to the interrupt request.
3. When the channel interrupt has been handled, clear the channel interrupt flag (CIR*n*[CIF]).

## 64.7 STM register descriptions

The STM programming model allows only 32-bit (word) accesses. An attempted reference using a different size or to a reserved address generates a bus error termination.

### 64.7.1 STM memory map

STM_0 base address: 4027_4000h

STM_1 base address: 4047_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Control (CR) | 32 | RW | 0000_0000h |
| 4h | Count (CNT) | 32 | RW | 0000_0000h |
| 10h | Channel Control (CCR0) | 32 | RW | 0000_0000h |
| 14h | Channel Interrupt (CIR0) | 32 | RW | 0000_0000h |
| 18h | Channel Compare (CMP0) | 32 | RW | 0000_0000h |
| 20h | Channel Control (CCR1) | 32 | RW | 0000_0000h |
| 24h | Channel Interrupt (CIR1) | 32 | RW | 0000_0000h |
| 28h | Channel Compare (CMP1) | 32 | RW | 0000_0000h |
| 30h | Channel Control (CCR2) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 34h | Channel Interrupt (CIR2) | 32 | RW | 0000_0000h |
| 38h | Channel Compare (CMP2) | 32 | RW | 0000_0000h |
| 40h | Channel Control (CCR3) | 32 | RW | 0000_0000h |
| 44h | Channel Interrupt (CIR3) | 32 | RW | 0000_0000h |
| 48h | Channel Compare (CMP3) | 32 | RW | 0000_0000h |

## 64.7.2 Control (CR)

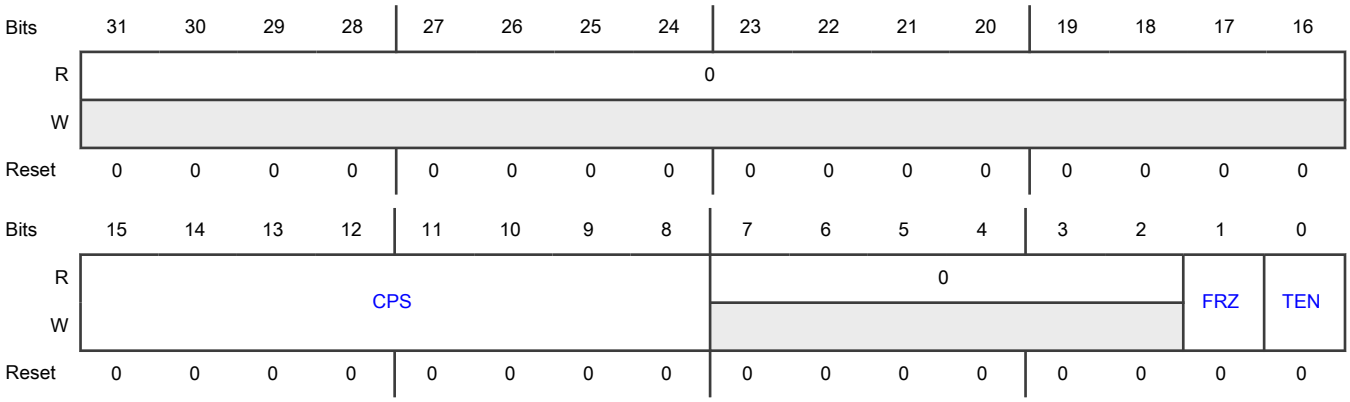
### Offset

| Register | Offset |
|---|---|
| CR | 0h |

### Function

Contains fields for the prescale value, freeze control, and timer enable.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>CPS | Counter Prescaler<br>Selects the module clock divide value for the prescaler (1–256). |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • 00h - Divide module clock by 1<br>• 01h - Divide module clock by 2<br>• ...<br>• FFh - Divide module clock by 256 |
| 7-2<br>— | Reserved |
| 1<br>FRZ | Freeze<br>Stops the timer when the chip enters Debug mode.<br>**NOTE**<br>When the chip enters Debug mode, it notifies STM, which in turn uses this field to determine timer operation.<br>0b - Timer runs in Debug mode<br>1b - Timer stops in Debug mode |
| 0<br>TEN | Timer Enable<br>Enables the module timer.<br>0b - Disabled<br>1b - Enabled |

### 64.7.3 Count (CNT)

#### Offset

| Register | Offset |
|---|---|
| CNT | 4h |

#### Function

Holds the timer count value.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>CNT | Timer Count<br>The time base for all compare channels. When enabled, the timer count increments at the rate of the module clock divided by the prescale value. |

## 64.7.4 Channel Control (CCR0 - CCR3)

### Offset

| Register | Offset |
|---|---|
| CCR0 | 10h |
| CCR1 | 20h |
| CCR2 | 30h |
| CCR3 | 40h |

### Function

Enables channel n of the timer.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>CEN | Channel Enable<br>0b - Disabled<br>1b - Enabled |

## 64.7.5 Channel Interrupt (CIR0 - CIR3)

### Offset

| Register | Offset |
|---|---|
| CIR0 | 14h |
| CIR1 | 24h |
| CIR2 | 34h |
| CIR3 | 44h |

### Function

Indicates and clears the interrupt flag for channel n of the timer.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | | | CIF |
| W | | | | | | | | | | | | | | | | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 0<br>CIF | Channel Interrupt Flag<br>Indicates the channel IRQ is asserted due to a match on the channel.<br>0b - Read: IRQ is not asserted. Write: No effect.<br>1b - Read: IRQ is asserted. Write: Clear the flag. |

### 64.7.6 Channel Compare (CMP0 - CMP3)

#### Offset

| Register | Offset |
|---|---|
| CMP0 | 18h |
| CMP1 | 28h |
| CMP2 | 38h |
| CMP3 | 48h |

#### Function
The compare value for channel n.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CMP | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | CMP | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-0<br>CMP | Channel Compare<br>If the channel is enabled (CCR*n*[CEN]), when the timer count (CNT) matches this value, STM asserts the channel IRQ and sets the channel interrupt flag (CIR*n*[CIF]). |

## 64.8 Glossary

| | |
|---|---|
| **IRQ** | Interrupt request |

# Chapter 65
# Periodic Interrupt Timer (PIT)

## 65.1 Chip-specific PIT information

### 65.1.1 PIT instances and configuration

This chip has up to 3 instances of the PIT each with 4 PIT timers/channels, each channel being 32 bits in length.

**Table 358. PIT instances**

| Instance | MCXE31B | MCXE317 | MCXE315/MCXE316 |
|---|---|---|---|
| PIT_0 | Yes | Yes | Yes |
| PIT_1 | Yes | Yes | Yes |
| PIT_2 | Yes | No | No |

**Table 359. PIT instances and features**

| Feature | PIT_0 | PIT_1 | PIT_2 |
|---|---|---|---|
| Real time interrupt (RTI) | Yes (32-bit)[1] | No | No |
| Lifetime Timer | Yes | No | No |
| Timer chaining to create a 64-bit timer | Yes | No | No |
| Number of 32-bit timer channels | 4 | 4 | 4 |
| Timer value unchanged by a non-destructive reset | No | No | No |

1. The RTI on PIT_0 supports operation in STANDBY mode

The RTI shall be possible to set the timer resolution to 1us independently to other timers. The interrupt can be generated by programming LDVAL0, LDVAL1, LDVAL2 and LDVAL3 to 48 in case of 48 MHz FIRC as system clock and 40 in case of 40 MHz system clock to generate an interrupt after 1us.

### 65.1.2 Input Output

All the PIT instances are capable of generating periodic triggers. PIT triggers are routed to motor control IPs such as eMIOS, LCU, BCTU, ADC etc. via TRGMUX. See the TRGMUX connectivity file attached to this document for details.

## 65.2 Overview

PIT includes:

- An array of timers that can be used to raise interrupts and triggers.
- A dedicated RTI timer that runs on a separate oscillator clock and can be used for system wakeup

### 65.2.1 Block diagram

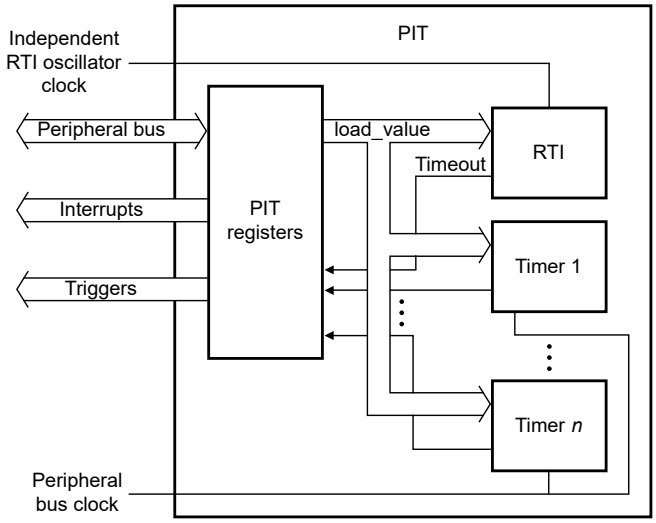



Figure 353. Block diagram

### 65.2.2 Features

The key features of the module are:

- 4 32-bit countdown timers
- Independent timeout periods for each timer
- Ability to chain two timers into a 64-bit lifetimer
- Ability of timers to generate interrupts
- Ability of timers to generate trigger pulses
- Maskable interrupts
- One RTI countdown timer to wake up the CPU
- Option to generate the RTI even when the bus clock is off
- Power saving with a separate input clock for the RTI timer. All other timers share a common core clock.

## 65.3 Functional description

See the chip-specific PIT information for the number of PIT modules and their channel configuration on this chip.

### 65.3.1 Modes of operation

| Mode | Description |
|---|---|
| Run | All functional parts of the PIT operate normally. |
| Debug | Timers run or stop depending on the Freeze setting (MCR[FRZ]). See Debug mode. |

### 65.3.2 Timer operation

When you enable a timer (write 1 to TCTRL*n*[TEN] or RTI_TCTRL[TEN]), the timer counts down (one count per clock cycle) from its initial start value (LDVAL*n*[TSV] or RTI_LDVAL[TSV]). When the timer period expires (CVAL*n*[TVL] or RTI_CVAL[TVL] reaches 0), PIT sets the Timer Interrupt Flag (writes 1 to TFLG*n*[TIE] or RTI_TFLG[TIE]). It then reloads the timer start value and starts the timer counting down again.

If the timer interrupt is enabled (TCTRL*n*[TIE] or RTI_TCTRL[TIE] = 1), it generates an interrupt when it sets the Timer Interrupt Flag. The timer cannot generate a new interrupt until you clear the flag from the previous interrupt. Because clearing the RTI crosses clock domains, you should use a minimum RTI timer load value of 32.

You can read the current counter value of any timer (CVAL*n*[TVL] or RTI_CVAL[TVL]). Because the RTI timer is synchronized to the RTI clock domain, the value of the RTI counter may be several cycles old when read.

See these related topics:

- Stop and start a timer
- Change timer period
- Change timer period dynamically

**NOTE**

- If Freeze is enabled (MCR[FRZ] = 1) when the timer nears 0, the Debug mode command may not reach PIT before the timer expires and generates a trigger.
- If a timer is at zero when Debug mode is asserted, the interrupt trigger remains asserted until the device exits Debug mode. Clear the Timer Interrupt Flag (write 1 to TFLG*n*[TIF] or RTI_TFLG[TIF]) upon Debug exit.
- If Freeze is enabled (MCR[FRZ] = 1), sufficient time must be ensured for the IP to react.

#### 65.3.2.1 Stop and start a timer

To restart a running timer:

1. Disable the timer (write 0 to TCTRL*n*[TEN] or RTI_TCTRL[TEN]).
2. Enable the timer (write 1 to TCTRL*n*[TEN] or RTI_TCTRL[TEN]).

   Enabling the timer reloads the start value and starts the timer counting down again. See Restart counter period.

#### 65.3.2.2 Restart counter period

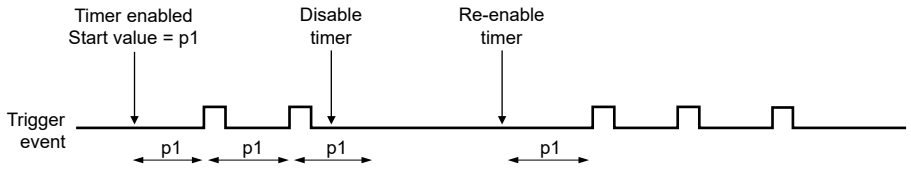



Figure 354. Restart counter period

#### 65.3.2.3 Change timer period

To change the counter period of a running timer:

1. Disable the timer (write 0 to TCTRL*n*[TEN] or RTI_TCTRL[TEN]).
2. Specify a new start value (LDVAL*n*[TSV] or RTI_LDVAL[TSV]).
3. Enable the timer (write 1 to TCTRL*n*[TEN] or RTI_TCTRL[TEN]).

   Enabling the timer loads the new start value and starts the timer counting down again. See Change current timer period.

#### 65.3.2.4 Change current timer period

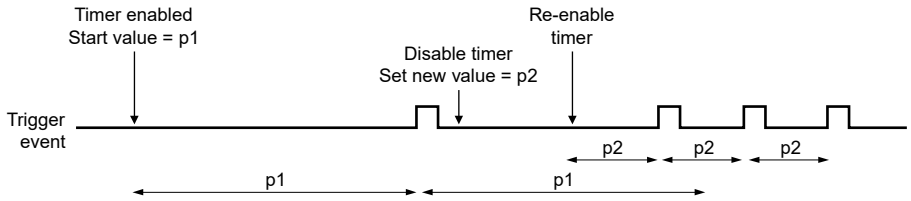



**Figure 355. Change timer period dynamically timing**

#### 65.3.2.5 Change timer period dynamically

To change the counter period of a running timer:

- Specify a new start value (LDVAL*n*[TSV] or RTI_LDVAL[TSV])

  The next time the timer expires, it loads the new start value. See Dynamically setting a new timer period.

#### 65.3.2.6 Dynamically setting a new timer period

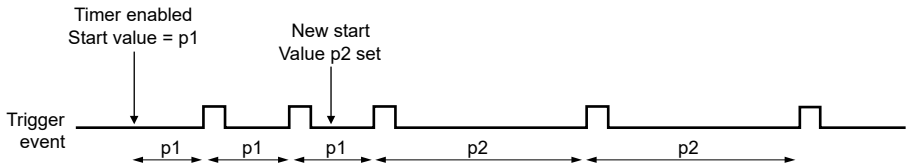



**Figure 356. Dynamically setting a new timer period**

### 65.3.3 Chained timers

PIT supports chaining two timers together to support timer periods greater than FFFFFFFFh. When a timer *n* has chain mode enabled, it decrements each time timer *n* - 1 expires. The chained timer sets its Timer Interrupt Flag when it expires (CVAL*n* = 0).

For example, assume you need to configure a timer period of 6,000,000,000 cycles. You could specify a start value of 10 for timer *n* and 599,999,999 for timer *n* - 1. With these values, the chained timer expires after timer *n* - 1 has expired ten times.

You cannot chain timer 0 or the RTI timer to another timer.

For an example configuration of a chained timer, see Example configuration using chained timers.

### 65.3.4 Lifetimer

PIT supports a 64-bit lifetimer. You configure the lifetimer by chaining timer 1 to timer 0 and populating the start values for both timers with the maximum value of FFFFFFFFh. You read the lifetimer in the LTMR64H and LTMR64L registers. LTMR64H returns the upper 32 bits of the lifetimer value and LMTR64L returns the lower 32 bits. When a timer 1 has chain mode enabled, it decrements each time timer 0 expires. The lifetimer expires when CVAL1 expires.

When you chain timer 1 to timer 0, timer 1 decrements every time timer 0 expires. To obtain the correct lifetimer value, read LTMR64H first and then read LTMR64L. When you read LTMR64H, it returns the value in CVAL1 (the chained timer value), and immediately latches the value from CVAL0 into LTMR64L. This latching eliminates the carryover effects of the running counter.

For an example configuration of the lifetimer, see Example configuration using the lifetimer.

### 65.3.5 Debug mode

If Freeze is enabled (MCR[FRZ] = 1), the timers stop if the device enters Debug mode. This freeze allows the software developer to halt the processor, investigate the current state of the system, including the PIT timer values, and then continue operation.

### 65.3.6 Clocking

The PIT general-purpose timers are driven by the peripheral bus clock. See the Clocking chapter for more information.

The RTI timer is driven by an independent oscillator clock. This clock input continues running when the device is in Stop mode. Driving the RTI timer with an independent clock allows the RTI to trigger periodic wake-up from Stop mode.

### 65.3.7 Interrupts

All of the timers support interrupt generation.

You enable interrupt generation for each individual timer (TCTRL*n*[TIE]). Each interrupt is available on a separate interrupt output.

When a timer expires (CVAL*n*[TVL] = 0), it sets the Timer Interrupt Flag (TFLG*n*[TIF]) regardless of whether the interrupt is enabled. To enable generation of an interrupt when the Timer Interrupt Flag is set, write 1 to TCTRL*n*[TIE]. To clear the Timer Interrupt Flag, write 1 to it.

When the RTI timer expires (RTI_CVAL*n*[TVL] = 0), PIT sets the RTI Timer Interrupt Flag (RTI_TFLG*n*[TIF]) regardless of whether the RTI is enabled. To enable generation of an RTI when the RTI Timer Interrupt Flag is set, write 1 to TCTRL*n*[TIE]. To clear the RTI Timer Interrupt Flag, write 1 to it.

The RTI is typically used for periodic wake-up from Standby mode, but it can also be used to generate a general-purpose interrupt.

## 65.4 External signals

PIT has no external inputs.

## 65.5 Initialization

To initialize PIT:

1. For each timer *n* you plan to use:
    a. Specify the timer period (LDVAL*n*[TSV]).

       To calculate the load value for a timer:

       Start value = (timer period / clock period) - 1

    b. If you want the timer to generate an interrupt each time it expires, enable the timer interrupt (write 1 to TCTRL*n*[TIE]).
    c. If you want to use this timer as the divisor of a pair of chained timers, chain this timer to the next lower-numbered timer (write 1 to TCTRL*n*[CHN]). You cannot chain timer 0. If you want to use the lifetimer, chain timer 1 to timer 0.
    d. Enable the timer (write 1 to TCTRL*n*[TEN]).
2. If you want to use the RTI feature:
    a. Specify the RTI timer period (RTI_LDVAL*n*[TSV]).
    b. Enable the RTI timer interrupt (write 1 to RTI_TCTRL*n*[TIE]).
    c. Enable the RTI timer (write 1 to RTI_TCTRL*n*[TEN]).
3. Activate, or enable, PIT (write 0 to MCR[MDIS]).

   All enabled timers load their start values and begin counting down.

For example configurations, see Application information.

## 65.6 Application information

### 65.6.1 Example configuration for general timers

In this example configuration:

- The PIT clock frequency is 50 MHz, which equates to a clock cycle of 20 ns.
- The RTI clock frequency is 10 MHz, which equates to a clock cycle of 100 ns.
- The RTI timer needs to generate a wake-up interrupt every 500 ms.
- Timer 1 needs to generate an interrupt every 5.12 ms.
- Timer 3 needs to generate a trigger event every 30 ms.

To configure PIT for this example:

1. Disable PIT (write 1 to MCR[MDIS]).
2. Use the formula in Initialization to calculate the load value for the RTI timer:

   (500 ms/100 ns) - 1 = 4,999,999 cycles, or 004C4B3Fh.
3. Specify the RTI timer load value (write 004C4B3Fh to RTI_LDVAL[TSV]).
4. Enable the RTI (write 1 to RTI_TCTRL[TIE])
5. Enable the RTI timer (write 1 to RTI_TCTRL*n*[TEN]).
6. Use the formula in Initialization to calculate the load value for timer 1:

   (5.12 ms/20 ns) - 1 = 255,999 cycles, or 0003E7FFh.
7. Disable timer 1 (write 0 to TCTRL1[TEN]).
8. Specify the timer 1 value (write 0003E7FFh to LDVAL1[TSV]).
9. Enable the timer 1 interrupt (write 1 to TCTRL1[TIE])
10. Enable timer 1 (write 1 to TCTRL1[TEN]).
11. Calculate the load value for timer 3:

    (30 ms/20 ns) - 1 = 1,499,999 cycles, or 0016E35Fh.
12. Disable timer 3 (write 0 to TCTRL3[TEN]).
13. Specify the timer 3 value (write 0016E35Fh to LDVAL3[TSV]).
14. Enable timer 3 (write 1 to TCTRL3[TEN]).
15. Activate, or enable, PIT (write 0 to MCR[MDIS]).

**NOTE**

You can configure timers without first disabling PIT or the timer itself. However, if PIT is running, be aware that when a timer begins using a new start value depends on whether the timer is running or disabled when you specify the new start value. For more information, see Timer operation.

The following example code provides the described configuration:

```
// Disable PIT (see note above)
PIT_MCR = 0x02;
```

```
// RTI
PIT_RTI_LDVAL = 0x004C4B3F; // Configure RTI for 5000000 cycles
```

```
PIT_RTI_TCTRL = PIT_TIE;    // Enable RTI
PIT_RTI_TCTRL |= PIT_TEN;   // Start RTI timer
```

```
// Timer 1
PIT_LDVAL1 = 0x0003E7FF;    // Configure timer 1 for 256000 cycles
PIT_TCTRL1 = TIE;           // Enable Timer 1 interrupts
PIT_TCTRL1 |= TEN;          // Start Timer 1
```

```
// Timer 3
PIT_LDVAL3 = 0x0016E35F;    // Configure timer 3 for 1500000 cycles
PIT_TCTRL3 = TEN;           // Enable Timer 3
```

```
 // Enable PIT
 PIT_MCR = 0x00;
```

### 65.6.2 Example configuration using chained timers

In this example configuration:

- The PIT clock frequency is 100 MHz, which equates to a clock cycle of 10 ns.
- Timers 1 and 2 are available.
- An interrupt is needed every minute. Using the formula in Initialization, a 1 minute timer period requires:

  (60 s/100 ns) - 1 = 5,999,999,999 cycles, or 165A0BBFFh

  This value requires more than 32 bits.

To configure PIT for this example:

1. Disable PIT (write 1 to MCR[MDIS]).
2. Disable timer 1 (write 0 to TCTRL1[TEN]).
3. Disable timer 2 (write 0 to TCTRL2[TEN]).
4. Chain timer 2 to timer 1 (write 1 to TCTRL2[CHN]).
5. For the timer 2 start value, select a value that divides evenly into the desired period. For this example, we are using 10. For the timer 1 value, divide the full timer period by the selected divisor: 6,000,000,000/10 = 600,000,000.
6. Specify the timer 2 value as the selected divisor (write 00000009h to LDVAL2[TSV]).
7. Specify the timer 1 value as the quotient minus 1, or 599,999,999 (write 0x23C345FFh to LDVAL1[TSV]).
8. Enable the timer 2 interrupt (write 1 to TCTRL2[TIE])
9. Enable timer 2 (write 1 to TCTRL2[TEN]).
10. Enable timer 1 (write 1 to TCTRL1[TEN]).
11. Activate, or enable, PIT (write 0 to MCR[MDIS]).

The following example code matches the described configuration:

```
// Disable PIT
PIT_MCR = 0x02;
```

```
// Timer 2
PIT_LDVAL2 = 0x00000009; // Configure Timer 2 to the divisor (10)
PIT_TCTRL2 = TIE;        // Enable Timer 2 interrupt
PIT_TCTRL2 |= CHN;       // Chain Timer 2 to Timer 1
PIT_TCTRL2 |= TEN;       // Enable Timer 2
```

```
// Timer 1
PIT_LDVAL1 = 0x23C345FF; // Configure Timer 1 to 600,000,000 cycles
PIT_TCTRL1 = TEN;        // Enable Timer 1
```

```
 // Enable PIT
 PIT_MCR = 0x00;
```

### 65.6.3 Example configuration using the lifetimer

To configure the lifetimer:

1. Disable PIT (write 1 to MCR[MDIS]).
2. Disable timer 0 (write 0 to TCTRL0[TEN]).
3. Disable timer 1 (write 0 to TCTRL1[TEN]).
4. Chain timer 1 to timer 0 (write 1 to TCTRL1[CHN]).
5. Specify the maximum start value for timer 1 (write FFFFFFFFh to LDVAL1[TSV]).
6. Specify the maximum start value for timer 0 (write FFFFFFFFh to LDVAL0[TSV]).
7. Enable timer 1 (write 1 to TCTRL1[TEN]).
8. Enable timer 0 (write 1 to TCTRL0[TEN]).
9. Activate, or enable, PIT (write 0 to MCR[MDIS]).

To access the lifetimer, first read LTMR64H and then LTMR64L. For more information, see Lifetimer.

The following example code matches the described setup:

```
// Disable PIT
PIT_MCR = 0x02;
```

```
// Timer 1
PIT_LDVAL1 = 0xFFFFFFFF; // setup timer 1 for maximum counting period
PIT_TCTRL1 = 0x0; // disable timer 1 interrupts
PIT_TCTRL1 |= CHN; // chain timer 1 to timer 0
PIT_TCTRL1 |= TEN; // start timer 1
```

```
// Timer 0
```

```
PIT_LDVAL0 = 0xFFFFFFFF; // setup timer 0 for maximum counting period
PIT_TCTRL0 = TEN; // start timer 0
```

```
// Read lifetimer
current_uptime = PIT_LTMR64H<<32;
current_uptime = current_uptime + PIT_LTMR64L;
```

```
 // Enable PIT
 PIT_MCR = 0x00;
```

## 65.7 PIT register descriptions

This section provides a detailed description of all PIT registers.

Program RTI registers only when the RTI clock is running.

### 65.7.1 PIT memory map

PIT_0 base address: 400B_0000h

PIT_1 base address: 400B_4000h

PIT_2 base address: 402F_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | PIT Module Control (MCR) | 32 | RW | See section |
| E0h | PIT Upper Lifetimer (LTMR64H) | 32 | R | 0000_0000h |
| E4h | PIT Lower Lifetimer (LTMR64L) | 32 | R | 0000_0000h |
| ECh | RTI Timer Load Value Sync Status (RTI_LDVAL_STAT) | 32 | RW | 0000_0000h |
| F0h | RTI Timer Load Value (RTI_LDVAL) | 32 | RW | 0000_0000h |
| F4h | Current RTI Timer Value (RTI_CVAL) | 32 | R | 0000_0000h |
| F8h | RTI Timer Control (RTI_TCTRL) | 32 | RW | 0000_0000h |
| FCh | RTI Timer Interrupt Flag (RTI_TFLG) | 32 | RW | 0000_0000h |
| 100h | Timer Load Value (LDVAL0) | 32 | RW | 0000_0000h |
| 104h | Current Timer Value (CVAL0) | 32 | R | 0000_0000h |
| 108h | Timer Control (TCTRL0) | 32 | RW | 0000_0000h |
| 10Ch | Timer Flag (TFLG0) | 32 | RW | 0000_0000h |
| 110h | Timer Load Value (LDVAL1) | 32 | RW | 0000_0000h |
| 114h | Current Timer Value (CVAL1) | 32 | R | 0000_0000h |
| 118h | Timer Control (TCTRL1) | 32 | RW | 0000_0000h |
| 11Ch | Timer Flag (TFLG1) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 120h | Timer Load Value (LDVAL2) | 32 | RW | 0000_0000h |
| 124h | Current Timer Value (CVAL2) | 32 | R | 0000_0000h |
| 128h | Timer Control (TCTRL2) | 32 | RW | 0000_0000h |
| 12Ch | Timer Flag (TFLG2) | 32 | RW | 0000_0000h |
| 130h | Timer Load Value (LDVAL3) | 32 | RW | 0000_0000h |
| 134h | Current Timer Value (CVAL3) | 32 | R | 0000_0000h |
| 138h | Timer Control (TCTRL3) | 32 | RW | 0000_0000h |
| 13Ch | Timer Flag (TFLG3) | 32 | RW | 0000_0000h |

## 65.7.2 PIT Module Control (MCR)

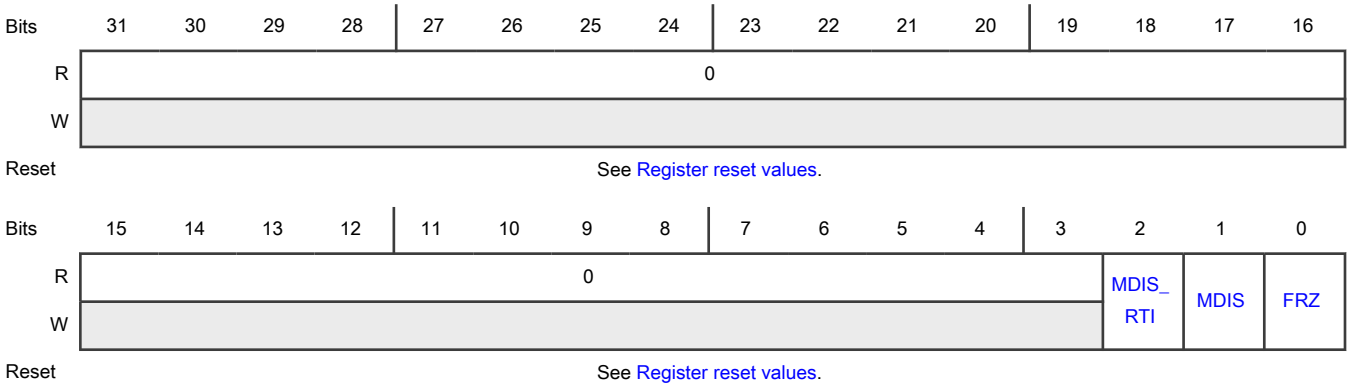
### Offset

| Register | Offset |
|---|---|
| MCR | 0h |

### Function

Enables the PIT timer clocks and specifies the behavior of the timers when PIT enters Debug mode.

### Diagram




### Register reset values

| Register | Reset value |
|---|---|
| MCR | PIT_0: 0000_0006h<br>PIT_1,PIT_2: 0000_0002h |

**Fields**

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2<br>MDIS_RTI | Module Disable for RTI<br>Disables the RTI timer. You must write 1 to this field before setting up an RTI.<br>**NOTE**<br>This field is not supported in every instance. The following table includes only supported registers.<br>0b - Enables<br>1b - Disables |
| 1<br>MDIS | Module Disable for PIT<br>Disables the standard timers. This field does not affect the RTI timer. You must disable the clock (write 1 to MDIS) before configuring PIT. After configuration, you must enable the clock (write 0 to MDIS).<br>0b - Enables<br>1b - Disables |
| 0<br>FRZ | Freeze<br>Stops the timers when the device enters Debug mode.<br>0b - Timers run in Debug mode<br>1b - Timers stop in Debug mode |

| Instance | Field supported in | Field not supported in |
|---|---|---|
| PIT_0 | MCR | — |
| PIT_1 | — | MCR |
| PIT_2 | — | MCR |

## 65.7.3 PIT Upper Lifetimer (LTMR64H)

**Offset**

| Register | Offset |
|---|---|
| LTMR64H | E0h |

**Function**

Combined with LTMR64L, provides a 64-bit lifetimer constructed from the values in two chained PIT timers.

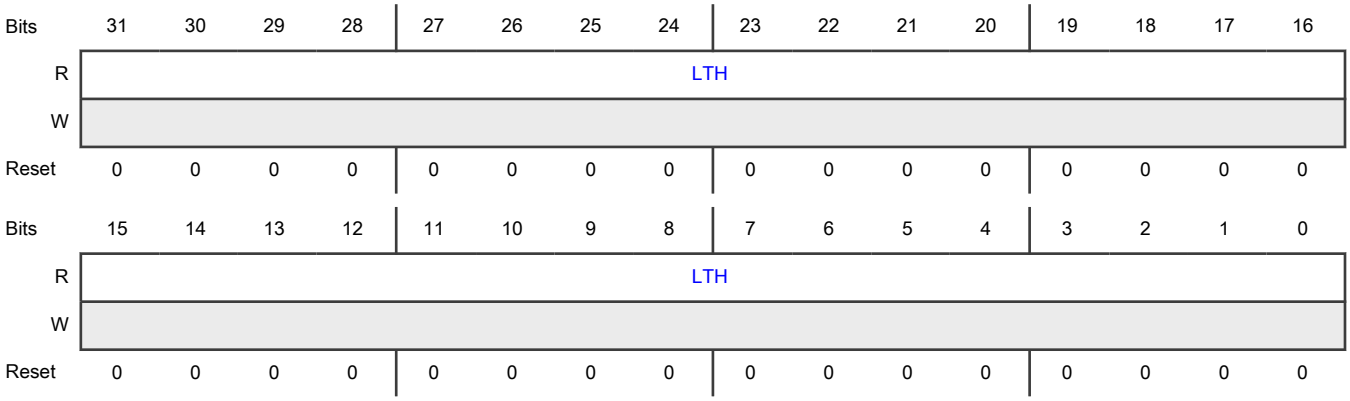
For more information, see Lifetimer.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| PIT_0 | LTMR64H | — |
| PIT_1 | — | LTMR64H |
| PIT_2 | — | LTMR64H |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>LTH | Lifetimer Value<br>Indicates the timer value of timer 1. This value is the upper 32 bits of the 64-bit lifetimer value. When you read this register at t1, the value of timer 0 at t1 is latched into LTMR64L. |

## 65.7.4 PIT Lower Lifetimer (LTMR64L)

### Offset

| Register | Offset |
|---|---|
| LTMR64L | E4h |

### Function

Combined with LTMR64H, provides a 64-bit lifetimer.

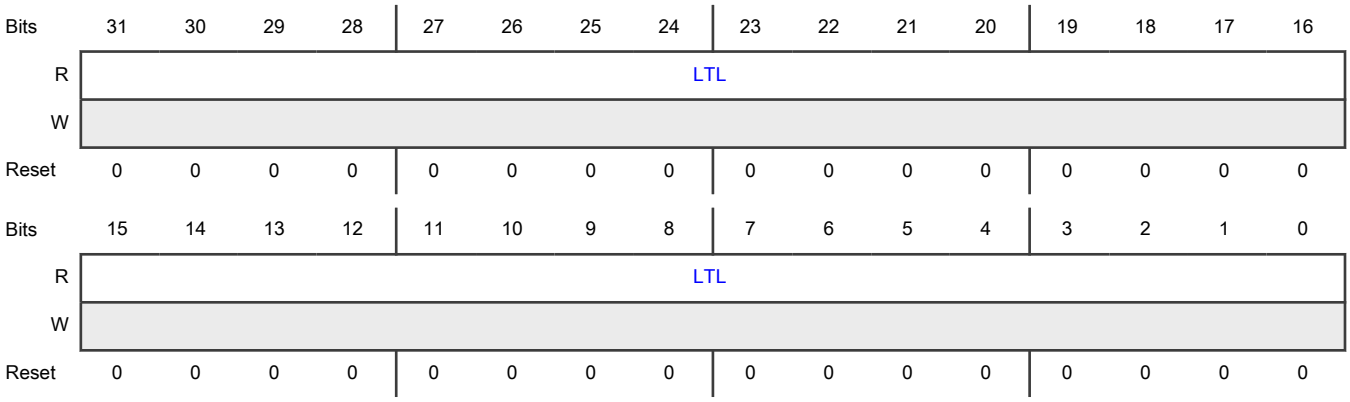
For more information, see Lifetimer.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| PIT_0 | LTMR64L | — |
| PIT_1 | — | LTMR64L |
| PIT_2 | — | LTMR64L |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>LTL | Lifetimer Value<br>Indicates the timer value of timer 0 at the moment LTMR64H was last read. This value is the lower 32 bits of the 64-bit lifetimer value. This field updates only when LTMR64H is read. |

## 65.7.5 RTI Timer Load Value Sync Status (RTI_LDVAL_STAT)

### Offset

| Register | Offset |
|---|---|
| RTI_LDVAL_STAT | ECh |

### Function

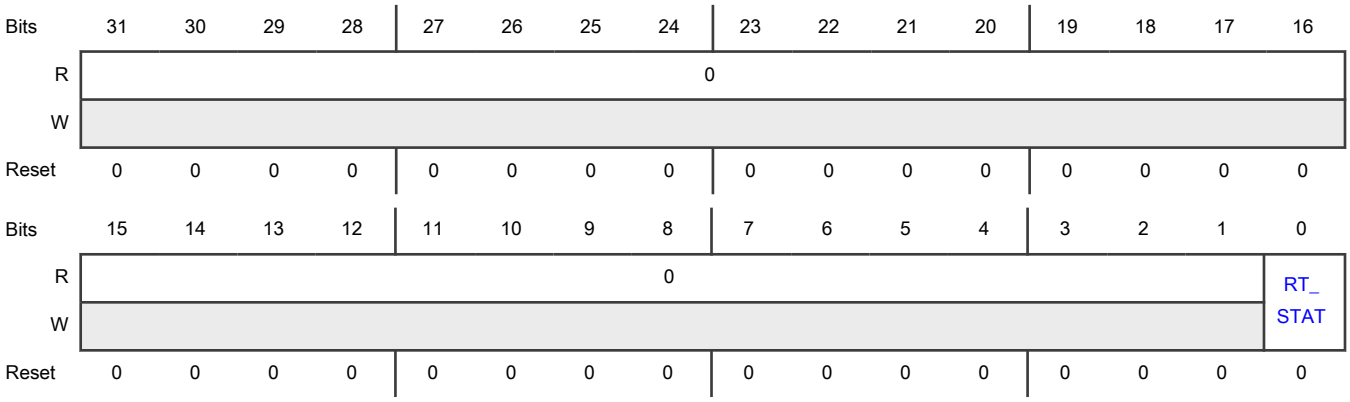
Indicates whether the RTI timer start value has been loaded (RTI_LDVAL). This status is necessary because it takes several cycles for the RTI start value to synchronize into the RTI clock domain.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| PIT_0 | RTI_LDVAL_STAT | — |
| PIT_1 | — | RTI_LDVAL_STAT |
| PIT_2 | — | RTI_LDVAL_STAT |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>RT_STAT | Sync Status<br>Indicates whether the RTI start value is loaded.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not loaded<br>1b - Loaded<br>When writing<br>0b - Clears status<br>1b - Clears status |

## 65.7.6 RTI Timer Load Value (RTI_LDVAL)

### Offset

| Register | Offset |
|---|---|
| RTI_LDVAL | F0h |

### Function

Specifies the length of the RTI timeout period in clock cycles.

It takes several cycles for this value to synchronize into the RTI clock domain.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| PIT_0 | RTI_LDVAL | — |
| PIT_1 | — | RTI_LDVAL |
| PIT_2 | — | RTI_LDVAL |

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>TSV | Timer Start Value<br>Specifies the starting RTI timer value. The timer counts down until it reaches 0, then sets the interrupt flag and reloads this value. When you write a new value to this register, the timer does not restart with the new value until the current timing period expires. To terminate the current period and start a new period with the new value, you must disable the timer (write 0 to RTI_TCTRL*n*[TEN]) and then enable it again (write 1 to RTI_TCTRL*n*[TEN]). |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | **NOTE**<br>Because it may take several cycles to clear the RTI interrupt, setting the value to something less than 32 cycles may result in a lost interrupt. |

## 65.7.7 Current RTI Timer Value (RTI_CVAL)

### Offset

| Register | Offset |
|---|---|
| RTI_CVAL | F4h |

### Function

Indicates the current RTI timer value. Because this timer might be driven from a different clock domain from the other timers, the value may be several cycles old.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| PIT_0 | RTI_CVAL | — |
| PIT_1 | — | RTI_CVAL |
| PIT_2 | — | RTI_CVAL |

### Diagram

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

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

**Fields**

| Field | Function |
|---|---|
| 31-0<br>TVL | Current Timer Value<br>Indicates the current RTI timer value.<br>**NOTE**<br>• If the timer is disabled (RTI_TCTRL*n*[TEN] = 0), this value has no meaning.<br>• If PIT is frozen in Debug mode (MCR[FRZ] = 1), timer values are frozen when the device is in Debug mode. |

## 65.7.8 RTI Timer Control (RTI_TCTRL)

**Offset**

| Register | Offset |
|---|---|
| RTI_TCTRL | F8h |

**Function**

Controls RTI timer behavior.

The RTI may take several RTI clock cycles to enable or update. Therefore, you must wait for at least three RTI clock cycles after RTI configuration before relying on the RTI timer.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| PIT_0 | RTI_TCTRL | — |
| PIT_1 | — | RTI_TCTRL |
| PIT_2 | — | RTI_TCTRL |

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | | Reserved | TIE | TEN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2<br>— | Reserved |
| 1<br>TIE | Timer Interrupt Enable<br>Enables the RTI timer interrupt. If the interrupt is enabled, when an interrupt is pending or the Timer Interrupt Flag is set (RTI_TFLG*n*[TIF] = 1), PIT generates an interrupt. To avoid an interrupt when you first enable the interrupt, clear the Timer Interrupt Flag (write 1 to RTI_TFLG*n*[TIF]) before you enable the interrupt.<br>0b - Disables<br>1b - Enables |
| 0<br>TEN | Timer Enable Bit<br>Enables the RTI timer.<br>0b - Disables<br>1b - Enables. The RTI timer begins counting down. |

## 65.7.9 RTI Timer Interrupt Flag (RTI_TFLG)

### Offset

| Register | Offset |
|---|---|
| RTI_TFLG | FCh |

### Function

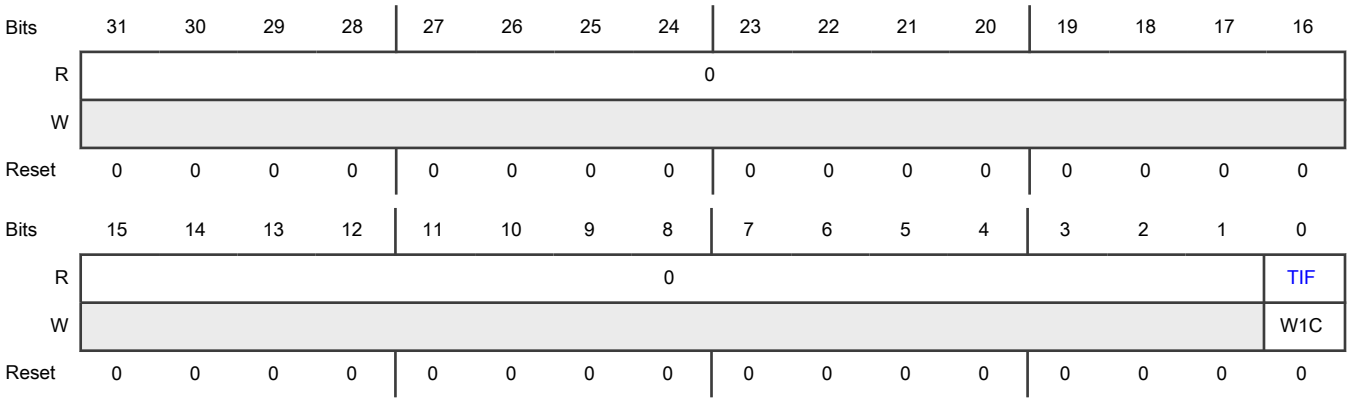
Indicates the RTI timer has expired.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| PIT_0 | RTI_TFLG | — |
| PIT_1 | — | RTI_TFLG |
| PIT_2 | — | RTI_TFLG |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>TIF | Timer Interrupt Flag<br>Indicates the RTI timer period has expired (CVAL*n*[TVL] = 0). If interrupts are enabled (RTI_TCTRL*n*[TIE] = 1), TIF triggers an interrupt request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Timer still counting down<br>1b - Timer has expired<br>When writing<br>0b - No effect<br>1b - Clears the flag |

## 65.7.10 Timer Load Value (LDVAL0 - LDVAL3)

### Offset

| Register | Offset |
|---|---|
| LDVAL0 | 100h |
| LDVAL1 | 110h |
| LDVAL2 | 120h |
| LDVAL3 | 130h |

### Function

Specifies the length of the timeout period in clock cycles.

The value change is visible immediately. The synchronization mechanism allows 0 wait states in this case.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSV | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSV | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>TSV | Timer Start Value<br>Specifies the starting timer value. The timer counts down until it reaches 0, then sets the interrupt flag and reloads this value. When you write a new value to this register, the timer does not restart with the new value until the current timing period expires. To terminate the current period and start a new period with the new value, you must disable the timer (write 0 to TCTRL*n*[TEN]) and then enable it again (write 1 to TCTRL*n*[TEN]).<br>**NOTE**<br>Because the CPU requires several cycles to service an interrupt, there is a practical limit to the lowest start value for a timer. Setting the value too low may result in a lost interrupt. |

## 65.7.11 Current Timer Value (CVAL0 - CVAL3)

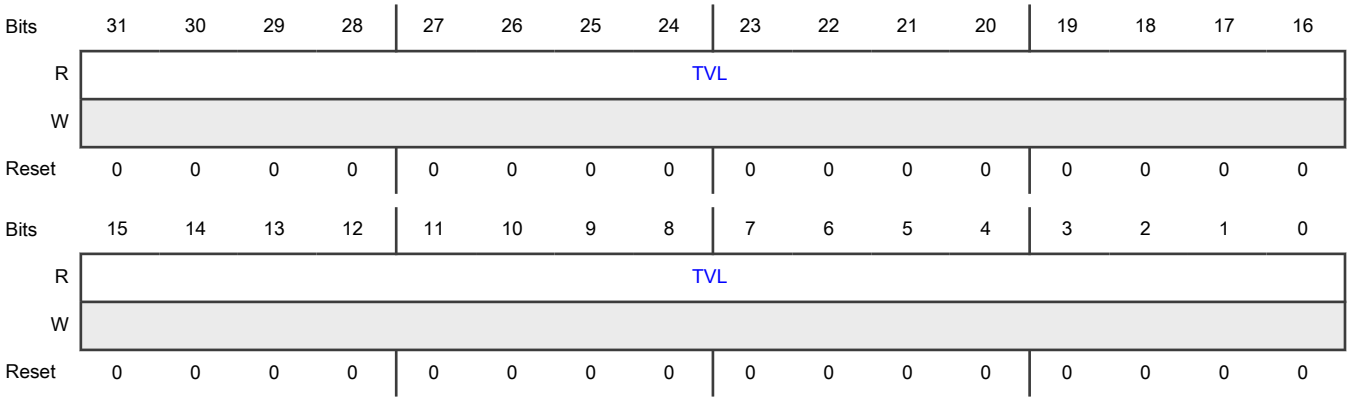
### Offset

| Register | Offset |
|---|---|
| CVAL0 | 104h |
| CVAL1 | 114h |
| CVAL2 | 124h |
| CVAL3 | 134h |

### Function

Indicates the current timer value.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>TVL | Timer Value<br>Indicates the current timer value.<br>**NOTE**<br>• If the timer is disabled (TCTRL*n*[TEN] = 0), this value has no meaning.<br>• If PIT is frozen in Debug mode (MCR[FRZ] = 1), timer values are frozen when the device is in Debug mode. |

## 65.7.12 Timer Control (TCTRL0 - TCTRL3)

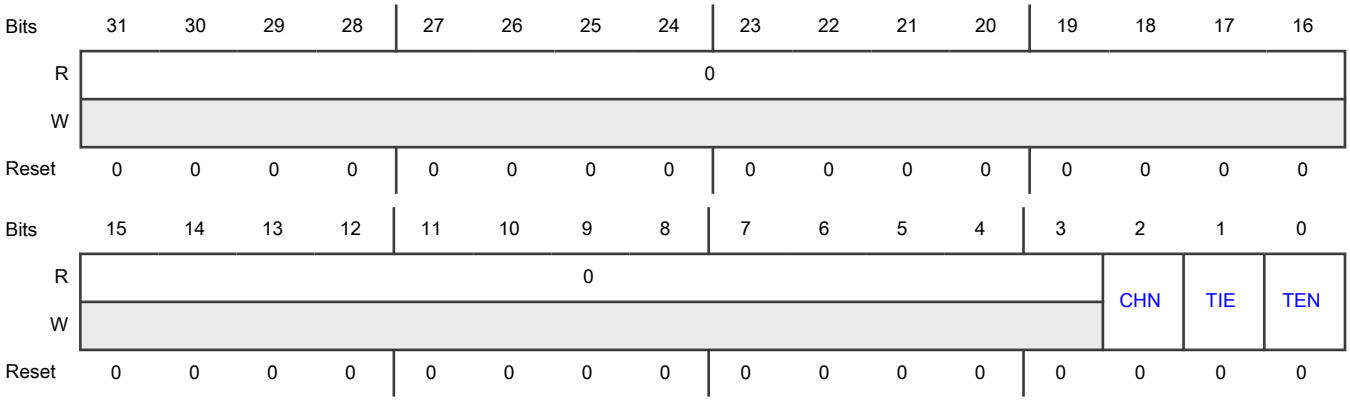
### Offset

| Register | Offset |
|---|---|
| TCTRL0 | 108h |
| TCTRL1 | 118h |
| TCTRL2 | 128h |
| TCTRL3 | 138h |

### Function

Controls timer behavior.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2<br>CHN | Chain Mode<br>Chains this timer (timer *n*, where *n* is > 0) to the next lower numbered timer (timer *n* - 1) to create a 64-bit lifetimer. When chained, timer *n* decrements when timer *n* - 1 expires. See LTMR64H and LTMR64L.<br>You cannot chain timer 0 or the RTI timer.<br>0b - Unchains<br>1b - Chains |
| 1<br>TIE | Timer Interrupt Enable<br>Enables the timer interrupt. If the interrupt is enabled, when an interrupt is pending or the Timer Interrupt Flag is set (TFLG*n*[TIF] = 1), PIT generates an interrupt. To avoid an interrupt when you first enable the interrupt, clear the Timer Interrupt Flag (write 1 to TFLG*n*[TIF]) before you enable the interrupt. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disables<br>1b - Enables |
| 0<br>TEN | Timer Enable<br>Enables the timer.<br>0b - Disables<br>1b - Enables. The timer begins counting down. |

## 65.7.13 Timer Flag (TFLG0 - TFLG3)

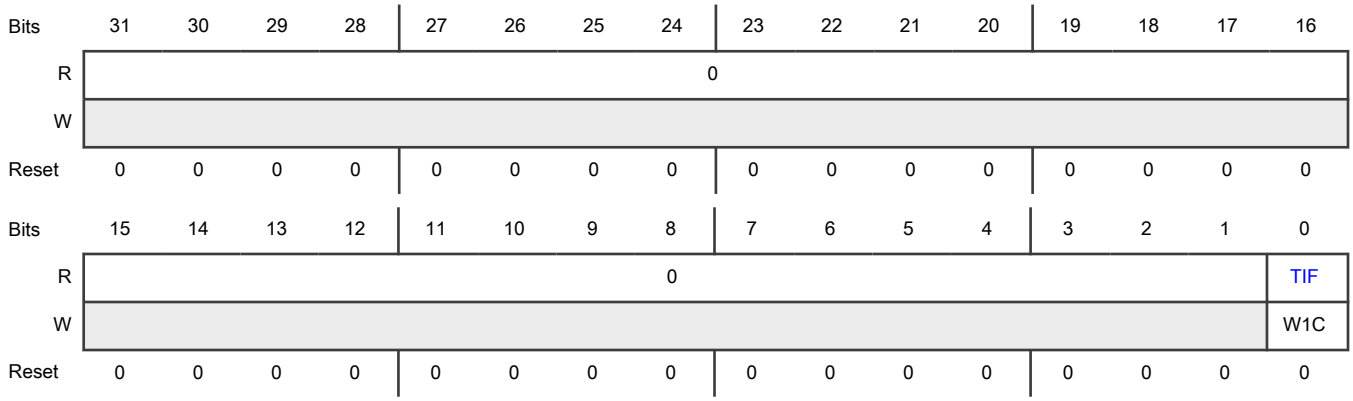
### Offset

| Register | Offset |
|---|---|
| TFLG0 | 10Ch |
| TFLG1 | 11Ch |
| TFLG2 | 12Ch |
| TFLG3 | 13Ch |

### Function

Indicates the PIT timer has expired.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 0<br>TIF | Timer Interrupt Flag<br>Indicates the timer period has expired (CVAL*n*[TVL] = 0). If interrupts are enabled (TCTRL*n*[TIE] = 1), TIF triggers an interrupt request.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Timer has not expired<br>1b - Timer expired<br>When writing<br>0b - No effect<br>1b - Clears flag |

## 65.8 Glossary

**RTI** Real-time interrupt

# Chapter 66
# Real Time Clock (RTC)

## 66.1 Chip-specific RTC information

### 66.1.1 RTC instances and configuration

The chip contains one instance of RTC (Real Time Clock) timer and API (Autonomous Periodic Interrupt) timer, where both can perform 32-bit comparisons.

The RTC is present in always ON domain, hence available in RUN mode as well as in STANDBY mode. Both RTC and API timers can generate interrupts as well as wakeup from low power modes.

RTC supports a resolution of 1µs with high speed internal oscillator clock (1/48MHz (FIRC) * 48 (RTC counter) = 1µs)

This chip supports seamless RTC operation across functional reset with the clock sources SIRC and SXOSC.

**NOTE**

SXOSC is not present in MCXE315/MCXE316, thus RTC will run from SIRC clock.

API has the capability to change the Timer compare value (APIVAL) independently without stopping the timer.

API can also be used in conjunction with the Comparator module. In STANDBY mode configuration, the API is configured to generate a START / NEXT type signal to inform the Comparator module that a 'compare' must be carried out.

**Table 360. RTC instances**

| Module | MCXE315/MCXE316/MCXE317/MCXE31B |
|---|---|
| RTC | Yes |

## 66.2 Overview

The Real-Time Clock (RTC) is a free-running counter used for time keeping applications. The RTC can be configured to generate an interrupt at a pre-defined interval independent of the mode of operation (run mode or low power mode). If in a low power mode, the RTC interval is reached, the RTC first generates a wakeup and then asserts the interrupt request. The RTC also supports an API function used to generate a periodic wakeup request to exit a low-power mode or an interrupt request.

### 66.2.1 Block diagram

The following figure shows clock gating for RTC clocks.

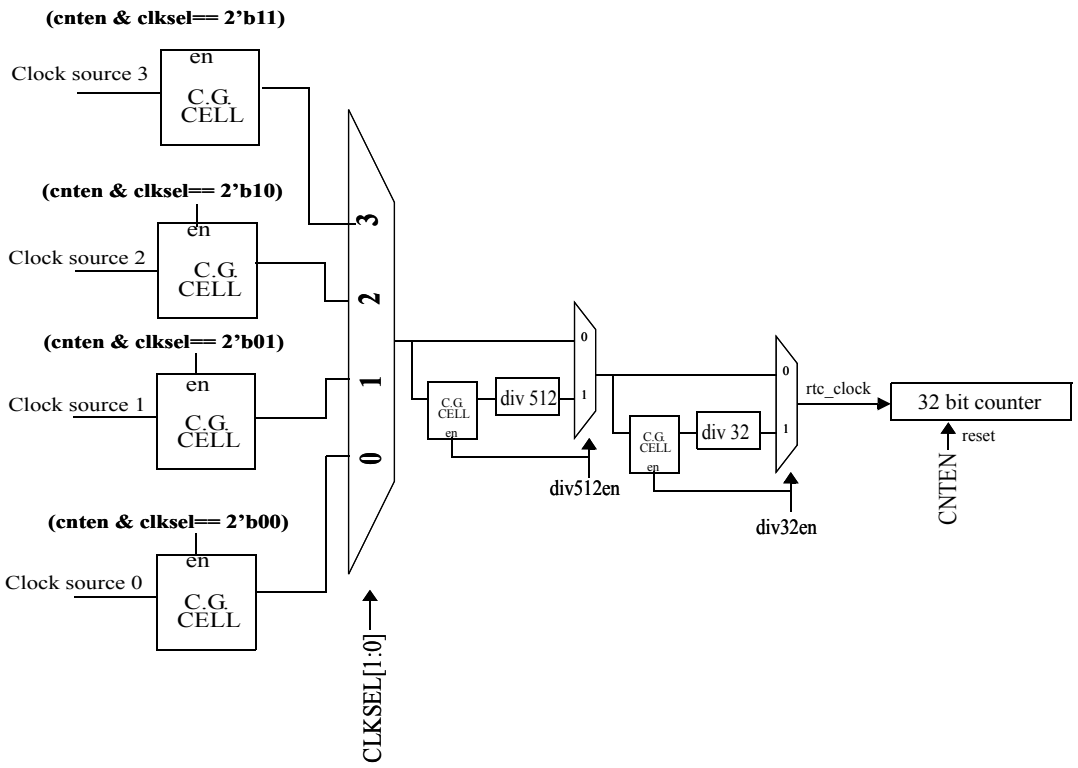



Figure 357. Clock gating for RTC clocks

The following figure shows the block diagram of RTC.

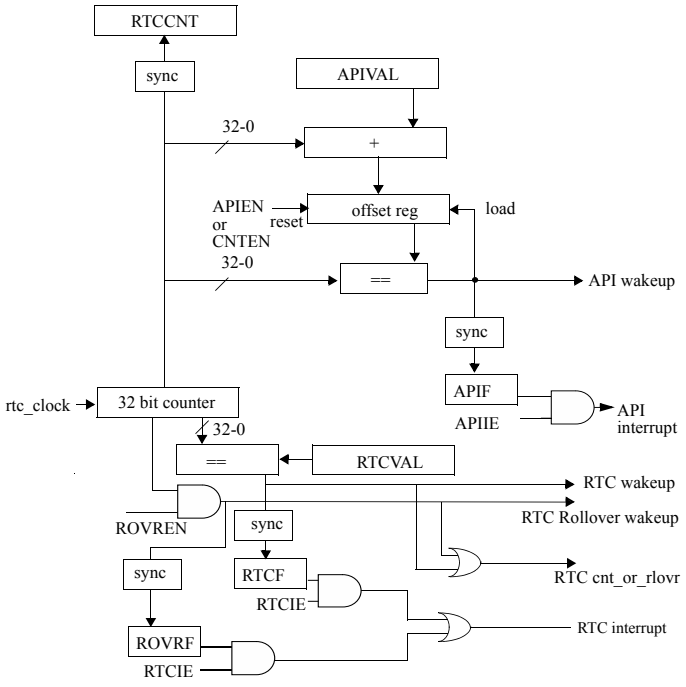



Figure 358. Block diagram

### 66.2.2 Features

RTC features include:

- 32-bit counter
- Selectable counter clock sources (IRCs and OSCs)
  - Clock source 0
  - Clock source 1
  - Clock source 2
  - Clock source 3

**NOTE**

See the chip-specific RTC information and clocking chapter for mapping of clock sources.

- Optional 512 prescaler and optional 32 prescaler to run the 32 bit counter.
- RTC interrupt with interrupt enable.
- Counter runs in all modes of operation.
- RTC counter is reset when the counter is disabled by software and by reset to RTC block.
- Autonomous periodic interrupt support includes:
  - 32-bit compare value to support range of wakeup intervals/interrupts
  - API logic has a separate enable to support changing compare value while RTC is running
  - API interrupt with interrupt enable
  - Operates in all modes
  - API compare value can be modified while RTC is running
- Optional interrupt for RTC match, API match, and RTC rollover.

## 66.3 Functional description

### 66.3.1 RTC

The RTC consists of a 32-bit free running counter enabled with the CNTEN bit (CNTEN, when negated, asynchronously resets the counter and synchronously enables the counter when enabled). After disabling CNTEN, RTCVAL, APIVAL, needs to be written again for desired functionality. The value of the counter may be read via the RTC Count (RTCCNT) register. Note that because of clock synchronization, the RTC Count (RTCCNT) value may represent a previous counter value. The difference between the counter and the read value depends on the ratio of counter clock and bus clock. Maximum possible difference between the two is 6 count values.

The clock source to the counter is selected with the CLKSEL field, which gives four options for clocking the RTC/API. The four clock sources are assumed to be on these:Clock source 0, Clock source 1, Clock source 2, and Clock source 3. The output of the clock mux can be optionally divided by a combination of 512 and 32 to give various count periods for different clock sources. Note that the CNTEN bit should be disabled when the RTC/API clock source is switched.

When the RTC Count (RTCCNT) counter value for counter bits 31-0 match the 32-bit value in the RTCVAL field, then the RTCF interrupt flag bit is set (after proper clock synchronization). If the RTCIE interrupt enable bit is set, then the RTC interrupt request is generated. RTC Compare value register (RTCVAL) register can be written only when INV_RTC bit is clear. Initially INV_RTC=0, and hence RTC Compare value register (RTCVAL) can be written once and hence INV_RTC gets set. This bit can now be cleared only by enabling the RTC counter. After the counter is enabled, RTC Compare value register (RTCVAL) can be written anytime, until RTC is disabled again. RTC Compare value register (RTCVAL) is first synchronized to the RTC clock domain, therefore, if RTC Compare value register (RTCVAL) is updated at the point where a counter match is due to happen in the next 2-3 RTC clocks because of previous RTC Compare value register (RTCVAL), the RTCF flag is set. However, if the RTC Compare value register

(RTCVAL) is updated at the point where no counter match is due as per the previous RTC Compare value register (RTCVAL), the RTCF flag is set when the counter matches the new RTC Compare value register (RTCVAL). If there is a match when in the low-power mode, then the RTC first generates a wakeup request to force a wakeup to run mode,and then the RTCF flag is set.

If RTC Compare value register (RTCVAL) is updated just after counter match, new RTC Compare value register (RTCVAL) value should not be within next 6 RTC counter values.

A rollover wakeup and/or interrupt can be generated when the RTC transitions from a count of 0xFFFF_FFFF to 0x0000_0000. The rollover flag is enabled by setting the ROVREN bit. If RTCIE is enabled, an interrupt request is generated for an RTC counter rollover. If system is in low power mode, RTC counter rollover with this bit causes a wakeup from the low-power mode.

Both rollover wakeup and RTC wakeup gets asynchronously de-asserted with disabling of CNTEN.

All the flags and counter values are synchronized with the Bus clock. It is assumed that the Bus clock and RTC clock selected through CLKSEL follows the below relation:

Bus clock >= (1.5 * RTC clock )/(div_factor)

- if both DIV32EN and DIV512EN bits are disabled, div_factor = 1
- if DIV32EN=1 and DIV512EN=0, div_factor = 32
- if DIV32EN=0 and DIV512EN=1, div_factor = 512
- if both DIV32EN and DIV512EN bits are enabled, div_factor = 512*32 = 16384

In case, RTC wakeup's are used as a wakeup source, bus clock (system clock or register interface clock of RTC) should be disabled (to save power in standby mode) after enabling the required wakeup and ensuring sufficient time gap (3-6 bus clock or RTC clock (rtc_clock) cycles, whichever is slower) between bus clock disabling and wakeup event. To ensure writing the register to enable the wakeup, the bus clock must be active, hence it should not be disabled before enabling wakeup. Correct operation is not guaranteed if the specification is not followed.

### 66.3.2 API functional description

Setting APIEN bit enables the autonomous interrupt function. The 32-bit APIVAL field selects the time interval for triggering an interrupt and/or wakeup event. Because the RTC is a free-running counter, the APIVAL is added to the current count to calculate an offset. When the counter reaches the offset count, an interrupt and/or wakeup request is generated. Then the offset value is recalculated and again re-triggers a new request when the new value is reached. API function is enabled only when CNTEN and APIEN bits are asserted and APIVAL is non-zero. Also APIVAL can be updated anytime. After APIVAL is updated, the first API interrupt is generated according to the previous value. From the second interrupt onwards, the API interrupt is generated with the new APIVAL. When a compare is reached, the APIF interrupt flag bit is set (after proper clock synchronization). If the APIIE interrupt enable bit is set, then the API interrupt request is generated. If there is a match while being in the low-power mode, then the API first generates a wakeup request to force a wakeup into normal operation, and then the APIF flag is set.

When the CNTEN is de-asserted, the API function is reset , though wakeup API is not asynchronously de-asserted with CNTEN. If APIEN is disabled when counter matches the offset APIF can be missed.

### 66.3.3 Modes of operation

#### 66.3.3.1 Functional mode

There are two functional modes of operation for RTC, normal operation and low-power mode. In normal operation, all RTC registers can be read or written . The RTC/API and associated interrupts are optionally enabled. In low-power mode, the bus interface is disabled. The RTC/API is enabled (if enabled prior to entry into low-power mode).

#### 66.3.3.2 Debug mode

On entering into the debug mode, the RTC counter freezes on the last valid count if the FRZEN is set. On exit from debug mode, counter continues from the frozen value.

## 66.4 Initialization

Recommended programming flow as follow

1. Program RTCVAL register with the value greater than 4 if RTCF related functionality is required.
2. Program APIVAL register with the value greater than 4 if API functionality is required.
3. Program all fields (as required) of RTC Control (RTCC) register with CNTEN=1. After setting CNTEN field, counter starts running and RTC/API functionality will be active as per the configurations.
4. If RTCVAL or APIVAL needs to be updated during run, check corresponding INV_RTC or INV_API bit is cleared before writing and do meet all the restrictions of the corresponding register.
5. If CLKSEL or DIV32EN or DIV512EN needs to be updated, first update RTC Control (RTCC) register with CNTEN=0 (when all INV_RTC or INV_API are 0) keeping other field values same. Wait for minimum 3 RTC clock cycles so that CNTEN will be synchronized to RTC clock domain. Then write the RTC Control (RTCC) register with new configuration required with CNTEN=1.

## 66.5 RTC register descriptions

The RTC registers are listed in this section.

**NOTE**

Address offset - 0x18h should not be accessed by application as corresponding feature/s are not available. Therefore, transfer error will not be generated at this offset.

**NOTE**

XFR error will be generated when RTCSUPV is accessed in user mode, any other register is accessed in user mode when SUPV bit is set, write attempt is made for RTCCNT register and any register accessed out of address range.

### 66.5.1 RTC memory map

RTC base address: 4028_8000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | RTC Supervisor control register (RTCSUPV) | 32 | RW | 8000_0000h |
| 4h | RTC Control (RTCC) | 32 | RW | 0000_0000h |
| 8h | RTC Status register (RTCS) | 32 | RW | 0000_0000h |
| Ch | RTC Count (RTCCNT) | 32 | R | 0000_0000h |
| 10h | API Compare value register (APIVAL) | 32 | RW | 0000_0000h |
| 14h | RTC Compare value register (RTCVAL) | 32 | RW | 0000_0000h |

### 66.5.2 RTC Supervisor control register (RTCSUPV)

**Offset**

| Register | Offset |
|---|---|
| RTCSUPV | 0h |

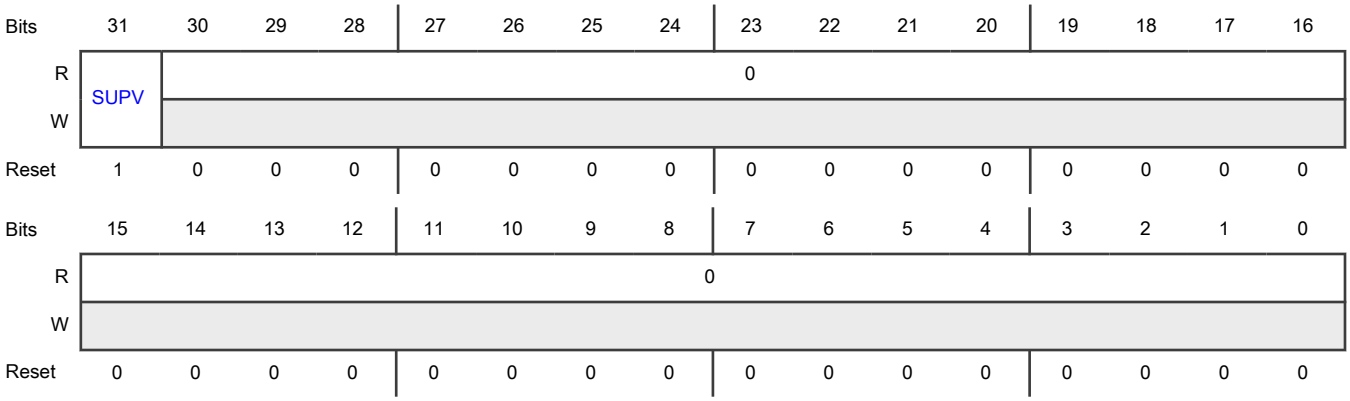
### Function

The RTCSUPV register contains the SUPV bit that determines whether other registers are accessible in supervisor mode or user mode.

**NOTE**

You can access this register only in Supervisor mode, and you must write a value only to the SUPV field of the register.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>SUPV | RTC Supervisor Bit<br>0b - All registers are accessible in both user as well as supervisor mode<br>1b - All other registers are accessible in the supervisor mode only |
| 30-0<br>— | Reserved |

## 66.5.3 RTC Control (RTCC)

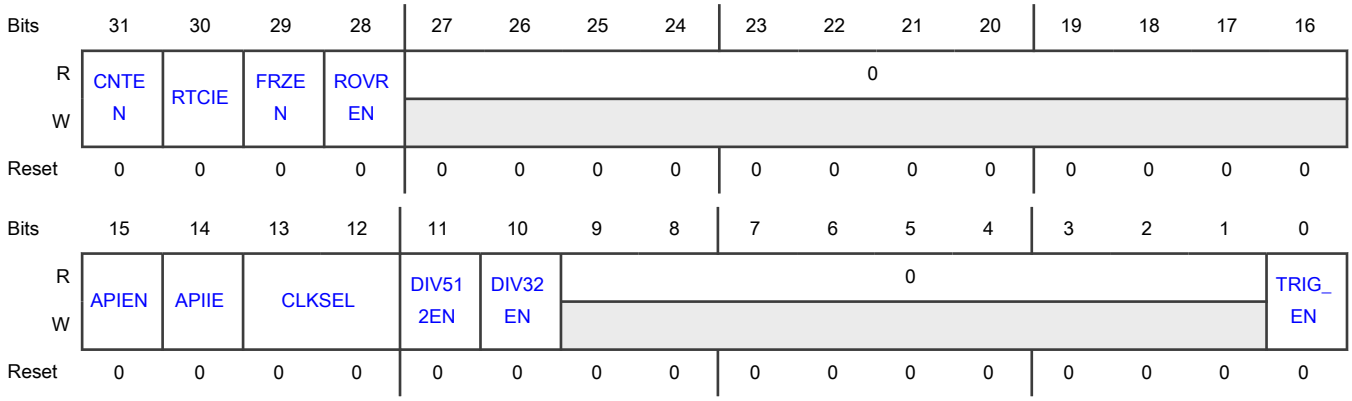
### Offset

| Register | Offset |
|---|---|
| RTCC | 4h |

### Function

RTC Control register

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>CNTEN | Counter Enable<br>The CNTEN bit enables the RTC counter. Setting CNTEN bit to 0b has the effect of asynchronously resetting (synchronous reset negation) all the RTC and API logic. This allows RTC configuration and clock source selection to be updated without causing synchronization issues.<br>**NOTE**<br>CNTEN should be disabled when INV_RTC, INV_API are cleared.<br>0b - Counter disabled<br>1b - Counter enabled |
| 30<br>RTCIE | RTC Interrupt Enable<br>The RTCIE bit enables interrupts requests to the system if RTCF is asserted.<br>0b - RTC interrupts disabled<br>1b - RTC interrupts enabled |
| 29<br>FRZEN | Freeze Enable Bit<br>The counter freezes on entering the debug mode on the last valid count value if the FRZEN bit is set. After passing of the debug mode counter starts from the frozen value. This bit should not be changed when debug mode is enabled.<br>0b - Counter does not freeze in debug mode<br>1b - Counter freezes in debug mode |
| 28<br>ROVREN | Counter Roll Over wakeup/Interrupt Enable<br>The ROVREN bit enables wakeup and interrupt requests when the RTC has rolled over from 0xFFFF_FFFF to 0x0000_0000. The RTCIE bit must also be set in order to generate an interrupt from a counter rollover.<br>0b - RTC rollover wakeup/interrupt disabled<br>1b - RTC rollover wakeup/interrupt enabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 27-16<br>— | Reserved |
| 15<br>APIEN | Autonomous Periodic Interrupt Enable<br>The APIEN bit enables the autonomous periodic interrupt function. Setting this bit to 0b, asynchronously disables API wakeup output of RTC as well.<br>0b - API disabled<br>1b - API enabled |
| 14<br>APIIE | API Interrupt Enable<br>The APIIE bit enables interrupts requests to the system if APIF is asserted.<br>0b - API interrupts disabled<br>1b - API interrupts enabled |
| 13-12<br>CLKSEL | Clock select<br>The CLKSEL[1:0] bits select the clock source for the RTC. CLKSEL may only be updated when CNTEN is 0. The user should ensure that oscillator is enabled before selecting it as a clock source for RTC.<br>**NOTE**<br>See the RTC-clocking and resolution information in the clocking chapter elsewhere in this document.<br>00b - Clock source 0<br>01b - Clock source 1<br>10b - Clock source 2<br>11b - Clock source 3 |
| 11<br>DIV512EN | Divide by 512 enable<br>The DIV512EN bit enables the 512 clock divider. DIV512EN may only be updated when CNTEN is 0.<br>0b - Divide by 512 is disabled<br>1b - Divide by 512 is enabled |
| 10<br>DIV32EN | Divide by 32 enable<br>The DIV32EN bit enables the 32 clock divider. DIV32EN may only be updated when CNTEN is 0.<br>0b - Divide by 32 is disabled<br>1b - Divide by 32 is enabled |
| 9-1<br>— | Reserved |
| 0 | Trigger enable for Analog Comparator |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| TRIG_EN | This bit (TRIG_EN) when set will de-assert the wakeup_api on the next RTC clock (required for CMP when bus clock is disabled).<br>**NOTE**<br>If API wakeup is asserted with TRIG_EN set, and bus clock enabled, setting of RTCS[APIF] may be missed. |

## 66.5.4 RTC Status register (RTCS)

### Offset

| Register | Offset |
|---|---|
| RTCS | 8h |

### Function

**NOTE**

W1C has priority over setting of the RTCF, APIF and ROVRF bits, in case both clearing and setting occurs at the same time.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | RTCF | | 0 | | | | | | | | | INV_RTC | INV_API | Reserved |
| W | | | W1C | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | APIF | | 0 | | ROVRF | | 0 | | | | | | | |
| W | | | W1C | | | | W1C | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29 | RTC Interrupt Flag |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| RTCF | The RTCF bit indicates that the RTC counter has reached the counter value matching RTC Compare value register (RTCVAL). RTCF is cleared by writing a 1 to RTCF. Writing a 0 to RTCF has no effect.<br>0b - RTC counter is not equal to RTCVAL<br>1b - RTC counter matches RTCVAL |
| 28-19<br>— | Reserved |
| 18<br>INV_RTC | Invalid RTC write<br>This bit returns value 1 after a value is written to the RTCVAL register and the synchronization process is in progress. During this synchronization period, any attempt to write to the RTCVAL register again is ignored. Synchronization will complete only when CNTEN is set. |
| 17<br>INV_API | Invalid APIVAL write<br>This bit returns value 1 after a value is written to the APIVAL register and the synchronization process is in progress. During this synchronization period, any attempt to write to the APIVAL register again is ignored. Synchronization will complete only when CNTEN is set. |
| 16<br>— | Reserved |
| 15-14<br>— | Reserved |
| 13<br>APIF | API Interrupt Flag<br>The APIF bit indicates that the RTC counter has reached the counter value matching API offset value. APIF is cleared by writing a 1 to APIF. Writing a 0 to APIF has no effect.<br>0b - Counter is not equal to API offset value<br>1b - Counter matches the API offset value |
| 12-11<br>— | Reserved |
| 10<br>ROVRF | Counter Roll Over Interrupt Flag<br>The ROVRF bit indicates that the RTC has rolled over from 0xFFFF_FFFF to 0x0000_0000. ROVRF is cleared by writing a 1 to ROVRF.<br>0b - RTC has not rolled over<br>1b - RTC has rolled over |
| 9-0<br>— | Reserved |

## 66.5.5 RTC Count (RTCCNT)

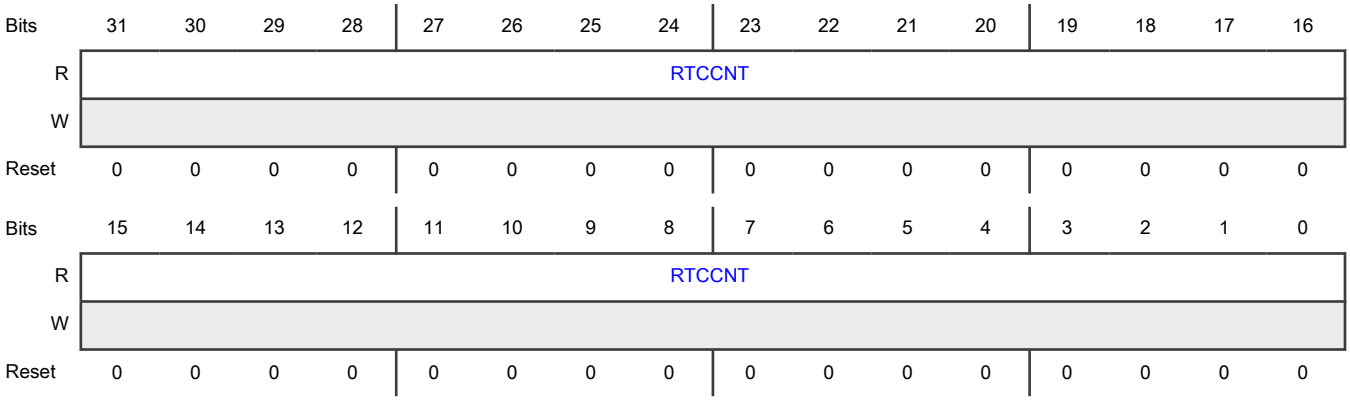
### Offset

| Register | Offset |
|---|---|
| RTCCNT | Ch |

### Function

RTC Counter register

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>RTCCNT | RTC Counter Value<br>Because of clock synchronization, the RTCCNT value may represent a previous counter value. |

## 66.5.6 API Compare value register (APIVAL)

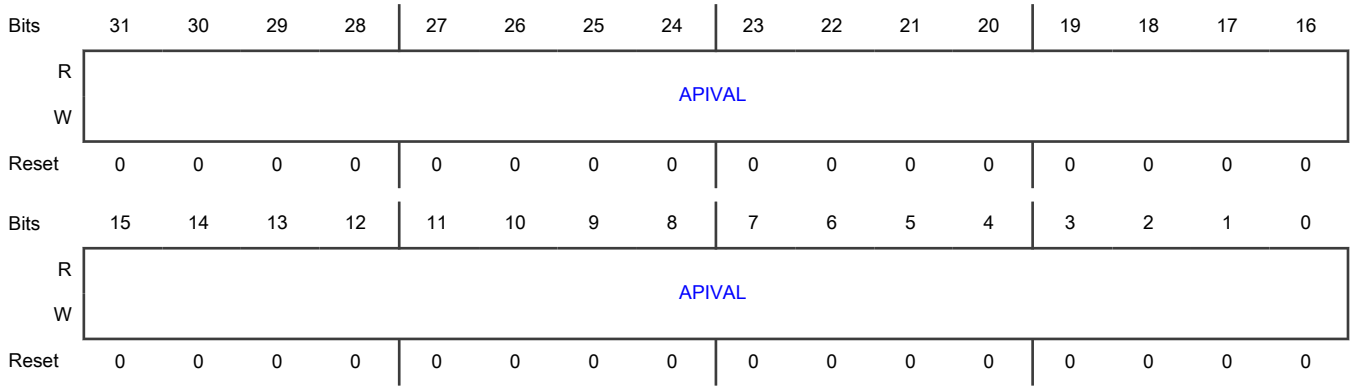
### Offset

| Register | Offset |
|---|---|
| APIVAL | 10h |

### Function

The APIVAL offset bits are compared to the RTC counter bits and if a match occurs, an interrupt/wakeup request is asserted.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>APIVAL | API Compare Value<br>APIVAL bits are added to the current count to calculate an offset. The APIVAL offset bits are compared to the RTC counter bits and if a match occurs, an interrupt/wakeup request is asserted.<br>**NOTE**<br>API functionality is active only when APIVAL is non zero. The first API interrupt takes two more cycles because of synchronization of APIVAL to RTC clock. After that, interrupts are periodic in nature and it takes APIVAL+1 cycles. The Minimum supported value of APIVAL is 4. This is because of synchronization. |

## 66.5.7 RTC Compare value register (RTCVAL)

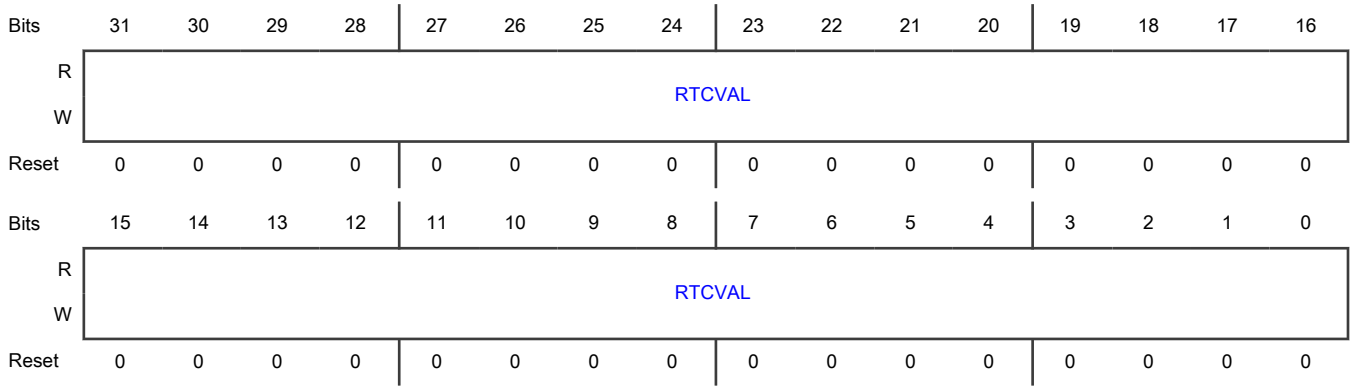
### Offset

| Register | Offset |
|---|---|
| RTCVAL | 14h |

### Function

The RTCVAL bits are compared to the RTC counter bits and if a match occurs, RTCF is set. The minimum value of RTCVAL should be 4.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>RTCVAL | RTC Compare Value<br>The RTCVAL bits are compared to the RTC counter bits and if a match occurs, RTCF is set. |

## 66.6 Glossary

| | |
|---|---|
| **API** | Autonomous Periodic Interrupt |
| **CMP** | Comparator |
| **IRCs** | Internal RC Oscillator |
| **OSCs** | Oscillator |

# Chapter 67
# Low Power Serial Peripheral Interface (LPSPI)

## 67.1 Chip-specific LPSPI information

### 67.1.1 LPSPI instances and configuration

Table 361. LPSPI instances

| Instance | MCXE31B | MCXE315/MCXE316/MCXE317 |
|---|---|---|
| LPSPI0 | Yes | Yes |
| LPSPI1 | Yes | Yes |
| LPSPI2 | Yes | Yes |
| LPSPI3 | Yes | Yes |
| LPSPI4 | Yes | No |
| LPSPI5 | Yes | No |

Table 362. LPSPI instances configuration

| Instances | TX FIFO Size | RX FIFO Size | Chip Selects |
|---|---|---|---|
| LPSPI0 | 4x32 bit | 4x32 bit | 8 |
| LPSPI1 | 4x32 bit | 4x32 bit | 6 |
| LPSPI2 | 4x32 bit | 4x32 bit | 4 |
| LPSPI3 | 4x32 bit | 4x32 bit | 4 |
| LPSPI4 | 4x32 bit | 4x32 bit | 4 |
| LPSPI5 | 4x32 bit | 4x32 bit | 4 |

- MCXE31B supports data rate up to 20 Mbps, MCXE315/MCXE316/MCXE317 supports data rate up to 15 Mbps. Only one LPSPI instance support 20 MHz in loopback mode that is LPSPI0
- All LPSPIs supports a minimum a 10 MHz data rate on standard plus pads and 7.5 MHz on standard pads.
- Low leakage and Wait modes are not supported in this device.

The number of chip selects that each LPSPI instance supports varies. Because of that, in the Configuration Register 1 (CFGR1), the Peripheral Chip Select (PCS) field and the Peripheral Chip Select Polarity (PCSPOL) field also vary. See the next 2 tables.

Table 363. LPSPI instances mapped against Peripheral Chip Select (PCS) field support

| PCS supported in | PCS not supported in |
|---|---|
| LPSPI0_TCR[26-24] | — |
| LPSPI1_TCR[26-24] | — |
| LPSPI2_TCR[25–24] | LPSPI2_TCR[26] |
| LPSPI3_TCR[25–24] | LPSPI3_TCR[26] |
| LPSPI4_TCR[25–24] | LPSPI4_TCR[26] |

*Table continues on the next page...*

Table 363. LPSPI instances mapped against Peripheral Chip Select (PCS) field support (continued)

| PCS supported in | PCS not supported in |
|---|---|
| LPSPI5_TCR[25–24] | LPSPI5_TCR[26] |

Table 364. LPSPI instances mapped against Peripheral Chip Select Polarity (PCSPOL) field support

| PCSPOL supported in | PCSPOL not supported in |
|---|---|
| LPSPI0_CFGR1[15-8] | — |
| LPSPI1_CFGR1[13–8] | LPSPI1_CFGR1[15–14] |
| LPSPI2_CFGR1[11–8] | LPSPI2_CFGR1[15–12] |
| LPSPI3_CFGR1[11–8] | LPSPI3_CFGR1[15–12] |
| LPSPI4_CFGR1[11–8] | LPSPI4_CFGR1[15–12] |
| LPSPI5_CFGR1[11–8] | LPSPI5_CFGR1[15–12] |

### 67.1.2 LPSPI HREQ considerations for MCXE31B

It is recommended that the HREQ pin (when PCS[1] is used as HREQ) should get de-asserted before the completion of frame transfer. In case if the HREQ state is still asserted and LPSPI returns to idle state after frame transfer completion, the HREQ state is internally latched and the next data is written to the transmit FIFO without waiting for the next HREQ assertion.

This limitation is present only while using PCS[1] as HREQ and HREQ remains asserted throughout frame transfer. In case of using trigger input as HREQ, there is no issue.

## 67.2 Overview

LPSPI provides an efficient interface (either as a controller or peripheral) to an SPI bus, which is a synchronous serial communication interface used in embedded systems. It is typically used to perform short distance communications between microcontrollers and peripheral devices, on printed circuit boards. Typical applications include interfacing with secure digital cards and LCD displays.

**NOTE**

The terminology in this chapter has been updated to align with NXP's inclusive language standards, as shown in the table below.

Table 365. Updated terms

| Updated term | Deprecated term |
|---|---|
| Controller | Master |
| Peripheral | Slave |

### 67.2.1 Block diagram

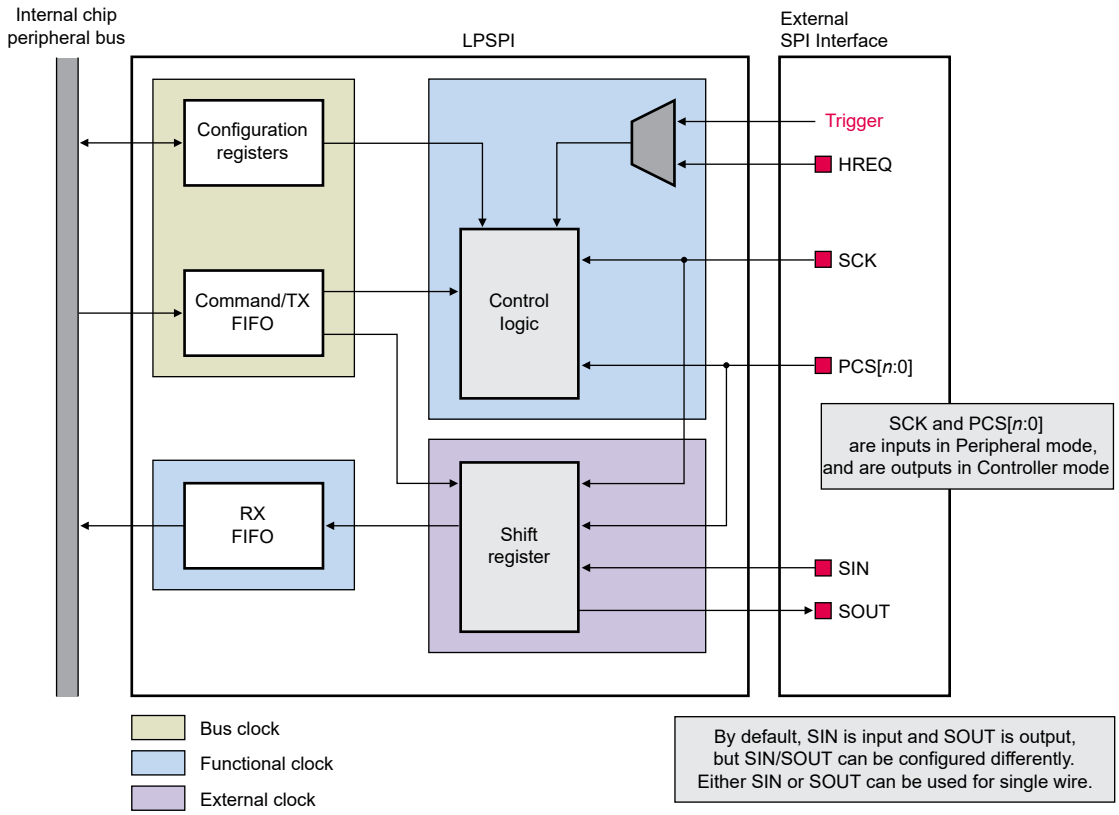



Figure 359. Block diagram

### 67.2.2 Features

- Minimal CPU overhead, with DMA transmit and receive requests supporting FIFO register accesses
- Support available for 32-bit word size
- Configurable clock polarity and phase
- Support available for 8 peripheral chip selects in Controller mode
- Support available for Peripheral mode
- 4-word transmit and command FIFO
- 4-word receive FIFO
- Flexible timing parameters in Controller mode, including SCK frequency and duty cycle, and delays between PCS and SCK edges
- Continuous transfer option to keep PCS asserted across multiple frames
- Full-duplex transfers that support 1-bit transmit and receive on each clock edge
- Half-duplex transfers that support:
  - 1-bit transmit or receive on each clock edge
  - 2-bit transmit or receive on each clock edge
  - 4-bit transmit or receive on each clock edge
  - 8-bit transmit or receive on each clock edge

- Option to use host request to control the start of an SPI bus transfer

## 67.3 Functional description

### 67.3.1 Controller mode

#### 67.3.1.1 Transmit and command FIFO commands

The transmit and command FIFO is a combined FIFO that includes both transmit data words and command words. You store:

- Transmit data words in the transmit and command FIFO, by writing to Transmit Data (TDR).
- Command words in the transmit and command FIFO, by writing to Transmit Command (TCR).

When a command word is at the top of the transmit and command FIFO, the actions that can occur depend on whether LPSPI is busy or between frames (see TCR[CONT] and TCR[CONTC]). See Table 366 for conditions and possible corresponding actions when a command word is at the top of the transmit and command FIFO.

**Table 366. Possible actions when a command word is at the top of the transmit and command FIFO**

| Condition | Action |
|---|---|
| LPSPI is enabled and idle. | The command word is pulled from the FIFO, and this command word controls all subsequent transfers. |
| LPSPI is busy and TCR[CONTC] is 0. | The SPI frame completes at the end of the existing word, ignoring TCR[FRAMESZ]. The command word is then pulled from the FIFO and that command word controls all subsequent transfers (or until the next update to the command word). Note that a command word with TCR[CONTC] = 0 always terminates the existing transfer regardless of the previous TCR[CONT] value. |
| LPSPI is busy; the existing TCR[CONT] value is 1 and the new TCR[CONTC] value is 1. | The command word must be updated at the frame boundary. The command word is pulled from the FIFO during the last SCK pulse of the existing frame (based on the value of FRAMESZ), and the frame continues using the new command value for the rest of the frame (or until the next update to the command word). When TCR[CONTC] = 1, only the lower 24 bits of the command word are updated. If the command word is updated at a word boundary, then the transfer halts (stops) after that word. TCR[CONTC] is ignored when not at a frame boundary, so the frame ends prematurely. |

TCR[CONT] = 1 keeps PCS asserted at end of frame, allowing the transfer to continue.

TCR[CONTC] = 1 specifies that this command word must not terminate the existing frame, and the transfer can continue using the new command word.

TCR[CONTC] = 1 is restricted in the sense that the new command must load on a frame boundary, and the only way for a transfer to continue from a frame boundary is when the previous command has TCR[CONT] = 1.

You can read the current state of the existing command word from Transmit Command (TCR). It requires at least three LPSPI functional clock cycles for Transmit Command (TCR) to update after you write to it (assuming an empty FIFO), and LPSPI must be enabled (CR[MEN] = 1).

Writing to Transmit Command (TCR) does not initiate an SPI bus transfer, unless TCR[TXMSK] = 1. When TCR[TXMSK] = 1, a new command word is not loaded until the end of the existing frame (based on the value of TCR[FRAMESZ]); at the end of the transfer, TCR[TXMSK] transitions to 0.

In Controller mode, the LPSPI command word in Transmit Command (TCR) controls SPI attributes based on the selections in register fields. See Table 367 for TCR fields and associated functionality related to data transfer.

Table 367. Command word in Controller mode

| Transmit Command (TCR) | | Description | Can this field be modified during a data transfer? |
|---|---|---|---|
| Field | Name | | |
| CPOL | Clock polarity | Specifies the polarity of the SCK pin. Any change of CPOL value causes a transition on the SCK pin. | N |
| CPHA | Clock phase | Specifies the clock phase of the transfer. | N |
| PRESCALE | Prescaler value | Specifies a prescaler used to divide the LPSPI functional clock, to generate the timing parameters of the SPI bus transfer. Changing PRESCALE in conjunction with PCS enables LPSPI to connect to different peripheral devices at different frequencies. | N |
| PCS | Peripheral chip select | Specifies which PCS pin asserts for the transfer; the polarity of PCS is static and specified by CFGR1[PCSPOL].<br>If CFGR1[PCSCFG] = 1, do not select PCS[3:2]. | N |
| LSBF | LSB first | Specifies whether LSB (bit 0) or MSB (bit 31 for a 32-bit word) is transmitted or received first. | Y |
| BYSW | Byte swap | Enables byte swap on each 32-bit word when transmitting and receiving data. Byte swapping can be useful when interfacing with devices that organize data as big-endian. | Y |
| CONT | Continuous transfer | Configures LPSPI for a continuous transfer that keeps PCS asserted between frames (as specified by FRAMESZ). You must write a new command word to cause PCS to negate. Also, this field supports changing the command word at frame size boundaries. | Y |
| CONTC | Continuing command | Indicates that this is a new command word for the existing continuous transfer. When CONTC = 1, the command word must only be written to the transmit and command FIFO on a frame boundary. | Y |
| RXMSK | Receive data mask | Masks the receive data and does not store the masked receive data in the receive FIFO or perform receive data matching. This option is useful for half-duplex transfers or to specify which fields are compared during receive data matching. | Y |
| TXMSK | Transmit data mask | Masks the transmit data; masked transmit data is not pulled from the transmit FIFO, and the output data pin is 3-stated (unless otherwise configured by CFGR1[OUTCFG]). This option is useful for half-duplex transfers. | Y |
| WIDTH | Transfer width | Specifies the number of bits shifted on each SCK pulse:<br>• 1-bit transfers support traditional SPI bus transfers in either half-duplex or full-duplex data formats.<br>• 2-bit and 4-bit half-duplex transfers are useful for interfacing with QuadSPI memory devices, and either TXMSK or RXMSK must also be 1.<br>• 8-bit half-duplex transfers are useful for interfacing with OctelSPI memory devices, and either TXMSK or RXMSK must also be 1. | Y |

*Table continues on the next page...*

Table 367. Command word in Controller mode (continued)

| Transmit Command (TCR) | | Description | Can this field be modified during a data transfer? |
|---|---|---|---|
| Field | Name | | |
| FRAMESZ | Frame size | Configures the frame size in number of bits equal to (FRAMESZ + 1):<br>• The minimum frame size is 8 bits, or 16 bits for an 8-bit transfer.<br>• If the frame size is larger than 32 bits, then the frame is divided into multiple words of 32 bits; each word is loaded from the transmit FIFO and stored in the receive FIFO separately.<br>• If the size of the frame is not divisible by 32, then the last load of the transmit FIFO and store of the receive FIFO contains the remaining bits. For example, a 72-bit transfer consists of three words: the first and second words are 32 bits, and the third word is 8 bits. | Y |

#### 67.3.1.1.1 SPI bus transfers

LPSPI initiates an SPI bus transfer when all these conditions are true:

- Data is written to the transmit FIFO.
- The HREQ pin is asserted (or the HREQ function is disabled).
- LPSPI is enabled.

To perform the SPI bus transfer, LPSPI uses the attributes configured in Transmit Command (TCR) and the timing parameters defined in Clock Configuration (CCR).

The SPI bus transfer ends after the number of bits indicated by the value of FRAMESZ have been transferred (provided CONT = 0), or at the end of a word when a new transmit command word is at the top of the transmit and command FIFO. When LPSPI is disabled, the SPI bus transfers end after the transmit FIFO is empty and LPSPI is idle.

The HREQ input is only checked when PCS is negated.

#### 67.3.1.1.2 Circular FIFO

The transmit and command FIFO supports a circular FIFO feature. This feature enables the LPSPI controller to (periodically) repeat a short data transfer that fits within the transmit and command FIFO, without requiring additional FIFO accesses. When the circular FIFO is enabled (CFGR0[CIRFIFO] = 1), the current state of the FIFO read pointer is saved and the status flags are not updated. After the FIFO is empty and LPSPI is idle, the FIFO read pointer is restored with the saved version, so the contents of the transmit and command FIFO are not permanently pulled from the FIFO when Circular FIFO mode is enabled.

### 67.3.1.2 Receive FIFO and data match

The receive FIFO stores received data during SPI bus transfers. When TCR[RXMSK] = 1, the received data is discarded instead of being stored in the receive FIFO:

- Received data is written to the receive FIFO when the last bit of the word is sampled.
- If the transmit FIFO is empty during a multiple-word or continuous transfer, then the receive data is written to the receive FIFO before the transfer stalls (assuming CFGR1[NOSTALL] = 0) while waiting for new transmit data or for a command word to be written.

LPSPI provides a receive data match function that can match received data against one of the two words in DMR0 and DMR1, or against a masked data word. You can also configure the received data match function to compare only the first one or two received data words since the start of the frame:

- Received data that is already discarded because of TCR[RXMSK] cannot cause the data match flag to set, and delays the receive data match on the first received data word, until all discarded data is received.
- You can configure the receive data match function to discard all received data until a data match is detected, using CFGR0[RDMO].
- After a receive data match, to allow all subsequent data to be received, write 0 to CFGR0[RDMO], and then write 0 to SR[DMF].

#### 67.3.1.3 Timing parameters

The timing parameters that are used for all SPI bus transfers are relative to the LPSPI functional clock divided by the selection specified in TCR[PRESCALE]. Although you can only change Clock Configuration (CCR) when LPSPI is disabled (CR[MEN] = 0), to support interfacing with different peripheral devices at different frequencies, you can change the TCR[PRESCALE] selection between SPI bus transfers by using Transmit Command (TCR).

**NOTE**

The minimum value shown in Table 368 is the minimum counter value, but the values of Clock Configuration (CCR) must also satisfy the data sheet specs based on the LPSPI functional clock frequency and prescaler value.

**Table 368. Timing parameters**

| Clock Configuration (CCR)<br>Clock Configuration 1 (CCR1) | | Description | Minimum value | Maximum value |
|---|---|---|---|---|
| Field | Name | | | |
| SCKSET | SCK setup phase | Configures the SCK setup phase to (SCKSET + 1) cycles. The setup phase is the SCK high period when either CPHA = 0 and CPOL = 1, or CPHA = 1 and CPOL = 0. Otherwise, it is the SCK low period. The SCK period is defined as (SCKSET + SCKHLD + 2) and the duty cycle is the difference between SCKSET and SCKHLD. | 0 (1 cycle) | 255 (256 cycles) |
| SCKHLD | SCK hold phase | Configures the SCK hold phase to (SCKHLD + 1) cycles. The hold phase is the SCK low period when either CPHA = 0 and CPOL = 1, or CPHA = 1 and CPOL = 0. Otherwise, it is the SCK high period. The SCK period is defined as (SCKSET + SCKHLD + 2) and the duty cycle is the difference between SCKSET and SCKHLD. | 0 (1 cycle) | 255 (256 cycles) |
| PCSPCS | PCS-to-PCS delay | Configures the minimum delay between PCS negation and the next PCS assertion to (PCSPCS + PCSPCS + 2) cycles. When the command word is updated between transfers, there is a minimum of (PCSPCS + 1) cycles between the command word update and any change on PCS pins. | 0 (2 cycles) | 255 (512 cycles) |
| SCKSCK | SCK-to-SCK delay | Configures the delay during a continuous transfer between the last SCK edge of a frame and the first SCK edge of the continuing frame to (SCKSCK + 1) cycles. This is useful when the external peripheral | 0 (1 cycle) | 255 (256 cycles) |

*Table continues on the next page...*

Table 368. Timing parameters (continued)

| Clock Configuration (CCR) Clock Configuration 1 (CCR1) | | Description | Minimum value | Maximum value |
|---|---|---|---|---|
| Field | Name | | | |
| | | requires a large delay between different words of an SPI bus transfer. | | |
| PCSSCK | PCS-to-SCK delay | Configures the minimum delay between PCS assertion and the first SCK edge to (PCSSCK + 1) cycles. | 0 (1 cycle) | 255 (256 cycles) |
| SCKPCS | SCK-to-PCS delay | Configures the minimum delay between the last SCK edge and the PCS negation to (SCKPCS + 1) cycles. | 0 (1 cycle) | 255 (256 cycles) |

Figure 360 shows the timing settings controlled by:

- TCR[CPHA]
- TCR[CPOL]
- CCR[SCKPCS]
- CCR[PCSSCK]
- CCR1[SCKSET]
- CCR1[SCKHLD]

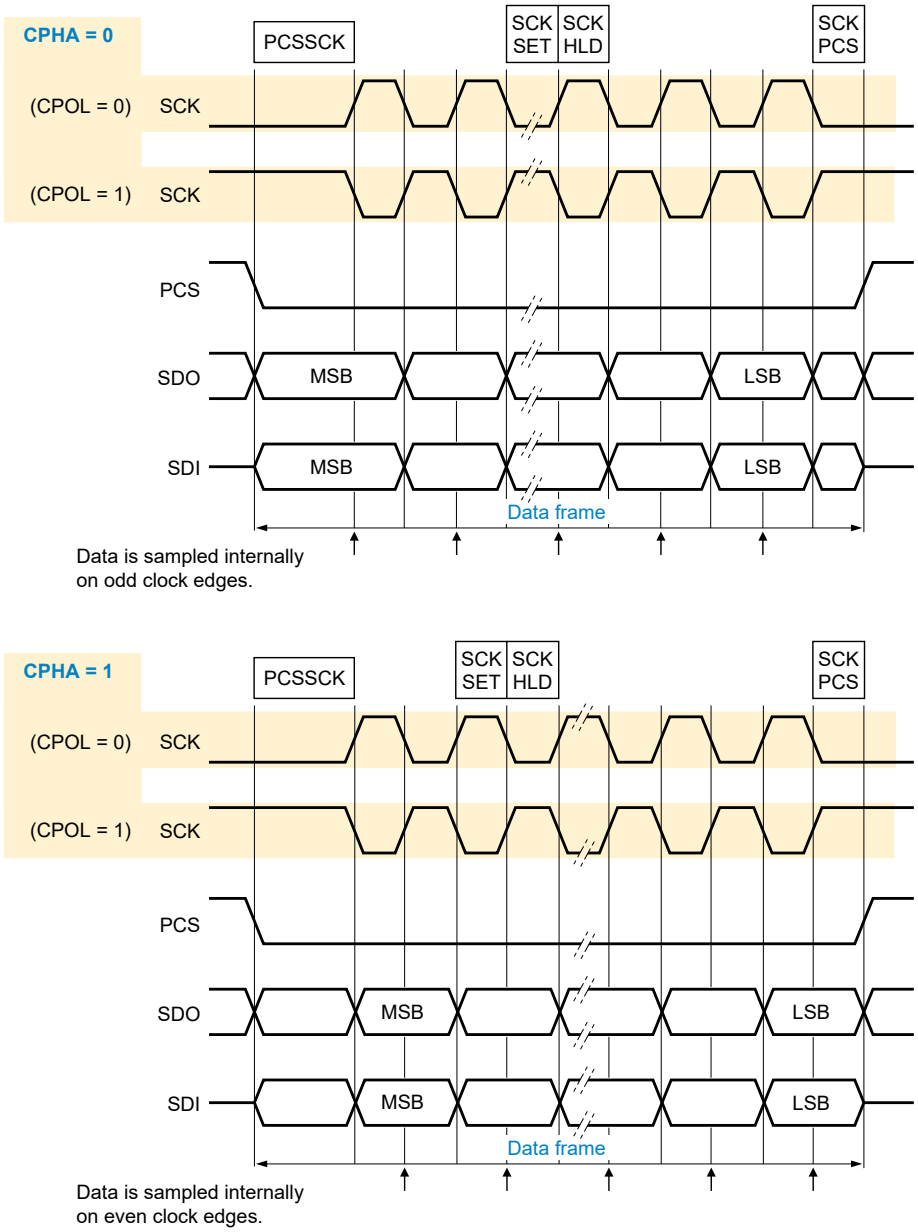



**Figure 360. Clock phase (TCR[CPHA]) timing diagram example**

To configure for a baud rate of 10 MHz with 50/50 duty cycle and with a functional clock frequency of 100 MHz, use the following settings:

- CCR1[SCKSET] = 0x4 (5 cycles)
- CCR1[SCKHLD] = 0x4 (5 cycles)
- CCR1[PCSPCS] = 0x8 (10 cycles)
- CCR1[SCKSCK] = 0x4 (5 cycles)
- CRR[PCSSCK] = 0x4 (5 cycles)
- CRR[SCKPCS] = 0x4 (5 cycles)
- TCR[PRESCALE] = 0x0 (divide by 1)

#### 67.3.1.4 Pin configuration

Following are the pin configuration settings for half-duplex transfers:

- To swap directions or to support half-duplex transfers on the same pin, you can configure the SIN and SOUT pins using CFGR1[PINCFG].
- To specify whether an output data pin (SOUT, for example) 3-states when PCS is negated, or if the output data pin retains the last value, use CFGR1[OUTCFG].
- When configuring half-duplex transfers, you must configure the output data pins to 3-state when PCS is negated (CFGR1[OUTCFG] = 1).
- When performing half-duplex 2-bit transfers, you can write any value to CFGR1[PCSCFG].
- When performing half-duplex 4-bit transfers, you must write 1h to CFGR1[PCSCFG].

#### 67.3.1.5 Clock loopback

Configure the LPSPI controller to use one of the following clocks to sample the input data:

- The SCK output clock
- A delayed version of the SCK output clock

The delayed version of the SCK is chosen by the SCK pin output delay, plus the SCK pin input delay, and is selected by writing 1 to CFGR1[SAMPLE]. Enabling the loopback version of the SCK pin can improve the setup time of the input data from the peripheral.

See the chip data sheet for the specific input setup time in Controller Loopback mode.

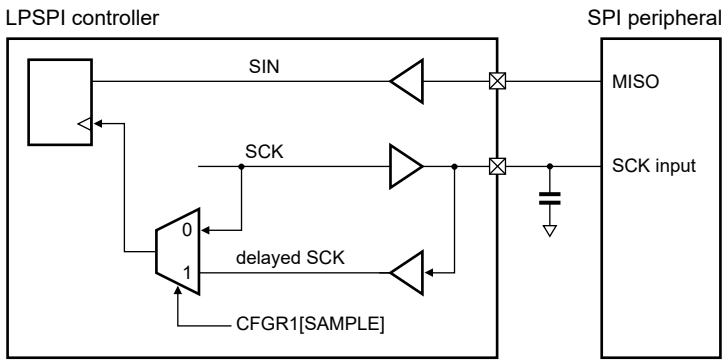



**Figure 361. Clock loopback**

### 67.3.2 Peripheral mode

LPSPI Peripheral mode:

- Uses the same shift register and logic that Controller mode uses.
- Does not use Clock Configuration (CCR).
- Requires Transmit Command (TCR) to remain static (unchanged) during SPI bus transfers.

#### 67.3.2.1 Transmit and command FIFO commands

You must initialize Transmit Command (TCR) before enabling LPSPI in Peripheral mode, although this register is not updated until after LPSPI is enabled. After LPSPI is enabled, you must make changes to this register only when LPSPI is idle. In Peripheral mode, the LPSPI command word in this register controls SPI attributes. Before the PCS input asserts, the transmit FIFO must be filled with transmit data, or the transmit error flag sets.

Table 369. Command word in Peripheral mode

| Transmit Command (TCR) | | Description |
|---|---|---|
| Field | Name | |
| CPOL | Clock polarity | Specifies the polarity of the external SCK input. |
| CPHA | Clock phase | Specifies the clock phase of transfer. |
| PRESCALE | Prescaler value | Specifies the LPSPI functional clock prescaler. |
| PCS | Peripheral chip select | Specifies which PCS is used. The polarity of PCS is static and configured by CFGR1[PCSPOL].<br>If CFGR1[PCSCFG] is not equal to zero, then do not select the PCS[3:2] pins. |
| LSBF | LSB first | Specifies whether LSB (bit 0) or MSB (bit 31 for a 32-bit word) is transmitted or received first. |
| BYSW | Byte swap | Enables byte swap on each 32-bit word when transmitting and receiving data. Byte swapping can be useful when interfacing with devices that organize data as big-endian. |
| CONT | Continuous transfer | When continuous transfer is selected in Peripheral mode, after the number of bits indicated by FRAMESZ are transferred, LPSPI passes through and transmits the received data until the next PCS negation. Whatever is shifted in on the receive data is shifted out as transmit data considering that there is a 32-bit shift register. |
| CONTC | Continuing command | When the continuing command is enabled in Peripheral mode, after the number of bits indicated by FRAMESZ are transferred, RXMSK is considered equal to 1 and TXMSK is considered equal to 0 until the next PCS negation. CONTC can be used to change the direction of a transfer after the number of bits indicated by FRAMESZ. |
| RXMSK | Receive data mask | Masks the receive data; LPSPI does not store masked receive data in the receive FIFO or perform receive data matching. This option is useful for half-duplex transfers or to specify which fields are compared during receive data matching. |
| TXMSK | Transmit data mask | Masks the transmit data so that the masked transmit data is not pulled from transmit FIFO, and the output data pin is 3-stated (unless otherwise specified in CFGR1[OUTCFG]). This option is useful for half-duplex transfers. |
| WIDTH | Transfer width | Specifies the number of bits shifted on each SCK pulse:<br>• 1-bit transfers support traditional SPI bus transfers in either half-duplex or full-duplex data formats.<br>• 2-bit and 4-bit half-duplex transfers are useful for interfacing with QuadSPI memory devices, and at least either TCR[TXMSK] or TCR[RXMSK] must be 1.<br>• 8-bit half-duplex transfers are useful for interfacing with OctelSPI memory devices, and at least either TCR[TXMSK] or TCR[RXMSK] must also be 1. |
| FRAMESZ | Frame size | Specifies the frame size in number of bits equal to (FRAMESZ + 1):<br>• The minimum frame size is 8 bits, or 16 bits for an 8-bit parallel transfer.<br>• If the frame size is larger than 32 bits, then the frame is divided into multiple words of 32 bits; each word is loaded from the transmit FIFO and stored in the receive FIFO separately.<br>• If the size of the frame is not divisible by 32, then the last load of the transmit FIFO and store of the receive FIFO contain the remainder bits. For example, a 72-bit transfer consists of three words: the first and second words are 32 bits, and the third word is 8 bits. |

#### 67.3.2.2 Receive FIFO and data match

The receive FIFO stores receive data during SPI bus transfers. When TCR[RXMSK] = 1, the received data is discarded instead of storing the received data in the receive FIFO.

Receive data supports a receive data match function that can match received data against one of the two words in DMR0 and DMR1 or against a masked data word. You can also configure the data match function to compare only the first one or two received data words since the start of the frame.

- Received data that is already discarded because TCR[RXMSK] = 1 cannot cause the data match to set, and delays the match on the first received data word, until all discarded data is received.
- By using CFGR0[RDMO], you can also configure the receiver match function to discard all received data until a data match is detected.
- After a receive data match, to allow all subsequent data to be received, first write 0 to CFGR0[RDMO], then clear SR[DMF].

#### 67.3.2.3 Partial received word

When the PCS pin deasserts and the receive shift register shifts in a partial word, you can configure the receive shift register to either discard the partial word or to store it in the receive FIFO. You must specify this using CFGR1[PARTIAL].

A partial word is defined as less than TCR[FRAMESZ] bits (when TCR[FRAMESZ] is equal or less than 32 bits, or it is the last word in a multi-word frame) or less than 32 bits (when TCR[FRAMESZ] is greater than 32 bits and not the last word in a multi-word frame).

A single-bit frame is not supported. A partial received word of 1 bit is supported, but a partial received frame of 1 bit is not supported.

#### 67.3.2.4 Clocked interface

LPSPI supports interfacing with external controllers that provide only clock and data pins (PCS is not required). This interface requires:

- Writing 1 to TCR[CPHA] (data is changed on the leading edge of SCK and captured on the following edge).
- Configuring the PCS input to be always asserted (CFGR1[PCSPOL*n*] = 1). For example, to configure PCS[0] to be always asserted, write 1 to PCSPOL[0], and do not configure PCS[0] in the pin muxing. The chip-level drives PCS to a certain value (ideally 1); you could use CFGR1[PCSPOL*n*] to invert that value.
- Writing 1 to CFGR1[AUTOPCS] to enable automatic PCS generation. When CFGR1[AUTOPCS] = 1, a minimum of four LPSPI functional clock cycles (divided by the selection specified in TCR[PRESCALE]) is required between the last SCK edge of one word and the first SCK edge of the next word.

### 67.3.3 Debug mode

**Table 370. Debug mode**

| Chip mode | LPSPI operation |
|---|---|
| Debug (the core is in Debug or Halted mode) | Can continue operating in Debug mode, if CR[DBGEN] = 1 |

### 67.3.4 Clocking

**Table 371. LPSPI clocks**

| Type of clock | Description |
|---|---|
| Functional | • Asynchronous to the bus clock.<br>• If the LPSPI functional clock remains enabled in low-power modes, then LPSPI can perform SPI bus transfers and low-power wakeups in both Controller and Peripheral modes.<br>• LPSPI divides the functional clock by a prescaler; the resulting frequency must be at least two times faster than the SPI external clock frequency (SCK). |
| External | • The LPSPI shift register is clocked directly by the SCK clock.<br>• How the SCK clock is generated or supplied depends on the mode (Controller or Peripheral):<br>— In Controller mode, the SCK clock is generated internally.<br>— In Peripheral mode, the SCK clock is supplied externally. |
| Bus | The bus clock is only used for bus accesses to the LPSPI control and configuration registers. The bus clock frequency must be high enough to support the data bandwidth requirements of the LPSPI registers, including the FIFOs. |

See the chip-specific LPSPI information for more.

### 67.3.5 Reset

**Table 372. LPSPI resets**

| Type of reset | Description |
|---|---|
| Chip | Resets the LPSPI logic and registers to their default states. |
| Software | • Resets the LPSPI logic and registers to their default states, except for the Control register.<br>• The LPSPI software reset is controlled using CR[RST]. |
| FIFO | • Resets the transmit and command FIFO and the receive FIFO.<br>• CR[RTF] and CR[RRF] are write-only.<br>• After being reset, FIFO is empty. |

### 67.3.6 Interrupts and DMA requests

The following table lists sources (status flags) that can generate LPSPI interrupts and LPSPI transmit and receive DMA requests.

**Table 373. Interrupts and DMA requests**

| Status (SR) | | Description | Can generate | | |
|---|---|---|---|---|---|
| Status flag | Name | | Interrupt? | DMA request? | Low-power wake-up? |
| TDF | Transmit data flag | Indicates that data can be written to transmit FIFO, as configured by the transmit FIFO watermark, FCR[TXWATER]. | Y | TX | Y |

*Table continues on the next page...*

Table 373. Interrupts and DMA requests (continued)

| Status (SR) | | Description | Can generate | | |
|---|---|---|---|---|---|
| Status flag | Name | | Interrupt? | DMA request? | Low-power wake-up? |
| RDF | Receive data flag | Indicates that data can be read from the receive FIFO, as configured by the receive FIFO watermark, FCR[RXWATER]. | Y | RX | Y |
| WCF | Word complete flag | Indicates that the word is complete and the last bit of the word has been sampled. | Y | N | Y |
| FCF | Frame complete flag | Indicates that the frame is complete and PCS is deasserted. | Y | N | Y |
| TCF | Transfer complete flag | Indicates that transfer is complete, PCS is deasserted, and the transmit and command FIFO is empty. | Y | N | Y |
| TEF | Transmit error flag | Indicates a transmit and command FIFO underrun. In Controller mode, when CFGR1[NOSTALL] = 0 (transfers stall when transmit FIFO is empty), TEF cannot be set. | Y | N | Y |
| REF | Receive error flag | Indicates a receive FIFO overflow. In Controller mode, when CFGR1[NOSTALL] = 0 (transfers stall when receive FIFO is full), REF cannot be set. | Y | N | Y |
| DMF | Data match flag | Indicates that the received data matches the configured data match value. | Y | N | Y |
| MBF | Module busy flag | Indicates that LPSPI is busy performing an SPI bus transfer. | N | N | N |

#### 67.3.6.1 DMA support registers

To support efficient DMA transfers to the transmit and control FIFO, an alias register supports 32-bit write access to the Transmit Command (TCR) and an alias region supports incrementing 32-bit write accesses to the Transmit Data (TDR).

- Transmit Command Burst (TCBR) is a 32-bit alias register for writing to TCR.
- Transmit Data Burst (TDBR0 - TDBR127) is a 512-byte alias region that supports writing to the TDR.

The burst alias locations are contiguous. A DMA transfer can start by writing to the TCBR register to initialize the transfer and then increment into the TDBR region without changing the DMA transfer size. The alias registers can also be used by the DMA to perform burst transfers when accessing the transmit FIFO.

The transmit FIFO prevents writes that overflow the FIFO, but this prevention does not signal an error. Do not perform writes to the TDBR unless there is sufficient room in the transmit FIFO.

Receive Data Burst (RDBR0 - RDBR127) is a 512-byte alias region that supports reading the Receive Data. This can be used by the DMA to perform burst transfers when accessing the receive FIFO.

### 67.3.7 Peripheral triggers

The connection of the LPSPI peripheral triggers with other peripherals depends on the device that is used.

Table 374. Peripheral triggers

| Type of trigger | Description | Additional information |
|---|---|---|
| Frame output | The frame output trigger:<br>• Asserts at the end of each frame (when PCS deasserts).<br>• Remains asserted for one cycle of the LPSPI functional clock divided by the configuration defined in TCR[PRESCALE]. | LPSPI generates two output triggers that can be connected to other peripherals on the chip. |
| Word output | The word output trigger:<br>• Asserts at the end of each received word.<br>• Remains asserted for one cycle of the LPSPI functional clock divided by the configuration defined in TCR[PRESCALE]. | |
| Input | To control the start of an LPSPI bus transfer, the LPSPI input trigger can be selected instead of the HREQ input:<br>• The LPSPI input trigger is synchronized, and must assert for at least two cycles of the LPSPI functional clock divided by the configuration defined in TCR[PRESCALE] so that the input trigger can be detected.<br>• When LPSPI is busy, the HREQ input (and therefore the LPSPI input trigger) is ignored.<br>• When LPSPI is busy, both the HREQ and LPSPI input triggers are ignored. They are used to start a new transfer when LPSPI is idle. | |

## 67.4 External signals

Table 375. External signals

| Signal | Name | Description | I/O |
|---|---|---|---|
| SCK | Serial clock | • Input in Peripheral mode<br>• Output in Controller mode | I/O |
| PCS[0] | Peripheral chip select | • Input in Peripheral mode<br>• Output in Controller mode | I/O |
| PCS[1]/HREQ | Peripheral chip select<br>or<br>host request | Host request pin is selected when CFGR0[HREN] = 1 and CFGR0[HRSEL] = 0:<br>• Input in either Peripheral mode or when used as controller host request<br>• Output in either Controller mode or when used as peripheral host request | I/O |
| PCS[2]/DATA[2] | Peripheral chip select or data pin 2 during parallel data transfers | When CFGR1[PCSCFG] = 0:<br>• Input in Peripheral mode<br>• Output in Controller mode<br>When CFGR1[PCSCFG] = 1: | I/O |

*Table continues on the next page...*

Table 375. External signals (continued)

| Signal | Name | Description | I/O |
|---|---|---|---|
| | | • Input in half-duplex parallel data receive transfers<br>• Output in half-duplex parallel data transmit transfers | |
| PCS[3]/DATA[3] | Peripheral chip select or data pin 3 during parallel data transfers | When CFGR1[PCSCFG] = 0:<br>• Input in Peripheral mode<br>• Output in Controller mode<br>When CFGR1[PCSCFG] = 1:<br>• Input in half-duplex parallel data receive transfers<br>• Output in half-duplex parallel data transmit transfers | I/O |
| PCS[4]/DATA[4] | Peripheral chip select or data pin 4 during parallel data transfers | When CFGR1[PCSCFG] = 0:<br>• Input in Peripheral mode<br>• Output in Controller mode<br>When CFGR1[PCSCFG] = 11b:<br>• Input in half-duplex parallel data receive transfers<br>• Output in half-duplex parallel data transmit transfers | I/O |
| PCS[5]/DATA[5] | Peripheral chip select or data pin 5 during parallel data transfers | When CFGR1[PCSCFG] = 0:<br>• Input in Peripheral mode<br>• Output in Controller mode<br>When CFGR1[PCSCFG] = 11b:<br>• Input in half-duplex parallel data receive transfers<br>• Output in half-duplex parallel data transmit transfers | I/O |
| PCS[6]/DATA[6] | Peripheral chip select or data pin 6 during parallel data transfers | When CFGR1[PCSCFG] = 0:<br>• Input in Peripheral mode<br>• Output in Controller mode<br>When CFGR1[PCSCFG] = 11b:<br>• Input in half-duplex parallel data receive transfers<br>• Output in half-duplex parallel data transmit transfers | I/O |

*Table continues on the next page...*

Table 375. External signals (continued)

| Signal | Name | Description | I/O |
|---|---|---|---|
| PCS[7]/DATA[7] | Peripheral chip select or data pin 7 during parallel data transfers | When CFGR1[PCSCFG] = 0:<br>• Input in Peripheral mode<br>• Output in Controller mode<br>When CFGR1[PCSCFG] = 11b:<br>• Input in half-duplex parallel data receive transfers<br>• Output in half-duplex parallel data transmit transfers | I/O |
| SOUT/DATA[0] | Serial data output | Can be configured as serial data input signal (used as data pin 0 in half-duplex parallel data transfers) | I/O |
| SIN/DATA[1] | Serial data input | Can be configured as serial data output signal (used as data pin 1 in half-duplex parallel data transfers) | I/O |

## 67.5 Initialization

This module does not require initialization.

## 67.6 Memory map and registers

**NOTE**

- Writing to a read-only register or reading a write-only register can cause bus errors.
- LPSPI does not check values programmed in registers for validity, so you must take care to write valid values only.

### 67.6.1 LPSPI register descriptions

#### 67.6.1.1 LPSPI memory map

LPSPI_0 base address: 4035_8000h

LPSPI_1 base address: 4035_C000h

LPSPI_2 base address: 4036_0000h

LPSPI_3 base address: 4036_4000h

LPSPI_4 base address: 404B_C000h

LPSPI_5 base address: 404C_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID (VERID) | 32 | R | 0200_0004h |
| 4h | Parameter (PARAM) | 32 | R | See section |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 10h | Control (CR) | 32 | RW | 0000_0000h |
| 14h | Status (SR) | 32 | RW | 0000_0001h |
| 18h | Interrupt Enable (IER) | 32 | RW | 0000_0000h |
| 1Ch | DMA Enable (DER) | 32 | RW | 0000_0000h |
| 20h | Configuration 0 (CFGR0) | 32 | RW | 0000_0000h |
| 24h | Configuration 1 (CFGR1) | 32 | RW | 0000_0000h |
| 30h | Data Match 0 (DMR0) | 32 | RW | 0000_0000h |
| 34h | Data Match 1 (DMR1) | 32 | RW | 0000_0000h |
| 40h | Clock Configuration (CCR) | 32 | RW | 0000_0000h |
| 44h | Clock Configuration 1 (CCR1) | 32 | RW | 0000_0000h |
| 58h | FIFO Control (FCR) | 32 | RW | 0000_0000h |
| 5Ch | FIFO Status (FSR) | 32 | R | 0000_0000h |
| 60h | Transmit Command (TCR) | 32 | RW | 0000_001Fh |
| 64h | Transmit Data (TDR) | 32 | W | 0000_0000h |
| 70h | Receive Status (RSR) | 32 | R | 0000_0002h |
| 74h | Receive Data (RDR) | 32 | R | 0000_0000h |
| 78h | Receive Data Read Only (RDROR) | 32 | R | 0000_0000h |
| 3FCh | Transmit Command Burst (TCBR) | 32 | W | 0000_0000h |
| 400h - 5FCh | Transmit Data Burst (TDBR0 - TDBR127) | 32 | W | 0000_0000h |
| 600h - 7FCh | Receive Data Burst (RDBR0 - RDBR127) | 32 | R | 0000_0000h |

### 67.6.1.2 Version ID (VERID)

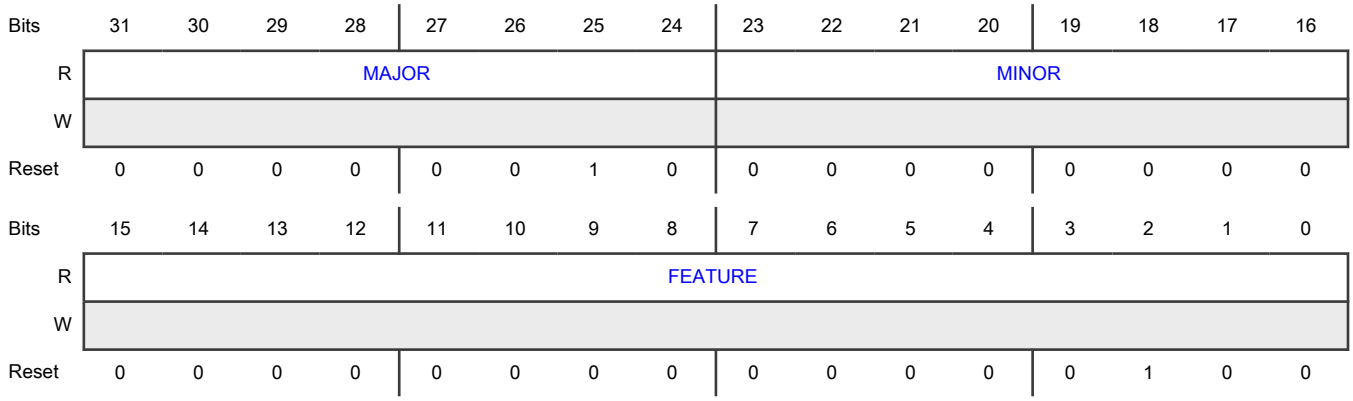
**Offset**

| Register | Offset |
|---|---|
| VERID | 0h |

**Function**

Contains version numbers for the module design and feature set.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>Indicates the major version number of the module specification. |
| 23-16<br>MINOR | Minor Version Number<br>Indicates the minor version number of the module specification. |
| 15-0<br>FEATURE | Module Identification Number<br>Indicates the feature set number<br>0000_0000_0000_0100b - Standard feature set supporting a 32-bit shift register.<br>All other values are reserved. |

## 67.6.1.3 Parameter (PARAM)

### Offset

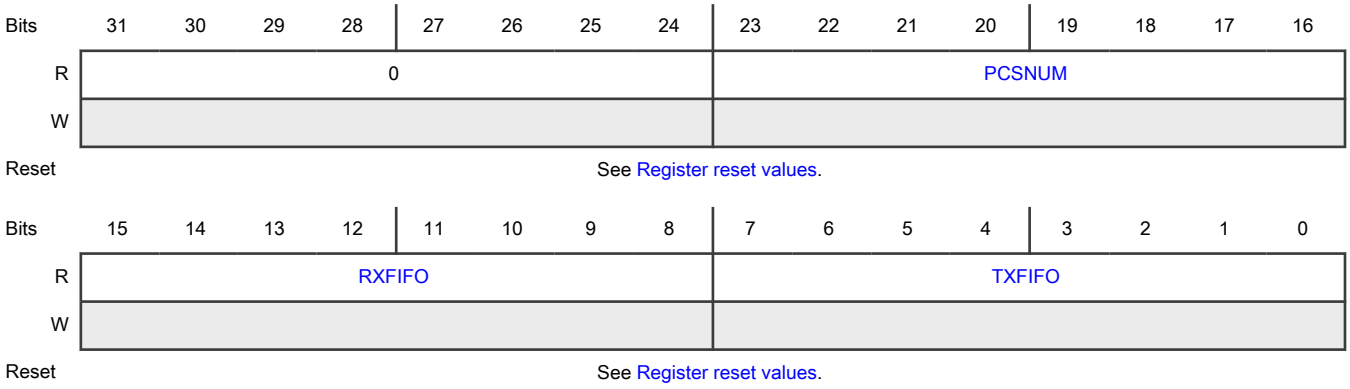
| Register | Offset |
|---|---|
| PARAM | 4h |

### Function

Contains:

- Number of PCS pins.
- Receive FIFO size.
- Transmit FIFO size.

### Diagram




### Register reset values

| Register | Reset value |
|---|---|
| PARAM | LPSPI_0: 0008_0202h<br>LPSPI_1: 0006_0202h<br>LPSPI_2–LPSPI_5: 0004_0202h |

### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16<br>PCSNUM | PCS Number<br>Indicates the number of PCS pins supported. |
| 15-8<br>RXFIFO | Receive FIFO Size<br>Indicates the maximum number of words in the receive FIFO. The maximum number of words is $2^{RXFIFO}$. |
| 7-0<br>TXFIFO | Transmit FIFO Size<br>Indicates the maximum number of words in the transmit FIFO. The maximum number of words is $2^{TXFIFO}$. |

## 67.6.1.4 Control (CR)

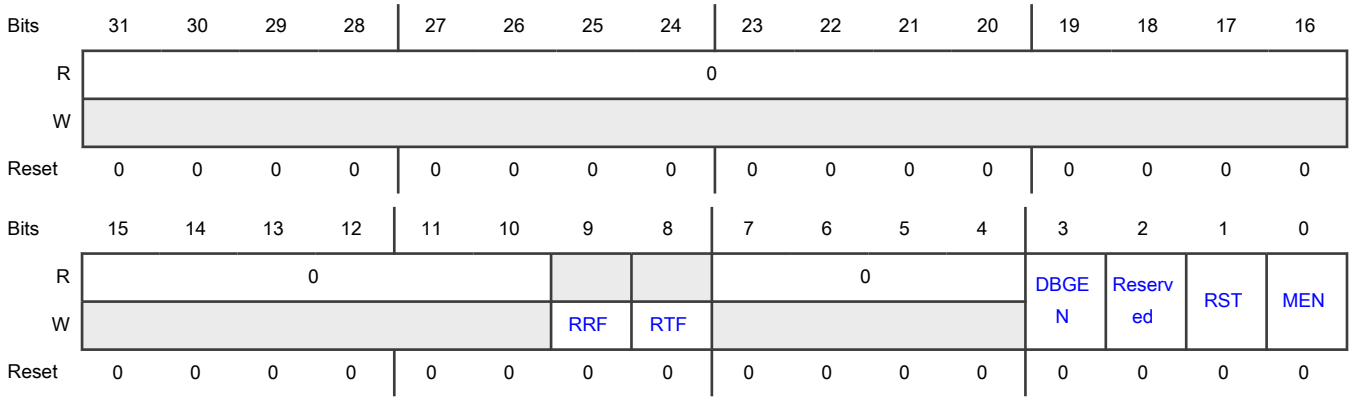
### Offset

| Register | Offset |
|---|---|
| CR | 10h |

### Function

Contains fields that control the module operation.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>RRF | Reset Receive FIFO<br>Deletes all entries in the receive FIFO. This field always reads 0.<br>0b - No effect<br>1b - Reset |
| 8<br>RTF | Reset Transmit FIFO<br>Deletes all entries in the transmit FIFO. This field always reads 0.<br>0b - No effect<br>1b - Reset |
| 7-4<br>— | Reserved |
| 3<br>DBGEN | Debug Enable<br>Enables LPSPI when the CPU is in Debug mode.<br>If this field is 0, LPSPI is disabled when the CPU is halted; the PCS pin is deasserted after the transmit FIFO is empty regardless of the state of Transmit Command (TCR).<br>You must update this field only when LPSPI is disabled (MEN = 0).<br>0b - Disable<br>1b - Enable |
| 2<br>— | Reserved |
| 1 | Software Reset |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| RST | Resets all internal logic and registers, except Control (CR). The reset takes effect immediately and remains asserted until you write 0 to it. There is no minimum delay required before clearing the software reset by writing 0.<br>0b - Not reset<br>1b - Reset |
| 0<br>MEN | Module Enable<br>Enables the module. After writing 0, MEN remains set until LPSPI has completed the current transfer and is idle.<br>0b - Disable<br>1b - Enable |

### 67.6.1.5 Status (SR)

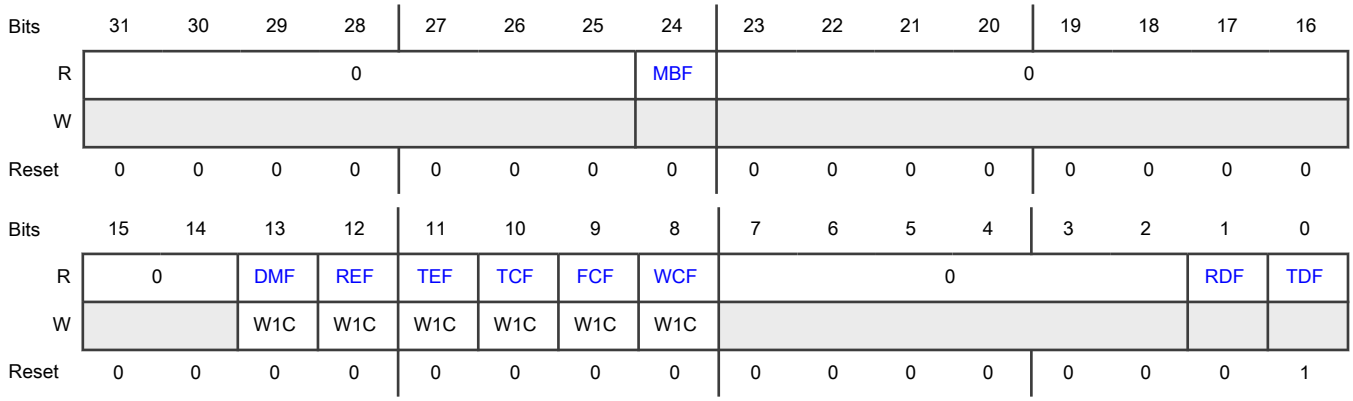
#### Offset

| Register | Offset |
|---|---|
| SR | 14h |

#### Function

Contains data flow status.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-25 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 24<br>MBF | Module Busy Flag<br>Indicates, in Controller mode, whether there is data to transmit and LPSPI is able to transmit (for example, the HREQ pin is asserted). The HREQ pin deasserts after the PCS pin deasserts and the LPSPI controller has waited for half the time specified in CCR[DBT] with no new data to transmit.<br>Peripheral mode sets this flag when LPSPI is enabled and PCS is asserted.<br>0b - LPSPI is idle<br>1b - LPSPI is busy |
| 23-14<br>— | Reserved |
| 13<br>DMF | Data Match Flag<br>Indicates whether the received data matches DMR0[MATCH0] and/or DMR1[MATCH1] (as configured by CFGR1[MATCFG]).<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No match<br>1b - Match<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 12<br>REF | Receive Error Flag<br>Indicates a receive FIFO overflow error. When this flag is set:<br>1. End the transfer.<br>2. Empty the receive FIFO.<br>3. Clear this flag.<br>4. Restart the transfer from the beginning.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No overflow<br>1b - Overflow<br>When writing |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No effect<br>1b - Clear the flag |
| 11<br>TEF | Transmit Error Flag<br>Indicates a transmit FIFO underrun error. When this flag is set:<br>1. End the transfer.<br>2. Clear this flag.<br>3. Restart the transfer from the beginning.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No underrun<br>1b - Underrun<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 10<br>TCF | Transfer Complete Flag<br>Indicates, in Controller mode, whether all transfers are complete and LPSPI has returned to the Idle state and the transmit FIFO is empty.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not complete<br>1b - Complete<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 9<br>FCF | Frame Complete Flag<br>Indicates whether a frame transfer is complete after PCS deasserts.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not complete |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Complete<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 8<br>WCF | Word Complete Flag<br>Indicates whether the last bit of a received word is sampled.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not complete<br>1b - Complete<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 7-2<br>— | Reserved |
| 1<br>RDF | Receive Data Flag<br>Indicates whether the number of words in the receive FIFO is greater than the value in FCR[RXWATER].<br>0b - Receive data not ready<br>1b - Receive data ready |
| 0<br>TDF | Transmit Data Flag<br>Indicates whether the number of words in the transmit FIFO is equal to or less than the value in FCR[TXWATER].<br>0b - Transmit data not requested<br>1b - Transmit data requested |

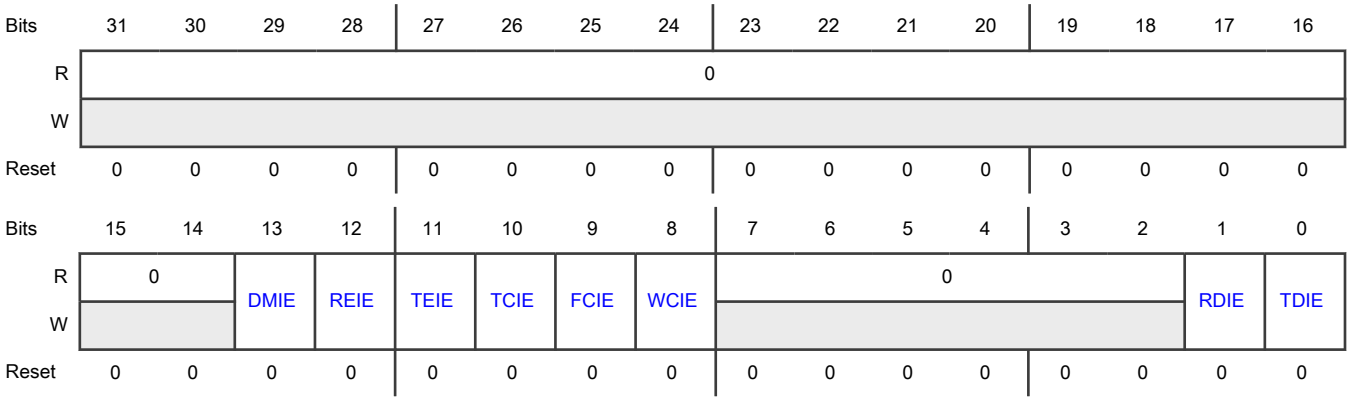
### 67.6.1.6 Interrupt Enable (IER)

**Offset**

| Register | Offset |
|---|---|
| IER | 18h |

**Function**
Enables interrupts based on data flow and errors.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-14<br>— | Reserved |
| 13<br>DMIE | Data Match Interrupt Enable<br>Enables the data match interrupt.<br>0b - Disable<br>1b - Enable |
| 12<br>REIE | Receive Error Interrupt Enable<br>Enables the receive error interrupt.<br>0b - Disable<br>1b - Enable |
| 11<br>TEIE | Transmit Error Interrupt Enable<br>Enables the transmit error interrupt.<br>0b - Disable<br>1b - Enable |
| 10<br>TCIE | Transfer Complete Interrupt Enable<br>Enables the transfer complete interrupt.<br>0b - Disable<br>1b - Enable |
| 9<br>FCIE | Frame Complete Interrupt Enable<br>Enables the frame complete interrupt.<br>0b - Disable<br>1b - Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 8<br>WCIE | Word Complete Interrupt Enable<br>Enables the word complete interrupt.<br>0b - Disable<br>1b - Enable |
| 7-2<br>— | Reserved |
| 1<br>RDIE | Receive Data Interrupt Enable<br>Enables the receive data interrupt.<br>0b - Disable<br>1b - Enable |
| 0<br>TDIE | Transmit Data Interrupt Enable<br>Enables the transmit data interrupt.<br>0b - Disable<br>1b - Enable |

### 67.6.1.7 DMA Enable (DER)

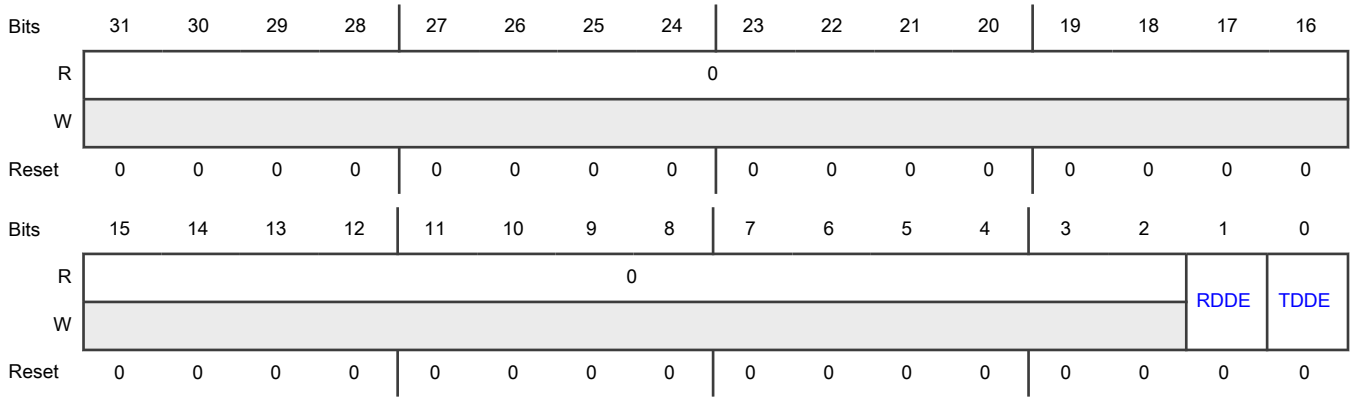
#### Offset

| Register | Offset |
|---|---|
| DER | 1Ch |

#### Function

Enables the DMA data flow.

#### Diagram



### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RDDE | Receive Data DMA Enable<br>Enables the receive data DMA.<br>0b - Disable<br>1b - Enable |
| 0<br>TDDE | Transmit Data DMA Enable<br>Enables the transmit data DMA.<br>0b - Disable<br>1b - Enable |

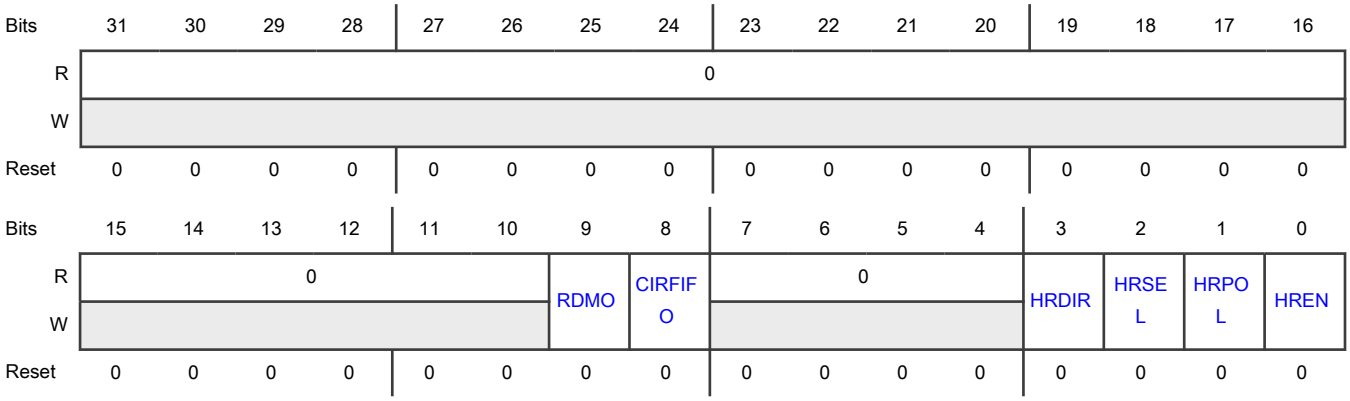
## 67.6.1.8 Configuration 0 (CFGR0)

### Offset

| Register | Offset |
|---|---|
| CFGR0 | 20h |

### Function
Includes fields to configure LPSPI.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>RDMO | Receive Data Match Only<br>Enables receive data match.<br>When enabled, all received data that does not cause SR[DMF] to assert is discarded:<br>• Write 1 to this field when LPSPI is idle and SR[DMF] = 0.<br>• After SR[DMF] = 1, this field is ignored.<br>• To ensure that no receive data is lost when disabling RDMO, write 0 to this field before clearing SR[DMF].<br>See CFGR1[MATCFG] for the received data matching options. When disabled, all received data is stored in the receive FIFO.<br>0b - Disable<br>1b - Enable |
| 8<br>CIRFIFO | Circular FIFO Enable<br>Enables circular FIFO.<br>When enabled, the transmit FIFO read pointer is saved to a temporary register. The transmit FIFO is emptied as in normal operation, but when LPSPI is idle and the transmit FIFO is empty, the read pointer value is restored from the temporary register.<br>This restoring of the read pointer causes the contents of the transmit FIFO to be cycled through repeatedly.<br>**NOTE**<br>The read pointer is restored for as long as this field is 1. Writing additional words to the FIFO when this field is 1 adds them to the end of the FIFO, up to the size of the transmit FIFO.<br>0b - Disable<br>1b - Enable |
| 7-4<br>— | Reserved |
| 3<br>HRDIR | Host Request Direction<br>Specifies the direction of the HREQ pin. You must configure the HREQ pin only as an output when LPSPI is in Peripheral mode. The HREQ pin direction must be an input for Controller mode.<br>0b - Input<br>1b - Output |
| 2<br>HRSEL | Host Request Select<br>Specifies the source of the host request input. When the host request function is enabled with the HREQ pin, the PCS[1] function is disabled. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - HREQ pin<br>1b - Input trigger |
| 1<br>HRPOL | Host Request Polarity<br>Specifies the polarity of the HREQ pin or input trigger.<br>0b - Active high<br>1b - Active low |
| 0<br>HREN | Host Request Enable<br>Enables LPSPI, in Controller mode, to start a new SPI bus transfer only if the host request input is asserted. When LPSPI is busy, the host request input is ignored.<br>In Peripheral mode, causes the HREQ output pin to assert when data is available to be transmitted.<br>0b - Disable<br>1b - Enable |

### 67.6.1.9 Configuration 1 (CFGR1)

#### Offset

| Register | Offset |
|---|---|
| CFGR1 | 24h |

#### Function

Includes fields to configure LPSPI. You must write to this register only when LPSPI is disabled.

In addition to pin and output configurations, this register contains match configuration details; the following table shows match conditions specified in MATCFG.

**Table 376. Match conditions for CFGR1[MATCFG]**

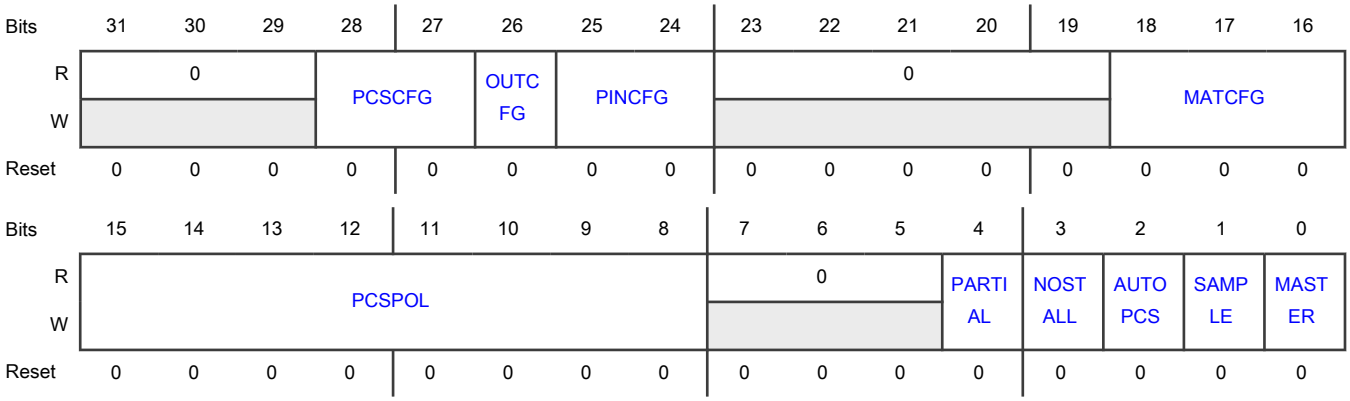
| Condition | Description |
|---|---|
| Match first data word with compare word | Match if first data word equals MATCH0 logically ORed with MATCH1<br>first_data_word == (MATCH0 \|\| MATCH1) |
| Match any data word with compare word | Match if any data word equals MATCH0 logically ORed with MATCH1<br>any_data_word == (MATCH0 \|\| MATCH1) |
| Sequential match, first data word | Match if first data word equals MATCH0, and second data word equals MATCH1<br>(first_data_word == MATCH0) && (second_data_word == MATCH1) |
| Sequential match, any data word | Match if any data word equals MATCH0, and the next data word equals MATCH1 |

*Table continues on the next page...*

Table 376. Match conditions for CFGR1[MATCFG] (continued)

| Condition | Description |
|---|---|
| | (any_data_word == MATCH0) && (next_data_word == MATCH1) |
| Match first data word (masked) with compare word (masked) | Match if first data word logically ANDed with MATCH1 equals MATCH0 logically ANDed with MATCH1<br>(first_data_word && MATCH1) == (MATCH0 && MATCH1) |
| Match any data word (masked) with compare word (masked) | Match if any data word logically ANDed with MATCH1 equals MATCH0 logically ANDed with MATCH1<br>(any_data_word && MATCH1) == (MATCH0 && MATCH1) |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |
| 28-27<br>PCSCFG | Peripheral Chip Select Configuration<br>Specifies PCS pin configuration. When performing parallel transfers, you must configure this field to enable the desired transfer.<br>**NOTE**<br>This field is not supported in every instance. The following table includes only supported registers. |

| Instance | Field supported in | Field not supported in |
|---|---|---|
| LPSPI_0 | CFGR1 | — |
| LPSPI_1 | CFGR1[27] | CFGR1[28] |
| LPSPI_2 | CFGR1[27] | CFGR1[28] |

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>LPSPI_3</td><td>CFGR1[27]</td><td>CFGR1[28]</td></tr>
<tr><td>LPSPI_4</td><td>CFGR1[27]</td><td>CFGR1[28]</td></tr>
<tr><td>LPSPI_5</td><td>CFGR1[27]</td><td>CFGR1[28]</td></tr>
</table>
NOTE<br>The descriptions of the field settings vary by module instance.
<table>
<tr><th>Instance</th><th>Field value and description</th></tr>
<tr><td>LPSPI_0</td><td>00b - PCS[7:2] configured for chip select function<br>01b - PCS[3:2] configured for half-duplex 4-bit transfers (PCS[3:2] = DATA[3:2])<br>11b - PCS[7:2] configured for half-duplex 4-bit and 8-bit transfers (PCS[7:2] = DATA[7:2])</td></tr>
<tr><td>LPSPI_1</td><td>0b - PCS[5:2] configured for chip select function<br>1b - PCS[3:2] configured for half-duplex 4-bit transfers (PCS[3:2] = DATA[3:2])</td></tr>
<tr><td>LPSPI_2<br>LPSPI_3<br>LPSPI_4<br>LPSPI_5</td><td>0b - PCS[3:2] configured for chip select function<br>1b - PCS[3:2] configured for half-duplex 4-bit transfers (PCS[3:2] = DATA[3:2])</td></tr>
</table>
</td></tr>
<tr><td>26<br>OUTCFG</td><td>Output Configuration<br>Specifies whether the output data is 3-stated between accesses (when PCS is deasserted). When performing half-duplex transfers, this field must be 1.<br>0b - Retain last value<br>1b - 3-stated</td></tr>
<tr><td>25-24<br>PINCFG</td><td>Pin Configuration<br>Specifies the pins used for input and output data during serial transfers.<br>NOTE<br>When performing parallel transfers, this field must be 00b or 10b.<br>00b - SIN is used for input data; SOUT is used for output data<br>01b - SIN is used for both input and output data; only half-duplex serial transfers are supported</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>10b - SOUT is used for both input and output data; only half-duplex serial transfers are supported<br>11b - SOUT is used for input data; SIN is used for output data</td></tr>
<tr><td>23-19<br>—</td><td>Reserved</td></tr>
<tr><td>18-16<br>MATCFG</td><td>Match Configuration<br>Specifies the condition that causes SR[DMF] to assert. See the match conditions listed in Table 1 for more information.<br>NOTE<br>When writing to this field, either the old value or new value must be in the disabled state (0). You cannot transition from a nonzero value to another nonzero value.<br>000b - Match is disabled<br>001b - Reserved<br>010b - Match first data word with compare word<br>011b - Match any data word with compare word<br>100b - Sequential match, first data word<br>101b - Sequential match, any data word<br>110b - Match first data word (masked) with compare word (masked)<br>111b - Match any data word (masked) with compare word (masked)</td></tr>
<tr><td>15-8<br>PCSPOL</td><td>Peripheral Chip Select Polarity<br>Specifies the polarity of each PCS pin. Bit n in this field (the least-significant bit is bit 0) corresponds to PCS[n].<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>LPSPI_0</td><td>CFGR1</td><td>—</td></tr><tr><td>LPSPI_1</td><td>CFGR1[13–8]</td><td>CFGR1[15–14]</td></tr><tr><td>LPSPI_2</td><td>CFGR1[11–8]</td><td>CFGR1[15–12]</td></tr><tr><td>LPSPI_3</td><td>CFGR1[11–8]</td><td>CFGR1[15–12]</td></tr><tr><td>LPSPI_4</td><td>CFGR1[11–8]</td><td>CFGR1[15–12]</td></tr></table></td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>LPSPI_5</td><td>CFGR1[11–8]</td><td>CFGR1[15–12]</td></tr>
</table>
NOTE<br>The descriptions of the field settings vary by module instance.
<table>
<tr><th>Instance</th><th>Field value and description</th></tr>
<tr><td>LPSPI_0</td><td>0000_0000b - Active low<br>0000_0001b - Active high</td></tr>
<tr><td>LPSPI_1</td><td>00_0000b - Active low<br>00_0001b - Active high</td></tr>
<tr><td>LPSPI_2<br>LPSPI_3<br>LPSPI_4<br>LPSPI_5</td><td>0000b - Active low<br>0001b - Active high</td></tr>
</table>
</td></tr>
<tr><td>7-5<br>—</td><td>Reserved</td></tr>
<tr><td>4<br>PARTIAL</td><td>Partial Enable<br>Specifies whether LPSPI, when in Peripheral mode, stores a partial received word in the receive FIFO, or discards it, when PCS deasserts. See Partial received word for more information.<br>0b - Discard<br>1b - Store</td></tr>
<tr><td>3<br>NOSTALL</td><td>No Stall<br>Disables a normal operating feature that causes LPSPI, when in Controller mode, to stall transfers when the transmit FIFO is empty or when the receive FIFO is full. This feature prevents transmit FIFO underruns and receive FIFO overruns. Writing 1 to this field disables this functionality.<br>0b - Disable<br>1b - Enable</td></tr>
<tr><td>2<br>AUTOPCS</td><td>Automatic PCS<br>Enables automatic PCS generation. For correct operation in Peripheral mode, LPSPI requires the PCS signal to deassert between frames. Writing 1 to this field generates an internal PCS signal at the end of each transfer word when TCR[CPHA] = 1.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When this field is 1, SCK must remain idle for at least four LPSPI functional clock cycles, divided by the prescaler (see TCR[PRESCALE]) selected between each word to ensure correct operation.<br>This field is ignored in Controller mode.<br>0b - Disable<br>1b - Enable |
| 1<br>SAMPLE | Sample Point<br>Specifies the SCK clock edge on which LPSPI, when in Controller mode, samples input data. Writing 1 to this field causes LPSPI to sample input data on a delayed loopback SCK clock edge, which improves the setup time when sampling data (see Clock loopback). In this configuration, the input data setup time in Controller mode is equal to the input data setup time in Peripheral mode.<br>In Peripheral mode, this field is ignored.<br>**NOTE**<br>When SAMPLE = 1, both the input and output buffers must be enabled for the SCK pin. Buffers are configured at the chip level.<br>0b - SCK edge<br>1b - Delayed SCK edge |
| 0<br>MASTER | Controller Mode<br>Specifies the LPSPI operating mode, Controller or Peripheral. This field directly controls the direction of the SCK and PCS pins.<br>0b - Peripheral mode<br>1b - Controller mode |

### 67.6.1.10 Data Match 0 (DMR0)

**Offset**

| Register | Offset |
|---|---|
| DMR0 | 30h |

**Function**

Specifies the match data to be used when data matching is enabled. See CFGR1[MATCFG] for the received data matching options.

**NOTE**

Do not change the value in this register when CFGR1[MATCFG] > 0.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCH0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCH0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>MATCH0 | Match 0 Value<br>Specifies the MATCH0 value to be compared against received data. |

## 67.6.1.11 Data Match 1 (DMR1)

### Offset

| Register | Offset |
|---|---|
| DMR1 | 34h |

### Function

Specifies the match data to be used when data matching is enabled. See CFGR1[MATCFG] for the received data matching options.

**NOTE**

Do not change the value in this register while CFGR1[MATCFG] > 0.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCH1 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MATCH1 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>MATCH1 | Match 1 Value<br>Specifies the MATCH1 value to be compared against received data. |

## 67.6.1.12 Clock Configuration (CCR)

### Offset

| Register | Offset |
|---|---|
| CCR | 40h |

### Function

Contains clock configuration fields that are used only in Controller mode; you can only change them when LPSPI is disabled (CR[MEN] = 0).

**Warning**

Writing a 32-bit value to this register overwrites Clock Configuration 1 (CCR1); DBT and SCKDIV always read 0.

To avoid overwriting CCR1, do one of the following:

- Write to all four fields in Clock Configuration (CCR) simultaneously and only once in a 32-bit data.
- Modify the values of CCR[SCKPCS] and/or CCR[PCSSCK]; write only these two upper bytes in a 16-bit data or one of them in an 8-bit data.
- Modify CCR1[PCSPCS] and CCR1[SCKSCK] only or CCR1[SCKSET] and CCR1[SCKHLD] only, write respectively to CCR[DBT] or CCR[SCKDIV] in 8-bit data.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SCKPCS | | | | | | | | PCSSCK | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31-24 | SCK-to-PCS Delay |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| SCKPCS | Configures SCK-to-PCS delay. In Controller mode, this field helps you configure the delay from the last SCK edge to PCS negation:<br>• The delay is equal to (SCKPCS + 1) cycles of the LPSPI functional clock divided by the selected prescaler (see TCR[PRESCALE]).<br>• The minimum delay is one cycle.<br>See Figure 360 for more information. |
| 23-16<br>PCSSCK | PCS-to-SCK Delay<br>Configures PCS-to-SCK delay. In Controller mode, this field helps you configure the delay from PCS assertion to the first SCK edge:<br>• The delay is equal to (PCSSCK + 1) cycles of the LPSPI functional clock divided by the selected prescaler (see TCR[PRESCALE]).<br>• The minimum delay is one cycle.<br>See Figure 360 for more information. |
| 15-8<br>DBT | Delay Between Transfers<br>Configures the delay between transfers. Writing to this field updates the contents of CCR1[PCSPCS] and CCR1[SCKSCK]. |
| 7-0<br>SCKDIV | SCK Divider<br>Updates the contents of CCR1[SCKSET] and CCR1[SCKHLD].<br>Baud rate = function clock ÷ (2^PRESCALE × (SCKSET + SCKHLD + 2)) |

### 67.6.1.13 Clock Configuration 1 (CCR1)

#### Offset

| Register | Offset |
|---|---|
| CCR1 | 44h |

#### Function

Contains clock configuration fields, which are used only in Controller mode. You can change them only when LPSPI is disabled (CR[MEN] = 0).

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SCKSCK | | | | | | | | PCSPCS | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SCKHLD | | | | | | | | SCKSET | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-24<br>SCKSCK | SCK Inter-Frame Delay<br>Configures SCK inter-frame delay in Controller mode:<br>• This field helps you configure the delay from the last SCK pulse of a frame and the first SCK pulse of the following frame, in a continuous transfer.<br>• The delay is equal to (SCKSCK + 1) cycles of the LPSPI functional clock divided by the selected prescaler (see TCR[PRESCALE]).<br>• The minimum delay is one cycle.<br>**NOTE**<br>For backward compatibility, writing to CCR[DBT] updates this field with the value written. |
| 23-16<br>PCSPCS | PCS to PCS Delay<br>Configures PCS to PCS delay in Controller mode:<br>• This field helps you configure the delay from the PCS negation to the next PCS assertion.<br>• The delay is equal to (PCSPCS + PCSPCS + 2) cycles of the LPSPI functional clock divided by the selected prescaler (see TCR[PRESCALE]).<br>• The minimum delay is two cycles.<br>• Half of the delay (PCSPCS + 1) occurs before PCS assertion and the other half of the delay (PCSPCS + 1) occurs after PCS negation. If the command word is updated between two transfers, then the command word is updated halfway between the PCS negation of the last transfer and PCS assertion of the next transfer.<br>• The command word specifies which PCS signal is used, the polarity and phase of the SCK signal, and the selected prescaler.<br>**NOTE**<br>For backward compatibility, writing to CCR[DBT] updates this field with (DBT÷2) rounded up. |
| 15-8 | SCK Hold |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| SCKHLD | Configures the hold phase of the SCK pin in Controller mode:<br>• The hold phase is the delay between the SCK edge that samples the receive data and the SCK edge that drives the transmit data.<br>• It is the SCK low period when CPHA = 0 and CPOL = 1, or CPHA = 1 and CPOL = 0. It is the SCK high period when CPHA = 0, CPOL = 0 and CPHA = 1, CPOL = 1.<br>• The SCK hold phase delay is equal to (SCKHLD + 1) cycles of the LPSPI functional clock divided by the selected prescaler (see TCR[PRESCALE]).<br>• The minimum delay is one cycle.<br>• The SCK period is equal to (SCKSET + SCKHLD + 2) cycles of the LPSPI functional clock divided by the selected prescaler (see TCR[PRESCALE]).<br>• The SCK duty cycle is based on the difference between SCKSET and SCKHLD. You must configure both these fields to the same value for a 50/50 duty cycle.<br>See Figure 360 for more information.<br>**NOTE**<br>For backward compatibility, writing to CCR[SCKDIV] updates this field with (SCKDIV ÷ 2) rounded down. |
| 7-0<br>SCKSET | SCK Setup<br>Configures the setup phase of the SCK pin in Controller mode:<br>• The setup phase is the delay between the SCK edge that drives the transmit data and the SCK edge that samples the receive data.<br>• It is the SCK high period when CPHA = 0 and CPOL = 1, or CPHA = 1 and CPOL = 0. It is the SCK low period when CPHA = 0 and CPOL = 0, or CPHA = 1 and CPOL = 1.<br>• The SCK setup phase delay is equal to (SCKSET + 1) cycles of the LPSPI functional clock divided by the selected prescaler (see TCR[PRESCALE]).<br>• The minimum delay is one cycle.<br>• The SCK period is equal to (SCKSET + SCKHLD + 2) cycles of the LPSPI functional clock divided by the selected prescaler (see TCR[PRESCALE]).<br>• The SCK duty cycle is based on the difference between SCKSET and SCKHLD. You must configure both these fields to the same value for a 50/50 duty cycle.<br>See Figure 360 for more information.<br>**NOTE**<br>For backward compatibility, writing to CCR[SCKDIV] updates this field with (SCKDIV ÷ 2) rounded up. |

### 67.6.1.14 FIFO Control (FCR)

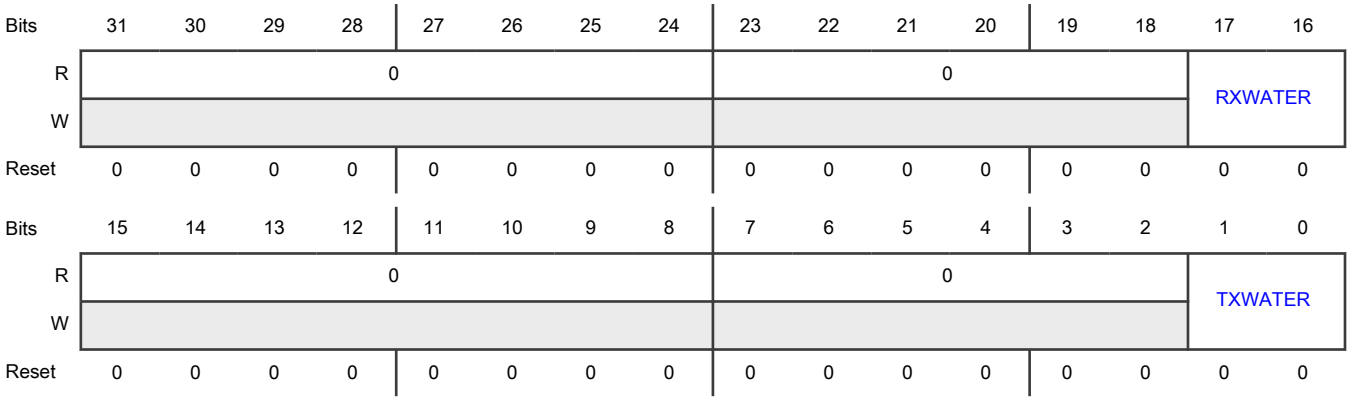
#### Offset

| Register | Offset |
|---|---|
| FCR | 58h |

#### Function

Contains the receive FIFO and transmit FIFO watermark values.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-18<br>— | Reserved |
| 17-16<br>RXWATER | Receive FIFO Watermark<br>Causes LPSPI to set SR[RDF] when the number of words in the receive FIFO is greater than RXWATER. Writing a value equal to or greater than the FIFO size truncates the written value. |
| 15-8<br>— | Reserved |
| 7-2<br>— | Reserved |
| 1-0<br>TXWATER | Transmit FIFO Watermark<br>Causes LPSPI to set SR[TDF] when the number of words in the transmit FIFO is equal to or less than TXWATER. Writing a value equal to or greater than the FIFO size truncates the written value. |

### 67.6.1.15 FIFO Status (FSR)

#### Offset

| Register | Offset |
|---|---|
| FSR | 5Ch |

#### Function

Contains fields that indicate the number of words currently stored in the receive and transmit FIFOs.

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-19<br>— | Reserved |
| 18-16<br>RXCOUNT | Receive FIFO Count<br>Indicates the number of words currently stored in the receive FIFO. |
| 15-8<br>— | Reserved |
| 7-3<br>— | Reserved |
| 2-0<br>TXCOUNT | Transmit FIFO Count<br>Indicates the number of words currently stored in the transmit FIFO. |

### 67.6.1.16 Transmit Command (TCR)

#### Offset

| Register | Offset |
|---|---|
| TCR | 60h |

#### Function

Pushes the data into the transmit FIFO, in the same order as written.

When you write to either this register or to Transmit Data (TDR), each write pushes data into the transmit FIFO. You must write to this register only using 32-bit writes, which are tagged and cause the command register to update; after that the entry reaches the top of the FIFO and LPSPI is enabled. This allows changes to the command word and the transmit data itself to be interleaved. That is, writes to the two registers can be interleaved (write command word, then data word, then command word, and so on). Changing the command word causes all subsequent SPI bus transfers to be performed using the new command word:

- In Controller mode, writing a new command word does not initiate a new transfer, unless TXMSK is 1. Transfers are initiated by transmit data in the transmit FIFO, or by a new command word (with TXMSK = 1). Hardware writes 0 to TXMSK when PCS deasserts.
- In Controller mode, if the command word is changed before an existing frame has completed, then the existing frame terminates and the command word updates. The command word can be changed during a continuous transfer, if CONTC of the new command word is 1 and the command word is written on a frame size boundary.
- In Peripheral mode, the command word must be changed only when LPSPI is idle and there is no SPI bus transfer.

Avoid resetting the transmit FIFO after writing to this register; wait for the command register to update from the FIFO first.

Avoid register reading problems: Reading this register returns the current state of the register. Reading this register at the same time that it is loaded from the transmit FIFO can return an incorrect register value. It is recommended to:

- Read this register when the transmit FIFO is empty.
- Read this register more than once and then compare the returned values.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | CPOL | CPHA | PRESCALE | | | PCS | | | LSBF | BYSW | CONT | CONTC | RXMSK | TXMSK | WIDTH | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31<br>CPOL | Clock Polarity<br>Specifies the value of SCK when it is idle. You can update this field only when PCS is deasserted.<br>See Figure 360 for more information.<br>0b - Inactive low<br>1b - Inactive high |
| 30<br>CPHA | Clock Phase<br>Indicates whether data is captured or changed on the leading edge of SCK and captured or changed on the following edge of SCK. You can update this field only when PCS is deasserted.<br>See Figure 360 for more information.<br>0b - Captured<br>1b - Changed |
| 29-27<br>PRESCALE | Prescaler Value<br>Specifies the division of the LPSPI functional clock. For all SPI bus transfers, this value is applied to Clock Configuration (CCR). You can update this field only when PCS is deasserted.<br>000b - Divide by 1<br>001b - Divide by 2<br>010b - Divide by 4<br>011b - Divide by 8<br>100b - Divide by 16<br>101b - Divide by 32<br>110b - Divide by 64<br>111b - Divide by 128 |

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 26-24<br>PCS | Peripheral Chip Select<br>Configures the peripheral chip select used for the transfer. This field is updated only when PCS is deasserted.<br>**NOTE**<br>This entire field is not fully supported in every LPSPI module instance. See the chip-specific LPSPI information.<br>**NOTE**<br>This field is not supported in every instance. The following table includes only supported registers. |

| Instance | Field supported in | Field not supported in |
|---|---|---|
| LPSPI_0 | TCR | — |
| LPSPI_1 | TCR | — |
| LPSPI_2 | TCR[25–24] | TCR[26] |
| LPSPI_3 | TCR[25–24] | TCR[26] |
| LPSPI_4 | TCR[25–24] | TCR[26] |
| LPSPI_5 | TCR[25–24] | TCR[26] |

**NOTE**
The descriptions of the field settings vary by module instance.

| Instance | Field value and description |
|---|---|
| LPSPI_0 | 000b - Transfer using PCS[0]<br>001b - Transfer using PCS[1]<br>010b - Transfer using PCS[2]<br>011b - Transfer using PCS[3]<br>100b - Transfer using PCS[4]<br>101b - Transfer using PCS[5]<br>110b - Transfer using PCS[6]<br>111b - Transfer using PCS[7] |
| LPSPI_1 | 000b - Transfer using PCS[0] |

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | <table><tr><th>Instance</th><th>Field value and description</th></tr><tr><td></td><td>001b - Transfer using PCS[1]<br>010b - Transfer using PCS[2]<br>011b - Transfer using PCS[3]<br>100b - Transfer using PCS[4]<br>101b - Transfer using PCS[5]</td></tr><tr><td>LPSPI_2<br>LPSPI_3<br>LPSPI_4<br>LPSPI_5</td><td>00b - Transfer using PCS[0]<br>01b - Transfer using PCS[1]<br>10b - Transfer using PCS[2]<br>11b - Transfer using PCS[3]</td></tr></table> |
| 23<br>LSBF | LSB First<br>Indicates whether data is transferred with MSB first or LSB first.<br>0b - MSB first<br>1b - LSB first |
| 22<br>BYSW | Byte Swap<br>Swaps the contents of [31:24] with [7:0] and [23:16] with [15:8] for each transmit data word read from the FIFO and for each received data word stored to the FIFO (or compared with match registers).<br>0b - Disable byte swap<br>1b - Enable byte swap |
| 21<br>CONT | Continuous Transfer<br>Enables continuous transfer:<br>• In Controller mode, this field keeps PCS asserted at the end of the frame size until a command word is received that starts a new frame.<br>• In Peripheral mode, when this field is enabled, LPSPI only transmits the first FRAMESZ bits, after which LPSPI transmits received data (assuming a 32-bit shift register) until the next PCS negation.<br>0b - Disable<br>1b - Enable |
| 20<br>CONTC | Continuing Command<br>Enables the command word to be changed within a continuous transfer in Controller mode:<br>• The initial command word must enable continuous transfer (CONT = 1).<br>• The continuing command must have CONTC = 1. |

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<ul><li>The continuing command word must be loaded on a frame size boundary.</li></ul>
For example, if the continuous transfer has a frame size of 64 bits, then a continuing command word must be loaded on a 64-bit boundary.<br>
In Peripheral mode, this field modifies the internal RXMSK and TXMSK configuration after the first FRAMESZ bits and until PCS negation:
<ul><li>Receive data is discarded after the first FRAMESZ bits. If CONT is also 1, this does not block the transmission of received data.</li><li>Transmit data is not masked after the first FRAMESZ bits. This allows the first FRAMESZ bits to be received and a response transmitted.</li></ul>
0b - Command word for start of new transfer<br>
1b - Command word for continuing transfer
</td></tr>
<tr><td>19<br>RXMSK</td><td>Receive Data Mask<br>Masks receive data (receive data is not stored in the receive FIFO).<br>0b - Normal transfer<br>1b - Mask receive data</td></tr>
<tr><td>18<br>TXMSK</td><td>Transmit Data Mask<br>Masks transmit data (no data is loaded from the transmit FIFO and the output pin is 3-stated). In Controller mode, TXMSK initiates a new transfer that cannot be aborted by another command word. TXMSK automatically transitions to 0 at the end of the transfer.<br>0b - Normal transfer<br>1b - Mask transmit data</td></tr>
<tr><td>17-16<br>WIDTH</td><td>Transfer Width<br>Configures serial (1-bit) or parallel transfers. For half-duplex parallel transfers, either RXMSK or TXMSK must be 1.<br>
NOTE<br>The descriptions of the field settings vary by module instance.
<table>
<tr><th>Instance</th><th>Field value and description</th></tr>
<tr><td>LPSPI_0</td><td>00b - 1-bit transfer<br>01b - 2-bit transfer<br>10b - 4-bit transfer<br>11b - 8-bit transfer</td></tr>
<tr><td>LPSPI_1<br>LPSPI_2</td><td>00b - 1-bit transfer<br>01b - 2-bit transfer</td></tr>
</table>
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | <table><tr><th>Instance</th><th>Field value and description</th></tr><tr><td>LPSPI_3<br>LPSPI_4<br>LPSPI_5</td><td>10b - 4-bit transfer<br>11b - Reserved</td></tr></table> |
| 15-12<br>— | Reserved |
| 11-0<br>FRAMESZ | Frame Size<br>Configures the frame size in number of bits equal to (FRAMESZ + 1):<br>• The minimum frame size is 8 bits, or 16 bits for an 8-bit transfer.<br>• If the frame size is larger than 32 bits, then the frame is divided into multiple words of 32 bits; each word is loaded from the transmit FIFO and stored in the receive FIFO separately.<br>• If the size of the frame is not divisible by 32, then the last load of the transmit FIFO and store of the receive FIFO contains the remainder bits. For example, a 72-bit transfer consists of three words: the first and second words are 32 bits, and the third word is 8 bits. |

### 67.6.1.17 Transmit Data (TDR)

#### Offset

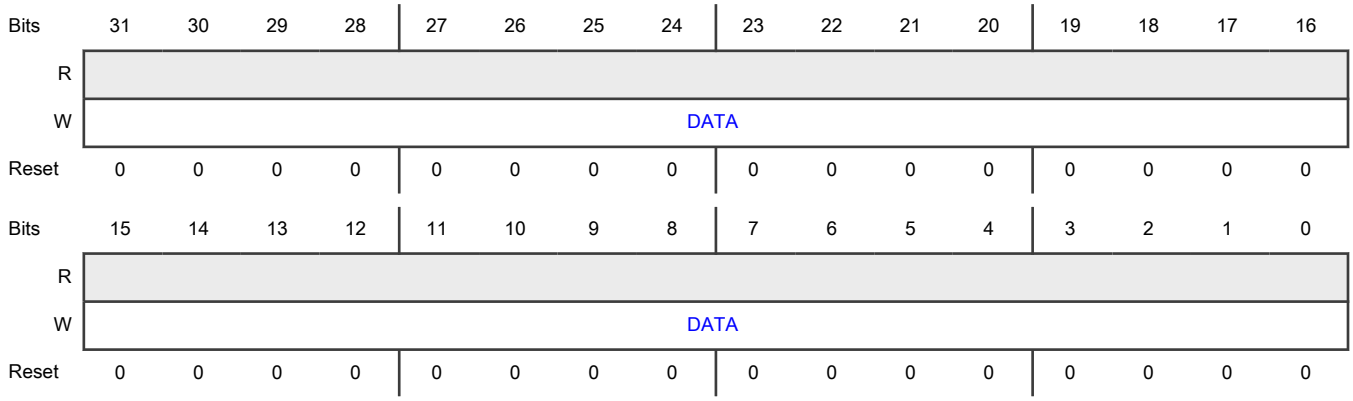
| Register | Offset |
|---|---|
| TDR | 64h |

#### Function

Pushes the data into the transmit FIFO, in the same order that the data is written. You can write to this register using 32-, 16-, or 8-bit writes.

When you write to this register or to Transmit Command (TCR), each write pushes data into the FIFO with zero pushed in unwritten bytes.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>DATA | Transmit Data<br>Indicates transmit data. Both 8-bit and 16-bit writes of transmit data zero-extend the data written and push the data into the transmit FIFO. To zero-extend 8-bit and 16-bit writes (to 32 bits) means that the higher order (most significant) empty parts of the 8-bit and 16-bit writes are filled with zeroes. |

## 67.6.1.18 Receive Status (RSR)

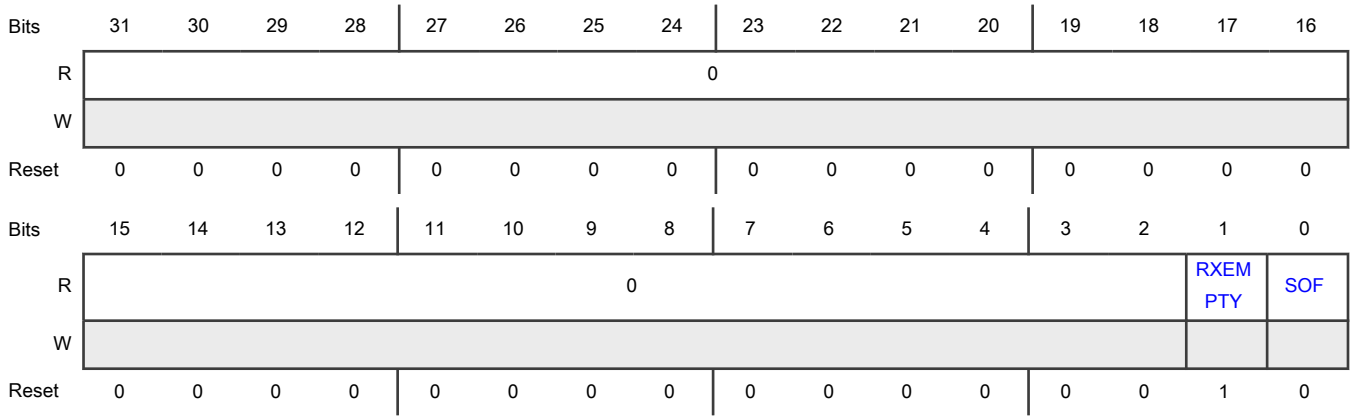
### Offset

| Register | Offset |
|---|---|
| RSR | 70h |

### Function

Contains data flow status fields for receive FIFO.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RXEMPTY | RX FIFO Empty<br>Indicates whether the receive FIFO is empty.<br>0b - Not empty<br>1b - Empty |
| 0<br>SOF | Start of Frame<br>Indicates whether this is the first data word received after PCS assertion.<br>0b - Subsequent data word or RX FIFO is empty (RXEMPTY=1).<br>1b - First data word |

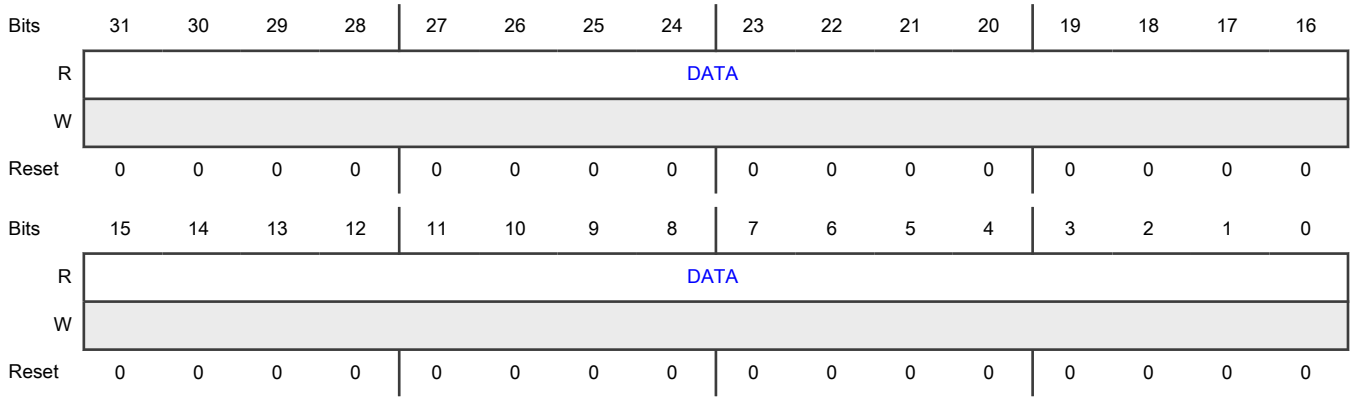
## 67.6.1.19 Receive Data (RDR)

### Offset

| Register | Offset |
|---|---|
| RDR | 74h |

### Function
Pulls the first entry from the receive FIFO.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-0<br>DATA | Receive Data |

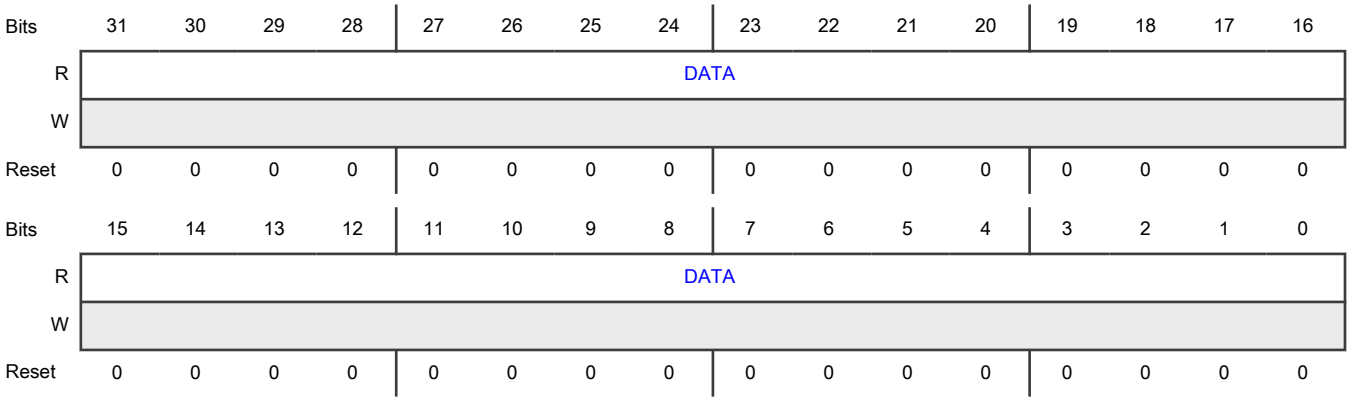
## 67.6.1.20 Receive Data Read Only (RDROR)

### Offset

| Register | Offset |
|---|---|
| RDROR | 78h |

### Function
Returns the first entry in the receive FIFO but does not remove the data from the FIFO.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>DATA | Receive Data |

## 67.6.1.21 Transmit Command Burst (TCBR)

### Offset

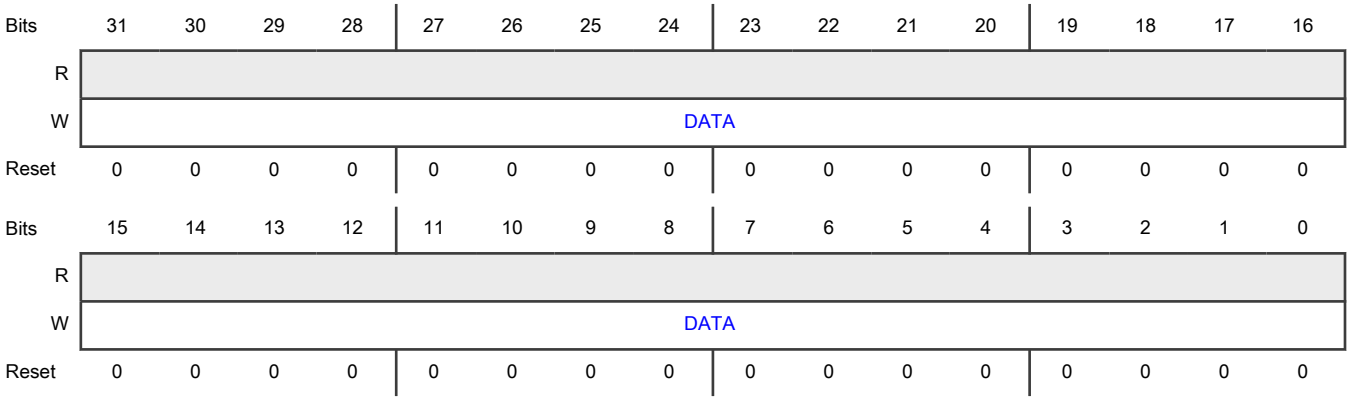
| Register | Offset |
|---|---|
| TCBR | 3FCh |

### Function

Supports burst transfers of command data to the transmit FIFO for use with the DMA controller.

See DMA support registers.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>DATA | Command Data<br>Writes data to Transmit Command (TCR). |

## 67.6.1.22 Transmit Data Burst (TDBR0 - TDBR127)

### Offset

For n = 0 to 127:

| Register | Offset |
|---|---|
| TDBRn | 400h + (n × 4h) |

### Function

Supports burst transfers of data to the transmit FIFO for use with the DMA controller. The size of this register is 512 bytes.

See DMA support registers.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>DATA | Data<br>Writes data to Transmit Data (TDR). |

## 67.6.1.23 Receive Data Burst (RDBR0 - RDBR127)

### Offset

For n = 0 to 127:

| Register | Offset |
|---|---|
| RDBRn | 600h + (n × 4h) |

### Function

Supports burst transfers of data from the receive FIFO. The size of this register is 512 bytes.

See DMA support registers.

### Diagram

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

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

**Fields**

| Field | Function |
|---|---|
| 31-0<br>DATA | Data<br>Reads data from Receive Data (RDR). |

## 67.7 Glossary

| | |
|---|---|
| **PCS** | Peripheral chip select |
| **SCK** | Serial clock |
| **SDI** | Peripheral data in |
| **SDO** | Peripheral data out |
| **SS** | Peripheral select |

# Chapter 68
# Low Power Inter-Integrated Circuit (LPI2C)

## 68.1 Chip-specific LPI2C information

### 68.1.1 LPI2C instances and configuration

This chip has two instances of LPI2C.

**Table 377. LPI2C instances**

| Instance | MCXE315/MCXE316/MCXE317/MCXE31B |
|---|---|
| LPI2C_0 | Yes |
| LPI2C_1 | Yes |

**Table 378. LPI2C configuration**

| Instances | TX FIFO Size | RX FIFO Size | SMBus[1] | Slave Mode Enable |
|---|---|---|---|---|
| LPI2C_0 | 4x11 bit | 4x8 bit | Yes | Yes |
| LPI2C_1 | 4x11 bit | 4x8 bit | Yes | Yes |

1. System Management Bus

- LPI2C slave mode operation supports up to high speed mode = 3.4MHz (3.4 Mbps data rate; effective data rate reduces according to I2C protocol).
- LPI2C master mode operation supports up to fast Mode = 400KHz (400 kbps data rate; effective data rate reduces according to I2C protocol).
- The LPI2C module includes SMBus (System Management Bus) support and DMA support.
- LPI2C RX FIFO is 8-bit and TX FIFO is 11-bit (8bit data + 3bit command)
- LPI2C supports fast data communication as per I2C standard v3.0.

## 68.2 Overview

LPI2C supports an efficient interface to an I2C bus as a controller and target:

- Implements logic support for Standard, Fast, Fast+, HS-mode (target only) and Ultra-Fast modes of operation
- Uses little CPU overhead, with DMA offloading of FIFO register accesses

LPI2C also complies with the System Management Bus (SMBus) Specification, version 3. The SMBus is a single-ended simple two-wire bus, which is typically used for low-bandwidth communications.

The Inter-Integrated Circuit ($I^2C$) serial bus is multi-controller, multi-target, packet-switched, and single-ended, and is often used to attach microcontroller ICs to lower-speed peripheral ICs.

**NOTE**

Terminology in this chapter has been updated to align with I²C-bus specification, Rev. 7.0, as shown in Table 379.

**Table 379. Updated terms**

| Updated term | Deprecated term |
|---|---|
| Controller | Master |
| Target | Slave |

## 68.2.1 Block diagram

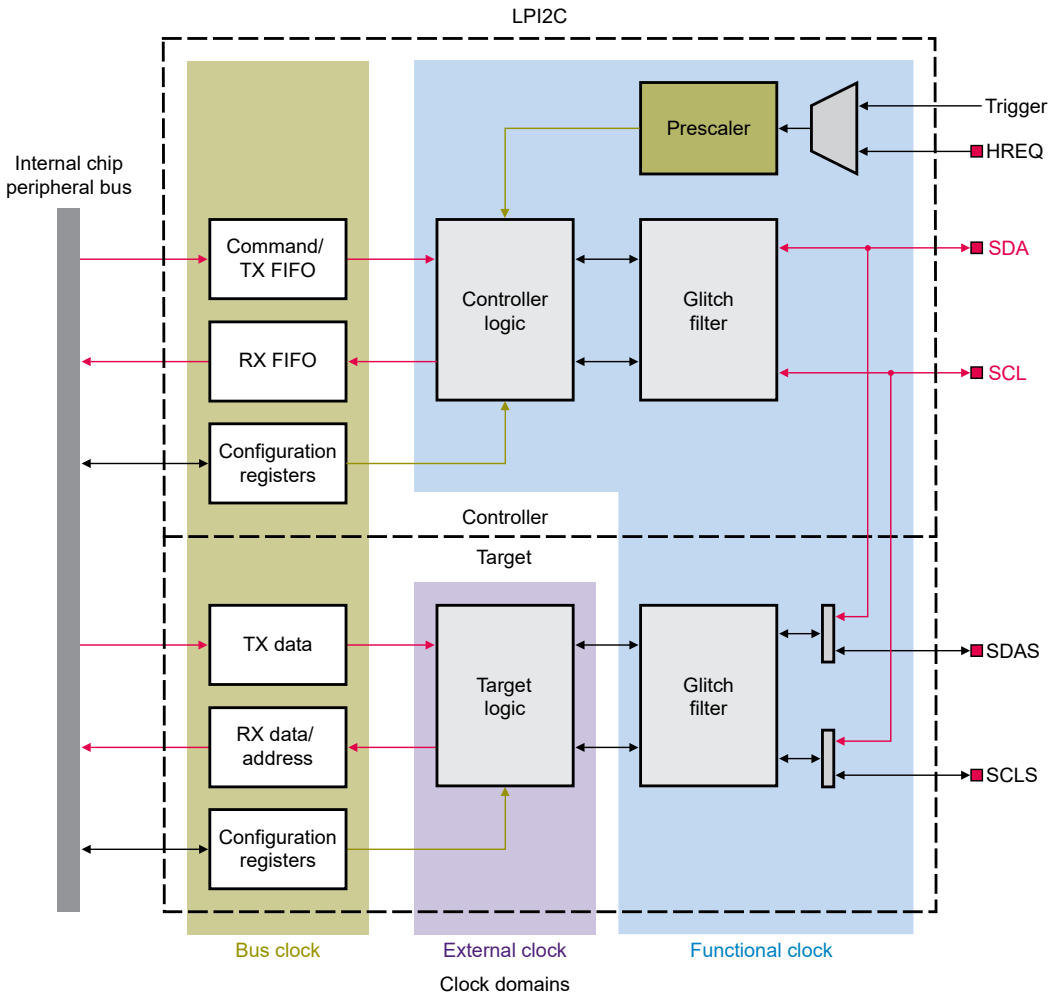



**Figure 362. Block diagram**

## 68.2.2 Features

LPI2C supports:

- Standard, Fast, Fast+ and Ultra Fast modes
- HS mode in target mode
- Multicontroller, including synchronization and arbitration, means that any number of controller nodes can be present. Also, controller and target roles can be changed between messages (after a Stop signal is sent).

- Clock stretching. Used on the SCL line, as an I2C flow control mechanism.
- Arbitration for when the system has more than one controller. When used on the SDA line, ensures that there is only one I2C transmitter at a time.
- General call, seven-bit addressing, and ten-bit addressing
- Software reset, Start byte, and device ID (also require software support)

The LPI2C controller supports:

- Command and transmit FIFO of 4 words (8-bit transmit data + 3-bit command)
- Receive FIFO of 4 words (8-bit receive data).
- Command FIFO waiting for an I2C idle bus before initiating a transfer
- Initiation of repeated Start and Stop conditions and one or more controller-receiver transfers by command FIFO
- Stop condition generation from command FIFO, or automatic generation of Stop condition when the transmit FIFO is empty
- Host request input to control the start time of an I2C bus transfer
- Interrupt generation on data match and unwanted data rejection, via flexible receive data match
- Flags and optional interrupt signals at repeated Start condition, Stop condition, loss of arbitration, unexpected NACK, and command word errors
- Configurable bus idle timeout and pin-stuck-low timeout

The LPI2C target supports:

- Separate I2C target registers to minimize software overhead because of controller or target switching
- 7-bit or 10-bit addressing, address range, SMBus alert, and general call address.
- Transmit data register that supports interrupt or DMA requests
- Receive data register that supports interrupt or DMA requests
- Software-controllable ACK or NACK, with optional clock stretching on ACK or NACK field
- Configurable clock stretching, to avoid transmit-FIFO-underrun and receive-FIFO-overrun errors
- Flags and optional interrupt at end of packet, Stop condition, or bit error detection

## 68.3 Functional description

### 68.3.1 Controller mode

The LPI2C controller logic operates independently from the target logic to perform all controller-mode transfers on the I2C bus.

#### 68.3.1.1 Transmit and Command FIFO commands

The transmit FIFO stores command data to initiate various I2C operations. The following operations can be initiated through commands in the transmit FIFO:

- Start or repeated Start condition with address byte, expecting ACK or NACK.
- Transmit data. This operation is the default for zero-extended-byte writes to the transmit FIFO.
- Receive 1-256 bytes of data. You can configure this operation to discard received data and not to store it in the receive FIFO.
- Stop condition. You can configure this operation to send a Stop condition when the transmit FIFO is empty.

Multiple transmit and receive commands can be inserted between the Start and Stop conditions. To comply with the I2C specification, transmit and receive commands must not be interleaved. The receive data command and the receive data and

discard commands can be interleaved. This interleaving ensures that only the desired received data is stored in the receive FIFO (or compared with the data match logic).

The LPI2C controller automatically transmits a NACK on the last byte of a receive data command. It transmits the NACK unless the next command in the FIFO is also a receive data command. If the transmit FIFO is empty when a receive data command completes, a NACK is also automatically transmitted.

The LPI2C controller supports 10-bit addressing via a (repeated) Start condition, followed by a transmit data byte containing the second address byte, followed by any number of data bytes with the controller transmit data.

A Start or repeated Start condition expecting a NACK (for example, HS mode controller code) must be followed by a Stop or (repeated) Start condition.

#### 68.3.1.2 Controller operations

When LPI2C is enabled, it monitors the I2C bus to detect when the I2C is idle (MSR[BBF]). If either SCL or SDA are low, the I2C bus is no longer considered idle. The bus becomes idle if a Stop condition is detected or if a bus idle timeout is detected (as configured by MCFGR2[BUSIDLE]). After the bus is idle, if the transmit FIFO is not empty and the host request is asserted or disabled, the LPI2C controller initiates a transfer on the bus. This transfer involves the following steps:

1. Wait the bus idle time equal to (MCCR0[CLKLO] + 1) multiplied by the prescaler (MCFGR1[PRESCALE]).
2. Transmit a Start condition and address byte using the timing configuration in Controller Clock Configuration 0 (MCCR0). If an HS mode transfer is configured, the timing configuration from Controller Clock Configuration 1 (MCCR1) is used instead.
3. Perform controller transmit or controller receive transfers, as configured by the transmit FIFO.
4. Transmit NACK on the last byte of a controller receive transfer. This action is performed unless the next command in the transmit FIFO is also a receive data command and the transmit FIFO is not empty.
5. Transmit a repeated Start or Stop condition as configured by the transmit FIFO or MCFGR1[AUTOSTOP]. A repeated Start can change which timing configuration register is used.

When the LPI2C controller is disabled, LPI2C continues emptying the transmit FIFO until a Stop condition is transmitted. (The controller could be disabled due to MCR[MEN] being 0, or automatically due to mode entry.) However, LPI2C no longer stalls the I2C bus by waiting for the transmit or receive FIFO. After the transmit FIFO is empty, LPI2C generates a Stop condition automatically.

The LPI2C controller can stall the I2C bus under certain conditions. This stalling results in SCL pulled low continuously on the first bit of a byte, until these conditions change:

- The LPI2C controller is enabled and busy, the transmit FIFO is empty, and MCFGR1[AUTOSTOP] is 0. The LPI2C controller continues to stall the bus until the transmit FIFO is loaded with more data.
- The LPI2C controller is enabled and receiving data, receive data is not being discarded (due to command or receive data match), and the receive FIFO is full. The LPI2C controller continues to stall the I2C bus until the receive FIFO is emptied.

#### 68.3.1.3 Receive FIFO and data matching

The receive FIFO stores receive data during controller-receiver transfers. You can configure the LPI2C controller to discard received data instead of storing it in the receive FIFO. This option is configured via the command word in the transmit FIFO.

Received data supports a receive data match function that can match received data against one of two bytes, or against a masked data byte. You can configure the data match function to compare only the first one or two data words received since the last (repeated) Start condition. Received data that is already discarded due to the command word cannot cause a data match. It delays the match on the first data word received until after the discarded data is received.

You can configure the receiver match function to discard all received data until a data match is detected, using MCFGR0[RDMO]. Following a data match, write 0 to MCFGR0[RDMO] before writing 0 to MSR[DMF] to allow all subsequent data to be received.

#### 68.3.1.4 Timing parameters

The LPI2C controller can configure the following timing parameters. Parameters are configured separately for HS mode (Controller Clock Configuration 1 (MCCR1)) and other modes (Controller Clock Configuration 0 (MCCR0)). This separation allows the HS mode controller code to be sent using regular timing parameters. Then it allows a switch to HS mode timing (following a repeated Start) until the next STOP condition.

Configure the LPI2C controller timing parameters, measured in LPI2C functional clock cycles, as shown in Table 380. You must configure these parameters to meet the I2C timing specification for the required mode.

**Table 380. Timing parameters**

| I2C specification timing parameter | I2C specification timing symbol | LPI2C timing parameter (in LPI2C functional clock cycles) |
|---|---|---|
| SCL clock period | tSCL | (CLKHI + CLKLO + 2 + SCL_LATENCY) × (2 ^ PRESCALE) |
| Hold time (repeated) Start condition | tHD:STA | (SETHOLD +1) × (2 ^ PRESCALE) |
| Low period of the SCL clock | tLOW | (CLKLO + 1) × (2 ^ PRESCALE) |
| High period of the SCL clock | tHIGH | (CLKHI + 1 + SCL_LATENCY) × (2 ^ PRESCALE) |
| Setup time for a repeated Start condition or Stop condition | tSU:STA, tSU:STO | (SETHOLD + 1 + SCL_LATENCY) × (2 ^ PRESCALE) |
| Data hold time | tHD:DAT | (DATAVD + 1) × (2 ^ PRESCALE) |
| Data setup time | tSU:DAT | (SDA_LATENCY + 1) × (2 ^ PRESCALE) |
| Bus free time between a Stop and Start condition | tBUF | (CLKLO + 1 + SDA_LATENCY) × (2 ^ PRESCALE) |
| Data valid time, data valid acknowledge time | tVD:DAT, tVD:ACK | (DATAVD + 1) × (2 ^ PRESCALE) |

Table 381 defines the latency parameters. These parameters assume that the risetime is less than one LPI2C functional clock cycle. The risetime depends on a number of factors, including the I/O propagation delay, the I2C bus loading, and the external pullup resistor sizing. A larger risetime increases the number of cycles that the signal takes to propagate through the synchronizer (and glitch filter), which increases the latency.

**Table 381. Synchronization latency**

| Timing parameter | Timing definition |
|---|---|
| SCL_LATENCY | ROUNDDOWN ((2 + FILTSCL + SCL_RISETIME) ÷ (2 ^ PRESCALE)) |
| SDA_LATENCY | ROUNDDOWN ((2 + FILTSDA + SDA_RISETIME) ÷ (2 ^ PRESCALE)) |

The following timing restrictions must be enforced to avoid unexpected Start or Stop conditions on the I2C bus. These restrictions also avoid unexpected Start or Stop conditions detected by the LPI2C controller. The timing restrictions can be summarized as "SDA cannot change when SCL is high outside a transmitted (repeated) Start or Stop condition."

**Table 382. LPI2C timing parameter restrictions**

| Timing parameter | Minimum | Maximum | Comment |
|---|---|---|---|
| CLKLO | 03h | — | CLKLO x (2 ^ PRESCALE) > SCL_LATENCY |

*Table continues on the next page...*

Table 382. LPI2C timing parameter restrictions (continued)

| Timing parameter | Minimum | Maximum | Comment |
|---|---|---|---|
| CLKHI | 01h | — | Configure CLKHI to meet the duty cycle requirements in the I2C specification |
| SETHOLD | 02h | — | SETHOLD × (2 ^ PRESCALE) > SDA_LATENCY |
| DATAVD | 01h | CLKLO – SDA_LATENCY – 1 | Configure DATAVD to meet the data hold requirement in the I2C specification |
| FILTSCL | 00h | [CLKLO × (2 ^ PRESCALE)] – 3 | FILTSCL and FILTSDA are the only parameters not multiplied by (2 ^ PRESCALE) |
| FILTSDA | FILTSCL | [CLKLO × (2 ^ PRESCALE)] – 3 | Configuring FILTSDA greater than FILTSCL can delay the SDA input to compensate for board level skew |
| BUSIDLE | (CLKLO + SETHOLD + 2) × 2 | — | Must also be greater than (CLKHI + 1) |

See the UM10204, *I2C-bus specification and user manual*.

See Application information for example LPI2C timing configurations.

#### 68.3.1.5 Error conditions

The LPI2C controller monitors errors while it is active. The following conditions generate an error flag and block a new Start condition from being sent, until the flag is cleared by software:

- A Start or Stop condition is detected and is not generated by the LPI2C controller (MSR[ALF] becomes 1).
- Transmitting data on SDA and different values are received (MSR[ALF] becomes 1).
- NACK is detected when transmitting data, and MCFGR1[IGNACK] is 0 (MSR[NDF] becomes 1).
- NACK is detected and is expecting ACK for the address byte, and MCFGR1[IGNACK] is 0 (MSR[NDF] becomes 1).
- ACK is detected and is expecting NACK for the address byte, and MCFGR1[IGNACK] is 0 (MSR[NDF] becomes 1).
- Transmit FIFO is requesting to transmit or receive data without a Start condition (MSR[FEF] becomes 1).
- SCL (or SDA if MCFGR1[TIMECFG] is 1) is low for (MCFGR2[TIMELOW] × 256) prescaler cycles without a pin transition (MSR[PLTF] becomes 1).

You must respond to MSR[PLTF] to terminate the existing command. You can terminate the command cleanly by writing 0 to MCR[MEN], or you can terminate it abruptly by writing 1 to MCR[RST].

You can use MCFGR2[BUSIDLE] to force the I2C bus to be considered idle when SCL and SDA remain high for (BUSIDLE + 1) prescaler cycles. The bus is considered idle when the LPI2C controller is first enabled. When BUSIDLE is configured greater than zero, then SCL or SDA must be high for (BUSIDLE + 1) prescaler cycles before the I2C bus is considered idle.

#### 68.3.1.6 Pin configuration

| Configuration | Description |
|---|---|
| Open-drain support | The LPI2C controller defaults to open-drain configuration of the SDA and SCL pins. Support for true open drain depends on the specific device, and requires the pins where LPI2C pins are muxed to support true open drain. |
| HS mode support | Support for HS mode depends on the specific device. This mode requires the SCL pin to support the current source pullup required in the I2C specification. |
| Ultra-Fast mode support | The LPI2C controller supports the output-only push-pull function required for I2C Ultra-Fast mode using the SDA and SCL pins. Support for Ultra-Fast mode also requires MCFGR1[IGNACK] to be 1. |
| Push-pull two-wire support | A push-pull two-wire configuration is available to the LPI2C controller. If LPI2C is the only controller and all I2C pins on the bus are at the same voltage, this configuration may support a partial HS mode. A partial HS mode, not a full HS mode, because this configuration actively drives high rather than enabling a current service pull-up. This configuration sets the SCL pin as push-pull for every clock except the first clock pulse, to allow HS-mode-compatible targets to perform clock stretching. In this mode, the SDA pin is tristated for controller-receive data bits and controller-transmit ACK/NACK bits, and is configured as push-pull at other times. To avoid the risk of contention when SDA is push-pull, configure the pin for open-drain operation, as part of the device-specific configuration. |
| Push-pull four-wire support | The push-pull four-wire configuration separates the SCL input data and output data into separate pins. It also separates the SDA input data and output data into separate pins. The SCL/SDA pins are used for input data; the SCLS/SDAS pins are used for output data, with configurable polarity. This configuration simplifies external connections when connecting the LPI2C to the I2C bus through external level shifters or discrete components. When using this four-wire configuration, the LPI2C controller logic and LPI2C target logic cannot connect to separate I2C buses. |

### 68.3.2 Target mode

To perform all target mode transfers on the I2C bus, the LPI2C target logic operates independently from the LPI2C controller logic.

#### 68.3.2.1 Address matching

You can configure the LPI2C target:

- To match one of two addresses, using either 7-bit or 10-bit addressing modes for each address.
- To match a range of addresses in either 7-bit or 10-bit addressing modes.
- To match the general call address and generate appropriate flags.
- To match the SMBus alert address and generate appropriate flags.
- To detect the HS mode controller code address, and to disable the digital filters and output valid delay time until the next Stop condition is detected.

After a valid address is matched, the LPI2C target automatically performs target-transmit or target-receive transfers until:

- A NACK is detected (unless SCFGR1[IGNACK] becomes 1).
- A bit error is detected (the LPI2C target is driving SDA, but a different value is sampled).
- A (repeated) Start or Stop condition is detected.

#### 68.3.2.2 Transmit and receive data

Target Transmit Data (STDR) and Target Receive Data (SRDR) are double-buffered and only update during a target-transmit and target-receive transfer, respectively.

You can configure the target address that was received to be read from SRDR (for example, when using DMA to transfer data) or from Target Address Status (SASR).

You can configure STDR to request data only after a target-transmit transfer is detected. You can also configure it to request new data whenever STDR is empty.

Write to STDR only when SSR[TDF] is set.

Read SRDR only when SSR[RDF] is set, or when SSR[AVF] is set and SCFGR1[RXCFG] = 1.

Read SASR only when SSR[AVF] is set.

#### 68.3.2.3 Clock stretching

The LPI2C target supports many configurable options for clock stretching. You can configure these conditions to perform clock stretching:

- SSR[AVF] is set during the ninth clock pulse of the address byte.
- SSR[TDF] is set during the ninth clock pulse of a target-transmit transfer.
- SSR[RDF] is set during the ninth clock pulse of a target-receive transfer.
- SSR[TAF] is set during the eighth clock pulse of an address byte or a target-receive transfer. In HS mode, this option is disabled.
- Clock stretching can be extended for a number of cycles equal to the value of SCFGR2[CLKHOLD] cycles. This stretching allows additional setup time to sample the SDA pin externally. In HS mode, this option is disabled.

When clock stretching is enabled, clock stretching extends for one peripheral bus clock cycle after SDA updates, unless extended by the SCFGR2[CLKHOLD] configuration.

#### 68.3.2.4 Timing parameters

The LPI2C target can configure the following timing parameters:

- SDA data valid time from SCL negation to SDA update
- SCL hold time when clock stretching is enabled to increase setup time when sampling SDA externally
- SCL glitch filter time
- SDA glitch filter time

These parameters are disabled when SCR[FILTEN] is 0, when SCR[FILTDZ] is 1 in Doze mode, and when LPI2C target detects HS mode. When disabled, the LPI2C target is clocked directly from the I2C bus. In this case, the target may not satisfy all timing requirements of the I2C specification (such as SDA minimum hold time in Standard/Fast mode).

The LPI2C target places the following restrictions on the timing parameters:

- You must configure SCFGR2[FILTSDA] to be greater than or equal to SCFGR2[FILTSCL] (unless compensating for board level skew between SDA and SCL).
- You must configure SCFGR2[DATAVD] to be less than the minimum SCL low period.

#### 68.3.2.5 Error conditions

The LPI2C target can flag the following error conditions:

- SSR[BEF] is set when the LPI2C target is driving SDA but it samples a different value than what is expected.
- SSR[FEF] is set due to a transmit data underrun or a receive data overrun. To eliminate the possibility of underrun and overrun, enable clock stretching.
- SSR[FEF] is also set due to an address overrun, but only when SCFGR1[RXCFG] is 1. To eliminate the possibility of overrun, enable clock stretching.

The LPI2C target does not implement a timeout due to SCL or SDA being stuck low. If this detection is required, use the LPI2C controller logic so you can reset the LPI2C target when this condition is detected.

### 68.3.3 Low-power modes

LPI2C remains functional during low-power modes, if MCR[DOZEN] = 0 and LPI2C uses an external or internal clock source that remains enabled. LPI2C can generate an interrupt or DMA request to cause a wake-up from low-power modes.

You can configure LPI2C to be disabled in low-power modes when MCR[DOZEN] = 1. In this case, LPI2C waits for the current transfer to complete any pending operation.

**NOTE**

See the chip-specific information for low-power modes available on your chip.

### 68.3.4 Debug mode

**Table 383. Debug mode**

| Mode | LPI2C operation |
|---|---|
| Debug | If MCR[DBGEN] = 1, can continue operating in Debug mode. |

### 68.3.5 Peripheral triggers

The connection of the LPI2C peripheral triggers to other peripherals depends upon the specific device being used.

**Table 384. LPI2C triggers**

| Trigger | Description |
|---|---|
| Controller output trigger | Generates an output trigger that can be connected to other peripherals on the device. The controller output trigger asserts on either a repeated Start or a Stop condition. The trigger remains asserted for one cycle of the LPI2C functional clock divided by MCFGR1[PRESCALE]. |
| Target output trigger | Generates an output trigger that can be connected to other peripherals on the device. The target output trigger asserts on either a repeated Start or a Stop condition that occurs after a target address match. The target output trigger remains asserted until the next target SCL pin negation. |
| Input trigger | To control the start of a LPI2C bus transfer, the LPI2C input trigger can be selected instead of the HREQ input. The input trigger is synchronized. To be detected, the input trigger must assert for at least two cycles of the LPI2C functional clock divided by the value of MCFGR1[PRESCALE]. When LPI2C is busy, the HREQ input (and therefore the input trigger) is ignored. |

### 68.3.6 Clocking

**Table 385. LPI2C clocks**

| Clock | Description |
|---|---|
| LPI2C functional clock | The LPI2C functional clock is asynchronous to the bus clock. It can remain enabled in low-power modes to support I2C bus transfers by the LPI2C controller. The functional clock is also used by the LPI2C target to support digital filter and data hold time configurations. The LPI2C controller divides the functional clock by a prescaler (MCFGR1[PRESCALE]) and the resulting frequency must be at least eight times faster than the I2C bus bandwidth. |

*Table continues on the next page...*

Table 385. LPI2C clocks (continued)

| Clock | Description |
|---|---|
| External clock | The LPI2C target logic is clocked directly from the external pins. These pins are SCL and SDA, or SCLS and SDAS if the controller and target are implemented on separate pins). This clocking allows the LPI2C target to remain operational, even when the LPI2C functional clock is disabled.<br>**NOTE**<br>If the LPI2C functional clock is disabled, the LPI2C target digital filter must be disabled. This condition can affect compliance with some timing parameters of the I2C specification, such as data hold time. |
| Bus clock | The bus clock is only used for bus accesses to the control and configuration registers. The bus clock frequency must be sufficient to support the data bandwidth requirements of the LPI2C controller and target registers. |

For chip-specific clocking information, see the Clocking chapter.

### 68.3.7 Reset

Table 386. LPI2C resets

| Reset | Description |
|---|---|
| Chip reset | The logic and registers for the LPI2C controller and target are reset to their default states after a chip reset. |
| Software reset | The LPI2C controller implements a software reset field in its control register. MCR[RST] resets all controller logic and registers to their default states, except for Controller Control (MCR) itself.<br>The LPI2C target implements a software reset field in its control register. SCR[RST] resets all target logic and registers to their default states, except for Target Control (SCR) itself. |
| FIFO reset | The LPI2C controller implements write-only control fields that reset the transmit FIFO (MCR[RTF]) and receive FIFO (MCR[RRF]). After a FIFO is reset, that FIFO is empty.<br>The LPI2C target implements write-only control fields that reset the transmit data register (SCR[RTF]) and receive data register (SCR[RRF]). After a data register is reset, that data register is empty. |

### 68.3.8 Interrupts and DMA requests

Depending on the configuration, interrupts and DMA requests can be combined:

- LPI2C controller and target interrupts
- LPI2C controller and target transmit DMA requests
- LPI2C controller and target receive DMA requests

#### 68.3.8.1 Controller mode

Table 387 lists the Controller mode sources that can generate LPI2C controller interrupts and LPI2C controller transmit and receive DMA requests.

Table 387. Controller interrupts and DMA requests

| Status flag | Description | Can generate | | |
|---|---|---|---|---|
| | | Interrupt? | DMA request? | Low-power wake-up? |
| Transmit Data Flag (MSR[TDF]) | Data can be written to transmit FIFO, as configured by MFCR[TXWATER]. | Y | TX | Y |
| Receive Data Flag (MSR[RDF]) | Data can be read from the receive FIFO, as configured by MFCR[RXWATER]. | Y | RX | Y |
| End Packet Flag (MSR[EPF]) | Controller has transmitted a repeated Start or Stop condition. | Y | N | Y |
| Stop Detect Flag (MSR[SDF]) | Controller has transmitted a Stop condition . | Y | N | Y |
| NACK Detect Flag (MSR[NDF]) | During an address byte, the controller expects an ACK but detects a NACK.<br>During an address byte, the controller expects a NACK but detects an ACK.<br>During a controller-transmitter data byte, the controller detects a NACK. | Y | N | Y |
| Arbitration Lost Flag (MSR[ALF]) | The controller lost arbitration due to a Start or Stop condition detected at the wrong time, or the controller was transmitting data but received data different from the data that was transmitted. | Y | N | Y |
| FIFO Error Flag (MSR[FEF]) | The controller expects a Start condition in the command FIFO, but the next entry in the command FIFO is not a Start condition. | Y | N | Y |
| Pin Low Timeout Flag (MSR[PLTF]) | Pin low timeout is enabled and SCL (or SDA, if configured) is low for longer than the configured timeout. | Y | N | Y |
| Data Match Flag (MSR[DMF]) | The received data matches the configured data match, but the received data is not discarded due to a command FIFO entry. | Y | N | Y |
| Controller Busy Flag (MSR[MBF]) | LPI2C controller is busy transmitting or receiving data. | N | N | N |
| Bus Busy Flag (MSR[BBF]) | LPI2C controller is enabled and activity is detected on the I2C bus, but no Stop condition is detected and no bus idle timeout (if enabled) occurred. | N | N | N |

#### 68.3.8.2 Target mode

Table 388 lists the target mode sources that can generate LPI2C target interrupts and LPI2C target transmit and receive DMA requests.

Table 388. Target interrupts and DMA requests

| Status flag | Description | Can generate | | |
|---|---|---|---|---|
| | | Interrupt? | DMA request? | Low-power wake-up? |
| Transmit Data Flag (SSR[TDF]) | Data can be written to Target Transmit Data (STDR). | Y | TX | Y |
| Receive Data Flag (SSR[RDF]) | Data can be read from Target Receive Data (SRDR). | Y | RX | Y |
| Address Valid Flag (SSR[AVF]) | Address can be read from Target Address Status (SASR). | Y | RX | Y |
| Transmit ACK Flag (SSR[TAF]) | ACK or NACK can be written to Target Transmit ACK (STAR). | Y | N | Y |
| Repeated Start Flag (SSR[RSF]) | Target has detected an address match followed by a repeated Start condition. | Y | N | Y |
| Stop Detect Flag (SSR[SDF]) | Target has detected an address match followed by a Stop condition. | Y | N | Y |
| Bit Error Flag (SSR[BEF]) | Target was transmitting data, but received data is different from what was transmitted. | Y | N | Y |
| FIFO Error Flag (SSR[FEF]) | This flag is set by:<br>• Transmit data underrun<br>• Receive data overrun<br>• Address status overrun when SCFGR1[RXCFG] = 1<br>This flag can only be set when clock stretching is disabled. | Y | N | Y |
| Address Match 0 Flag (SSR[AM0F]) | Target detected an address match SAMR[ADDR0]. | Y | N | N |
| Address Match 1 Flag (SSR[AM1F]) | Target detected an address match with SAMR[ADDR1] or using an address range. | Y | N | N |
| General Call Flag (SSR[GCF]) | Target detected an address match with the general call address. | Y | N | N |
| SMBus Alert Response Flag (SSR[SARF]) | Target detected an address match with the SMBus alert address. | Y | N | N |
| Target Busy Flag (SSR[SBF]) | LPI2C target is busy receiving an address byte or is transmitting or receiving data. | N | N | N |
| Bus Busy Flag (SSR[BBF]) | LPI2C target is enabled and a Start condition is detected on I2C bus, but no Stop condition detected. | N | N | N |

## 68.4 External signals

**Table 389. External signals**

| Signal | Name | Two-wire scheme | Four-wire scheme | Direction |
|---|---|---|---|---|
| SCL | LPI2C clock line | SCL | In Four-Wire mode, this pin is the SCL input pin. | Input or output |
| SDA | LPI2C data line | SDA | In Four-Wire mode, this pin is the SDA input pin. | Input or output |
| SCLS | Secondary I2C clock line | Not used | In Four-Wire mode, this pin is the SCLS output pin. If LPI2C controller/target are configured to use separate pins, then this pin is the LPI2C target SCL pin. | Input or output |
| SDAS | Secondary I2C data line | Not used | In Four-Wire mode, this pin is the SDAS output pin. If LPI2C controller/target are configured to use separate pins, then this pin is the LPI2C target SDA pin. | Input or output |
| HREQ | Host request | If host request is asserted and the I2C bus is idle, then it initiates an LPI2C controller transfer.<br>HREQ is an additional pin separate from the two-wire or four-wire scheme. | | Input |

Figure 363 shows the two-signal connection.

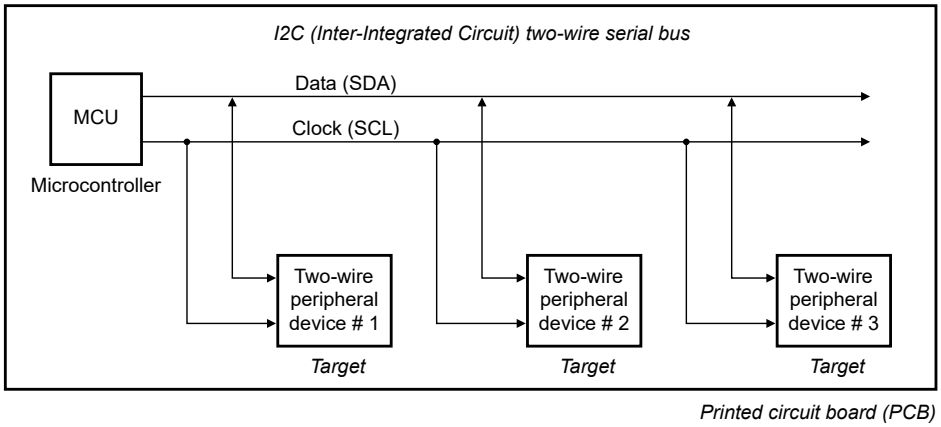



**Figure 363. I²C two-wire serial bus**

Figure 364 shows a possible four-signal connection.

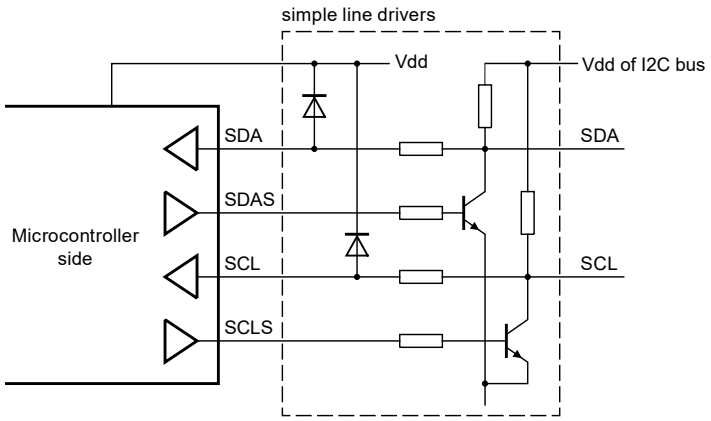



Figure 364. I2C four-wire serial bus

## 68.5 Initialization

To initialize the LPI2C controller:

1. Configure Controller Configuration 0 (MCFGR0)–Controller Configuration 3 (MCFGR3) as required by the application.
2. Configure Controller Clock Configuration 0 (MCCR0) and Controller Clock Configuration 1 (MCCR1) to satisfy the timing requirements of the I2C mode supported by the application.
3. Enable controller interrupts and DMA requests as required by the application.
4. Enable the LPI2C controller by writing 1 to MCR[MEN].

To initialize the LPI2C target:

1. Configure Target Address Match (SAMR) with the I2C address of the target location on the I2C bus.
2. Configure Target Configuration 1 (SCFGR1) as required by the application.
3. Configure Target Configuration 2 (SCFGR2) to satisfy the timing requirements of the I2C mode supported by the application.
4. Enable target interrupts and DMA requests as required by the application.
5. Enable the LPI2C target by writing 1 to SCR[SEN].

## 68.6 Application information

Configure the I2C timing parameters to meet the requirements of the I2C specification. This configuration depends on the supported mode and LPI2C functional clock frequency. When switching between two modes using different clock configuration registers (for example, Fast mode and HS mode), MCFGR1[PRESCALE] must remain constant between the modes.

Table 390. Example timing configurations

| I2C mode | Clock frequency | Baud rate | PRESCALE | FILTSCL / FILTSDA | SETHOLD | CLKLO | CLKHI | DATAVD |
|---|---|---|---|---|---|---|---|---|
| Standard | 8 MHz | 100 kbit/s | 0h | 0h/0h | 24h | 28h | 24h | 02h |
| Standard | 48 MHz | 100 kbit/s | 2h | 1h/1h | 37h | 3Fh | 37h | 03h |
| Standard | 60 MHz | 100 kbit/s | 2h | 1h/1h | 45h | 50h | 44h | 04h |
| Fast | 8 MHz | 400 kbit/s | 0h | 0h/0h | 04h | 0Bh | 05h | 02h |

*Table continues on the next page...*

Table 390. Example timing configurations (continued)

| I2C mode | Clock frequency | Baud rate | PRESCALE | FILTSCL / FILTSDA | SETHOLD | CLKLO | CLKHI | DATAVD |
|---|---|---|---|---|---|---|---|---|
| Fast+ | 8 MHz | 1 Mbit/s | 0h | 0h/0h | 02h | 03h | 01h | 01h |
| Fast | 48 MHz | 400 kbit/s | 0h | 1h/1h | 1Dh | 3Eh | 35h | 0Fh |
| Fast | 48 MHz | 400 kbit/s | 2h | 1h/1h | 07h | 11h | 0Bh | 03h |
| Fast+ | 48 MHz | 1 Mbit/s | 2h | 1h/1h | 03h | 06h | 04h | 04h |
| HS | 48 MHz | 3.2 Mbit/s | 0h | 0h/0h | 07h | 08h | 03h | 01h |
| Fast | 60 MHz | 400 kbit/s | 1h | 2h/2h | 11h | 28h | 1Fh | 08h |
| Fast+ | 60 MHz | 1 Mbit/s | 1h | 2h/2h | 07h | 0Fh | 0Bh | 01h |
| HS | 60 MHz | 3.33 Mbit/s | 1h | 0h/0h | 04h | 04h | 02h | 01h |

## 68.7 Memory map and registers

### 68.7.1 LPI2C register descriptions

Writing to a read-only register or reading from a write-only register can cause bus errors. This module does not check whether programmed values in the registers are correct; you must ensure that valid programmed values are written to the registers.

#### 68.7.1.1 LPI2C memory map

LPI2C_0 base address: 4035_0000h

LPI2C_1 base address: 4035_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID (VERID) | 32 | R | 0102_0003h |
| 4h | Parameter (PARAM) | 32 | R | 0000_0202h |
| 10h | Controller Control (MCR) | 32 | RW | 0000_0000h |
| 14h | Controller Status (MSR) | 32 | RW | 0000_0001h |
| 18h | Controller Interrupt Enable (MIER) | 32 | RW | 0000_0000h |
| 1Ch | Controller DMA Enable (MDER) | 32 | RW | 0000_0000h |
| 20h | Controller Configuration 0 (MCFGR0) | 32 | RW | 0000_0000h |
| 24h | Controller Configuration 1 (MCFGR1) | 32 | RW | 0000_0000h |
| 28h | Controller Configuration 2 (MCFGR2) | 32 | RW | 0000_0000h |
| 2Ch | Controller Configuration 3 (MCFGR3) | 32 | RW | 0000_0000h |
| 40h | Controller Data Match (MDMR) | 32 | RW | 0000_0000h |
| 48h | Controller Clock Configuration 0 (MCCR0) | 32 | RW | 0000_0000h |
| 50h | Controller Clock Configuration 1 (MCCR1) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 58h | Controller FIFO Control (MFCR) | 32 | RW | 0000_0000h |
| 5Ch | Controller FIFO Status (MFSR) | 32 | R | 0000_0000h |
| 60h | Controller Transmit Data (MTDR) | 32 | W | 0000_0000h |
| 70h | Controller Receive Data (MRDR) | 32 | R | 0000_4000h |
| 110h | Target Control (SCR) | 32 | RW | 0000_0000h |
| 114h | Target Status (SSR) | 32 | RW | 0000_0000h |
| 118h | Target Interrupt Enable (SIER) | 32 | RW | 0000_0000h |
| 11Ch | Target DMA Enable (SDER) | 32 | RW | 0000_0000h |
| 124h | Target Configuration 1 (SCFGR1) | 32 | RW | 0000_0000h |
| 128h | Target Configuration 2 (SCFGR2) | 32 | RW | 0000_0000h |
| 140h | Target Address Match (SAMR) | 32 | RW | 0000_0000h |
| 150h | Target Address Status (SASR) | 32 | R | 0000_4000h |
| 154h | Target Transmit ACK (STAR) | 32 | RW | 0000_0000h |
| 160h | Target Transmit Data (STDR) | 32 | W | 0000_0000h |
| 170h | Target Receive Data (SRDR) | 32 | R | 0000_4000h |

#### 68.7.1.2 Version ID (VERID)

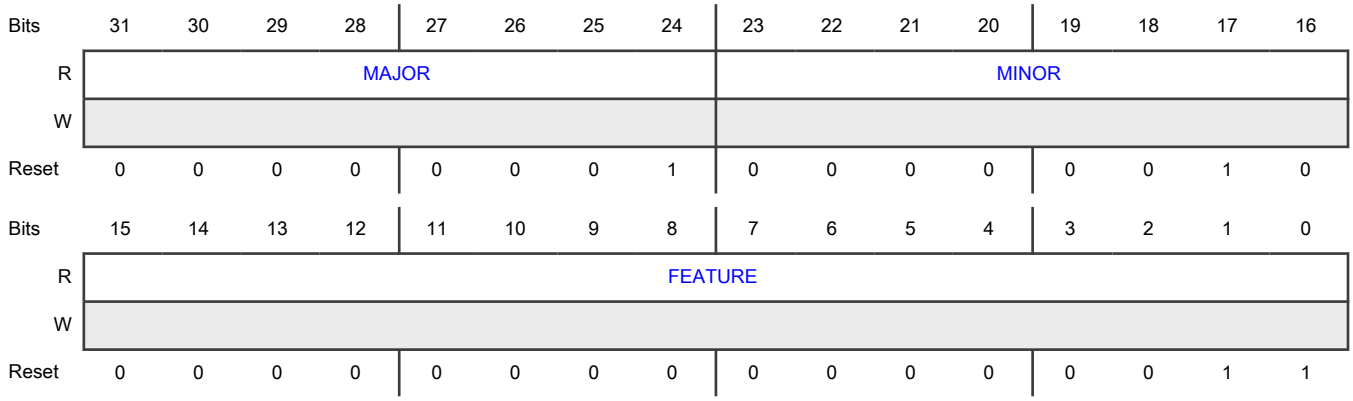
**Offset**

| Register | Offset |
|---|---|
| VERID | 0h |

**Function**

Contains version numbers for the module design and feature set.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>Returns the major version number for the module design specification. |
| 23-16<br>MINOR | Minor Version Number<br>Returns the minor version number for the module design specification. |
| 15-0<br>FEATURE | Feature Specification Number<br>Returns the feature set number.<br>0000_0000_0000_0010b - Controller only, with standard feature set<br>0000_0000_0000_0011b - Controller and target, with standard feature set |

## 68.7.1.3 Parameter (PARAM)

### Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

### Function

Contains parameter values implemented in the module.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-12<br>— | Reserved |
| 11-8<br>MRXFIFO | Controller Receive FIFO Size<br>Configures the number of words in the controller receive FIFO to $2^{MRXFIFO}$. |
| 7-4<br>— | Reserved |
| 3-0<br>MTXFIFO | Controller Transmit FIFO Size<br>Configures the number of words in the controller transmit FIFO to $2^{MTXFIFO}$. |

## 68.7.1.4 Controller Control (MCR)

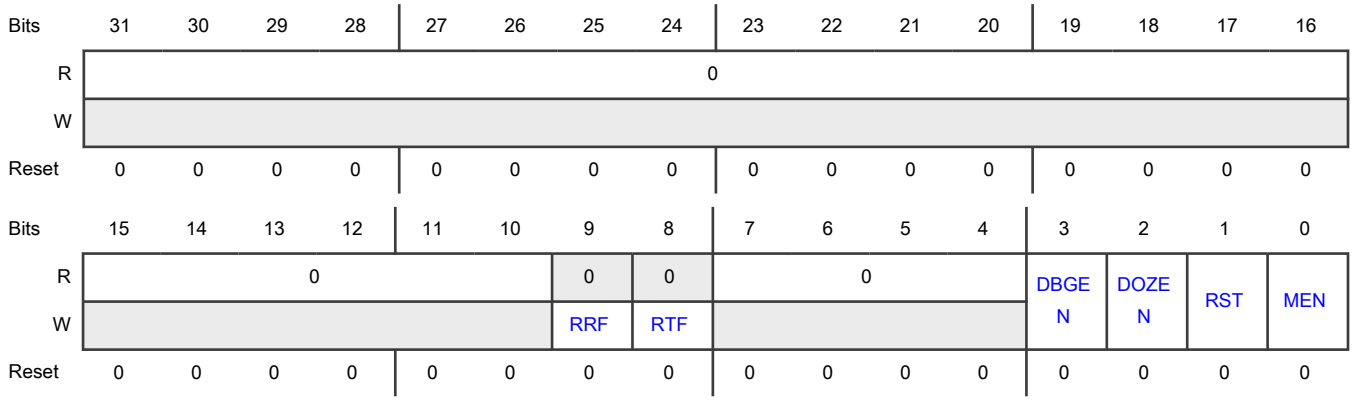
### Offset

| Register | Offset |
|---|---|
| MCR | 10h |

### Function

Contains resets, debug enable, and other controller control settings.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>RRF | Reset Receive FIFO<br>Resets the receive FIFO in Controller Receive Data (MRDR).<br>0b - No effect<br>1b - Reset receive FIFO |
| 8<br>RTF | Reset Transmit FIFO<br>Resets the transmit FIFO in Controller Transmit Data (MTDR).<br>0b - No effect<br>1b - Reset transmit FIFO |
| 7-4<br>— | Reserved |
| 3<br>DBGEN | Debug Enable<br>Enables the controller in Debug mode.<br>0b - Disable<br>1b - Enable |
| 2<br>DOZEN | Doze Mode Enable<br>Enables the controller in Doze mode.<br>0b - Enable<br>1b - Disable |
| 1 | Software Reset |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| RST | Resets all internal controller logic and registers except Controller Control (MCR).<br>This field remains 1 (enabled) until you write 0 to it. The reset takes effect immediately and remains asserted until negated by software. There is no minimum delay required before clearing the software reset.<br>0b - No effect<br>1b - Reset |
| 0<br>MEN | Controller Enable<br>Enables the controller logic.<br>0b - Disable<br>1b - Enable |

### 68.7.1.5 Controller Status (MSR)

#### Offset

| Register | Offset |
|---|---|
| MSR | 14h |

#### Function

Contains status flags for transmit and receive data, for start and stop conditions, and for bus and controller busy or idle status.

#### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | DMF | PLTF | FEF | ALF | NDF | SDF | EPF | 0 | | | | | | RDF | TDF |
| W | | W1C | W1C | W1C | W1C | W1C | W1C | W1C | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

#### Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 25<br>BBF | Bus Busy Flag<br>Specifies whether the I2C bus is busy.<br>0b - Idle<br>1b - Busy |
| 24<br>MBF | Controller Busy Flag<br>Specifies whether the I2C controller is busy.<br>0b - Idle<br>1b - Busy |
| 23-16<br>— | Reserved |
| 15<br>— | Reserved |
| 14<br>DMF | Data Match Flag<br>Indicates whether the received data matches MDMR[MATCH0] or MDMR[MATCH1] (as configured by MCFGR1[MATCFG]). Received data discarded due to MTDR[CMD] does not cause this flag to set.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Matching data not received<br>1b - Matching data received<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 13<br>PLTF | Pin Low Timeout Flag<br>Indicates whether pin low timeout has occurred. Sets when the SCL or SDA input is low for more than the number of PINLOW cycles defined by MCFGR3[PINLOW], even when the LPI2C controller is idle.<br>You must resolve the pin low condition via software. PLTF cannot be cleared as long as the pin low timeout continues. Before LPI2C can initiate a Start condition, you must clear this flag.<br>See MCFGR1[TIMECFG] for the SCL and/or SDA timeout settings.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Pin low timeout did not occur<br>1b - Pin low timeout occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 12<br>FEF | FIFO Error Flag<br>Detects the LPI2C controller's attempt to send or receive data without first generating a (repeated) Start condition. This error can occur when the transmit FIFO underflows when MCFGR1[AUTOSTOP] = 1. When this flag is set, the LPI2C controller sends a Stop condition (if busy). The controller does not initiate a new Start condition until the flag is cleared.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No FIFO error<br>1b - FIFO error<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 11<br>ALF | Arbitration Lost Flag<br>Indicates whether arbitration is lost. Either of these conditions sets this flag:<br>• The LPI2C controller transmits a logic 1 and detects a logic 0 on the I2C bus.<br>• The LPI2C controller detects a Start or Stop condition when the LPI2C controller is transmitting data.<br>When ALF is set, the LPI2C controller releases the I2C bus (goes idle), and the LPI2C controller does not initiate a new Start condition until the ALF is cleared.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Controller did not lose arbitration<br>1b - Controller lost arbitration<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 10 | NACK Detect Flag |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| NDF | Indicates whether an unexpected NACK has been detected. This flag is set when the LPI2C controller detects a NACK it was not expecting when transmitting an address or data. When set, the controller does not initiate a new Start condition until this flag is cleared. If a NACK is expected for a given address (as configured by the command word), this flag is set if a NACK is not generated.<br>When this flag is set, the LPI2C controller automatically transmits a Stop condition if MCFGR1[AUTOSTOP] = 1, or if the transmit FIFO is not empty.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No unexpected NACK detected<br>1b - Unexpected NACK detected<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 9<br>SDF | Stop Detect Flag<br>Indicates whether the LPI2C controller has generated a Stop condition.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No Stop condition generated<br>1b - Stop condition generated<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 8<br>EPF | End Packet Flag<br>Indicates whether the LPI2C controller has generated a repeated Start condition or a Stop condition. When the controller first generates a Start condition, this flag is not set.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No Stop or repeated Start generated<br>1b - Stop or repeated Start generated<br>When writing |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No effect<br>1b - Clear the flag |
| 7-2<br>— | Reserved |
| 1<br>RDF | Receive Data Flag<br>Indicates whether the receive data is ready. This flag is set when the number of words in the receive FIFO is greater than MFCR[RXWATER].<br>0b - Receive data not ready<br>1b - Receive data ready |
| 0<br>TDF | Transmit Data Flag<br>Indicates whether transmit data is requested. This flag is set when the number of words in the transmit FIFO is equal or less than MFCR[TXWATER].<br>0b - Transmit data not requested<br>1b - Transmit data requested |

### 68.7.1.6 Controller Interrupt Enable (MIER)

**Offset**

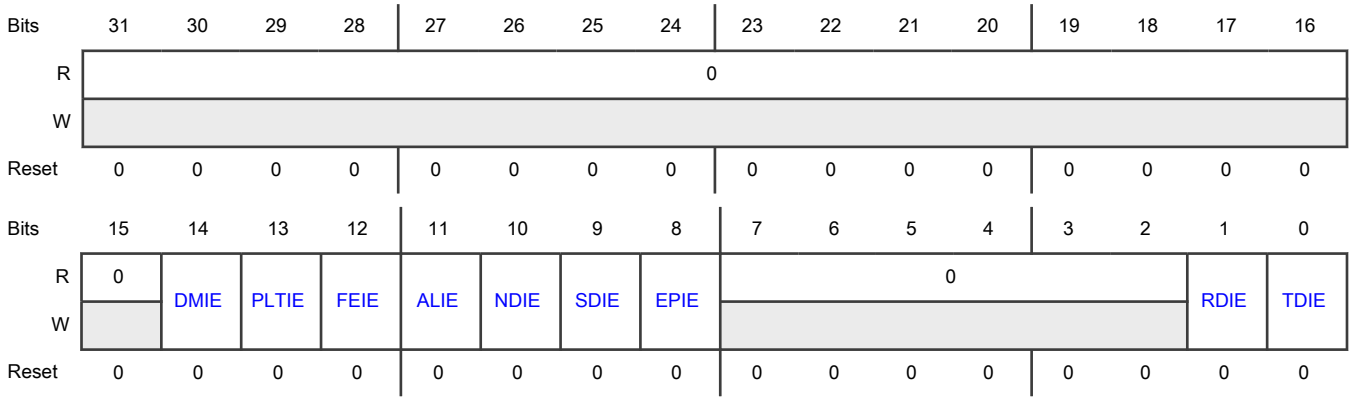
| Register | Offset |
|---|---|
| MIER | 18h |

**Function**

Contains enables for:

- Transmit and receive data interrupts
- Start, Stop, and NACK detection interrupts
- DMA interrupts

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>— | Reserved |
| 14<br>DMIE | Data Match Interrupt Enable<br>Enables interrupt for data match.<br>0b - Disable<br>1b - Enable |
| 13<br>PLTIE | Pin Low Timeout Interrupt Enable<br>Enables interrupt for pin-low timeout.<br>0b - Disable<br>1b - Enable |
| 12<br>FEIE | FIFO Error Interrupt Enable<br>Enables interrupt for FIFO error.<br>0b - Disable<br>1b - Enable |
| 11<br>ALIE | Arbitration Lost Interrupt Enable<br>Enables interrupt for arbitration lost.<br>0b - Disable<br>1b - Enable |
| 10 | NACK Detect Interrupt Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| NDIE | Enables interrupt for NACK detection.<br>0b - Disable<br>1b - Enable |
| 9<br>SDIE | Stop Detect Interrupt Enable<br>Enables interrupt for Stop detection.<br>0b - Disable<br>1b - Enable |
| 8<br>EPIE | End Packet Interrupt Enable<br>Enables interrupt for end packet.<br>0b - Disable<br>1b - Enable |
| 7-2<br>— | Reserved |
| 1<br>RDIE | Receive Data Interrupt Enable<br>Enables interrupt for receive data.<br>0b - Disable<br>1b - Enable |
| 0<br>TDIE | Transmit Data Interrupt Enable<br>Enables interrupt for transmit data.<br>0b - Disable<br>1b - Enable |

### 68.7.1.7 Controller DMA Enable (MDER)

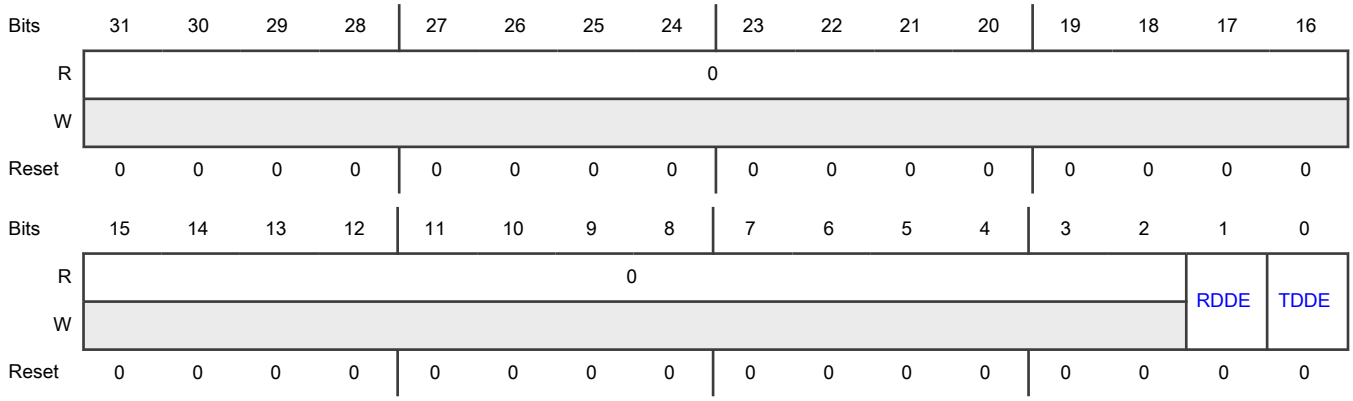
**Offset**

| Register | Offset |
|---|---|
| MDER | 1Ch |

**Function**

Contains DMA transmit, request, and receive enables.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RDDE | Receive Data DMA Enable<br>Enables DMA receive data.<br>0b - Disable<br>1b - Enable |
| 0<br>TDDE | Transmit Data DMA Enable<br>Enables DMA transmit data.<br>0b - Disable<br>1b - Enable |

## 68.7.1.8 Controller Configuration 0 (MCFGR0)

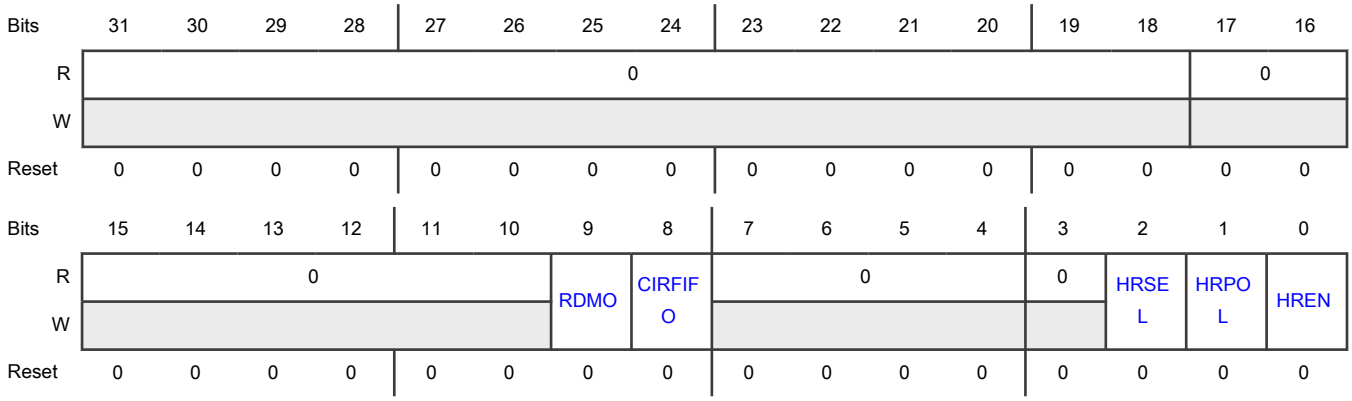
### Offset

| Register | Offset |
|---|---|
| MCFGR0 | 20h |

### Function

Contains host settings and other receive and transfer settings.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-18<br>— | Reserved |
| 17-16<br>— | Reserved |
| 15-10<br>— | Reserved |
| 9<br>RDMO | Receive Data Match Only<br>Determines whether all received data that does not set MSR[DMF] is discarded. After MSR[DMF] is set, the RDMO configuration is ignored. When disabling RDMO, write 0 to this field before writing 0 to MSR[DMF] to ensure that no receive data is lost.<br>0b - Received data is stored in the receive FIFO<br>1b - Received data is discarded unless MSR[DMF] is set |
| 8<br>CIRFIFO | Circular FIFO Enable<br>Enables the transmit FIFO read pointer to be saved to a temporary register. The transmit FIFO empties as normal. After the LPI2C controller is idle and the transmit FIFO is empty, the read pointer value is restored from the temporary register. This setting causes the contents of the transmit FIFO to be cycled through repeatedly. If MCFGR1[AUTOSTOP] is 1, then a Stop condition is sent whenever the transmit FIFO is empty and the read pointer is restored.<br>0b - Disable<br>1b - Enable |
| 7-4<br>— | Reserved |
| 3 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 2<br>HRSEL | Host Request Select<br>Selects the source of the host request input. When host request is enabled, this field must not change.<br>0b - Host request input is pin HREQ<br>1b - Host request input is input trigger |
| 1<br>HRPOL | Host Request Polarity<br>Configures the polarity of the host request input. When host request is enabled, this field must not change.<br>HRPOL sets the polarity for both the HREQ pin and the input trigger.<br>• When HRPOL=0, the polarity is configured for active low, so host request is asserted if the HREQ pin or input trigger are logic 0.<br>• When HRPOL=1, the polarity is configured for active high, so host request is asserted if the HREQ pin or input trigger are logic 1.<br>0b - Active low<br>1b - Active high |
| 0<br>HREN | Host Request Enable<br>Enables host request. When enabled, the LPI2C controller only initiates a Start condition if the host request input is asserted and the bus is idle. A repeated Start condition is not affected by the host request.<br>0b - Disable<br>1b - Enable |

### 68.7.1.9 Controller Configuration 1 (MCFGR1)

#### Offset

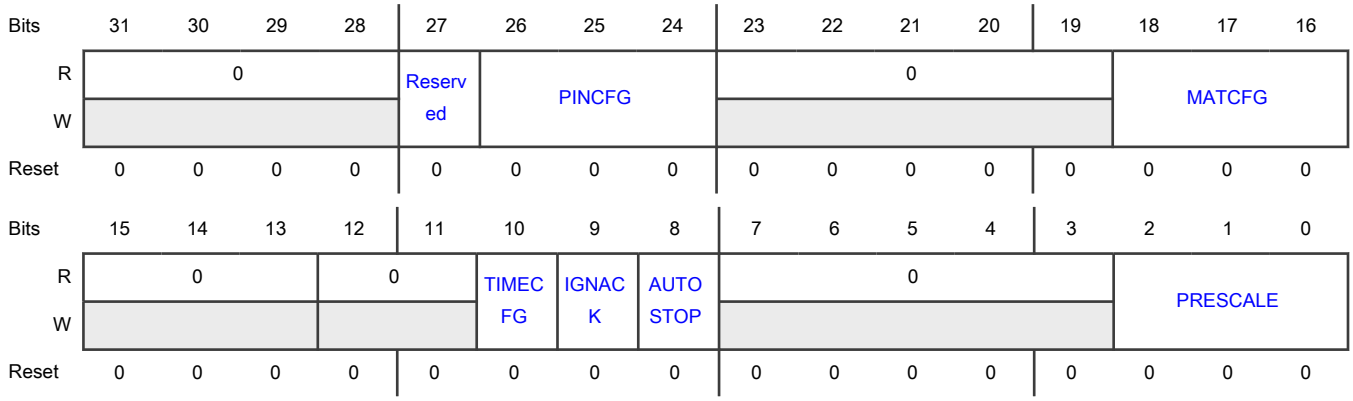
| Register | Offset |
|---|---|
| MCFGR1 | 24h |

#### Function

Contains controls for pin configuration, clock prescaler, and various other control settings.

Write to this register only when the I2C controller is disabled.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27<br>— | Reserved |
| 26-24<br>PINCFG | Pin Configuration<br>Configures the pin mode for LPI2C.<br>000b - Two-pin open drain mode. SCL/SDA pins: Bidirectional open drain for controller and target. SCLS/SDAS pins: Not used.<br>001b - Two-pin output only mode (Ultra-Fast mode). SCL/SDA pins: Output-only (Ultra-Fast mode) open drain for controller and target. SCLS/SDAS pins: Not used.<br>010b - Two-pin push-pull mode. SCL/SDA pins: Bidirectional push-pull for controller and target. SCLS/SDAS pins: Not used.<br>011b - Four-pin push-pull mode. SCL/SDA pins: Input only for controller and target. SCLS/SDAS pins: Output-only push-pull for controller and target.<br>100b - Two-pin open-drain mode with separate LPI2C target. SCL/SDA pins: Bidirectional open drain for controller. SCLS/SDAS pins: Bidirectional open drain for target.<br>101b - Two-pin output only mode (Ultra-Fast mode) with separate LPI2C target. SCL/SDA pins: Output-only (Ultra-Fast mode) open drain for controller. SCLS/SDAS pins: Output-only open drain for target.<br>110b - Two-pin push-pull mode with separate LPI2C target. SCL/SDA pins: Bidirectional push-pull for controller. SCLS/SDAS pins: Bidirectional push-pull for target.<br>111b - Four-pin push-pull mode (inverted outputs). SCL/SDA pins: Input only for controller and target. SCLS/SDAS pins: Inverted output-only push-pull for controller and target. |
| 23-19<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 18-16<br>MATCFG | Match Configuration<br>Configures the condition that sets MSR[DMF]. See Controller Data Match (MDMR).<br>000b - Match is disabled<br>001b - Reserved<br>010b - Match is enabled: first data word equals MDMR[MATCH0] OR MDMR[MATCH1]<br>011b - Match is enabled: any data word equals MDMR[MATCH0] OR MDMR[MATCH1]<br>100b - Match is enabled: (first data word equals MDMR[MATCH0]) AND (second data word equals MDMR[MATCH1)<br>101b - Match is enabled: (any data word equals MDMR[MATCH0]) AND (next data word equals MDMR[MATCH1)<br>110b - Match is enabled: (first data word AND MDMR[MATCH1]) equals (MDMR[MATCH0] AND MDMR[MATCH1])<br>111b - Match is enabled: (any data word AND MDMR[MATCH1]) equals (MDMR[MATCH0] AND MDMR[MATCH1]) |
| 15-13<br>— | Reserved |
| 12-11<br>— | Reserved |
| 10<br>TIMECFG | Timeout Configuration<br>Configures which signals must be low for longer than the configured timeout to set MSR[PLTF].<br>When this field is 0, MSR[PLTF] is set when SCL is low for longer than the configured timeout.<br>0b - SCL<br>1b - SCL or SDA |
| 9<br>IGNACK | Ignore NACK<br>Determines whether the LPI2C controller ignores a received NACK and treats it as an ACK. This field must be 1 in Ultra-Fast mode.<br>0b - No effect<br>1b - Treat a received NACK as an ACK |
| 8<br>AUTOSTOP | Automatic Stop Generation<br>Determines whether a Stop condition is generated when the LPI2C controller is busy and the transmit FIFO is empty. A Stop condition can also be generated using a transmit FIFO command.<br>When this field is 1, a Stop condition is automatically generated when the transmit FIFO is empty and the LPI2C controller is busy.<br>0b - No effect |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Stop automatically generated |
| 7-3<br>— | Reserved |
| 2-0<br>PRESCALE | Prescaler<br>Configures the clock prescaler used for all LPI2C controller logic except the digital glitch filters.<br>000b - Divide by 1<br>001b - Divide by 2<br>010b - Divide by 4<br>011b - Divide by 8<br>100b - Divide by 16<br>101b - Divide by 32<br>110b - Divide by 64<br>111b - Divide by 128 |

### 68.7.1.10 Controller Configuration 2 (MCFGR2)

#### Offset

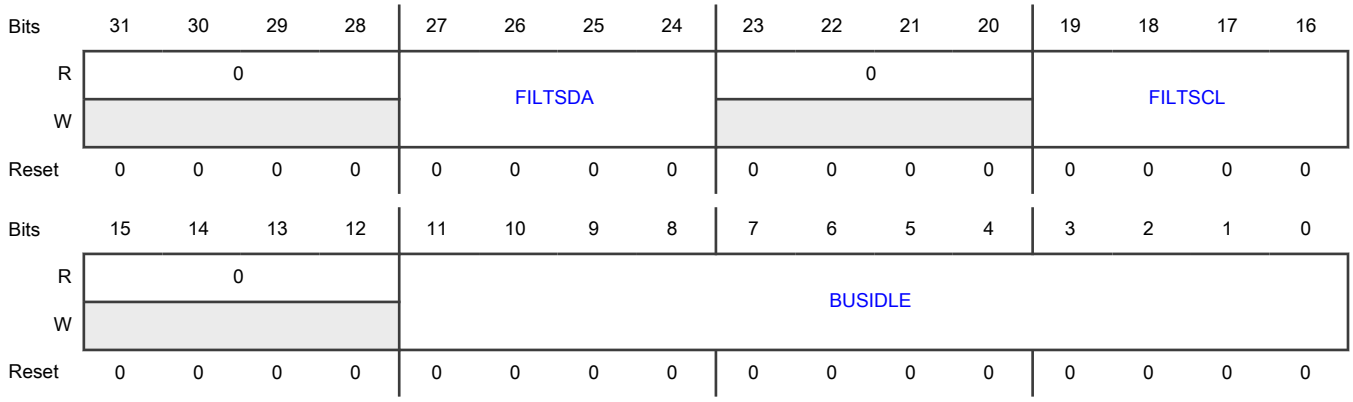
| Register | Offset |
|---|---|
| MCFGR2 | 28h |

#### Function

Contains the configuration for the bus idle timeout and glitch filters for SDA and SCL.

Write to this register only when the I2C controller is disabled.

#### Diagram



**Fields**

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>FILTSDA | Glitch Filter SDA<br>Configures the I2C controller digital glitch filters for the SDA input.<br>The latency through the glitch filter is equal to the number of cycles defined by this field. The value of this field must be less than the minimum SCL low or high period.<br>Glitches equal to or less than the number of cycles defined by this field are filtered out and ignored. Writing 0 to this field disables the glitch filter.<br>MCFGR1[PRESCALE] does not affect the glitch filter cycle count. It is automatically bypassed in HS mode. |
| 23-20<br>— | Reserved |
| 19-16<br>FILTSCL | Glitch Filter SCL<br>Configures the I2C controller digital glitch filters for SCL input.<br>The latency through the glitch filter is equal to the number of cycles defined by this field. The value of this field must be less than the minimum SCL low or high period.<br>Glitches equal to or less than the number of cycles defined by this field are filtered out and ignored. These cycles are based on the functional clock. Writing 0 to this field disables the glitch filter.<br>MCFGR1[PRESCALE] does not affect the glitch filter cycle count. It is automatically bypassed in HS mode. |
| 15-12<br>— | Reserved |
| 11-0<br>BUSIDLE | Bus Idle Timeout<br>Configures the bus idle timeout period, in clock cycles.<br>If both SCL and SDA are high for longer than the number of cycles defined by this field, the I2C bus is assumed to be idle and the controller can generate a Start condition.<br>Writing 0 to this field disables the bus idle timeout. |

### 68.7.1.11 Controller Configuration 3 (MCFGR3)

**Offset**

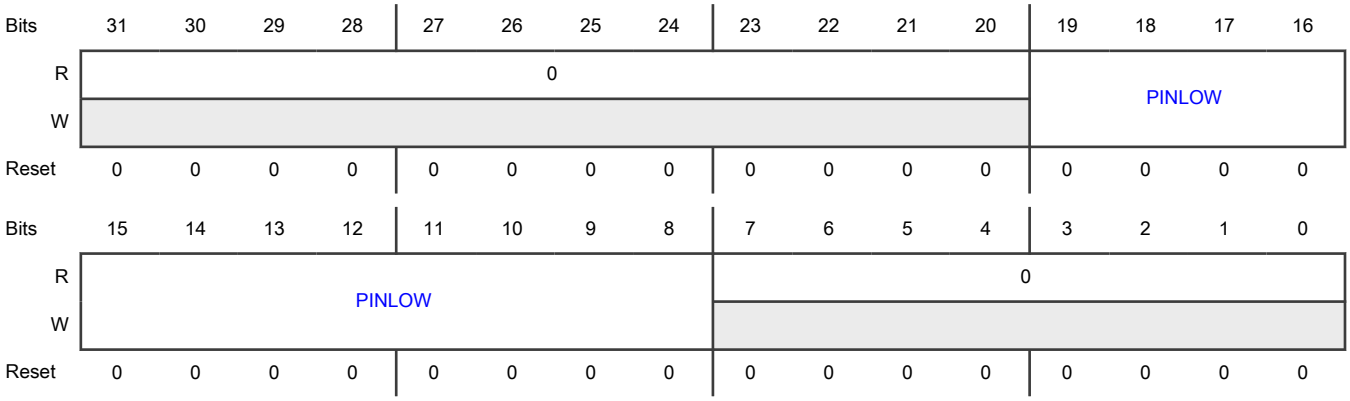
| Register | Offset |
|---|---|
| MCFGR3 | 2Ch |

**Function**

Configures the threshold value for the pin low timeout flag.

Write to this register only when the I2C controller is disabled.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19-8<br>PINLOW | Pin Low Timeout<br>Configures the threshold value, in clock cycles, that sets MSR[PLTF].<br>If SCL or SDA (selected by MCFGR1[TIMECFG]) is low for longer than (PINLOW × 256) cycles, MSR[PLTF] is set.<br>When this field is 0, the pin low timeout feature is disabled. |
| 7-0<br>— | Reserved |

## 68.7.1.12 Controller Data Match (MDMR)

### Offset

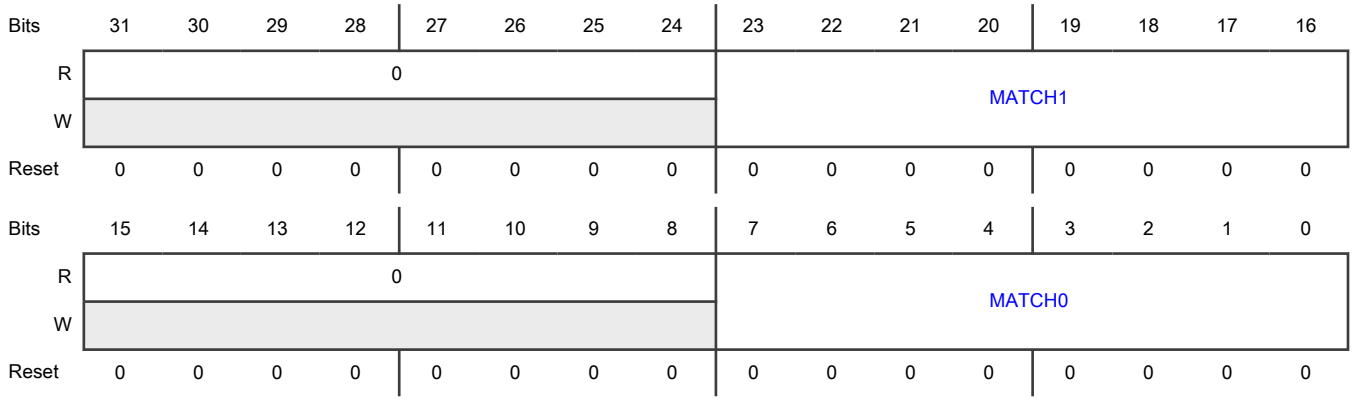
| Register | Offset |
|---|---|
| MDMR | 40h |

### Function

Contains data match values.

Write to this register only when the I2C controller is disabled or idle.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16<br>MATCH1 | Match 1 Value<br>Specifies match 1 value that is compared to the received data when receive data match is enabled. |
| 15-8<br>— | Reserved |
| 7-0<br>MATCH0 | Match 0 Value<br>Specifies match 0 value that is compared to the received data when receive data match is enabled. |

## 68.7.1.13 Controller Clock Configuration 0 (MCCR0)

### Offset

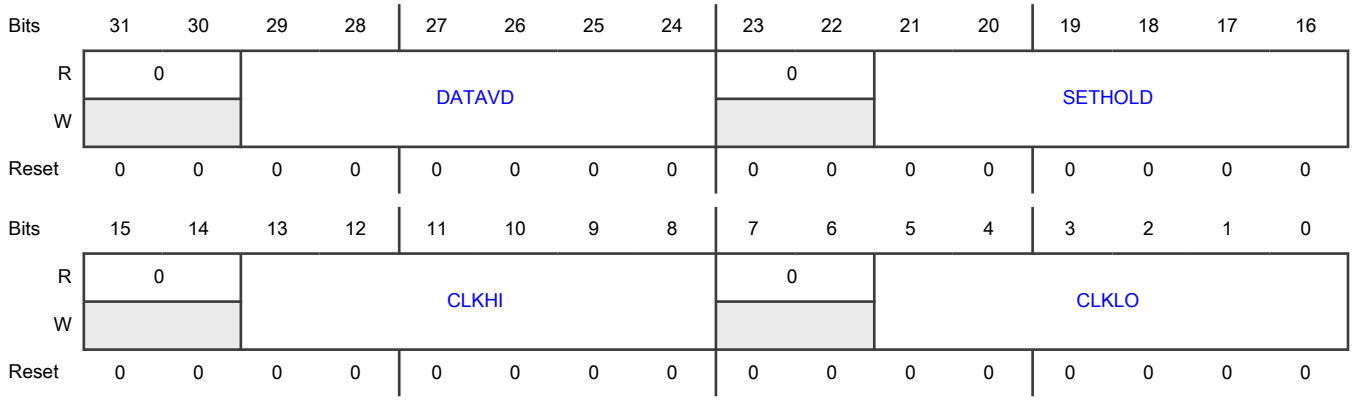
| Register | Offset |
|---|---|
| MCCR0 | 48h |

### Function

Configures various clock controls.

You cannot make changes to this register when the I2C controller is enabled and is used for standard, fast, fast-mode plus, and ultra-fast transfers.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-24<br>DATAVD | Data Valid Delay<br>Specifies the minimum number of cycles (minus one) used as the data hold time for SDA. This value must be less than the minimum SCL low period. |
| 23-22<br>— | Reserved |
| 21-16<br>SETHOLD | Setup Hold Delay<br>Specifies the minimum number of cycles (minus one) used by the controller for these conditions:<br>• Hold time for a Start<br>• Setup and hold time for a repeated Start<br>• Setup time for a Stop<br>The setup time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) ÷ 2^PRESCALE cycles. |
| 15-14<br>— | Reserved |
| 13-8<br>CLKHI | Clock High Period<br>Specifies the minimum number of cycles (minus one) that the controller drives the SCL clock high. The SCL high time is extended by the time needed to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) ÷ 2^PRESCALE cycles. |
| 7-6<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 5-0<br>CLKLO | Clock Low Period<br>Specifies the minimum number of cycles (minus one) that the controller drives the SCL clock low. This value is also used for the minimum bus free time between a Stop and a Start condition. This period is extended by the time needed to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) ÷ 2^PRESCALE cycles. |

### 68.7.1.14 Controller Clock Configuration 1 (MCCR1)

#### Offset

| Register | Offset |
|---|---|
| MCCR1 | 50h |

#### Function

Configures various clock controls.

You cannot makes changes to this register when the I2C controller is enabled and is used for HS mode transfers. The separate clock configuration for HS mode allows arbitration to take place in Fast mode (with timing configured by Controller Clock Configuration 0 (MCCR0)), before switching to HS mode (with timing configured by MCCR1).

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-24<br>DATAVD | Data Valid Delay |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Specifies the minimum number of cycles (minus one) used as the data hold time for SDA. This value must be less than the minimum SCL low period. |
| 23-22<br>— | Reserved |
| 21-16<br>SETHOLD | Setup Hold Delay<br>Specifies the minimum number of cycles (minus one) used by the controller for these conditions:<br>• Hold time for a Start condition<br>• Setup and hold time for a repeated Start condition<br>• Setup time for a Stop condition<br>The setup time is extended by the time needed to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) ÷ 2^PRESCALE cycles. |
| 15-14<br>— | Reserved |
| 13-8<br>CLKHI | Clock High Period<br>Specifies the minimum number of cycles (minus one) that the controller drives the SCL clock high. The SCL high time is extended by the time needed to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) ÷ 2^PRESCALE cycles. |
| 7-6<br>— | Reserved |
| 5-0<br>CLKLO | Clock Low Period<br>Specifies the minimum number of cycles (minus one) that the controller drives the SCL clock low. This value is also used for the minimum bus free time between a Stop and a Start condition. This period is extended by the time needed to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) ÷ 2^PRESCALE cycles. |

#### 68.7.1.15 Controller FIFO Control (MFCR)

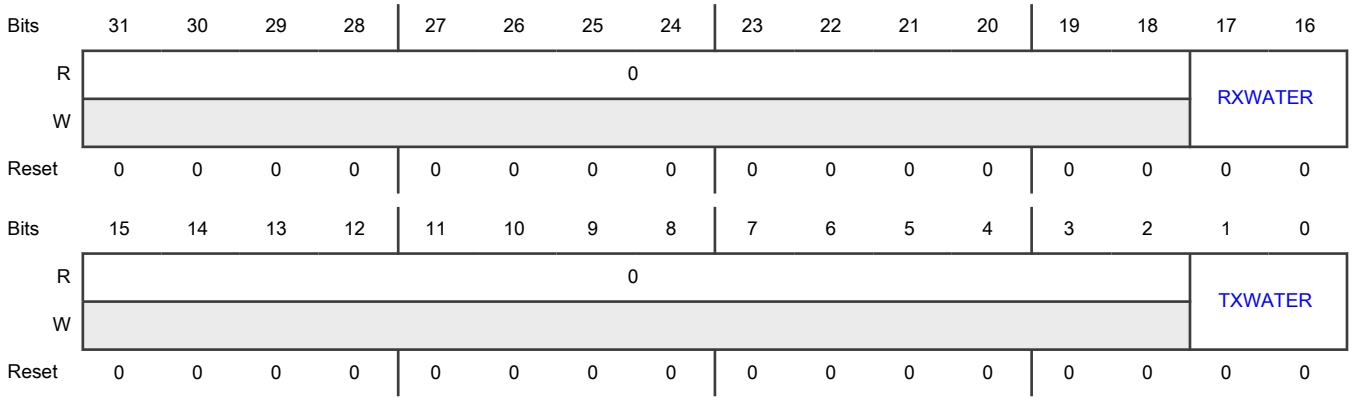
**Offset**

| Register | Offset |
|---|---|
| MFCR | 58h |

**Function**

Controls the receive and transmit FIFO watermark values.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-18<br>— | Reserved |
| 17-16<br>RXWATER | Receive FIFO Watermark<br>Determines the watermark for setting SSR[RDF]. That flag is set when the number of words in the receive FIFO is greater than the value of this field. Writing a value equal to or greater than the FIFO size truncates the value. |
| 15-2<br>— | Reserved |
| 1-0<br>TXWATER | Transmit FIFO Watermark<br>Determines the watermark for setting SSR[TDF]. That flag is set when the number of words in the transmit FIFO is equal or less than the value of this field. Writing a value equal to or greater than the FIFO size truncates the value. |

## 68.7.1.16 Controller FIFO Status (MFSR)

### Offset

| Register | Offset |
|---|---|
| MFSR | 5Ch |

### Function

Specifies the number of words in the transmit and receive FIFOs.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18-16<br>RXCOUNT | Receive FIFO Count<br>Specifies the number of words in the receive FIFO. |
| 15-3<br>— | Reserved |
| 2-0<br>TXCOUNT | Transmit FIFO Count<br>Specifies the number of words in the transmit FIFO. |

## 68.7.1.17 Controller Transmit Data (MTDR)

### Offset

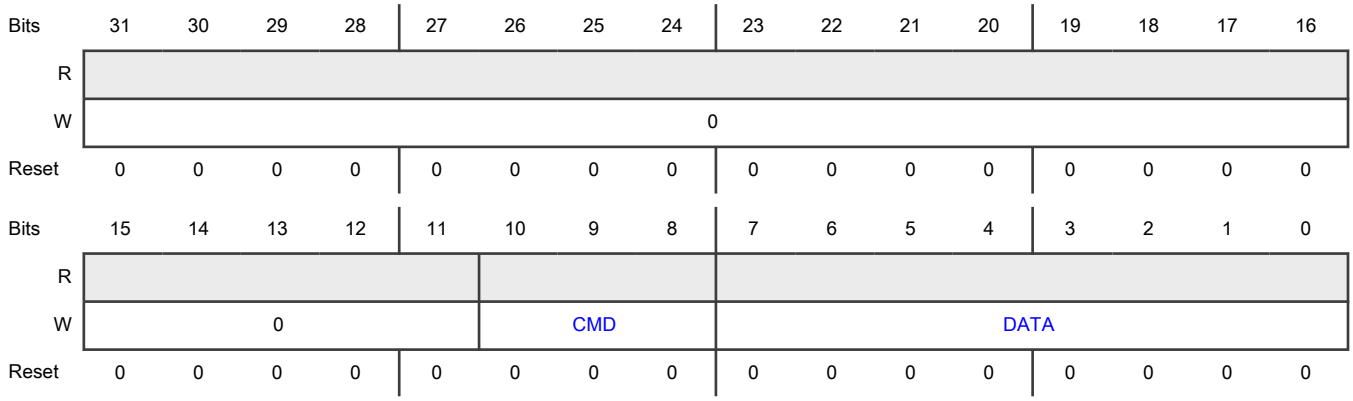
| Register | Offset |
|---|---|
| MTDR | 60h |

### Function

Configures transmit data:

- An 8-bit write to MTDR[CMD] is ignored and does not increment the FIFO write pointer.
- An 8-bit write to MTDR[DATA] zero-extends the value of MTDR[CMD] and increments the FIFO write pointer.
- A 16-bit or 32-bit write operation writes to both MTDR[CMD] and MTDR[DATA] and increments the FIFO write pointer.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-11<br>— | Reserved |
| 10-8<br>CMD | Command Data<br>Selects command transmitted by controller.<br>000b - Transmit the value in DATA[7:0]<br>001b - Receive (DATA[7:0] + 1) bytes. DATA[7:0] is used as a byte counter. Receive that many bytes and check each for a data match (if configured) before storing the received data in the receive FIFO.<br>010b - Generate Stop condition on I2C bus<br>011b - Receive and discard (DATA[7:0] + 1) bytes. DATA[7:0] is used as a byte counter. Receive that many bytes but do not check for a data match or store those bytes in the receive FIFO.<br>100b - Generate (repeated) Start on the I2C bus and transmit the address in DATA[7:0]<br>101b - Generate (repeated) Start on the I2C bus and transmit the address in DATA[7:0] (this transfer expects a NACK to be returned)<br>110b - Generate (repeated) Start on the I2C bus and transmit the address in DATA[7:0] using HS mode<br>111b - Generate (repeated) Start on the I2C bus and transmit the address in DATA[7:0] using HS mode (this transfer expects a NACK to be returned) |
| 7-0<br>DATA | Transmit Data<br>Contains data used by the commands listed in MTDR[CMD]. Performing an 8-bit write to this field zero-extends the value of MTDR[CMD]. |

### 68.7.1.18 Controller Receive Data (MRDR)

#### Offset

| Register | Offset |
|---|---|
| MRDR | 70h |

#### Function

Contains the status of the receive FIFO and the data received by the I2C controller that has not been discarded.

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-15<br>— | Reserved |
| 14<br>RXEMPTY | Receive Empty<br>Indicates whether the controller receive data FIFO is empty.<br>0b - Not empty<br>1b - Empty |
| 13-8<br>— | Reserved |
| 7-0<br>DATA | Receive Data<br>Contains data received by the I2C controller that has not been discarded. Received data can be discarded due to the command in MTDR[CMD], or the controller can be configured to discard nonmatching data. |

### 68.7.1.19 Target Control (SCR)

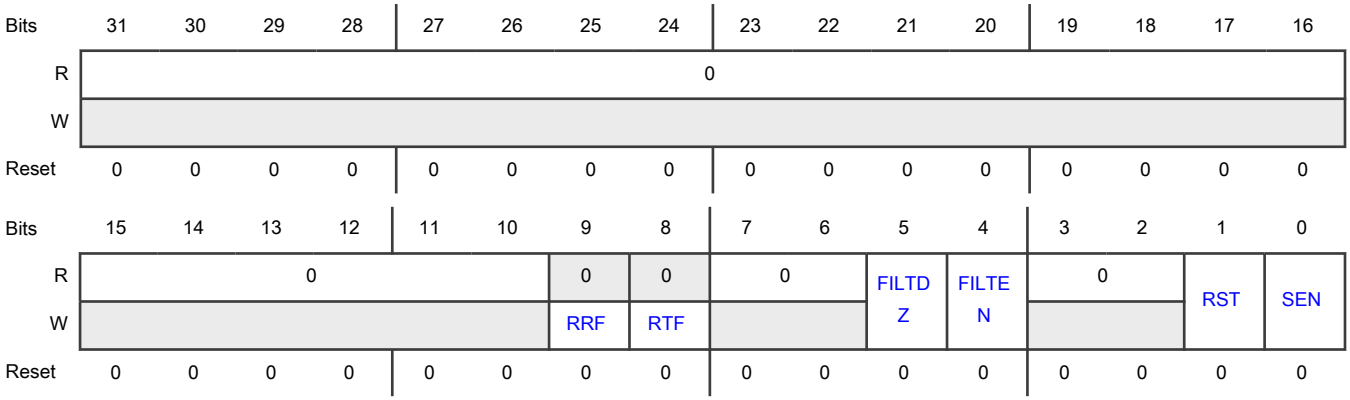
#### Offset

| Register | Offset |
|---|---|
| SCR | 110h |

#### Function

Contains resets and other target control settings.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved |
| 9<br>RRF | Reset Receive FIFO<br>Empties the receive FIFO in Target Receive Data (SRDR).<br>0b - No effect<br>1b - SRDR is now empty |
| 8<br>RTF | Reset Transmit FIFO<br>Empties the transmit FIFO in Target Transmit Data (STDR).<br>0b - No effect<br>1b - STDR is now empty |
| 7-6<br>— | Reserved |
| 5 | Filter Doze Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| FILTDZ | Enables filter in Doze mode. Update this field only when the I2C target is disabled.<br>0b - Enable<br>1b - Disable |
| 4<br>FILTEN | Filter Enable<br>Enables digital filter and output delay counter for target mode. Update this field only when the I2C target is disabled.<br>0b - Disable<br>1b - Enable |
| 3-2<br>— | Reserved |
| 1<br>RST | Software Reset<br>Resets target mode logic. The reset takes effect immediately. The value of this field remains 1 until you write 0 to it. There is no minimum delay required before clearing the software reset.<br>0b - Not reset<br>1b - Reset |
| 0<br>SEN | Target Enable<br>Enables I2C Target mode.<br>0b - Disable<br>1b - Enable |

### 68.7.1.20 Target Status (SSR)

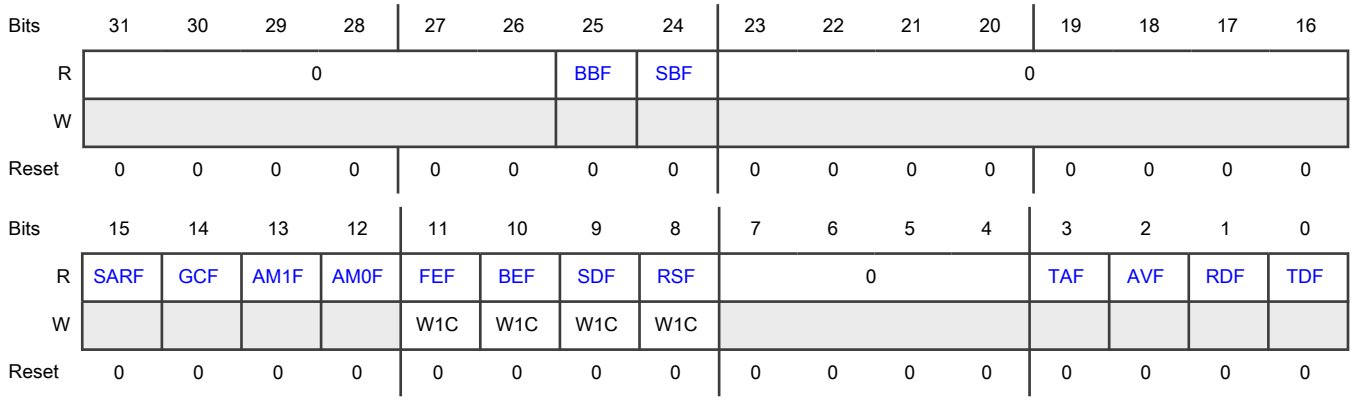
**Offset**

| Register | Offset |
|---|---|
| SSR | 114h |

**Function**

Contains status flags for transmit and receive data, for error conditions, and for bus and target busy or idle status.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25<br>BBF | Bus Busy Flag<br>Indicates whether an I2C bus is idle or busy.<br>0b - Idle<br>1b - Busy |
| 24<br>SBF | Target Busy Flag<br>Indicates whether an I2C target is idle or busy.<br>0b - Idle<br>1b - Busy |
| 23-16<br>— | Reserved |
| 15<br>SARF | SMBus Alert Response Flag<br>Indicates whether an SMBus alert response has been detected.<br>You can clear this flag by reading Target Address Status (SASR). This flag cannot generate an asynchronous wakeup.<br>0b - Disabled or not detected<br>1b - Enabled and detected |
| 14<br>GCF | General Call Flag<br>Indicates whether a target has detected the general call address.<br>You can clear this flag by reading Target Address Status (SASR). This flag cannot generate an asynchronous wakeup. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - General call address disabled or not detected<br>1b - General call address detected |
| 13<br>AM1F | Address Match 1 Flag<br>Indicates whether the received address matches the value in ADDR1, or it falls within the ADDR0 to ADDR1 range as configured by SCFGR1[ADDRCFG].<br>This flag is cleared by reading Target Address Status (SASR). This flag cannot generate an asynchronous wakeup.<br>0b - Matching address not received<br>1b - Matching address received |
| 12<br>AM0F | Address Match 0 Flag<br>Indicates whether the received address matches the ADDR0 field, as configured by SCFGR1[ADDRCFG].<br>This flag is cleared by reading Target Address Status (SASR). This flag cannot generate an asynchronous wakeup.<br>0b - ADDR0 matching address not received<br>1b - ADDR0 matching address received |
| 11<br>FEF | FIFO Error Flag<br>Indicates whether there is a FIFO error. This flag can only be set when clock stretching is disabled.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No FIFO error<br>1b - FIFO error<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 10<br>BEF | Bit Error Flag<br>Indicates whether the LPI2C target has transmitted a logic 1 and detects a logic 0 on the I2C bus. The target ignores the rest of the transfer until the next (repeated) Start condition.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No bit error occurred |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Bit error occurred<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 9<br>SDF | Stop Detect Flag<br>Indicates whether the LPI2C target detects a Stop condition, and if the LPI2C target matched the last address byte.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No Stop detected<br>1b - Stop detected<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 8<br>RSF | Repeated Start Flag<br>Indicates whether the LPI2C target detects a repeated Start condition and if the LPI2C target matched the last address byte. This flag is not set when the target first detects a Start condition.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No repeated Start detected<br>1b - Repeated Start detected<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 7-4<br>— | Reserved |
| 3<br>TAF | Transmit ACK Flag<br>Indicates whether a transmit ACK or NACK is required. You can clear this flag by writing to Target Transmit ACK (STAR).<br>0b - Not required<br>1b - Required |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 2<br>AVF | Address Valid Flag<br>Indicates whether the contents of Target Address Status (SASR) are valid. You can clear this flag by reading SASR. When SCFGR1[RXCFG] = 1, this flag is also cleared by reading Target Receive Data (SRDR).<br>0b - Not valid<br>1b - Valid |
| 1<br>RDF | Receive Data Flag<br>Indicates whether receive data is ready. You can clear this flag by reading Target Receive Data (SRDR). When SCFGR1[RXCFG] = 1, this flag is not cleared when reading Target Receive Data (SRDR) if SSR[AVF] = 1.<br>0b - Not ready<br>1b - Ready |
| 0<br>TDF | Transmit Data Flag<br>Indicates whether transmit data has been requested. This flag is cleared by writing to Target Transmit Data (STDR). When SCFGR1[TXCFG] = 0, if a NACK, repeated Start, or Stop condition is detected, this flag is also cleared.<br>0b - Transmit data not requested<br>1b - Transmit data is requested |

### 68.7.1.21 Target Interrupt Enable (SIER)

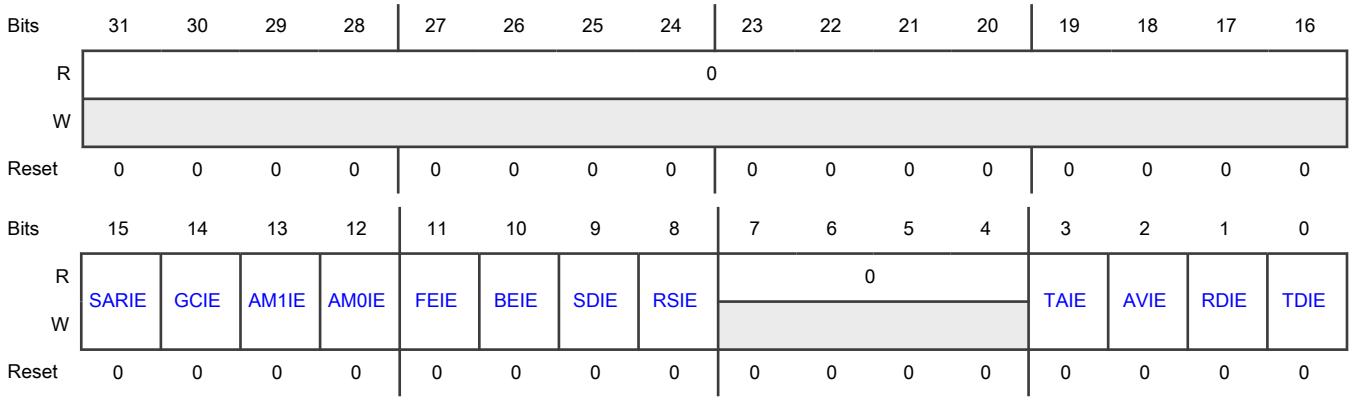
**Offset**

| Register | Offset |
|---|---|
| SIER | 118h |

**Function**

Contains transmit and receive data interrupt enables, start and stop detect interrupt enables, and other target interrupt enables.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>SARIE | SMBus Alert Response Interrupt Enable<br>Enables interrupt for SMBus alert response.<br>0b - Disable<br>1b - Enable |
| 14<br>GCIE | General Call Interrupt Enable<br>Enables interrupt for general call.<br>0b - Disabled<br>1b - Enabled |
| 13<br>AM1IE | Address Match 1 Interrupt Enable<br>Enables interrupt for address match 1.<br>0b - Disable<br>1b - Enable |
| 12<br>AM0IE | Address Match 0 Interrupt Enable<br>Enables interrupt for address match 0.<br>0b - Disable<br>1b - Enable |
| 11<br>FEIE | FIFO Error Interrupt Enable<br>Enables interrupt for FIFO error.<br>0b - Disable<br>1b - Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 10<br>BEIE | Bit Error Interrupt Enable<br>Enables interrupt for bit error.<br>0b - Disable<br>1b - Enable |
| 9<br>SDIE | Stop Detect Interrupt Enable<br>Enables interrupt for Stop detection.<br>0b - Disable<br>1b - Enable |
| 8<br>RSIE | Repeated Start Interrupt Enable<br>Enables interrupt for repeated start.<br>0b - Disable<br>1b - Enable |
| 7-4<br>— | Reserved |
| 3<br>TAIE | Transmit ACK Interrupt Enable<br>Enables interrupt for transmit ACK.<br>0b - Disable<br>1b - Enable |
| 2<br>AVIE | Address Valid Interrupt Enable<br>Enables interrupt for valid address.<br>0b - Disable<br>1b - Enable |
| 1<br>RDIE | Receive Data Interrupt Enable<br>Enables interrupt for receive data.<br>0b - Disable<br>1b - Enable |
| 0<br>TDIE | Transmit Data Interrupt Enable<br>Enables interrupt for transmit data.<br>0b - Disable<br>1b - Enable |

### 68.7.1.22 Target DMA Enable (SDER)

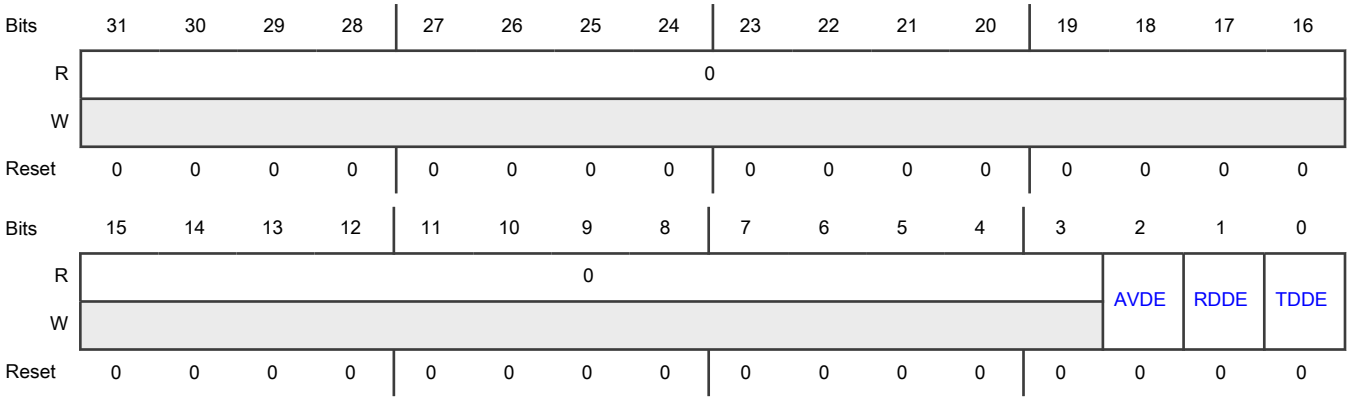
#### Offset

| Register | Offset |
|---|---|
| SDER | 11Ch |

#### Function

Contains the transmit, request, and receive enables for DMA.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2<br>AVDE | Address Valid DMA Enable<br>Enables address valid DMA request. The address valid DMA request is shared with the receive data DMA request. If both are enabled, write 1 to SCFGR1[RXCFG] to allow the DMA to read the address from Target Receive Data (SRDR).<br>0b - Disable<br>1b - Enable |
| 1<br>RDDE | Receive Data DMA Enable<br>Enables receive data for DMA.<br>0b - Disable DMA request<br>1b - Enable DMA request |
| 0<br>TDDE | Transmit Data DMA Enable<br>Enables transmit data for DMA. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disable<br>1b - Enable |

### 68.7.1.23 Target Configuration 1 (SCFGR1)

#### Offset

| Register | Offset |
|---|---|
| SCFGR1 | 124h |

#### Function

Configures various aspects of the target.

Write to this register only when the I2C target is disabled.

#### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | HSMEN | IGNACK | RXCFG | TXCFG | SAEN | GCEN | 0 | | | 0 | ACKSTALL | TXDSTALL | RXSTALL | ADRSTALL |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26-24<br>— | Reserved |
| 23-19<br>— | Reserved |
| 18-16 | Address Configuration |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| ADDRCFG | Configures the condition that causes an address to match.<br>000b - Address match 0 (7-bit)<br>001b - Address match 0 (10-bit)<br>010b - Address match 0 (7-bit) or address match 1 (7-bit)<br>011b - Address match 0 (10-bit) or address match 1 (10-bit)<br>100b - Address match 0 (7-bit) or address match 1 (10-bit)<br>101b - Address match 0 (10-bit) or address match 1 (7-bit)<br>110b - From address match 0 (7-bit) to address match 1 (7-bit)<br>111b - From address match 0 (10-bit) to address match 1 (10-bit) |
| 15-14<br>— | Reserved |
| 13<br>HSMEN | HS Mode Enable<br>Enables detection of the HS mode controller code of target address 0000_1XX, but does not cause an address match on this code. When this field is 1 and any HS mode controller code is detected, SCR[FILTEN] and SCFGR1[ACKSTALL] are ignored until the next Stop condition is detected.<br>0b - Disable<br>1b - Enable |
| 12<br>IGNACK | Ignore NACK<br>Determines whether the target ends transfer when a NACK condition is detected. When this field is 1, the LPI2C target continues transfers after a NACK is detected. This field is required to be 1 in Ultra-Fast mode.<br>0b - End transfer on NACK<br>1b - Do not end transfer on NACK |
| 11<br>RXCFG | Receive Data Configuration<br>Configures which data is returned and which flags are cleared when reading Target Receive Data (SRDR).<br>When this field is 0, reading SRDR returns received data and clears SSR[RDF].<br>When this field is 1, reading SRDR:<br>• Returns the value of Target Address Status (SASR) and clears SSR[AVF] when SSR[AVF] is set.<br>• Returns received data and clears SSR[RDF] when SSR[AVF] is not set.<br>0b - Return received data, clear SSR[RDF]<br>1b - Return SASR and clear SSR[AVF] when SSR[AVF] is set, return received data and clear SSR[RDF] when SSR[AFV] is not set |
| 10<br>TXCFG | Transmit Flag Configuration<br>Determines which conditions set SSR[TDF]. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | This field always becomes 1 before a NACK is detected at the end of a target-transmit transfer. This change can cause an extra word to be written to the transmit data FIFO.<br>When this field is 0, Target Transmit Data (STDR) is automatically emptied when a target-transmit transfer is detected. SSR[TDF] is set when a target-transmit transfer is detected, and SSR[TDF] is cleared at the end of the target-transmit transfer.<br>When this field is 1, SSR[TDF] is set when STDR is empty, and SSR[TDF] is cleared when STDR is full. This setting allows STDR to be filled before a target-transmit transfer is detected. However, it can cause STDR to be written before a NACK is detected on the last byte of a target-transmit transfer.<br>0b - SSR[TDF] is set only during a target-transmit transfer when STDR is empty<br>1b - SSR[TDF] is set whenever STDR is empty |
| 9<br>SAEN | SMBus Alert Enable<br>Enables a match on an SMBus alert.<br>0b - Disable<br>1b - Enable |
| 8<br>GCEN | General Call Enable<br>Enables a general call address.<br>0b - Disable<br>1b - Enable |
| 7-5<br>— | Reserved |
| 4<br>— | Reserved |
| 3<br>ACKSTALL | ACK SCL Stall<br>Enables SCL clock stretching during target-transmit address bytes and target-receiver address and data bytes, so you can write to Target Transmit ACK (STAR) before the ACK or NACK is transmitted. Clock stretching occurs when transmitting the ninth bit, and is therefore not compatible with HS mode.<br>If this field is 1:<br>• You do not need to write 1 to SCFGR1[RXSTALL] or SCFGR1[ADRSTALL].<br>• When there is an address match on the first byte of a 10-bit address, SSR[AVF] is set, allowing you to read the received address before writing to Target Transmit ACK (STAR).<br>0b - Disable<br>1b - Enable |
| 2<br>TXDSTALL | Transmit Data SCL Stall |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Enables SCL clock stretching when SSR[TDF] = 1 during a target-transmit transfer. Clock stretching occurs following the ninth bit, and is therefore compatible with HS mode.<br>0b - Disable<br>1b - Enable |
| 1<br>RXSTALL | RX SCL Stall<br>Enables SCL clock stretching when SSR[RDF] = 1 during a target-receive transfer. Clock stretching occurs following the ninth bit, and is therefore compatible with HS mode.<br>0b - Disable<br>1b - Enable |
| 0<br>ADRSTALL | Address SCL Stall<br>Enables SCL clock stretching when SSR[AVF] = 1. Clock stretching only occurs following the ninth bit, and is therefore compatible with HS mode.<br>0b - Disable<br>1b - Enable |

### 68.7.1.24 Target Configuration 2 (SCFGR2)

#### Offset

| Register | Offset |
|---|---|
| SCFGR2 | 128h |

#### Function

Configures data valid delay, clock hold time, and glitch filters for SDA and SCL.

Write to this register only when the I2C target is disabled.

#### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>FILTSDA | Glitch Filter SDA<br>Configures the I2C target digital glitch filters for SDA input.<br>Writing 0 to this field disables the glitch filter.<br>Glitches equal to or less than the number of cycles defined by this field are filtered out and ignored.<br>The latency through the glitch filter is equal to the number of cycles defined by this field + 3. The latency must be configured to be less than the minimum SCL low or high period.<br>MCFGR1[PRESCALE] does not affect the glitch filter cycle count, and the glitch filter cycle count is disabled in HS mode. |
| 23-20<br>— | Reserved |
| 19-16<br>FILTSCL | Glitch Filter SCL<br>Configures the I2C target digital glitch filters for SCL input.<br>Writing 0 to this field disables the glitch filter.<br>Glitches equal to or less than the number of cycles defined by this field are filtered out and ignored.<br>The latency through the glitch filter is equal to the number of cycles defined by this field + 3. The latency must be configured to be less than the minimum SCL low or high period.<br>MCFGR1[PRESCALE] does not affect the glitch filter cycle count, and the glitch filter cycle count is disabled in HS mode. |
| 15-14<br>— | Reserved |
| 13-8<br>DATAVD | Data Valid Delay<br>Configures the SDA data valid delay time for the I2C target, which is equal to FILTSCL + DATAVD + 3 cycles.<br>The data valid delay must be configured to be less than the minimum SCL low period.<br>MCFGR1[PRESCALE] does not affect the I2C target data valid delay time, and the I2C target data valid delay time is disabled in HS mode. |
| 7-4<br>— | Reserved |
| 3-0<br>CLKHOLD | Clock Hold Time<br>Configures the minimum clock hold time for the I2C target, when clock stretching is enabled.<br>The minimum hold time is equal to the number of cycles defined by this field + 3. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | MCFGR1[PRESCALE] does not affect the I2C target clock hold time, and the I2C target clock hold time is disabled in HS mode. |

### 68.7.1.25 Target Address Match (SAMR)

#### Offset

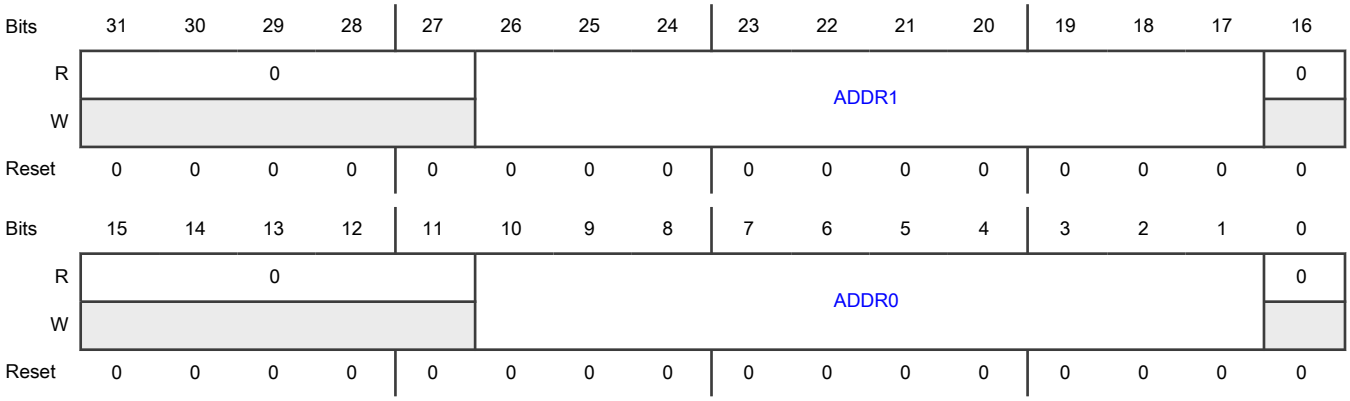
| Register | Offset |
|---|---|
| SAMR | 140h |

#### Function

Contains address values for received target match comparison.

Write to this register only when the I2C target is disabled.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26-17<br>ADDR1 | Address 1 Value<br>Contains the value of address 1, which is compared to the received address to detect the target address.<br>In 10-bit mode, the first address byte is compared to {11110, ADDR1[26:25]} and the second address byte is compared to ADDR1[24:17].<br>In 7-bit mode, the address is compared to ADDR1[23:17]. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 16-11<br>— | Reserved |
| 10-1<br>ADDR0 | Address 0 Value<br>Contains the value of address 0, which is compared to the received address to detect the target address.<br>In 10-bit mode, the first address byte is compared to {11110, ADDR0[10:9]} and the second address byte is compared to ADDR0[8:1].<br>In 7-bit mode, the address is compared to ADDR0[7:1]. |
| 0<br>— | Reserved |

### 68.7.1.26 Target Address Status (SASR)

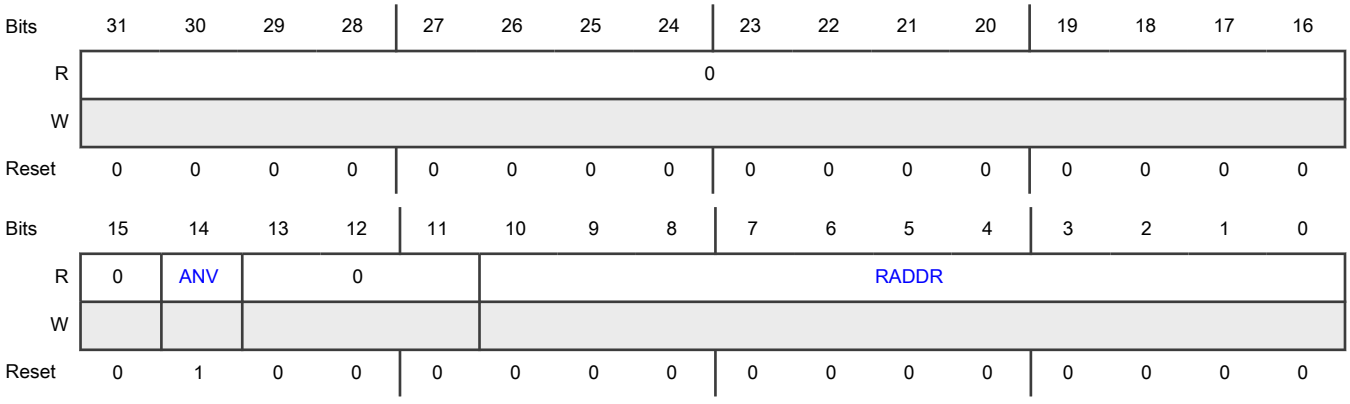
#### Offset

| Register | Offset |
|---|---|
| SASR | 150h |

#### Function

Contains the received address and its validity.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-15 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 14<br>ANV | Address Not Valid<br>Indicates whether SASR[RADDR] is valid.<br>0b - Valid<br>1b - Not valid |
| 13-11<br>— | Reserved |
| 10-0<br>RADDR | Received Address<br>Contains the received address. Updates whenever SSR[AM0F] or SSR[AM1F] is set. Reading Target Address Status (SASR) clears SSR[AM0F] and SSR[AM1F].<br>In 7-bit mode, the address byte is stored in RADDR[7:0].<br>In 10-bit mode, the first address byte is {11110, RADDR[10:9], RADDR[0]} and the second address byte is RADDR[8:1]. The Read-or-Write bit is therefore always stored in RADDR[0].<br>When SCFGR1[ACKSTALL] = 1, if the first address byte matches in 10-bit mode, the first address byte is stored in RADDR[7:0] so you can read this field before writing the Transmit ACK. If the second address byte matches, this field is then updated with the full 10-bit address. |

### 68.7.1.27 Target Transmit ACK (STAR)

#### Offset

| Register | Offset |
|---|---|
| STAR | 154h |

#### Function

Configures choice of ACK or NACK on each received word.

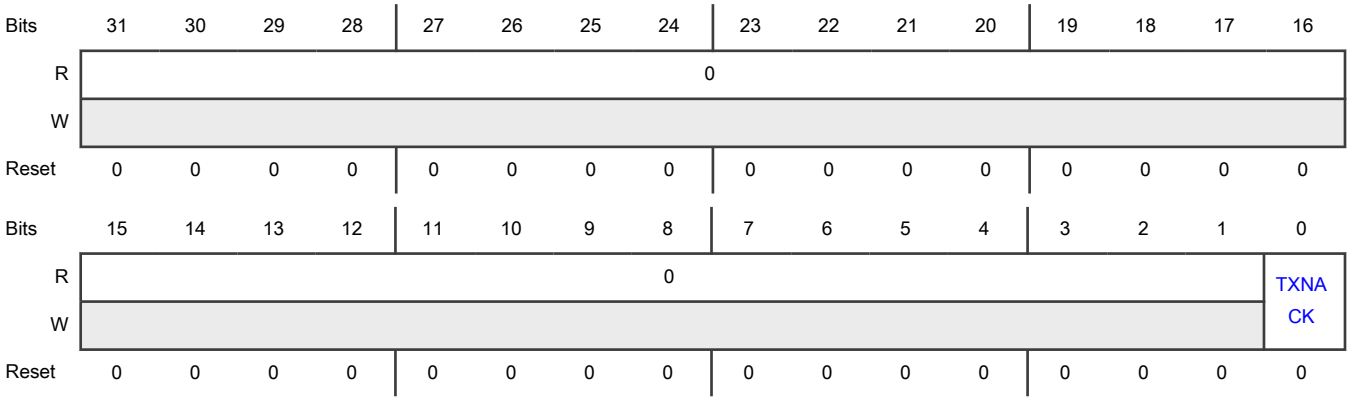
You can write to this register only when SCFGR1[ACKSTALL] = 1.

SCFGR1[ACKSTALL] enables clock stretching during the ACK-or-NACK bit slot. During this time, you can write to this register.

The logic ensures that the clock stretching continues for at least one bus clock cycle after this register is updated.

This clock stretching time can be extended via SCFGR2[CLKHOLD].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>TXNACK | Transmit NACK<br>Selects whether transmit ACK (logic 0) or NACK (logic 1) is returned on the bus by the I2C target after receiving each word.<br>• When SCFGR1[ACKSTALL] = 1, a transmit NACK signal must be written once for each matching address byte and each received word. SCFGR1[ACKSTALL] must be 1, because that setting stalls the data transfer until software reads the received word (and determines whether to respond with an ACK or NACK).<br>• To configure the default (ACK or NACK), you can write to this field when LPI2C target is disabled or idle.<br>0b - Transmit ACK<br>1b - Transmit NACK |

## 68.7.1.28 Target Transmit Data (STDR)

### Offset

| Register | Offset |
|---|---|
| STDR | 160h |

### Function

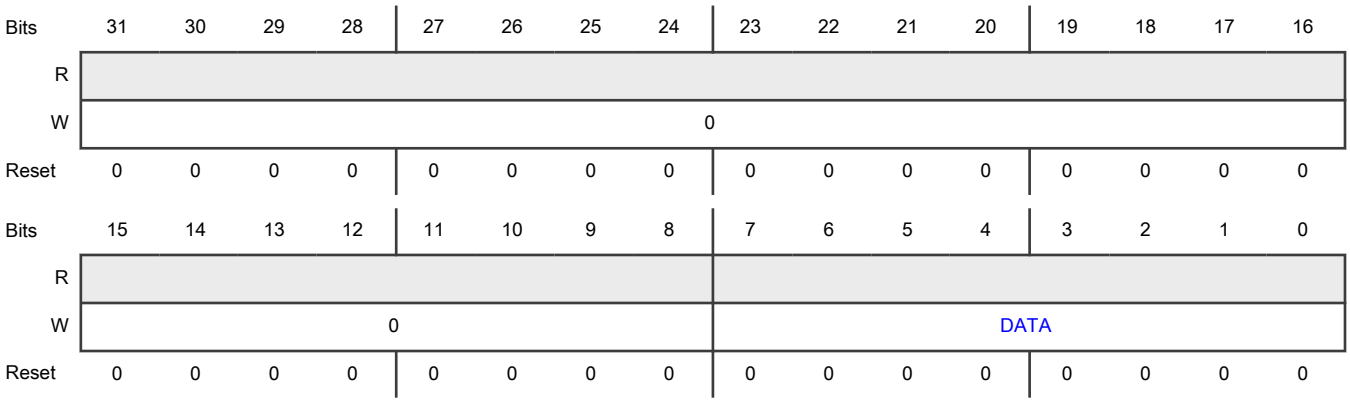
Contains the I2C target data to transmit.

Clock stretching (enabled or disabled) affects when the transmit data is transferred. SCFGR1[TXDSTALL] enables clock stretching during the first data bit of a target-transmit transfer.

If clock stretching is enabled (SCFGR1[TXDSTALL] = 1), the transmit data transfer is stalled until this register is updated. Clock stretching is extended by at least 1 bus clock cycle after this register is updated. Clock stretching can be delayed further by using SCFGR2[CLKHOLD].

If clock stretching is disabled (SCFGR1[TXDSTALL] = 0), the transmit data must be written before the start of the target-transmit transfer, otherwise SSR[FEF] is set.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>DATA | Transmit Data<br>Contains the I2C target data to transmit. Writing data to this register stores I2C target transmit data in this register. |

## 68.7.1.29 Target Receive Data (SRDR)

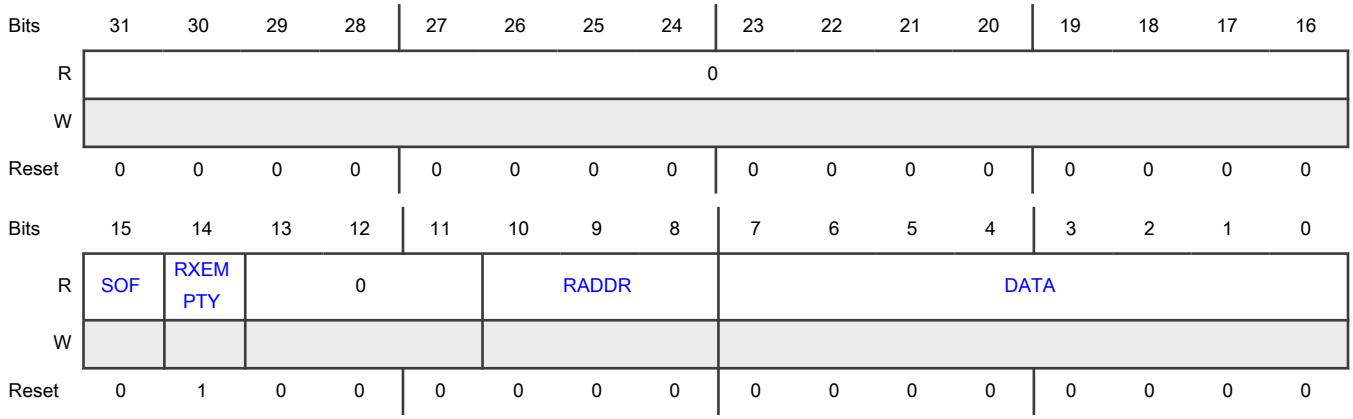
### Offset

| Register | Offset |
|---|---|
| SRDR | 170h |

### Function

Contains status of target receive data transfer.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>SOF | Start of Frame<br>Indicates whether this data word is the first data word since a (repeated) Start or Stop condition.<br>0b - Not first<br>1b - First |
| 14<br>RXEMPTY | Receive Empty<br>Indicates whether this register is empty.<br>0b - Not empty<br>1b - Empty |
| 13-11<br>— | Reserved |
| 10-8<br>RADDR | Received Address<br>Contains the address received by the IC2 target. When both SCFGR1[RXCFG] and SSR[AVF] are 1, bits [10:8] of SASR[RADDR] are returned. Otherwise, this field returns zero. |
| 7-0<br>DATA | Received Data<br>Contains the data received by the I2C target. When both SCFGR1[RXCFG] and SSR[AVF] are 1, bits [7:0] of SASR[RADDR] are returned. |

## 68.8 Glossary

| | |
|---|---|
| **HREQ** | Host request |
| **SCL** | Serial clock line |

| | |
|---|---|
| **SCLS** | Secondary serial clock line |
| **SDA** | Serial data line |
| **SDAS** | Secondary serial data line |

# Chapter 69
# Flexible I/O (FlexIO)

## 69.1 Chip-specific FlexIO information

### 69.1.1 FlexIO instances and configuration

The device contains one instance of FlexIO.

FlexIO is a highly configurable module providing a wide range of functionality including:

- Emulation of a variety of serial communication protocols like UART, I2C, SPI, SAI. It can work either as master or slave.
- Flexible 16-bit timers with support for a variety of trigger, reset, enable and disable conditions.

**Table 391. FlexIO instances**

| Chip | Instance | No. of pins | No. of timers | No. of shift registers | No. of bits in counter |
|---|---|---|---|---|---|
| MCXE317/ MCXE31B | FlexIO | 32 | 8 | 8 | 16 |
| MCXE315/ MCXE316 | FlexIO | 16 | 8 | 8 | 16 |

**Table 392. Data rate limitation**

| FlexIO operation | Data rate limitation |
|---|---|
| Master Tx | FlexIO_clk/4 |
| Slave Tx | FlexIO_clk/10 |
| Master Rx | FlexIO_clk/8 |
| Slave Rx | FlexIO_clk/6 |

**NOTE**

FlexIO supports 32 bit accesses only.

**NOTE**

When I2S is emulated on FlexIO, you could observe a skew between bit-clock (BCLK) and frame sync pulse amounting to 1/2 of BCLK.

**Table 393. FlexIO configuration**

| No. of pins | No. of timers | No. of shift registers | No. of bits in counter |
|---|---|---|---|
| 32 | 8 | 8 | 16 |

## 69.2 Overview

FLEXIO is a highly configurable module that provides:

- Emulation of various serial or parallel communication protocols.
- Flexible 16-bit timers with support for various trigger, reset, enable, and disable conditions.

- Programmable logic blocks that allow the implementation of digital logic functions on-chip and configurable interaction of internal and external modules.
- Programmable state machine for offloading basic system control functions from the CPU.

### 69.2.1 Block diagram

The following diagram provides a high-level overview of the FLEXIO timer and shifter configuration.

FLEXIO uses shifters, timers, and external triggers to shift data into or out of FLEXIO. As shown in the block diagram, timers control the timing of this data shift. You can configure the timers to use generic timer functions, external triggers, or various other conditions to determine the control.

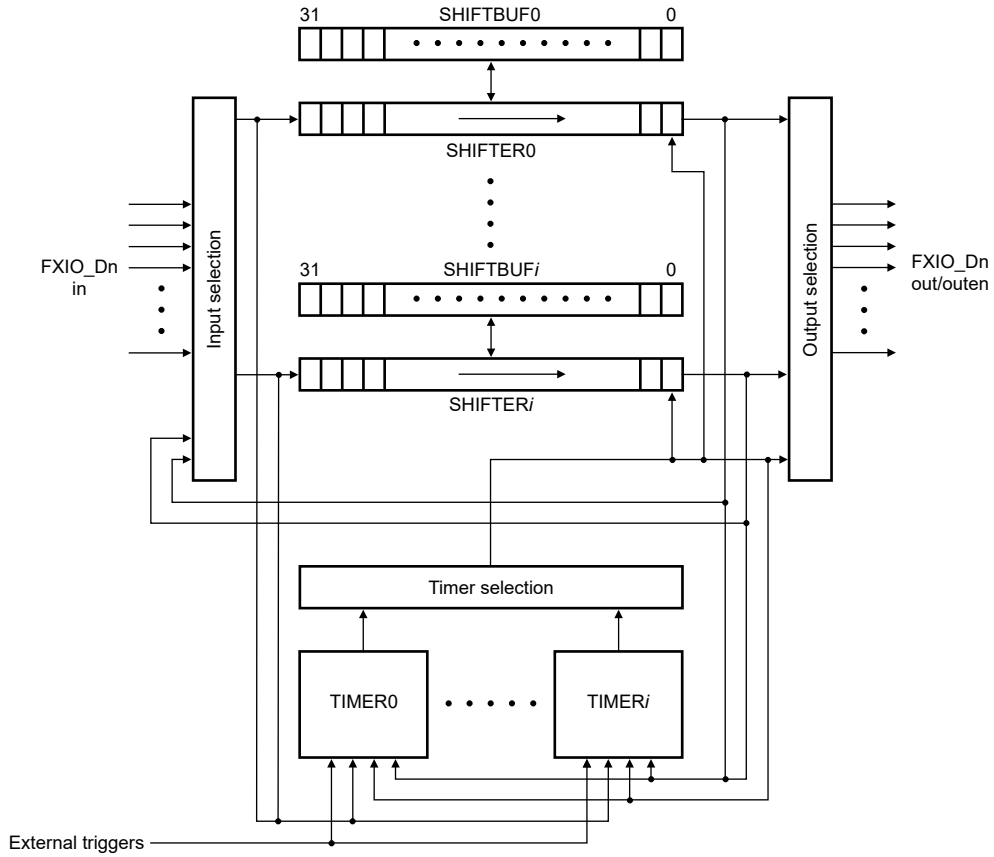



Figure 365. Block diagram

### 69.2.2 Features

- Array of 32-bit shift registers with transmit, receive, data match, logic, and state modes:
  - Double-buffered shifter operation for continuous data transfer
  - Shifter concatenation to support large transfer sizes
  - Automatic start and stop bit generation
  - 1, 2, 4, 8, 16, or 32 multi-bit shift widths for parallel interface support
  - Interrupt, DMA, or polled transmit and receive operation
- Highly flexible 16-bit timers with support for various internal or external triggers, reset, enable, and disable conditions:
  - Programmable baud rates independent of bus clock frequency

— Programmable logic mode for integrating external digital logic functions on-chip, or combining pin, shifter, or timer functions to generate complex outputs

— Programmable state machine for offloading basic system control functions from CPU, with support for up to eight states, eight outputs, and three selectable inputs per state

- Integrated general-purpose I/O registers and pin rising or falling edge interrupts to simplify software support
- Support for a wide range of protocols, including but not limited to:
  — UART
  — I2C
  — SPI
  — I2S
  — Camera IF
  — Motorola 68K or Intel 8080 bus
  — PWM or waveform generation
  — Input-capture (pulse-edge interval measurement), such as SENT

## 69.3 Functional description

### 69.3.1 Shifter operation

Shifters are responsible for buffering and shifting data into or out of FLEXIO. The timer assigned to the shifter controls the timing of shift, load, and store events via SHIFTCTL*n*[TIMSEL]. Shifters are designed to support either DMA, interrupt, or polled operations. The following figure provides a detailed view of the shifter microarchitecture.

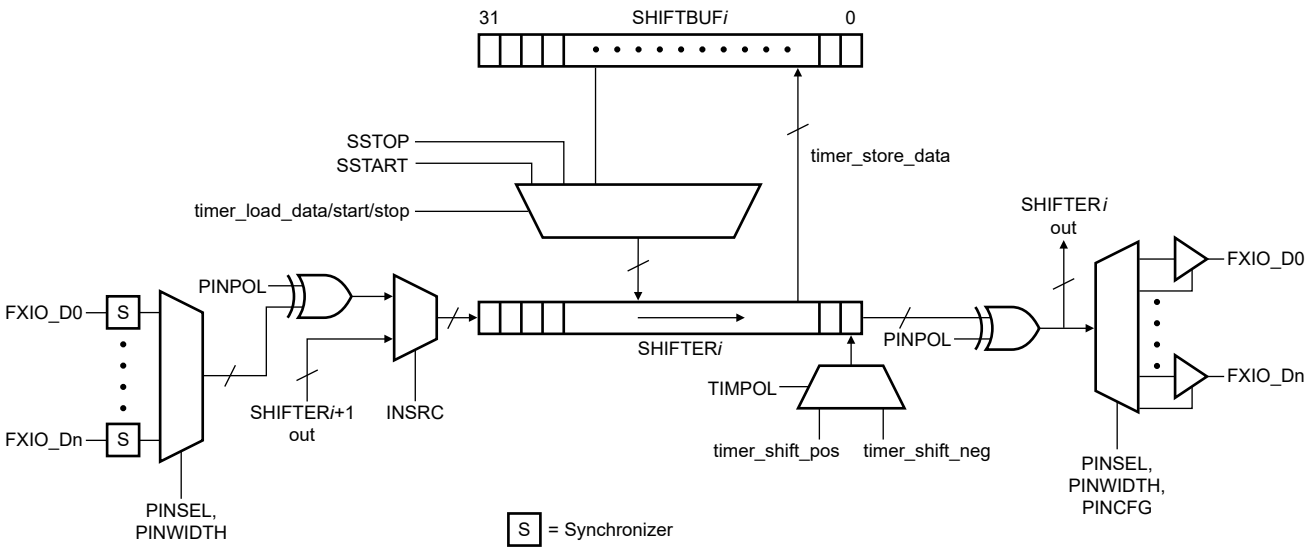



**Figure 366. Shifter microarchitecture**

#### 69.3.1.1 Transmit mode

In Transmit mode (SHIFTCTL*n*[SMOD] = 010b), the shifter loads data from Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) and shifts data out when the assigned timer signals a load event. An optional start and stop bit can be automatically loaded before or after SHIFTBUF register data by configuring either SHIFTCFG[SSTART] and TIMCFG[TSTART], or SHIFTCFG[SSTOP] and TIMCFG[TSTOP] in the shifter and timer.

**NOTE**

If a stop bit is enabled, the shifter immediately loads a stop bit when it is initially configured for Transmit mode.

The shifter status flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests are set when data has either been loaded from the SHIFTBUF register into the shifter or when the shifter is initially configured for Transmit mode. To clear the flag, write 1 or write new data to SHIFTBUF. In Transmit mode, write any value to the SHIFTBUF register to clear the corresponding shifter status flag, which is cleared regardless of what is writing to the register (DMA or interrupt), or the state of the DMA or interrupt enables. See the functional description of SHIFTSTAT[SSF] for information on how the flag is set and cleared for each mode.

The shifter error flag (SHIFTERR[SEF]) and any enabled interrupts are set when an attempt to load data from an empty SHIFTBUF register occurs (buffer underrun). Clear the flag by writing 1.

#### 69.3.1.2 Receive mode

When the assigned timer signals a store event in Receive mode (SHIFTCTL*n*[SMOD] = 001b), the shifter shifts and stores data in Shifter Buffer (SHIFTBUF0 - SHIFTBUF7). You can check for a start and stop bit before or after the shifter data is sampled by configuring either SHIFTCFG[SSTART] and TIMCFG[TSTART], or SHIFTCFG[SSTOP] and TIMCFG[TSTOP] in the shifter and timer.

The shifter status flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests are set when data is stored in the SHIFTBUF register from the shifter. To clear the flag, write 1 to or read the data from SHIFTBUF. Any read of the SHIFTBUF register clears the corresponding shifter status flag when the shifter is in Receive mode. The flag is cleared regardless of what is reading the register (DMA or interrupt) or the state of the DMA or interrupt enables. See the functional description of SHIFTSTAT[SSF] for information on how the flag is set or cleared for each mode.

The shifter error flag (SHIFTERR[SEF]) and any enabled interrupts are set either when an attempt to store data into a full SHIFTBUF register occurs (buffer overrun) or when a mismatch occurs on a start or stop bit check. Write 1 to clear the flag.

#### 69.3.1.3 Match Store mode

In Match Store mode (SHIFTCTL*n*[SMOD] = 100b), the shifter shifts data in, checks for a match result, and stores matched data in Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) when the assigned timer signals a store event. By configuring either SHIFTCFG[SSTART], TIMCFG[TSTART], and SHIFTCFG[SSTOP], or TIMCFG[TSTOP] in the shifter and timer, you can check for a start and stop bit before or after the shifter data is sampled. You can compare up to 16 bits of data using SHIFTBUF[31:16] to configure the data to be matched and SHIFTBUF[15:0] to mask the match result.

The shifter status flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests are set when a match occurs and the matched data is stored in the SHIFTBUF register from the shifter. To clear the flag, read the matched data from the SHIFTBUF register or write 1 to the flag. Any read of the SHIFTBUF register clears the corresponding shifter status flag when the shifter is configured in Match Store mode. The flag is cleared regardless of what is reading the register (DMA or interrupt) or the state of the DMA or interrupt enables. See the functional description for SHIFTSTAT[SSF] to know how the flag is set or cleared for each mode.

The shifter error flag (SHIFTERR[SEF]) and any enabled interrupts are set when an attempt to store matched data into a full SHIFTBUF register occurs (buffer overrun), or when a mismatch occurs on a start or stop bit check. Write 1 to clear the flag.

#### 69.3.1.4 Match Continuous mode

In Match Continuous mode (SHIFTCTL*n*[SMOD] = 101b), the shifter shifts data in and continuously checks for a match result whenever a shift event is signaled by the assigned timer. You can compare up to 16 bits of data using SHIFTBUF[31:16] to configure the data to be matched and SHIFTBUF[15:0] to mask the match result.

The shifter status flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests are set when a match occurs. The flag clears automatically as soon as no match exists between the shifter data and Shifter Buffer (SHIFTBUF0 - SHIFTBUF7).

You cannot clear the flag by reading the SHIFTBUF register.

The shifter error flag (SHIFTERR[SEF]) and any enabled interrupts are set when a match occurs. To clear the flag, write 1 or perform a read from the SHIFTBUF register.

### 69.3.1.5 State mode

State mode enables you to implement any state machine with up to eight states, eight outputs, and three selectable inputs per state. This feature allows basic control functions to be offloaded from the CPU.

In State mode (SHIFTCTL*n*[SMOD] = 110b), when the shifter is selected by the current state pointer (SHIFTSTATE[STATE]), use the SHIFTBUF register to drive the output and compute the next state values. The following figure provides a detailed view of the shifter microarchitecture when configured for State mode.

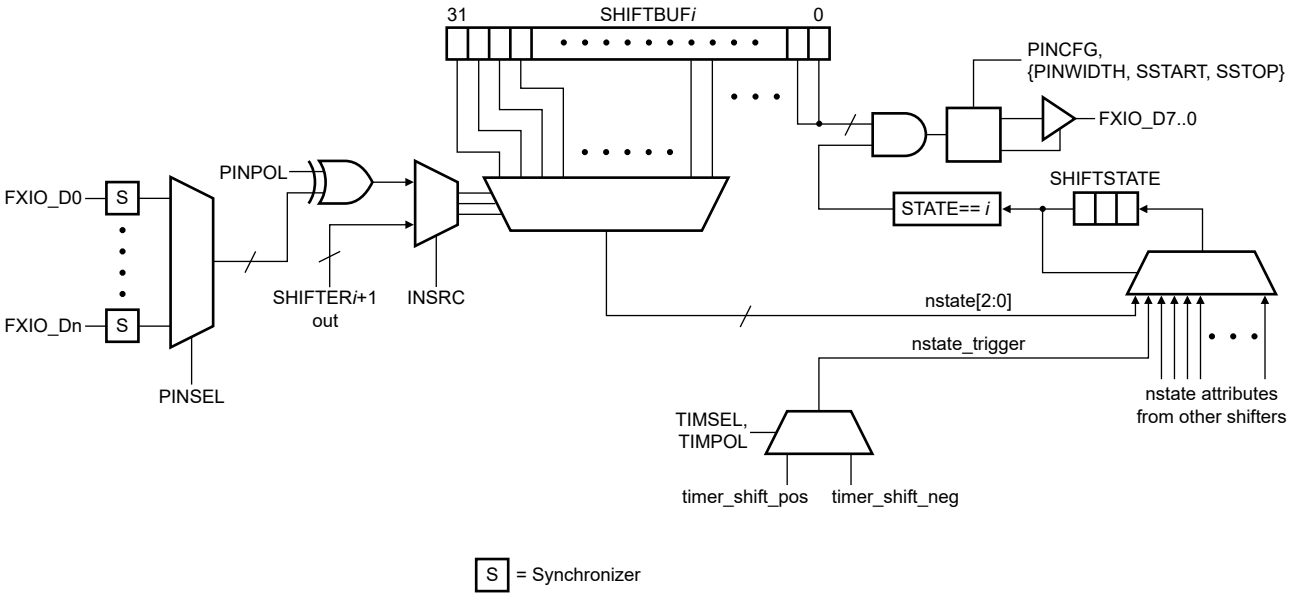



Figure 367. State microarchitecture

When the current state pointer selects a specific shifter (shifter *n*), output pins FXIO_D[7:0] are driven by SHIFTBUF*n*[31:24]; the configuration is defined by SHIFTCTL*n*[PINCFG]. Write 1 to Shifter Configuration (SHIFTCFG0 - SHIFTCFG7) {PWIDTH[3:0],SSTOP[1:0],SSTART[1:0]} to disable the output drive on pins FXIO_D[7:0] for state machine applications that require less than eight output pins.

Use the three input pins selected by SHIFTCTL*n*[PINSEL] and SHIFTBUF*n*[23:0] to compute the next state value.

**NOTE**

Each state can use a different set of three input pins.

The following table shows how the next state value is computed when the current state pointer is pointing to shifter *n*.

Table 394. Next state computation for SHIFTSTATE[STATE] = *n*

| FXIO_D[PINSEL + 2] | FXIO_D[PINSEL + 1] | FXIO_D[PINSEL] | Next state value |
|---|---|---|---|
| 0 | 0 | 0 | SHIFTBUF*n*[2:0] |
| 0 | 0 | 1 | SHIFTBUF*n*[5:3] |
| 0 | 1 | 0 | SHIFTBUF*n*[8:6] |
| 0 | 1 | 1 | SHIFTBUF*n*[11:9] |
| ... | ... | ... | ... |
| 1 | 1 | 1 | SHIFTBUF*n*[23:21] |

**NOTE**

You can configure other shifters and timers to drive the input pins of a given state, allowing you to create complex combinations of shifters and timers as needed. For example, the output of a shifter configured for Logic mode can be used to drive a state machine input.

The next state transition is triggered using the timer output selected by SHIFTCTL*n*[TIMSEL], with polarity controlled by SHIFTCTL*n*[TIMPOL].

**NOTE**

Each state can use a different timer to trigger each next state transition, allowing various internal or external trigger sources and clocking configurations to be used. See Timer section for more information.

The current state pointer defaults to shifter 0 at reset; however, you can write to select a different shifter for the initial state. If the current state pointer selects a shifter that is not configured for State mode, then outputs are not driven and the next state transition is never triggered.

The shifter status flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests are set when the shifter is selected by the current state pointer. The flag is cleared when the current state pointer is updated to a different shifter.

### 69.3.1.6 Logic mode

Logic mode enables you to implement a small amount of programmable digital logic within a FLEXIO shifter.

In Logic mode (SHIFTCTL*n*[SMOD] = 111b), use Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) to implement a 5-input, 32-bit programmable logic lookup table. The following figure provides a detailed view of shifter microarchitecture when configured for Logic mode.

Use the SHIFTBUF register to configure the lookup table for the four pin inputs. You can also use SHIFTER*n* to configure a feedback or delayed pin source as the fifth input to the lookup table.

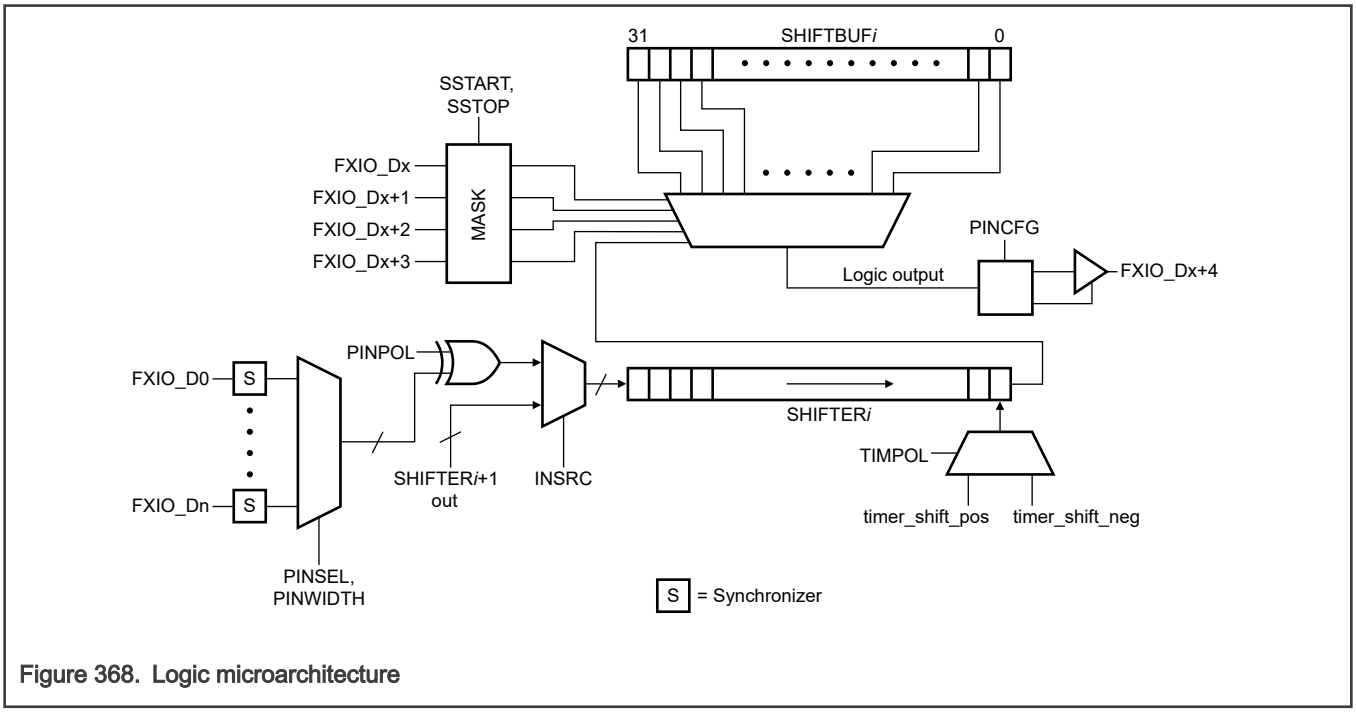



**Figure 368. Logic microarchitecture**

The lookup table is driven using four pin inputs (maskable using SHIFTCFG[SSTOP] and SHIFTCFG[SSTART]), plus one input from the internal shifter. It can be configured to drive an output pin using SHIFTCTL[PINCFG]. Pin inputs and outputs are fixed for each logic lookup table and are not selectable. The following table lists the logic output value selected by the lookup table for shifter *n*.

Table 395. Logic lookup table for shifter *n*

| SHIFTER*n*[0] | FXIO_D[*x* + 3][1] | FXIO_D[*x* + 2] | FXIO_D[*x* + 1] | FXIO_D[*x*] | Logic output to FXIO_D[*x* + 4] |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | SHIFTBUF*n*[0] |
| 0 | 0 | 0 | 0 | 1 | SHIFTBUF*n*[1] |
| 0 | 0 | 0 | 1 | 0 | SHIFTBUF*n*[2] |
| 0 | 0 | 0 | 1 | 1 | SHIFTBUF*n*[3] |
| ... | ... | ... | ... | ... | ... |
| 1 | 1 | 1 | 1 | 1 | SHIFTBUF*n*[31] |

1. for shifters *n* = 0...3, *x* = *n*

   for shifter *n* = 4...7, *x* = *n* + 4

To minimize output glitches, use SHIFTCFG*n*[SSTOP] and SHIFTCFG*n*[SSTART] to mask unused input pins. When these fields are 1, {SSTOP[1:0] and SSTART[1:0]} mask FXIO_D[*x* + 3]...FXIO_D[*x*] inputs respectively so that any transitions on these pins do not cause the logic output to glitch.

**NOTE**

You can configure other shifters and timers to drive the input pins of a given lookup table, allowing you to concatenate lookup tables or create complex combinations of shifters and timers as needed.

SHIFTCFG[PWIDTH] controls the number of delay stages introduced by the internal shifter input (SHIFTER*n*[0]). For example, when configured for a 1-bit shift (SHIFTCFG[PWIDTH] = 0), the internal shifter introduces a 32-shift clock delay before passing its input (selected by SHIFTCTL[PINSEL]) to the lookup table. When configured for a 32-bit shift (SHIFTCFG[PWIDTH] = 16...31), the internal shifter introduces a 1-shift clock delay to its input.

The shifter status flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests are set whenever the output pin allocated to the logic lookup table has a value of 1 after being synchronized with the FLEXIO clock. The flag clears when the output pin has a value of 0. This also allows SHIFTSTAT[SSF] to be used as a trigger to a timer if needed.

The shifter error flag (SHIFTERR[SEF]) and any enabled interrupts are set when the output pin allocated to the logic lookup table is asserted. Clear the flag by writing 1 to it.

The Logic mode input pins, including pins driven by other shifters and timers, are synchronized with the FLEXIO functional clock before they are input to the programmable logic lookup table.

### 69.3.2 Timer operation

The FLEXIO 16-bit timers control the loading, shifting, and storing of the shift registers. The counters load the contents of the compare register and decrement down to zero on the FLEXIO clock. The counters can perform generic timer functions such as generating a clock, select output, or a PWM waveform. You can configure these timers to perform any of the following functions:

- Enable in response to a trigger, pin, or shifter condition.
- Decrement always or only on a trigger or pin edge.
- Reset in response to a trigger or pin condition.
- Disable on a trigger or pin condition or on a timer compare.

Timers can optionally include a start condition and a stop condition.

Although each timer operates independently, you can configure a timer to enable or disable at the same time as the previous timer (for example, timer 1 can enable or disable at the same time as timer 0) and a timer output can be used to trigger any other timer. The trigger used by each timer is configured independently as a timer output, shifter status flag, pin input, or an external trigger

input. The trigger configuration is separate from pin configuration; you can perform it to configure input, output data, or output enable. See the chip-specific FLEXIO information for information on external trigger connections.

You must configure Timer Configuration (TIMCFG0 - TIMCFG7) before writing 1 to TIMCTL*n*[TIMOD].

#### 69.3.2.1 Timer 8-bit Baud Counter mode

In 8-bit Baud Counter mode, the 16-bit counter is divided into two 8-bit counters. The lower 8 bits are used to configure the baud rate of the shift clock and the upper 8 bits are used to configure the number of shift clock edges in the transfer. When the lower 8 bits decrement to zero, the timer output is toggled and the lower 8 bits reload from the compare register. The upper 8 bits decrement when the lower 8 bits become zero and decrement.

**NOTE**

A timer reset event in 8-bit Baud Counter mode only resets the lower 8-bit counter. The upper 8-bit counter is not affected and can decrement if the timer reset is configured to update the state of the timer output, which toggles as a result of the timer reset event.

A timer compare event occurs when the upper 8 bits equal zero and decrement. The timer status flag is set on a timer compare event.

#### 69.3.2.2 Timer 8-bit High PWM mode

In 8-bit High PWM mode, the 16-bit counter is divided into two 8-bit counters. The lower 8 bits are used to configure the timer output high period and the upper 8 bits are used to configure the timer output low period. The lower 8 bits decrement when the output is high. When the lower 8 bits become zero and decrement, the timer output is cleared and the lower 8 bits are reloaded from the compare register. The upper 8 bits decrement when the output is low. When the upper 8 bits become zero and decrement, the timer output is set and the upper 8 bits are reloaded from the compare register.

A timer compare event occurs when the upper 8 bits become zero and decrement. The timer status flag is set on a timer compare event.

#### 69.3.2.3 Timer 16-bit Counter mode

In 16-bit Counter mode, you can use the 16-bit counter to configure either the baud rate of the shift clock (for example, TIMDEC[1:0] ≠ 10 or 11) or the number of shift clock edges in the transfer (for example, TIMDEC[1:0] = 10 or 11). When the 16-bit counter equals zero and decrements, the timer output toggles and the counter reloads from the compare register.

A timer compare event occurs when the 16-bit counter equals zero and decrements. The timer status flag is set on a timer compare event.

#### 69.3.2.4 Timer 16-bit Counter Disable mode

In 16-bit Counter Disable mode, the 16-bit counter can be used to configure either the baud rate of the shift clock (for example, TIMDEC[1:0] ≠ 10 or 11) or the number of shift clock edges in the transfer (for example, TIMDEC[1:0] = 10 or 11). When the 16-bit counter equals zero and decrements, the timer output toggles and the counter reloads from the compare register.

A timer compare event occurs when the 16-bit counter equals zero and decrements. The timer status flag is set on a timer disable event.

#### 69.3.2.5 Timer 8-bit Word Counter mode

In 8-bit Word Counter mode, the 16-bit counter is divided into two 8-bit counters. The lower 8 bits are used to configure the number of shift clock edges in each word and the upper 8 bits are used to configure the number of words in the transfer. When the lower 8 bits decrement to zero, the timer output is toggled and the lower 8 bits reload from the compare register. The upper 8 bits only decrement when the lower 8 bits become zero and decrement.

A timer compare event occurs when the lower 8 bits become zero and decrement. The timer status flag is set when the upper 8 bits become zero and decrement.

#### 69.3.2.6 Timer 8-bit Low PWM mode

In 8-bit Low PWM mode, the 16-bit counter is divided into two 8-bit counters. The lower 8 bits are used to configure the timer output low period and the upper 8 bits are used to configure the timer output high period. The lower 8 bits decrement when the output is low. When the lower 8 bits become zero and decrement, the timer output is set and the lower 8 bits are reloaded from the compare register. The upper 8 bits decrement when the output is high. When the upper 8 bits become zero and decrement, the timer output is cleared and the upper 8 bits are reloaded from the compare register.

A timer compare event occurs when the upper 8 bits become zero and decrement. The timer status flag is set on a timer compare event.

#### 69.3.2.7 Timer enable and start functions

The following events occur when you configure TIMCTL*n*[TIMOD] for the desired mode and the condition configured by the timer enable (TIMCFG*n*[TIMENA]) is detected. When TIMCTL*n*[ONETIM] is 1, the timer status flag must be clear to generate a timer enable event; otherwise, the timer enable event is blocked. You can use this to enforce software intervention after each timer iteration:

- The timer counter loads the current value of the compare register and starts decrementing, as configured by TIMCFG*n*[TIMDEC].
- The timer output may update to its initial state depending on the configuration of TIMCFG*n*[TIMOUT]. Shifters that are controlled by this timer do not see this as a rising edge on the timer shift clock.
- Transmit shifters controlled by this timer either output their start bit value or load the shift register from the shift buffer and output the first bit, as configured by SHIFTCFG*n*[SSTART].

If the timer start bit is enabled, the timer counter reloads with the compare register on the first rising edge of the shift clock after the timer starts decrementing. If there is no falling edge on the shift clock before the first rising edge (for example, when TIMCFG*n*[TIMOUT] = 1), a shifter that is configured to shift on the falling edge and load on the first shift does not load correctly.

#### 69.3.2.8 Timer decrement and reset functions

The timer generates the timer output and timer shift clock depending on the fields, TIMCTL*n*[TIMOD] and TIMCFG*n*[TIMDEC]. The shifter clock is either equal to the timer output (when TIMCFG*n*[TIMDEC] ≠ 10 or 11) or equal to the decrement clock (when TIMCFG*n*[TIMDEC] = 10 or 11). If you configure TIMCFG*n*[TIMDEC] to decrement from a pin or trigger, the timer decrements on both rising and falling edges.

If a timer is configured to decrement on the FLEXIO functional clock divided by 16 or 256 (when TIMCFG*n*[TIMDEC] = 100 or 101), then a common prescaler that is shared by all timers is used to generate the two divide ratios. This prescaler is reset when all timers are either idle or configured not to use the prescaler (TIMCFG*n*[TIMDEC] ≠ 100 or 101).

If you configure the timer to reset as determined by TIMCFG*n*[TIMRST], then the timer counter loads the current value of the compare register again. You can configure the timer output and timer shift clock to update on timer reset, as configured by TIMCFG*n*[TIMOUT]. If the time output toggles as a result of the timer reset, this can result in a timer shift clock edge. In 8-bit Baud Counter mode, this also decrements the upper 8 bits of the counter.

In general, when the timer counter decrements to zero, a timer compare event is triggered. The timer compare event causes:

- The timer counter to load the contents of the timer compare register.
- The timer output to toggle.
- Any configured transmit shift registers to load.
- Any configured receive shift registers to store.

Depending on the timer mode, the timer status flag may also be set.

#### 69.3.2.9 Timer disable and stop functions

When the timer is configured to add a stop bit on each compare, the following additional events occur:

- Transmit shifters controlled by this timer output their stop bit value (if configured by SHIFTCFG*n*[SSTOP]).

- Receive shifters controlled by this timer store the contents of the shift register in their shift buffer, as configured by SHIFTCFG*n*[SSTOP].
- The timer counter reloads the current value of the compare register on the first rising edge of the shifter clock after the compare.

If you configure the timer to insert a stop bit on each compare, you must configure the transmit shifters to load on the first shift.

When the condition configured by timer disable (TIMCFG*n*[TIMDIS]) is detected, the following events occur:

- Timer counter reloads the current value of the compare register and starts decrementing as configured by TIMCFG*n*[TIMDEC].
- Timer output clears. Shifters that are controlled by this timer do not see this as a falling edge on the timer shift clock, but can generate a shift event if the timer shift clock otherwise generates one.
- Transmit shifters controlled by this timer output their stop bit value (if configured by SHIFTCFG*n*[SSTOP]).
- Receive shifters controlled by this timer store the contents of the shift register in their shift buffer, as configured by SHIFTCFG*n*[SSTOP].

If the timer stop bit is enabled, the timer counter continues decrementing until the next rising edge of the shift clock is detected, at which point it finishes decrementing. Although the timer output is forced low during the stop bit, the timer shift clock can toggle during the stop bit. The timer output does not generate shift events during the stop bit.

A timer enable condition can be detected in the same cycle as a timer disable condition (if timer stop bit is disabled), or on the first rising edge of the shift clock after the disable condition (if stop bit is enabled). When TIMCTL*n*[ONETIM] is 1, the timer status flag must be clear before the next timer enable condition is detected. When the timer is in the stop state condition, receive shift registers with stop bit enabled store the contents of the shift register into the shift buffer and verify the state of the input data on the configured shift edge. If there is no configured edge between the timer disable and the next rising edge of the shift clock, then the final store and verify do not occur.

### 69.3.3 Pin operation

The pin configuration for each timer and shifter can be set to use any FLEXIO pin with either polarity. You can configure each timer and shifter as an input, output data, output enable, or bidirectional output. A pin configured for output enable can be used as an open drain (with inverted polarity because the output enable assertion causes logic zero to be output on the pin) or to control the enable on the bidirectional output. You can configure any timer or shifter to control the output enable for a pin where the bidirectional output data is driven by another timer or shifter.

When more than one shifter or timer is configured to use the same pin for output data, then the output data to the pin is combined using an OR function. This also applies if more than one timer or shifter is configured to use the same pin for output enable.

#### 69.3.3.1 Parallel interface

You can configure shifters to use multiple FLEXIO pins simultaneously by using SHIFTCFG*n*[PWIDTH], which is used to configure the following settings of a shifter:

1. Number of bits shifted per shift clock.
2. Number of pins driven by the shifter per shift clock (only on shifters supporting parallel transmit—that is, SHIFTER0 and SHIFTER4.)
3. Number of pins sampled by the shifter per shift clock (only on shifters supporting parallel receive—that is, SHIFTER3 and SHIFTER7.)

When configured for parallel shift, either 4, 8, 16, or 32 bits can be shifted on every shift clock. If an adjacent shifter is selected as the input source (SHIFTCFG*n*[INSRC] = 1), the least significant 4, 8, 16, or 32 bits from the adjacent shifter are sampled on each shift clock.

For shifters supporting parallel receive (SHIFTER3, SHIFTER7), you can configure the shifter to sample multiple pins (INSRC = 0), with PWIDTH and PINSEL selecting the pins as FXIO_D[PINSEL+PWIDTH]:FXIO_D[PINSEL].

**NOTE**

If PWIDTH is less than the number of bits being shifted on each shift clock, then the most significant bits are masked with 0. For example, if PINSEL = 7 and PWIDTH = 6, then SHIFTER[31:24] samples {0,0,FXIO_D[12:7]} on each shift clock.

For shifters supporting parallel transmit (SHIFTER0, SHIFTER4), you can configure the shifter to drive multiple pins using SHIFTCTL*n*[PINCFG], with PWIDTH and PINSEL selecting the pins as follows: FXIO_D[PINSEL+PWIDTH]:FXIO_D[PINSEL].

**NOTE**

If PWIDTH is less than the number of bits being shifted on each shift clock, then the most significant pins are not driven. For example, if PINSEL = 7 and PWIDTH = 6, then SHIFTER[5:0] drives only FXIO_D[12:7] on each shift clock.

#### 69.3.3.2 Pin synchronization

When you configure a pin as an input (this includes a timer trigger configured as a pin input), the input signal is first synchronized with the FLEXIO clock before a timer or shifter could use the signal. This introduces a small latency of 0.5–1.5 FLEXIO clock cycles when using an external pin input to generate an output or control a shifter. This sets the maximum setup time at 1.5 FLEXIO clock cycles.

If an input is used by more than one timer or shifter, then the synchronization occurs once to ensure any edge is seen on the same cycle by all timers and shifters using that input.

**NOTE**

FLEXIO pins are also connected internally. Configuring a FLEXIO shifter or timer to output data on an unused pin makes an internal connection that allows other shifters and timers to use this pin as an input. This allows a shifter output to trigger a timer or a timer output to be shifted into a shifter. This path is also synchronized with the FLEXIO clock and therefore incurs a one-cycle latency.

When using a pin input as a timer trigger, timer clock, or shifter data input, the following synchronization delays occur:

- 0.5–1.5 FLEXIO clock cycles for an external pin
- One FLEXIO clock cycle for an internally driven pin

See Application information for timing considerations such as output valid time and input setup time for specific applications (SPI controller, SPI target, I2C controller, I2S controller, and I2S target).

#### 69.3.3.3 Pin override

You can change the state of any FLEXIO pin at any time. Pin Output Enable (PINOUTE) configures any pin as an output and drives that pin with the value in Pin Output Data (PINOUTD).

Alias registers for PINOUTE and data registers also exist. Writing a logic 1 to an alias register updates the corresponding register fields in both PINOUTE and PINOUTD as follows:

- Pin Output Disable (PINOUTDIS) clears Pin Output Enable (PINOUTE) and Pin Output Data (PINOUTD).
- Pin Output Clear (PINOUTCLR) sets Pin Output Enable (PINOUTE) and clears Pin Output Data (PINOUTD).
- Pin Output Set (PINOUTSET) sets Pin Output Enable (PINOUTE) and Pin Output Data (PINOUTD).
- Pin Output Toggle (PINOUTTOG) sets Pin Output Enable (PINOUTE) and toggles Pin Output Data (PINOUTD).

#### 69.3.3.4 Pin interrupt

You can read the state of any FLEXIO pin at any time and also configure any pin to set a status flag when either a rising or falling edge is detected on that pin. Additionally, you can configure the pin status flag to generate an interrupt.

### 69.3.4 Low-power modes

FLEXIO remains functional during low-power modes, if the FLEXIO functional clock remains enabled.

### 69.3.5 Debug mode

FLEXIO remains functional in Debug mode, provided the value of CTRL[DBGE] is 1.

### 69.3.6 Clocking

**Table 396. FLEXIO clocks**

| Clock | Description |
|---|---|
| Functional clock | Is asynchronous to the bus clock and can remain enabled in low-power modes. You must enable the FLEXIO functional clock before accessing any of the FLEXIO registers. Provided the FLEXIO functional clock is at least equal to the bus clock, you can configure CTRL[FASTACC] to support fast register accesses. |
| Bus clock | Is used only for bus accesses to the control and configuration registers. |

### 69.3.7 Reset

**Table 397. FLEXIO reset types**

| Reset | Description |
|---|---|
| Chip reset | Resets the FLEXIO logic and registers to their default states on chip reset. |
| Software reset | Resets, using CTRL[SWRST], all logic and registers to their default states, except for the Control register. |

### 69.3.8 Interrupts and DMA requests

The following table shows the status flags that generate the FLEXIO interrupt and DMA requests.

**Table 398. FLEXIO interrupts and DMA requests**

| Flag | Description | Interrupt | DMA request | Low-power wake-up |
|---|---|---|---|---|
| SHIFTSTAT[SSF] | Shifter status flag | Y | Y | Y |
| SHIFTERR[SEF] | Shifter error flag | Y | N | Y |
| TIMSTAT[TSF] | Timer status flag | Y | Y | Y |
| PINSTAT[PSF] | Pin status flag | Y | N | Y |
| TRGSTAT[ETSF] | External trigger status flag | Y | N | Y |

### 69.3.9 Peripheral triggers

The connection between FLEXIO peripheral triggers and other peripherals is device-specific.

#### 69.3.9.1 Output triggers

Each FLEXIO timer generates an output trigger equal to the timer output. The output trigger is not affected by the timer pin polarity configuration.

#### 69.3.9.2 Input trigger

FLEXIO supports multiple external trigger inputs that can be used to trigger one or more FLEXIO timers. If a rising edge is detected on an external trigger when FLEXIO is enabled, then the external trigger status flag is set. The external triggers are synchronized to the FLEXIO functional clock and must assert for at least two cycles of the FLEXIO functional clock to be sampled correctly.

## 69.4 External signals

**Table 399. External signals**

| Signal | Description | Direction |
|---|---|---|
| FXIO_D*n* (*n* = 0...31) | Bidirectional FLEXIO shifter and timer pin | Input or output |

## 69.5 Initialization

Perform the following procedure to initialize FLEXIO registers:

1. Enable FLEXIO by writing 1 to CTRL[FLEXEN].
2. Configure shift registers for the given application. It is recommended to write to Shifter Configuration (SHIFTCFG0 - SHIFTCFG7) before writing to the corresponding register, Shifter Control (SHIFTCTL0 - SHIFTCTL7).
3. Configure timer registers for the given application. It is recommended to write to Timer Compare (TIMCMP0 - TIMCMP7) and Timer Configuration (TIMCFG0 - TIMCFG7) before writing to the corresponding register, Timer Control (TIMCTL0 - TIMCTL7).
4. Enable interrupts and/or DMA requests, as appropriate, for the given application.
5. Write transmit data to initiate a transfer (depending on the given application).

## 69.6 Application information

This section provides examples for a variety of FLEXIO module applications. See FLEXIO register descriptions for more information.

### 69.6.1 UART transmit

UART transmit can be supported using one timer, one shifter, and one pin (two pins, if supporting CTS). The start and stop bit insertion is handled automatically, and multiple transfers are supported using the DMA controller. The timer status flag is used to indicate when the stop bit of each word is transmitted.

Break and idle characters require software intervention. Before transmitting a break or idle character, you must modify SHIFTCFG*n*[SSTART] and SHIFTCFG*n*[SSTOP] to transmit the required state, and the data to transmit must equal FFh or 00h. Supporting a second stop bit requires the stop bit to be inserted into the data stream using software (and increasing the number of bits to transmit). When performing byte writes to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) (or Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7) for transmitting MSB first), the rest of the register remains unaltered. This allows an address mark bit or additional stop bit to remain undisturbed.

**NOTE**

FLEXIO does not support automatic insertion of parity bits.

**Table 400. UART transmit configuration**

| Register | Value | Configuration |
|---|---|---|
| SHIFTCFG*n* | 0000_0032h | Configure start bit of 0 and stop bit of 1. |

*Table continues on the next page...*

Table 400. UART transmit configuration (continued)

| Register | Value | Configuration |
|---|---|---|
| SHIFTCTL*n* | 0003_0002h | Configure transmit using timer 0 on the positive edge of clock with output data on pin 0. You can configure the PINPOL field to invert output data, or support open-drain by writing 1h to the PINPOL and PINCFG fields. |
| TIMCMP*n* | 0000_0F01h | Configure 8-bit transfer with baud rate of divide by 4 of the FLEXIO clock. Set TIMCMP[15:8] as (number of bits × 2) - 1, and set TIMCMP[7:0] as (baud rate divider ÷ 2) - 1. |
| TIMCFG*n* | 0000_2222h | Configure start bit, stop bit, enable on trigger asserted and disable on compare. You can support CTS by configuring the TIMENA field as 3h. |
| TIMCTL*n* | 01C0_0001h | Configure the dual 8-bit counter using the shifter 0 status flag as an inverted internal trigger source. To support CTS, configure the PINSEL (for pin 1) and PINPOL fields as 1h. |
| SHIFTBUF*n* | Data to transmit | Transmit data can be written to SHIFTBUF[7:0] to initiate an 8-bit transfer. Use the shifter status flag to indicate when data can be written using an interrupt or a DMA request. Write to SHIFTBUFBBS[7:0] instead to support MSB first transfer. |

The following table shows an alternative configuration that supports slower baud rates. This configuration requires two timers.

Table 401. UART transmit configuration for slow baud rate

| Register | Value | Configuration |
|---|---|---|
| SHIFTCFG*n* | 0000_0032h | Configure start bit of 0 and stop bit of 1. |
| SHIFTCTL*n* | 0003_0002h | Configure transmit using timer 0 on the positive edge of clock with output data on pin 0. Invert output data by writing 1 to the PINPOL field. Support open-drain by configuring the PINPOL and PINCFG fields as 1h. |
| TIMCMP*n* | 0000_000Fh | Configure for 8-bit transfer, and configure TIMCMP[15:0] as (number of bits × 2) - 1. |
| TIMCFG*n* | 0030_2622h | Configure start bit, stop bit, enable on trigger rising edge, decrement on trigger and disable on compare. |

*Table continues on the next page...*

Table 401. UART transmit configuration for slow baud rate (continued)

| Register | Value | Configuration |
|---|---|---|
| TIMCTL*n* | 0740_0003h | Configure the 16-bit counter using the timer 1 output as an internal trigger source. |
| TIMCMP(*n* + 1) | 0000_0001h | Configure baud rate of divide by 4 of the FLEXIO clock, and configure TIMCMP[15:0] as (baud rate divider ÷ 2) - 1. |
| TIMCFG(*n* + 1) | 0000_1200h | Configure enable on trigger asserted and disable on timer 0 disable. You can configure the TIMEN field as 3h to support CTS. |
| TIMCTL(*n* + 1) | 01C0_0003h | Configure the 16-bit counter using the shifter 0 status flag as an inverted internal trigger source. You can support CTS by configuring the PINSEL (for pin 1) and PINPOL fields as 1h. |
| SHIFTBUF*n* | Data to transmit | Transmit data can be written to SHIFTBUF[7:0] to initiate an 8-bit transfer. Use the shifter status flag to indicate when data can be written using an interrupt or a DMA request. Write to SHIFTBUFBBS[7:0] instead to support MSB first transfer. |

### 69.6.2 UART receive

UART receive can be supported using one timer, one shifter, and one pin (two timers and two pins, if supporting RTS). The start and stop bit verification is handled automatically and multiple transfers are supported using the DMA controller. The timer status flag is used to indicate when the stop bit of each word is received.

FLEXIO does not support triple voting of the received data, which is sampled only once in the middle of each bit. You can use a timer to implement a glitch filter on the incoming data and a different timer to detect an idle line of programmable length. Break characters cause the error flag to set, and the shifter buffer register returns 00h.

**NOTE**

FLEXIO does not support automatic verification of parity bits.

Table 402. UART receiver configuration

| Register | Value | Configuration |
|---|---|---|
| SHIFTCFG*n* | 0000_0032h | Configure start bit of 0 and stop bit of 1. |
| SHIFTCTL*n* | 0080_0001h | Configure receive using timer 0 on the negative edge of clock with input data on pin 0. You can invert input data by writing 1 to the PINPOL field. |
| TIMCMP*n* | 0000_0F01h | Configure 8-bit transfer with baud rate of divide by 4 of the FLEXIO clock. Set |

*Table continues on the next page...*

Table 402. UART receiver configuration (continued)

| Register | Value | Configuration |
|---|---|---|
| | | TIMCMP[15:8] as (number of bits × 2) - 1, and set TIMCMP[7:0] as (baud rate divider ÷ 2) - 1. |
| TIMCFG*n* | 0204_2422h | Configure start bit, stop bit, enable on pin positive edge and disable on compare. Enable resynchronization to received data with TIMOUT = 2h and TIMRST = 4h. |
| TIMCTL*n* | 0000_0081h | Configure the dual 8-bit counter using the inverted pin 0 input. |
| SHIFTBUF*n* | Data to receive | You can read received data from SHIFTBUFBYS[7:0]. Use the shifter status flag to indicate when data can be read using interrupt or DMA request. Read from SHIFTBUFBIS[7:0] instead to support MSB first transfer. |

The UART receiver with RTS configuration uses a second timer to generate the RTS output. RTS asserts when the start bit is detected and negates when the data is read from the shifter buffer register. If no start bit is detected when the RTS is asserted, the received data is ignored.

Table 403. UART receiver with RTS configuration

| Register | Value | Configuration |
|---|---|---|
| SHIFTCFG*n* | 0000_0032h | Configure start bit of 0 and stop bit of 1. |
| SHIFTCTL*n* | 0080_0001h | Configure receive using timer 0 on the negative edge of clock with input data on pin 0. Invert input data by writing 1 to the PINPOL field. |
| TIMCMP*n* | 0000_0F01h | Configure 8-bit transfer with baud rate of divide by 4 of the FLEXIO clock. Set TIMCMP[15:8] as (number of bits × 2) - 1, and set TIMCMP[7:0] as (baud rate divider ÷ 2) - 1. |
| TIMCFG*n* | 0204_2522h | Configure start bit, stop bit, enable on pin positive edge with trigger asserted and disable on compare. Enable resynchronization to received data with TIMOUT = 2h and TIMRST = 4h. |
| TIMCTL*n* | 02C0_0081h | Configure dual 8-bit counter using the inverted pin 0 input. Trigger is internal using the inverted pin 1 input. |
| TIMCMP(*n* + 1) | 0000_FFFFh | Never compare. |

*Table continues on the next page...*

Table 403. UART receiver with RTS configuration (continued)

| Register | Value | Configuration |
|---|---|---|
| TIMCFG(*n* + 1) | 0030_6100h | Enable on timer *n* enable and disable on the trigger falling edge. Decrement on trigger to ensure no compare. |
| TIMCTL(*n* + 1) | 0143_0003h | Configure 16-bit counter and output on pin 1. Trigger is internal using the shifter 0 flag. |
| SHIFTBUF*n* | Data to receive | You can read received data using SHIFTBUFBYS[7:0]. Use the shifter status flag to indicate when data can be read using interrupt or DMA request. Read from SHIFTBUFBIS[7:0] instead to support MSB first transfer. |

### 69.6.3 SPI controller

SPI Controller mode can be supported using two timers, two shifters, and four pins. Using the DMA controller, either CPHA = 0 or CPHA = 1 and transfers can be supported. For CPHA = 1, the chip select can remain asserted for multiple transfers and the timer status flag can be used to indicate the end of the transfer.

The stop bit is used to guarantee a minimum of one clock cycle between the target chip select negating and before the next transfer. To initiate each transfer, either the core or DMA writes to the transmit buffer.

**NOTE**

Because of synchronization delays, the setup time for the serial input data is 1.5 FLEXIO clock cycles. This means the maximum baud rate is divide by 4 of the FLEXIO clock frequency.

Table 404. SPI controller (CPHA = 0) configuration

| Register | Value | Configuration |
|---|---|---|
| SHIFTCFG*n* | 0000_0000h | Start and stop bit disabled. |
| SHIFTCTL*n* | 0083_0002h | Configure transmit using timer 0 on the negative edge of clock with output data on pin 0. |
| SHIFTCFG(*n* + 1) | 0000_0000h | Start and stop bit disabled. |
| SHIFTCTL(*n* + 1) | 0000_0101h | Configure receive using timer 0 on the positive edge of clock with input data on pin 1. |
| TIMCMP*n* | 0000_3F01h | Configure 32-bit transfer with baud rate of divide by 4 of the FLEXIO clock. Set TIMCMP[15:8] as (number of bits × 2) - 1, and set TIMCMP[7:0] as (baud rate divider ÷ 2) - 1. |
| TIMCFG*n* | 0100_2222h | Configure start bit, stop bit, enable on trigger high and disable on compare; initial clock state is logic 0. |

*Table continues on the next page...*

Table 404. SPI controller (CPHA = 0) configuration (continued)

| Register | Value | Configuration |
|---|---|---|
| TIMCTL*n* | 01C3_0201h | Configure dual 8-bit counter using the pin 2 output (shift clock), with shifter 0 flag as the inverted trigger. Write 1 to the PINPOL field to invert the output shift clock. |
| TIMCMP(*n* + 1) | 0000_FFFFh | Never compare. |
| TIMCFG(*n* + 1) | 0000_1100h | Enable when timer 0 is enabled and disable when timer 0 is disabled. |
| TIMCTL(*n* + 1) | 0003_0383h | Configure 16-bit counter (never compare) using the inverted pin 3 output as target select. |
| SHIFTBUF*n* | Data to transmit | You can write transmit data to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7). Use the shifter status flag to indicate when data can be written using interrupt or DMA request. Write to Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7) instead to support MSB first transfer. |
| SHIFTBUF(*n* + 1) | Data to receive | Received data can be read from Shifter Buffer (SHIFTBUF0 - SHIFTBUF7). Use the shifter status flag to indicate when data can be read using interrupt or DMA request. Read from Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7) instead to support MSB first transfer. |

Table 405. SPI controller (CPHA = 1) configuration

| Register | Value | Configuration |
|---|---|---|
| SHIFTCFG*n* | 0000_0021h | Start bit loads data on first shift. |
| SHIFTCTL*n* | 0003_0002h | Configure transmit using timer 0 on the positive edge of clock with output data on pin 0. |
| SHIFTCFG(*n* + 1) | 0000_0000h | Start and stop bit disabled. |
| SHIFTCTL(*n* + 1) | 0080_0101h | Configure receive using timer 0 on the negative edge of clock with input data on pin 1. |
| TIMCMP*n* | 0000_3F01h | Configure 32-bit transfer with baud rate of divide by 4 of the FLEXIO clock. Set TIMCMP[15:8] as (number of bits x 2) - 1, and set TIMCMP[7:0] as (baud rate divider ÷ 2) - 1. |

*Table continues on the next page...*

Table 405. SPI controller (CPHA = 1) configuration (continued)

| Register | Value | Configuration |
|---|---|---|
| TIMCFG*n* | 0100_2222h | Configure start bit, stop bit, enable on trigger high and disable on compare; initial clock state is logic 0. |
| TIMCTL*n* | 01C3_0201h | Configure dual 8-bit counter using pin 2 output (shift clock), with the shifter 0 flag as the inverted trigger. Write 1 to the PINPOL field to invert the output shift clock, and set the TIMDIS field as 3 to keep target select asserted for as long as there is data in the transmit buffer. |
| TIMCMP(*n* + 1) | 0000_FFFFh | Never compare. |
| TIMCFG(*n* + 1) | 0000_1100h | Enable when timer 0 is enabled and disable when timer 0 is disabled. |
| TIMCTL(*n* + 1) | 0003_0383h | Configure 16-bit counter (never compare) using inverted pin 3 output (as target select). |
| SHIFTBUF*n* | Data to transmit | Transmit data can be written to SHIFTBUF. Use the shifter status flag to indicate when data can be written using interrupt or DMA request. Write to Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7) instead to support MSB first transfer. |
| SHIFTBUF(*n* + 1) | Data to receive | Received data can be read from Shifter Buffer (SHIFTBUF0 - SHIFTBUF7). Use the shifter status flag to indicate when data can be read using interrupt or DMA request. Read from Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7) instead to support MSB first transfer. |

### 69.6.4 SPI target

SPI Target mode can be supported using one timer, two shifters, and four pins. Either CPHA = 0 or CPHA = 1 can be supported and transfers can be supported using the DMA controller. For CPHA = 1, the select can remain asserted for multiple transfers and the timer status flag can be used to indicate the end of the transfer.

You must write the transmit data to the transmit buffer register before the external target select asserts; otherwise, the shifter error flag is set.

**NOTE**

Because of synchronization delays, the output valid time for the serial output data is 2.5 FLEXIO clock cycles. This means the maximum baud rate is divide by 6 of the FLEXIO clock frequency.

**Table 406. SPI target (CPHA = 0) configuration**

| Register | Value | Configuration |
|---|---|---|
| SHIFTCFG*n* | 0000_0000h | Start and stop bit disabled. |
| SHIFTCTL*n* | 0083_0002h | Configure transmit using timer 0 on the falling edge of shift clock with output data on pin 0. |
| SHIFTCFG(*n* + 1) | 0000_0000h | Start and stop bit disabled. |
| SHIFTCTL(*n* + 1) | 0000_0101h | Configure receive using timer 0 on the rising edge of shift clock with input data on pin 1. |
| TIMCMP*n* | 0000_003Fh | Configure 32-bit transfer. Set TIMCMP[15:0] as (number of bits × 2) - 1. |
| TIMCFG*n* | 0120_0600h | Configure enable on trigger rising edge. Initial clock state is logic 0 and decrements on pin input. |
| TIMCTL*n* | 06C0_0203h | Configure 16-bit counter using pin 2 input (shift clock), with pin 3 input (target select) as the inverted trigger. |
| SHIFTBUF*n* | Data to transmit | Transmit data can be written to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7). Use the shifter status flag to indicate when data can be written using interrupt or DMA request. Write to Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7) instead to support MSB first transfer. |
| SHIFTBUF(*n* + 1) | Data to receive | Received data can be read from Shifter Buffer (SHIFTBUF0 - SHIFTBUF7). Use the shifter status flag to indicate when data can be read using interrupt or DMA request. Read from Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7) instead to support MSB first transfer. |

**Table 407. SPI target (CPHA = 1) configuration**

| Register | Value | Configuration |
|---|---|---|
| SHIFTCFG*n* | 0000_0001h | Shifter configured to load on first shift and stop bit disabled. |
| SHIFTCTL*n* | 0003_0002h | Configure transmit using timer 0 on rising edge of shift clock with output data on pin 0. |
| SHIFTCFG(*n* + 1) | 0000_0000h | Start and stop bit disabled. |

*Table continues on the next page...*

Table 407. SPI target (CPHA = 1) configuration (continued)

| Register | Value | Configuration |
|---|---|---|
| SHIFTCTL(*n* + 1) | 0080_0101h | Configure receive using timer 0 on falling edge of shift clock with input data on pin 1. |
| TIMCMP*n* | 0000_003Fh | Configure 32-bit transfer. Set TIMCMP[15:0] as (number of bits × 2) - 1). |
| TIMCFG*n* | 0120_6602h | Configure start bit, enable on trigger rising edge, disable on trigger falling edge. Initial clock state is logic 0 and decrements on pin input. |
| TIMCTL*n* | 06C0_0203h | Configure 16-bit counter using pin 2 input (shift clock), with pin 3 input (target select) as the inverted trigger. |
| SHIFTBUF*n* | Data to transmit | Transmit data can be written to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7). Use the shifter status flag to indicate when data can be written using interrupt or DMA request. Write to Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7) instead to support MSB first transfer. |
| SHIFTBUF(*n* + 1) | Data to receive | Received data can be read from Shifter Buffer (SHIFTBUF0 - SHIFTBUF7). Use the shifter status flag to indicate when data can be read using interrupt or DMA request. Read from Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7) instead to support MSB first transfer. |

### 69.6.5 I2C controller

I2C Controller mode can be supported using two timers, two shifters, and two pins. One timer is used to generate the SCL output and another one is used to control the shifters. The two shifters that are used to transmit and receive for every word, when receiving the transmitter, must transmit FFh to 3-state the output. FLEXIO inserts a stop bit after every word to generate and verify the ACK or NACK. FLEXIO waits for the first write to the transmit data buffer before enabling SCL generation. Data transfers can be supported using the DMA controller and the shifter error flag sets on transmit underrun or receive overflow.

The first timer generates the bit clock for the entire packet (start to repeated start or stop condition), so you must program the compare register with the total number of clock edges in the packet (minus one). The timer supports clock stretching using the reset counter when pin is equal to output. However, this increases both the clock high and clock low periods by at least one FLEXIO clock cycle each. The second timer uses the SCL input pin to control the transmit and receive shift registers. This enforces an SDA data hold time by an extra two FLEXIO clock cycles.

Both the transmit and receive shifters must be serviced for each word in the transfer. The transmit shifter must transmit FFh when receiving, and the receive shifter returns the data present on the SDA pin. The transmit shifter loads one additional word on the last falling edge of the SCL pin. When generating a stop condition or a repeated start condition, this word must be 00h and FFh, respectively. During the last word of a controller-receiver transfer, you must set the transmit SHIFTCFG*n*[SSTOP] field to generate a NACK.

The receive shift register asserts an error interrupt if a NACK is detected, but you are responsible for generating the stop or repeated start condition. If a NACK is detected during controller-transmit, the interrupt routine must immediately write 00h (when generating a stop condition) or FFh (when generating a repeated start condition) to the transmit shifter register. You must wait for the next rising edge on SCL before disabling both timers. The transmit shifter must be disabled after the setup delay for a repeated start or stop condition.

**NOTE**

Because of synchronization delays, the data valid time for the transmit output is two FLEXIO clock cycles. This means the maximum baud rate is divide by 6 of the FLEXIO clock frequency.

To guarantee SDA hold time, the I2C controller data valid is delayed by two cycles because the clock output is passed through a synchronizer before clocking the transmit or receive shifter. Because the SCL output is synchronous with FLEXIO clock, the synchronization delay is one cycle, and then an additional cycle is involved to generate the output.

**Table 408. I2C controller configuration**

| Register | Value | Configuration |
|---|---|---|
| SHIFTCFG*n* | 0000_0032h | Start bit enabled (logic 0) and stop bit enabled (logic 1). |
| SHIFTCTL*n* | 0101_0082h | Configure transmit using timer 1 on the rising edge of clock with inverted output enable (open-drain output) on pin 0. |
| SHIFTCFG(*n* + 1) | 0000_0020h | Start bit disabled and stop bit enabled (logic 0) for ACK or NACK detection. |
| SHIFTCTL(*n* + 1) | 0180_0001h | Configure receive using timer 1 on the falling edge of clock with input data on pin 0. |
| TIMCMP*n* | 0000_2501h | Configure 2 word transfer with baud rate of divide by 4 of the FLEXIO clock. Set TIMCMP[15:8] as (number of words × 18) + 1, and set TIMCMP[7:0] as (baud rate divider ÷ 2) - 1. |
| TIMCFG*n* | 0102_2222h | Configure start bit, stop bit, enable on trigger high, disable on compare, reset if output equals pin. Initial clock state is logic 0 and is not affected by reset. |
| TIMCTL*n* | 01C1_0101h | Configure dual 8-bit counter using pin 1 output enable (SCL open-drain), with the shifter 0 flag as the inverted trigger. |
| TIMCMP(*n* + 1) | 0000_000Fh | Configure 8-bit transfer. Set TIMCMP[15:0] as (number of bits x 2) - 1. |
| TIMCFG(*n* + 1) | 0020_1112h | Enable when timer 0 is enabled; disable when timer 0 is disabled. Enable start bit and stop bit at the end of each word and decrement on pin input. |
| TIMCTL(*n* + 1) | 01C0_0183h | Configure 16-bit counter using inverted pin 1 input (SCL). |

*Table continues on the next page...*

Table 408. I2C controller configuration (continued)

| Register | Value | Configuration |
|---|---|---|
| SHIFTBUF*n* | Data to transmit | Transmit data can be written to SHIFTBUFBBS[7:0]. Use the shifter status flag to indicate when data can be written using interrupt or DMA request. |
| SHIFTBUF(*n* + 1) | Data to receive | Received data can be read from SHIFTBUFBIS[7:0]. Use the shifter status flag to indicate when data can be read using interrupt or DMA request. |

### 69.6.6 I2S controller

I2S Controller mode can be supported using two timers, two shifters, and four pins. One timer is used to generate the bit clock and control the shifters and another timer is used to generate the frame sync. FLEXIO waits for the first write to the transmit data buffer before enabling bit clock and frame sync generation. Data transfers are supported using the DMA controller and the shifter error flag sets on transmit underrun or receive overflow.

The bit clock frequency is an even integer divide of the FLEXIO clock frequency. The initial frame sync assertion occurs at the same time as the first bit clock edge. The timer uses the start bit to ensure that the frame sync is generated one clock cycle before the first output data.

**NOTE**

Because of synchronization delays, the setup time for the receiver input is 1.5 FLEXIO clock cycles. This means that the maximum baud rate is divide by 4 of the FLEXIO clock frequency.

Table 409. I2S controller configuration

| Register | Value | Configuration |
|---|---|---|
| SHIFTCFG*n* | 0000_0001h | Load transmit data on first shift and stop bit disabled. |
| SHIFTCTL*n* | 0003_0002h | Configure transmit using timer 0 on the rising edge of clock with output data on pin 0. |
| SHIFTCFG(*n* + 1) | 0000_0000h | Start and stop bit disabled. |
| SHIFTCTL(*n* + 1) | 0080_0101h | Configure receive using timer 0 on the falling edge of clock with input data on pin 1. |
| TIMCMP*n* | 0000_3F01h | Configure 32-bit transfer with baud rate of divide by 4 of the FLEXIO clock. Set TIMCMP[15:8] as (number of bits × 2) - 1, and set TIMCMP[7:0] as (baud rate divider ÷ 2) - 1. |
| TIMCFG*n* | 0000_0202h | Configure start bit, enable on trigger high and never disable. Initial clock state is logic 1. |
| TIMCTL*n* | 01C3_0281h | Configure dual 8-bit counter using inverted pin 2 output (bit clock), with |

*Table continues on the next page...*

Table 409. I2S controller configuration (continued)

| Register | Value | Configuration |
|---|---|---|
| | | shifter 0 flag as the inverted trigger. Write 0 to the PINPOL field to invert the polarity of the output shift clock. |
| TIMCMP(*n* + 1) | 0000_007Fh | Configure 32-bit transfer with baud rate of divide by 4 of the FLEXIO clock. Set TIMCMP[15:0] as (number of bits × baud rate divider) ÷ 1. |
| TIMCFG(*n* + 1) | 0000_0100h | Enable when timer 0 is enabled and never disable. |
| TIMCTL(*n* + 1) | 0003_0383h | Configure 16-bit counter using inverted pin 3 output (as frame sync). Write 0 to the PINPOL field to invert the polarity of the output frame sync. |
| SHIFTBUF*n* | Data to transmit | Transmit data can be written to Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7). Use the shifter status flag to indicate when data can be written using interrupt or DMA request. Write to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) instead to support LSB first transfer. |
| SHIFTBUF(*n* + 1) | Data to receive | Received data can be read from Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7). Use the shifter status flag to indicate when data can be read using interrupt or DMA request. Read from Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) instead to support LSB first transfer. |

### 69.6.7 I2S target

I2S Target mode can be supported using three timers, two shifters, and four pins. For single transmit and single receive, other combinations of transmit and receive are possible.

The transmit data must be written to the transmit buffer register before the external frame sync asserts, otherwise the shifter error flag is set.

**NOTE**

Because of synchronization delays, the output valid time for the serial output data is 2.5 FLEXIO clock cycles. This means the maximum baud rate is divide by 6 of the FLEXIO clock frequency.

The output valid time of I2S target is maximum 2.5 cycles because there is a maximum 1.5 cycle delay on the clock synchronization, plus one cycle to output the data.

Timer 2 detects the falling edge of frame sync (start of new frame) and asserts output until the rising edge of bit clock (when the frame sync is normally sampled). Timer 0 detects the rising edge of bit clock with timer 2 output asserted and asserts output for length of frame. Timer 1 detects the falling edge of bit clock with timer 0 output asserted and controls shift registers for 32-bit transfers.

Table 410. I2S target configuration

| Register | Value | Configuration |
|---|---|---|
| SHIFTCFG*n* | 0000_0000h | Start and stop bit disabled. |
| SHIFTCTL*n* | 0103_0002h | Configure transmit using timer 1 on the rising edge of shift clock with output data on pin 0. |
| SHIFTCFG(*n* + 1) | 0000_0000h | Start and stop bit disabled. |
| SHIFTCTL(*n* + 1) | 0180_0101h | Configure receive using timer 1 on the falling edge of shift clock with input data on pin 1. |
| TIMCMP*n* | 0000_007Fh | Configure two 32-bit transfers per frame. Set TIMCMP[15:0] as (number of bits × 4) - 1. |
| TIMCFG*n* | 0020_2500h | Configure enable on pin rising edge (inverted bit clock) with trigger high (timer 2) and disable on compare. Initial clock state is logic 1 and decrements on pin input (bit clock). |
| TIMCTL*n* | 0B40_0203h | Configure 16-bit counter using pin 2 input (bit clock), with timer 2 output as the trigger. |
| TIMCMP(*n* + 1) | 0000_003Fh | Configure 32-bit transfers. Set TIMCMP[15:0] as (number of bits × 2) - 1. |
| TIMCFG(*n* + 1) | 0020_2500h | Configure enable on pin (bit clock) rising edge with trigger (timer 0) high and disable on compare. Initial clock state is logic 1 and decrement on pin input (bit clock). |
| TIMCTL(*n* + 1) | 0340_0283h | Configure 16-bit counter using inverted pin 2 input (bit clock), with timer 0 output as the trigger. |
| TIMCMP(*n* + 2) | 0000_0000h | Compare on zero (first edge). |
| TIMCFG(*n* + 2) | 0020_6400h | Configure enable on inverted pin (frame sync) rising edge and disable on trigger falling edge (bit clock). Initial clock state is logic 1 and decrement on inverted pin input (frame sync). |
| TIMCTL(*n* + 2) | 04C0_0383h | Configure 16-bit counter using inverted pin 3 input (frame sync), with pin 2 inverted input (bit clock) as the trigger. |
| SHIFTBUF*n* | Data to transmit | Transmit data can be written to Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7). Use the shifter status flag to indicate when data can be written |

*Table continues on the next page...*

Table 410. I2S target configuration (continued)

| Register | Value | Configuration |
|---|---|---|
| | | using interrupt or DMA request. Write to the SHIFTBUF register instead to support LSB first transfer. |
| SHIFTBUF(*n* + 1) | Data to receive | Received data can be read from Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7). Use the shifter status flag to indicate when data can be read using interrupt or DMA request. Read from the SHIFTBUF register instead to support LSB first transfer. |

### 69.6.8 SENT receiver

The SENT receiver can be supported by using one timer and one pin. The timer is configured as Input-Capture mode, and captures the counter value at the falling edge of the pin input. After the counter value is captured, the counter is automatically restarted from counter value 0. Therefore, the captured value always indicates the period between the previous falling edge and current falling edge. You can configure the CPU interrupt or DMA trigger at each capture of the counter. The CPU software performs the entire SENT frame decoding with the latest tick width adjustment.

Table 411. SENT receiver configuration

| Register | Value | Configuration |
|---|---|---|
| TIMCMP*n* | — | Stores counter value at the falling edge of the pin. |
| TIMCFG*n* | 0000_6000h | Timer is always enabled. It is disabled on trigger falling edge, decrement counter on FLEXIO clock. |
| TIMCTL*n* | 0040_0007h | Single 16-bit input capture mode. The PINSEL field selects the timer pin input and output. Timer pin output disabled, internal trigger selected, select pin 0 as trigger. |

### 69.6.9 Camera interface

Camera interface can be supported using one timer, one or more shifters, and multiple pins. Multiple transfers can be supported using the DMA controller.

The example configuration shown in the following table describes the FLEXIO configuration for interfacing with an 8-bit CMOS sensor with PCLK, VSYNC, HREF, and D[7:0] outputs. The example uses a 128-bit buffer to capture 16 pixels of image data before an interrupt or DMA transfer. You can use a bigger or smaller buffer depending on system DMA performance and FLEXIO resource usage by other applications.

**NOTE**

You can use additional timers to track the number of pixels per row and number of rows per frame, or HREF or VSYNC can be assigned as GPIO interrupts for software tracking.

Table 412. Camera interface configuration for 8-bit CMOS sensor

| Register | Value | Configuration |
|---|---|---|
| SHIFTCFG*n*...*n*+2[1] | 0007_0100h | Configure 8-bit parallel shift in from adjacent shifter. |
| SHIFTCFG*n*+3 | 0007_0000h | Configure 8-bit parallel shift in from pins FXIO_D[7:0] (D[7:0]). |
| SHIFTCTL*n*...*n*+3 | 0080_0001h | Configure receive using timer 0 on the negative edge of clock. |
| TIMCMP*n* | 0000_001Fh | Configure 16 pixel (8 bits/pixel × 16 pixels = 128 bits) transfer. Set TIMCMP[15:0] as (number of pixels × 2) - 1. |
| TIMCFG*n* | 0120_6600h | Configure enable on trigger (HREF) rising edge and disable on trigger falling edge. Initial shift clock state is logic 0 and decrement on PCLK input. |
| TIMCTL*n* | 12C0_0803h | Configure 16-bit counter using FXIO_D[8] input (PCLK), with FXIO_D[9] input (HREF) as the inverted trigger. |
| SHIFTBUF*n*...*n*+3 | Data to receive | Received data can be read from SHIFTBUF*n*...*n*+3. Use the shifter status flag to indicate when data can be read using interrupt or DMA request. |

1. *n* = 0 or 4

## 69.6.10 Motorola 68K and Intel 8080 bus interface

Motorola 68K and Intel 8080 bus are two parallel interfaces commonly used by smart and asynchronous LCD controllers. With GPIO, FLEXIO can drive these interfaces using one timer and one shifter. Additional shifters can be used to support large transfers via the DMA controller.

Example configurations are provided for 1-beat and 16-beat transfers. Other lengths are also supported.

The following table provides an example of how to drive a 16-bit 68K or 8080 bus. For an 8080 bus, two GPIOs are used to drive the nCS and RS pins. For a 68K bus, an additional GPIO is required to drive the RDWR pin.

Table 413. Motorola 68K and Intel 8080 write configuration

| Register | Value | Configuration |
|---|---|---|
| SHIFTCFG0...7 | 000F_0100h | Configure 16-bit parallel shift in from adjacent shifter. |
| SHIFTCTL0 | 0003_0002h | Configure transmit using timer 0 on the positive edge of clock with data output to FXIO_D[15:0]. |
| SHIFTCTL1...7 | 0000_0002h | Configure transmit using timer 0 on the positive edge of clock. |

*Table continues on the next page...*

Table 413. Motorola 68K and Intel 8080 write configuration (continued)

| Register | Value | Configuration |
|---|---|---|
| TIMCMP0 | 0000_0101h (1 beat)<br>0000_1F01h (16 beats) | Configure 1 or 16-beat transfer with baud rate of divide by 4 of the FLEXIO clock. Set TIMCMP[15:8] as (number of beats × 2) - 1, and set TIMCMP[7:0] as (baud rate divider ÷ 2) - 1. |
| TIMCFG0 | 0000_2200h | Configure enable on trigger high and disable on compare. Timer output high on enable. |
| TIMCTL0 | 01C3_1001h (Motorola 68K, 1 beat)<br>1DC3_1001h (Motorola 68K, 16 beats)<br>01C3_1081h (Intel 8080, 1 beat)<br>1DC3_1081h (Intel 8080, 16 beats) | Configure dual 8-bit counter using FXIO_D[16] output (EN pin for 68K, nWR pin for 8080), with shifter 0 (1-beat) or shifter 7 (16 beats) flag as the inverted trigger. |
| SHIFTBUF0...7 | Data to transmit | Transmit data can be written to SHIFTBUF0 (1 beat) or SHIFTBUF0...7 (16-beats) to initiate a transfer; use the shifter status flag to indicate when data can be written using interrupt or DMA request. |

Table 414. Motorola 68K and Intel 8080 read configuration

| Register | Value | Configuration |
|---|---|---|
| SHIFTCFG0...6 | 000F_0100h | Configure 16-bit parallel shift in from the adjacent shifter. |
| SHIFTCFG7 | 000F_0000h | Configure 16-bit parallel shift in from pin. |
| SHIFTCTL0...7 | 0080_0001h | Configure receive using timer 0 on the negative edge of clock with data input from FXIO_D[15:0]. |
| TIMCMP0 | 0000_0101h (1 beat)<br>0000_1F01h (16 beats) | Configure 1 or 16-beat transfer with baud rate of divide by 4 of the FLEXIO clock. Set TIMCMP[15:8] = (number of beats × 2) - 1, and set TIMCMP[7:0] = (baud rate divider ÷ 2) - 1. |
| TIMCFG0 | 0000_2220h | Configure stop_bit. Enable on trigger high and disable on compare. Timer output high on enable. |
| TIMCTL0 | 1DC3_1001h (Motorola 68K, 1 beat)<br>01C3_1001h (Motorola 68K, 16 beats)<br>1DC3_1181h (Intel 8080, 1 beat)<br>01C3_1181h (Intel 8080, 16 beats) | Configure dual 8-bit counter using either FXIO_D[16] output (EN pin for 68K) or FXIO_D[17] output (nRD pin for 8080), with shifter 7 flag (1-beat) or shifter 0 flag (16 beats) as the inverted trigger. |

*Table continues on the next page...*

Table 414. Motorola 68K and Intel 8080 read configuration (continued)

| Register | Value | Configuration |
|---|---|---|
| SHIFTBUF0...7 | Data received | Received data can be read from SHIFTBUF0 (1 beat) or SHIFTBUF0...7 (16 beats). Use the shifter status flag to indicate when data can be read using interrupt or DMA request. |

In general, any operation to a 68K or 8080 bus target begins with a register write cycle followed by one or more data read or write cycles. To accomplish this, perform the following procedure:

1. Configure FLEXIO with 1-beat write configuration.
2. Configure GPIO to assert the nCS and RS pins (and deassert the RDWR pin for 68K).
3. Write register index data to SHIFTBUF0[15:0].
4. Configure GPIO to deassert the RS pin (and assert the RDWR pin for 68K data read).
5. Configure FLEXIO with the desired read or write configuration (1 or 16 beats).
6. Use the shifter status flag to trigger interrupt or DMA driven data transfers to and from Shifter Buffer (SHIFTBUF0 - SHIFTBUF7).
7. Configure GPIO to deassert the nCS pin.

### 69.6.11 Low-power state machine

Table 415 shows an example of a hypothetical state machine to illustrate the flexibility allowed in Shifter State mode.

In this example, FLEXIO waits for the FXIO_D[2] pin to assert and then drives a complementary square wave output at a frequency of FLEXIO_CLK/131072 on the FXIO_D[1:0] pins while the comparator output is asserted. This assumes that the comparator is connected to external trigger 15. See the chip-specific FLEXIO information for actual FLEXIO trigger mappings. Figure 369 shows the states and transitions implemented by this example.

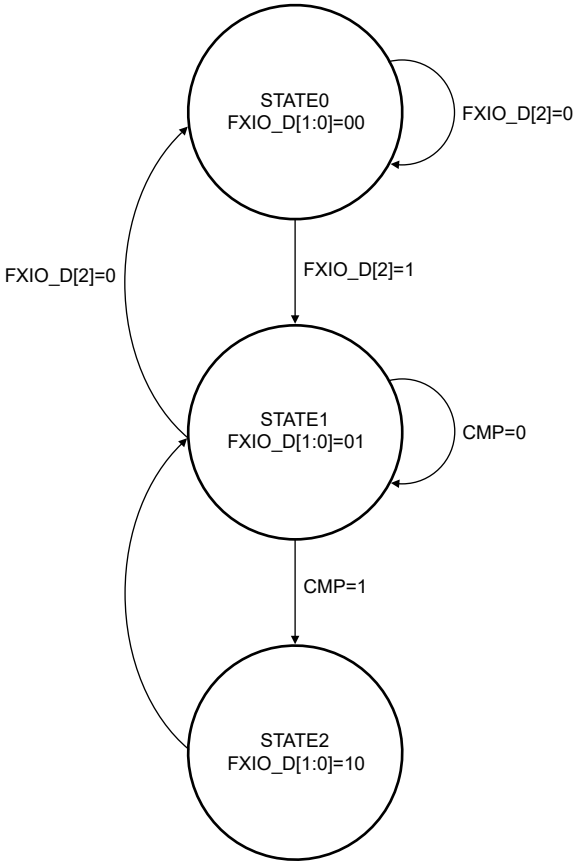



Figure 369. State diagram

Table 415. State machine configuration

| Register | Value | Configuration |
|---|---|---|
| SHIFTCFG0...2 | 0000_0003h | Enable FXIO_D[1:0] as outputs. |
| SHIFTCTL0 | 0080_0206h | Configure for State mode using FXIO_D[4:2] as inputs to select next state and timer 0 output low to trigger next state. |
| SHIFTBUF0 | 0020_8208h | State0: Drive FXIO_D[1:0] = 00; transition to state0 if FXIO_D[2] = 0, state1 if FXIO_D[2] = 1. |
| SHIFTCTL1 | 0000_0206h | Configure for State mode using FXIO_D[4:2] as inputs to select next state and timer 0 output high to trigger next state. |
| SHIFTBUF1 | 0140_8408h | State1: Drive FXIO_D[1:0]=01; transition to state0 if FXIO_D[2]=0, state1 if CMP = 0, state2 if CMP = 1 (FXIO_D[3]=1). |

*Table continues on the next page...*

Table 415. State machine configuration (continued)

| Register | Value | Configuration |
|---|---|---|
| SHIFTCTL2 | 0080_0206h | Configure for State mode using FXIO_D[4:2] as inputs to select next state and timer 0 output low to trigger next state. |
| SHIFTBUF2 | 0224_9249h | State2: Drive FXIO_D[1:0] = 10, transition to state1 with timer 0 output low. |
| TIMCMP0 | 0000_FFFFh | Configure baud rate of divide by 131072 of the FLEXIO clock. Set TIMCMP[7:0] as (baud rate divider ÷ 2) - 1. |
| TIMCFG0 | 0000_0000h | Configure timer always enabled. |
| TIMCTL0 | 0000_0003h | Configure single 16-bit counter. |
| TIMCFG1 | 0010_7600h | Configure timer enabled on trigger rising edge, disabled on trigger falling edge, decrement on trigger edge. |
| TIMCTL1 | 0F03_0303h | Configure timer output enabled on FXIO_D[3], external trigger 15 (comparator output) selected. |

### 69.6.12 Keypad interface

The keypad interface can support a 3 × 4 keypad matrix using three timers and three shifters, although a larger matrix can be supported using additional shifters. The configuration is designed for four columns configured as active low open-drain pins and three rows configured as input pins with pull-up resistors enabled.

One shifter is configured in Logic mode to assert its output when any of the row inputs are low, indicating a key is pressed. Use a timer to filter the shifter output to ensure that the key is pressed for a minimum amount of time before performing the column scan.

A different shifter is configured for parallel transmit. Use this shifter to scan each column when a keypress is detected. When not scanning, the shifter output is configured to assert all active low open-drain column outputs to detect any keypress. Use a dedicated timer to control the transmit shifter.

The last shifter is configured for parallel receive. Use this shifter to capture the result of the column scanning so that you can decode which key (or keys) was pressed. This configuration captures the state of both row and column pins for each scan, although the row state can also be deduced by the shift order. Use a dedicated timer to control the receive shifter, which shifts at half the frequency of the transmit shifter.

When the result of the key scan is available in the receive shifter register, FLEXIO continues to monitor the row inputs and can trigger multiple scans from a single keypress. To support debouncing, you can decide how many consecutive scans must be considered as a single keypress.

**NOTE**

Because the pins used in Logic mode are fixed per shifter and the shifters that support parallel shifts are limited, this configuration is restricted to what pins and shifters it can use. Increasing the matrix beyond four-row inputs requires multiple shifters in Logic mode. Increasing beyond four-column outputs requires concatenating transmit shifters to create a larger shift register.

Table 416. Keypad interface configuration

| Register | Value | Configuration |
|---|---|---|
| SHIFTCFG0 | 0003_0032h | Start bit enabled (logic 0) and stop bit enabled (logic 1), configured for 4-bit shift. |
| SHIFTCTL0 | 0101_0402h | Configure transmit using timer 1 on the rising edge of clock generating open-drain output on pins[7:4] (column outputs). |
| SHIFTBUF0 | 0804_0201h | Static data containing the column scan pattern; each column is scanned one-hot with dead time in between. |
| SHIFTCFG1 | 0000_0020h | In Logic mode, mask input 3. |
| SHIFTCTL1 | 0000_0007h | Configure Logic mode. |
| SHIFTBUF1 | 07FF_07FFh | Static data configuring Logic mode LUT. Output asserts if pins [3:1] are logic 0. |
| SHIFTCFG3 | 0007_0000h | Configured for 8-bit shift. |
| SHIFTCTL3 | 0280_0001h | Configure receive using timer 2 on falling edge of clock with input data on pins [7:0] (both rows and columns; pin 0 is don't care). |
| TIMCMP0 | 0000_00ffh | Configure prescanning glitch filter to 256 FLEXIO clock cycles. For different filter cycles, configure TIMCMP[15:0] as (filter cycles) - 1. |
| TIMCFG0 | 0103_6600h | Configure enable on trigger rising edge and disable on trigger falling edge. Initial clock state is logic 0 and is not affected by reset. |
| TIMCTL0 | 0540_0003h | Configure 16-bit counter using the shifter 1 flag (logic state) as trigger. |
| TIMCMP1 | 0000_0F3Fh | Configure eight shifts (twice for each column) at column scan rate of divide by 128. For a different scan frequency, define TIMCMP[7:0] as (scan divider ÷ 2) - 1. |
| TIMCFG1 | 0020_2622h | Enable on trigger rising edge, disable on compare. Start and stop bits are enabled. |
| TIMCTL1 | 0340_0001h | Configure dual 8-bit counter using timer 0 output as the trigger. |
| TIMCMP2 | 0000_0801h | Configure four shifts at half the frequency of the timer 1 trigger. |

*Table continues on the next page...*

Table 416. Keypad interface configuration (continued)

| Register | Value | Configuration |
|---|---|---|
| TIMCFG2 | 0110_2100h | Enable on timer 1 enable, disable on compare, decrement on trigger input with output initially negated, and not affected by reset. |
| TIMCTL2 | 0740_0001h | Configure dual 8-bit counter using timer 1 output as the trigger. |
| SHIFTBUF3 | Keypad scan result | Keypad scan result can be read from SHIFTBUF3. Each byte is the result of one scan with both row and column pin state from the scan (pin 0 is not used). Use the shifter status flag to indicate when data can be written using interrupt or DMA request. |

## 69.7 Memory map and registers

### 69.7.1 FLEXIO register descriptions

**NOTE**

Invalid register accesses, which include reading a write-only register, writing to a read-only register, or accessing an invalid address, result in a bus error.

#### 69.7.1.1 FLEXIO memory map

FlexIO base address: 4032_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID (VERID) | 32 | R | 0201_0003h |
| 4h | Parameter (PARAM) | 32 | R | 0420_0808h |
| 8h | FLEXIO Control (CTRL) | 32 | RW | 0000_0000h |
| Ch | Pin State (PIN) | 32 | R | 0000_0000h |
| 10h | Shifter Status (SHIFTSTAT) | 32 | RW | 0000_0000h |
| 14h | Shifter Error (SHIFTERR) | 32 | RW | 0000_0000h |
| 18h | Timer Status Flag (TIMSTAT) | 32 | RW | 0000_0000h |
| 20h | Shifter Status Interrupt Enable (SHIFTSIEN) | 32 | RW | 0000_0000h |
| 24h | Shifter Error Interrupt Enable (SHIFTEIEN) | 32 | RW | 0000_0000h |
| 28h | Timer Interrupt Enable (TIMIEN) | 32 | RW | 0000_0000h |
| 30h | Shifter Status DMA Enable (SHIFTSDEN) | 32 | RW | 0000_0000h |
| 38h | Timer Status DMA Enable (TIMERSDEN) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 40h | Shifter State (SHIFTSTATE) | 32 | RW | 0000_0000h |
| 48h | Trigger Status (TRGSTAT) | 32 | RW | 0000_0000h |
| 4Ch | External Trigger Interrupt Enable (TRIGIEN) | 32 | RW | 0000_0000h |
| 50h | Pin Status (PINSTAT) | 32 | RW | 0000_0000h |
| 54h | Pin Interrupt Enable (PINIEN) | 32 | RW | 0000_0000h |
| 58h | Pin Rising Edge Enable (PINREN) | 32 | RW | 0000_0000h |
| 5Ch | Pin Falling Edge Enable (PINFEN) | 32 | RW | 0000_0000h |
| 60h | Pin Output Data (PINOUTD) | 32 | RW | 0000_0000h |
| 64h | Pin Output Enable (PINOUTE) | 32 | RW | 0000_0000h |
| 68h | Pin Output Disable (PINOUTDIS) | 32 | RW | 0000_0000h |
| 6Ch | Pin Output Clear (PINOUTCLR) | 32 | RW | 0000_0000h |
| 70h | Pin Output Set (PINOUTSET) | 32 | RW | 0000_0000h |
| 74h | Pin Output Toggle (PINOUTTOG) | 32 | RW | 0000_0000h |
| 80h - 9Ch | Shifter Control (SHIFTCTL0 - SHIFTCTL7) | 32 | RW | 0000_0000h |
| 100h - 11Ch | Shifter Configuration (SHIFTCFG0 - SHIFTCFG7) | 32 | RW | 0000_0000h |
| 200h - 21Ch | Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) | 32 | RW | 0000_0000h |
| 280h - 29Ch | Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7) | 32 | RW | 0000_0000h |
| 300h - 31Ch | Shifter Buffer Byte Swapped (SHIFTBUFBYS0 - SHIFTBUFBYS7) | 32 | RW | 0000_0000h |
| 380h - 39Ch | Shifter Buffer Bit Byte Swapped (SHIFTBUFBBS0 - SHIFTBUFBBS7) | 32 | RW | 0000_0000h |
| 400h - 41Ch | Timer Control (TIMCTL0 - TIMCTL7) | 32 | RW | 0000_0000h |
| 480h - 49Ch | Timer Configuration (TIMCFG0 - TIMCFG7) | 32 | RW | 0000_0000h |
| 500h - 51Ch | Timer Compare (TIMCMP0 - TIMCMP7) | 32 | RW | 0000_0000h |
| 680h - 69Ch | Shifter Buffer Nibble Byte Swapped (SHIFTBUFNBS0 - SHIFTBUFNBS7) | 32 | RW | 0000_0000h |
| 700h - 71Ch | Shifter Buffer Halfword Swapped (SHIFTBUFHWS0 - SHIFTBUFHWS7) | 32 | RW | 0000_0000h |
| 780h - 79Ch | Shifter Buffer Nibble Swapped (SHIFTBUFNIS0 - SHIFTBUFNIS7) | 32 | RW | 0000_0000h |
| 800h - 81Ch | Shifter Buffer Odd Even Swapped (SHIFTBUFOES0 - SHIFTBUFOES7) | 32 | RW | 0000_0000h |
| 880h - 89Ch | Shifter Buffer Even Odd Swapped (SHIFTBUFEOS0 - SHIFTBUFEOS7) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 900h - 91Ch | Shifter Buffer Halfword Byte Swapped (SHIFTBUFHBS0 - SHIFTBUFHBS7) | 32 | RW | 0000_0000h |

### 69.7.1.2 Version ID (VERID)

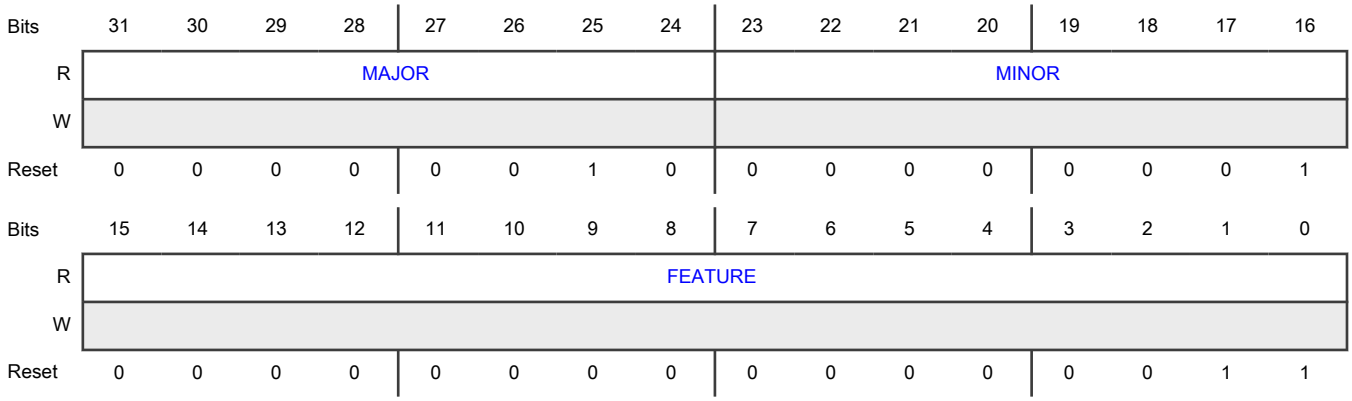
#### Offset

| Register | Offset |
|---|---|
| VERID | 0h |

#### Function

Indicates the version of FLEXIO.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>Indicates the major version number of the module specification. |
| 23-16<br>MINOR | Minor Version Number<br>Indicates the minor version number of the module specification. |
| 15-0<br>FEATURE | Feature Specification Number<br>Indicates the feature set number.<br>0000_0000_0000_0000b - Standard features implemented |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0000_0000_0000_0001b - State, logic, and parallel modes supported<br>0000_0000_0000_0010b - Pin control registers supported<br>0000_0000_0000_0011b - State, logic, and parallel modes, plus pin control registers supported |

### 69.7.1.3 Parameter (PARAM)

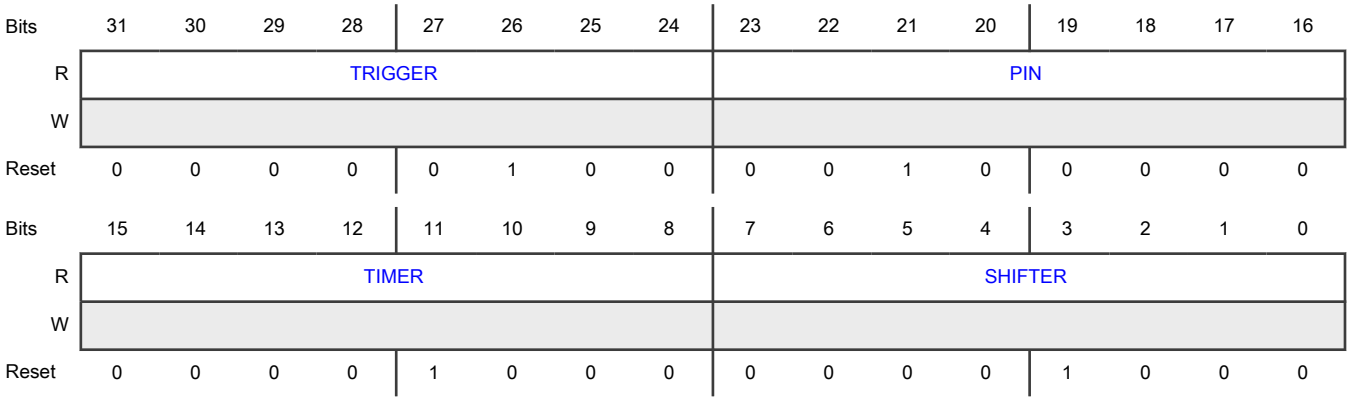
#### Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

#### Function

Contains the number of shifters, timers, pins, and triggers.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-24<br>TRIGGER | Trigger Number<br>Indicates the number of external triggers implemented. |
| 23-16<br>PIN | Pin Number<br>Indicates the number of pins implemented. |
| 15-8<br>TIMER | Timer Number<br>Indicates the number of timers implemented. |
| 7-0<br>SHIFTER | Shifter Number<br>Indicates the number of shifters implemented. |

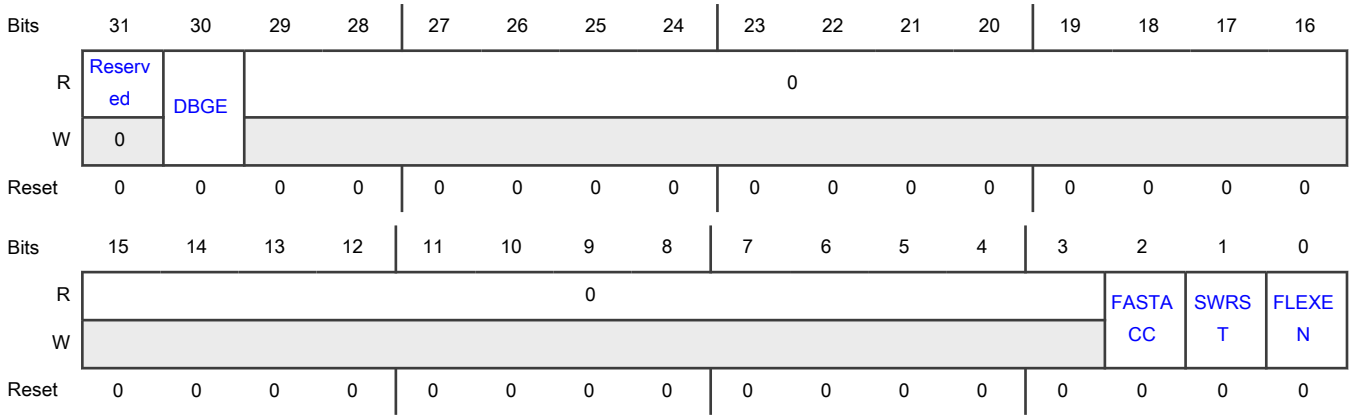
### 69.7.1.4 FLEXIO Control (CTRL)

#### Offset

| Register | Offset |
|---|---|
| CTRL | 8h |

#### Function
Controls various aspects of the FLEXIO operation.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31<br>— | Any write value to this bit must be zero. |
| 30<br>DBGE | Debug Enable<br>Enables the FLEXIO operation in Debug mode.<br>0b - Disable<br>1b - Enable |
| 29-3<br>— | Reserved |
| 2<br>FASTACC | Fast Access<br>Configures fast or normal register accesses to FLEXIO registers, but requires the FLEXIO functional clock to be at least equal to the frequency of the bus clock.<br>0b - Normal<br>1b - Fast |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 1<br>SWRST | Software Reset<br>Specifies whether software reset is enabled. The software reset does not affect this register but it affects all other logic in FLEXIO. All other register accesses are ignored until this field is cleared. The field remains 1 until software clears it and the reset has cleared in the FLEXIO clock domain. If you write 1 to this field, all FLEXIO registers except the Control register are reset.<br>0b - Disabled<br>1b - Enabled |
| 0<br>FLEXEN | FLEXIO Enable<br>Enables FLEXIO.<br>0b - Disable<br>1b - Enable |

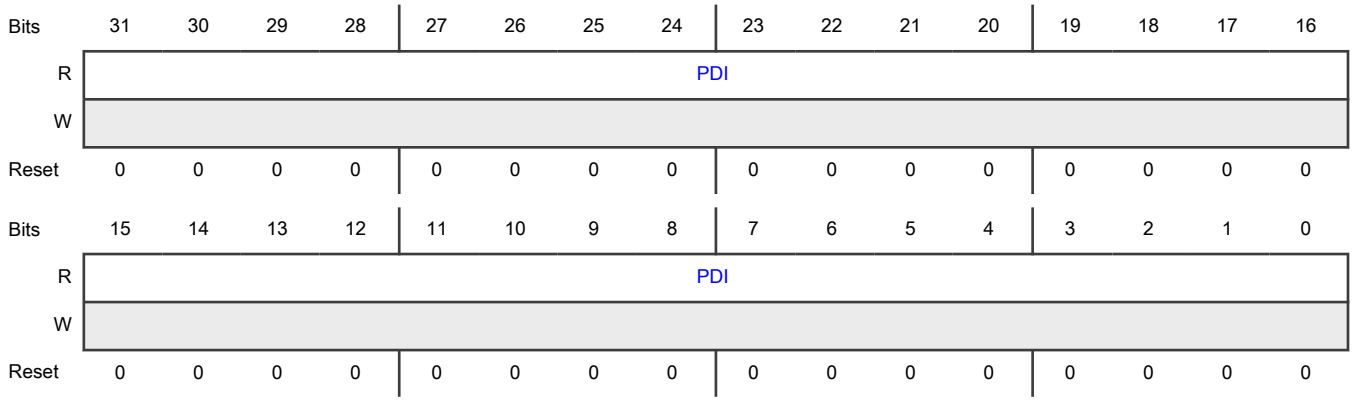
### 69.7.1.5 Pin State (PIN)

#### Offset

| Register | Offset |
|---|---|
| PIN | Ch |

#### Function
Indicates the status of the pin data input.

#### Diagram



**Fields**

| Field | Function |
|---|---|
| 31-0<br>PDI | Pin Data Input<br>Indicates the input data on each of the FLEXIO pins. |

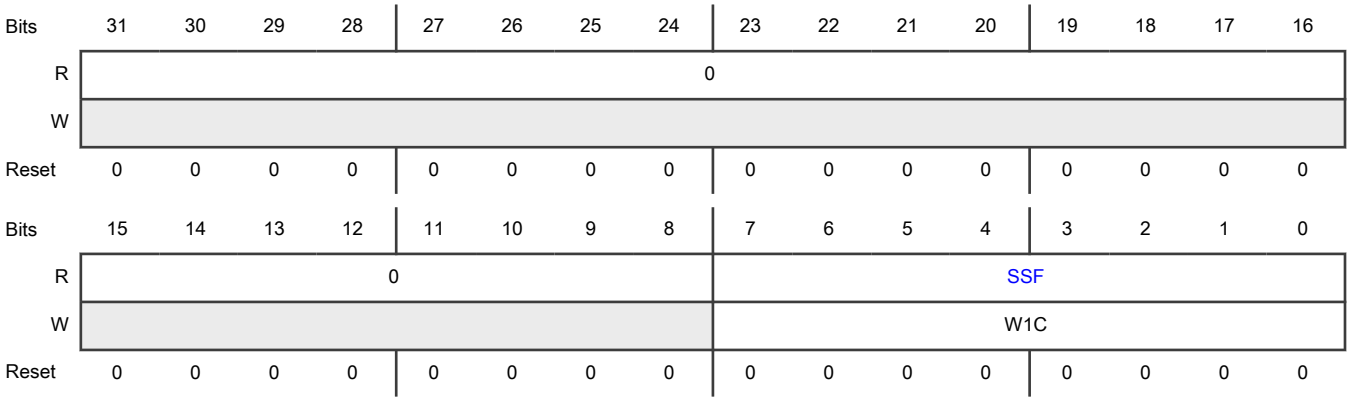
### 69.7.1.6 Shifter Status (SHIFTSTAT)

**Offset**

| Register | Offset |
|---|---|
| SHIFTSTAT | 10h |

**Function**
Contains shifter status flags.

**Diagram**




**Fields**

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>SSF | Shifter Status Flag<br>Indicates the shifter status. This flag is updated in one of the following cases:<br>• If SHIFTCTL*n*[SMOD] = 001b (Receive mode), the status flag is set when SHIFTBUF is loaded with data from the shifter (SHIFTBUF is full). The status flag is cleared when you read Shifter Buffer (SHIFTBUF0 - SHIFTBUF7). |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • If SHIFTCTL*n*[SMOD] = 010b (Transmit mode), the status flag is set when SHIFTBUF data is transferred to the shifter (SHIFTBUF is empty) or when SHIFTCTL*n*[SMOD] is initially configured as 010b (Transmit mode). The status flag is cleared when you write to the SHIFTBUF register.<br>• If SHIFTCTL*n*[SMOD] = 100b (Match Store mode), the status flag is set when a match occurs between SHIFTBUF and the shifter. The status flag is cleared when you read the SHIFTBUF register.<br>• If SHIFTCTL*n*[SMOD] = 101b (Match Continuous mode), the status flag returns the current match result between SHIFTBUF and the shifter. You cannot clear the status flag by reading the SHIFTBUF register.<br>• If SHIFTCTL*n*[SMOD] = 110b (State mode), the status flag for a shifter sets when it is selected by the current state pointer.<br>• If SHIFTCTL*n*[SMOD] = 111b (Logic mode), the status flag returns the current value of the programmable logic block output.<br>You can clear this status flag by writing a logic one to the flag for all modes except Match Continuous mode, State mode, and Logic mode.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0000_0000b - Clear<br>0000_0001b - Set<br>When writing<br>0000_0000b - No effect<br>0000_0001b - Clear the flag |

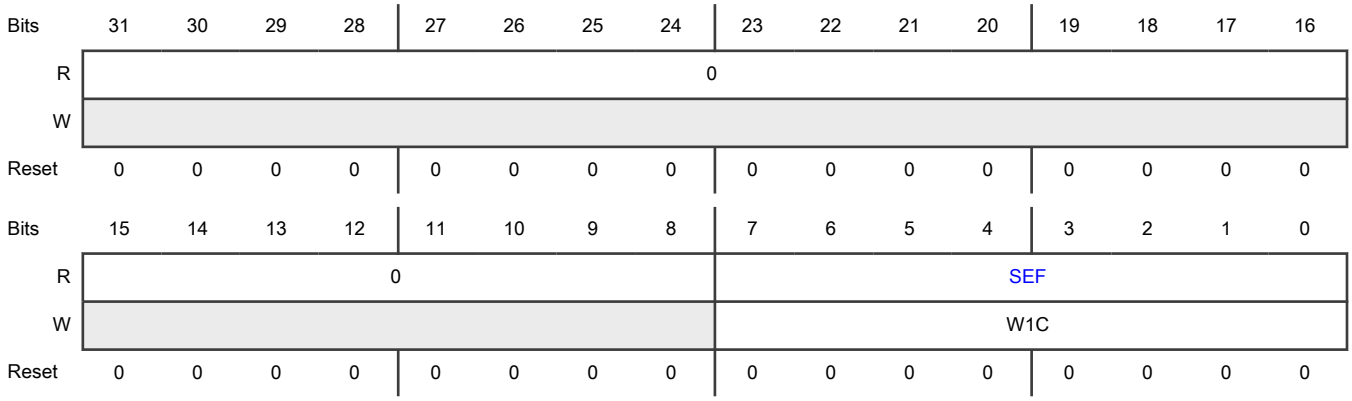
### 69.7.1.7 Shifter Error (SHIFTERR)

**Offset**

| Register | Offset |
|---|---|
| SHIFTERR | 14h |

**Function**
Reports shifter errors.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>SEF | Shifter Error Flag<br>Indicates shifter error flag status. This flag is set when one of the following events occurs:<br>• If SHIFTCTL*n*[SMOD] = 001b (Receive mode), it indicates that either the shifter is ready to store new data into SHIFTBUF before the previous data is read from SHIFTBUF (SHIFTBUF overrun), or the received start or stop bit does not match the expected value.<br>• If SHIFTCTL*n*[SMOD] = 010b (Transmit mode), it indicates that the shifter is ready to load new data from SHIFTBUF before new data is written into SHIFTBUF (SHIFTBUF underrun).<br>• If SHIFTCTL*n*[SMOD] = 100b (Match Store mode), it indicates the occurrence of a match event before the previous match data is read from SHIFTBUF (SHIFTBUF overrun).<br>• If SHIFTCTL*n*[SMOD] = 101b (Match Continuous mode), the error flag is set when a match occurs between SHIFTBUF and the shifter.<br>• If SHIFTCTL*n*[SMOD] = 111b (Logic mode), the error flag is set when the output of the programmable logic block is asserted.<br>For SHIFTCTL*n*[SMOD] = 101b (Match Continuous mode), the flag can also be cleared when you read Shifter Buffer (SHIFTBUF0 - SHIFTBUF7).<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0000_0000b - Clear<br>0000_0001b - Set<br>When writing<br>0000_0000b - No effect<br>0000_0001b - Clear the flag |

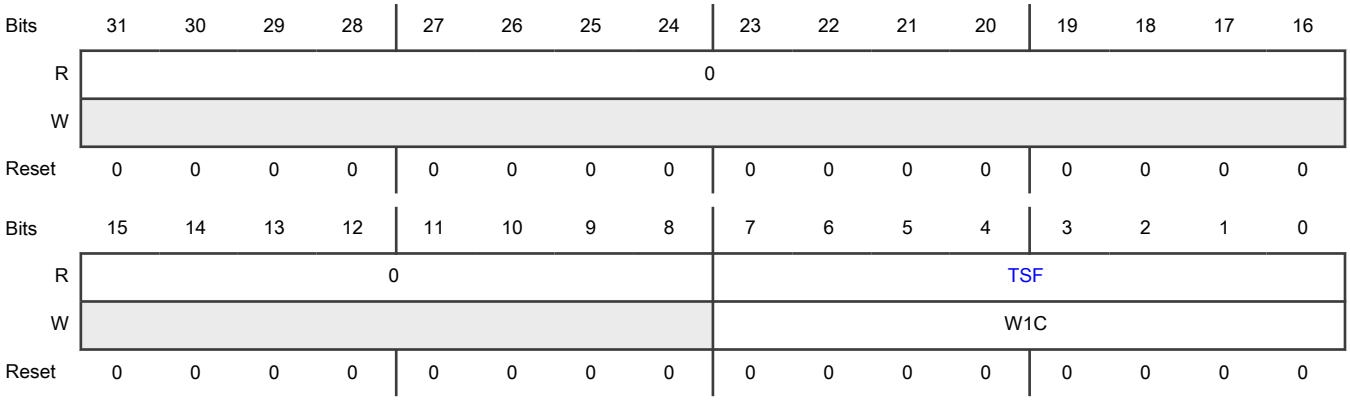
### 69.7.1.8 Timer Status Flag (TIMSTAT)

#### Offset

| Register | Offset |
|---|---|
| TIMSTAT | 18h |

#### Function
Reports timer status.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>TSF | Timer Status Flag<br>Indicates timer status. This flag is set depending on Timer mode:<br>• In 8-bit baud counter mode, this flag is set when the upper 8-bit counter equals zero and decrements.<br>• In 8-bit high PWM mode, this flag is set when the upper 8-bit counter equals zero and decrements.<br>• In 16-bit counter mode, this flag is set when the 16-bit counter equals zero and decrements.<br>• In 16-bit counter disable mode, TSF is set when a timer disable event is detected.<br>• In 8-bit word counter mode, TSF is set when the upper 8-bit counter equals zero and decrements.<br>• In 8-bit low PWM mode, TSF is set when the upper 8-bit counter equals zero and decrements.<br>• In 16-bit input capture mode, TSF is set when a timer disable event is detected and the flag is clear. In this mode, you must read Timer Control (TIMCTL0 - TIMCTL7) only when TSF is set.<br>**NOTE**<br>This field behaves differently for register reads and writes. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When reading<br>0000_0000b - Clear<br>0000_0001b - Set<br>When writing<br>0000_0000b - No effect<br>0000_0001b - Clear the flag |

### 69.7.1.9 Shifter Status Interrupt Enable (SHIFTSIEN)

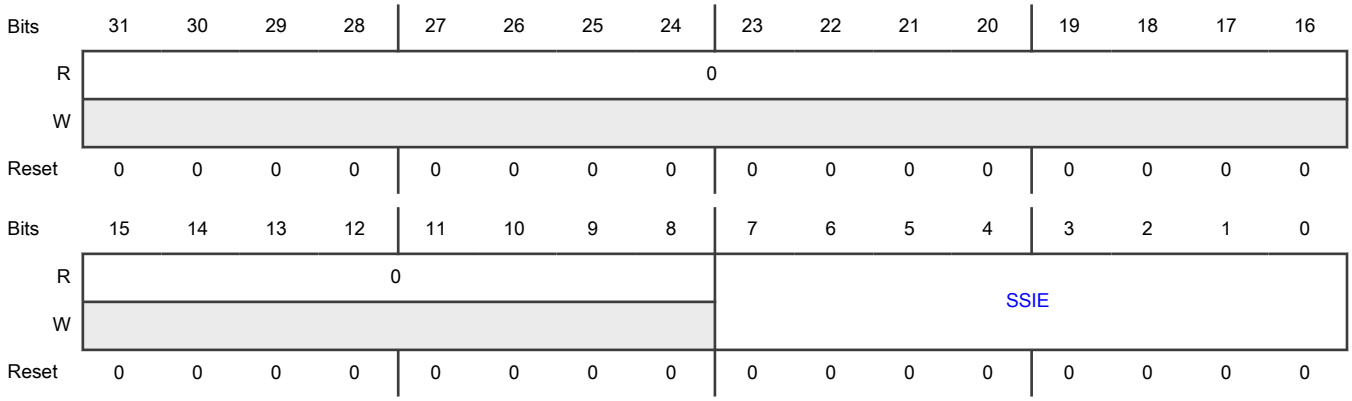
#### Offset

| Register | Offset |
|---|---|
| SHIFTSIEN | 20h |

#### Function

Enables shifter status interrupts.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>SSIE | Shifter Status Interrupt Enable<br>Enables interrupt generation when the corresponding SHIFTSTAT[SSF] flag is set. If you write 0 to this field, SHIFTSTAT[SSF] is disabled; and if you write 1 to this field, SHIFTSTAT[SSF] is enabled. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disable<br>1b - Enable |

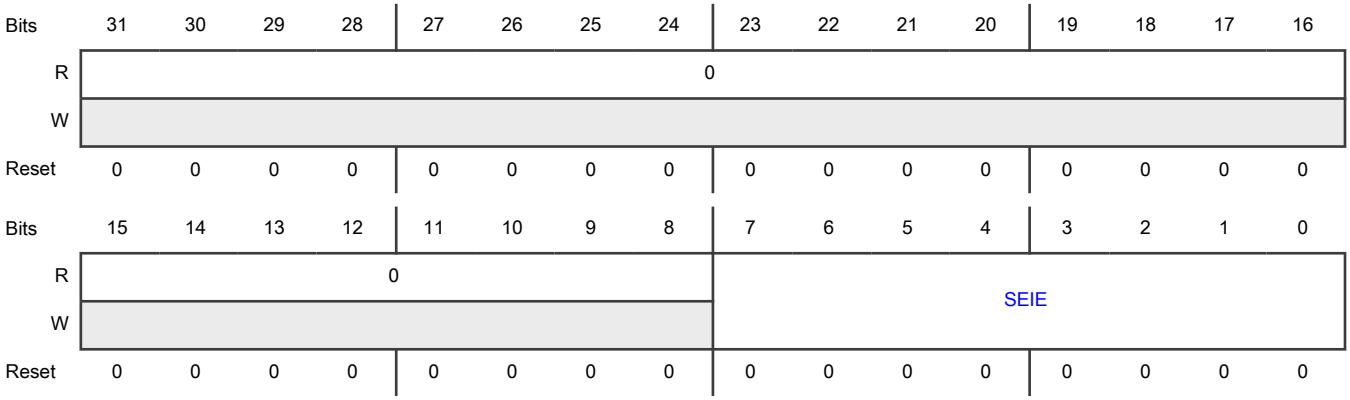
### 69.7.1.10 Shifter Error Interrupt Enable (SHIFTEIEN)

#### Offset

| Register | Offset |
|---|---|
| SHIFTEIEN | 24h |

#### Function
Enables shifter error interrupts.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>SEIE | Shifter Error Interrupt Enable<br>Enables interrupt generation when the corresponding SHIFTERR[SEF] flag is set. If you write 0 to this field, SHIFTERR[SEF] is disabled; and if you write 1 to this field, SHIFTERR[SEF] is enabled.<br>0b - Disable<br>1b - Enable |

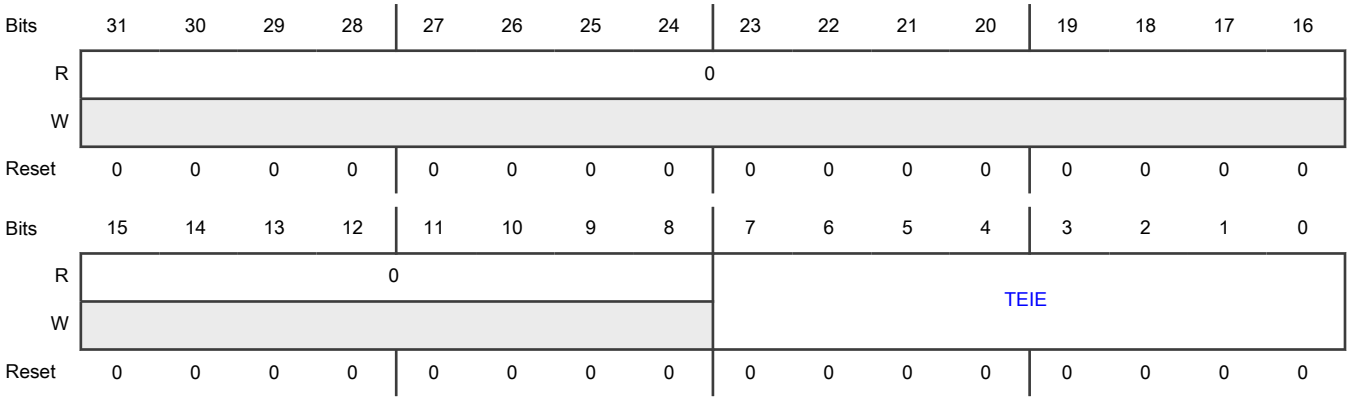
### 69.7.1.11 Timer Interrupt Enable (TIMIEN)

#### Offset

| Register | Offset |
|---|---|
| TIMIEN | 28h |

#### Function
Enables timer status interrupts.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>TEIE | Timer Status Interrupt Enable<br>Enables interrupt generation when the corresponding TIMSTAT[TSF] flag is set. If you write 0 to this field, TIMSTAT[TSF] is disabled; and if you write 1 to this field, TIMSTAT[TSF] is enabled.<br>0b - Disable<br>1b - Enable |

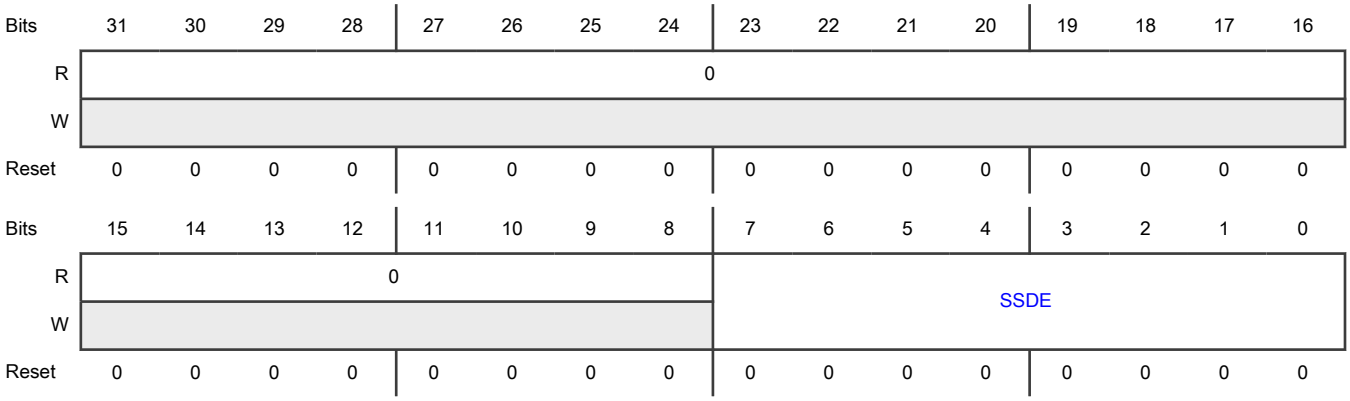
### 69.7.1.12 Shifter Status DMA Enable (SHIFTSDEN)

#### Offset

| Register | Offset |
|---|---|
| SHIFTSDEN | 30h |

#### Function
Enables shifter DMA requests.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>SSDE | Shifter Status DMA Enable<br>Enables DMA request generation when the corresponding SHIFTSTAT[SSF] flag is set. If you write 0 to this field, SHIFTSTAT[SSF] is disabled; and if you write 1 to this field, SHIFTSTAT[SSF] is enabled.<br>0b - Disable<br>1b - Enable |

## 69.7.1.13 Timer Status DMA Enable (TIMERSDEN)

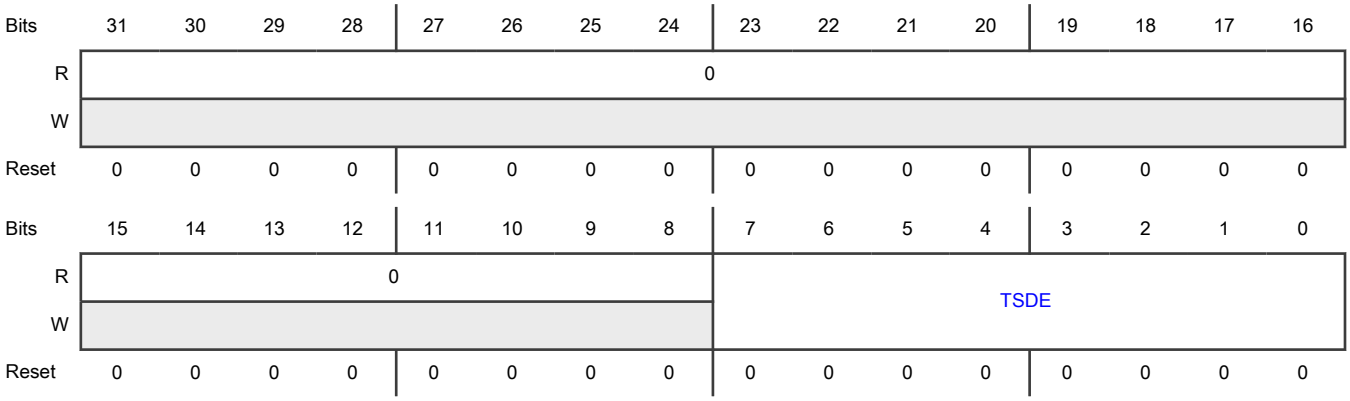
### Offset

| Register | Offset |
|---|---|
| TIMERSDEN | 38h |

### Function

Enables DMA requests when the timer status flag is set.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-0<br>TSDE | Timer Status DMA Enable<br>Enables DMA request generation when the corresponding TIMSTAT[TSF] flag is set.<br>When the timer status DMA request is enabled, reading or writing to a timer compare register clears the corresponding timer status register. The DMA must therefore read or write to the timer compare register as part of the DMA transfer; otherwise, the DMA request remains asserted.<br>0b - Disable<br>1b - Enable |

## 69.7.1.14 Shifter State (SHIFTSTATE)

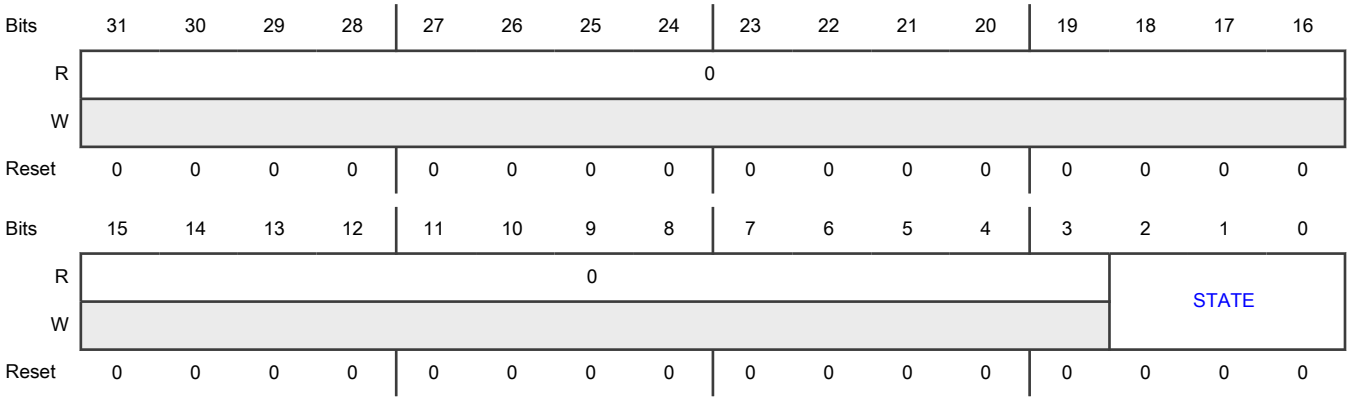
### Offset

| Register | Offset |
|---|---|
| SHIFTSTATE | 40h |

### Function

Contains a pointer to track the current shifter.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2-0<br>STATE | Current State Pointer<br>Maintains a pointer to track the current shifter (configured for State mode) enabled to drive outputs and compute the next state. Reading this register when the state pointer is updating can result in the return of an incorrect state.<br>The value that you write to this field overrides the current state. |

## 69.7.1.15 Trigger Status (TRGSTAT)

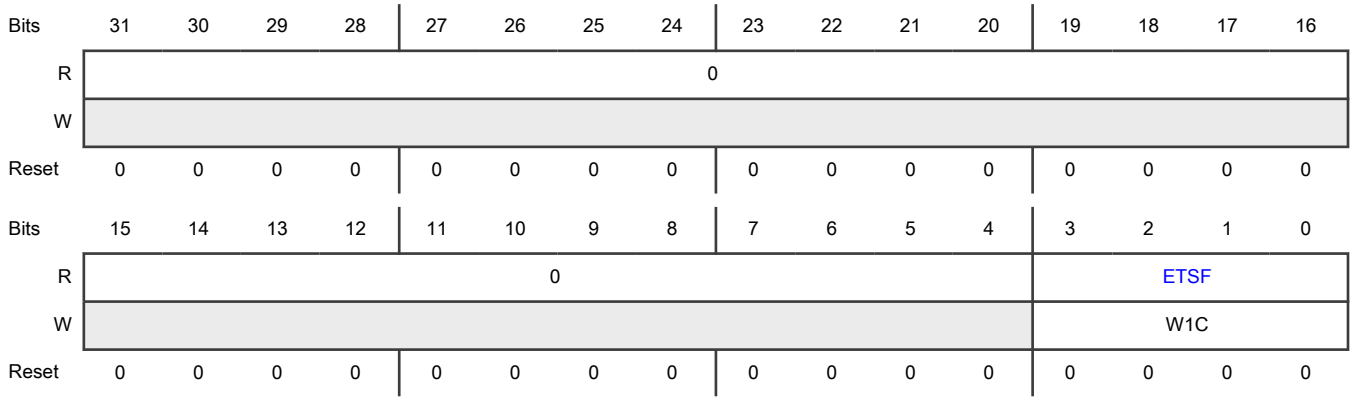
### Offset

| Register | Offset |
|---|---|
| TRGSTAT | 48h |

### Function

Contains external trigger status flags.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>ETSF | External Trigger Status Flag<br>Specifies whether the external trigger status flag is set when a rising edge is detected on the corresponding external trigger input.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0000b - Clear<br>0001b - Set<br>When writing<br>0000b - No effect<br>0001b - Clear the flag |

## 69.7.1.16 External Trigger Interrupt Enable (TRIGIEN)

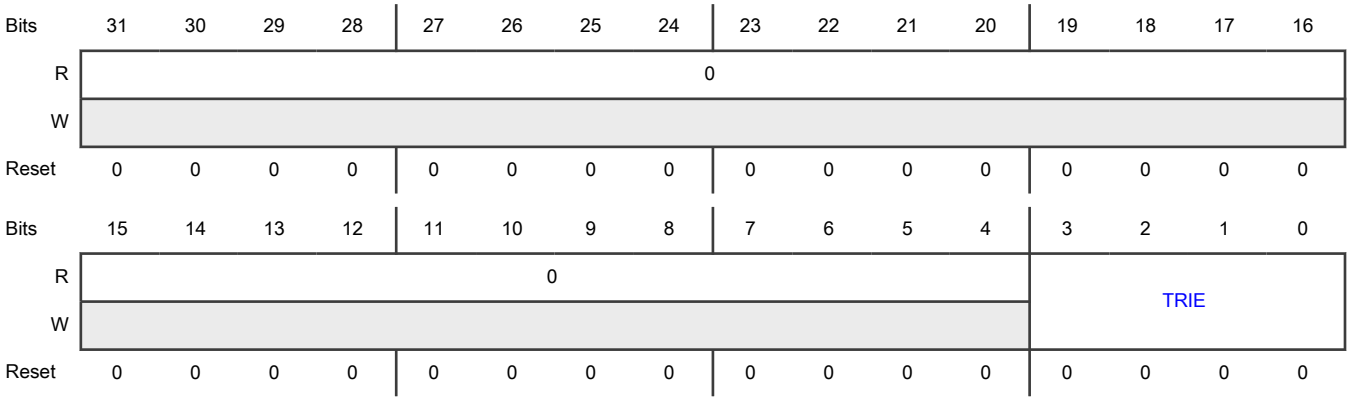
### Offset

| Register | Offset |
|---|---|
| TRIGIEN | 4Ch |

### Function

Enables external trigger interrupts.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3-0<br>TRIE | External Trigger Interrupt Enable<br>Enables interrupt generation when the corresponding TRGSTAT[ETSF] flag is set. If you write 0 to this field, TRGSTAT[ETSF] is disabled, and if you write 1 to this field, TRGSTAT[ETSF] is enabled.<br>0b - Disable<br>1b - Enable |

## 69.7.1.17 Pin Status (PINSTAT)

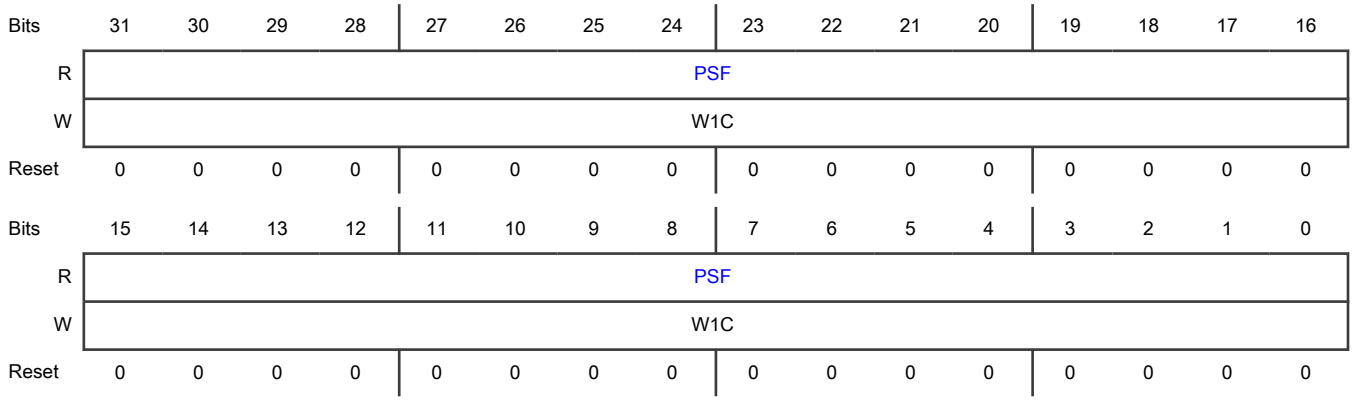
### Offset

| Register | Offset |
|---|---|
| PINSTAT | 50h |

### Function

Contains pin status flags.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>PSF | Pin Status Flag<br>Indicates whether the pin status flag is set when a rising edge or falling edge (if configured) is detected on the corresponding pin, as configured by the pin.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0000_0000_0000_0000_0000_0000_0000_0000b - Clear<br>0000_0000_0000_0000_0000_0000_0000_0001b - Set<br>When writing<br>0000_0000_0000_0000_0000_0000_0000_0000b - No effect<br>0000_0000_0000_0000_0000_0000_0000_0001b - Clear the flag |

## 69.7.1.18 Pin Interrupt Enable (PINIEN)

### Offset

| Register | Offset |
|---|---|
| PINIEN | 54h |

### Function

Enables pin status interrupts.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PSIE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PSIE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>PSIE | Pin Status Interrupt Enable<br>Enables interrupt generation when the corresponding PINSTAT[PSF] flag is set. If you write 0 to this field, PINSTAT[PSF] is disabled, and if you write 1 to this field, PINSTAT[PSF] is enabled.<br>0b - Disable<br>1b - Enable |

## 69.7.1.19 Pin Rising Edge Enable (PINREN)

### Offset

| Register | Offset |
|---|---|
| PINREN | 58h |

### Function
Enables the pin status flag on a rising edge.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PRE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PRE | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>PRE | Pin Rising Edge<br>Specifies whether the pin status flag is set whenever a rising edge is detected on the pin.<br>0b - Not set<br>1b - Set |

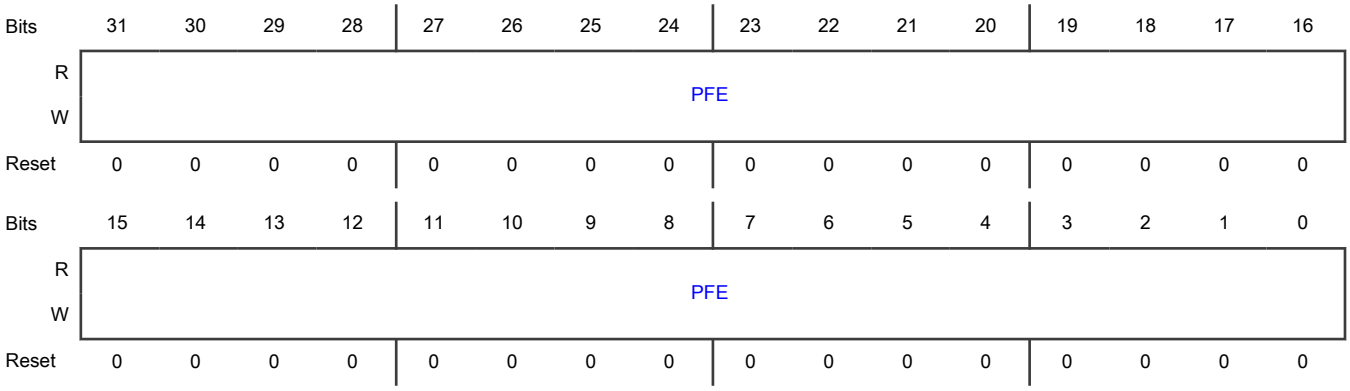
## 69.7.1.20 Pin Falling Edge Enable (PINFEN)

### Offset

| Register | Offset |
|---|---|
| PINFEN | 5Ch |

### Function
Enables the pin status flag on a falling edge.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>PFE | Pin Falling Edge<br>Specifies whether the pin status flag is set whenever a falling edge is detected on the pin.<br>0b - Not set<br>1b - Set |

### 69.7.1.21 Pin Output Data (PINOUTD)

#### Offset

| Register | Offset |
|---|---|
| PINOUTD | 60h |

#### Function
Contains data output when direct pin output is enabled.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | OUTD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | OUTD | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-0<br>OUTD | Output Data<br>Configures the value driven on the corresponding pin when direct pin output is enabled.<br>0b - Logic zero<br>1b - Logic one |

### 69.7.1.22 Pin Output Enable (PINOUTE)

#### Offset

| Register | Offset |
|---|---|
| PINOUTE | 64h |

#### Function
Enables pin output.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>OUTE | Output Enable<br>Enables direct output on the corresponding pin. If this field is 0, the pin is controlled by timer/shifter configuration, and if this field is 1, pin is an output and driven with the value of Pin Output Data (PINOUTD).<br>0b - Controlled by timer/shifter configuration<br>1b - Output; driven with value of PINOUTD |

## 69.7.1.23 Pin Output Disable (PINOUTDIS)

### Offset

| Register | Offset |
|---|---|
| PINOUTDIS | 68h |

### Function
Disables pin output.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>OUTDIS | Output Disable<br>Configures the corresponding pins to disable direct output. If this field is 1, the corresponding fields in Pin Output Data (PINOUTD) and Pin Output Enable (PINOUTE) become 0.<br>0b - No effect<br>1b - Corresponding fields become 0 |

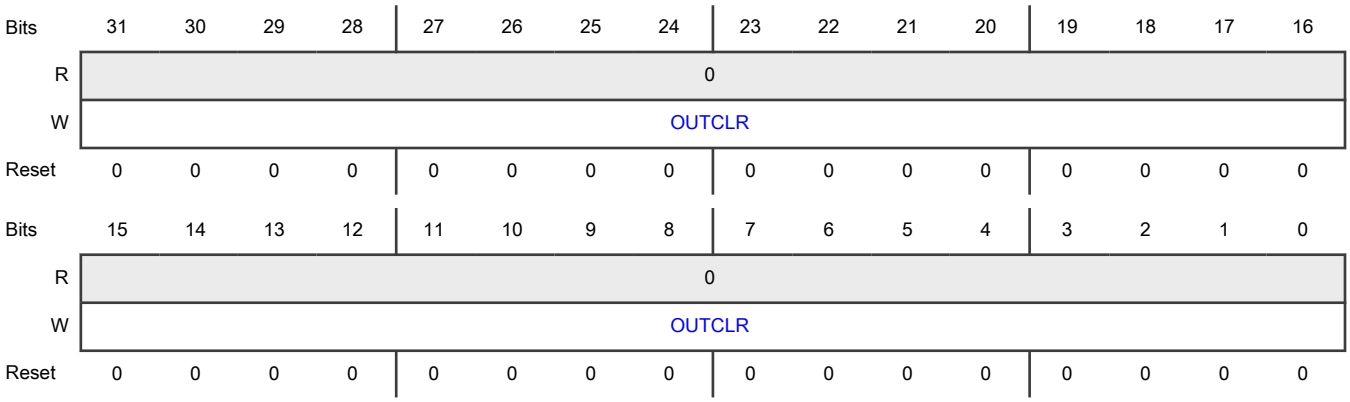
## 69.7.1.24 Pin Output Clear (PINOUTCLR)

### Offset

| Register | Offset |
|---|---|
| PINOUTCLR | 6Ch |

### Function
Clears pin output.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>OUTCLR | Output Clear<br>Configures the corresponding pins to output zero. If this field is 1, the corresponding field in Pin Output Data (PINOUTD) becomes 0 and the one in Pin Output Enable (PINOUTE) becomes 1.<br>0b - No effect<br>1b - Corresponding field in PINOUTD becomes 0; corresponding field in PINOUTE becomes 1 |

### 69.7.1.25 Pin Output Set (PINOUTSET)

#### Offset

| Register | Offset |
|---|---|
| PINOUTSET | 70h |

#### Function
Sets pin output.

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-0<br>OUTSET | Output Set<br>Configures the corresponding pins to output logic one. If this field is 1, the corresponding fields in Pin Output Data (PINOUTD) and Pin Output Enable (PINOUTE) become 1.<br>0b - No effect<br>1b - Corresponding fields become 1 |

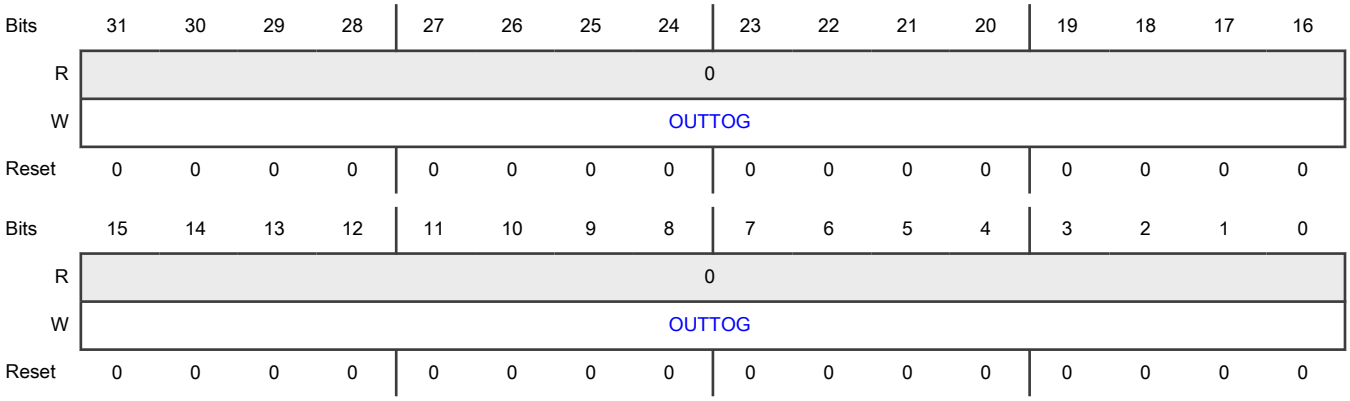
### 69.7.1.26 Pin Output Toggle (PINOUTTOG)

#### Offset

| Register | Offset |
|---|---|
| PINOUTTOG | 74h |

#### Function
Toggles pin output.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>OUTTOG | Output Toggle<br>Configures the corresponding pins to toggle. If this field is 1, the corresponding field in Pin Output Data (PINOUTD) is inverted and the one in Pin Output Enable (PINOUTE) becomes 1.<br>0b - No effect<br>1b - Corresponding field in PINOUTD is inverted; corresponding field in PINOUTE becomes 1 |

## 69.7.1.27 Shifter Control (SHIFTCTL0 - SHIFTCTL7)

### Offset

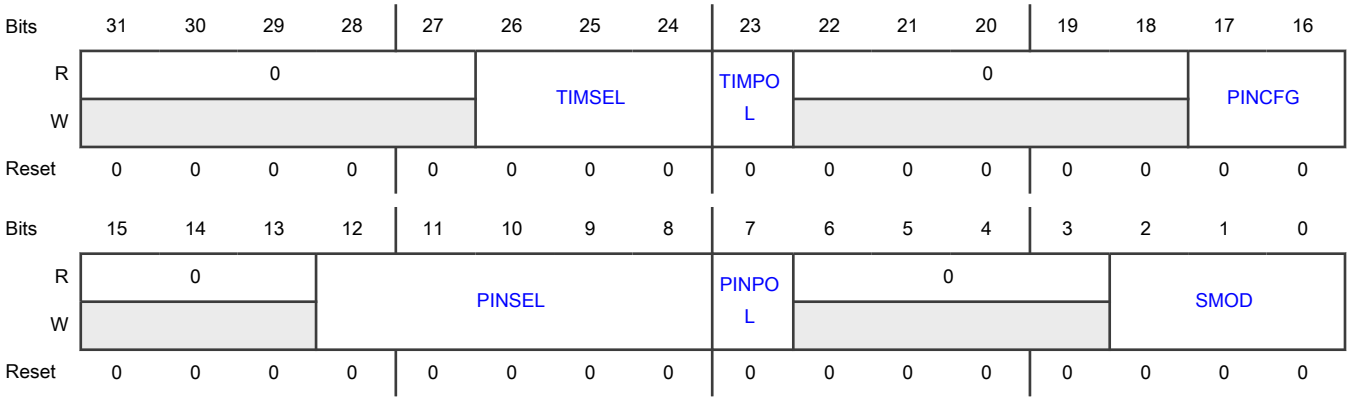
For n = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTCTLn | 80h + (n × 4h) |

### Function

Provides shifter controls.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26-24<br>TIMSEL | Timer Select<br>Selects which timer is used for controlling the logic or shift register and generating the shift clock. TIMSEL = i selects TIMERi. |
| 23<br>TIMPOL | Timer Polarity<br>Determines whether the shift occurs on the positive edge or negative edge of the shift clock.<br>0b - Positive edge<br>1b - Negative edge |
| 22-18<br>— | Reserved |
| 17-16<br>PINCFG | Shifter Pin Configuration<br>Specifies shifter pin configuration.<br>For pins configured as an output (PINCFG = 11b), this field takes effect when you write to the register.<br>**NOTE**<br>When initially configuring PINCFG as 11b, FLEXIO may briefly drive the pin low. To avoid this, you can configure PINCFG as 10b along with the rest of the Control register and then perform a subsequent write to set PINCFG as 11b.<br>Likewise, when changing the value of PINCFG from 11b to 00b, you must perform an initial write to set PINCFG as 10b and then perform a subsequent write to update the rest of the Control register with the value of PINCFG as 00b.<br>00b - Shifter pin output disabled<br>01b - Shifter pin open-drain or bidirectional output enable<br>10b - Shifter pin bidirectional output data<br>11b - Shifter pin output |
| 15-13<br>— | Reserved |
| 12-8<br>PINSEL | Shifter Pin Select<br>Selects the pin that is used by the shifter input or output. PINSEL = i selects the FXIO_Di pin. For pins configured as an output (PINCFG = 11b), this field takes effect when you write to the register. |
| 7<br>PINPOL | Shifter Pin Polarity<br>Specifies the shifter pin polarity. For pins configured as an output (PINCFG = 11b), this field takes effect when you write to this register. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Active high<br>1b - Active low |
| 6-3<br>— | Reserved |
| 2-0<br>SMOD | Shifter Mode<br>Configures the mode of the shifter.<br>000b - Disable<br>001b - Receive mode; capture the current shifter content into SHIFTBUF on expiration of the timer<br>010b - Transmit mode; load SHIFTBUF contents into the shifter on expiration of the timer<br>011b - Reserved<br>100b - Match Store mode; shifter data is compared to SHIFTBUF content on expiration of the timer<br>101b - Match Continuous mode; shifter data is continuously compared to SHIFTBUF contents<br>110b - State mode; SHIFTBUF contents store programmable state attributes<br>111b - Logic mode; SHIFTBUF contents implement programmable logic lookup table |

#### 69.7.1.28 Shifter Configuration (SHIFTCFG0 - SHIFTCFG7)

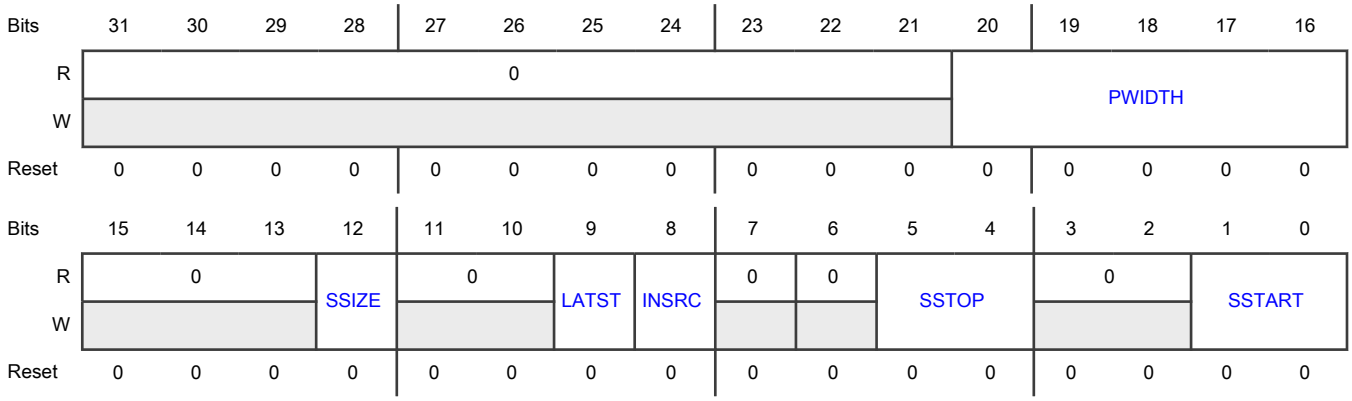
**Offset**

For n = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTCFGn | 100h + (n × 4h) |

**Function**
Provides fields for shifter configuration.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-21<br>— | Reserved |
| 20-16<br>PWIDTH | Parallel Width<br>Configures the number of bits to be shifted on each shift clock for all shifters:<br>• 1-bit shift for PWIDTH = 0<br>• 2-bit shift for PWIDTH = 1<br>• 4-bit shift for PWIDTH = 2...3<br>• 8-bit shift for PWIDTH = 4...7<br>• 16-bit shift for PWIDTH = 8...15<br>• 32-bit shift for PWIDTH = 16...31<br>For shifters that support parallel transmit (SHIFTER0, SHIFTER4, ...) or parallel receive (SHIFTER3, SHIFTER7, ...), this field, together with PINSEL, also selects the pins to be driven or sampled on each shift clock: FXIO_D[PINSEL+PWIDTH]:FXIO_D[PINSEL].<br>Shifters that do not support parallel transmit or parallel receive only support parallel shift when SHIFTCFG*n*[INSRC] = 1.<br>If SHIFTCTL*n*[SMOD] = 110b (State mode), use this field to disable state outputs (see State mode). |
| 15-13<br>— | Reserved |
| 12<br>SSIZE | Shifter Size<br>Configures the size of the Shift registers.<br>A 24-bit Shift register shifts data only into bits [23:0] and does not update bits [31:24] during shift operations.<br>When the Shift register is configured for a 24-bit shift, configuring PWIDTH as 8..15 performs a 12-bit shift and PWIDTH as 16..31 performs a 24-bit shift. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - 32-bit<br>1b - 24-bit |
| 11-10<br>— | Reserved |
| 9<br>LATST | Late Store<br>Configures what happens when a receive or match Shift register is configured to both shift and store on the same cycle.<br>0b - Store the pre-shift register state<br>1b - Store the post-shift register state |
| 8<br>INSRC | Input Source<br>Selects the input source for the shifter. Configuring this field as 1 is not supported for the last shifter.<br>0b - Pin<br>1b - Shifter n+1 output |
| 7<br>— | Reserved |
| 6<br>— | Reserved |
| 5-4<br>SSTOP | Shifter Stop<br>Allows automatic stop bit insertion, if the selected timer has also enabled a stop bit, when SHIFTCTL*n*[SMOD] is 10b (Transmit mode).<br>If SHIFTCTL*n*[SMOD] is 1b or 100b (Receive mode or Match Store mode), this field allows automatic stop bit checking if the selected timer has also enabled a stop bit.<br>If SHIFTCTL*n*[SMOD] is 110b (State mode), this field disables state outputs (see State mode).<br>If SHIFTCTL*n*[SMOD] is 111b (Logic mode), this field masks logic pin inputs (see Logic mode).<br>00b - Stop bit disabled for Transmitter, Receiver, and Match Store modes<br>01b - Stop bit disabled for Transmitter, Receiver, and Match Store modes; when timer is in stop condition, Receiver and Match Store modes store receive data on the configured shift edge<br>10b - Transmitter mode outputs stop bit value 0 in Match Store mode; if stop bit is not 0, Receiver and Match Store modes set error flag (when timer is in stop condition, these modes also store receive data on the configured shift edge)<br>11b - Transmitter mode outputs stop bit value 1 in Match Store mode; if stop bit is not 1, Receiver and Match Store modes set error flag (when timer is in stop condition, these modes also store receive data on the configured shift edge) |
| 3-2 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 1-0<br>SSTART | Shifter Start<br>Allows automatic start bit insertion, if the selected timer has also enabled a start bit, when SHIFTCTL*n*[SMOD] is 10b (Transmit mode).<br>If SHIFTCTL*n*[SMOD] = 1b (Receive mode) or 100b (Match Store mode), this field allows automatic start bit checking if the selected timer has also enabled a start bit.<br>If SHIFTCTL*n*[SMOD] is 110b (State mode), this field disables state outputs (see State mode).<br>If SHIFTCTL*n*[SMOD] = 111b (Logic mode), this field masks logic pin inputs (see Logic mode).<br>00b - Start bit disabled for Transmitter, Receiver, and Match Store modes; Transmitter mode loads data on enable<br>01b - Start bit disabled for Transmitter, Receiver, and Match Store modes; Transmitter mode loads data on first shift<br>10b - Transmitter mode outputs start bit value 0 before loading data on first shift; if start bit is not 0, Receiver and Match Store modes set error flag<br>11b - Transmitter mode outputs start bit value 1 before loading data on first shift; if start bit is not 1, Receiver and Match Store modes set error flag |

### 69.7.1.29 Shifter Buffer (SHIFTBUF0 - SHIFTBUF7)

#### Offset

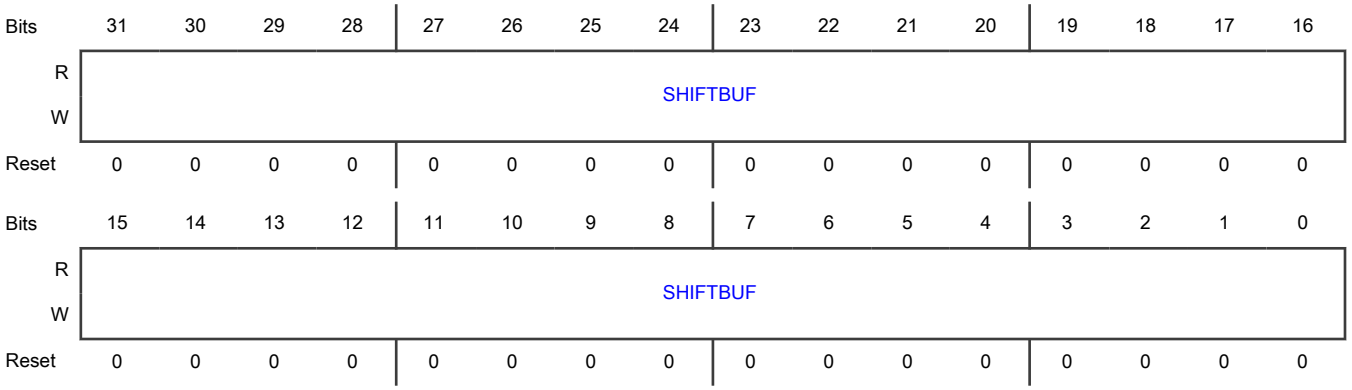
For n = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTBUFn | 200h + (n × 4h) |

#### Function

Contains shift buffer data.

#### Diagram



**Fields**

| Field | Function |
|---|---|
| 31-0<br>SHIFTBUF | Shift Buffer<br>Contains the data to be matched with the shifter contents and is used for various other functions, depending on the setting of SHIFTCTL*n*[SMOD]:<br>• If SHIFTCTL0[SMOD] is 1b (Receive mode), shifter data is transferred into SHIFTBUF at the expiration of the timer. You must read this register only when the corresponding SHIFTSTAT[SSF] flag is set, indicating that new shifter data is available.<br>• If SHIFTCTL0[SMOD] is 10b (Transmit mode), SHIFTBUF data is transferred into the shifter before the timer begins.<br>• If SHIFTCTL0[SMOD] is 100b (Match Store mode), SHIFTBUF[31:16] contains the data to be matched with the shifter contents and SHIFTBUF[15:0] can be used to mask the match result (1 = mask, 0 = no mask). The match is checked when the timer expires. Shifter data [31:16] is written to SHIFTBUF[31:16] whenever a match event occurs. You must read this register only when the corresponding shifter status flag is set, indicating that new shifter data is available.<br>• If SHIFTCTL0[SMOD] is 101b (Match Continuous mode), SHIFTBUF[31:16] contains the data to be matched with the shifter contents, and SHIFTBUF[15:0] can be used to mask the match result (1 = mask, 0 = no mask).<br>• If SHIFTCTL0[SMOD] is 111b (Logic mode), SHIFTBUF[31:0] implements a 5-input, 32-bit programmable logic lookup table (see Logic mode).<br>• If SHIFTCTL0[SMOD] is 110b (State mode), use SHIFTBUF[31:24] to drive the output value when this shifter is selected by the current state pointer and use SHIFTBUF[23:0] to configure the value of the next state transition (see State mode). |

### 69.7.1.30 Shifter Buffer Bit Swapped (SHIFTBUFBIS0 - SHIFTBUFBIS7)

**Offset**

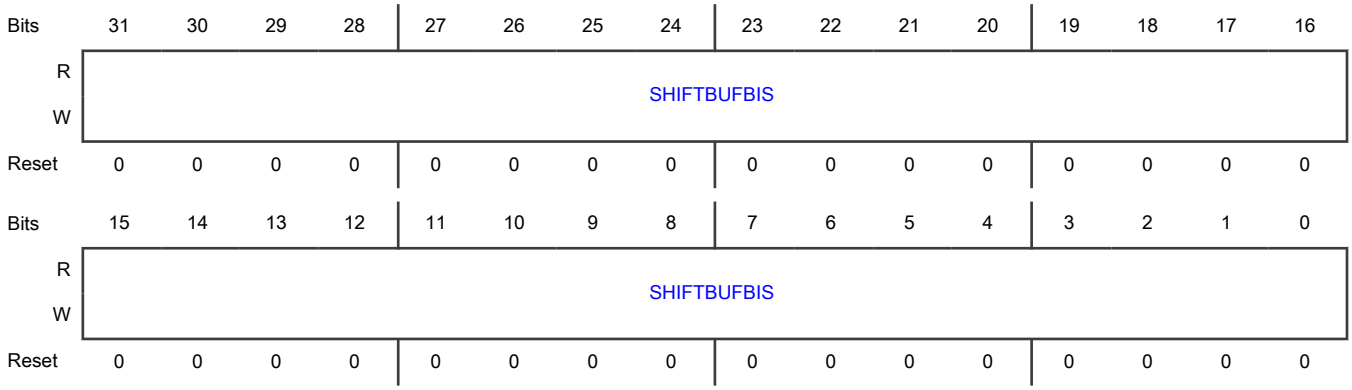
For n = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTBUFBISn | 280h + (n × 4h) |

**Function**

Contains Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) content, but it is bit-swapped.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>SHIFTBUFBIS | Shift Buffer<br>Acts as an alias to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7), but reads or writes to this register are bit-swapped. Reads return SHIFTBUF[0:31]. |

## 69.7.1.31 Shifter Buffer Byte Swapped (SHIFTBUFBYS0 - SHIFTBUFBYS7)

### Offset

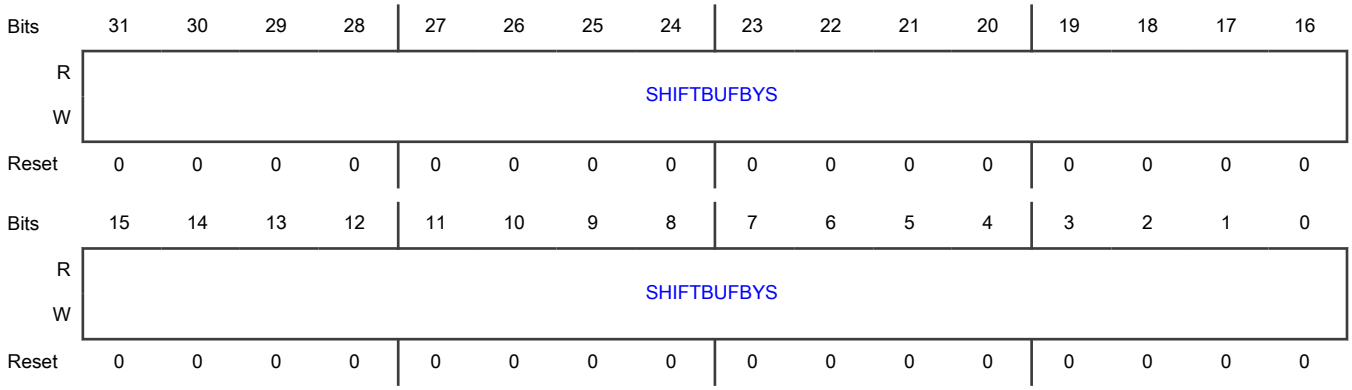
For n = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTBUFBYSn | 300h + (n × 4h) |

### Function

Contains Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) content, but it is byte-swapped.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-0<br>SHIFTBUFBYS | Shift Buffer<br>Acts as an alias to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7), but reads or writes to this register are byte-swapped. Reads return {SHIFTBUF[7:0], SHIFTBUF[15:8], SHIFTBUF[23:16], SHIFTBUF[31:24]}. |

## 69.7.1.32 Shifter Buffer Bit Byte Swapped (SHIFTBUFBBS0 - SHIFTBUFBBS7)

### Offset

For n = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTBUFBBSn | 380h + (n × 4h) |

### Function

Contains the register data for Shifter Buffer (SHIFTBUF0 - SHIFTBUF7), but it is bit-swapped within each byte.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFBBS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFBBS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>SHIFTBUFBBS | Shift Buffer<br>Acts as an alias to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7), except that reads or writes to this register are bit-swapped within each byte. Reads return {SHIFTBUF[24:31], SHIFTBUF[16:23], SHIFTBUF[8:15], SHIFTBUF[0:7]}. |

## 69.7.1.33 Timer Control (TIMCTL0 - TIMCTL7)

### Offset

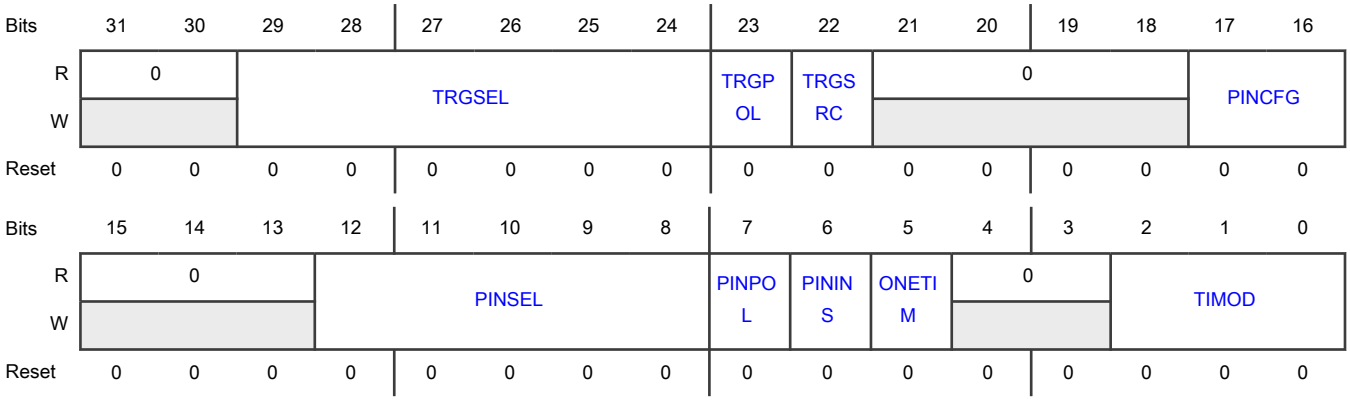
For n = 0 to 7:

| Register | Offset |
|---|---|
| TIMCTLn | 400h + (n × 4h) |

### Function

Controls various settings for timer *n*.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-24<br>TRGSEL | Trigger Select<br>Selects the trigger.<br>The valid values for TRGSEL depend on the configuration of Parameter (PARAM):<br>• If TRGSRC = 1, the valid values for *n* depend on the settings of PARAM[PIN], PARAM[TIMER], and PARAM[SHIFTER].<br>• If TRGSRC = 0, the valid values for *n* depend on PARAM[TRIGGER].<br>See the chip-specific FLEXIO information for external trigger selection.<br>**NOTE**<br>For a pin, *n* = 0 to 31, for a shifter, *n* = 0 to 7, and for a timer, *n* = 0 to 7.<br>If TRGSRC = 0, configure the trigger selection as *n* = external trigger *n* input.<br>If TRGSRC = 1, you can configure the internal trigger to select an input pin as 2×*n* = pin *n* input.<br>If TRGSRC = 1, you can configure the internal trigger to select a shifter or timer signal as:<br>• 4×*n* + 1 = shifter *n* status flag<br>• 4×*n* + 3 = timer *n* trigger output<br>Following are the values for expanded internal trigger selection (TRGSRC = 1): |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • 0000 = Pin 0<br>• 0001 = Shifter 0 flag<br>• 0010 = Pin 1<br>• 0011 = Timer 0 trigger<br>• 0100 = Pin 2<br>• 0101 = Shifter 1 flag<br>• 0110 = Pin 3<br>• 0111 = Timer 1 trigger<br>• ...<br>• This continues up to pin 31, shifter 7, and timer 7. |
| 23<br>TRGPOL | Trigger Polarity<br>Specifies whether the trigger is active high or active low.<br>0b - Active high<br>1b - Active low |
| 22<br>TRGSRC | Trigger Source<br>Specifies whether the selected trigger source is external or internal.<br>0b - External<br>1b - Internal |
| 21-18<br>— | Reserved |
| 17-16<br>PINCFG | Timer Pin Configuration<br>Configures the direction of the timer pin. For pins configured as an output (PINCFG = 11b), this field takes effect when you write to the register.<br>**NOTE**<br>When you initially configure PINCFG as 11b, FLEXIO may briefly drive the pin low. To avoid this, configure PINCFG as 10b along with the rest of the Control register and then perform a subsequent write to set the value of PINCFG as 11b.<br>Likewise, when changing the value of PINCFG from 11b to 00b, you must perform an initial write to set PINCFG as 10b, and then perform a subsequent write to update the rest of the Control register with PINCFG as 00b.<br>00b - Timer pin output disabled<br>01b - Timer pin open-drain or bidirectional output enable<br>10b - Timer pin bidirectional output data<br>11b - Timer pin output |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 15-13<br>— | Reserved |
| 12-8<br>PINSEL | Timer Pin Select<br>Selects the pin that is used by the timer input or output. PINSEL = i selects the FXIO_Di pin. For pins configured as an output (PINCFG = 11b), this field takes effect when you write to the register. |
| 7<br>PINPOL | Timer Pin Polarity<br>Specifies the timer pin polarity. For pins configured as an output (PINCFG = 11b), this field takes effect when you write to the register.<br>0b - Active high<br>1b - Active low |
| 6<br>PININS | Timer Pin Input Select<br>Specifies what selects the timer pin input. If this field is 1, the timer input pin is different from the timer output pin. PINSEL must select an even-numbered pin when this field is 1, which means that the output pin is even-numbered and input pin is odd-numbered.<br>0b - PINSEL selects timer pin input and output<br>1b - PINSEL + 1 selects the timer pin input; timer pin output remains selected by PINSEL |
| 5<br>ONETIM | Timer One Time Operation<br>Configures the timer to perform a single enable or disable iteration. Clear the timer status flag for the timer to be enabled again.<br>0b - Generate the timer enable event as normal<br>1b - Block the timer enable event unless the timer status flag is clear |
| 4-3<br>— | Reserved |
| 2-0<br>TIMOD | Timer Mode<br>Specifies the timer mode:<br>• In 8-bit baud counter mode, the lower 8 bits of the counter and compare register are used to configure the baud rate of the timer shift clock. The upper 8 bits are used to configure the shifter bit count.<br>• In 8-bit PWM high mode, the lower 8 bits of the counter and compare register are used to configure the high period of the timer shift clock. The upper 8 bits are used to configure the low period of the timer shift clock. The shifter bit count is configured using another timer or external signal.<br>• In 16-bit counter mode, the full 16 bits of the counter and compare register are used to configure either the baud rate of the shift clock or the shifter bit count.<br>• In 16-bit counter disable mode, the full 16 bits of the counter and compare register are used to configure either the baud rate of the shift clock or the shifter bit count. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • In 8-bit word counter mode, the lower 8 bits of the counter and compare register are used to configure the shifter bit count. The upper 8 bits are used to configure the shifter word count.<br>• In 8-bit PWM low mode, the lower 8 bits of the counter and compare register are used to configure the low period of the timer shift clock. The upper 8 bits are used to configure the high period of the timer shift clock. Use another timer or external signal to configure the shifter bit count.<br>• In 16-bit input capture mode, the inverted value of the 16-bit counter is latched into the compare register when a timer counter disable condition is detected (as configured by TIMCFG*n*[TIMDIS]). This also sets the timer status flag. The timer counter is immediately restarted from FFFFh.<br>000b - Timer disabled<br>001b - Dual 8-bit counters baud mode<br>010b - Dual 8-bit counters PWM high mode<br>011b - Single 16-bit counter mode<br>100b - Single 16-bit counter disable mode<br>101b - Dual 8-bit counters word mode<br>110b - Dual 8-bit counters PWM low mode<br>111b - Single 16-bit input capture mode |

### 69.7.1.34 Timer Configuration (TIMCFG0 - TIMCFG7)

#### Offset

For n = 0 to 7:

| Register | Offset |
|---|---|
| TIMCFGn | 480h + (n × 4h) |

#### Function

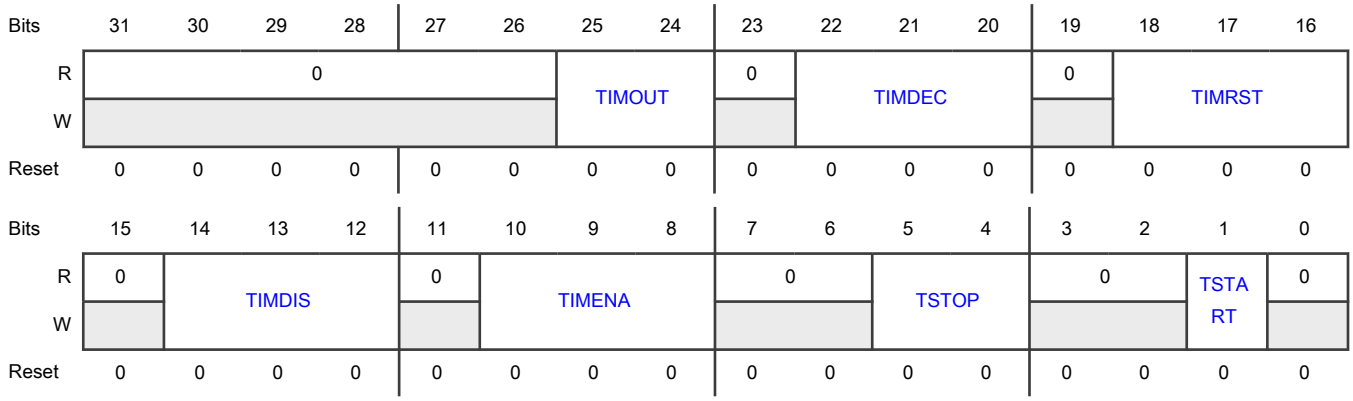
Controls various aspects of timer configuration.

The options to enable or disable the timer using the timer *n* - 1 enable or disable are reserved when *n* is evenly divisible by 4 (timer 0, for example).

**NOTE**

The pin and trigger level and edges specified in this register refer to the signal state after being modified by the settings of TIMCTL*n*[PINPOL] and TIMCTL*n*[TRGPOL]. For example, "trigger low" means that a trigger is actually at logic level 1 if TIMCTL*n*[TRGPOL] is 1 (active low).

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25-24<br>TIMOUT | Timer Output<br>Configures the initial state of the timer output and whether it is affected by the timer reset.<br>00b - Logic one when enabled; not affected by timer reset<br>01b - Logic zero when enabled; not affected by timer reset<br>10b - Logic one when enabled and on timer reset<br>11b - Logic zero when enabled and on timer reset |
| 23<br>— | Reserved |
| 22-20<br>TIMDEC | Timer Decrement<br>Configures the source of the timer decrement and that of the shift clock.<br>000b - Decrement counter on FLEXIO clock; shift clock equals timer output<br>001b - Decrement counter on trigger input (both edges); shift clock equals timer output<br>010b - Decrement counter on pin input (both edges); shift clock equals pin input<br>011b - Decrement counter on trigger input (both edges); shift clock equals trigger input<br>100b - Decrement counter on FLEXIO clock divided by 16; shift clock equals timer output<br>101b - Decrement counter on FLEXIO clock divided by 256; shift clock equals timer output<br>110b - Decrement counter on pin input (rising edge); shift clock equals pin input<br>111b - Decrement counter on trigger input (rising edge); shift clock equals trigger input |
| 19<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 18-16<br>TIMRST | Timer Reset<br>Configures the condition that causes the timer counter (and optionally the timer output) to be reset. In 8-bit counter mode, the timer reset only resets the lower 8 bits that configure the baud rate. In all other modes, the timer reset resets full 16 bits of the counter.<br>000b - Never reset timer<br>001b - Timer reset on timer output high.<br>010b - Timer reset on timer pin equal to timer output<br>011b - Timer reset on timer trigger equal to timer output<br>100b - Timer reset on timer pin rising edge<br>101b - Reserved<br>110b - Timer reset on trigger rising edge<br>111b - Timer reset on trigger rising or falling edge |
| 15<br>— | Reserved |
| 14-12<br>TIMDIS | Timer Disable<br>Configures the condition that causes the timer to be disabled and stop decrementing.<br>000b - Timer never disabled<br>001b - Timer disabled on timer n-1 disable<br>010b - Timer disabled on timer compare (upper 8 bits match and decrement)<br>011b - Timer disabled on timer compare (upper 8 bits match and decrement) and trigger low<br>100b - Timer disabled on pin rising or falling edge<br>101b - Timer disabled on pin rising or falling edge provided trigger is high<br>110b - Timer disabled on trigger falling edge<br>111b - Reserved |
| 11<br>— | Reserved |
| 10-8<br>TIMENA | Timer Enable<br>Configures the condition that causes the timer to be enabled and start decrementing.<br>000b - Timer always enabled<br>001b - Timer enabled on timer n-1 enable<br>010b - Timer enabled on trigger high<br>011b - Timer enabled on trigger high and pin high |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 100b - Timer enabled on pin rising edge<br>101b - Timer enabled on pin rising edge and trigger high<br>110b - Timer enabled on trigger rising edge<br>111b - Timer enabled on trigger rising or falling edge |
| 7-6<br>— | Reserved |
| 5-4<br>TSTOP | Timer Stop<br>Specifies whether the stop bit is enabled. The stop bit can be added on a timer compare (between each word) or on a timer disable. When stop bit is enabled, configured shifters output the contents of the stop bit when the timer is disabled. When stop bit is enabled on timer disable, the timer remains disabled until the next rising edge of the shift clock. If configured for both timer compare and timer disable, only one stop bit is inserted on timer disable.<br>00b - Disabled<br>01b - Enabled on timer compare<br>10b - Enabled on timer disable<br>11b - Enabled on timer compare and timer disable |
| 3-2<br>— | Reserved |
| 1<br>TSTART | Timer Start<br>Specifies whether the start bit is enabled. If it is enabled, configured shifters output the contents of the start bit when the timer is enabled. The timer counter reloads from the compare register on the first rising edge of the shift clock.<br>0b - Disabled<br>1b - Enabled |
| 0<br>— | Reserved |

### 69.7.1.35 Timer Compare (TIMCMP0 - TIMCMP7)

#### Offset

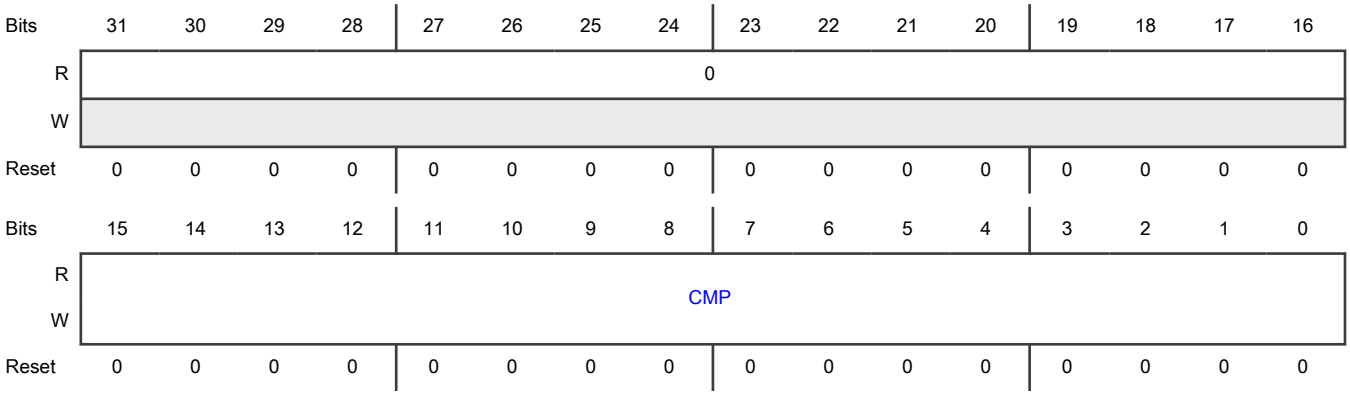
For n = 0 to 7:

| Register | Offset |
|---|---|
| TIMCMPn | 500h + (n × 4h) |

### Function

Contains the timer compare value.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>CMP | Timer Compare Value<br>Loads into the timer counter when the timer is first enabled, when the timer is reset, and when the timer decrements down to zero.<br>In 8-bit baud counter mode, the lower 8 bits configure the baud rate divider as (CMP[7:0] + 1) × 2. The upper 8 bits configure the number of bits in each word as (CMP[15:8] + 1) ÷ 2.<br>In 8-bit PWM high mode, the lower 8 bits configure the high period of the output to (CMP[7:0] + 1) and the upper 8 bits configure the low period of the output to (CMP[15:8] + 1).<br>In 16-bit counter mode, the compare value can be used to generate the baud rate divider (if shift clock source is timer output) as (CMP[15:0] + 1) × 2. When the shift clock source is a pin or trigger input, the compare register is used to set the number of bits in each word as (CMP[15:0] + 1) ÷ 2.<br>In 16-bit counter disable mode, the compare value can be used to generate the baud rate divider (if shift clock source is timer output) as (CMP[15:0] + 1) × 2. When the shift clock source is a pin or trigger input, the compare register is used to set the number of bits in each word as (CMP[15:0] + 1) ÷ 2.<br>In 8-bit word counter mode, the lower 8 bits configure the number of bits in each word as (CMP[7:0] + 1) ÷ 2. The upper 8 bits configure the number of words to transfer equal to (CMP[15:8] + 1) ÷ 2.<br>In 8-bit PWM low mode, the lower 8 bits configure the low period of the output to (CMP[7:0] + 1) and the upper 8 bits configure the high period of the output to (CMP[15:8] + 1).<br>In 16-bit input capture mode, the compare register is updated with the inverse of the timer counter value whenever the timer status flag is set. You must read this register only when the timer status flag is set. |

## 69.7.1.36 Shifter Buffer Nibble Byte Swapped (SHIFTBUFNBS0 - SHIFTBUFNBS7)

### Offset

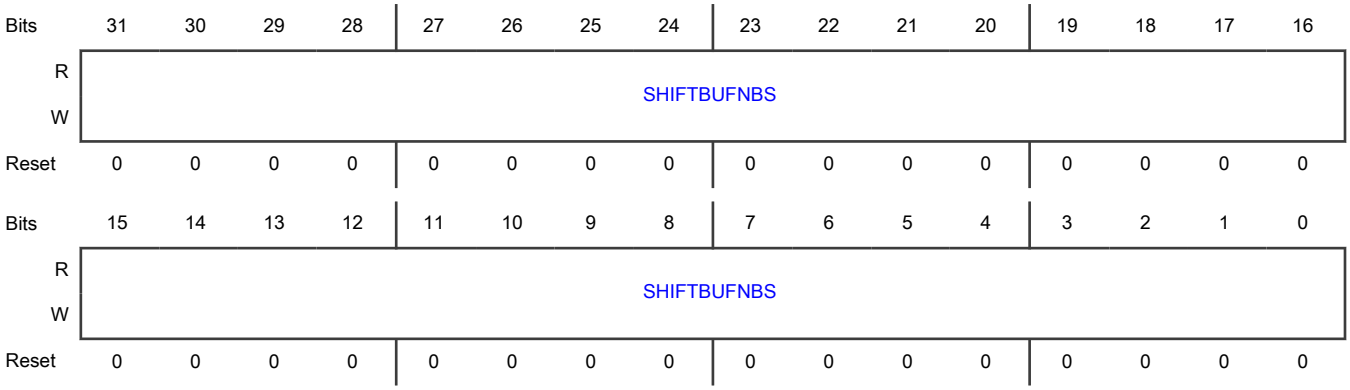
For n = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTBUFNBSn | 680h + (n × 4h) |

### Function

Contains Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) content, but it is nibble-swapped within each byte.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>SHIFTBUFNBS | Shift Buffer<br>Acts as an alias to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7), but reads or writes to this register are nibble-swapped within each byte. Reads return {SHIFTBUF[27:24], SHIFTBUF[31:28], SHIFTBUF[19:16], SHIFTBUF[23:20], SHIFTBUF[11:8], SHIFTBUF[15:12], SHIFTBUF[3:0], SHIFTBUF[7:4]}. |

## 69.7.1.37 Shifter Buffer Halfword Swapped (SHIFTBUFHWS0 - SHIFTBUFHWS7)

### Offset

For n = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTBUFHWSn | 700h + (n × 4h) |

### Function

Contains Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) content, but it is halfword-swapped.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFHWS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFHWS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>SHIFTBUFHWS | Shift Buffer<br>Acts as an alias to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7), but reads or writes to this register are halfword-swapped. Reads return {SHIFTBUF[15:0], SHIFTBUF[31:16]}. |

## 69.7.1.38 Shifter Buffer Nibble Swapped (SHIFTBUFNIS0 - SHIFTBUFNIS7)

### Offset

For n = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTBUFNISn | 780h + (n × 4h) |

### Function

Contains Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) content, but it is nibble-swapped.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFNIS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFNIS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>SHIFTBUFNIS | Shift Buffer<br>Acts as an alias to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7), but reads or writes to this register are nibble-swapped. Reads return {SHIFTBUF[3:0], SHIFTBUF[7:4], SHIFTBUF[11:8], SHIFTBUF[15:12], SHIFTBUF[19:16], SHIFTBUF[23:20], SHIFTBUF[27:24], SHIFTBUF[31:28]}. |

## 69.7.1.39 Shifter Buffer Odd Even Swapped (SHIFTBUFOES0 - SHIFTBUFOES7)

### Offset

For n = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTBUFOESn | 800h + (n × 4h) |

### Function

Contains Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) content, but it has odd and even bits partitioned separately.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFOES | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFOES | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>SHIFTBUFOES | Shift Buffer<br>Acts as an alias to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7), but reads or writes to this register have the odd and even bits partitioned separately. Only 32-bit accesses are supported for this register. Reads return {SHIFTBUF[31], SHIFTBUF[29], SHIFTBUF[27],SHIFTBUF[25],SHIFTBUF[23],SHIFTBUF[21],SHIFTBUF[19],SHIFTBUF[17],SHIFTBUF[15],SHIFTBUF[13],SHIFTBUF[11],SHIFTBUF[9], SHIFTBUF[7], SHIFTBUF[5], SHIFTBUF[3], SHIFTBUF[1], SHIFTBUF[30], SHIFTBUF[28], SHIFTBUF[26], SHIFTBUF[24], SHIFTBUF[22], SHIFTBUF[20], SHIFTBUF[18], SHIFTBUF[16], SHIFTBUF[14], SHIFTBUF[12], SHIFTBUF[10], SHIFTBUF[8], SHIFTBUF[6], SHIFTBUF[4], SHIFTBUF[2], SHIFTBUF[0]}. |

### 69.7.1.40 Shifter Buffer Even Odd Swapped (SHIFTBUFEOS0 - SHIFTBUFEOS7)

#### Offset

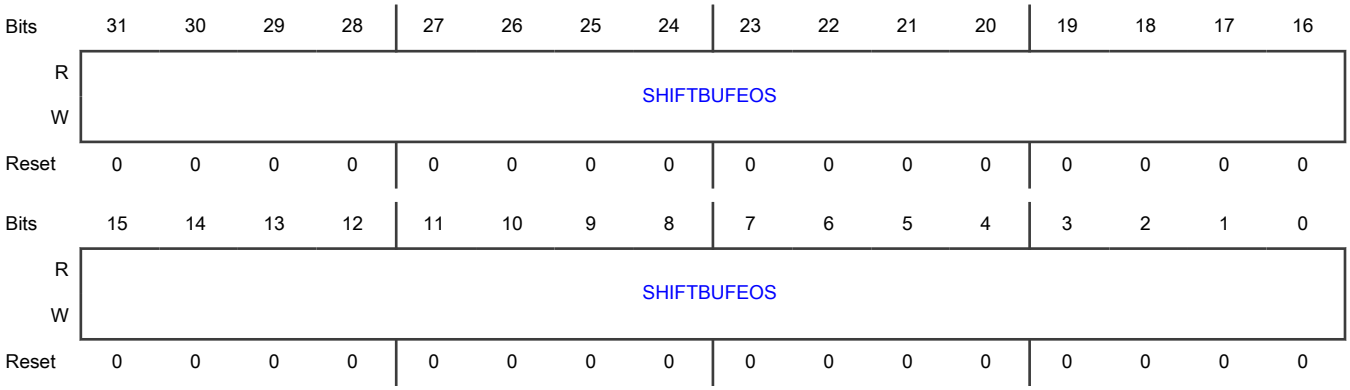
For n = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTBUFEOSn | 880h + (n × 4h) |

#### Function

Contains Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) content, with even and odd bits partitioned separately.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-0<br>SHIFTBUFEOS | Shift Buffer<br>Acts as an alias to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7), but reads or writes to this register have the even and odd bits partitioned separately. Only 32-bit accesses are supported for this register. Reads return {SHIFTBUF[30], SHIFTBUF[28], SHIFTBUF[26], SHIFTBUF[24], SHIFTBUF[22], SHIFTBUF[20], SHIFTBUF[18], SHIFTBUF[16], SHIFTBUF[14], SHIFTBUF[12], SHIFTBUF[10], SHIFTBUF[8], SHIFTBUF[6], SHIFTBUF[4], SHIFTBUF[2], SHIFTBUF[0], SHIFTBUF[31], SHIFTBUF[29], SHIFTBUF[27],SHIFTBUF[25],SHIFTBUF[23],SHIFTBUF[21],SHIFTBUF[19],SHIFTBUF[17],SHIFTBUF[15],SHIFTBUF[13],SHIFTBUF[11],SHIFTBUF[9], SHIFTBUF[7], SHIFTBUF[5], SHIFTBUF[3], SHIFTBUF[1]}. |

### 69.7.1.41 Shifter Buffer Halfword Byte Swapped (SHIFTBUFHBS0 - SHIFTBUFHBS7)

#### Offset

For n = 0 to 7:

| Register | Offset |
|---|---|
| SHIFTBUFHBSn | 900h + (n × 4h) |

### Function

Contains Shifter Buffer (SHIFTBUF0 - SHIFTBUF7) content, but it is halfword byte-swapped.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFHBS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SHIFTBUFHBS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>SHIFTBUFHBS | Shift Buffer<br>Acts as an alias to Shifter Buffer (SHIFTBUF0 - SHIFTBUF7), but reads or writes to this register are halfword byte-swapped. Reads return {SHIFTBUF[23:16], SHIFTBUF[31:24], SHIFTBUF[7:0], SHIFTBUF[15:8]}. |

## 69.8 Glossary

| | |
|---|---|
| **Baud rate** | Number of bits that LPUART transmits or receives per second |
| **Break character** | Break character is generated when the transmitter is holding the data line at the space level for at least 1 character time |
| **Idle character** | Idle character is generated when the transmitter is holding the data line at logic 1 for at least 1 character time |
| **PWM** | Pulse width modulation |

# Chapter 70
# CAN (FlexCAN)

## 70.1 Chip-specific FlexCAN information

### 70.1.1 FlexCAN instances and configuration

This chip contains up to 8 instances of FlexCAN:

- FlexCAN_0
- FlexCAN_1
- FlexCAN_2
- FlexCAN_3
- FlexCAN_4
- FlexCAN_5

**Table 417. FlexCAN configuration and instances**

| Chip | Instances | Flexible Data Rate (CAN FD) protocol specification and CAN protocol. | Number of message buffers (assuming 8B payload) | Rx FIFO ID Filtering | ECC | External tick time | DMA | Enhanced RX FIFO feature (size) |
|---|---|---|---|---|---|---|---|---|
| MCXE31B | FlexCAN_0 | Yes | 96 | Yes | Yes | Yes (PIT_RTI_2) | Yes | Yes (20) |
| | FlexCAN_1 | Yes | 64 | Yes | Yes | Yes (PIT_RTI_2) | Yes | No |
| | FlexCAN_2 | Yes | 64 | Yes | Yes | Yes (PIT_RTI_2) | Yes | No |
| | FlexCAN_3 | Yes | 32 | Yes | Yes | Yes (PIT_RTI_2) | Yes | No |
| | FlexCAN_4 | Yes | 32 | Yes | Yes | Yes (PIT_RTI_2) | Yes | No |
| | FlexCAN_5 | Yes | 32 | Yes | Yes | Yes (PIT_RTI_2) | Yes | No |
| MCXE317 | FlexCAN_0 | Yes | 64 | Yes | Yes | Yes (PIT_RTI_1) | Yes | Yes (20) |
| | FlexCAN_1 | Yes | 64 | Yes | Yes | Yes (PIT_RTI_1) | Yes | No |
| | FlexCAN_2 | Yes | 64 | Yes | Yes | Yes (PIT_RTI_1) | Yes | No |
| | FlexCAN_3 | Yes | 32 | Yes | Yes | Yes (PIT_RTI_1) | Yes | No |

*Table continues on the next page...*

Table 417. FlexCAN configuration and instances (continued)

| Chip | Instances | Flexible Data Rate (CAN FD) protocol specification and CAN protocol. | Number of message buffers (assuming 8B payload) | Rx FIFO ID Filtering | ECC | External tick time | DMA | Enhanced RX FIFO feature (size) |
|---|---|---|---|---|---|---|---|---|
| | FlexCAN_4 | Yes | 32 | Yes | Yes | Yes (PIT_RTI_1) | Yes | No |
| | FlexCAN_5 | Yes | 32 | Yes | Yes | Yes (PIT_RTI_1) | Yes | No |
| MCXE315/ MCXE316 | FlexCAN_0 | Yes | 64 | Yes | Yes | Yes (PIT_RTI_1) | Yes | Yes (20) |
| | FlexCAN_1 | Yes | 64 | Yes | Yes | Yes (PIT_RTI_1) | Yes | No |
| | FlexCAN_2 | Yes | 64 | Yes | Yes | Yes (PIT_RTI_1) | Yes | No |

**NOTE**

See attached interrupt map file for the vector numbers for the CAN instances and their corresponding interrupt sources.

## 70.1.2 FlexCAN error injection

### Error injection address mapping

Use the following table to convert from the memory map address to the location in the physical FlexCAN RAM (where pairs of values are provided, the first is the address for MCR[FDEN] negated, the second is for MCR[FDEN] asserted):

| RAM contents | Memory map | Injection address | | | | |
|---|---|---|---|---|---|---|
| — | — | FlexCAN0 | FlexCAN0 | FlexCAN1/2 | FlexCAN3 | FlexCAN4/5 |
| | | MCXE31B | MCXE317 | MCXE31 | MCXE31B | MCXE31B |
| | | | | | MCXE315/ MCXE316 | MCXE317 |
| | | | | | MCXE317 | |
| FlexCAN Registers | — | Not mapped | Not mapped | Not mapped | Not mapped | Not mapped |
| MBs | 0080h | 0000h | 0000h | 0000h | 0000h | 0000h |
| RXIMRs | 0880h | 600h | 400h | 400h | 200h | 200h |
| RXIFR_0 | 0A80h | 780h | 500h | 500h | 280h | 280h |
| RXIFR_1 | 0A84h | 784h | 504h | 504h | 284h | 284h |
| RXIFR_2 | 0A88h | 788h | 508h | 508h | 288h | 288h |

*Table continues on the next page...*

*Table continued from the previous page...*

| RAM contents | Memory map | Injection address | | | | |
|---|---|---|---|---|---|---|
| RXIFR_3 | 0A8Ch | 78Ch | 50Ch | 50Ch | 28Ch | 28Ch |
| RXIFR_4 | 0A90h | 790h | 510h | 510h | 290h | 290h |
| RXIFR_5 | 0A94h | 794h | 514h | 514h | 294h | 294h |
| Reserved | 0A98h | — | — | — | — | — |
| RXMGMASK | 0AA0h | 7A0h | 520h | 520h | 2A0h | 2A0h |
| RXFGMASK | 0AA4h | 7A4h | 524h | 524h | 2A4h | 2A4h |
| RX14MASK | 0AA8h | 7A8h | 528h | 528h | 2A8h | 2A8h |
| RX15MASK | 0AACh | 7ACh | 52Ch | 52Ch | 2ACh | 2Ach |
| Tx_SMB | 0AB0h/0F28h | 7B0h | 530h | 530h | 2B0h | 2B0h |
| Rx_SMB0 | 0AC0h/0F70h | 7C0h/7F8h | 540h/578h | 540h/578h | 2C0h/2F8h | 2C0h/2F8h |
| Rx_SMB1 | 0AD0h/0FB8h | 7D0h/840h | 550h/5C0h | 550h/5C0h | 2D0h/340h | 2D0h/340h |
| Rx_SMB0_TIME_STAMP | 0C20h | 888h | 608h | 608h | 388h | 388h |
| Rx_SMB1_TIME_STAMP | 0C24h | 88Ch | 60Ch | 60Ch | 38Ch | 38Ch |
| HR_TIME_STAMP | 0C30h | 890h | 610h | 610h | 390h | 390h |
| Enhanced RX FIFO | 2000h | A10h | 710h | — | — | — |
| ERFFEL | 3000h | 1050h | D50h | — | — | — |

### FlexCAN memory initialization

All FlexCAN memory must be initialized before starting its operation in order to have the parity bits in memory properly updated. CTRL2[WRMFRZ] grants write access to all memory positions that require initialization.

These locations are as listed in this table from 080h–ADFh and from C20h–31FFh.:

| Chip | Instance | Offset address ranges to be initialized |
|---|---|---|
| MCXE31B | FlexCAN0 | 080h–ADFh; C20h–31FFh |
| | FlexCAN1–FlexCAN5 | 080h–ADFh; C20h–FFFh |
| MCXE317 | FlexCAN0 | 080h–DFh; C20h–31FFh |
| | FlexCAN1–FlexCAN5 | 080h–ADFh; C20h–FFFh |
| MCXE315/MCXE316 | FlexCAN0 | 080h–ADFh; C20h–31FFh |
| | FlexCAN1–FlexCAN2 | 080h–ADFh; C20h–FFFh |

**NOTE**

The reset value of CAN0_CTRL2 register for MCXE317 is 0080_0000h.

### 70.1.3 Reset value of MDIS bit

The CAN_MCR[MDIS] bit is set to 1 when the module is reset. Therefore, FlexCAN module is disabled following a reset. For details of CAN_MCR[MDIS] register refer to "CAN register description" section.

### 70.1.4 CAN Timestamp Implementation

FlexCAN uses Ethernet Media Access Controller (EMAC) and System Timer Module (STM_0) counters as time source. For details see 'FlexCAN timestamp implementation' section in Clocking chapter.

### 70.1.5 CAN clock setup

For clock setup, see Clocking chapter.

## 70.2 Overview

FlexCAN is a communication controller implementing the CAN protocol according to the ISO 11898-1:2015 standard and CAN 2.0 Part B protocol specifications.

The CAN protocol was primarily designed to be used as a vehicle serial data bus, meeting the specific real-time processing and reliable operation requirements in the electromagnetic interference (EMI) environment of a vehicle. FlexCAN is a full implementation of the CAN protocol specification, the CAN with flexible data rate (CAN FD) protocol, and the CAN version 2.0 Part B protocol. It supports both standard and extended message frames and long payloads.

**NOTE**

Legacy Receive (RX) FIFO cannot be used in CAN Flexible Data (FD) mode. In CAN FD mode, use the Enhanced Receive FIFO feature instead of the Legacy Receive FIFO.

### 70.2.1 Block diagram

Figure 370 shows the main submodules implemented in FlexCAN.

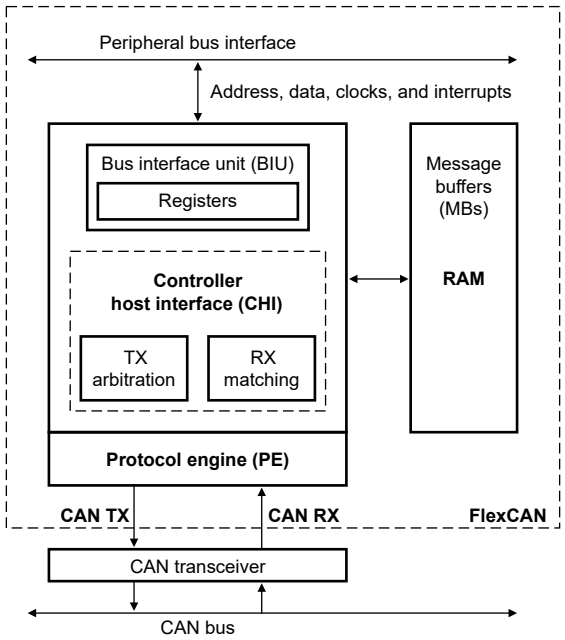



Figure 370. Block diagram

The Protocol Engine (PE) submodule manages the serial communication on the CAN bus:

- RAM access for receiving and transmitting message frames

- Receive message validation
- Error handling
- CAN FD message detection

Controller Host Interface (CHI) manages:

- Message buffer (MB) selection for reception and transmission
- Arbitration and ID matching algorithms for both CAN FD and non-CAN FD message formats

Bus Interface Unit (BIU) controls access to and from the internal interface bus to establish connection to the CPU and to other blocks. The BIU manages access to:

- Clocks
- Address and data buses
- Interrupt outputs
- DMA
- Test signals

### 70.2.2 Features

- Full implementation of *CAN with Flexible data rate (CAN FD) protocol specification* and *CAN Specification Version 2.0, Part B*:
  - Standard data frames
  - Extended data frames
  - Data length of 0–64 bytes
  - Programmable bit rate (see chip-specific FlexCAN information for specific maximum rate configuration)
  - Content-related addressing
- Compliance with ISO 11898-1:2015 standard
- Flexible message buffers that can be configured to store a payload of 0, 8, 16, 32, or 64 bytes
  - Increasing the payload size decreases the number of supported message buffers (see FlexCAN memory partition for CAN FD).
  - Message buffers are configurable as receive or transmit, supporting standard and extended messages.
- Individual Receive Mask registers for each message buffer
- Full-featured Legacy RX FIFO with storage capacity for six frames and automatic internal pointer handling with DMA support
- Full-featured Enhanced RX FIFO with storage capacity of 20 CAN FD frames and automatic internal pointer handling with DMA support
- Transmission abort capability
- Optional general purpose RAM space, using RAM not used by reception or transmission structures
- Listen-Only mode
- Programmable loopback mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number, or highest priority
- Timestamp based on 32-bit free-running timer, with optional external time tick
- Global network time, synchronized by specific message
- Maskable interrupts

- Independence from transmission medium (external transceiver is assumed)
- Short latency time due to arbitration scheme for high-priority messages
- Low-power mode
- Transceiver delay compensation when transmitting CAN FD messages at faster data rates
- Management of remote request frames, automatically or by software
- Restriction only to write CAN bit time settings and configuration bits in Freeze mode
- Transmit message buffer status (lowest-priority buffer or empty buffer)
- Identifier Acceptance Filter Hit Indicator (IDHIT) register for received frames
- ESR1[SYNCH] to indicate module is synchronous with CAN bus
- CRC status for transmitted message
- Legacy RX FIFO Global Mask register
- Selectable priority between message buffers and receive FIFO during matching process
- Powerful legacy RX FIFO ID filtering, capable of matching incoming IDs against either 128 extended IDs, 256 standard IDs, or 512 partial (8-bit) IDs, with 32 individual masking capability
- Powerful enhanced RX FIFO ID filtering, capable of matching incoming IDs against either 64 extended or 128 standard ID filter elements with three filtering schemes: mask plus filter, range, and two filters without mask
- Complete backward compatibility with previous FlexCAN version
- Detection and correction of errors in memory read accesses:
  - Each byte of FlexCAN memory is associated to five parity bits.
  - The error correction mechanism ensures that errors in one bit of this 13-bit word can be corrected (correctable errors).
  - Errors in two bits can be detected but not corrected (noncorrectable errors).

## 70.3 Functional description

FlexCAN is a CAN protocol engine with a flexible message buffer system for transmitting and receiving CAN frames. The system is a set of message buffers (MBs) that stores configuration and control data, timestamp, message ID, and data (see Message buffer structure). The memory corresponding to the first 38 message buffers can be configured to support a Legacy FIFO reception scheme with a powerful ID filtering mechanism. This scheme can check incoming frames against a table of IDs (up to 128 extended IDs or 256 standard IDs or 512 8-bit ID slices), with an individual mask register for up to 32 ID filter table elements.

For classical CAN frames, FlexCAN supports simultaneous reception through Legacy FIFO and message buffer. For CAN FD frames, FlexCAN supports reception through message buffer and enhanced receive FIFO.

For message buffer reception, a matching algorithm makes it possible to store received frames only into MBs that have the same ID programmed in the ID field. A masking scheme makes it possible to match the ID programmed on the message buffer with a range of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of message buffers to be transmitted based on the message ID (optionally augmented by three local priority bits) or the message buffer ordering.

A message buffer is active at a given time if it can participate in both the matching and arbitration processes. A receive message buffer with a 0b code is inactive (see Table 453). A transmit message buffer with a 1000b or 1001b code is also inactive (see Table 454).

FlexCAN can receive and transmit messages in CAN FD format. The message buffers are sized to store the quantity of data bytes selected by FDCTRL[MBDSRn]. The quantity of FD message buffers available for a given quantity of data bytes is described in the CAN FD Control (FDCTRL) register. See also FlexCAN memory partition for CAN FD.

### 70.3.1 Modes of operation

FlexCAN has these functional modes:

**Table 418. Functional modes**

| Mode | Description |
|---|---|
| Normal (User or Supervisor) | In Normal mode, FlexCAN receives and transmits message frames, manages errors normally, and enables all CAN Protocol functions. User and Supervisor modes differ in the access to some restricted control registers. |
| Freeze | Freeze mode is enabled when MCR[FRZ] = 1. If enabled, FlexCAN enters Freeze mode when MCR[HALT] is 1 or when is requested at chip level and FlexCAN writes 1 to MCR[FRZACK]. In this mode, no transmission or reception of frames is done, and synchronicity to the CAN bus is lost. See Freeze mode. |
| Loopback | FlexCAN enters this mode when CTRL1[LPB] becomes 1. In this mode, FlexCAN performs an internal loopback that can be used for self-test. The bit stream output of the transmitter is internally fed back to the receiver input. The receiving CAN input pin is ignored and the transmitting CAN output goes to the recessive state (logic 1). FlexCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. To ensure proper reception of its own message, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field. Both transmit and receive interrupts are generated. |
| Listen-Only | FlexCAN enters this mode when CTRL1[LOM] becomes 1. In this mode, transmission is disabled, all error counters are frozen, and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station are received. If FlexCAN detects an unacknowledged message, it flags a BIT0 error (without changing the REC), as if it was trying to acknowledge the message. |
| CAN FD Active | In this mode, FlexCAN can transmit and receive all messages formatted according to the CAN FD Standard (2.0) and 2.0B Protocol in an interleaved fashion. The CPU can put FlexCAN into CAN FD Active mode by configuring MCR[FDEN] in Freeze mode. |

Some features available in classical CAN are unavailable in CAN FD mode.

**Table 419. Differences between classical CAN and CAN FD**

| Feature | Classical CAN | CAN FD |
|---|---|---|
| Legacy RX FIFO DMA | Yes | No |
| Legacy RX FIFO | Yes | No |
| Enhanced RX FIFO DMA | Yes | Yes |
| Enhanced RX FIFO | Yes | Yes |

FlexCAN can operate in these low-power modes:

**Table 420. Low-power modes**

| Mode | Description |
|---|---|
| Module Disable | FlexCAN enters this mode when the CPU writes 1 to MCR[MDIS] and FlexCAN writes 1 to MCR[LPMACK]. After FlexCAN is disabled, it issues a request to disable the clocks to the CAN Protocol Engine and Controller Host Interface submodules. Writing 0 to MCR[MDIS] exits this mode. See Module Disable mode. |

#### 70.3.1.1 Modes of operation details

FlexCAN has functional modes and low-power modes. See Modes of operation for an introductory description of all the modes of operation. The following subsections contain functional details about Freeze mode and low-power modes.

**CAUTION**

FlexCAN does not support "Permanent Dominant" failure on the CAN bus line. If a Low-Power request or Freeze mode request occurs during a "Permanent Dominant" condition, the corresponding acknowledgment field can never be 1.

##### 70.3.1.1.1 Freeze mode

This mode is requested either by the CPU writing 1 to MCR[HALT] or when the chip is put into Debug mode. MCR[FRZ] must be 1 and the module must not be in a low-power mode.

When MECR[NCEFAFRZ] becomes 1 and a noncorrectable error is detected in a memory read access performed by FlexCAN internal processes (see Error response), FlexCAN also requests Freeze mode. This request occurs via both MCR[HALT] and MCR[FRZ] automatically becoming 1.

To obtain acknowledgment, FlexCAN writes 1 to MCR[FRZACK]. The CPU must only consider FlexCAN to be in Freeze mode when both request and acknowledgment conditions are satisfied.

When Freeze mode is requested, FlexCAN:

1. Waits to be in either Intermission, Error Passive, Bus Off, or Idle state.
2. Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in does not prevent entering Freeze mode.
3. Ignores the receive input pin and drives the transmit pin as recessive.
4. Stops the prescaler, halting all CAN protocol activities.
5. Grants write access to the Error Counters register, which is read-only in other modes.
6. Writes 1 to MCR[NOTRDY] and MCR[FRZACK].

After requesting Freeze mode, you must wait for MCR[FRZACK] to become 1 before executing any other action, otherwise FlexCAN may operate unpredictably. In Freeze mode, all memory-mapped registers are accessible.

Freeze mode is exited in one of these conditions:

- CPU writes 0 to MCR[FRZ].
- The chip is removed from Debug mode or the MCR[HALT] becomes 0.

MCR[FRZACK] becomes 0 after the protocol engine recognizes the negation of the freeze request. After leaving Freeze mode, FlexCAN tries to resynchronize to the CAN bus by waiting for 11 consecutive recessive bits.

##### 70.3.1.1.2 Module Disable mode

This low-power mode is normally used to disable a complete FlexCAN block temporarily, with no power consumption. The CPU requests this mode by writing 1 to MCR[MDIS], and FlexCAN acknowledges the request by writing 1 to MCR[LPMACK]. The CPU must only consider FlexCAN to be in Disable mode when both the request and acknowledgment conditions are satisfied.

If FlexCAN is disabled during Freeze mode, the module requests to disable the clocks to the PE and CHI submodules, writes 1 to MCR[LPMACK] and writes 0 to MCR[FRZACK].

If the module is disabled during transmission or reception, FlexCAN:

1. Waits to be in either Idle or Bus Off state, or waits for the third bit of Intermission and then checks that it is recessive.
2. Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. FlexCAN does not take a pending move-in into account.
3. Ignores its receive input pin and drives its transmit pin as recessive.

4. Shuts down the clocks to the PE and CHI submodules.
5. Writes 1 to MCR[NOTRDY] and MCR[LPMACK].

In this mode, the Bus Interface Unit continues to operate, enabling the CPU to access memory-mapped registers, except for:

- The Receive Message Buffers Global Mask registers
- The Receive Buffer 14 Mask register
- The Receive Buffer 15 Mask register
- The Legacy Receive FIFO Global Mask register

When in Disable mode, these items may not be accessed:

- The Legacy Receive FIFO Information register
- The message buffers
- The Receive Individual Mask registers
- The reserved words within RAM

To exit this mode, the CPU writes 0 to MCR[MDIS], causing FlexCAN to request to resume the clocks and write 0 to MCR[LPMACK]. This write occurs after the CAN protocol engine recognizes the negation of disable mode requested by the CPU.

### 70.3.2 Transmission process

**NOTE**

Instances of MB_CS in this topic refer to items in message buffers. See Message buffer structure for details.

To transmit a CAN frame, the CPU must prepare a message buffer for transmission by executing the following procedure:

1. Check whether the respective interrupt flag is set, and clear it if necessary.
2. If the message buffer is active (transmission pending), request an abort of the transmission. Write the ABORT code (1001b) to the CODE field of the Control and Status word.
3. Poll the IFLAG register until the corresponding IFLAG flag is set, or wait for the interrupt request (if enabled by the respective IMASK field).
4. Read the CODE field to identify whether the transmission was aborted or transmitted (see Transmission abort mechanism).
5. Clear the corresponding interrupt flag.
6. Write the ID register (containing the local priority if enabled via MCR[LPRIOEN]).
7. Write payload data bytes.
8. Configure the Control and Status word as needed.
    a. Set ID type via MB_CS[IDE].
    b. Set Remote Transmission Request (if needed) via MB_CS[RTR].
    c. If MCR[FDEN] = 1, configure MB_CS[EDL] and MB_CS[BRS]. For details about the relationship between the written value and transmitted value of MB_CS[ESI], see Table 431.[7]
    d. Configure Data Length Code in bytes via MB_CS[DLC]. See Table 457 for detailed information.
    e. To transmit the CAN frame, activate the message buffer by writing Ch to MB_CS[CODE].

[7] There is no need to write the ESI field; it is automatically transmitted as dominant by error active nodes and as recessive by error passive nodes. If FlexCAN is operating as a network gateway, however, the CPU writes MB_CS[ESI] according to the error status of the node that sent the message.

**NOTE**

To maximize software performance, configure all the fields in the MB_CS word in a single 32-bit write operation. If the fields are configured in separate writes, MB_CS[CODE] must be the last write in the Control and Status word.

When the MB is activated, it participates in the arbitration process and its contents are eventually transmitted according to its priority. When the DLC value stored in the MB selected for transmission exceeds the MB payload size, FlexCAN completes the expected DLC by adding a constant CCh pattern.

After a successful transmission:

1. The value of the free-running timer is written into the timestamp field.
2. The CODE field in the Control and Status word is updated.
3. Both Cyclic Redundancy Check (CRCR) and CAN FD CRC (FDCRC) are updated.
4. A status flag is set in the Interrupt Flag register.
5. If allowed by the corresponding Interrupt Mask Register field, an interrupt is generated.

After transmission, the new CODE field depends on the code used to activate the message buffer (see Table 453 and Table 454 in Message buffer structure).

When the Abort feature is enabled (MCR[AEN] is 1), after the Interrupt flag is set for an MB configured as transmit buffer, the message buffer is blocked. The CPU cannot update the message buffer until the Interrupt Flag is cleared by the CPU. The CPU must clear the corresponding IFLAG flag before preparing this MB for a new transmission or reception.

**NOTE**

For backward compatibility (MCR[AEN] is 0), write the INACTIVE code (1000b) to the CODE field to deactivate the MB. In this case, the pending frame may be transmitted without notification (see Message buffer inactivation).

### 70.3.3 Arbitration process

The arbitration process scans the message buffers, searching for the transmission MB that holds the message to be sent at the next opportunity. This MB is called the arbitration winner. The scan starts from the lowest number MB and continues to the higher ones.

The arbitration process is triggered when at least one of the following occur:

- CRC field of the CAN frame arrives. The starting point depends on the value of CTRL2[TASD].
- Error Delimiter field of a CAN frame is in progress.
- Overload Delimiter field of a CAN frame is in progress.
- The winner of the arbitration is deactivated, and the CAN bus has not reached the first bit of the Intermission field.
- CPU writes to the Control and Status word of a winner MB, and the CAN bus has not reached the first bit of the Intermission field.
- CHI is in the Idle state and the CPU writes to the Control and Status word of any message buffer.
- FlexCAN exits Bus Off state.
- FlexCAN leaves Freeze mode or a low-power mode.

If the arbitration process does not evaluate all message buffers before the CAN bus reaches the first bit of the Intermission field, the temporary arbitration winner is invalidated. FlexCAN will not compete for the CAN bus at the next opportunity.

The arbitration process selects the winner among the active transmission message buffers at the end of the scan according to the values of CTRL1[LBUF] and MCR[LPRIOEN].

See Arbitration process (continued) for more information about this process.

#### 70.3.3.1 Lowest-number message buffer first

If CTRL1[LBUF] is 1, the first (lowest number) active transmission message buffer found is the arbitration winner. MCR[LPRIOEN] has no effect when CTRL1[LBUF] is 1.

#### 70.3.3.2 Highest-priority message buffer first

If CTRL1[LBUF] is 0, the arbitration process searches for the active transmission message buffer with the highest priority. The frame of this message buffer has a higher probability of winning the arbitration on the CAN bus when multiple external nodes compete for the bus simultaneously.

The sequence of bits considered for this arbitration is called the arbitration value of the message buffer. The transmission message buffer with the lowest arbitration value among all transmission message buffers has the highest priority.

If two or more message buffers have equivalent arbitration values, the message buffer with the lowest number is the arbitration winner.

The composition of the arbitration value depends on MCR[LPRIOEN].

##### 70.3.3.2.1 Local priority disabled

If MCR[LPRIOEN] = 0, the arbitration value is built using the exact sequence of bits that would be transmitted in a CAN frame where local priority is disabled.

**Table 421. Composition of the arbitration value when local priority is disabled**

| Format | Message buffer arbitration value (32 bits) | | | | |
|---|---|---|---|---|---|
| Standard (IDE = 0) | Standard ID (11 bits) | RTR (1 bit) | IDE (1 bit) | — (18 bits) | — (1 bit) |
| Extended (IDE = 1) | Extended ID[28:18] (11 bits) | SRR (1 bit) | IDE (1 bit) | Extended ID[17:0] (18 bits) | RTR (1 bit) |

##### 70.3.3.2.2 Local priority enabled

To enable local priority, MCR[LPRIOEN] must be 1. In this case, the message buffer PRIO field (see Message buffer structure) is included at the very left of the arbitration value.

**Table 422. Composition of the arbitration value when local priority is enabled**

| Format | Message buffer arbitration value (35 bits) | | | | | |
|---|---|---|---|---|---|---|
| Standard (IDE = 0) | PRIO (3 bits) | Standard ID (11 bits) | RTR (1 bit) | IDE (1 bit) | — (18 bits) | — (1 bit) |
| Extended (IDE = 1) | PRIO (3 bits) | Extended ID[28:18] (11 bits) | SRR (1 bit) | IDE (1 bit) | Extended ID[17:0] (18 bits) | RTR (1 bit) |

Because the PRIO field is the most significant part of the arbitration value, message buffers with low PRIO values have higher priority than message buffers with high PRIO values. This priority is maintained regardless of the rest of their arbitration values.

The PRIO field is not part of the frame on the CAN bus. Its purpose is only to affect the internal arbitration process.

#### 70.3.3.3 Arbitration process (continued)

After the arbitration winner is found (see Arbitration process), its content is copied to a hidden auxiliary message buffer called a transmit serial message buffer (TX SMB). The TX SMB has the same structure as a normal message buffer, but is not user-accessible. This copy operation is called move-out. After it is done, write access to the Control and Status word of the corresponding MB is blocked (if MCR[AEN] = 1). Write access is restored in one of the following events:

- The CPU clears the corresponding IFLAG flag after the message buffer is transmitted.
- FlexCAN enters Freeze mode or Bus Off state.
- FlexCAN loses the bus arbitration, or there is an error during the transmission.

At the first opportunity window on the CAN bus, the message on the TX SMB is transmitted according to the CAN protocol rules.

The arbitration process can be triggered under the following conditions:

- During RX and TX frames from CAN CRC field to end of frame. The value of CTRL2[TASD] may be changed to optimize the arbitration start point.
- During CAN Bus Off state from TX_ERR_CNT = 124 to 128. The value of CTRL2[TASD] may be changed to optimize the arbitration start point.
- During Control and Status write by CPU in Bus Idle. The first Control and Status write starts the arbitration process, and a second Control and Status write during this same arbitration restarts the process. If other Control and Status writes are performed, the transmission arbitration process is pending. If there is no arbitration winner after the arbitration process has finished, then the TX arbitration machine begins a new arbitration process. If there is a pending arbitration and Bus Idle state starts, then an arbitration process is triggered. In this case, the first and second Control and Status writes in Bus Idle do not restart the arbitration process. If there is not enough time to finish arbitration in the Wait for Bus Idle state and the next state is Idle, the scan is not interrupted. That scan is completed during Bus Idle state. During this arbitration, a Control and Status write does not cause an arbitration restart.
- Deactivation of an arbitration winner during a valid arbitration window.
- Upon exiting Freeze mode (first bit of the Wait for Bus Idle state). If a resynchronization occurs during the Wait for Bus Idle state, the arbitration process is restarted.

The arbitration process stops when:

- All message buffers are scanned.
- A transmission message buffer is found (if lowest buffer feature is enabled).
- An arbitration winner deactivates or aborts during any arbitration process.
- There is not enough time to finish the transmission arbitration process (for instance, when a deactivation is performed near the end of frame). In this case, the arbitration process is pending.
- An error or overload flag occurs in the bus.
- A low-power or Freeze mode request occurs in Idle state.

Arbitration is considered pending when:

- It is not possible to finish arbitration process in time.
- A Control and Status write occurs during arbitration, when that write is performed in an MB whose number is lower than the transmission arbitration pointer.
- Any Control and Status write occurs when there is no transmission arbitration process in progress.
- RX Match has updated an RX code to TX code.
- FlexCAN enters the Bus Off state.

If a Control and Status write is performed in the arbitration winner, a new process is restarted immediately.

If a Control and Status write is performed in an MB whose number is higher than the transmission arbitration pointer, the ongoing arbitration process scans this MB as normal.

### 70.3.4 Receive process

To be able to receive CAN frames into a message buffer, the CPU must prepare it for reception by executing the following steps:

1. If the message buffer is active (either TX or RX), deactivate it (see Message buffer inactivation), preferably with a safe deactivation (see Transmission abort mechanism).

2. Write the ID word into the message buffer.
3. To activate the message buffer, write the EMPTY code (0100b) to the CODE field of the Control and Status word. No setup is required for EDL, BRS, and ESI fields. The respective fields in the received message overwrite these fields.

After the MB is activated, it can receive frames that match the programmed filter. At the end of a successful reception, the move-in process updates the message buffer (see Move-in) as follows:

1. The received data field (up to eight bytes for Classical CAN message format and up to 64 bytes for CAN FD message format) is stored.
2. The received Identifier field is stored.
3. The value of the free-running timer when the second bit of the Identifier field of the frame is written into the Timestamp field of the MB.
4. The received SRR, IDE, RTR, EDL, BRS, ESI, and DLC fields are stored.
5. The CODE field in the Control and Status word is updated (see Table 453 and Table 454 in Section Message buffer structure).
6. If allowed by the corresponding Interrupt Mask field, a status flag is set in the Interrupt Flag register and an interrupt is generated.

The recommended way for the CPU to service (read) the frame received in a message buffer is:

1. Read the Control and Status word of that message buffer.
2. Verify that the BUSY bit is 0, indicating that the message buffer is locked. If it is not 0, repeat step 1 until it becomes 0. See Message buffer lock mechanism.
3. Read the contents of the message buffer. After the message buffer is locked, FlexCAN move-in processes do not modify its contents. See Move-in.
4. Acknowledge the proper flag in the IFLAG registers.
5. Unlock the message buffer by reading the free-running timer.

To verify frame reception, the CPU should poll the status flag bit for that specific message buffer in one of the IFLAG registers, not the CODE field of that message buffer. Polling the CODE field does not work in this case. After a frame is received and the CPU services the message buffer (by reading the Control and Status word and unlocking the message buffer), the CODE field does not return to EMPTY. It remains FULL, as explained in Table 453. If the CPU tries to work around this behavior by writing to the Control and Status word to force an EMPTY code after reading the message buffer without a prior safe deactivation, a newly received frame matching the filter of that message buffer may be lost.

**CAUTION**

In summary: never poll by reading the Control and Status word of the message buffers directly. Instead, read the IFLAG registers.

The Identifier field of the received frame is always stored in the matching message buffer. If the match was due to masking, the contents of the ID field in a message buffer may change. When MCR[SRXDIS] becomes 1, FlexCAN does not store frames transmitted by itself in any MB, even if it contains a matching receive MB. Also, no interrupt flag or interrupt signal is generated. Otherwise, when MCR[SRXDIS] becomes 0, if a matching receive MB exists, FlexCAN can receive frames transmitted by itself.

To be able to receive CAN frames through the Legacy RX FIFO, the CPU must enable and configure the Legacy RX FIFO during Freeze mode (see Legacy RX FIFO). Upon receiving the Frames Available in the Legacy RX FIFO interrupt (see MCR[BUF5I]), the CPU must service the received frame using the following procedure:

1. If a mask was used for IDE and RTR bits, read the Control and Status word.
2. If a mask was used, read the ID field.
3. Read the data field.
4. If the Identifier Acceptance Filter Hit Indicator (IDHIT) is required in the application, read Legacy RX FIFO Information (RXFIR).

5. Clear the Frames Available in Legacy RX FIFO interrupt by writing one to IFLAG1[BUF5I]. This step is mandatory; it releases the MB and allows the CPU to read the next RX FIFO entry.

When MCR[DMA]) becomes 1, upon receiving a frame in the Legacy FIFO, IFLAG1[BUF5I] generates a DMA request and does not generate a CPU interrupt (see Legacy RX FIFO in DMA operation). The IMASK1 fields in the Legacy RX FIFO region are not used.

The DMA controller must service the received frame using the following procedure:

1. Read the Control and Status word (read 80h address, optional).
2. Read the ID field (read 84h address, optional).
3. Read all data bytes (start read at 88h address, optional).
4. Read the last data bytes (read 8Ch address, mandatory).

### 70.3.5 Matching process

The matching process scans the message buffer memory for RX MBs programmed with the same ID as the one received from the CAN bus. If the Legacy RX or Enhanced RX FIFO is enabled, the priority of scanning can be selected between MBs and FIFO filters. The matching starts from the lowest number message buffer and continues toward the higher-numbered ones. If no match is found within the first structure, the other is scanned afterward. If the FIFO is full, the matching algorithm always looks for a matching MB outside the FIFO region.

For enhanced RX FIFO, see Enhanced RX FIFO.

For legacy RX FIFO, see Legacy RX FIFO.

As the frame is received, it is stored in a hidden auxiliary MB called Receive Serial Message Buffer (RX SMB).

The starting point of the matching process depends on the following conditions:

- If the received frame is a remote frame, the starting point is the CRC field of the frame.
- If the received frame is a data frame with DLC field equal to zero, the starting point is the CRC field of the frame.
- If the received frame is a data frame and the DLC field has a nonzero value, the starting point is the DATA field of the frame.

If a matching ID is found in the FIFO table or in one of the message buffers, the move-in process transfers the contents of the RX SMB to the FIFO or to the matched MB. If any CAN protocol error is detected, no match results are transferred to the FIFO or to the matched MB at the end of reception.

The matching process scans all matching elements of the RX FIFO (if enabled) and active receive MBs (CODE is EMPTY, FULL, OVERRUN, or RANSWER). The process searches for a successful comparison to the matching elements of the RX SMB that is receiving the frame on the CAN bus. The RX SMB has the same structure as a message buffer. The reception structures (RX FIFO or message buffers) associated with the matching elements that had a successful comparison are the matched structures. The matching winner is selected at the end of the scan among those matched structures. The matching winner depends on conditions described in Table 423.

**Table 423. Matching architecture**

| Structure | SMB[RTR] | CTRL2[RRS] | CTRL2[EACEN] | MB[IDE] | MB[RTR] | MB[ID[1]] | MB[CODE] |
|---|---|---|---|---|---|---|---|
| Message buffer | 0 | — | 0 | cmp[2] | no_cmp[3] | cmp_msk[4] | EMPTY, FULL, or OVERRUN |
| Message buffer | 0 | — | 1 | cmp_msk | cmp_msk | cmp_msk | EMPTY, FULL, or OVERRUN |

*Table continues on the next page...*

Table 423. Matching architecture (continued)

| Structure | SMB[RTR] | CTRL2[RRS] | CTRL2[EACEN] | MB[IDE] | MB[RTR] | MB[ID[1]] | MB[CODE] |
|---|---|---|---|---|---|---|---|
| Message buffer | 1 | 0 | — | cmp | no_cmp | cmp | RANSWER |
| Message buffer | 1 | 1 | 0 | cmp | no_cmp | cmp_msk | EMPTY, FULL, or OVERRUN |
| Message buffer | 1 | 1 | 1 | cmp_msk | cmp_msk | cmp_msk | EMPTY, FULL, or OVERRUN |
| Legacy RX FIFO[5] | — | — | — | cmp_msk | cmp_msk | cmp_msk | — |

1. For message buffer structure, if SMB[IDE] is 1, the ID is 29 bits (ID Standard plus ID Extended). If SMB[IDE] is 0, the ID is 11 bits (ID Standard). For Legacy RX FIFO structure, the ID depends on IDAM.
2. cmp: Compares the RX SMB contents to the MB contents regardless of masks.
3. no_cmp: The RX SMB contents are not compared to the MB contents.
4. cmp_msk: Compares the RX SMB contents to MB contents, accounting for masks.
5. SMB[IDE] and SMB[RTR] are not considered when IDAM is type C.

**NOTE**

For Enhanced RX FIFO, see Enhanced RX FIFO matching process.

A reception structure is free-to-receive when any of the following conditions is satisfied:

- The CODE field of the message buffer is EMPTY.
- The CODE field of the message buffer is either FULL or OVERRUN, and it has already been serviced (the CPU has read the Control and Status word and unlocked it as described in Message buffer lock mechanism).
- The CODE field of the message buffer is either FULL or OVERRUN and an inactivation (see Message buffer inactivation) is performed.
- The Legacy RX FIFO or Enhanced RX FIFO is not full.

The scan order for message buffers and Legacy Receive FIFO is from the matching element with the lowest number to the higher ones.

### 70.3.5.1 Matching priority

MCR[IRMQ] affects the matching winner search for MBs. If the field is 0, the matching winner is the first matched MB whether it is free-to-receive or not. If it is 1, the matching winner is selected according to this priority:

1. The first free-to-receive matched message buffer
2. The last non-free-to-receive matched message buffer

It is possible to select the priority of scan between MBs and Legacy RX FIFO or Enhanced RX FIFO with CTRL2[MRP].

If the selected priority is RX FIFO first:

- If the RX FIFO is a matched structure and is free-to-receive, the RX FIFO is the matching winner regardless of the scan for MBs.
- Otherwise, the matching winner is searched for among MBs as described above.

If the selected priority is MBs first:

- If a free-to-receive matched MB is found, it is the matching winner regardless of the scan for RX FIFO.

- If no matched MB is found, then the matching winner is searched for in the scan for the RX FIFO.
- If both conditions above are not satisfied and a non-free-to-receive matched MB is found, then the matching winner depends on the value of MCR[IRMQ]:
  - If MCR[IRMQ] = 0, the matching winner is the first matched MB.
  - If MCR[IRMQ] = 1, the matching winner is the RX FIFO if it is a free-to-receive matched structure. Otherwise, the matching winner is the last non-free-to-receive matched MB.

**Table 424. Matching possibilities and resulting reception structures**

| RFEN or ERFEN | IRMQ | MRP | Matched in MB | Matched in FIFO | Reception structure | Description |
|---|---|---|---|---|---|---|
| **No FIFO, only MB, match is always MB first** | | | | | | |
| 0 | 0 | X[1] | None[2] | —[3] | None | Frame lost by no match |
| 0 | 0 | X | Free[4] | — | First MB | — |
| 0 | 1 | X | None | — | None | Frame lost by no match |
| 0 | 1 | X | Free | — | First MB | — |
| 0 | 1 | X | Not free | — | Last MB | Overrun |
| **FIFO enabled, no match in FIFO; is as if FIFO does not exist** | | | | | | |
| 1 | 0 | X | None | None[5] | None | Frame lost by no match |
| 1 | 0 | X | Free | None | First MB | — |
| 1 | 1 | X | None | None | None | Frame lost by no match |
| 1 | 1 | X | Free | None | First message buffer | — |
| 1 | 1 | X | Not free | None | Last message buffer | Overrun |
| **FIFO enabled; queue disabled** | | | | | | |
| 1 | 0 | 0 | X | Not full[6] | FIFO | — |
| 1 | 0 | 0 | None | Full[7] | None | Frame lost by FIFO full (FIFO overflow) |
| 1 | 0 | 0 | Free | Full | First MB | — |
| 1 | 0 | 0 | Not free | Full | First MB | |
| 1 | 0 | 1 | None | Not full | FIFO | |
| 1 | 0 | 1 | None | Full | None | Frame lost by FIFO full (FIFO overflow) |
| 1 | 0 | 1 | Free | X | First message buffer | — |

*Table continues on the next page...*

Table 424. Matching possibilities and resulting reception structures (continued)

| RFEN or ERFEN | IRMQ | MRP | Matched in MB | Matched in FIFO | Reception structure | Description |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | Not free | X | First message buffer | Overrun |
| **FIFO enabled; queue enabled** | | | | | | |
| 1 | 1 | 0 | X | Not full | FIFO | — |
| 1 | 1 | 0 | None | Full | None | Frame lost by FIFO full (FIFO overflow) |
| 1 | 1 | 0 | Free | Full | First message buffer | — |
| 1 | 1 | 0 | Not free | Full | Last message buffer | Overrun |
| 1 | 1 | 1 | None | Not full | FIFO | — |
| 1 | 1 | 1 | Free | X | First message buffer | — |
| 1 | 1 | 1 | Not free | Not full | FIFO | — |
| 1 | 1 | 1 | Not free | Full | Last message buffer | Overrun |

1. Indicates a don't care condition.
2. Matched in message buffer "None" means that the frame has not matched any message buffer (free-to-receive or non-free-to-receive).
3. This condition is forbidden.
4. Matched in message buffer "Free" means that the frame matched at least one message buffer free-to-receive regardless of whether it has matched MBs that are non-free-to-receive.
5. Matched in FIFO, "None" means that the frame has not matched any filter in FIFO. It is as if the FIFO did not exist (CTRL2[RFFN] = 0 and ERFCR[ERFEN] = 0).
6. Matched in FIFO, "Not full" means that the frame has matched a FIFO filter and has empty slots to receive it.
7. Matched in FIFO, "Full" means that the frame matched a FIFO filter but could not store it, because it has no empty slots to receive it.

### 70.3.5.2 Special cases

If a non-safe MB inactivation (see Message buffer inactivation) occurs during the matching process and the inactivated MB is the temporary matching winner, the temporary matching winner is invalidated. The matching elements scan is not stopped and not restarted; it continues normally. The consequence is that the current matching process works as if the matching elements compared before the inactivation did not exist. In this case, a message may be lost.

Consider an example where:

- The FIFO is disabled.
- MCR[IRMQ] is 1.
- There are two message buffers with the same ID: the second and fifth MBs in the array.
- FlexCAN starts receiving messages with that ID.

When the first message arrives, the matching algorithm finds the first match in message buffer number 2. The code of this message buffer is EMPTY, so the message is stored in that MB. When the second message arrives, the matching algorithm finds MB number 2 again, but it is not "free-to-receive." It continues looking, finds MB number 5, and stores the message in that MB.

If yet another message with the same ID arrives, the matching algorithm finds no matching free-to-receive MBs, so it overwrites the last matched message buffer (MB number 5). In doing so, it updates the CODE field of the message buffer to OVERRUN.

The ability to match the same ID in more than one MB can be used to implement a reception queue (in addition to the full-featured FIFO) to allow more time for the CPU to service the MBs. By programming more than one MB with the same ID, received messages are queued into the message buffers. The CPU can examine the Timestamp field of the message buffers to determine the order in which the messages arrived.

Matching a range of IDs is possible via ID acceptance masks. FlexCAN supports individual masking per message buffer (see Receive Individual Mask (RXIMR0 - RXIMR95)). During the matching algorithm, if a mask field is 1, the corresponding ID bit is compared. If the mask field is 0, the corresponding ID bit is a "don't care". Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed when the module is in Freeze mode; otherwise, the module blocks them.

FlexCAN also supports an alternate masking scheme with only Legacy RX FIFO Global Mask (RXFGMASK), RX Message Buffers Global Mask (RXMGMASK), Receive 14 Mask (RX14MASK), and Receive 15 Mask (RX15MASK) for backward compatibility with legacy applications. This alternate masking scheme is enabled when the MCR[IRMQ] = 0.

## 70.3.6 Move process

There are two types of move process: move-in and move-out.

### 70.3.6.1 Move-in

The move-in process is the copying of a message received by an RX SMB to an RX message buffer or FIFO that has matched it. If the move destination is the Legacy RX FIFO, attributes of the message are also copied to the CAN_RXFIR FIFO. Each RX SMB has its own move-in process, but only one is performed at a given time. The move-in starts only when the message held by the RX SMB has a corresponding match (see Matching process) and all of the following conditions are true:

- The CAN bus has reached or already gone past:
  - The second bit of Intermission field next to the frame that carried the message that is in the RX SMB.
  - The first bit of an overload frame next to the frame that carried the message in the RX SMB.
- There is no ongoing matching process.
- The CPU is not locking the destination message buffer.
- No move-in process from another RX SMB is ongoing. If more than one move-in process is to be started at the same time, both are performed and the newest process substitutes for the oldest.

The term "pending move-in" is used throughout the documentation and stands for a move-to-be that does not satisfy all of the above conditions.

If any of the following conditions is satisfied, the move-in is canceled and the RX SMB is able to receive another message:

- The destination message buffer is inactivated after the CAN bus has reached the first bit of the Intermission field next to the frame that carried the message. Also, its matching process has finished.
- There is a previous pending move-in to the same destination message buffer.
- The RX SMB is receiving a frame transmitted by FlexCAN itself and self-reception is disabled (MCR[SRXDIS] = 1).
- Any CAN protocol error is detected.

If the module enters Freeze or Low-Power mode, the pending move-in is not canceled. It remains on hold, waiting for Freeze and Low-Power mode to be exited and for the module to be unlocked. If a message buffer is unlocked during Freeze mode, the move-in occurs immediately.

The move-in process is FlexCAN executing the following steps:

1. Push IDHIT into the RXFIR FIFO if the message is destined for the Legacy RX FIFO.
2. Read all data words from the RX SMB in accordance with the selected payload size for the RX storage element.

3. Write all data words to the RX message buffer according to the selected payload size for the RX storage element. If the data size of the storage element is smaller than the original payload size described in the DLC field of the message, the payload is truncated. The high-order bytes that do not fit the destination size are lost.
4. Read the Control and Status and ID words from the RX SMB.
5. Write the Control and Status and ID words to the RX message buffer, and update the CODE field.

The move-in process is not atomic; the inactivation of the destination message buffer immediately cancels it (see Message buffer inactivation). In this case, the message buffer may remain partially updated, and therefore incoherent. When the move-in destination is a Legacy or Enhanced RX FIFO message buffer, however, the process cannot be canceled.

To alert the CPU that the message buffer content is temporarily incoherent, the BUSY Bit (least significant bit of the CODE field) of the destination message buffer becomes 1 during move-in.

#### 70.3.6.2 Move-out

The move-out process is the copying of content from a TX message buffer to the TX SMB when a message for transmission is available (see Arbitration process). The move-out occurs in the following conditions:

- In the first bit of the Intermission field
- During the Bus Off state, when the TX error counter is in the 124 to 128 range
- During the Bus Idle state
- During the Wait for Bus Idle state

The move-out process is not atomic. Only the CPU has priority to access the memory concurrently outside the Bus Idle state. In Bus Idle, the move-out has the lowest priority of the concurrent memory accesses.

### 70.3.7 Data coherence

In order to maintain data coherency and proper FlexCAN operation, the CPU must obey the rules described in Transmission process and Receive process.

#### 70.3.7.1 Transmission abort mechanism

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform the CPU whether the transmission was aborted or the frame could not be aborted and was transmitted instead.

These primary conditions must be fulfilled in order to abort a transmission:

- MCR[AEN] must be 1.
- The first CPU action must be the writing of abort code (1001b) into the CODE field of the Control and Status word.

Active message buffers configured for transmission must be aborted before they can be updated. The write operation is blocked and the transmission is not disturbed when the abort code is written to:

- A message buffer currently being transmitted.
- A message buffer that was already loaded into the TX SMB for transmission.

In this case, the abort request is captured and kept pending until one of the following conditions is satisfied:

- The module loses the bus arbitration.
- There is an error during the transmission.
- The module is put into Freeze mode.
- The module enters the Bus Off state.
- There is an overload frame.

If none of the conditions above are reached:

1. The message buffer is transmitted correctly.
2. The interrupt flag is set in the proper IFLAG register.
3. If enabled, an interrupt to the CPU is generated.

The abort request is automatically cleared when the interrupt flag is set. If only one of the above conditions is reached, the frame is not transmitted. In this case:

1. The abort code is written into the CODE field.
2. The interrupt flag is set in the proper IFLAG register.
3. Optionally, an interrupt is generated to the CPU.

If the CPU writes the abort code before the transmission begins internally, the write operation is not blocked. The MB is updated and the interrupt flag is set. In this way, the CPU only needs to read the abort code to verify that the active MB was safely inactivated. In this case, although MCR[AEN] = 1 and the CPU wrote the abort code, the MB is inactivated and not aborted, because the transmission did not start yet. A message buffer is aborted only when the abort request is captured and kept pending until one of the previous conditions is satisfied.

#### 70.3.7.2 Message buffer inactivation

Inactivation protects the message buffer against updates by FlexCAN internal processes. It allows the CPU to rely on message buffer data coherence after having updated it, even in Normal mode.

Inactivation of transmission message buffers must be performed only when MCR[AEN] = 0.

If a message buffer is inactivated, it does not participate in the arbitration process or the matching process until it is reactivated. See Transmission process and Receive process for detailed instructions on how to inactivate and reactivate a message buffer.

To inactivate a message buffer, the CPU must update its CODE field to INACTIVE (either 0b or 1000b).

Because you cannot synchronize the CODE field update with FlexCAN internal processes, an inactivation can have the following consequences:

- A frame in the bus that matches the filtering of an inactivated RX message buffer may be lost without notice. This loss can occur even if there are other message buffers with the same filter.
- A frame containing the message within an inactivated TX message buffer may be transmitted without setting the respective IFLAG flag.

To perform a safe inactivation and avoid the above consequences for TX message buffers, the CPU must use the transmission abort mechanism (see Transmission abort mechanism).

The inactivation automatically unlocks the message buffer (see Message buffer lock mechanism).

**NOTE**

Message buffers that are part of the Legacy RX FIFO or Enhanced RX FIFO cannot be inactivated. There is no write protection on the Legacy FIFO region by FlexCAN. The CPU must maintain data coherency in the Legacy FIFO region when MCR[RFEN] = 1.

#### 70.3.7.3 Message buffer lock mechanism

In addition to message buffer inactivation, FlexCAN uses a message buffer lock mechanism to maintain data coherence for the receive process. When the CPU reads the Control and Status word of an RX message buffer with codes FULL or OVERRUN, FlexCAN is configured to allow the CPU to read the whole message buffer in an atomic operation. FlexCAN sets an internal lock flag for that message buffer.

The lock is released in any of the following events:

- The CPU reads the free-running timer (global unlock operation).
- The CPU reads the Control and Status word of another message buffer, regardless of its code.

- The CPU writes into the Control and Status word. This procedure is not recommended for normal unlocking, because it cancels a pending move and may lose a received message.

The message buffer lock prevents a new frame from being written into the message buffer when the CPU is reading it.

**NOTE**

The locking mechanism applies only to RX message buffers that are not part of the Legacy RX FIFO and have a code other than INACTIVE (0b) or EMPTY[1] (0100b). TX message buffers cannot be locked.

Consider an example where:

- The Legacy RX FIFO is disabled.
- The second and the fifth message buffers of the array are programmed with the same ID.
- FlexCAN has already received and stored messages into these two message buffers.
- The CPU reads message buffer number 5 while another message with the same ID is arriving.

When the CPU reads the Control and Status word of message buffer number 5, this message buffer is locked. The new message arrives and the matching algorithm finds no free-to-receive message buffers, so it overrides message buffer number 5. This message buffer is locked, so the new message cannot be written to it. The message remains in the RX SMB until the message buffer is unlocked, and only then is it written to the message buffer.

If the message buffer remains locked and another new message with the same ID arrives, the new message overwrites the one in the RX SMB. There is no indication of lost messages in the CODE field of the message buffer or in Error and Status 1 (ESR1).

When the message is moved from the RX SMB to the message buffer, the BUSY bit on the CODE field becomes 1. If the CPU reads the Control and Status word and identifies that the BUSY bit is 1, it must wait until the BUSY bit becomes 0 to access the MB.

**NOTE**

If the BUSY bit is 1 or the message buffer is empty, reading the Control and Status word does not lock the message buffer.

Inactivation takes precedence over locking. If the CPU inactivates a locked receive message buffer, then its lock status is negated and the message buffer is marked as invalid for the current matching round. Any pending message on the RX SMB is not transferred to the message buffer. A message buffer is unlocked when the CPU reads Free-Running Timer (TIMER) or the Control and Status word of another message buffer.

The lock and unlock mechanisms have the same functionality in Normal and Freeze modes.

An unlock during Normal or Freeze mode results in the move-in of the pending message. If unlocking occurs during a low-power mode, however, the move-in is postponed (see Modes of operation). Move-in takes place only when the module returns to Normal or Freeze mode.

### 70.3.8 Enhanced RX FIFO

FlexCAN supports an enhanced RX FIFO engine which can store up to 20 CAN FD messages. The region 2000h–204Fh contains the output of the FIFO, which the CPU should read. To enable the enhanced RX FIFO, write 1 to ERFCR[ERFEN]. FlexCAN has two FIFO options, Legacy RX FIFO and Enhanced RX FIFO, but both options cannot be enabled at the same time. See Legacy RX FIFO for additional information.

To configure the enhanced RX FIFO watermark, write a value to ERFCR[ERFWM]. If ERFCR[ERFWM] is configured, the CPU is notified only if a minimum number of messages is stored in the FIFO. When the number of stored messages is greater than the value in ERFCR[ERFWM], the module sets ERFSR[ERFWMI]. Optionally, if MCR[DMA] or ERFIER[ERFWMIIE] are enabled, a DMA transfer or an interrupt can be triggered, respectively.

For the enhanced RX FIFO to receive, the CPU must execute the configuration procedure below. If the CPU must change any configurations of the Enhanced RX FIFO, the same procedure must be followed.

[1] In previous FlexCAN versions, reading the Control and Status word locked the message buffer even when CODE indicates it is EMPTY. This behavior is maintained when the IRMQ bit is 0.

**Prerequisites**

MCR[RFEN] must be 0.

**Procedure**

| Step | Purpose | Programming | Notes |
|---|---|---|---|
| 1 | Enter Freeze mode. | See Freeze mode. | — |
| 2 | If enhanced RX FIFO is not already enabled, enable it. | Write 1 to ERFCR[ERFEN]. | — |
| 3 | Reset enhanced RX FIFO engine. | Write 1 to ERFSR[ERFCLR]. | — |
| 4 | If the enhanced RX FIFO error flags are set, clear them. | Write 1 to these flags:<br>• ERFSR[ERFUFW]<br>• ERFSR[ERFOVF]<br>• ERFSR[ERFWMI]<br>• ERFSR[ERFDA] | — |
| 5 | Specify the total number of enhanced RX FIFO filter elements to be used in Enhanced RX FIFO reception. | Write the number to ERFCR[NFE]. | — |
| 6 | Specify the number of extended ID and standard ID filter elements to be used in Enhanced RX FIFO reception. | Write the number to ERFCR[NEXIF]. | ERFCR[NEXIF] ≤ ERFCR[NFE] + 1. |
| 7 | If you are using DMA, enable DMA. | Write 1 to MCR[DMA]. | — |
| 8 | If you are using DMA, specify the number of words to transfer for each Enhanced RX FIFO data element. | Write the number to ERFCR[DMALW]. | — |
| 9 | Specify the Enhanced RX FIFO watermark. | Write the number to ERFCR[ERFWM]. | If MCR[DMA] = 1, ERFCR[ERFWM] should be 0h. |
| 10 | If you are using interrupts, enable the interrupts. | Write 1 to the interrupt enables in Enhanced RX FIFO Interrupt Enable (ERFIER). | — |
| 11 | Configure the filter elements. | Write to the ERFFEL*n* registers. | ERFFEL*n* registers are implemented in RAM; you must explicitly initialize them before prior any reception. |
| 12 | Exit Freeze mode. | See Freeze mode. | — |

There are two types of enhanced RX FIFO filter elements that can be stored in ERFFEL*n* registers: extended-ID filter elements and standard-ID filter elements. Each extended-ID filter element is stored in two ERFFEL*n* registers, and each standard-ID filter element is stored in one ERFFEL*n* register. ERFCR[NFE] defines the total number of Enhanced RX FIFO filter elements.

In addition, the filter memory space can be split into two regions: one for extended-ID filter elements and another for standard-ID filter elements, according to ERFCR[NEXIF]. Figure 371 shows how the enhanced RX filter elements are defined. See Enhanced RX FIFO matching process for information about the Enhanced RX FIFO matching process and filter element formats.

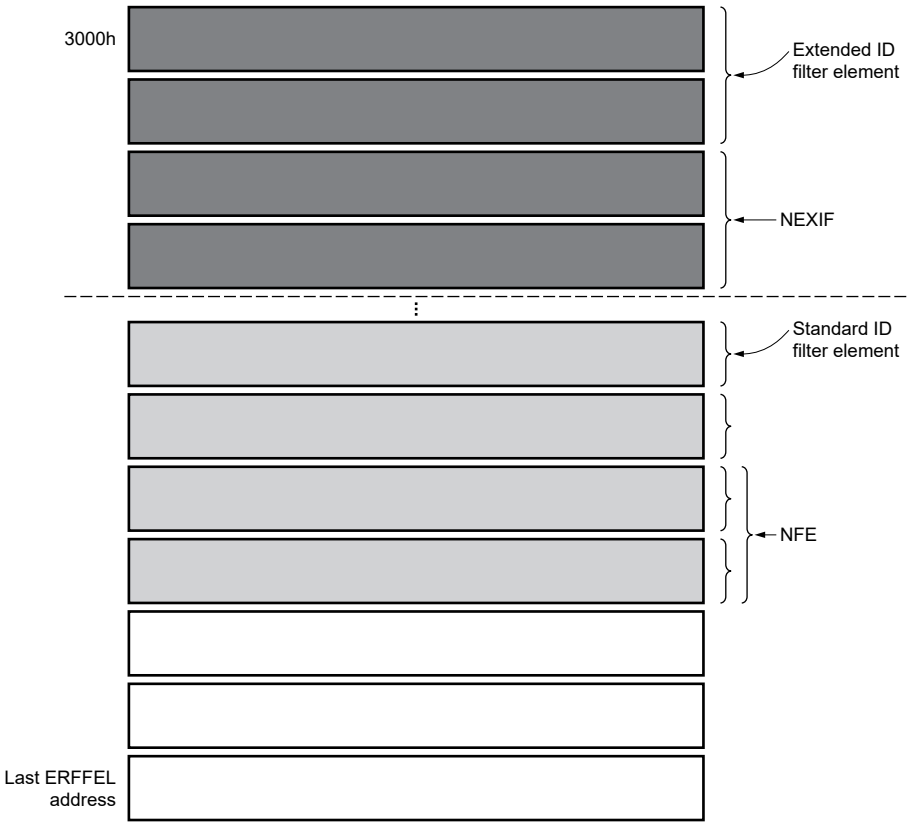



Figure 371. Number of enhanced RX FIFO filter elements

### 70.3.8.1 Enhanced RX FIFO matching process

When ERFCR[ERFEN] = 1, FlexCAN scans the ERFFEL*n* memory region. If at least one filter element satisfies the matching criteria, the CAN message content is transferred to the enhanced RX FIFO memory. If multiple filters match the incoming message ID, the first matching filter found by the matching process must be indicated in IDHIT.

Each ERFFEL*n* register can store one standard filter element. ERFFEL*n*[FEL][31:30], also called FSCH, determines the matching criteria in this way:

- If FSCH = b00, the filter scheme is based on mask and filter. A CAN message matches a standard ID filter element only if these criteria are reached:
  1. CAN message is base-frame format (IDE = 0).
  2. (ID[*n*] = STD ID filter [*n*]) or (STD ID Mask[*n*] = 0) for each bit *n* from 0 to 10.
  3. (RTR = RTR Filter) or (RTR MASK = 0).

In this explanation, RTR and ID are the Remote Transmit Request field and the ID from a CAN message, respectively.

If FSCH = b00, the filters and masks are defined as shown in Table 425.

Table 425. Standard ID filter element with filter and mask scheme (FSCH = b00)

| 31 | 30 | 29 | 28 | 27 | 26 | 16 | 15 | 12 | 11 | 10 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| FSCH = b00 | | Reserved | | RTR Filter | STD ID Filter | | Reserved | | RTR MASK | STD ID MASK | |

- If FSCH = b01, the filter scheme is based on range. A CAN message matches a standard ID filter element only if these criteria are reached:
  1. CAN message is base frame format (IDE = 0).
  2. ID ≥ STD ID Filter1.
  3. ID ≤ STD ID Filter2.
  4. (RTR = RTR filter) or (RTR MASK = 0).

RTR and ID are the Remote Transmit Request bit and ID from a CAN message, respectively. If FSCH = b01, the filters and mask are defined as shown in Table 426.

**Table 426. Standard ID filter element with range scheme (FSCH = b01)**

| 31 | 30 | 29 | 28 | 27 | 26 | 16 | 15 | 12 | 11 | 10 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| FSCH = b01 | | Reserved | | RTR Filter | STD ID Filter2 | | Reserved | | RTR MASK | STD ID Filter1 | |

- If FSCH = b10, the filter scheme is based on two filters without masks. A CAN message matches a standard ID filter element only if these criteria are reached:
  1. CAN message is base frame format (IDE = 0).
  2. (ID[*n*] = STD ID Filter1[*n*]) or (ID[*n*] = STD ID Filter2[*n*]) for each bit *n* from 0 to 10.
  3. (RTR = RTR Filter1) or (RTR = RTR Filter2).

RTR and ID are the Remote Transmit Request bit and ID from a CAN message, respectively. If FSCH = b10, the filters are defined as shown in Table 427.

**Table 427. Standard ID filter element with two-filter scheme (FSCH = b10)**

| 31 | 30 | 29 | 28 | 27 | 26 | 16 | 15 | 12 | 11 | 10 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| FSCH = b10 | | Reserved | | RTR Filter 2 | STD ID Filter2 | | Reserved | | RTR Filter 1 | STD ID Filter1 | |

Each pair of ERFFEL*n* registers can store one extended filter element. ERFFEL*n*[FSCH] determines the matching criteria in this way:

- If FSCH = b00, the filter scheme is based on mask and filter. A CAN message matches an extended ID filter element only if these criteria are reached:
  1. CAN message is extended frame format (IDE = 1).
  2. (ID[*n*] = EXT ID filter [*n*]) or (EXT ID Mask[*n*] = 0) for each bit *n* from 0 to 28.
  3. (RTR = RTR Filter) or (RTR MASK = 0).

If FSCH = b00, the filters and masks are defined as shown in Table 428.

**Table 428. Extended ID filter element with filter + mask scheme (FSCH = b00)**

| 31 | 30 | 29 | 28 | 0 |
|---|---|---|---|---|
| FSCH | | RTR Filter | EXT ID Filter | |
| Reserved | | RTR MASK | EXT ID MASK | |

- If FSCH = b01, the filter scheme is based on range. A CAN message matches an extended ID filter element only if the following criteria are reached:
  1. CAN message is extended frame format (IDE = 1).
  2. ID ≥ EXT ID Filter1.
  3. ID ≤ EXT ID Filter2.
  4. (RTR = RTR Filter) or (RTR MASK = 0).

If FSCH = b01, the filters and masks are defined as shown in Table 429.

**Table 429. Extended ID filter element with range scheme (FSCH = b01)**

| 31 | 30 | 29 | 28 | 0 |
|---|---|---|---|---|
| FSCH | | RTR Filter | EXT ID Filter2 | |
| Reserved | | RTR MASK | EXT ID Filter 1 | |

- If FSCH = b10, the filter scheme is based on two filters without masks. A CAN message matches an extended ID filter element only if these criteria are reached:
  1. CAN message is extended frame format (IDE = 1).
  2. (ID[*n*] = EXT ID Filter1[*n*]) or (ID[*n*] = EXT ID Filter2[*n*]) for each bit *n* from 0 to 28.
  3. (RTR = RTR Filter1) or (RTR = RTR Filter2).

If FSCH = b10, the filters are defined as shown in Table 430.

**Table 430. Extended ID filter element with two-filter scheme (FSCH = b10)**

| 31 | 30 | 29 | 28 | 0 |
|---|---|---|---|---|
| FSCH | | RTR Filter2 | EXT ID Filter2 | |
| Reserved | | RTR Filter1 | EXT ID Filter 1 | |

### 70.3.8.2 Enhanced RX FIFO under DMA operation

You can enable the DMA feature by writing 1 to both ERFCR[ERFEN] and MCR[DMA]. The DMA controller can read the received message by reading a message buffer structure at the enhanced FIFO output port at the address range defined in Enhanced RX FIFO structure.

**NOTE**

FlexCAN supports 32-bit access only for DMA transfers.

For proper FIFO engine operation, the CPU should not access the Enhanced FIFO output port address range during DMA operation. Before writing 1 to MCR[DMA], the CPU must service Enhanced RX FIFO status bits. Otherwise, these bits may show that the FIFO has data to be serviced, and mistakenly generate a DMA request. Before writing 0 to MCR[DMA], the CPU must first clear the ERFSR[ERFUFW], ERFSR[ERFOVF], ERFSR[ERFWMI], and ERFSR[ERFDA] flags. It must then clear the enhanced RX FIFO engine by writing one to ERFSR[ERFCLR].

When there is one frame available to be read from the Enhanced RX FIFO, FlexCAN sets ERFSR[ERFDA]. Upon receiving the request, the DMA controller can read the message in the Enhanced RX FIFO output. Each message reading process must end by the address defined in ERFCR[DMALW].

Follow these rules for Enhanced RX FIFO DMA operation:

- Because a DMA transfer cannot be changed dynamically, program ERFCR[DMALW] so the enhanced RX FIFO element can store the largest CAN message present on the CAN bus.
- Data bytes are valid according to the DLC field. See Table 457.

Each time the DMA controller reads one message from the FIFO, FlexCAN clears ERFSR[ERFDA]. If there is at least one message stored in the FIFO, FlexCAN sets it again.

Consider an example where the maximum number of bytes in the data field of a CAN frame for a certain application is eight, and high-resolution timestamp is enabled. In that case, the last enhanced RX FIFO address offset can be found in Table 467 and Table 468. Using this address offset, ERFCR[DMALW] can be determined in this way:

- Maximum number of data bytes = 8
- HR_TIME_STAMP enabled
- Last address offset = TS_OFF = 2014h
- DMALW = 5

#### 70.3.8.3 Enhanced RX FIFO clear operation

When ERFCR[ERFEN] is 1, the CPU can clear the Enhanced RX FIFO by writing 1 to ERFSR[ERFCLR]. The clear operation resets the internal FIFO pointers, but the FIFO content stored in RAM is not changed. This operation can only be performed in Freeze mode; the module blocks the operation in other modes. This operation does not clear ERFSR[ERFUFW], ERFSR[ERFOVF], ERFSR[ERFDA], or ERFSR[ERFWMI]. The CPU must service all these fields before executing the clear FIFO operation.

### 70.3.9 Legacy RX FIFO

The Legacy RX FIFO is receive-only. To enable it, write 1 to MCR[RFEN]. To maintain software backward compatibility with previous versions of FlexCAN that did not have the Legacy FIFO feature, the reset value of this field is zero.

**CAUTION**

Do not enable Legacy RX FIFO when the CAN FD feature is enabled.

The Legacy FIFO is six messages deep. The memory region the Legacy FIFO structure occupies (both message buffers and Legacy FIFO engine) is described in legacy RX FIFO structure. The CPU can read the received messages sequentially, in the order they were received, by repeatedly reading a message buffer structure at the output of the Legacy FIFO.

IFLAG1[BUF5I] (Frames Available in Legacy RX FIFO) is set when at least one frame is available to be read from the Legacy FIFO. If the corresponding mask bit enables it, an interrupt is generated. Upon receiving the interrupt, the CPU can read the message (accessing the output of the Legacy FIFO as a message buffer) and Legacy RX FIFO Information (RXFIR), then clear the interrupt. If there are more messages in the Legacy FIFO, clearing the interrupt:

1. Updates the output of the Legacy FIFO with the next message.
2. Updates RXFIR with the attributes of that message.
3. Reissues the interrupt to the CPU.

Otherwise, the flag remains cleared. The output of the Legacy FIFO is valid only when IFLAG1[BUF5I] is set.

IFLAG1[BUF6I] (Legacy RX FIFO Warning) is set when the Legacy RX FIFO receives a new message that increases the number of unread messages from four to five. This change means that the Legacy RX FIFO is almost full. The flag remains set until the CPU clears it.

IFLAG1[BUF7I] (Legacy RX FIFO Overflow) is set when an incoming message is lost because the Legacy RX FIFO is full. The flag is not set when the Legacy RX FIFO is full and a message buffer captures the message. The flag remains set until the CPU clears it.

Clearing one of the three flags above does not affect the state of the other two.

If an IFLAG flag is set and the corresponding mask bit is 1, an interrupt is generated.

A powerful filtering scheme is provided to accept only frames intended for the target application, reducing the interrupt servicing workload. The filtering criteria are specified by programming a table of up to 128 32-bit registers, according to CTRL2[RFFN]. This table can be configured to one of the following formats (see also legacy RX FIFO structure):

- Format A: 128 Identifier Acceptance Filters (IDAFs) — extended or standard IDs including IDE and RTR
- Format B: 256 IDAFs — standard IDs or extended 14-bit ID slices including IDE and RTR
- Format C: 512 IDAFs — standard or extended 8-bit ID slices

**NOTE**

A chosen format is applied to all entries of the filter table. It is not possible to mix formats within the table.

Every frame available in the Legacy RX FIFO has a corresponding Identifier Acceptance Filter Hit Indicator (IDHIT). The IDHIT can be read in the IDHIT field in the Control and Status word, as shown in the Legacy RX FIFO Structure description. The CPU can also obtain this information by accessing Legacy RX FIFO Information (RXFIR). RXFIR[IDHIT] refers to the message at the output of the Legacy FIFO, and is valid when IFLAG1[BUF5I] is set. RXFIR must be read only before clearing the flag, guaranteeing that the information refers to the correct frame within the Legacy FIFO.

The Individual Mask Registers (RXIMR*n*) individually affect up to 32 elements of the filter table, according to the value of CTRL2[RFFN]. This configuration allows very powerful filtering criteria to be defined. If MCR[IRMQ] is 0, the Legacy RX FIFO filter table is affected by Legacy RX FIFO Global Mask (RXFGMASK).

**NOTE**

See Table 419 for information about the difference between FD and non-FD regarding this feature.

#### 70.3.9.1 Legacy RX FIFO in DMA operation

The receive-only Legacy FIFO can support DMA. To enable this feature, write 1 to both MCR[RFEN] and MCR[DMA]. To maintain backward compatibility with previous versions of the module that did not have the DMA feature, the reset value of MCR[DMA] is zero.

The DMA controller can read the received message by reading a message buffer structure at the Legacy FIFO output port in the 80h–8Ch address range.

**NOTE**

FlexCAN supports 32-bit access only for DMA transfers.

When MCR[DMA] = 1, the CPU must not access the Legacy FIFO output port address range. Before writing 1 to MCR[DMA], the CPU must service the IFLAG flags set in the Legacy RX FIFO region. Otherwise, these flags may indicate that the FIFO has data to be serviced, and mistakenly generate a DMA request. Before writing 0 to MCR[DMA], the CPU must perform a clear Legacy FIFO operation.

When at least one frame available to be read from the FIFO, IFLAG1[BUF5I] (Frames available in Legacy RX FIFO) is set. A DMA request is generated simultaneously. Upon receiving the request, the DMA controller can read the message (accessing the output of the Legacy FIFO as a message buffer). The DMA reading process must end by reading address 8Ch. This read operation:

- Clears IFLAG1[BUF5I].
- Updates the FIFO output with the next message (if the FIFO is not empty).
- Updates Legacy RX FIFO Information (RXFIR) with the attributes of the new message.

If there are more messages stored in the FIFO, IFLAG1[BUF5I] is reasserted and another DMA request is issued. Otherwise, the flag remains cleared.

**NOTE**

RXFIR contents cannot be read after DMA completes the Legacy FIFO read. The IDHIT information is also available in the Control and Status word at address 080h (see legacy RX FIFO structure).

IFLAG1[BUF6I] and IFLAG1[BUF7I] are not used when the DMA feature is enabled.

When FlexCAN is working with DMA, the CPU does not receive any Legacy RX FIFO interruption and must not clear the related IFLAG flags. The related IMASK bits are not used to mask the generation of DMA requests.

**NOTE**

See Table 419 for information about the difference between FD and non-FD regarding this feature.

#### 70.3.9.2 Clear legacy FIFO

When MCR[RFEN] = 1, you can use the clear Legacy FIFO operation to empty Legacy FIFO contents. When the CPU writes 1 to IFLAG1[BUF0I], the clear FIFO operation occurs. This operation can only be performed in Freeze mode; FlexCAN blocks it in other modes. This operation does not clear the FIFO IFLAG flags; the CPU must service all FIFO IFLAG flags before executing the clear FIFO operation.

When Legacy RX FIFO is working with DMA, the clear FIFO operation clears IFLAG1[BUF5I], and the DMA request is canceled.

**CAUTION**

The clear Legacy FIFO operation does not clear IFLAG flags, except when MCR[DMA] = 1; in this case, only IFLAG1[BUF5I] is cleared.

### 70.3.10 CAN protocol-related features

#### 70.3.10.1 CAN FD ISO compliance

The CAN FD protocol has been improved to increase the failure-detection capability that was in the original CAN FD protocol. This original protocol is also called non-ISO CAN FD, by CAN in Automation (CiA). A three-bit stuff counter and a parity bit have been introduced in the improved CAN FD protocol, now called ISO CAN FD. The CRC calculation has also been modified. All these improvements make the ISO CAN FD protocol incompatible with the non-ISO CAN FD protocol. FlexCAN still supports non-ISO CAN FD, so it can be used during an intermediate phase, for evaluation and development purposes.

It is recommended that you configure FlexCAN with the ISO CAN FD protocol by writing 1 to CTRL2[ISOCANFDEN].

#### 70.3.10.2 CAN FD frames

ISO 11898-1:2015 specifies the Classical Frame format compliant to ISO 11898-1:2003 (2003) and introduces the CAN Flexible Data Rate Frame format (CAN FD). The Classical Frame format allows bit rates up to one Mbit/s and payloads up to eight bytes per frame. The Flexible Data Rate Frame format allows bit rates faster than one Mbit/s and payloads longer than eight bytes per frame. FlexCAN can receive and transmit CAN FD messages interleaved with Classical CAN messages.

There are additional control bits in the CAN FD frame:

- The Extended Data Length (EDL) bit enables a longer data payload with different data length coding.
- The Bit Rate Switch (BRS) bit decides whether the bit rate is switched inside a CAN FD format frame.
- The Error State Indicator (ESI) flag is transmitted dominant by error active nodes, and recessive by error passive nodes.

There are no Remote Frames (see Remote frames) in the CAN FD format. A message configured to transmit a Remote Frame is always sent out in Classical CAN format. When an FD frame is received and matches a message buffer, the RTR bit in the receiving message buffer becomes 0. The RTR bit must be considered in classical frames only.

##### 70.3.10.2.1 CAN FD messages

CAN FD messages may be formatted as long frames where the data field exceeds eight bytes, and may range from 12 up to 64 bytes. They can also be configured to support bit rate switching. In this case, the control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate than the beginning and end of the frame. Messages in the classical CAN format are limited to transport a maximum payload of eight bytes at nominal rate. Figure 372 illustrates the message formats for classical and FD frames with either standard or extended ID.

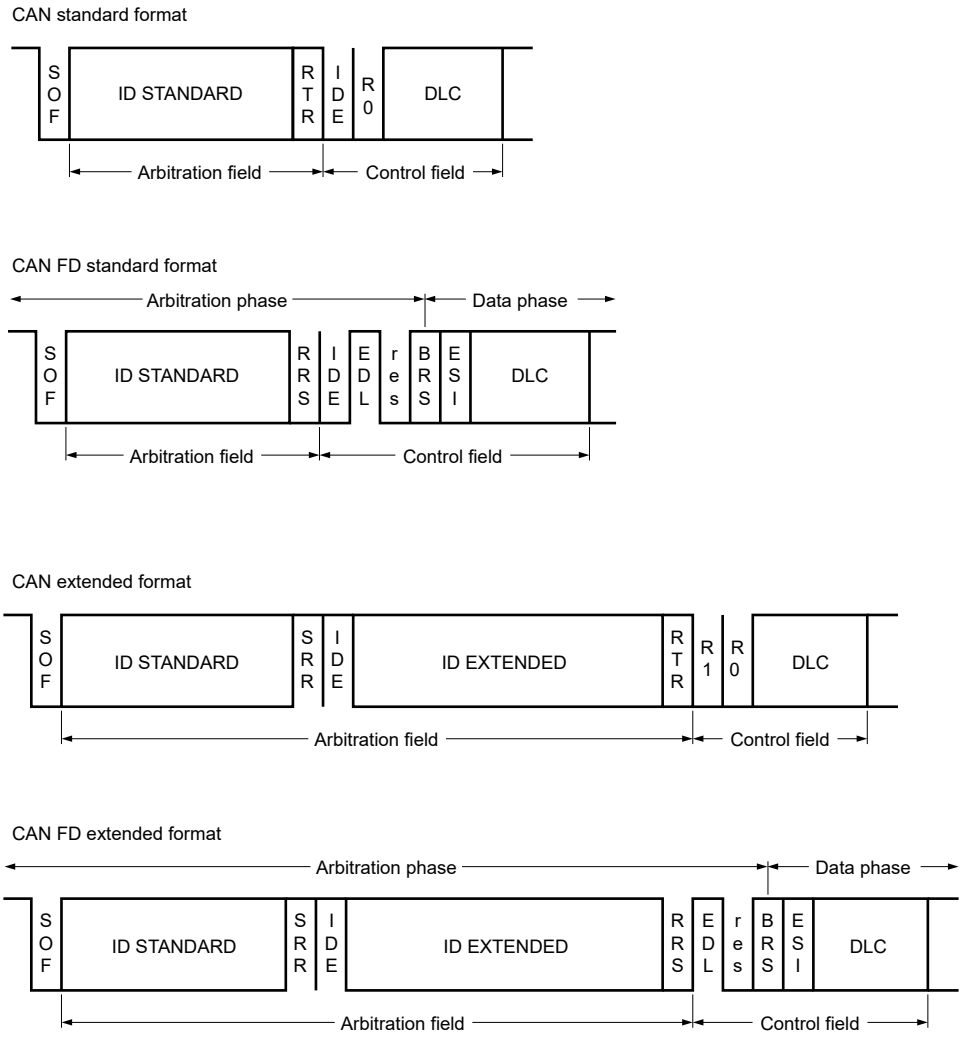



**Figure 372. CAN message formats**

MCR[FDEN] enables the ability to receive and transmit CAN FD messages. A recessive R0 bit in CAN frames with 11-bit identifiers, or a recessive R1 bit in CAN frames with 29-bit identifiers, is decoded as an EDL bit (not a reserved one). A recessive EDL bit identifies a CAN FD frame, and a dominant EDL bit identifies a Classical CAN frame. The BRS bit specifies whether this frame switches the bit rate in its data phase. A long frame is decoded according to the DLC field value (see DLC definition in Message buffer structure).

CAN FD messages can be transmitted with two different bit rates. The first part of a CAN FD frame, from the Start of Frame (SOF) bit until the Bit Rate Switch (BRS) bit is called the arbitration phase. This part is transmitted with the nominal bit rate based on a set of nominal CAN bit timing configuration values. The second part, from the BRS bit until the CRC Delimiter bit, is called the data phase. When this second part is transmitted, a second set of CAN data bit timing configuration values determines the data bit rate. Finally, from the CRC delimiter until the Intermission bits, the transmission returns to nominal bit rate.

### 70.3.10.2.2 BRS in CAN FD

In CAN FD frames with bit rate switching, the bit timing changes inside the frame at the sample point of the BRS bit if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by CAN Bit Timing (CBT). (Control 1 (CTRL1) also defines this timing for backward compatibility.) Upon detecting a recessive BRS bit, the CAN data bit timing is used as defined by CAN FD Bit Timing (FDCBT).

**NOTE**

If the time quantum length in nominal bit timing and in the data bit timing are not identical, a quantization error of up to one time quantum of the arbitration phase may be present as a phase error. This situation can occur after the switch from arbitration to data phase, and it lasts until the next synchronization event. The length of the time quantum should be the same in nominal and data bit timing. This configuration minimizes the chance of error frames on the CAN bus, and optimizes the clock tolerance in networks that use FD frames.

If BRS = 1 in the selected TX MB, FDCTRL[FDRATE] enables the transmission of all frames with bit rate switching. If FDCTRL[FDRATE] = 0, the transmission is performed at nominal rate regardless of the BRS bit value. FDCTRL[FDRATE] can be written at any time but takes effect only for the next message transmitted or received.

Nominal bit timing is resumed at the sample point of the CRC Delimiter bit or when an error is detected, whichever occurs first. Figure 373 describes the mechanism for entering and leaving the data phase when the BRS bit is recessive.

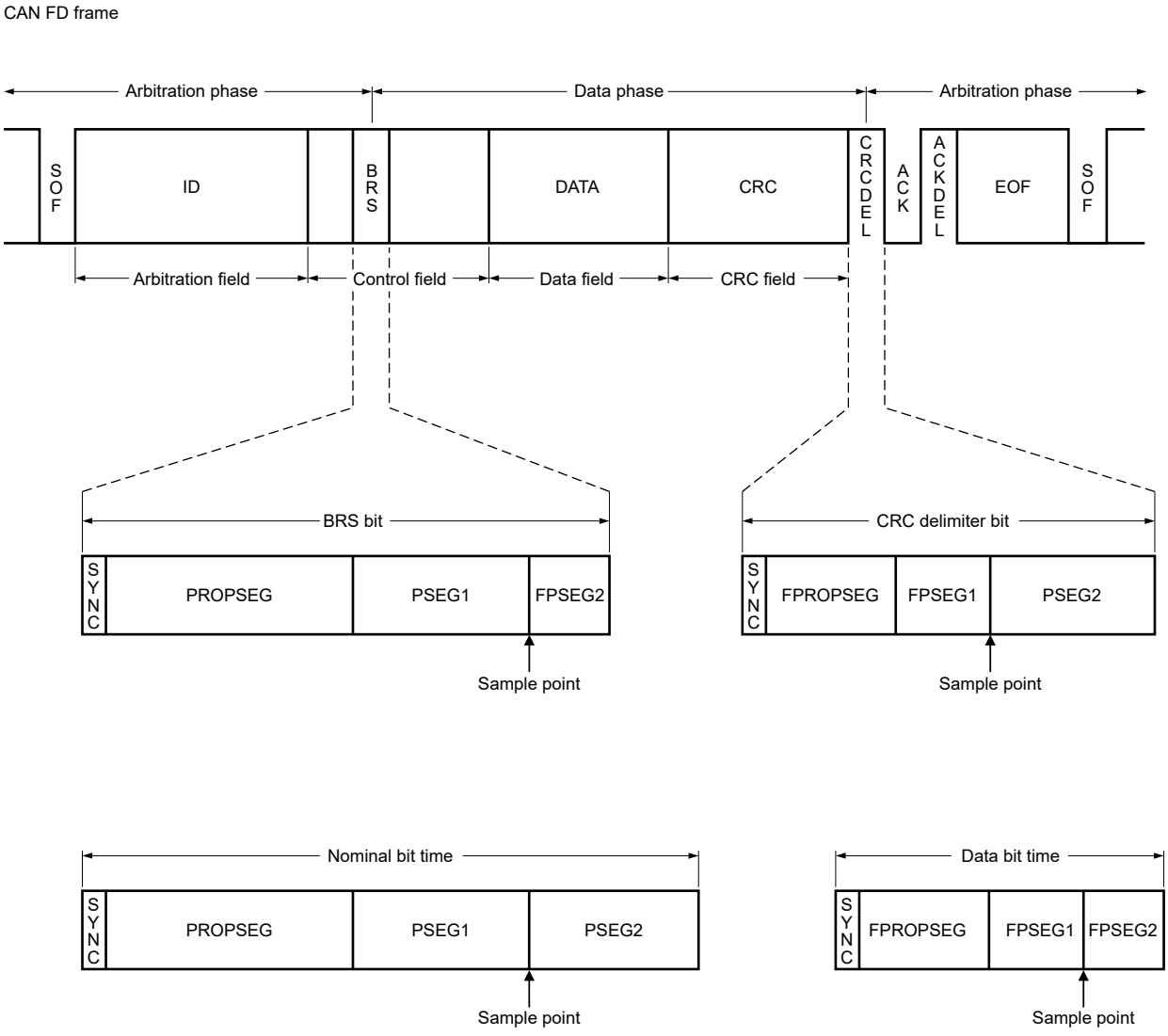



**Figure 373. Bit rate switching mechanism for CAN FD messages**

**NOTE**

In Classical CAN frames, the CRC delimiter is one recessive bit. In CAN FD frames, the CRC delimiter may consist of one or two recessive bits. FlexCAN sends only one recessive bit as the CRC delimiter. It accepts two recessive bits before the recessive-to-dominant edge that starts the acknowledge slot. As a receiver, FlexCAN sends its acknowledge bit after the first CRC delimiter bit. In CAN FD frames, FlexCAN accepts a two-bit dominant ACK slot as a valid ACK to compensate for phase shifts between the receivers.

The maximum configurable bit rate in the CAN FD data phase depends on the clock frequency of the CAN_PE subblock. For example, for a CAN_PE clock frequency of 40 MHz and the shortest configurable bit time of 5 time quanta, the bit rate in the data phase is 8 Mbit/s.

**NOTE**

The frequency used in this example may not be supported on this chip. It is shown only to demonstrate how the maximum configurable bit rate is calculated.

#### 70.3.10.2.3 ESI in CAN FD

The value of the ESI field is determined by either of the following:

- The error state of the transmitter at the start of the transmission, if the frame is originated in the FlexCAN node
- The original transmitting node when FlexCAN is acting as a gateway for the message

If the transmitter is error-passive, ESI is transmitted recessive; otherwise, it is transmitted dominant. The permutations of the relationship between the written value and the transmitted value of the ESI are shown in Table 431.

**Table 431. Written versus transmitted values of the ESI field**

| FlexCAN fault confinement status at start of frame | ESI field of TX MB | Transmitted ESI |
|---|---|---|
| Error active | 0 | 0 (Error Active) |
| Error passive | 0 | 1 (Error Passive) |
| Error active | 1 | 1 (Error Passive) |
| Error passive | 1 | 1 (Error Passive) |

#### 70.3.10.2.4 CRC calculations in CAN FD

Different CAN frame formats have different CRC polynomials. The first polynomial, CRC_15, is used for all frames in Classical CAN format. The second, CRC_17, is used for frames in CAN FD format with a data field up to 16 bytes long. The third, CRC_21, is used for frames in CAN FD format with a data field longer than 16 bytes. Each polynomial results in a Hamming distance of 6. At the start of the frame, all three CRC polynomials are calculated concurrently. The values of the EDL bit and the DLC field select the CRC sequence to be transmitted. When receiving a message, FlexCAN decodes EDL and DLC to select the adequate CRC polynomial to check for a CRC error.

In CAN FD format frames, stuff bits are included in the bit stream for CRC calculation. In Classical CAN format frames, stuff bits are not included. After the transmission of the last bit relevant to the CRC calculation, CAN FD CRC (FDCRC) stores the calculated CRC for the transmitted message. This storage is performed with adequate length for the type of message, for CAN FD and non-FD messages. Cyclic Redundancy Check (CRCR) reports a valid CRC for Classical CAN messages only.

In CAN FD format frames, the CAN bit stuffing method changes for the CRC sequence, so the stuff bits are inserted at fixed positions. When FlexCAN is transmitting a CAN FD frame, a fixed stuff bit is inserted just before the first bit of the CRC sequence. This insertion occurs even if the last bits of the preceding field do not fulfill the CAN stuff condition. Additional stuff bits are inserted after each fourth bit of the CRC sequence. The value of any fixed stuff bit is the inverse value of its preceding bit. When FlexCAN receives a CAN FD frame, it discards the fixed stuff bits from the bit stream for the CRC check. A stuff error is detected if the fixed stuff bit has the same value as its preceding bit.

#### 70.3.10.2.5 CAN FD errors

FlexCAN detects errors in CAN FD frames the same way as in Classical CAN frames. The error counters ECR[RXERRCNT] and ECR[TXERRCNT] accumulate the counts of RX and TX errors, respectively, for both FD and non-FD frames indiscriminately. Two extra error counters, ECR[RXERRCNT_FAST] and ECR[TXERRCNT_FAST], accumulate RX and TX errors occurring in the data phase of CAN FD frames with BRS = 1 only. The rules for updating the error counters are the same for both CAN FD and non-FD frames (see Error Counter (ECR)).

These error flags report errors in both CAN FD and non-FD frames:

- ESR1[BIT1ERR]
- ESR1[BIT0ERR]
- ESR1[ACKERR]
- ESR1[CRCERR]
- ESR1[FRMERR]
- ESR1[STFERR]

If CTRL1[ERRMSK] = 1, they also generate the ERRINT interrupt.

These additional error flags indicate the occurrence of errors in the data phase of CAN FD frames with BRS = 1:

- ESR1[BIT1ERR_FAST]
- ESR1[BIT0ERR_FAST]
- ESR1[CRCERR_FAST]
- ESR1[FRMERR_FAST]
- ESR1[STFERR_FAST]

No ACKERR is detected in the data phase of a CAN FD frame. Fault confinement status reported in ESR1[FLTCONF] is the same for both CAN FD and Classical CAN frames, and is based on ECR[RXERRCNT] and ECR[TXERRCNT] only. Information in ECR[RXERRCNT_FAST] and ECR[TXERRCNT_FAST] may be considered as status to help detect the error nature related to the bit rate value.

When FlexCAN detects an error while transmitting or receiving a CAN FD message in the data phase, it immediately switches:

- Back to the arbitration phase, and
- Back to the nominal rate to start an error flag.

#### 70.3.10.2.6 CAN FD synchronization

Resynchronization and hard synchronization occur in CAN FD frames in the same way as in Classical CAN ones. A hard synchronization is also performed at the recessive-to-dominant edge from EDL to R0 in CAN FD format frames. FlexCAN does not resynchronize when transmitting in the CAN FD data phase.

### 70.3.10.3 Transceiver delay compensation

The CAN FD protocol allows the transmission and reception of data at a higher bit rate than the nominal rate used in the arbitration phase, when BRS = 1 in the message. This feature enables the use of rates up to 8 Mbit/s.

During the data phase of a CAN FD frame, if the transmitter cannot receive its own latest transmitted bit at the sample point of that bit, it detects a bit error. When bit rate switching is enabled (BRS = 1), the CAN bit time in the data phase can become shorter than the loop delay of the transceiver. This condition impedes the correct comparison between the transmitted bit and the received bit within the current CAN bit time interval.

The transceiver delay compensation (TDC) process defines a secondary sample point where the transmitted bit is correctly compared to the received bit to check for bit errors.

You can enable the TDC mechanism via FDCTRL[TDCEN] or ETDC[ETDCEN]. The TDC mechanism is effective only during the data phase of FD frames with BRS = 1. It has no effect on either non-FD frames or FD frames transmitted at the normal bit rate. When the transmitted message has BRS = 1, TDC is active from the sample point of the BRS bit until the sample point of the CRC Delimiter bit. When TDC is active, the real received bit is compared to the delayed transmitted bit, where the delay is calculated based on the measured transceiver loop delay.

**NOTE**

The transmitters using TDC disregard the value of the CRC delimiter bit. A global error at the end of the CRC field causes the receivers to send error frames that the transmitter detects during Acknowledge or End of Frame.

For every transmitted FD frame with BRS = 1, the transition from the recessive EDL bit to the dominant R0 bit triggers the delay measurement (as shown in Figure 374). The loop delay is measured in Protocol Engine (PE) clock periods (CANCLK, see Protocol timing), from the transmitted EDL-R0 edge to the received EDL-R0 edge. The measured loop delay time added to an offset value specified in FDCTRL[TDCOFF] or ETDC[ETDCOFF] determines the position of the secondary sample point. FDCTRL[TDCVAL] or ETDC[ETDCVAL] stores the result of this calculation. The TDCVAL and ETDCVAL value saturates at its maximum value of 63 CANCLK and 255 CANCLK when the delay measurement is too long.

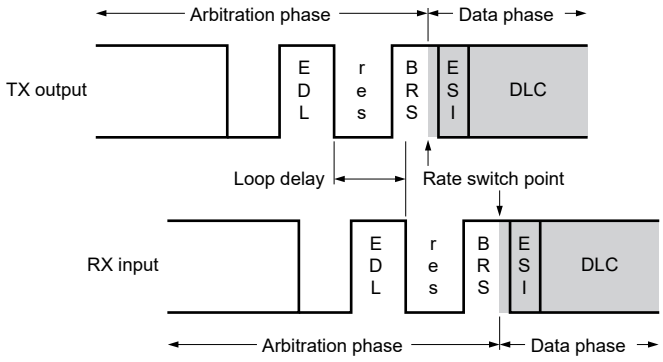



Figure 374. Transceiver loop delay measurement

The measured loop delay is not enough to define the secondary sample point, because it relates to the CAN bit edges. The transceiver delay compensation offset FDCTRL[TDCOFF] or ETDC[ETDCOFF] is used to shift the secondary sample point to an intermediate point inside the bit time, far away from its edges. The value of FDCTRL[TDCOFF] or ETDC[ETDCOFF] cannot be larger than the CAN bit duration in the data phase.

If the secondary sample point is set very near the CAN bit edge (SYNC field), problems may occur during the bit sampling in the data phase. For the TDC to work reliably, the offset must use optimal settings. To ensure that bit sampling is performed in the best region, configure the TDC offset as shown in this equation:

$$Offset = (FPSEG1 + FPROPSEG + 2) \times (FPRESDIV + 1)$$
*or*
$$Offset = (DTSEG1 + 2) \times (EDPRESDIV + 1), \textit{ if ETDCEN}$$

Equation 12. TDC offset calculation

Figure 375 shows the SSP position when these settings are used.

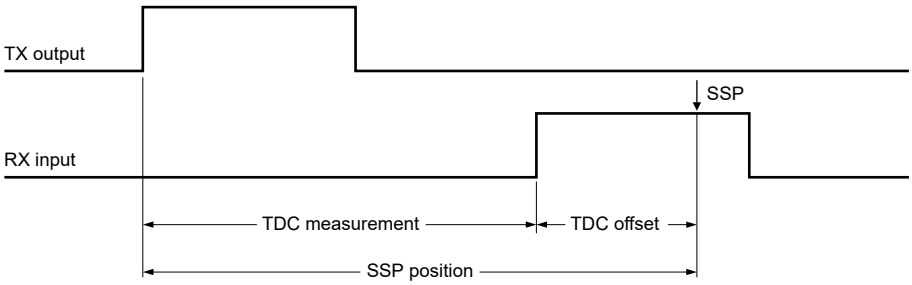



**Figure 375. SSP position with optimal values**

Alternatively, if CTRL2[BTE] and ETDC[ETDCEN] are 1, you can write 1 to ETDC[TDMDIS] to disable the transceiver delay measurement. In this case, only ETDC[ETDCOFF] defines the SSP position. Figure 376 shows the secondary sample point position when the transceiver delay measurement is disabled.

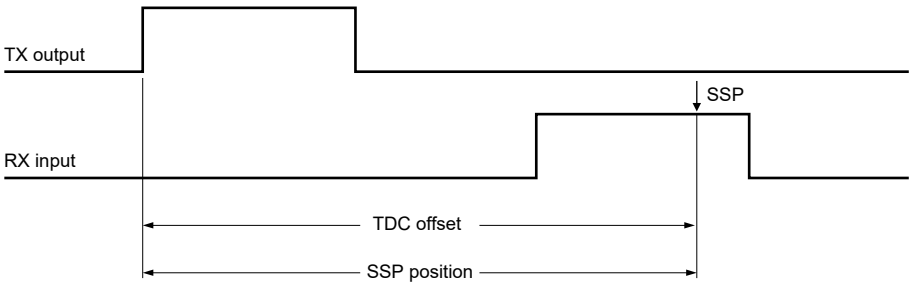



**Figure 376. SSP position with TDC measurement disabled by ETDCEN and TDMDIS**

During the data phase of CAN FD frames with bit rate switching enabled, at the onset of every TX CAN bit:

- The transmitted TX bit value is temporarily stored in a buffer.
- A time countdown based on FDCTRL[TDCVAL] or ETDC[ETDCVAL] is started. This countdown ends with the comparison of the received RX bit (delayed by the external loop delay plus the specified offset) to the stored TX bit.

If a bit error is detected at the secondary sample point, FlexCAN issues an error flag to the CAN bus at the next sample point.

During the arbitration phase, delay compensation is always disabled. During the data phase, the TDC mechanism of FlexCAN can compensate a maximum delay of 3 CAN bit times − 2 Tq. Beyond this limit, the FDCTRL[TDCFAIL] or ETDC[ETDCFAIL] flag is set. The flag indicates when the TDC mechanism is out of range and is unable to compensate the transceiver loop delay.

### 70.3.10.4 Remote frames

A remote frame is a special type of frame. You can program a message buffer to be a remote request frame by configuring the message buffer as Transmit with the RTR = 1. After the remote request frame is transmitted successfully, the message buffer becomes a receive message buffer, with the same ID as before.

When FlexCAN receives a remote request frame, the frame can be treated in different ways, depending on remote request storing (CTRL2[RRS]) and RX FIFO Enable (MCR[RFEN]):

- If RRS = 0, the ID of the frame is compared to the IDs of the transmit message buffers with the CODE field 1010b. If a matching ID exists, this message buffer frame is transmitted. If the matching message buffer has the RTR = 1, FlexCAN transmits a remote frame as a response. The received remote request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response.

  The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match. If a remote request frame is received and matches a message buffer, this message buffer immediately enters the internal arbitration process. However, it is considered a normal TX message buffer, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.

- If CTRL2[RRS] = 1, the ID of the frame is compared to the IDs of the receive message buffers with the CODE field 0100b, 0010b, or 0110b. If a matching ID exists, this message buffer stores the remote frame in the same fashion as a data frame. No automatic remote response frame is generated. The mask registers are used in the matching process.
- If MCR[RFEN] = 1, FlexCAN does not generate an automatic response for remote request frames that match the Legacy FIFO filtering criteria. If the remote frame matches one of the target IDs, it is stored in the Legacy FIFO and presented to the CPU.

  For filtering formats A and B (see legacy RX FIFO structure), it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted if they match the ID. Remote request frames are considered as normal frames. They generate a Legacy FIFO overflow when a successful reception occurs and the Legacy FIFO is already full.
- If ERFCR[ERFEN] = 1, FlexCAN does not generate an automatic response for remote request frames that match the Enhanced RX FIFO filtering criteria. Remote Request Frames are considered normal frames. They generate an Enhanced RX FIFO overflow when a successful reception occurs and the enhanced RX FIFO is already full.

**NOTE**

There is no remote frame in the CAN FD format. A fixed dominant RRS bit replaces the RTR bit. FlexCAN receives and transmits remote frames in the Classical CAN format.

#### 70.3.10.5 Overload frames

When a dominant bit is detected on the CAN bus in these locations, FlexCAN transmits overload frames:

- The first or second bit of Intermission.
- The seventh bit (last) of End of Frame field (RX frames).
- The eighth bit (last) of Error Frame Delimiter or Overload Frame Delimiter.

#### 70.3.10.6 Message buffer timestamp

The value of the free-running timer is sampled at the beginning of the Identifier field on the CAN bus. This value is stored at the end of move-in in the TIME_STAMP field of a message buffer, providing network behavior regarding time.

When CTRL2[TIMER_SRC] = 1, an external time tick continuously clocks the free-running timer.

When CTRL2[TIMER_SRC] = 0, the FlexCAN bit clock clocks the free-running timer, which defines the baud rate on the CAN bus. During a message transmission or reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate.

The free-running timer is not incremented during Disable and Freeze modes. It can be reset upon a specific frame reception, enabling network time synchronization. See CTRL1[TSYN].

Alternatively, by configuring CTRL2[MBTSBASE], the timestamp of the message buffer can capture the lower or higher 16 bits of the high-resolution dedicated counter. See the chip-specific FlexCAN timestamp implementation.

#### 70.3.10.7 High-resolution timestamp

The high-resolution timestamp (HR_TIME_STAMP) uses a dedicated timer with a 32-bit counter operating in free-running mode. CTRL2[TSTAMPCAP] enables the high-resolution timestamp. When this field is not zero, the dedicated 32-bit counter value is captured during a valid CAN frame and stored in an HR_TIME_STAMP*n* register.

Each HR_TIME_STAMP*n* corresponds to a specific message buffer. For example, HR_TIME_STAMP0 stores the 32-bit timestamp associated with message buffer 0. HR_TIME_STAMP1 stores the 32-bit timestamp associated with message buffer 1, and so on.

The counter value is captured according to CTRL2[TSTAMPCAP]. For classical CAN frames, the capture points can be the start of frame bit or the point in time a CAN frame is considered valid. This valid point is the seventh bit of end of frame for transmission and the sixth bit of end of frame for reception.

For CAN FD frames, the capture points can be:

- The start of frame

- The point in time a CAN FD frame is considered valid
- The res bit of a CAN FD frame

The 16-bit timestamp of the message buffer can be configured to capture the lower or higher 16 bits of the high-resolution timer. This configuration is made by CTRL2[MBTSBASE]. See the chip-specific FlexCAN timestamp implementation.

#### 70.3.10.8 Protocol timing

Figure 377 shows the structure of the clock generation circuitry that feeds the CAN Protocol Engine (PE) submodule.

**NOTE**

To identify the proper clock source, see the clock distribution chapter (module clocks table).

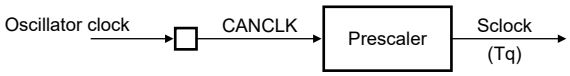



**Figure 377. CAN engine clocking scheme**

##### 70.3.10.8.1 Bit timing configuration

FlexCAN supports various means to configure bit timing parameters required by the CAN protocol. Control 1 (CTRL1) has various fields to control bit timing parameters:

- CTRL1[PRESDIV]
- CTRL1[PROPSEG]
- CTRL1[PSEG1]
- CTRL1[PSEG2]
- CTRL1[RJW]

CAN Bit Timing (CBT) extends the range of the CAN bit timing variables in CTRL1. Enhanced Data Phase CAN Bit Timing (EDCBT) provides a second set of CAN bit timing variables to be applied at the data phase of CAN FD frames with the Bit Rate Switch (BRS) = 1.

Enhanced Nominal CAN Bit Timing (ENCBT) extends the range of CAN bit timing variables in CBT. Enhanced Nominal CAN Bit Timing (ENCBT) extends the range of CAN bit timing variables in FDCBT. When using ENCBT and EDCBT, you must program the nominal bit timing and data phase serial clock (Sclock) dividers in Enhanced CAN Bit Timing Prescalers (EPRS).

**NOTE**

When the CAN FD feature is enabled, always write 1 to CBT[BTF] or CTRL2[BTE] and specify the CAN bit timing variables in CBT or ENCBT. See CAN Bit Timing (CBT) or Enhanced Nominal CAN Bit Timing (ENCBT).

CTRL1[PRESDIV], and its extended range CBT[EPRESDIV] (or EPRS[ENPRESDIV]) and FDCBT[FPRESDIV] (or EPRS[EDPRESDIV]) for the data phase bits of CAN FD messages, defines the prescaler value that generates the serial clock (Sclock). (See Equation 13.) The period of Sclock defines the time quantum used to compose the CAN waveform. A time quantum (Tq) is the atomic unit of time managed by the CAN engine. It is the smallest time unit for all configuration values.

$$Tq = \frac{(PRESDIV + 1)}{f_{CANCLK}}$$

**Equation 13. Time quantum**

The bit rate, which defines the rate the CAN message is received or transmitted, is calculated with the formula:

$$CAN\ bit\ time = (Number\ of\ time\ quanta\ in\ 1\ bit\ time) \times Tq$$

$$Bit\ rate = \frac{1}{CAN\ bit\ time}$$

**Equation 14. CAN bit time and baud rate**

### 70.3.10.8.2 Bit time segments

A bit time is subdivided into three segments as shown in Figure 378. See also Figure 379 , Figure 380, and Table 432.

**NOTE**

For further explanation of the underlying concepts, see ISO 11898-1:2015. See also *CAN Specification Version 2.0, Part A and Part B* for bit timing.

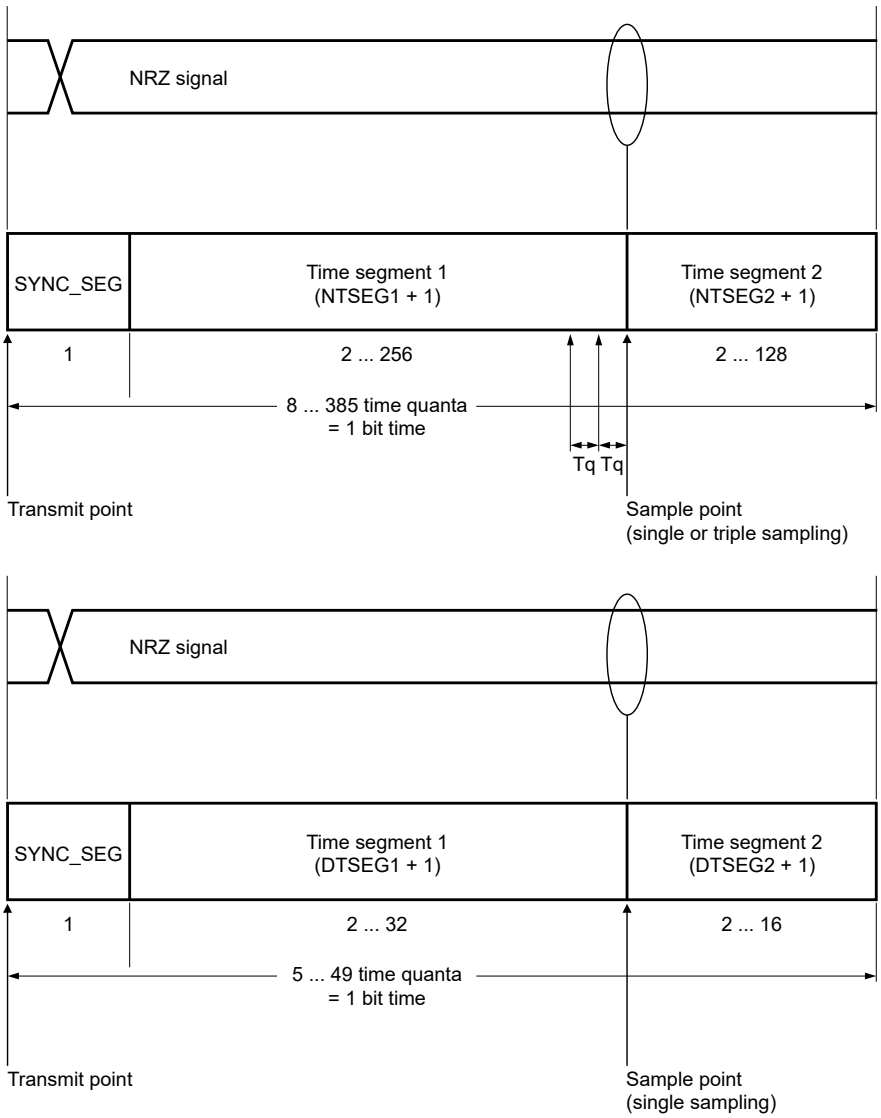



**Figure 378. Segments within the bit timing (example using ENCBT and EDCBT bit timing variables)**

The three bit time segments are:

- SYNC_SEG—this segment has a fixed length of one time quantum. Signal edges are expected to occur within this section.
- Time Segment 1—this segment includes the propagation segment and the phase segment 1 of the CAN standard.

  It can be programmed by configuring CTRL1[PROPSEG] and CTRL1[PSEG1] so that the sum (plus 2) is 2–16 time quanta. When CBT[BTF] = 1, FlexCAN uses CBT[EPROPSEG] and CBT[EPSEG1] so that the sum (plus 2) is 2–96 time quanta. For messages in CAN FD format with the BRS = 1, FlexCAN uses FDCBT[FPROPSEG] and FDCBT[FPSEG1] so that the sum (plus 1) is 2–39 time quanta.

  If CTRL2[BTE] = 1, FlexCAN uses ENCBT[NTSEG1] to configure time segment 1 to 2–256 time quanta. For the data phase in CAN FD messages with BRS = 1, EDCBT[DTSEG1] must be used for configuring time segment 1 to 2–32 time quanta.
- Time Segment 2—this segment represents the phase segment 2 of the CAN standard.

  It can be programmed by configuring CTRL1[PSEG2] (plus 1) to be 2–8 time quanta. When CBT[BTF] = 1, FlexCAN uses CBT[EPSEG2] so that its value (plus 1) is 2–32 time quanta. For messages in CAN FD format with the BRS = 1, FlexCAN uses FDCBT[FPSEG2] instead, so that its value (plus 1) is 2–8 time quanta. Time segment 2 cannot be smaller than the Information Processing Time (IPT), which is 2 time quanta in FlexCAN.

  If CTRL2[BTE] = 1, FlexCAN uses ENCBT[NTSEG2] to configure time segment 2 to 2–128 time quanta. For the data phase in CAN FD messages with BRS = 1, EDCBT[DTSEG2] must configure time segment 2 to 2–16 time quanta.

**NOTE**

The bit time defined by the above time segments must not be smaller than five time quanta. For bit time calculations, use an Information Processing Time (IPT) of two, which is the value implemented in the FlexCAN module.

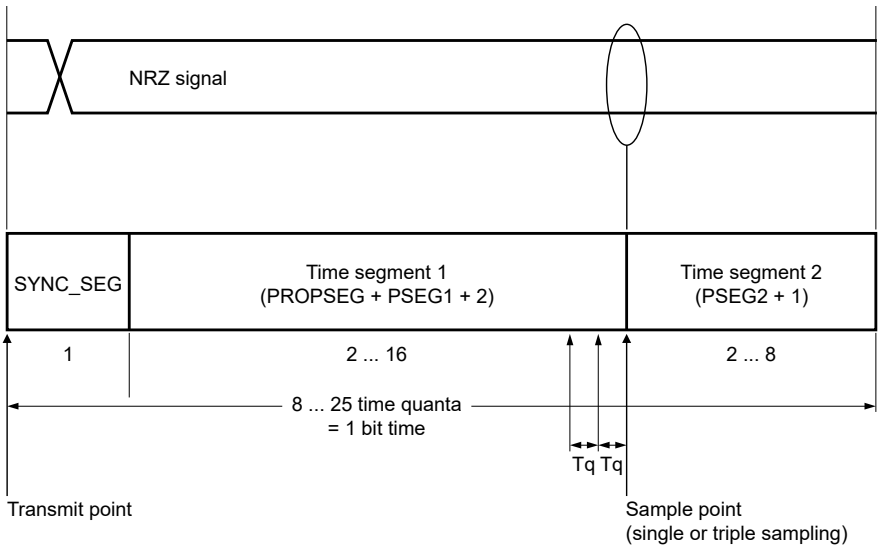



**Figure 379. Segments within the bit time (example using CTRL1 bit timing variables for Classical CAN format)**

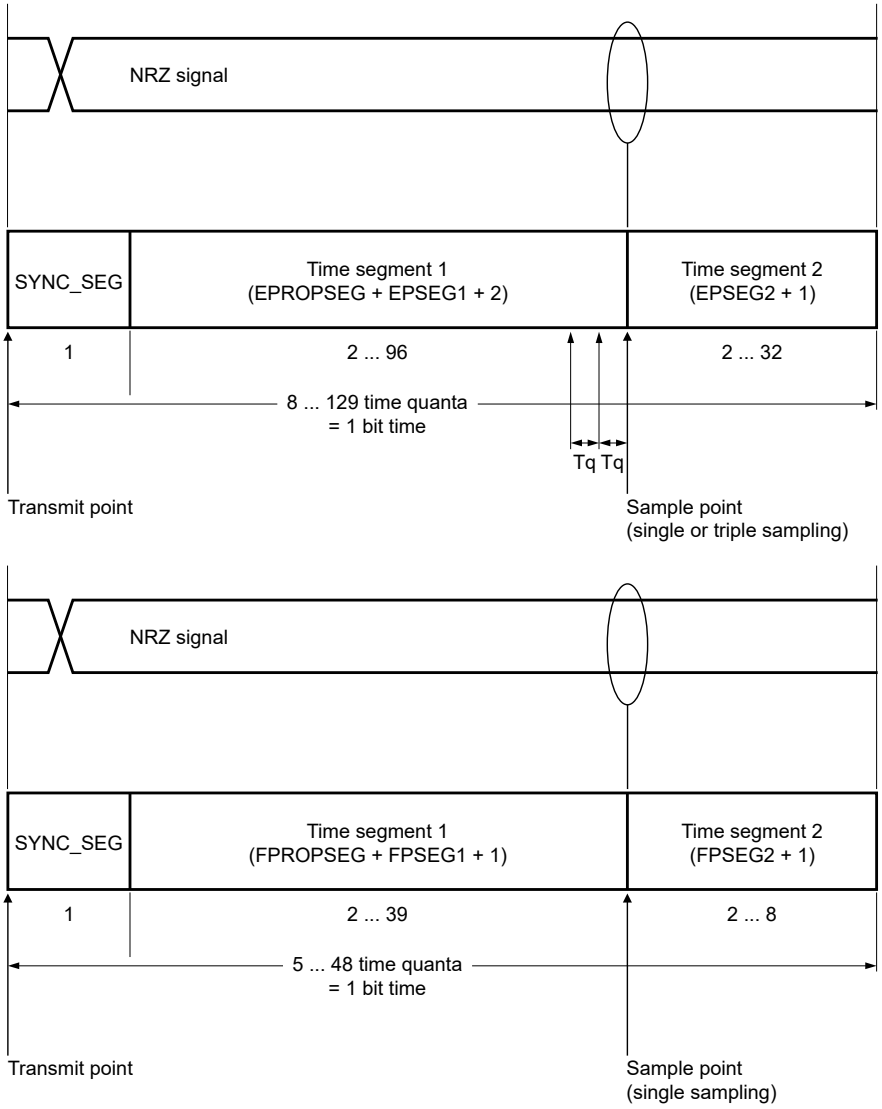



**Figure 380. Segments within the bit time (example using CBT and FDCBT bit timing variables for CAN FD format)**

**Table 432. Time segment syntax**

| Syntax | Description |
|---|---|
| SYNC_SEG | Period during which the system expects transitions to occur on the bus |
| TSEG1 | Period corresponding to the sum of PROPSEG and PSEG1 |
| TSEG2 | Period corresponding to the PSEG2 value |
| Transmit point | Point at which a node in Transmit mode transfers a new value to the CAN bus |
| Sample point | Point at which a node samples the bus. If the option of three samples per bit is selected, this point marks the position of the third sample. |

Table 433 gives some examples of the CAN-compliant segment settings for Classical CAN format (Bosch CAN 2.0B) (non-FD) messages.

Table 433. Bosch CAN 2.0B standard compliant bit time segment settings

| Time segment 1 | Time segment 2 | Resynchronization jump width |
|---|---|---|
| 5 to 10 | 2 | 1 to 2 |
| 4 to 11 | 3 | 1 to 3 |
| 5 to 12 | 4 | 1 to 4 |
| 6 to 13 | 5 | 1 to 4 |
| 7 to 14 | 6 | 1 to 4 |
| 8 to 15 | 7 | 1 to 4 |
| 9 to 16 | 8 | 1 to 4 |

**NOTE**

You must ensure the bit time settings comply with the CAN Protocol standard (ISO 11898-1:2015).

### 70.3.10.8.3 Calculating peripheral clocks

A CAN bit can be used as a measure of duration (for example, estimating the occurrence of a CAN bit event in a message). When a CAN bit is used in this way, the number of peripheral clocks in one CAN bit (NumClkBit) can be calculated as:

$$NumClkBit = \frac{f_{SYS}}{f_{CANCLK}} \times (PRESDIV + 1) \times (PROPSEG + PSEG1 + PSEG2 + 4)$$

Equation 15. Number of peripheral clocks per CAN bit when CTRL2[BTE] = 0

Or, if CTRL2[BTE] = 1:

$$NumClkBit = \frac{f_{SYS}}{f_{CANCLK}} \times (ENPRESDIV + 1) \times (NTSEG1 + NTSEG2 + 3)$$

Equation 16. Number of peripheral clocks per CAN bit when CTRL2[BTE] = 1

Where:

- NumClkBit is the number of peripheral clocks in one CAN bit.
- $f_{CANCLK}$ is the Protocol Engine (PE) Clock (see Figure 377), in Hz.
- $f_{SYS}$ is the frequency of operation of the system (CHI) clock, in Hz.
- PSEG1 is the value of CTRL1[PSEG1].
- PSEG2 is the value of CTRL1[PSEG2].
- PROPSEG is the value of CTRL1[PROPSEG].
- PRESDIV is the value in CTRL1[PRESDIV].
- ENPRESDIV is the value of EPRS[ENPRESDIV].
- NTSEG1 is the value of ENCBT[NTSEG1].
- NTSEG2 is the value of ENCBT[NTSEG2].

The formula above is also applicable to the alternative CAN bit timing variables described in:

- CAN Bit Timing (CBT)

- Enhanced Nominal CAN Bit Timing (ENCBT)
- CAN FD Bit Timing (FDCBT)
- Enhanced Nominal CAN Bit Timing (ENCBT)

For example, 180 CAN bits = (180 × NumClkBit) peripheral clock periods.

### 70.3.10.9 Arbitration and matching timing

During normal reception and transmission, the matching, arbitration, move-in, and move-out processes are executed during certain time windows inside the CAN frame. These windows are shown in the following figures.

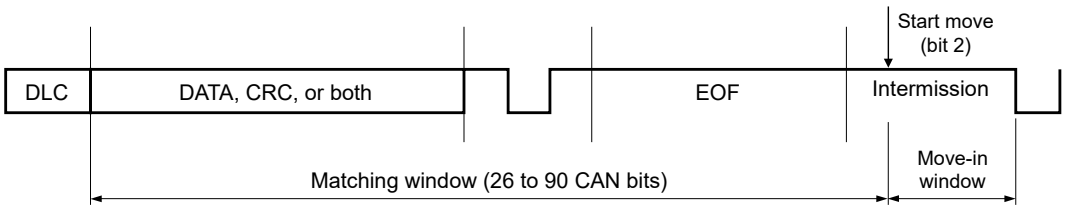



Figure 381. Matching and move-in time windows

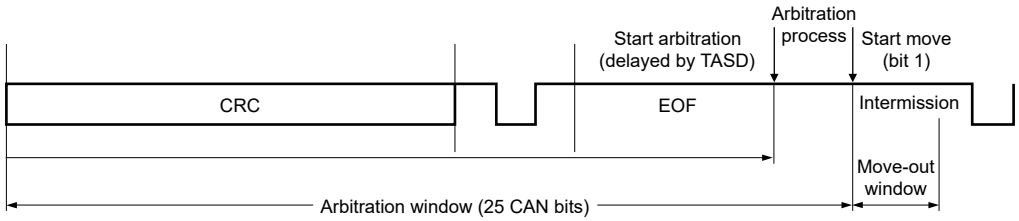



Figure 382. Arbitration and move-out time windows

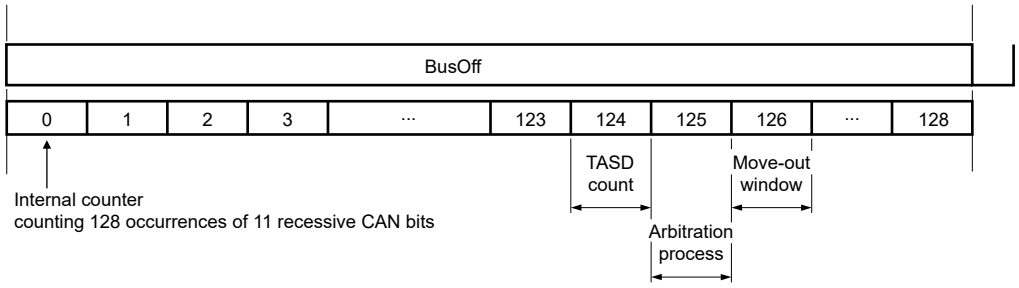



Figure 383. Arbitration at the end of bus off and move-out time windows

**NOTE**

In these figures, the matching and arbitration timing do not consider delays caused by concurrent memory access due to the CPU or other internal FlexCAN subblocks.

### 70.3.10.10 TX arbitration start delay

TX Arbitration Start Delay (CTRL2[TASD]) indicates the number of CAN bits that FlexCAN uses to delay the TX arbitration process starting point from the first bit of the CRC field of the current frame. This variable can be written only in Freeze mode; FlexCAN blocks it in other modes.

The ability of the CPU to reconfigure message buffers for transmission after the end of the internal arbitration process impacts transmission performance. In the arbitration process, FlexCAN finds the winner MB for transmission (see Arbitration process). If

the arbitration ends too early (before the first bit of the Intermission field) the CPU may reconfigure some TX message buffers. It is possible that the winning message buffer is no longer the best candidate to be transmitted.

TASD can optimize the transmission performance by defining the arbitration start point, as shown in Figure 384, based on factors such as:

- Peripheral-to-oscillator clock ratio
- CAN bit timing variables that determine the CAN bit rate
- Number of message buffers in use by the matching and arbitration processes

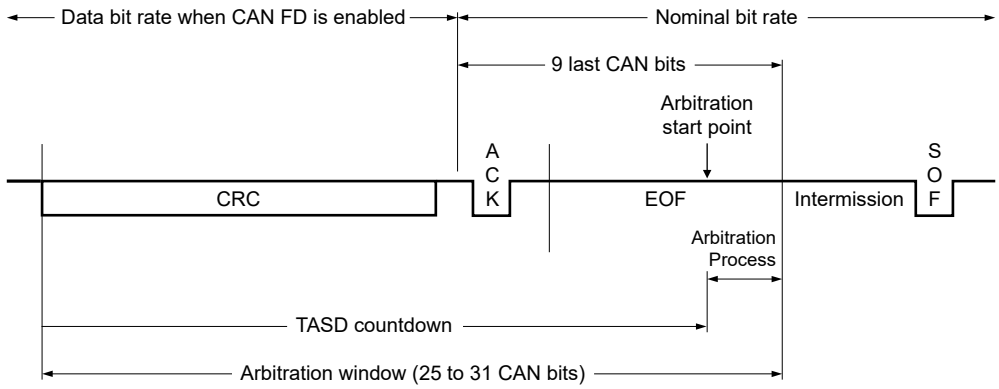



**Figure 384. Optimal TX arbitration start point**

The duration of an arbitration process, in terms of CAN bits, is:

- Directly proportional to the number of available message buffers
- Directly proportional to the CAN bit rate
- Inversely proportional to the peripheral clock frequency

The optimal arbitration timing occurs when the last message buffer is scanned immediately before the first bit of the Intermission field of a CAN frame. For instance, if the following are true:

- There are few message buffers.
- The peripheral-to-oscillator clock ratio is high.
- The CAN baud rate is low.

Then the arbitration can be placed closer to the end of the frame, adding more delay to its starting point, and vice versa.

If CTRL2[TASD] = 0, the arbitration start is not delayed, and more time is reserved for arbitration. Alternatively, if CTRL2[TASD] is close to 24, the CPU can configure a TX message buffer later, and less time is reserved for arbitration. If too little time is reserved for arbitration, FlexCAN may not be able to find a winner MB in time. The transmitted arbitration winner may not have the best chance to win the bus arbitration against external nodes on the CAN bus.

The optimal TASD value can be calculated as follows:

For CAN FD frames and (MAXMB + 1) ≤ $NMB_{END}$

$$TASD = 31 - \frac{2 * (MAXMB + 1) + 4}{CPCB_N}$$

For CAN FD frames and (MAXMB + 1) > $NMB_{END}$

$$TASD = 22 - \frac{2 * (MAXMB + 1) - NMB_{END}}{CPCB_F}$$

For non-FD frames

$$TASD = 25 - \frac{2 * (MAXMB + 1) + 4}{CPCB}$$

**Equation 17. Optimal value for TASD**

Where:

$$NMB_{END} = \frac{(9 \times CPCB_N) - 4}{2}$$

$$BITRATE_N = \left( \frac{f_{CANCLK}}{[1 + (EPSEG1 + 1) + (EPSEG2 + 1) + (EPROPSEG + 1)] \times (EPRESDIV + 1)} \right)$$

$$BITRATE_F = \left( \frac{f_{CANCLK}}{[1 + (FPSEG1 + 1) + (FPSEG2 + 1) + FPROPSEG] \times (FPRESDIV + 1)} \right)$$

$$CPCB_N = \frac{f_{SYS}}{BITRATE_N}$$

$$CPCB_F = \frac{f_{SYS}}{BITRATE_F}$$

$$CPCB = CPCB_N$$

**Equation 18. Variables used in TASD calculation**

- MAXMB is the value in MCR[MAXMB].
- $NMB_{END}$ is the number of message buffers that the arbitration process can scan during the last nine CAN bits at the end of a frame. (See Equation 18.)
- $BITRATE_N$ is the CAN bit rate in bits per second calculated by the nominal CAN bit time variables.
- $BITRATE_F$ is the CAN bit rate in bits per second calculated by the data CAN bit time variables.
- $CPCB_N$ is the number of peripheral clocks per CAN bit in nominal bit rate for CAN FD frames.
- $CPCB_F$ is the number of peripheral clocks per CAN bit in data bit rate for CAN FD frames.
- CPCB is the number of peripheral clocks per CAN bit for non-FD frames.
- $f_{CANCLK}$ is the oscillator clock, in Hz.
- $f_{SYS}$ is the peripheral clock, in Hz.
- EPSEG1 is the value in CBT[EPSEG1] (CTRL1[PSEG1] can also be used).
- EPSEG2 is the value in CBT[EPSEG2] (CTRL1[PSEG2] can also be used).
- EPROPSEG is the value in CBT[EPROPSEG] (CTRL1[PROPSEG] can also be used).
- EPRESDIV is the value in CBT[EPRESDIV] (CTRL1[PRESDIV] can also be used).
- FPSEG1 is the value in FDCBT[FPSEG1].

- FPSEG2 is the value in FDCBT[FPSEG2].
- FPROPSEG is the value in FDCBT[FPROPSEG].
- FPRESDIV is the value in FDCBT[FPRESDIV].
- NTSEG1 is the value in ENCBT[NTSEG1].
- NTSEG2 is the value in ENCBT[NTSEG2].
- ENPRESDIV is the value in EPRS[ENPRESDIV].
- DTSEG1 is the value in EDCBT[DTSEG1].
- DTSEG2 is the value in EDCBT[DTSEG2].
- EDPRESDIV is the value in EPRS[EDPRESDIV].

If CTRL2[BTE] = 1, then:

$$BITRATE_N = \frac{f_{CANCLK}}{[1 + (NTSEG1 + 1) + (NTSEG2 + 1)] \times (ENPRESDIV + 1)}$$

**Equation 19. Nominal baud rate when CTRL2[BTE] = 1**

$$BITRATE_F = \frac{f_{CANCLK}}{[1 + (DTSEG1 + 1) + (DTSEG2 + 1)] \times (EDPRESDIV + 1)}$$

**Equation 20. Fast baud rate when CTRL2[BTE] = 1**

See also Protocol timing for more details.

#### 70.3.10.10.1 TASD configuration examples

The following tables show the TASD value calculated for some configuration cases.

Case 1:

- Clock ratio = 2:1 (for example, peripheral clock 80 MHz and oscillator clock 40 MHz)
- Bit rate in arbitration phase = 1 Mbaud

**Table 434. TASD values in Case 1**

| Number of message buffers | TASD value | Maximum bit rate in data phase (MBd) |
|---|---|---|
| 16 | 24 | Invalid |
| 32 | 24 | 8.0 |
| 64 | 23 | 8.0 |
| 96 | 22 | 8.0 |

Case 2:

- Clock ratio = 1:1 (for example, peripheral clock 40 MHz and oscillator clock 40 MHz)
- Bit rate in arbitration phase = 1 Mbaud

**Table 435. TASD values in Case 2**

| Number of message buffers | TASD value | Maximum bit rate in data phase (MBd) |
|---|---|---|
| 16 | 24 | Invalid |

*Table continues on the next page...*

Table 435. TASD values in Case 2 (continued)

| Number of message buffers | TASD value | Maximum bit rate in data phase (MBd) |
|---|---|---|
| 32 | 23 | 6.67 |
| 54 | 22 | 5.0 |
| 64 | 21 | 3.33 |
| 96 | 20 | 1.6 |

Case 3:

- Clock ratio = 2:1 (for example, peripheral clock 40 MHz and oscillator clock 20 MHz)
- Bit rate in arbitration phase = 1 Mbaud

Table 436. TASD values in Case 3

| Number of message buffers | TASD value | Maximum bit rate in data phase (MBd) |
|---|---|---|
| 16 | 24 | Invalid |
| 32 | 23 | 4.0 |
| 54 | 22 | 4.0 |
| 64 | 21 | 3.33 |
| 96 | 20 | 1.54 |

## 70.3.11 Clocks

The following table describes the clock sources for FlexCAN. See the chip clocking chapter for clock setting, configuration, and gating information.

Table 437. FlexCAN clocks

| Clock name | Description |
|---|---|
| MODULE_CLK (system_clk) | Peripheral clock |
| MODULE_CLK_CHI (host_clock) | Control Host Interface (CHI) clock |
| MODULE_CLK_PE (protocol_engine_clock) | Protocol Engine (PE) clock |
| MODULE_CLK_PE_NOGATE (protocol_engine_clock_nogate) | Protocol Engine clock (no gating) |
| MODULE_CLK_S (system_clock_nogate) | Peripheral access clock |

### 70.3.11.1 Clock domains and restrictions

FlexCAN has two clock domains asynchronous to each other:

- The bus domain feeds the Control Host Interface (CHI) submodule.
- The oscillator domain feeds the CAN Protocol Engine (PE) submodule.

When the two domains are connected to clocks with different frequencies or phases, the frequency relationship between the two clock domains is restricted. In asynchronous operation, the bus domain clock frequency must always be greater than the oscillator domain clock frequency.

**NOTE**

Asynchronous operation with a 1:1 ratio between peripheral and oscillator clocks is not allowed.

When performing matching and arbitration, FlexCAN must scan the whole message buffer memory during the time slot of one CAN frame, comprised of a number of CAN bits. To provide sufficient time for the scan, observe the following requirements:

- The peripheral clock frequency cannot be less than the oscillator clock frequency.
- There must be a minimum number of peripheral clocks per CAN bit, as specified in Table 438.

**Table 438. Minimum number of peripheral clocks per CAN bit for Classical CAN format**

| Number of message buffers | Value of MCR[RFEN] | Value of ERFCR[ERFEN] | Minimum number of peripheral clocks per CAN bit |
|---|---|---|---|
| 16 | 0 | 0 | 16 |
| 32 | 0 | 0 | 16 |
| 64 | 0 | 0 | 25 |
| 96 | 0 | 0 | 37 |
| 16 | 1 | 0 | 16 |
| 32 | 1 | 0 | 17 |
| 64 | 1 | 0 | 30 |
| 96 | 1 | 0 | 42 |
| 16 | 0 | 1 | 16 |
| 32 | 0 | 1 | 19 |
| 64 | 0 | 1 | 31 |
| 96 | 0 | 1 | 42 |

For classical frame format, the minimum number of peripheral clocks per CAN bit specified in Table 438 determines the minimum peripheral clock frequency for a given number of message buffers and for an expected CAN bit rate. The CAN bit rate depends on the number of time quanta in a CAN bit. This number can be defined by adjusting one or more of the bit timing values contained in:

- Control 1 (CTRL1)
- CAN Bit Timing (CBT)
- Enhanced Nominal CAN Bit Timing (ENCBT)

The time quantum (Tq) is defined in Protocol timing. The minimum number of time quanta per CAN bit must be eight, so the oscillator clock frequency should be at least eight times the CAN bit rate.

#### 70.3.11.1.1 Clock restrictions for CAN FD

For CAN FD frame format, some constraints must be satisfied. The equation below calculates the number of peripheral clocks per CAN bit in nominal bit rate (NumClkNomBit).

$$NumClkNomBit = \frac{f_{SYS}}{f_{CANCLK}} \times (PRESDIV + 1) \times (PROPSEG + PSEG1 + PSEG2 + 4)$$

$$= \frac{f_{SYS}}{NomBitRate}$$

**Equation 21. Number of peripheral clocks per nominal CAN bit**

Where PRESDIV, PSEG1, and PSEG2 are CAN bit time values in Control 1 (CTRL1). Alternatively, EPRESDIV, EPSEG1, and EPSEG2 values in CAN Bit Timing (CBT) or the values of EPRS[ENPRESDIV], ENCBT[NTSEG1], and ENCBT[NTSEG2] can be used instead. NumClkNomBit can also be calculated as a function of the expected nominal bit rate used in the arbitration phase (NomBitRate), as shown in the equation above.

The number of CAN bits in the data phase of an FD frame with BRS = 1 (fast CAN bits) depends on the number of data bytes in the payload. The number of fast CAN bits (NumOfFastBits) can be determined in Table 439. Having fewer data bytes means having fewer fast CAN bits. It also means that less time is available for FlexCAN to scan the whole message buffer memory during the internal matching and arbitration processes.

**Table 439. Number of fast CAN bits in a CAN FD frame**

| Minimum number of data bytes | DLC field | NumOfFastBits |
|---|---|---|
| 0 | 0h | 21 |
| 1 | 1h | 29 |
| 2 | 2h | 37 |
| 3 | 3h | 45 |
| 4 | 4h | 53 |
| 5 | 5h | 61 |
| 6 | 6h | 69 |
| 7 | 7h | 77 |
| 8 | 8h | 85 |
| 12 | 9h | 117 |
| 16 | Ah | 149 |
| 20 | Bh | 186 |
| 24 | Ch | 218 |
| 32 | Dh | 282 |
| 48 | Eh | 410 |
| 64 | Fh | 538 |

The critical part of a CAN FD frame is during the data phase, where the CAN bit rate is faster than in the arbitration phase. The minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) can be calculated to guarantee that enough time is available for FlexCAN to scan the message buffer memory during reception and transmission. The equation below calculates this constraint.

$$MinNumClkFastBit_A = \frac{(8.5 \times MaxNumOfMb) + [ERFEN \times (2 \times NFE + 4)] + 64 - (9 \times NumClkNomBit)}{NumOfFastBits}$$

**Equation 22. Minimum number of peripheral clocks per fast CAN bit for FlexCAN scan process**

Where MaxNumOfMb is the maximum number of available message buffers defined in MCR[MAXMB]. NFE and ERFEN are the fields defined in Enhanced RX FIFO Control (ERFCR).

The clock-domain-crossing circuit between the CHI and PE subblocks also imposes a minimum number of peripheral clocks per fast CAN bit. This minimum is required for the handshake mechanism to work properly without losing status information through the interface, as shown in the equation below.

$$MinNumClkFastBit_B = 3 \times \left(1 + \frac{f_{SYS}}{f_{CANCLK}}\right)$$

**Equation 23. Minimum number of peripheral clocks per fast CAN bit for FlexCAN clock domain interface**

Therefore, the larger of the two values calculated above determines the minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit).

$$MinNumClkFastBit = Maximum\ (MinNumClkFastBitA, MinNumClkFastBitB)$$

**Equation 24. Minimum number of peripheral clocks per fast CAN bit**

Then, the maximum CAN bit rate in the data phase of CAN FD frames (DataBitRateMAX) can be calculated as below.

$$DataBitrate_{MAX} = \frac{f_{CANCLK}}{ROUNDUP\left(\frac{MinNumClkFastBit \times f_{CANCLK}}{f_{SYS}}\right)}$$

**Equation 25. Maximum achievable baud rate for data phase**

These factors affect the maximum data bit rate attainable by FlexCAN in CAN FD mode:

- The peripheral and oscillator clock frequencies
- The maximum number of message buffers
- The expected nominal bit rate

Also, the data bit rate depends on the minimum payload size of FD frames used in a given application.

To illustrate how the configuration of FlexCAN variables affects the CAN FD bit rate, consider this application example:

- The peripheral clock frequency is set to 50 MHz
- The oscillator clock frequency is set to 40 MHz

1. Considering the nominal bit rate as 1 Mbit/s, the number of peripheral clocks per CAN bit in nominal bit rate is calculated as below.

   $$NumClkNomBit = \frac{50 \times 10^6}{1 \times 10^6} = 50$$

   **Equation 26. Calculation example for number of peripheral clocks per nominal CAN bit**

2. The number of fast CAN bits (NumOfFastBits) is determined in Table 439. For example, if the minimum payload in FD frames is 8 bytes, there are 85 CAN bits in the data phase.
3. Assuming the maximum number of message buffers is 96, and Enhanced RX FIFO is disabled, the minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) can be calculated.

   $$MinNumClkFastBit_A = \frac{(8.5 \times 96) + 64 - (9 \times 50)}{85} = 5.06$$

   **Equation 27. Calculation example for number of peripheral clocks per fast CAN bit for FlexCAN scan process**

$$MinNumClkFastBit_B = 3 \times \left(1 + \frac{50}{40}\right) = 6.75$$

**Equation 28. Calculation example for number of peripheral clocks per fast CAN bit for FlexCAN clock domain interface**

$$MinNumClkFastBit = Maximum\ (\ 5.06, 6.75\ ) = 6.75$$

**Equation 29. Calculation example for number of peripheral clocks per fast CAN bit**

4. The maximum CAN bit rate in the data phase can finally be found.

$$DataBitRate_{MAX} = \frac{40 \times 10^6}{ROUNDUP\left(\frac{6.75 \times 40 \times 10^6}{50 \times 10^6}\right)} = 6.667\ Mbps$$

**Equation 30. Calculation example for maximum achievable baud rate**

Even though the oscillator clock frequency (40 MHz) is adequate to generate a data rate of 8 Mbit/s in CAN FD mode, the specific FlexCAN configuration limits this rate to 6.667 Mbit/s. This limitation is mainly due to the low peripheral clock frequency that imposes the MinNumClkFastBitB bound.

Table 440 shows the maximum data rate for CAN FD with Enhanced RX FIFO disabled according to clock frequencies, payload size, and number of available message buffers. For some cases, if the number of available message buffers is reduced, FlexCAN can then achieve a data rate up to 8 Mbit/s.

**Table 440. Maximum CAN bit rate in data phase on CAN FD frames with Enhanced RX FIFO disabled**

| Peripheral clock frequency (MHz) | Payload size | Number of available message buffers | Maximum data rate (Mbit/s) |
|---|---|---|---|
| 40 | 8 | 94 | 6.667 |
| 40 | 8 | 114 | >5.0 |
| 40 | 12 | >117 | 6.667 |
| 40 | 12 | 128 | 5.714 |
| 50 | 12–64 | 128 | 6.667 |
| 60 | 8 | 126 | 8.0 |
| 60 | 12 | 128 | 8.0 |
| 67 | 6 | 128 | 8.0 |
| 80 | 3 | 128 | 8.0 |
| 100 | 0 | 128 | 8.0 |

## 70.3.12 Reset

You can reset FlexCAN in the following ways:

- Soft reset:
  - MCR[SOFTRST], which resets some of the memory-mapped registers synchronously. To see which registers soft reset affects, see Table 443.
  - Chip-level soft reset, which has the same effect as MCR[SOFTRST].

Soft reset is synchronous and must follow an internal request-and-acknowledge procedure across clock domains. Therefore, it may take some time to propagate its effects fully. MCR[SOFTRST] remains 1 when soft reset is pending, so software can poll this field to identify when the reset has completed. Soft reset cannot be applied when clocks are shut down in a low-power mode. The low-power mode should be exited and the clocks resumed before applying soft reset.

When the module is enabled (MCR[MDIS] becomes 0), FlexCAN automatically enters Freeze mode. In Freeze mode:

1. FlexCAN is unsynchronized to the CAN bus.
2. MCR[HALT] and MCR[FRZ] become 1.
3. The internal state machines are disabled.
4. MCR[FRZACK] and MCR[NOTRDY] become 1.

The TX pin is in the recessive state and FlexCAN does not initiate any transmission or reception of CAN frames. Reset does not affect the message buffers and the RX Individual Mask registers, so they are not automatically initialized.

### 70.3.13 Interrupts

FlexCAN has many interrupt sources:

- Interrupts due to message buffers
- Interrupts due to interrupts combined via an OR operator from:
  - Message buffers
  - Bus Off
  - Bus Off Done
  - Error
  - Error Fast (errors detected in the data phase of CAN FD format messages with BRS = 1)
  - TX Warning
  - RX Warning

If its corresponding IMASK bit is 1, each message buffer can be an interrupt source. There is no distinction between TX and RX interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each buffer has an assigned flag bit in the IFLAG registers. When the corresponding buffer completes a successful transfer, the flag is set. When the CPU writes 1 to it, the flag is cleared (unless another interrupt is generated at the same time).

**NOTE**

The CPU must clear only the bit causing the current interrupt. For this reason, do not use bit manipulation instructions (BSET) to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags which are set after entering the current interrupt handler.

If the Legacy RX FIFO is enabled (MCR[RFEN] = 1) and DMA is disabled (MCR[DMA] = 0), the interrupts corresponding to message buffers 0–7 have different meanings.

- Bit 7 of Interrupt Flags 1 (IFLAG1) becomes the Legacy FIFO Overflow flag
- Bit 6 becomes the Legacy FIFO Warning flag
- Bit 5 becomes the Frames Available in Legacy FIFO flag
- Bits 4–0 are unused.

See Interrupt Flags 1 (IFLAG1) for more information.

If both Legacy RX FIFO and DMA are enabled (MCR[RFEN] = 1 and MCR[DMA] = 1), FlexCAN does not generate any Legacy FIFO interrupt. Bit 5 of IFLAG1 still indicates Frames Available in Legacy FIFO and generates a DMA request. Bits 7, 6, and 4–0 are unused.

**CAUTION**

Legacy FIFO cannot be enabled when CAN FD is enabled.

When multiple message buffer interrupt sources are combined via an OR operator into a single interrupt, the interrupt is generated when any associated message buffer (or FIFO, if applicable) generates an interrupt. In this case, the CPU must read the IFLAG registers to determine which message buffer or FIFO source caused the interrupt.

These interrupt sources generate interrupts like the message buffer interrupt sources, and can be read from Error and Status 1 (ESR1):

- Bus Off
- Bus Off Done
- Error
- Error Fast
- TX Warning
- RX Warning

The Bus Off, Error, TX Warning, and RX Warning interrupt masks are located in Control 1 (CTRL1).

### 70.3.14 Bus interface

CPU access to FlexCAN registers is subject to the following rules:

- Unrestricted read and write access to supervisor registers results in an access error. In Table 443, supervisor registers are registers identified with an S, and registers that are identified with S/U that are in Supervisor Mode
- Read and write access to implemented reserved address space results in an access error.
- Write access to positions whose bits are all currently read-only results in an access error. If at least one of the bits is not read-only, no access error is issued. Write permission to specific positions or some of their bits can change depending on the mode of operation or transitory state. See register and field descriptions for details.
- Read and write access to unimplemented address space results in an access error.
- Read and write access to RAM-located positions during Low-Power mode results in an access error.
- The RXIMR memory region can be considered as general-purpose memory and available for access via these methods:
  - If you write 0 to MCR[IRMQ], the individual masks (RXIMR) are disabled. In this case, the RXIMR memory region is considered general-purpose memory.
  - If MCR[MAXMB] is programmed with a value smaller than the available number of message buffers, the unused memory space can be used as general-purpose RAM space. Reserved words within RAM cannot be used.
    For example, suppose the RAM in FlexCAN can support up to 16 message buffers, CTRL2[RFFN] = 0h, and MCR[MAXMB] = 0.
    - In this case, the maximum number of message buffers becomes one.
    - The RAM starts at 0080h, and the space 0080h–008Fh is used by the one message buffer.
    - The memory space 0090h–017Fh is available.
    - The space 0180h–087Fh is reserved.
    - The space 0880h–0883h is used by the one individual mask and the available memory in the mask register space is 0884h–08BFh.
    - In the space from 08C0h–09DFh, there are reserved words for internal use which cannot be used as general-purpose RAM.

    As a rule, free memory space for general purpose depends only on MCR[MAXMB].
  - If MCR[FDEN] = 1, general-purpose memory can be used only outside Freeze mode.

Table 441. Access permissions

| Supervisor access | 0 | | | 1 | | | |
|---|---|---|---|---|---|---|---|
| Supervisor mode<br>MCR[SUPV] = 1 | 0 | | | 1 | Any | | |
| Modes of operation | Normal | Freeze | Low-power | Any | Normal | Freeze | Low-power |
| MCR | Bus error | Bus error | Bus error | Bus error | Read and write | Read and write | Read and write |
| CTRL1 | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| TIMER | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| TCR | Bus error | Bus error | Bus error | Bus error | Bus error | Bus error | Bus error |
| RXMGMASK[1] | Bus error for write operation | Read and write | Bus error for write operation | Bus error | Bus error for write operation | Read and write | Bus error for write operation |
| RX14MASK[1] | Bus error for write operation | Read and write | Bus error for write operation | Bus error | Bus error for write operation | Read and write | Bus error for write operation |
| RX15MASK[1] | Bus error for write operation | Read and write | Bus error for write operation | Bus error | Bus error for write operation | Read and write | Bus error for write operation |
| ECR | Bus error for write operation | Read and write | Bus error for write operation | Bus error | Bus error for write operation | Read and write | Bus error for write operation |
| ESR1 | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| IMASK2 | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| IMASK1 | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| IFLAG2 | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| IFLAG1 | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| CTRL2 | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| ESR2 | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| CRCR | Bus error for write operation | Bus error for write operation | Bus error for write operation | Bus error | Bus error for write operation | Bus error for write operation | Bus error for write operation |

*Table continues on the next page...*

Table 441. Access permissions (continued)

| Supervisor access | 0 | | | | 1 | | |
|---|---|---|---|---|---|---|---|
| Supervisor mode<br>MCR[SUPV] = 1 | 0 | | | 1 | Any | | |
| Modes of operation | Normal | Freeze | Low-power | Any | Normal | Freeze | Low-power |
| RXFGMASK[1] | Bus error for write operation | Read and write | Bus error for write operation | Bus error | Bus error for write operation | Read and write | Bus error for write operation |
| RXFIR[1] | Bus error for write operation | Bus error for write operation | Bus error for write operation | Bus error | Bus error for write operation | Bus error for write operation | Bus error for write operation |
| CBT | Bus error for write operation | Read and write | Bus error for write operation | Bus error | Bus error for write operation | Read and write | Bus error for write operation |
| DBG1 | Bus error for write operation | Bus error for write operation | Bus error for write operation | Bus error for write operation | Bus error for write operation | Bus error for write operation | Bus error for write operation |
| DBG2 | Bus error for write operation | Bus error for write operation | Bus error for write operation | Bus error for write operation | Bus error for write operation | Bus error for write operation | Bus error for write operation |
| IMASK3 | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| IFLAG3 | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| MB[1] [2] | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| Legacy FIFO header[1] | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| Legacy FIFO reserved space[1] | Bus error | Bus error | Bus error | Bus error | Bus error | Bus error | Bus error |
| Legacy FIFO filters[1] | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| RXIMR[1] | Bus error | Read and write | Bus error | Bus error | Bus error | Read and write | Bus error |
| MECR | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| ERRIAR | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| ERRIDPR | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| ERRIPPR | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |

*Table continues on the next page...*

Table 441. Access permissions (continued)

| Supervisor access | 0 | | | | 1 | | |
|---|---|---|---|---|---|---|---|
| Supervisor mode<br>MCR[SUPV] = 1 | 0 | | | 1 | Any | | |
| Modes of operation | Normal | Freeze | Low-power | Any | Normal | Freeze | Low-power |
| RERRAR [3] | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| RERRDR[3] | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| RERRSYNR[3] | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| ERRSR | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| EPRS | Bus error for write operation | Read and write | Bus error for write operation | Bus error | Bus error for write operation | Read and write | Bus error for write operation |
| ENCBT | Bus error for write operation | Read and write | Bus error for write operation | Bus error | Bus error for write operation | Read and write | Bus error for write operation |
| EDCBT | Bus error for write operation | Read and write | Bus error for write operation | Bus error | Bus error for write operation | Read and write | Bus error for write operation |
| ETDC | Bus error for write operation | Read and write | Bus error for write operation | Bus error | Bus error for write operation | Read and write | Bus error for write operation |
| FDCTRL | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| FDCBT | Read and write[3] | Read and write | Read and write[3] | Bus error | Read and write[3] | Read and write | Read and write[3] |
| FDCRC | Bus error for write operation | Bus error for write operation | Bus error for write operation | Bus error | Bus error for write operation | Bus error for write operation | Bus error for write operation |
| ERFCR | Read and write[3] | Read and write | Read and write[3] | Bus error | Read and write[3] | Read and write | Read and write[3] |
| ERFIER | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| ERFSR | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| HR_TIME_STAMP | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |

*Table continues on the next page...*

Table 441. Access permissions (continued)

| Supervisor access | 0 | | | | 1 | | |
|---|---|---|---|---|---|---|---|
| Supervisor mode<br>MCR[SUPV] = 1 | 0 | | | 1 | Any | | |
| Modes of operation | Normal | Freeze | Low-power | Any | Normal | Freeze | Low-power |
| Enhanced Rx FIFO header[4] | Bus error for write operation | Bus error for write operation | Bus error for write operation | Bus error | Bus error for write operation | Bus error for write operation | Bus error for write operation |
| Enhanced Rx FIFO reserved space[4] | Bus error | Bus error | Bus error | Bus error | Bus error | Bus error | Bus error |
| ERFFEL | Bus error for write operation | Read and write | Bus error for write operation | Bus error | Bus error for write operation | Read and write | Bus error for write operation |
| General purpose RAM[1] | Read and write | Read and write | Read and write | Bus error | Read and write | Read and write | Read and write |
| Reserved space (used)[1] | Bus error | Bus error | Bus error | Bus error | Bus error | Bus error | Bus error |
| Reserved space (empty)[1] | Bus error | Bus error | Bus error | Bus error | Bus error | Bus error | Bus error |

1. Access in low power is only possible if RAM_clk and MODULE_CLK are enabled.
2. If MCR[RFEN] = 1, see Legacy FIFO access rules in the rows below.
3. Write operation has no effect.
4. If MCR[RFEN] = 1, do not access Enhanced RX FIFO.

### 70.3.15 Detection and correction of memory errors

To update the parity bits in memory properly, all FlexCAN memory must be initialized before starting its operation. CTRL2[WRMFRZ] grants write access to all memory positions that require initialization, from 080h–ADFh and from C20h–31FFh. You must also initialize these registers:

- RX Message Buffers Global Mask (RXMGMASK)
- Receive 14 Mask (RX14MASK)
- Receive 15 Mask (RX15MASK)
- Legacy RX FIFO Global Mask (RXFGMASK)

MCR[RFEN] and ERFCR[ERFEN] must not be 1 during memory initialization.

FlexCAN supports detection and correction of errors in memory read accesses. Each byte of FlexCAN memory is associated with five parity bits that ensure a Hamming distance of 4. The error correction mechanism ensures that in this 13-bit word, errors in one bit can be corrected (correctable errors). Errors in two bits can be detected but not corrected (uncorrectable errors). Errors in more than two bits may not be detected. For uncorrectable errors, the error correction logic does not change the corrupted data. When a read access is performed, the parity bits are used to calculate a syndrome, which indicates the error in each byte.

When an all-zeroes or an all-ones read occurs, FlexCAN detects an uncorrectable error. See Error Report Syndrome (RERRSYNR).

Memory errors are indicated to the host via status register (Error Status (ERRSR)) and bus transfer errors. Memory errors are reported via these report registers:

- Error Report Address (RERRAR)
- Error Report Data (RERRDR)

- Error Report Syndrome (RERRSYNR)

MECR[ECCDIS] determines whether the error detection and correction mechanism can be activated. When disabled, updates on indications and reporting registers are stopped. To ensure that memory has consistent parity bits associated with the data, the parity bits are still calculated and written with data in memory write operations.

To avoid accidentally changing the critical error correction configuration, follow this protocol to enable the update of Memory Error Control (MECR):

1. Write 1 to CTRL2[ECRWRE]. (By default, CTRL2[ECRWRE] is 0 and MECR[ECRWRDIS] is 1.)
2. Write 0 to MECR[ECRWRDIS].
3. All writes to Memory Error Control (MECR) must keep MECR[ECRWRDIS] = 0.
4. After configuration is done, lock MECR by writing 1 to MECR[ECRWRDIS] or writing 0 to CTRL2[ECRWRE].

#### 70.3.15.1 Sources of memory access

These major sources (or requestors) can access FlexCAN memory:

- Host (CPU). The largest word accessed is 32-bit.
- FlexCAN internal processes:
  - RX matching
  - TX arbitration
  - Move-in on reception
  - Move-out on transmission

  The largest word accessed is 64-bit.

The source of access determines the way that uncorrectable errors are indicated and reported.

#### 70.3.15.2 Error indication

These flags indicate memory errors:

- ERRSR[HANCEIF]
- ERRSR[FANCEIF]
- ERRSR[CEIF]

Uncorrectable errors detected in memory reads requested by the host are indicated separately from the errors detected in requests by FlexCAN internal processes. When correctable errors are detected, FlexCAN makes no distinction of the source of the access. There are three independent flags for these three cases, and each flag raises an interrupt unless a mask in Memory Error Control (MECR) masks it. If both uncorrectable and correctable errors are found in different bytes in the same read operation, both flags are set.

An uncorrectable error detected in host access is also indicated as a bus transfer error. A bus wait request may be asserted to extend the memory transaction to the moment the report registers are updated. This indication cannot be masked. If ERRSR[HANCEIF] is not masked, the same uncorrectable error raises a bus transfer error and an interrupt request.

Each indication flag has one overrun flag in Error Status (ERRSR). The overrun flags do not request interrupts. Overrun flags for uncorrectable errors indicate that other errors of the same nature were detected after the current error being treated. Overrun flags for correctable errors indicate that other errors of the same nature were detected before the current error being treated.

The recommended handling sequence for error indication is:

1. Get error report information from report registers.
2. Use this information to take proper measures in the application.
3. Clear the ERRSR[HANCEIF], ERRSR[FANCEIF], and ERRSR[CEIF] flags.

4. If the overrun flag is active:
    a. Alert the application that at least one error could not be managed.
    b. Clear the overrun flag.

FlexCAN internal processes can access memory in transactions larger than 32 bits. For the indication, this kind of access is considered a consecutive sequence of 32-bit accesses. If errors are found in two or more 32-bit words, the interrupt and overrun flags are set simultaneously.

### 70.3.15.3 Error reporting

The error report registers provide detailed information about the address read, raw data, and syndrome read with error. The flags described in Error indication indicate these report registers:

- Error Report Address (RERRAR)
- Error Report Data (RERRDR)
- Error Report Syndrome (RERRSYNR)

The address, data, and syndrome registers are updated simultaneously with the error flags, according to these rules:

1. If either of the uncorrectable error flags is set, the report registers are not updated. The previous uncorrectable error reporting is preserved.
2. Otherwise, either no error flag is set or only the correctable error flag is set. The report registers are updated according to the new error, or according to the most severe new error, if uncorrectable and correctable errors are simultaneously detected.

Reporting of errors detected in accesses larger than 32 bits follows the rules described in Error indication and in the description of Error Report Address (RERRAR).

The addresses reported in RERRAR and defined in ERRIAR are not the same as the addresses listed in the module memory map. The relation between the reported addresses and the respective ones in the module memory map is shown in the description of Error Injection Address (ERRIAR).

Addresses reported when reading memory portions organized as FIFOs refer to the address of the specific entry accessed in the FIFO, not to the FIFO base address. Such memory portions include:

- Legacy RX FIFO Structure
- Enhanced RX FIFO Structure
- Legacy RX FIFO Information (RXFIR)

To ensure coherence of error report registers, disable the report update by writing 1 to MECR[RERRDIS] before reading the report registers.

### 70.3.15.4 Error response

Correctable errors have no consequence to FlexCAN operation because the host or FlexCAN internal processes corrects affected data before its use.

For host-initiated reads, an uncorrectable error may affect the host, but does not affect FlexCAN operation.

Uncorrectable errors detected on memory read operations requested by FlexCAN internal processes may result in incorrect operation depending on the state of MECR[NCEFAFRZ], as follows:

- During reception (either matching or move-in process), when an uncorrectable error occurs:
    — An incorrect destination may be selected to store the incoming frame.
    — A corrupted frame may be stored in the correct destination.
    — Both of these events may occur.

If MECR[NCEFAFRZ] = 1, FlexCAN stops operation automatically and enters Freeze mode to prevent corrupted data from being treated as valid by FlexCAN internal processes. If MECR[NCEFAFRZ] = 0, FlexCAN continues working, and a corrupted frame is received.

- During arbitration, when an uncorrectable error occurs:
  - A non-highest-priority TX message buffer may be mistakenly selected for transmission.
  - Its data may be corrupted.

  If MECR[NCEFAFRZ] = 1, FlexCAN stops operation automatically and enters Freeze mode before starting the move-out. If MECR[NCEFAFRZ] = 0, FlexCAN proceeds to move-out with a corrupted frame that will be transmitted on the CAN bus.
- During move-out, when an uncorrectable error occurs, a corrupted frame is copied from the selected TX MB that won the arbitration to the TX SMB for transmission. If MECR[NCEFAFRZ] = 1, FlexCAN stops operation automatically and enters Freeze mode before starting the transmission. When MECR[NCEFAFRZ] = 0, the corrupted frame is transferred from the TX SMB to the Protocol Engine (PE) subblock and is transmitted on the CAN bus.
- An uncorrectable error can also be detected beyond move-out, when TX data is read from TX SMB (buffer located in RAM) to be transferred to the PE subblock for transmission. In this case, a frame with corrupted ID or data is transmitted on the CAN bus.

  To prevent the external nodes from successfully receiving the frame, FlexCAN inverts all bits in the CRC field (CRC sequence plus CRC delimiter). Also, it transmits an error flag just after CRC delimiter due to self-detecting a Bit1 error and a form error due to the CRC field inversion. When MECR[NCEFAFRZ] = 1, FlexCAN stops operation automatically and enters Freeze mode just after the error frame. When MECR[NCEFAFRZ] = 0, FlexCAN may attempt to retransmit the same frame, as long as no other higher-priority TX MB is configured for transmission after.

  If the uncorrectable error persists, FlexCAN eventually reaches the Bus Off state due to consecutive error detections. ECR[TXERRCNT] is updated every time FlexCAN inverts the CRC field, causing errors as described above.

When MECR[NCEFAFRZ] = 1 and FlexCAN enters Freeze mode, only the CPU can cause FlexCAN to exit Freeze mode and resume Normal mode. MECR[NCEFAFRZ] becoming 1 is the only way to prevent corrupted frames from being transmitted on the CAN bus to the move-out internal process.

The error report registers can provide information to the application for customized management of these situations.

#### 70.3.15.5 Error injection

These error injection registers are used to inject errors in memory reads to force errors and update the indication and reporting registers:

- Error Injection Address (ERRIAR)
- Error Injection Data Pattern (ERRIDPR)
- Error Injection Parity Pattern (ERRIPPR)

The relation between the error injection addresses and the corresponding addresses in the module memory map is shown in Error Injection Address (ERRIAR).

The injection is done by flipping the data and parity bits corresponding to the bits set to 1 in ERRIDPR and ERRIPPR. You can select injection specifically for memory accesses requested by the host or by FlexCAN internal processes.

For accesses larger than 32 bits, MECR[EXTERRIE] extends the injection pattern, replicating it in 32-bit words to fill the width of the access.

**NOTE**

It is possible, but very unlikely, for error injection to correct a bit with error. This correction does not raise the error flags and reports as expected.

To ensure coherence among error injection registers and avoid spurious error injections, you must clear MECR[HAERRIE] and MECR[FAERRIE] when configuring the memory injection registers.

## 70.4 External signal descriptions

FlexCAN has two I/O signals connected to the external chip pins. These signals are summarized in Table 442 and described in the following subsections.

**Table 442. FlexCAN signal descriptions**

| Signal | Description | I/O |
|---|---|---|
| CAN RX | CAN receive pin | Input |
| CAN TX | CAN transmit pin | Output |

### 70.4.1 CAN RX

This pin is the receive pin from the CAN bus transceiver. Logic level 0 represents its dominant state. Logic level 1 represents its recessive state.

### 70.4.2 CAN TX

This pin is the transmit pin to the CAN bus transceiver. Logic level 0 represents its dominant state. Logic level 1 represents its recessive state.

## 70.5 Initialization and application information

### 70.5.1 FlexCAN initialization sequence

For any configuration change or initialization, you must put FlexCAN into Freeze mode (see Freeze mode). The module must be initialized after every reset. FlexCAN memory must be initialized before switching to functional mode.

The following is a generic initialization sequence applicable to FlexCAN:

1. Initialize Module Configuration (MCR).
    a. Enable the individual filtering per message buffer and reception queue features by writing 1 to MCR[IRMQ].
    b. Enable the warning interrupts by writing 1 to MCR[WRNEN].
    c. If required, disable frame self-reception by writing 1 to MCR[SRXDIS].
    d. Enable the Legacy RX FIFO by writing 1 to MCR[RFEN] or enable the Enhanced RX FIFO by writing 1 to ERFCR[ERFEN].
    e. If Legacy RX FIFO or Enhanced RX FIFO is enabled and DMA is required, write 1 to MCR[DMA].
    f. Enable the abort mechanism by writing 1 to MCR[AEN].
    g. Enable the local priority feature by writing 1 to MCR[LPRIOEN].
2. Initialize Control 1 (CTRL1) and CAN FD Bit Timing (FDCBT). Optionally initialize CAN Bit Timing (CBT).
    a. Determine the bit timing parameters: CTRL1[PROPSEG], CTRL1[PSEG1], CTRL1[PSEG2], and CTRL1[RJW].
    b. Optionally determine the bit timing parameters: CBT[EPROPSEG], CBT[EPSEG1], CBT[EPSEG2], and CBT[ERJW].
    c. Determine the CAN FD bit timing parameters: FDCBT[FPROPSEG], FDCBT[FPSEG1], FDCBT[FPSEG2], and FDCBT[FRJW].
    d. Determine the bit rate by programming CTRL1[PRESDIV] and optionally programming CBT[EPRESDIV].
    e. Determine the CAN FD bit rate by programming FDCBT[FPRESDIV].
    f. Determine the internal arbitration mode by programming CTRL1[LBUF].

3. If Error Code Correction (ECC) is enabled, you must initialize all FlexCAN memory. See Detection and correction of memory errors.
4. Initialize the message buffers. (See Message buffer structure for message buffer details.)
    a. The control and status word of all message buffers must be initialized.
    b. If RX FIFO is enabled, the ID filter table must be initialized.
    c. Other entries in each message buffer should be initialized as required.
5. Initialize Receive Individual Mask (RXIMR0 - RXIMR95).
6. Write 1 to required interrupt mask bits in:
    - IMASK registers (for all message buffer interrupts)
    - Control 1 (CTRL1) and Control 2 (CTRL2) (for Bus Off and Error interrupts)
7. Write 0 to MCR[HALT].

After the last step listed above, FlexCAN attempts to synchronize to the CAN bus.

## 70.6 Memory map and register definition

This section describes the registers and data structures in FlexCAN. The base address of the module depends on the particular memory map of the chip.

### 70.6.1 FlexCAN memory mapping

The address space occupied by FlexCAN has 128 bytes for registers starting at the module base address, followed by embedded RAM starting at address offset 0080h.

Each individual register is identified by its complete name and corresponding mnemonic. The access type can be Supervisor (S) or Unrestricted (U). Most registers can be configured to have either Supervisor or Unrestricted access by programming MCR[SUPV]. These registers are identified as S/U in the Access column of Table 443.

**NOTE**

An invalid register access results in a bus error. Invalid accesses include reading a write-only register, writing a read-only register, and accessing an invalid address.

**NOTE**

To update the parity bits in memory properly, all FlexCAN memory must be initialized before reading registers which are implemented in memory. CTRL2[WRMFRZ] grants write access to all memory positions that require initialization, from 080h–ADFh and from C20h–31FFh. You must also initialize RX Message Buffers Global Mask (RXMGMASK), Receive 14 Mask (RX14MASK), Receive 15 Mask (RX15MASK) and Legacy RX FIFO Global Mask (RXFGMASK). MCR[RFEN] and ERFCR[ERFEN] must not be 1 during memory initialization.

**Table 443. Register access and reset information**

| Register | Access type | Affected by hard reset | Affected by soft reset |
|---|---|---|---|
| Module Configuration (MCR) | S | Yes | Yes |
| Control 1 (CTRL1) | S/U | Yes | No |
| Free-Running Timer (TIMER) | S/U | Yes | Yes |
| RX Message buffers Global Mask (RXMGMASK) | S/U | No | No |

*Table continues on the next page...*

Table 443. Register access and reset information (continued)

| Register | Access type | Affected by hard reset | Affected by soft reset |
|---|---|---|---|
| RX Buffer 14 Mask (RX14MASK) | S/U | No | No |
| RX Buffer 15 Mask (RX15MASK) | S/U | No | No |
| Error Counter (ECR) | S/U | Yes | Yes |
| Error and Status 1 (ESR1) | S/U | Yes | Yes |
| Interrupt Masks 2 (IMASK2) | S/U | Yes | Yes |
| Interrupt Masks 1 (IMASK1) | S/U | Yes | Yes |
| Interrupt Flags 2 (IFLAG2) | S/U | Yes | Yes |
| Interrupt Flags 1 (IFLAG1) | S/U | Yes | Yes |
| Control 2 (CTRL2) | S/U | Yes | No |
| Error and Status 2 (ESR2) | S/U | Yes | Yes |
| Cyclic Redundancy Check (CRCR) | S/U | Yes | Yes |
| RX FIFO Global Mask (RXFGMASK) | S/U | No | No |
| RX FIFO Information (RXFIR) | S/U | No | No |
| CAN Bit Timing (CBT) | S/U | Yes | No |
| Interrupt Masks 3 (IMASK3) | S/U | Yes | Yes |
| Interrupt Flags 3 (IFLAG3) | S/U | Yes | Yes |
| Message buffers | S/U | No | No |
| RX Individual Masks | S/U | No | No |
| Memory Error Control (MECR) | S/U | Yes | Yes |
| Error Injection Address (ERRIAR) | S/U | Yes | Yes |
| Error Injection Data Pattern (ERRIDPR) | S/U | Yes | Yes |
| Error Injection Parity Pattern (ERRIPPR) | S/U | Yes | Yes |
| Error Report Address (RERRAR) | S/U | Yes | Yes |
| Error Report Data (RERRDR) | S/U | Yes | Yes |
| Error Report Syndrome (RERRSYNR) | S/U | Yes | Yes |

*Table continues on the next page...*

Table 443. Register access and reset information (continued)

| Register | Access type | Affected by hard reset | Affected by soft reset |
|---|---|---|---|
| Error Status (ERRSR) | S/U | Yes | Yes |
| Enhanced CAN Bit Timing Prescalers (EPRS) | S/U | Yes | No |
| Enhanced Nominal CAN Bit Timing (ENCBT) | S/U | Yes | No |
| Enhanced Data Phase CAN bit Timing (EDCBT) | S/U | Yes | No |
| Enhanced Transceiver Delay Compensation (ETDC) | S/U | Yes | No |
| CAN FD Control (FDCTRL) | S/U | Yes | No |
| CAN FD Bit Timing (FDCBT) | S/U | Yes | No |
| CAN FD CRC (FDCRC) | S/U | Yes | Yes |
| Enhanced RX FIFO Control (ERFCR) | S/U | Yes | Yes |
| Enhanced RX FIFO Interrupt Enable (ERFIER) | S/U | Yes | Yes |
| Enhanced RX FIFO Status (ERFSR) | S/U | Yes | Yes |
| High-Resolution Timestamp (HR_TIME_STAMP) | S/U | No | No |
| Enhanced RX FIFO | S/U | No | No |
| Enhanced RX FIFO Filter Element (ERFFEL) | S/U | No | No |

FlexCAN can store CAN messages for transmission and reception using message buffers and RX FIFO structures.

## 70.6.2 FlexCAN register descriptions

The table below shows the FlexCAN memory map.

The address range from offset 80h–67Fh allocates the ninety-six 128-bit message buffers. The memory maps for the message buffers are in FlexCAN message buffer memory map.

The address range from offset 2000h–204Ch allocates the Enhanced RX FIFO output, and the address range from offset 2050h–263Ch allocates the rest of Enhanced RX FIFO 19 elements. The memory map for the Enhanced RX FIFO is in Enhanced RX FIFO structure.

### 70.6.2.1 FlexCAN memory map

FLEXCAN_0 base address: 4030_4000h

FLEXCAN_1 base address: 4030_8000h

FLEXCAN_2 base address: 4030_C000h

FLEXCAN_3 base address: 4031_0000h

FLEXCAN_4 base address: 4031_4000h

FLEXCAN_5 base address: 4031_8000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Module Configuration (MCR) | 32 | RW | D890_000Fh |
| 4h | Control 1 (CTRL1) | 32 | RW | 0000_0000h |
| 8h | Free-Running Timer (TIMER) | 32 | RW | 0000_0000h |
| 10h | RX Message Buffers Global Mask (RXMGMASK) | 32 | RW | See section |
| 14h | Receive 14 Mask (RX14MASK) | 32 | RW | See section |
| 18h | Receive 15 Mask (RX15MASK) | 32 | RW | See section |
| 1Ch | Error Counter (ECR) | 32 | RW | 0000_0000h |
| 20h | Error and Status 1 (ESR1) | 32 | RW | 0000_0000h |
| 24h | Interrupt Masks 2 (IMASK2) | 32 | RW | 0000_0000h |
| 28h | Interrupt Masks 1 (IMASK1) | 32 | RW | 0000_0000h |
| 2Ch | Interrupt Flags 2 (IFLAG2) | 32 | RW | 0000_0000h |
| 30h | Interrupt Flags 1 (IFLAG1) | 32 | RW | 0000_0000h |
| 34h | Control 2 (CTRL2) | 32 | RW | See section |
| 38h | Error and Status 2 (ESR2) | 32 | R | 0000_0000h |
| 44h | Cyclic Redundancy Check (CRCR) | 32 | R | 0000_0000h |
| 48h | Legacy RX FIFO Global Mask (RXFGMASK) | 32 | RW | See section |
| 4Ch | Legacy RX FIFO Information (RXFIR) | 32 | R | See section |
| 50h | CAN Bit Timing (CBT) | 32 | RW | 0000_0000h |
| 6Ch | Interrupt Masks 3 (IMASK3) | 32 | RW | 0000_0000h |
| 74h | Interrupt Flags 3 (IFLAG3) | 32 | RW | 0000_0000h |
| 880h - 9FCh | Receive Individual Mask (RXIMR0 - RXIMR95) | 32 | RW | See section |
| AE0h | Memory Error Control (MECR) | 32 | RW | 800C_0080h |
| AE4h | Error Injection Address (ERRIAR) | 32 | RW | 0000_0000h |
| AE8h | Error Injection Data Pattern (ERRIDPR) | 32 | RW | 0000_0000h |
| AECh | Error Injection Parity Pattern (ERRIPPR) | 32 | RW | 0000_0000h |
| AF0h | Error Report Address (RERRAR) | 32 | R | 0000_0000h |
| AF4h | Error Report Data (RERRDR) | 32 | R | 0000_0000h |
| AF8h | Error Report Syndrome (RERRSYNR) | 32 | R | 0000_0000h |
| AFCh | Error Status (ERRSR) | 32 | RW | 0000_0000h |
| BF0h | Enhanced CAN Bit Timing Prescalers (EPRS) | 32 | RW | 0000_0000h |
| BF4h | Enhanced Nominal CAN Bit Timing (ENCBT) | 32 | RW | 0000_0000h |
| BF8h | Enhanced Data Phase CAN Bit Timing (EDCBT) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| BFCh | Enhanced Transceiver Delay Compensation (ETDC) | 32 | RW | 0000_0000h |
| C00h | CAN FD Control (FDCTRL) | 32 | RW | 8000_0100h |
| C04h | CAN FD Bit Timing (FDCBT) | 32 | RW | 0000_0000h |
| C08h | CAN FD CRC (FDCRC) | 32 | R | 0000_0000h |
| C0Ch | Enhanced RX FIFO Control (ERFCR) | 32 | RW | 0000_0000h |
| C10h | Enhanced RX FIFO Interrupt Enable (ERFIER) | 32 | RW | 0000_0000h |
| C14h | Enhanced RX FIFO Status (ERFSR) | 32 | RW | 0000_0000h |
| C30h - DACh | High-Resolution Timestamp (HR_TIME_STAMP0 - HR_TIME_STAMP95) | 32 | RW | See section |
| 3000h - 31FCh | Enhanced RX FIFO Filter Element (ERFFEL0 - ERFFEL127) | 32 | RW | See section |

### 70.6.2.2 Module Configuration (MCR)

#### Offset

| Register | Offset |
|---|---|
| MCR | 0h |

#### Function

Defines global system configurations, including the module operation modes and the maximum message buffer configuration.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MDIS | FRZ | RFEN | HALT | NOTRDY | Reserved | SOFTRST | FRZACK | SUPV | Reserved | WRNEN | LPMACK | Reserved | Reserved | SRXDIS | IRMQ |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | DMA | 0 | LPRIOEN | AEN | FDEN | 0 | IDAM | | 0 | MAXMB | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

### Fields

| Field | Function |
|---|---|
| 31<br>MDIS | Module Disable<br>Disables FlexCAN. When disabled, FlexCAN disables the clocks to the CAN Protocol Engine and Controller Host Interface submodules. Soft reset does not affect this field.<br>0b - Enable<br>1b - Disable |
| 30<br>FRZ | Freeze Enable<br>Specifies FlexCAN behavior when MCR[HALT] = 1 or when Debug mode is requested at chip level. When this field becomes 1, FlexCAN can enter Freeze mode. Writing 0 to this field causes FlexCAN to exit from Freeze mode. The chip writes 1 to this field when a noncorrectable error is detected (MECR[NCEFAFRZ] becomes 1).<br>0b - Disable<br>1b - Enable |
| 29<br>RFEN | Legacy RX FIFO Enable<br>Enables the Legacy RX FIFO feature. When this field is 1, message buffers 0–5 cannot be used for normal reception and transmission. The corresponding memory region (80h–DCh) is used by the FIFO engine and additional message buffers (up to 32, depending on CTRL2[RFFN]). These message buffers are used as Legacy RX FIFO ID filter table elements. This field also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in Table 438. This field can be written in Freeze mode only; the module blocks it in other modes.<br>**NOTE**<br>When CAN FD operation is enabled (see MCR[FDEN]), you cannot write 1 to this field.<br>**NOTE**<br>This field must not be 1 if ERFCR[ERFEN] = 1.<br>0b - Disable<br>1b - Enable |
| 28<br>HALT | Halt FlexCAN<br>Puts FlexCAN into Freeze mode. The CPU should write 0 to this field after initializing the message buffers and Control 1 (CTRL1) and Control 2 (CTRL2). FlexCAN performs no reception or transmission before this field becomes 0. Freeze mode cannot be entered when FlexCAN is in a low-power mode. The module writes 1 to this field when a noncorrectable error is detected and MECR[NCEFAFRZ] = 1.<br>0b - No request<br>1b - Enter Freeze mode, if MCR[FRZ] = 1. |
| 27<br>NOTRDY | FlexCAN Not Ready<br>Indicates whether FlexCAN is in Disable mode or Freeze mode. When FlexCAN has exited these modes, this field becomes 0. Soft reset does not affect this field. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - FlexCAN is in Normal mode, Listen-Only mode, or Loopback mode.<br>1b - FlexCAN is in Disable mode or Freeze mode. |
| 26<br>— | Reserved<br>When writing to this field, always write the reset value. |
| 25<br>SOFTRST | Soft Reset<br>Resets internal state machines of FlexCAN and some memory-mapped registers.<br>The CPU can write 1 to this field directly. Because soft reset is synchronous and must follow a request-and-acknowledge procedure across clock domains, it may take some time to propagate its effect fully. When reset is pending, this field remains 1; it automatically becomes 0 when reset completes. You can poll this field to know when the soft reset has completed.<br>Soft reset cannot be applied when clocks are shut down in a low-power mode. Transfer the module out of the low-power mode before applying soft reset. Soft reset does not affect this field.<br>**NOTE**<br>This field becomes 0 within 2 CAN bits after assertion of this bit.<br>0b - No reset<br>1b - Soft reset affects reset registers |
| 24<br>FRZACK | Freeze Mode Acknowledge<br>Indicates whether FlexCAN is in Freeze mode and its prescaler is stopped. The Freeze mode request cannot be granted until current transmission or reception processes have finished. Therefore, you can poll this field to know when FlexCAN has entered Freeze mode. If the Freeze mode request is negated, this field becomes 0 after the FlexCAN prescaler is running again. If Freeze mode is requested when FlexCAN is in a low-power mode, this field becomes 1 only when the low-power mode is exited. See Freeze mode. Soft reset does not affect this field.<br>**NOTE**<br>This field becomes 1 within 180 CAN bits if current transmission or reception is ongoing for classical frames and 730 CAN bits if current transmission or reception is ongoing for FD frames after the low-power mode requested by CPU. This field becomes 0 within 2 CAN bits after the Freeze mode request removal (see Protocol timing).<br>0b - Not in Freeze mode, prescaler running.<br>1b - In Freeze mode, prescaler stopped. |
| 23<br>SUPV | Supervisor Mode<br>Configures FlexCAN to be either in Supervisor or User mode. The registers that this field affects are marked as S/U in the Access type column of Table 443. The affected registers start with Supervisor access allowance only. In User mode, affected registers allow both Supervisor and Unrestricted accesses.<br>In Supervisor mode, affected registers allow only Supervisor access. Unrestricted access behaves as though the access is performed on an unimplemented register location.<br>This field can be written only in Freeze mode; the module blocks it in other modes. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - User mode<br>1b - Supervisor mode |
| 22<br>— | Reserved<br>When writing to this field, always write the reset value. |
| 21<br>WRNEN | Warning Interrupt Enable<br>Enables the generation of the flags ESR1[TWRNINT] and ESR1[RWRNINT]. When this field is 1, TWRNINT and RWRNINT flags are set when the respective error counter transitions from less than 96 to greater than or equal to 96. When this field is 0, the TWRNINT and RWRNINT flags are always zero, independent of the values of the error counters. No warning interrupt is generated. This field can be written in Freeze mode only; the module blocks it in other modes.<br>0b - Disable<br>1b - Enable |
| 20<br>LPMACK | Low-Power Mode Acknowledge<br>Indicates whether FlexCAN is in a low-power mode (Disable mode). A low-power mode cannot be entered until all current transmission and reception processes have finished. The CPU can poll this field to know when FlexCAN has entered low-power mode. Soft reset does not affect this field.<br>**NOTE**<br>This field becomes 1 within 180 CAN bits if current transmission or reception is ongoing for classical frames and 730 CAN bits if current transmission or reception is ongoing for FD frames after the low-power mode requested by CPU. This field becomes 0 within 2 CAN bits after the low-power mode request removal (see Protocol timing).<br>0b - Not in a low-power mode<br>1b - In a low-power mode |
| 19<br>— | Reserved<br>When writing to this field, always write the reset value. |
| 18<br>— | Reserved<br>When writing to this field, always write the reset value. |
| 17<br>SRXDIS | Self-Reception Disable<br>Determines whether FlexCAN can receive frames transmitted by itself. If 1, frames transmitted by the module are not stored in any MB, regardless of whether the MB is programmed with an ID that matches the transmitted frame. No interrupt flag or interrupt signal is generated due to the frame reception. This field can be written only in Freeze mode; the module blocks it in other modes.<br>0b - Enable<br>1b - Disable |
| 16 | Individual RX Masking and Queue Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| IRMQ | Indicates whether RX matching process is based on individual masking and queue, or based on a masking scheme with RX Message Buffers Global Mask (RXMGMASK), Receive 14 Mask (RX14MASK), Receive 15 Mask (RX15MASK), and Legacy RX FIFO Global Mask (RXFGMASK).<br>When this field is disabled, for backward compatibility with legacy applications, reading the Control and Status word locks the MB even if it is empty.<br>This field can be written in Freeze mode only. The module blocks it in other modes.<br>0b - Disable<br>1b - Enable |
| 15<br>DMA | DMA Enable<br>Enables DMA. The DMA feature can only be used in Legacy RX FIFO or Enhanced RX FIFO, so MCR[RFEN] or ERFCR[ERFEN] must be 1. When DMA and RFEN are 1, IFLAG1[BUF5I] generates the DMA request, and no RX FIFO interrupt is generated. This field can be written in Freeze mode only; the module blocks it in other modes.<br>0b - Disable<br>1b - Enable |
| 14<br>— | Reserved |
| 13<br>LPRIOEN | Local Priority Enable<br>Enables the local priority feature. It is used to expand the ID used during the arbitration process. With this expanded ID concept, the arbitration process is done based on the full 32-bit word. However, the actual transmitted ID is 11 bits for standard frames and 29 bits for extended frames.<br>This field can be written only in Freeze mode; the module blocks it in other modes.<br>This bit is provided for backward compatibility with legacy applications.<br>0b - Disable<br>1b - Enable |
| 12<br>AEN | Abort Enable<br>Enables the TX abort mechanism. This mechanism guarantees a safe procedure for aborting a pending transmission, so that no frame is sent in the CAN bus without notification. This field can be written only in Freeze mode; the module blocks it in other modes.<br>**NOTE**<br>When this field is 1, only use the abort mechanism (see Transmission abort mechanism) to update message buffers configured for transmission.<br>**CAUTION**<br>Writing the abort code into RX message buffers can cause unpredictable results when this field is 1. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |
| 11<br>FDEN | CAN FD Operation Enable<br>Enables the CAN with flexible data rate (CAN FD) operation. This field can be written in Freeze mode only. FlexCAN can receive and transmit messages in CAN 2.0 format. If this field is enabled, FlexCAN can also receive and transmit messages in CAN FD format.<br>FlexCAN can transmit FD frame format according to ISO 11898-1:2015.<br>**NOTE**<br>If the value of this field is 1, the Legacy RX FIFO Enable (MCR[RFEN]) field cannot be 1.<br>0b - Disable<br>1b - Enable |
| 10<br>— | Reserved |
| 9-8<br>IDAM | ID Acceptance Mode<br>Identifies the format of the Legacy RX FIFO ID filter table elements. This field configures all elements of the table at the same time; they are all the same format. See legacy RX FIFO structure. This field can be written only in Freeze mode; the module blocks it in other modes.<br>00b - Format A: One full ID (standard and extended) per ID filter table element.<br>01b - Format B: Two full standard IDs or two partial 14-bit (standard and extended) IDs per ID filter table element.<br>10b - Format C: Four partial 8-bit standard IDs per ID filter table element.<br>11b - Format D: All frames rejected. |
| 7<br>— | Reserved |
| 6-0<br>MAXMB | Number of the Last Message Buffer<br>Defines the number of the last message buffer that takes part in the matching and arbitration processes. The reset value (0Fh) is equivalent to a 16-MB configuration. This field can be written only in Freeze mode; the module blocks it in other modes.<br>**NOTE**<br>You must write a value smaller than or equal to the number of available message buffers to this field, as described in FlexCAN memory partition for CAN FD.<br>Additionally, the MAXMB value must consider the region of message buffers occupied by Legacy RX FIFO and its ID filters table space defined by CTRL2[RFFN]. MAXMB also impacts the definition of the minimum number of peripheral clocks per CAN bit, as described in Table 438. |

### 70.6.2.3 Control 1 (CTRL1)

#### Offset

| Register | Offset |
|---|---|
| CTRL1 | 4h |

#### Function

Contains specific FlexCAN control features related to the CAN bus. These features include bit rate, programmable sampling point within an RX bit, Loopback mode, Listen-Only mode, Bus Off recovery behavior, and interrupt enabling (Bus-Off, Error, Warning). It also determines the division factor for the clock prescaler.

The CAN bit timing variables (CTRL1[PRESDIV], CTRL1[PROPSEG], CTRL1[PSEG1], CTRL1[PSEG2], and CTRL1[RJW]) can also be configured in CAN Bit Timing (CBT), which extends the range of all these variables. If CBT[BTF] = 1, CTRL1[PRESDIV], CTRL1[PROPSEG], CTRL1[PSEG1], CTRL1[PSEG2], and CTRL1[RJW] become read-only.

If CTRL2[BTE] = 1, CTRL1[PRESDIV], CTRL1[PROPSEG], CTRL1[PSEG1], CTRL1[PSEG2], and CTRL1[RJW] are not used by the module. Instead, these fields are read as zero, and a write operation to them has no effect.

**NOTE**

When the CAN FD feature is enabled, do not use CTRL1[PRESDIV], CTRL1[PROPSEG], CTRL1[PSEG1], CTRL1[PSEG2], and CTRL1[RJW] for CAN bit timing. Instead use CBT[EPRESDIV], CBT[ERJW], CBT[EPSEG1], CBT[EPSEG2], and CBT[EPROPSEG].

The CAN bit variables in CTRL1 and in CBT are stored in the same internal register.

Soft reset does not affect the contents of this register.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PRESDIV | | | | | | | | RJW | | PSEG1 | | PSEG2 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BOFF MSK | ERRM SK | Reserv ed | LPB | TWRN MSK | RWRN MSK | 0 | 0 | SMP | BOFF REC | TSYN | LBUF | LOM | PROPSEG | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-24<br>PRESDIV | Prescaler Division Factor<br>Determines the ratio between the PE clock frequency and the serial clock (Sclock) frequency. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency. The maximum value of this field is FFh, which gives a minimum Sclock frequency |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | equal to the PE clock frequency divided by 256. See Protocol timing for more information. This field can be written only in Freeze mode; the module blocks it in other modes.<br>Sclock frequency = PE clock frequency ÷ (PRESDIV + 1). |
| 23-22<br>RJW | Resync Jump Width<br>Defines the maximum number of time quanta that one resynchronization can change a bit time. One time quantum is equal to one Sclock period. The valid programmable values are 0–3. See Protocol timing for more information. This field can be written only in Freeze mode; the module blocks it in other modes.<br>Resync Jump Width = RJW + 1. |
| 21-19<br>PSEG1 | Phase Segment 1<br>Defines the length of phase segment 1 in the bit time. The valid programmable values are 0–7. See Protocol timing for more information. This field can be written only in Freeze mode; the module blocks it in other modes.<br>Phase Buffer Segment 1 = (PSEG1 + 1) × Time Quanta. |
| 18-16<br>PSEG2 | Phase Segment 2<br>Defines the length of phase segment 2 in the bit time. The valid programmable values are 1–7. See Protocol timing for more information. This field can be written only in Freeze mode; the module blocks it in other modes.<br>Phase Buffer Segment 2 = (PSEG2 + 1) × Time Quanta. |
| 15<br>BOFFMSK | Bus Off Interrupt Mask<br>Provides a mask for the Bus Off interrupt ESR1[BOFFINT].<br>0b - Interrupt disabled<br>1b - Interrupt enabled |
| 14<br>ERRMSK | Error Interrupt Mask<br>Provides a mask for the Error interrupt ESR1[ERRINT].<br>0b - Interrupt disabled<br>1b - Interrupt enabled |
| 13<br>— | Reserved<br>**NOTE**<br>Writeable only if the module is disabled. Otherwise the access type is read-only. |
| 12<br>LPB | Loopback Mode<br>Configures FlexCAN to operate in Loopback mode. In this mode, FlexCAN performs an internal loopback that can be used for self-test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The RX CAN input pin is ignored and the TX CAN output goes to the recessive state (logic 1). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field. It generates an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. This field can be written only in Freeze mode; the module blocks it in other modes.<br>**NOTE**<br>In this mode, MCR[SRXDIS] cannot become 1, because it would impede the self-reception of a transmitted message.<br>**NOTE**<br>FDCTRL[TDCEN] and MCR[ETDCEN] must be 0 when this field is 1.<br>0b - Disabled<br>1b - Enabled |
| 11<br>TWRNMSK | TX Warning Interrupt Mask<br>Provides a mask for the TX Warning interrupt associated with the ESR1[TWRNINT] flag. When MCR[WRNEN] = 0, this field is read as 0. This field can be written only if MCR[WRNEN] = 1.<br>0b - Disabled<br>1b - Enabled |
| 10<br>RWRNMSK | RX Warning Interrupt Mask<br>Provides a mask for the RX Warning interrupt associated with the ESR1[RWRNINT] flag. When MCR[WRNEN] = 0, this field is read as 0. This field can be written only if MCR[WRNEN] = 1.<br>0b - Disabled<br>1b - Enabled |
| 9<br>— | Reserved |
| 8<br>— | Reserved |
| 7<br>SMP | CAN Bit Sampling<br>Determines the sampling mode of CAN bits at the RX input. This field can be written in Freeze mode only; the module blocks it in other modes.<br>**NOTE**<br>For proper operation, to write 1 to this field, you must guarantee a minimum value of two time quanta in CTRL1[PSEG1] (or CBT[EPSEG1]).This bit cannot become 1 when CAN FD is enabled (MCR[FDEN] = 1).<br>0b - One sample is used to determine the bit value. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Three samples are used to determine the value of the received bit: the regular one (sample point) and two preceding samples. A majority rule is used. |
| 6<br>BOFFREC | Bus Off Recovery<br>Determines how FlexCAN recovers from Bus Off state. If 0, automatic recovering from Bus Off state occurs according to the CAN Specification 2.0B. If 1, automatic recovering from Bus Off is disabled. The module remains in Bus Off state until you write 1 to this field.<br>If this field becomes 0 before 128 sequences of 11 recessive bits are detected on the CAN bus, Bus Off recovery happens as if this field had never become 1. If this field becomes 0 after 128 sequences of 11 recessive bits occurred, FlexCAN resynchronizes to the bus. It waits for 11 recessive bits before joining the bus.<br>After this field becomes 0, it can become 1 again during Bus Off, but it will only be effective the next time the module enters Bus Off. If this field becomes 0 when the module is in Bus Off, writing 1 to this field is not effective for the current Bus Off recovery.<br>**NOTE**<br>See Bus Off in the CAN Protocol standard (ISO 11898-1:2015) for details.<br>0b - Enabled<br>1b - Disabled |
| 5<br>TSYN | Timer Sync<br>Enables a mechanism that resets the free-running timer each time a message is received in message buffer 0. This feature provides the means to synchronize multiple FlexCAN stations with a special "SYNC" message (that is, global network time). If MCR[RFEN] = 1 (Legacy RX FIFO enabled), the first available message buffer, according to CTRL2[RFFN], is used for timer synchronization instead of MB0. This field can be written in Freeze mode only; the module blocks it in other modes.<br>0b - Disable<br>1b - Enable |
| 4<br>LBUF | Lowest Buffer Transmitted First<br>Determines the ordering mechanism for message buffer transmission. When 1, MCR[LPRIOEN] does not affect the priority arbitration. This field can be written in Freeze mode only; the module blocks it in other modes.<br>0b - Buffer with highest priority is transmitted first.<br>1b - Lowest number buffer is transmitted first. |
| 3<br>LOM | Listen-Only Mode<br>Configures FlexCAN to operate in Listen-Only mode. In this mode, transmission is disabled, all error counters described in Error Counter (ECR) are frozen, and the module operates in CAN Error Passive mode. Only messages acknowledged by another CAN station are received. If FlexCAN detects an unacknowledged message, it flags a BIT0 error without changing the receive error counter (ECR[RXERRCNT]), as if it is trying to acknowledge the message. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | FlexCAN acknowledges Listen-Only mode when ESR1[FLTCONF] = 1, indicating the Error Passive state. There can be some delay between the Listen-Only mode request and its acknowledgment.<br>This field can be written in Freeze mode only; the module blocks it in other modes.<br>0b - Listen-Only mode is deactivated.<br>1b - FlexCAN module operates in Listen-Only mode. |
| 2-0<br>PROPSEG | Propagation Segment<br>Defines the length of the propagation segment in the bit time. The valid programmable values are 0–7. This field can be written only in Freeze mode; the module blocks it in other modes.<br>Propagation segment time = (PROPSEG + 1) × Time Quanta.<br>One Time Quantum = one Sclock period. |

### 70.6.2.4 Free-Running Timer (TIMER)

#### Offset

| Register | Offset |
|---|---|
| TIMER | 8h |

#### Function

Represents a 16-bit free-running counter that the CPU can read and write. The timer starts from 0h after reset, counts linearly to FFFFh, and wraps around.

When CTRL2[TIMER_SRC] = 1, an external time tick continuously increments the timer. The time tick must be synchronous to the peripheral clock, with a minimum pulse width of one clock cycle.

When CTRL2[TIMER_SRC] = 0, the CAN bit clock increments the timer, which defines the baud rate on the CAN bus. During a message transmission or reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate. The timer is not incremented during Disable and Freeze modes.

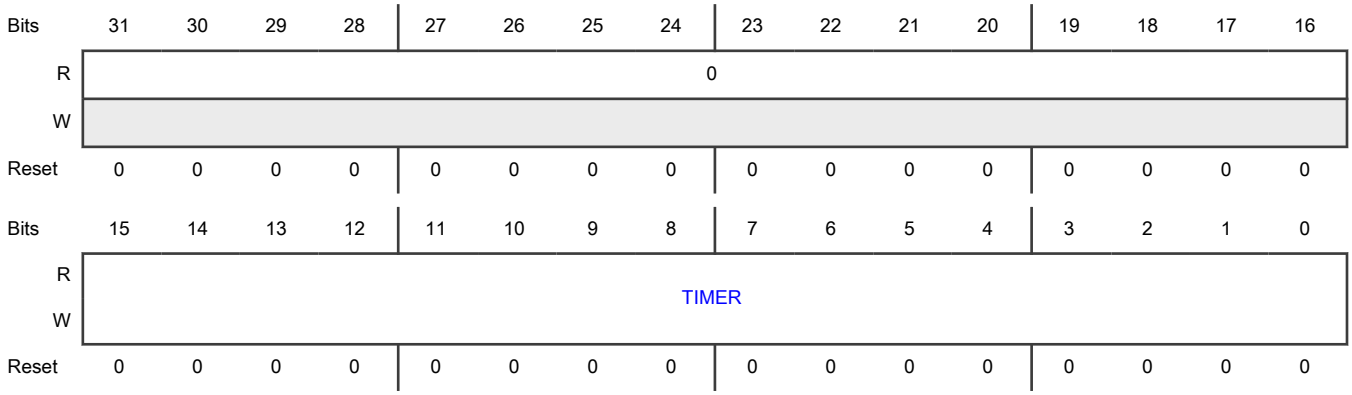
The timer value is captured when the second bit of the identifier field of any frame is on the CAN bus. This captured value is written into the timestamp entry in a message buffer after a successful reception or transmission of a message.

If CTRL1[TSYN] = 1, the timer is reset whenever a message is received in the first available message buffer, according to CTRL2[RFFN].

The CPU can write to this register anytime. However, if the write occurs simultaneously with the timer being reset by a reception in the first message buffer, the write value is discarded.

Reading this register affects the message buffer unlocking procedure (see Message buffer lock mechanism).

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>TIMER | Timer Value<br>Contains the free-running counter value. |

## 70.6.2.5 RX Message Buffers Global Mask (RXMGMASK)

### Offset

| Register | Offset |
|---|---|
| RXMGMASK | 10h |

### Function

Masks the filter bits of all RX message buffers, excluding MB14 and MB15, which have individual mask registers.

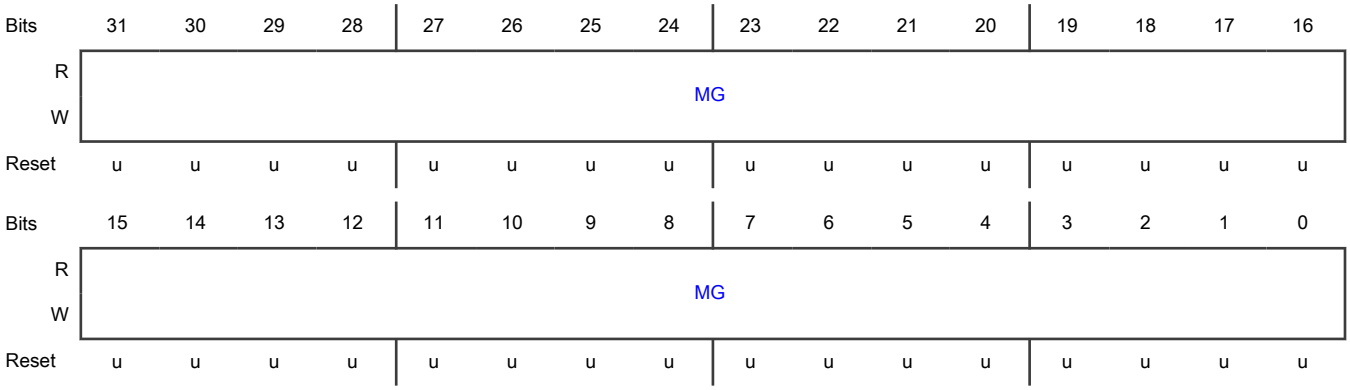
This register is located in RAM.

RXMGMASK is provided for legacy application support.

- When MCR[IRMQ] is 0, RXMGMASK is always in effect. The bits in RXMGMASK[MG] mask the MB filter bits.
- When MCR[IRMQ] is 1, RXMGMASK has no effect. The bits in RXMGMASK[MG] do not mask the MB filter bits.

This register can only be written in Freeze mode; the module blocks it in other modes.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>MG | Global Mask for RX Message Buffers<br>Masks the message buffer filter bits. The alignment with the ID word of the message buffer is imperfect. The two most significant MG bits affect the fields RTR and IDE, which are located in the Control and Status word of the MB. The following table shows which MG bits mask each MB filter field. |

| SMB[RTR] [1] | CTRL2[RRS] | CTRL2[EACEN] | Message buffer filter fields | | | |
|---|---|---|---|---|---|---|
| | | | MB[RTR] | MB[IDE] | MB[ID] | Reserved |
| 0 | - | 0 | Note [2] | Note [3] | MG[28:0] | MG[31:29] |
| 0 | - | 1 | MG[31] | MG[30] | MG[28:0] | MG[29] |
| 1 | 0 | - | - | - | - | MG[31:0] |
| 1 | 1 | 0 | - | - | MG[28:0] | MG[31:29] |
| 1 | 1 | 1 | MG[31] | MG[30] | MG[28:0] | MG[29] |

1. RTR bit of the incoming frame. It is saved into an auxiliary MB called RX serial message buffer (RX SMB).
2. If CTRL2[EACEN] is 0, the RTR bit of MB is never compared with the RTR bit of the incoming frame.
3. If CTRL2[EACEN] is 0, the IDE bit of MB is always compared with the IDE bit of the incoming frame.

0b - The corresponding bit in the filter is "don't care."

1b - The corresponding bit in the filter is checked.

## 70.6.2.6 Receive 14 Mask (RX14MASK)

### Offset

| Register | Offset |
|---|---|
| RX14MASK | 14h |

### Function

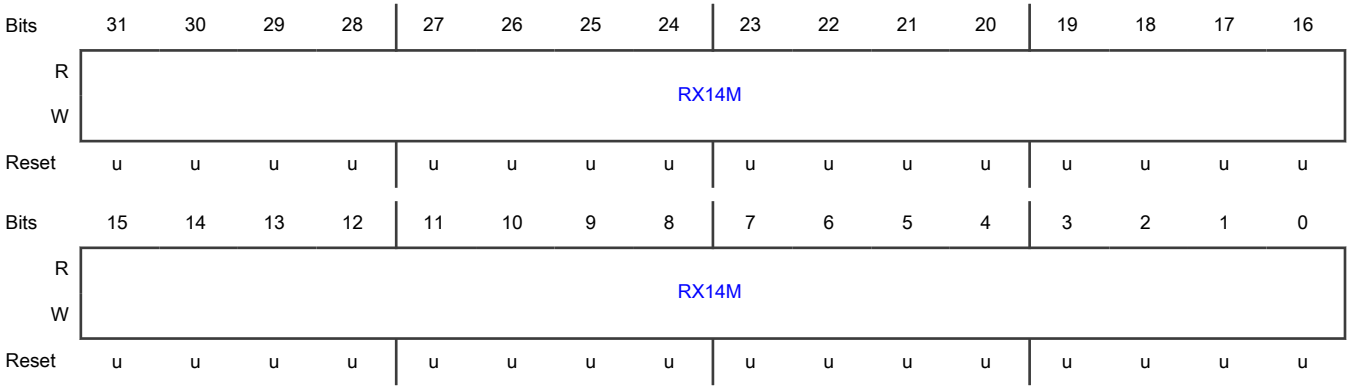
Masks the filter fields of MB14.

This register is located in RAM.

RX14MASK is provided for legacy application support. When MCR[IRMQ] = 1, RX14MASK has no effect.

This register can only be programmed when the module is in Freeze mode; the module blocks it in other modes.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>RX14M | RX Buffer 14 Mask Bits<br>Masks the corresponding MB14 filter field in the same way that RX Message Buffers Global Mask (RXMGMASK) masks the filters of the other message buffers.<br>0b - The corresponding bit in the filter is "don't care."<br>1b - The corresponding bit in the filter is checked. |

## 70.6.2.7 Receive 15 Mask (RX15MASK)

### Offset

| Register | Offset |
|---|---|
| RX15MASK | 18h |

### Function

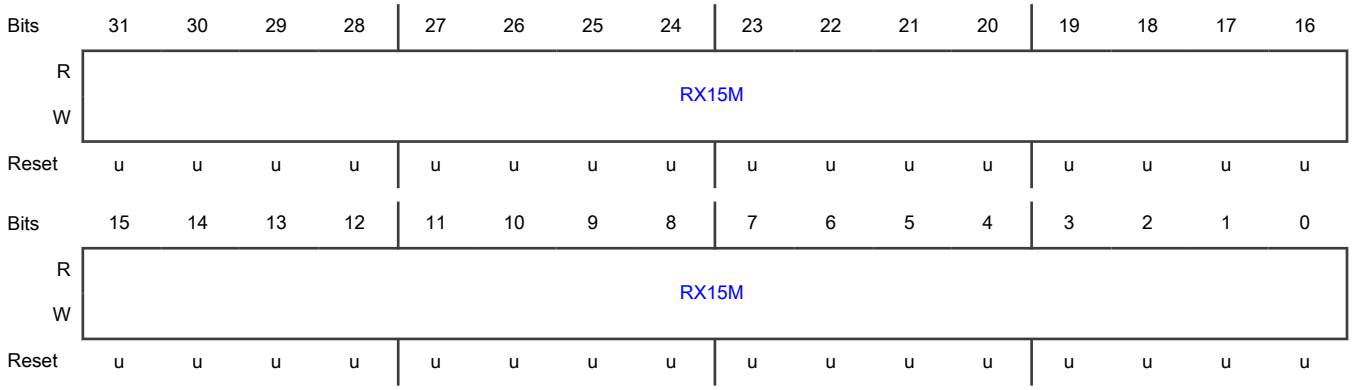
Masks the filter fields of MB15.

This register is located in RAM.

RX15MASK is provided for legacy application support. When MCR[IRMQ] = 1, RX15MASK has no effect.

This register can be programmed only when the module is in Freeze mode; the module blocks it in other modes.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>RX15M | RX Buffer 15 Mask Bits<br>Masks the corresponding MB15 filter field in the same way that RX Message Buffers Global Mask (RXMGMASK) masks the filters of other message buffers.<br>0b - The corresponding bit in the filter is "don't care."<br>1b - The corresponding bit in the filter is checked. |

## 70.6.2.8 Error Counter (ECR)

### Offset

| Register | Offset |
|---|---|
| ECR | 1Ch |

### Function

Contains error counters for received and transmitted messages.

TXERRCNT and RXERRCNT consider all errors in both CAN FD and non-FD message formats. TXERRCNT_FAST and RXERRCNT_FAST count only the errors that occur in the data phase of CAN FD frames that have BRS = 1.

The Fault Confinement state (ESR1[FLTCONF]) is updated based on TXERRCNT and RXERRCNT counters only. The rules for increasing and decreasing these counters are described in the CAN protocol and are entirely implemented in FlexCAN.

The basic rules for FlexCAN bus state transitions are:

- If the value of TXERRCNT or RXERRCNT becomes greater than or equal to 128, ESR1[FLTCONF] is updated to reflect Error Passive state.
- If the state of FlexCAN is Error Passive, and TXERRCNT or RXERRCNT decrements to a value less than 128 when the other already satisfies this condition, ESR1[FLTCONF] is updated to reflect Error Active state.
- If the value of TXERRCNT becomes greater than 255, ESR1[FLTCONF] is updated to reflect Bus Off state, and an interrupt may be issued. The value of TXERRCNT is then reset to zero.
- If FlexCAN is in Bus Off, TXERRCNT is cascaded with another internal counter to count the occurrences of 11 consecutive recessive bits on the bus. TXERRCNT is reset to zero. It counts in a manner where the internal counter

counts 11 such bits and then wraps around when incrementing the TXERRCNT. When TXERRCNT reaches the value of 128, ESR1[FLTCONF] is updated to Error Active, and both error counters are reset to zero. Upon any instance of a dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the TXERRCNT value. The TXERRCNT_FAST counter is frozen during Bus Off.

- If only one node is operating during system startup, its TXERRCNT increases upon each attempted message transmission, as a result of acknowledge errors (indicated by ESR1[ACKERR]). After the transition to Error Passive state, TXERRCNT no longer increments upon acknowledge errors. The chip never goes into the Bus Off state.
- If RXERRCNT increases to a value greater than 127, it is not incremented further, even if more errors are detected when being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to return to the Error Active state.
- TXERRCNT_FAST and RXERRCNT_FAST error counter values increment and decrement based on errors detected only in the data phase of CAN FD frames that have BRS = 1. These counters follow the same increment and decrement rules as TXERRCNT and RXERRCNT. These counters do not wrap around and get stuck at their maximum value (255). They stop counting and keep their values frozen when FlexCAN is in the Bus Off state. They are reset when FlexCAN leaves the Bus Off state and resume counting after FlexCAN returns to the Error Active state.

**NOTE**

See Fault confinement in the CAN Protocol standard (ISO 11898-1:2015) for details.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-24<br>RXERRCNT_FAST | Receive Error Counter for Fast Bits<br>Counts errors detected in the data phase of received CAN FD messages that have BRS = 1. This field is read-only except in Freeze mode, when the CPU can write an 8-bit zero value only. |
| 23-16<br>TXERRCNT_FAST | Transmit Error Counter for Fast Bits<br>Counts errors detected in the data phase of transmitted CAN FD messages that have BRS = 1. This field is read-only except in Freeze mode, when the CPU can write an 8-bit zero value only. |
| 15-8<br>RXERRCNT | Receive Error Counter<br>Counts all errors detected in received messages. This field is read-only except in Freeze mode, when the CPU can write to it. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 7-0<br>TXERRCNT | Transmit Error Counter<br>Counts all errors detected in transmitted messages. This field is read-only except in Freeze mode, when the CPU can write to it. |

### 70.6.2.9 Error and Status 1 (ESR1)

#### Offset

| Register | Offset |
|---|---|
| ESR1 | 20h |

#### Function

Reports various error conditions detected in the reception and transmission of a CAN frame. This register provides status information about the chip, and is the source of some interrupts to the CPU. The reported error conditions are:

**NOTE**

Reading host can clear these fields.

- Errors detected in CAN frames of any format:
  - BIT1ERR
  - BIT0ERR
  - ACKERR
  - CRCERR
  - FRMERR
  - STFERR
- Errors detected in the data phase of CAN FD frames with the BRS bit set only:
  - BIT1ERR_FAST
  - BIT0ERR_FAST
  - CRCERR_FAST
  - FRMERR_FAST
  - STFERR_FAST

One or more error flags may report an error detected in a single CAN frame. To account for more error events occurring in subsequent frames when the CPU does not attempt to read this register, error reporting is cumulative.

Status flags:

- TXWRN
- RXWRN
- IDLE
- TX

- FLTCONF
- RX
- SYNCH

Interrupt flags:

- BOFFINT
- BOFFDONEINT
- ERRINT
- ERRINT_FAST
- TWRNINT
- RWRNINT

The CPU should follow this procedure when servicing interrupt requests generated by these flags:

1. Read this register to capture all error condition and status flags. This action clears the respective flags that were set since the last read access.
2. Write 1 to clear the interrupt flag that triggered the interrupt request.
3. Write 1 to clear the ERROVR flag, if it is set.

Starting from all error flags cleared, a first error event sets either ERRINT or ERRINT_FAST (provided the corresponding mask bit is 1). If other error events in subsequent frames occur before the CPU serves the interrupt request, the ERROVR flag is set to indicate that errors from different frames have accumulated.

**Table 444. CAN bus status**

| SYNCH | IDLE | TX | RX | FlexCAN state |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Not synchronized to CAN bus |
| 1 | 1 | X | X | Idle |
| 1 | 0 | 1 | 0 | Transmitting |
| 1 | 0 | 0 | 1 | Receiving |

**NOTE**

See Fault confinement in the CAN Protocol standard (ISO 11898-1:2015) for details.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BIT1ERR... | BIT0ERR... | 0 | CRCERR_... | FRMERR_... | STFERR_... | 0 | | 0 | 0 | ERROVR | ERRINT_... | BOFFDON... | SYNCH | TWRNINT | RWRNINT |
| W | | | | | | | | | | | W1C | W1C | W1C | | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BIT1ERR | BIT0ERR | ACKERR | CRCERR | FRMERR | STFERR | TXWRN | RXWRN | IDLE | TX | FLTCONF | | RX | BOFFINT | ERRINT | 0 |
| W | | | | | | | | | | | | | | W1C | W1C | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>BIT1ERR_FAST | Fast Bit1 Error Flag<br>Indicates when an inconsistency occurs between the transmitted and the received bit in the data phase of CAN FD frames that have BRS = 1.<br>After a read operation, the field's value clears to 0.<br>0b - No such occurrence.<br>1b - At least one bit transmitted as recessive is received as dominant. |
| 30<br>BIT0ERR_FAST | Fast Bit0 Error Flag<br>Indicates when an inconsistency occurs between the transmitted and the received bit in the data phase of CAN FD frames that have BRS = 1.<br>After a read operation, the field's value clears to 0.<br>0b - No such occurrence.<br>1b - At least one bit transmitted as dominant is received as recessive. |
| 29<br>— | Reserved |
| 28<br>CRCERR_FAST | Fast Cyclic Redundancy Check Error Flag<br>Indicates that the receiver node has detected a CRC error in the CRC field of CAN FD frames that have BRS = 1. This error means that the calculated CRC is different from the received CRC.<br>After a read operation, the field's value clears to 0.<br>0b - No such occurrence.<br>1b - A CRC error occurred since last read of this register. |
| 27<br>FRMERR_FAST | Fast Form Error Flag<br>Indicates whether the receiver node has detected a form error in the data phase of CAN FD frames that have BRS = 1. This error means that a fixed-form bit field contains at least one illegal bit.<br>After a read operation, the field's value clears to 0.<br>0b - No such occurrence.<br>1b - A form error occurred since last read of this register. |
| 26<br>STFERR_FAST | Fast Stuffing Error Flag<br>Indicates that a stuffing error has been detected in the data phase of CAN FD frames that have BRS = 1.<br>After a read operation, the field's value clears to 0.<br>0b - No such occurrence.<br>1b - A stuffing error occurred since last read of this register. |
| 25-24<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 23<br>— | Reserved |
| 22<br>— | Reserved |
| 21<br>ERROVR | Error Overrun Flag<br>Indicates that an error condition occurred when any error flag is already set.<br>0b - No overrun<br>1b - Overrun |
| 20<br>ERRINT_FAST | Fast Error Interrupt Flag<br>Indicates that at least one error flag detected in the data phase of CAN FD frames that have BRS = 1 (BIT1ERR_FAST, BIT0ERR_FAST, CRCERR_FAST, FRMERR_FAST, or STFERR_FAST) is set. If CTRL2[ERRMSK_FAST] = 1, an interrupt is generated to the CPU.<br>0b - No such occurrence.<br>1b - Error flag set in the data phase of CAN FD frames that have BRS = 1. |
| 19<br>BOFFDONEINT | Bus Off Done Interrupt Flag<br>Indicates whether ECR[TXERRCNT] has finished counting 128 occurrences of 11 consecutive recessive bits on the CAN bus and is ready to leave Bus Off. If CTRL2[BOFFDONEMSK] = 1, an interrupt is generated to the CPU.<br>0b - No such occurrence<br>1b - FlexCAN module has completed Bus Off process. |
| 18<br>SYNCH | CAN Synchronization Status Flag<br>Indicates whether FlexCAN is synchronized to the CAN bus and able to participate in the communication process. FlexCAN sets and clears this flag. See the table in Error and Status 1 (ESR1).<br>0b - Not synchronized<br>1b - Synchronized |
| 17<br>TWRNINT | TX Warning Interrupt Flag<br>Indicates whether TX error counter changed from less than 96 to greater than or equal to 96.<br>If MCR[WRNEN] = 1, this flag is set when the TXWRN flag transitions from 0 to 1, meaning that the TX error counters reached 96. If CTRL1[TWRNMSK] = 1, an interrupt is sent to the CPU. When MCR[WRNEN] = 0, this flag is masked. The CPU must clear this flag before writing 0 to MCR[WRNEN]. Otherwise, this flag is set when MCR[WRNEN] = 1 again. Writing 0 has no effect.<br>This flag is not generated when in the Bus Off state. This flag is not updated during Freeze mode.<br>0b - No such occurrence<br>1b - TX error counter changed from less than 96 to greater than or equal to 96. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 16<br>RWRNINT | RX Warning Interrupt Flag<br>Indicates whether the RX error counter changed from less than 96 to greater than or equal to 96.<br>If MCR[WRNEN] = 1, this flag is set when the RXWRN flag transitions from 0 to 1, meaning that the RX error counters reached 96. If CTRL1[RWRNMSK] = 1, an interrupt is sent to the CPU. When MCR[WRNEN] = 0, this flag is masked. The CPU must clear this flag before writing 0 to MCR[WRNEN]. Otherwise, this flag is set when MCR[WRNEN] = 1 again. Writing 0 has no effect.<br>This flag is not updated during Freeze mode.<br>0b - No such occurrence<br>1b - RX error counter changed from less than 96 to greater than or equal to 96. |
| 15<br>BIT1ERR | Bit1 Error Flag<br>Indicates when an inconsistency occurs between the transmitted and the received bit in a non-CAN FD message or in the arbitration or data phase of a CAN FD message.<br>**NOTE**<br>A transmitter does not set this flag for an arbitration field or ACK slot. It is not set for a node sending an error passive flag that detects dominant bits.<br>After a read operation, the field's value clears to 0.<br>0b - No such occurrence.<br>1b - At least one bit sent as recessive is received as dominant. |
| 14<br>BIT0ERR | Bit0 Error Flag<br>Indicates when an inconsistency occurs between the transmitted and the received bit in a non-CAN FD message or in the arbitration or data phase of a CAN FD message.<br>After a read operation, the field's value clears to 0.<br>0b - No such occurrence.<br>1b - At least one bit sent as dominant is received as recessive. |
| 13<br>ACKERR | Acknowledge Error Flag<br>Indicates whether the transmitter node has detected an acknowledge error. This error means that a dominant bit has not been detected during the ACK SLOT.<br>After a read operation, the field's value clears to 0.<br>0b - No error<br>1b - Error occurred since last read of this register. |
| 12<br>CRCERR | Cyclic Redundancy Check Error Flag<br>Indicates whether the receiver node has detected a cyclic redundancy check (CRC) error either in a non-FD message or in the arbitration or data phase of a frame in CAN FD format. This error means that the calculated CRC is different from the received.<br>After a read operation, the field's value clears to 0. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No error<br>1b - Error occurred since last read of this register. |
| 11<br>FRMERR | Form Error Flag<br>Indicates whether a form error has been detected in a non-FD message or in the arbitration or data phase of an FD message by the receiver node. This error means that a fixed-form field contains at least one illegal bit.<br>After a read operation, the field's value clears to 0.<br>0b - No error<br>1b - Error occurred since last read of this register. |
| 10<br>STFERR | Stuffing Error Flag<br>Indicates that a stuffing error has been detected in a non-FD message or in the arbitration or data phase of an FD message by the receiver node.<br>After a read operation, the field's value clears to 0.<br>0b - No error<br>1b - Error occurred since last read of this register. |
| 9<br>TXWRN | TX Error Warning Flag<br>Indicates when repetitive errors occur during message transmission. Only the value of ECR[TXERRCNT] affects this flag. This flag is not updated during Freeze mode.<br>After a read operation, the field's value clears to 0.<br>0b - No such occurrence.<br>1b - TXERRCNT is 96 or greater. |
| 8<br>RXWRN | RX Error Warning Flag<br>Indicates when repetitive errors occur during message reception. Only the value of ECR[RXERRCNT] affects this flag. This flag is not updated during Freeze mode.<br>After a read operation, the field's value clears to 0.<br>0b - No such occurrence.<br>1b - RXERRCNT is greater than or equal to 96. |
| 7<br>IDLE | Idle<br>Indicates whether CAN bus is in IDLE state. See Table 444.<br>0b - Not IDLE<br>1b - IDLE |
| 6<br>TX | FlexCAN In Transmission<br>Indicates whether FlexCAN is transmitting a message. See Table 444. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Not transmitting<br>1b - Transmitting |
| 5-4<br>FLTCONF | Fault Confinement State<br>Indicates the confinement state of FlexCAN.<br>If CTRL1[LOM] = 1, after a delay that depends on the CAN bit timing, this field indicates Error Passive. The same delay affects the way that this field reflects an update to ECR register by the CPU. It may be necessary to wait up to one CAN bit time for coherence to be restored.<br>Soft reset affects this field, but if CTRL1[LOM] = 1, its reset value lasts for only one CAN bit. After that time, this field reports Error Passive.<br>00b - Error Active<br>01b - Error Passive<br>1xb - Bus Off |
| 3<br>RX | FlexCAN in Reception Flag<br>Indicates whether FlexCAN is receiving a message. See the table in Error and Status 1 (ESR1).<br>0b - Not receiving<br>1b - Receiving |
| 2<br>BOFFINT | Bus Off Interrupt Flag<br>Indicates whether FlexCAN has entered Bus Off state. If CTRL1[BOFFMSK]) = 1, an interrupt is generated to the CPU. Writing 0 to this field has no effect.<br>0b - No such occurrence.<br>1b - FlexCAN module entered Bus Off state. |
| 1<br>ERRINT | Error Interrupt Flag<br>Indicates that at least one of the error flags (ESR1[BIT1ERR], ESR1[BIT0ERR], ESR1[ACKERR], ESR1[CRCERR], ESR1[FRMERR], or ESR1[STFERR]) is set. If the corresponding mask CTRL1[ERRMSK] = 1, an interrupt is generated to the CPU. Writing 0 to this field has no effect.<br>0b - No such occurrence.<br>1b - Indicates setting of any error flag in the Error and Status register. |
| 0<br>— | Reserved |

### 70.6.2.10 Interrupt Masks 2 (IMASK2)

#### Offset

| Register | Offset |
|---|---|
| IMASK2 | 24h |

#### Function

Masks interrupt flags. This register allows any of the 32 message buffer interrupts to be enabled or disabled for MB63–MB32. It contains one interrupt mask bit per buffer. This configuration allows the CPU to determine which buffer generates an interrupt after a successful transmission or reception when the corresponding Interrupt Flags 2 (IFLAG2) flag is set.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| FLEXCAN_0 | IMASK2 | — |
| FLEXCAN_1 | IMASK2 | — |
| FLEXCAN_2 | IMASK2 | — |
| FLEXCAN_3 | — | IMASK2 |
| FLEXCAN_4 | — | IMASK2 |
| FLEXCAN_5 | — | IMASK2 |

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BUF63TO32M | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BUF63TO32M | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-0 | Buffer MBi Mask |

*Table continues on the next page...*

| Field | Function |
|---|---|
| BUF63TO32M | Masks the corresponding FlexCAN message buffer interrupt for MB63–MB32.<br>**NOTE**<br>If the corresponding Interrupt Flags 2 (IFLAG2) flag is set, writing 1 or 0 to a field in IMASK2 can set or clear an interrupt request.<br>0b - The corresponding buffer interrupt is disabled.<br>1b - The corresponding buffer interrupt is enabled. |

### 70.6.2.11 Interrupt Masks 1 (IMASK1)

#### Offset

| Register | Offset |
|---|---|
| IMASK1 | 28h |

#### Function

Masks interrupt flags. This register allows any of the 32 message buffer interrupts to be enabled or disabled for MB31–MB0. It contains one interrupt mask bit per buffer. This configuration allows the CPU to determine which buffer generates an interrupt after a successful transmission or reception when the corresponding Interrupt Flags 1 (IFLAG1) flag is set.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BUF31TO0M | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | BUF31TO0M | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-0<br>BUF31TO0M | Buffer MBi Mask<br>Enables or disables the corresponding FlexCAN message buffer interrupt for MB31–MB0.<br>**NOTE**<br>If the corresponding Interrupt Flags 1 (IFLAG1) flag is set, writing 1 or 0 to a field in IMASK1 can set or clear an interrupt request. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | 0b - The corresponding buffer interrupt is disabled.<br>1b - The corresponding buffer interrupt is enabled. |

### 70.6.2.12 Interrupt Flags 2 (IFLAG2)

#### Offset

| Register | Offset |
|---|---|
| IFLAG2 | 2Ch |

#### Function

Contains the flags for the 32 message buffer interrupts for MB63–MB32. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the respective flag in this register. If the corresponding Interrupt Masks 2 (IMASK2) bit is set, an interrupt is generated.

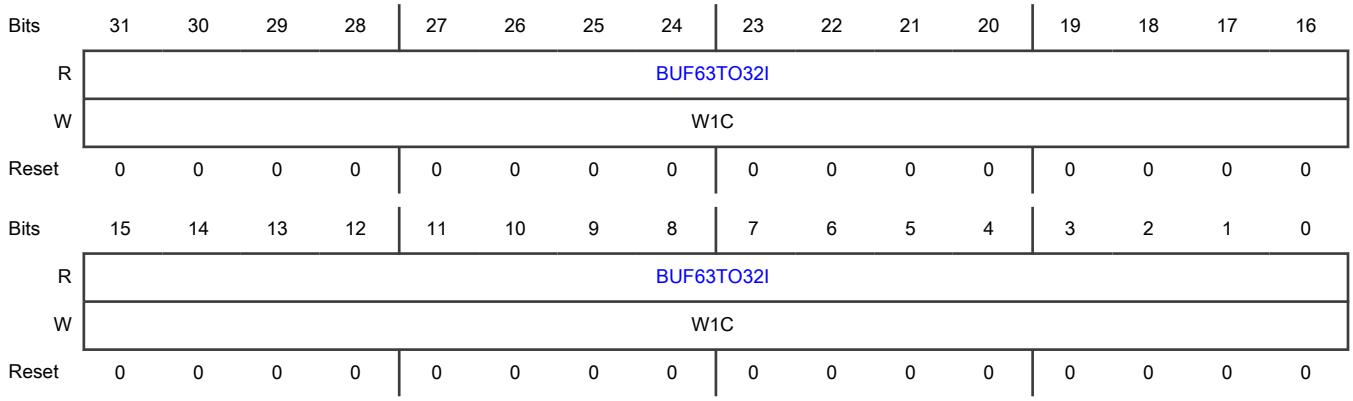
Before updating MCR[MAXMB], the CPU must service the IFLAG2 flags whose MB value is greater than the MAXMB to be updated. Otherwise, those flags remain set and are inconsistent with the number of message buffers available.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| FLEXCAN_0 | IFLAG2 | — |
| FLEXCAN_1 | IFLAG2 | — |
| FLEXCAN_2 | IFLAG2 | — |
| FLEXCAN_3 | — | IFLAG2 |
| FLEXCAN_4 | — | IFLAG2 |
| FLEXCAN_5 | — | IFLAG2 |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>BUF63TO32I | Buffer MBi Interrupt<br>Flags the corresponding FlexCAN message buffer interrupt for MB63–MB32.<br>0b - The corresponding buffer has no occurrence of successfully completed transmission or reception.<br>1b - The corresponding buffer has successfully completed transmission or reception. |

## 70.6.2.13 Interrupt Flags 1 (IFLAG1)

### Offset

| Register | Offset |
|---|---|
| IFLAG1 | 30h |

### Function

Contains the flags for the 32 message buffer interrupts for MB31–MB0. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG1 bit. If the corresponding Interrupt Masks 1 (IMASK1) bit is set, an interrupt is generated. There is an exception when DMA for Legacy RX FIFO is enabled, as described below.

The BUF7I–BUF5I flags also represent Legacy FIFO interrupts when the Legacy RX FIFO is enabled. When MCR[RFEN] is 1 and MCR[DMA] is 0, the function of the eight least significant interrupt flags changes:

- BUF7I, BUF6I, and BUF5I indicate operating conditions of the Legacy FIFO.
- BUF4I–BUF1I fields are reserved.
- BUF0I empties the Legacy FIFO.

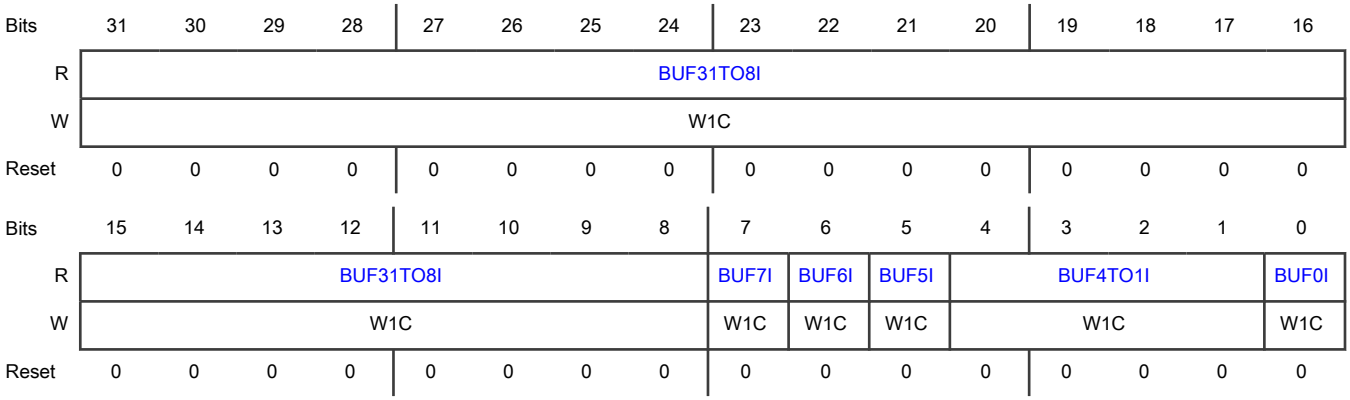
Before writing 1 to MCR[RFEN], the CPU must service the IFLAG flags set in the Legacy RX FIFO region; see Legacy RX FIFO. Otherwise, these IFLAG flags mistakenly show the related message buffers now belonging to Legacy FIFO as having contents to be serviced. When MCR[RFEN] is 0, the Legacy FIFO flags must be cleared. The same care must be taken when a CTRL2[RFFN] value is selected, extending Legacy RX FIFO filters beyond MB7. For example, when RFFN is 8h, Legacy RX FIFO filters occupy the MB23–MB0 range, and related IFLAG flags must be cleared.

When MCR[RFEN] and MCR[DMA] are 1 (DMA feature for Legacy RX FIFO enabled), the function of the eight least significant interrupt flags (BUF7I–BUF0I) changes to support DMA operation. BUF7I, BUF6I, and BUF4I–BUF1I are not used.

BUF5I indicates the operating condition of the Legacy FIFO, and BUF0I empties the Legacy FIFO. Moreover, BUF5I does not generate a CPU interrupt, but it does generate a DMA request. IMASK1 bits in the Legacy RX FIFO region are not considered when bit MCR[DMA] = 1. In addition, the CPU must not clear the BUF5I flag when DMA is enabled. Before writing 1 to MCR[DMA], the CPU must service the IFLAG flags set in the Legacy RX FIFO region. When MCR[DMA] is 0, the Legacy FIFO must be empty. Legacy FIFO must be disabled when MCR[FDEN] = 1.

Before updating MCR[MAXMB], the CPU must service the IFLAG1 flags whose MB value is greater than the MCR[MAXMB] to be updated. Otherwise, those flags remain set and are inconsistent with the number of message buffers available.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-8<br>BUF31TO8I | Buffer MBi Interrupt<br>Flags the corresponding FlexCAN message buffer interrupt for MB31–MB8.<br>0b - The corresponding buffer has no occurrence of successfully completed transmission or reception.<br>1b - The corresponding buffer has successfully completed transmission or reception. |
| 7<br>BUF7I | Buffer MB7 Interrupt or Legacy RX FIFO Overflow<br>Flags the interrupt for MB7 when MCR[RFEN] = 0 (Legacy RX FIFO disabled).<br>**NOTE**<br>When a CPU write changes the value of MCR[RFEN], FlexCAN clears this flag.<br>When MCR[RFEN] = 1, this flag represents a Legacy RX FIFO overflow. In this case, the flag indicates that a message was lost because the Legacy RX FIFO is full. When the Legacy RX FIFO is full and a message buffer captures the message, this flag is not set.<br>0b - No occurrence of MB7 completing transmission or reception, or no FIFO overflow.<br>1b - MB7 completed transmission or reception, or FIFO overflow. |
| 6<br>BUF6I | Buffer MB6 Interrupt or Legacy RX FIFO Warning<br>Flags the interrupt for MB6 when MCR[RFEN] = 0 (Legacy RX FIFO disabled). |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>When a CPU write changes the value of MCR[RFEN], FlexCAN clears this flag.<br><br>When MCR[RFEN] = 1, this flag represents a Legacy RX FIFO warning. In this case, the flag indicates when the number of unread messages within the Legacy RX FIFO is increased to five from four due to the reception of a new message. In other words, the Legacy RX FIFO is almost full.<br><br>If this flag is cleared when there are more than four unread messages, it does not set again until the number of unread messages in the Legacy RX FIFO decreases to four or fewer.<br><br>0b - No occurrence of MB6 completing transmission or reception, or FIFO not almost full.<br><br>1b - MB6 completed transmission or reception, or FIFO almost full. |
| 5<br><br>BUF5I | Buffer MB5 Interrupt or Frames available in Legacy RX FIFO<br><br>Flags the interrupt for MB5 when MCR[RFEN] = 0 (Legacy RX FIFO disabled).<br><br>**NOTE**<br>When a CPU write changes the value of MCR[RFEN], FlexCAN clears this flag.<br><br>When MCR[RFEN] = 1, the BUF5I flag represents frames available in Legacy RX FIFO. In this case, the flag indicates that at least one frame is available to be read from the Legacy RX FIFO.<br><br>When MCR[DMA] = 1, this flag generates a DMA request. The CPU must not clear this field by writing 1 to BUF5I.<br><br>0b - No occurrence of completed transmission or reception, or no frames available<br><br>1b - MB5 completed transmission or reception, or frames available |
| 4-1<br><br>BUF4TO1I | Buffer MBi Interrupt or Reserved<br><br>Flags the interrupts for MB4–MB1 when MCR[RFEN] = 0 (Legacy RX FIFO disabled).<br><br>**NOTE**<br>When a CPU write changes the value of MCR[RFEN], FlexCAN clears these flags.<br><br>When MCR[RFEN] = 1, the BUF4TO1I flags are reserved.<br><br>0b - The corresponding buffer has no occurrence of successfully completed transmission or reception.<br><br>1b - The corresponding buffer has successfully completed transmission or reception. |
| 0<br><br>BUF0I | Buffer MB0 Interrupt or Clear Legacy FIFO bit<br><br>Flags the interrupt for MB0 when MCR[RFEN] = 0 (Legacy RX FIFO disabled).<br><br>If MCR[RFEN] = 1, this field is used to trigger the clear Legacy FIFO operation. This operation empties the Legacy FIFO contents. Before performing this operation, the CPU must service all Legacy FIFO-related IFLAG flags.<br><br>When MCR[DMA] = 1, this operation also clears the BUF5I flag, aborting the DMA request. The clear Legacy FIFO operation occurs when the CPU writes 1 to BUF0I. This operation is only allowed in Freeze mode; the module blocks it in other conditions. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - MB0 has no occurrence of successfully completed transmission or reception.<br>1b - MB0 has successfully completed transmission or reception. |

### 70.6.2.14 Control 2 (CTRL2)

#### Offset

| Register | Offset |
|---|---|
| CTRL2 | 34h |

#### Function

Complements Control 1 (CTRL1), providing control bits for memory write-access in Freeze mode. This register extends Legacy FIFO filter quantity, and adjusts the operation of internal FlexCAN processes such as matching and arbitration.

Soft reset does not affect the contents of this register.

Table 445 shows how the value of CTRL2[RFFN] determines the Legacy RX FIFO filter structure.

**Table 445. Possible Legacy RX FIFO filter structures**

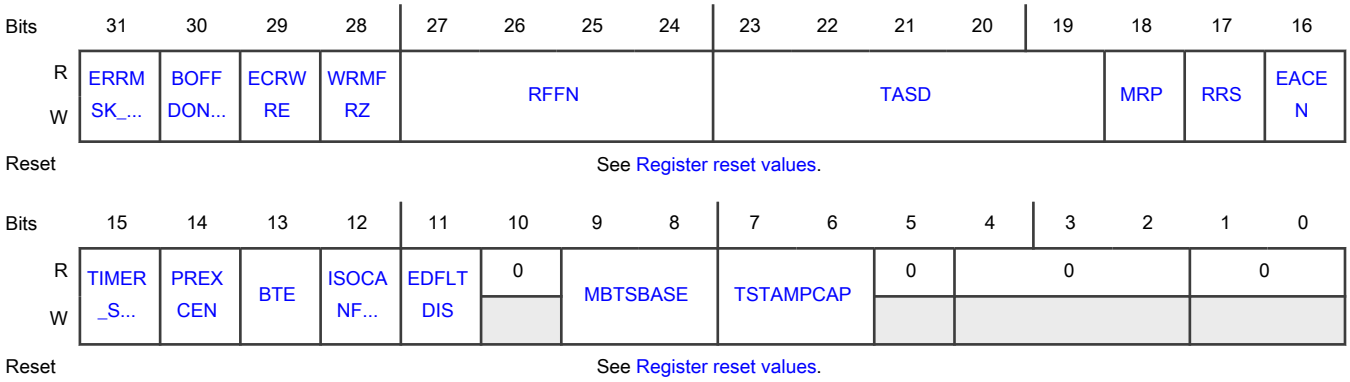
| RFFN[3:0] | Number of Legacy RX FIFO filter elements | Message buffers occupied by Legacy RX FIFO and ID filter table | Remaining available message buffers | Legacy RX FIFO ID filter table elements affected by RX individual masks | Legacy RX FIFO ID filter table elements affected by Legacy RX FIFO global mask |
|---|---|---|---|---|---|
| 0h | 8 | MB 0–7 | MB 8–95 | Elements 0–7 | None |
| 1h | 16 | MB 0–9 | MB 10–95 | Elements 0–9 | Elements 10–15 |
| 2h | 24 | MB 0–11 | MB 12–95 | Elements 0–11 | Elements 12–23 |
| 3h | 32 | MB 0–13 | MB 14–95 | Elements 0–13 | Elements 14–31 |
| 4h | 40 | MB 0–15 | MB 16–95 | Elements 0–15 | Elements 16–39 |
| 5h | 48 | MB 0–17 | MB 18–95 | Elements 0–17 | Elements 18–47 |
| 6h | 56 | MB 0–19 | MB 20–95 | Elements 0–19 | Elements 20–55 |
| 7h | 64 | MB 0–21 | MB 22–95 | Elements 0–21 | Elements 22–63 |
| 8h | 72 | MB 0–23 | MB 24–95 | Elements 0–23 | Elements 24–71 |
| 9h | 80 | MB 0–25 | MB 26–95 | Elements 0–25 | Elements 26–79 |
| Ah | 88 | MB 0–27 | MB 28–95 | Elements 0–27 | Elements 28–87 |
| Bh | 96 | MB 0–29 | MB 30–95 | Elements 0–29 | Elements 30–95 |
| Ch | 104 | MB 0–31 | MB 32–95 | Elements 0–31 | Elements 32–103 |
| Dh | 112 | MB 0–33 | MB 34–95 | Elements 0–31 | Elements 32–111 |

*Table continues on the next page...*

Table 445. Possible Legacy RX FIFO filter structures (continued)

| RFFN[3:0] | Number of Legacy RX FIFO filter elements | Message buffers occupied by Legacy RX FIFO and ID filter table | Remaining available message buffers | Legacy RX FIFO ID filter table elements affected by RX individual masks | Legacy RX FIFO ID filter table elements affected by Legacy RX FIFO global mask |
|---|---|---|---|---|---|
| Eh | 120 | MB 0–35 | MB 36–95 | Elements 0–31 | Elements 32–119 |
| Fh | 128 | MB 0–37 | MB 38–95 | Elements 0–31 | Elements 32–127 |

### Diagram




### Register reset values

| Register | Reset value |
|---|---|
| CTRL2 | FLEXCAN_0: 0060_0000h<br>FLEXCAN_1,FLEXCAN_2: 0080_0000h<br>FLEXCAN_3–FLEXCAN_5: 00A0_0000h |

### Fields

| Field | Function |
|---|---|
| 31<br>ERRMSK_FAST | Error Interrupt Mask for Errors Detected in the Data Phase of Fast CAN FD Frames<br>Enables the ESR1[ERRINT_FAST] interrupt.<br>0b - Disable<br>1b - Enable |
| 30<br>BOFFDONEMSK | Bus Off Done Interrupt Mask<br>Enables the Bus Off Done interrupt, ESR1[BOFFDONEINT].<br>0b - Disable<br>1b - Enable |
| 29 | Error Correction Configuration Register Write Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| ECRWRE | Enables updates for Memory Error Control (MECR). If the protocol described in Detection and correction of memory errors is not followed, this field is automatically 0.<br>0b - Disable<br>1b - Enable |
| 28<br>WRMFRZ | Write Access to Memory in Freeze Mode<br>Enables unrestricted write access to FlexCAN memory in Freeze mode. When this field is 0, write access restrictions are maintained. This field can only be written in Freeze mode, and has no effect out of Freeze mode.<br>**NOTE**<br>Do not write 1 to MCR[RFEN] during FlexCAN memory initialization.<br>0b - Disable<br>1b - Enable |
| 27-24<br>RFFN | Number of Legacy Receive FIFO Filters<br>Defines the number of Receive Legacy FIFO filters, as shown in Table 445. The chip determines the maximum selectable number of filters. Do not program this field with values that cause the number of message buffers occupied by Legacy RX FIFO and Legacy RX FIFO ID Filter to exceed MCR[MAXMB]. MCR[MAXMB] defines the number of message buffers present.<br>This field can only be written in Freeze mode; the module blocks it in other modes.<br>Each group of eight filters occupies a memory space equivalent to two message buffers. The more filters are implemented, the fewer message buffers are available.<br>The Legacy RX FIFO occupies the memory space originally reserved for MB5–MB0. This field should be programmed with a value corresponding to a number of filters less than the number of available memory words. The number of available words can be calculated as follows:<br>(SETUP_MB - 6) × 4<br>Where SETUP_MB is the smaller of the parameter NUMBER_OF_MB and MCR[MAXMB].<br>The number of remaining message buffers available is:<br>(SETUP_MB - 8) - (RFFN × 2)<br>If the number of Legacy RX FIFO filters programmed through RFFN exceeds the SETUP_MB value (memory space available), the exceeding ones are not functional.<br>**NOTE**<br>• The number of the last remaining available message buffers is the smaller of (NUMBER_OF_MB - 1) and MCR[MAXMB].<br>• If RX Individual Mask registers are not enabled, the Legacy RX FIFO Global Mask affects all Legacy RX FIFO filters. |
| 23-19<br>TASD | Transmission Arbitration Start Delay |

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>Indicates by how many CAN bits the transmission arbitration process start point can be delayed from the first bit of CRC field on CAN bus. See TX arbitration start delay for details. This field can be written only in Freeze mode; the module blocks it in other modes.</td></tr>
<tr><td>18<br>MRP</td><td>Message Buffers Reception Priority<br>Sets the priority for the matching process.<br>This field can be written only in Freeze mode; the module blocks it in other modes.<br>NOTE<br>The descriptions of the field settings vary by module instance.<br><table><tr><th>Instance</th><th>Field value and description</th></tr><tr><td>FLEXCAN_0</td><td>0b - Matching starts from Legacy RX FIFO or Enhanced RX FIFO and continues on message buffers.<br>1b - Matching starts from message buffers and continues on Legacy RX FIFO or Enhanced RX FIFO.</td></tr><tr><td>FLEXCAN_1<br>FLEXCAN_2<br>FLEXCAN_3<br>FLEXCAN_4<br>FLEXCAN_5</td><td>0b - Matching starts from RX FIFO and continues on message buffers.<br>1b - Matching starts from message buffers and continues on RX FIFO.</td></tr></table></td></tr>
<tr><td>17<br>RRS</td><td>Remote Request Storing<br>Determines what the module does with a remote request. The remote request frame is submitted to a matching process.<br>If this field is 1, the frame is stored in the corresponding message buffer in the same fashion as a data frame. No automatic remote response frame is generated.<br>If this field is 0, an automatic remote response frame is generated if a message buffer with CODE = 1010b is found with the same ID.<br>You can only write to this field in Freeze mode. The module blocks it in other modes.<br>0b - Generated<br>1b - Stored</td></tr>
<tr><td>16<br>EACEN</td><td>Entire Frame Arbitration Field Comparison Enable for RX Message Buffers<br>Controls the comparison of IDE and RTR fields within RX message buffer filters with their corresponding bits in the incoming frame by the matching process. If enabled, the IDE and RTR fields of the RX message buffer are compared to their corresponding bits within the incoming frame (mask bits apply). If disabled, the IDE field of the RX message buffer filter is always compared and RTR is never compared despite mask bits.<br>This field does not affect matching for Legacy RX FIFO or Enhanced RX FIFO.<br>You can only write to this field in Freeze mode; the module blocks it in other modes.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disable<br>1b - Enable |
| 15<br>TIMER_SRC | Timer Source<br>Selects the time tick source used for incrementing the free-running timer counter.<br>If CAN bit clock is selected, it defines the baud rate on the CAN bus.<br>If External time tick is selected, the period can be adjusted to match the baud rate on the CAN bus. It can also be adjusted to a different value as required. See the chip-specific section for details about the external time tick.<br>You can only write to this field in Freeze mode.<br>0b - CAN bit clock<br>1b - External time tick |
| 14<br>PREXCEN | Protocol Exception Enable<br>Enables the protocol exception feature.<br>You can only write to this field in Freeze mode.<br>**NOTE**<br>See Protocol exception event in the CAN Protocol standard (ISO 11898-1:2015) for details.<br>0b - Disabled<br>1b - Enabled |
| 13<br>BTE | Bit Timing Expansion Enable<br>Enables the use of Enhanced CAN Bit Timing Prescalers (EPRS), Enhanced Data Phase CAN Bit Timing (EDCBT), and Enhanced Nominal CAN Bit Timing (ENCBT) to configure the CAN bit timing segments, instead of using the bit timing fields of CAN Bit Timing (CBT), CAN FD Bit Timing (FDCBT), and Control 1 (CTRL1).<br>If this field is 1:<br>• CTRL1[PRESDIV], CTRL1[PROPSEG], CTRL1[PSEG1], CTRL1[PSEG2], and CTRL1[RJW] are read as zero. A write operation to these fields has no effect.<br>• CBT[EPRESDIV], CBT[ERJW], CBT[EPROPSEG], CBT[EPSEG1], and CBT[EPSEG2], and the corresponding fields in CAN FD Bit Timing (FDCBT), are read as zero. A write operation to these fields has no effect.<br>• ETDC[ETDCOFF], ETDC[ETDCEN], ETDC[ETDCFAIL], and ETDC[ETDCVAL] are used by FlexCAN instead of FDCTRL[TDCOFF], FDCTRL[TDCEN], FDCTRL[TDCFAIL], and FDCTRL[TDCVAL]. These fields are read as zero, and a write operation to them has no effect.<br>• ETDC[TDMDIS] can be used to disable transceiver delay measurement.<br>0b - Disable<br>1b - Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 12<br>ISOCANFDEN | ISO CAN FD Enable<br>Enables the CAN FD protocol according to ISO specification (ISO 11898-1:2015) (see CAN FD ISO compliance). When disabled, FlexCAN operates using the non-ISO CAN FD protocol.<br>You can only write to this field in Freeze mode.<br>**NOTE**<br>FlexCAN is able to transmit FD frame format according to CAN Protocol standard (ISO 11898-1:2015).<br>0b - Disable<br>1b - Enable |
| 11<br>EDFLTDIS | Edge Filter Disable<br>Disables the edge filter used during the Bus Integration state.<br>When the Edge Filter is enabled, two consecutive nominal time quanta with dominant bus states are required to detect an edge that causes synchronization. When synchronization occurs, the counting of the sequence of 11 consecutive recessive bits is restarted. The edge filter prevents dominant pulses that are shorter than a nominal bit time (present during the data phase of an FD frame) from being mistaken for an idle condition.<br>You can only write to this field in Freeze mode.<br>**NOTE**<br>See Bus Integration state in the CAN Protocol standard (ISO 11898-1:2015) for details.<br>0b - Enabled<br>1b - Disabled |
| 10<br>— | Reserved |
| 9-8<br>MBTSBASE | Message Buffer Timestamp Base<br>Selects the timebase used for capturing the 16-bit TIME_STAMP field of the message buffer register.<br>This field can be written in Freeze Mode only.<br>00b - TIMER<br>01b - Lower 16 bits of high-resolution timer<br>10b - Upper 16 bits of high-resolution timer<br>11b - Reserved |
| 7-6<br>TSTAMPCAP | Timestamp Capture Point<br>Configures the point in time when a 32-bit timebase is captured during a CAN frame. This base is stored in the high-resolution timestamp register (HR_TIME_STAMP). |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | For classical CAN frames, capture points can be the start-of-frame bit or the point when CAN frame is considered valid. This point is the seventh bit of the end-of-frame for transmission and the sixth bit of the end-of-frame for reception. For CAN FD frames, the high-resolution timestamp can be captured at the start of frame, when a CAN FD frame is considered valid, or the res bit.<br>You can only write to this field in Freeze mode.<br>00b - Disabled<br>01b - End of the CAN frame<br>10b - Start of the CAN frame<br>11b - Start of frame for classical CAN frames; res bit for CAN FD frames |
| 5<br>— | Reserved |
| 4-2<br>— | Reserved |
| 1-0<br>— | Reserved |

### 70.6.2.15 Error and Status 2 (ESR2)

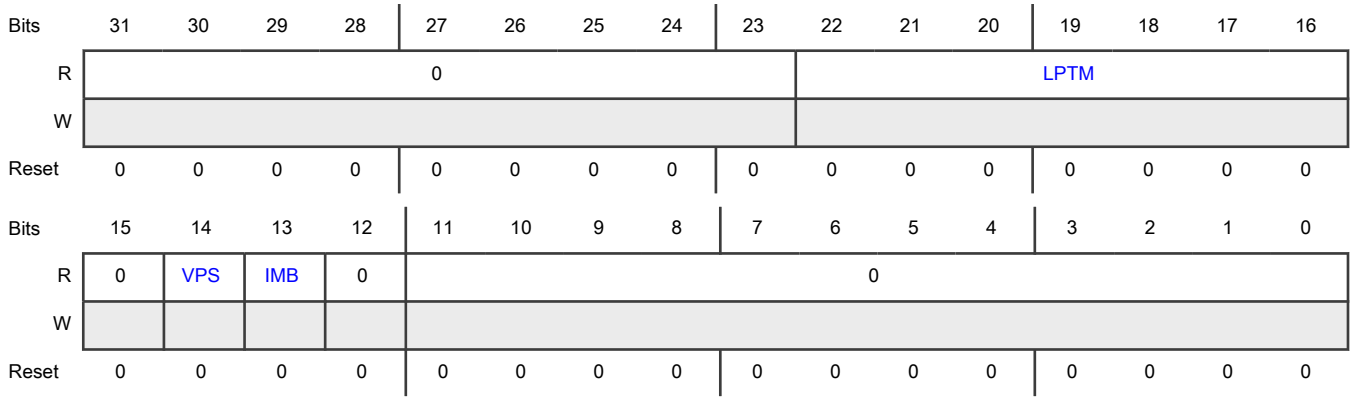
#### Offset

| Register | Offset |
|---|---|
| ESR2 | 38h |

#### Function

Reports general status information.

#### Diagram



### Fields

| Field | Function |
|---|---|
| 31-23<br>— | Reserved |
| 22-16<br>LPTM | Lowest Priority TX Message Buffer<br>Indicates the lowest number inactive message buffer when ESR2[VPS] = 1 (see ESR2[IMB]). If no message buffer is inactive, the message buffer indicated depends on the value of CTRL1[LBUF]. If CTRL1[LBUF] = 0, the message buffer indicated is the one with the greatest arbitration value (see Highest-priority message buffer first). If CTRL1[LBUF] = 1, the message buffer indicated is the active TX message buffer with the highest number.<br>If a TX message buffer is being transmitted, it is not considered in the LPTM calculation. If ESR2[IMB] is not 0 and a frame is transmitted successfully, the value of LPTM is updated with its message buffer number. |
| 15<br>— | Reserved |
| 14<br>VPS | Valid Priority Status<br>Indicates whether the contents of ESR2[IMB] and ESR2[LPTM] are valid. It becomes 1 upon every complete TX arbitration process, unless the CPU writes to the Control and Status word of a message buffer already scanned. In other words, it is behind the TX Arbitration Pointer, during the TX arbitration process. If there is no inactive message buffer and only one TX message buffer that is being transmitted, this field remains 0. This field becomes 0 upon the start of every TX arbitration process or upon a write to the Control and Status word of any message buffer.<br>**NOTE**<br>No CPU write to the Control and Status of a message buffer that the abort mechanism blocks affects this field. When MCR[AEN] = 1, the abort code write to the Control and Status of an MB being transmitted (pending abort) is blocked. Any write attempt to a TX MB with its IFLAG flag set is also blocked.<br>0b - Invalid<br>1b - Valid |
| 13<br>IMB | Inactive Message Buffer<br>Indicates whether any message buffer is inactive (CODE field is either 1000b or 0b) when ESR2[VPS] = 1. This field becomes 1 when:<br>• A lowest-priority TX message buffer (ESR2[LPTM]) is found and it is inactive during arbitration.<br>• This field is not 1, and a frame is transmitted successfully.<br>This field always becomes 0 at the start of arbitration (see Arbitration process).<br>If a message buffer is successfully transmitted and this field is 0 (no inactive message buffer), ESR2[VPS] and this field both become 1. The index related to the MB transmitted is loaded into ESR2[LPTM]. In this case, the value of ESR2[LPTM] is the number of the first inactive message buffer.<br>0b - Message buffer indicated by ESR2[LPTM] is not inactive.<br>1b - At least one message buffer is inactive. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 12<br>— | Reserved |
| 11-0<br>— | Reserved |

### 70.6.2.16 Cyclic Redundancy Check (CRCR)

#### Offset

| Register | Offset |
|---|---|
| CRCR | 44h |

#### Function

Provides information about the CRC of transmitted messages for non-FD messages. This register only reports the 15 low-order bits of CRC calculations for messages in CAN FD format that require either 17 or 21 bits. For CAN FD format frames, you must use the CAN FD CRC (FDCRC). This register is updated at the same time that the TX interrupt flag is set.

**NOTE**

See CRC sequence calculation in the CAN Protocol standard (ISO 11898-1:2015) for details.

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-23<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 22-16<br>MBCRC | CRC Message Buffer<br>Indicates the number of the message buffer corresponding to the value in CRCR[TXCRC]. |
| 15<br>— | Reserved |
| 14-0<br>TXCRC | Transmitted CRC value<br>Indicates the CRC value of the last transmitted message for non-FD frames. For FD frames, CRC value is reported in CAN FD CRC (FDCRC). |

### 70.6.2.17 Legacy RX FIFO Global Mask (RXFGMASK)

#### Offset

| Register | Offset |
|---|---|
| RXFGMASK | 48h |

#### Function

Masks the Legacy RX FIFO ID filter table elements that do not have a corresponding RXIMR according to CTRL2[RFFN], when Legacy RX FIFO is enabled.

This register is located in RAM.

You can only write to this register in Freeze mode; the module blocks it in other modes.

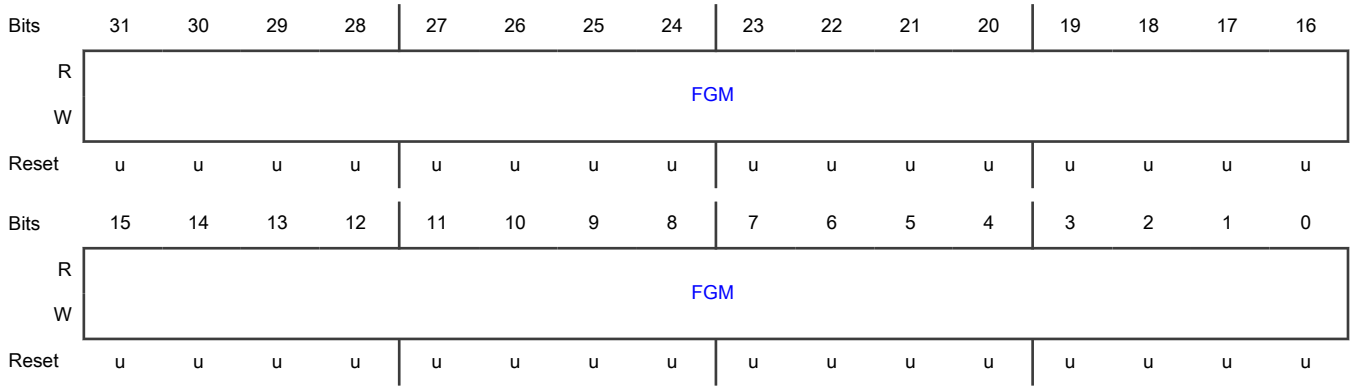
The following table shows how the FGM bits correspond to each IDAF field.

**Table 446. Correspondence of Legacy RX FIFO global mask bits to IDF fields**

| Legacy RX FIFO ID filter table elements format (MCR[IDAM]) | Identifier acceptance filter fields | | | | | |
|---|---|---|---|---|---|---|
| | RTR | IDE | RXIDA | RXIDB [1] | RXIDC [2] | Reserved |
| A | FGM[31] | FGM[30] | FGM[29:1] | — | — | FGM[0] |
| B | FGM[31],<br>FGM[15] | FGM[30],<br>FGM[14] | — | FGM[29:16],<br>FGM[13:0] | | — |
| C | — | — | | — | FGM[31:24],<br>FGM[23:16],<br>FGM[15:8],<br>FGM[7:0] | |

1. If MCR[IDAM] is equivalent to format B, only the 14 most significant bits of the identifier of the incoming frame are compared with the Legacy RX FIFO filter.
2. If MCR[IDAM] is equivalent to format C, only the eight most significant bits of the identifier of the incoming frame are compared with the Legacy RX FIFO filter.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>FGM | Legacy RX FIFO Global Mask Bits<br>Masks the ID filter table elements bits in a perfect alignment.<br>0b - The corresponding bit in the filter is "don't care."<br>1b - The corresponding bit in the filter is checked. |

## 70.6.2.18 Legacy RX FIFO Information (RXFIR)

### Offset

| Register | Offset |
|---|---|
| RXFIR | 4Ch |

### Function

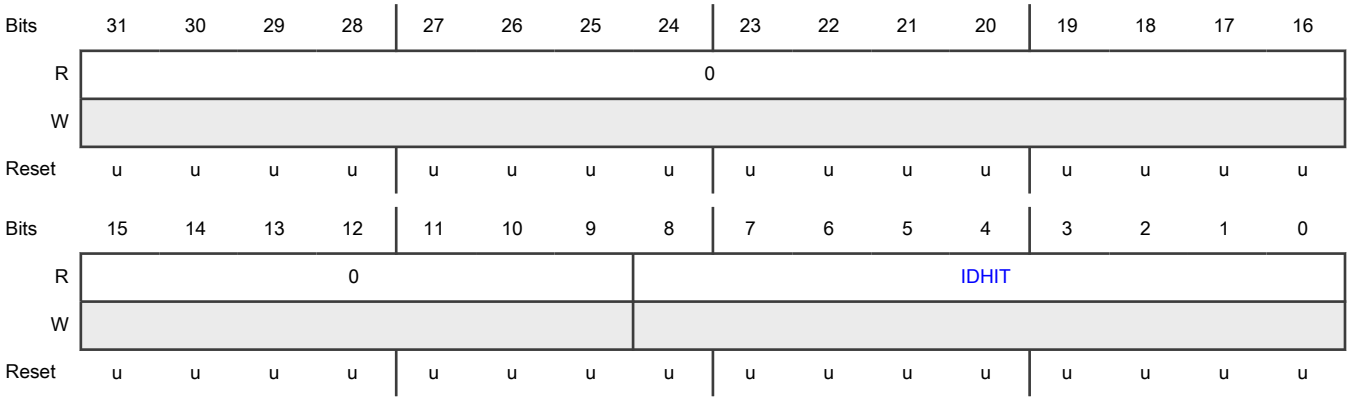
Provides information about Legacy RX FIFO.

This register is the port through which the CPU accesses the output of the Legacy RXFIR FIFO located in RAM. FlexCAN writes to the Legacy RXFIR FIFO when a new message is moved into the Legacy RX FIFO. Also, its output is updated whenever the output of the Legacy RX FIFO is updated with the next message. See Legacy RX FIFO for instructions on reading this register.

**NOTE**

RXFIR can be written only during memory initialization, due to the error code correction (ECC) feature. In every other case, this register is read-only.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8-0<br>IDHIT | Identifier Acceptance Filter Hit Indicator<br>Indicates which Identifier Acceptance filter that the received message hit in the output of the Legacy RX FIFO. If multiple filters match the incoming message ID, the first matching IDAF found (lowest number) by the matching process is indicated. This field is valid only when IFLAG1[BUF5I] is set. |

## 70.6.2.19 CAN Bit Timing (CBT)

### Offset

| Register | Offset |
|---|---|
| CBT | 50h |

### Function

Provides an alternative way to store the CAN bit timing variables described in Control 1 (CTRL1). EPRESDIV, EPROPSEG, EPSEG1, EPSEG2, and ERJW are extended versions of CTRL1[PRESDIV], CTRL1[PROPSEG], CTRL1[PSEG1], CTRL1[PSEG2], and CTRL1[RJW] respectively.

**NOTE**

The CAN bit variables in CTRL1 and in CBT are stored in the same register.

CBT[BTF] selects the use of the timing variables defined in this register.

When the CAN FD feature is enabled (MCR[FDEN] = 1), always write 1 to CBT[BTF].

Soft reset does not affect the contents of this register.

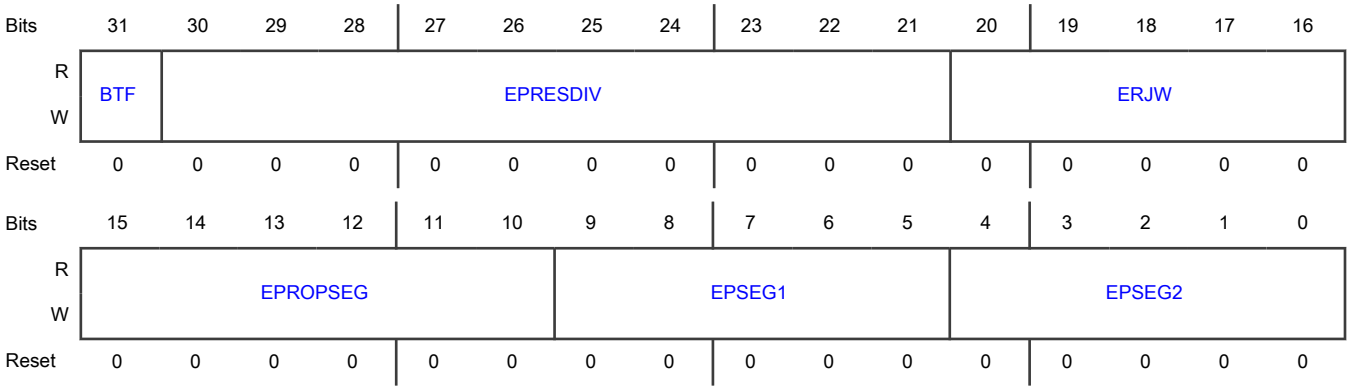
**NOTE**

Ensure that bit time settings and protocol engine tolerance are in compliance with the CAN Protocol standard (ISO 11898-1:2015).

**NOTE**

If CTRL2[BTE] = 1, EPRESDIV, ERJW, EPROPSEG, EPSEG1, and EPSEG2 are read as zero. A write operation to them has no effect.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>BTF | Bit Timing Format Enable<br>Enables the use of extended CAN bit timing fields EPRESDIV, EPROPSEG, EPSEG1, EPSEG2, and ERJW. These fields replace the CAN bit timing variables defined in Control 1 (CTRL1). This field can be written in Freeze mode only.<br>0b - Disable<br>1b - Enable |
| 30-21<br>EPRESDIV | Extended Prescaler Division Factor<br>Defines the ratio between the PE clock frequency and the serial clock (Sclock) frequency when CBT[BTF] = 1, otherwise it has no effect. It extends the CTRL1[PRESDIV] value range.<br>The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency (see Protocol timing). This field can be written only in Freeze mode; the module blocks it in other modes.<br>Sclock frequency = PE clock frequency ÷ (EPRESDIV + 1) |
| 20-16<br>ERJW | Extended Resync Jump Width<br>Defines the maximum number of time quanta that one resynchronization can change a bit time when CBT[BTF] = 1. Otherwise, it has no effect. It extends the CTRL1[RJW] value range.<br>This field can be written only in Freeze mode; the module blocks it in other modes.<br>Resync Jump Width = ERJW + 1.<br>One Time Quantum = one Sclock period. |
| 15-10<br>EPROPSEG | Extended Propagation Segment |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Defines the length of the propagation segment in the bit time when CBT[BTF] = 1, otherwise it has no effect. It extends the CTRL1[PROPSEG] value range. This field can be written only in Freeze mode; the module blocks it in other modes.<br>Propagation Segment Time = (EPROPSEG + 1) × Time Quanta.<br>One Time Quantum = one Sclock period. |
| 9-5<br>EPSEG1 | Extended Phase Segment 1<br>Defines the length of phase segment 1 in the bit time when CBT[BTF] = 1, otherwise it has no effect. It extends the CTRL1[PSEG1] value range. This field can be written only in Freeze mode; the module blocks it in other modes.<br>Phase Buffer Segment 1 = (EPSEG1 + 1) × Time Quanta.<br>One Time Quantum = one Sclock period. |
| 4-0<br>EPSEG2 | Extended Phase Segment 2<br>Defines the length of phase segment 2 in the bit time when CBT[BTF] = 1, otherwise it has no effect. It extends the CTRL1[PSEG2] value range. This field can be written only in Freeze mode; the module blocks it in other modes.<br>Phase Buffer Segment 1 = (EPSEG2 + 1) × Time Quanta.<br>One Time Quantum = one Sclock period. |

### 70.6.2.20 Interrupt Masks 3 (IMASK3)

#### Offset

| Register | Offset |
|---|---|
| IMASK3 | 6Ch |

#### Function

Enables or disables any number of the 32 message buffer interrupts for MB95–MB64. It contains one interrupt mask bit per buffer. This configuration allows the CPU to determine which buffer generates an interrupt after a successful transmission or reception when the corresponding IFLAG3 flag is set.

**NOTE**

Each module instance supports a different number of registers.

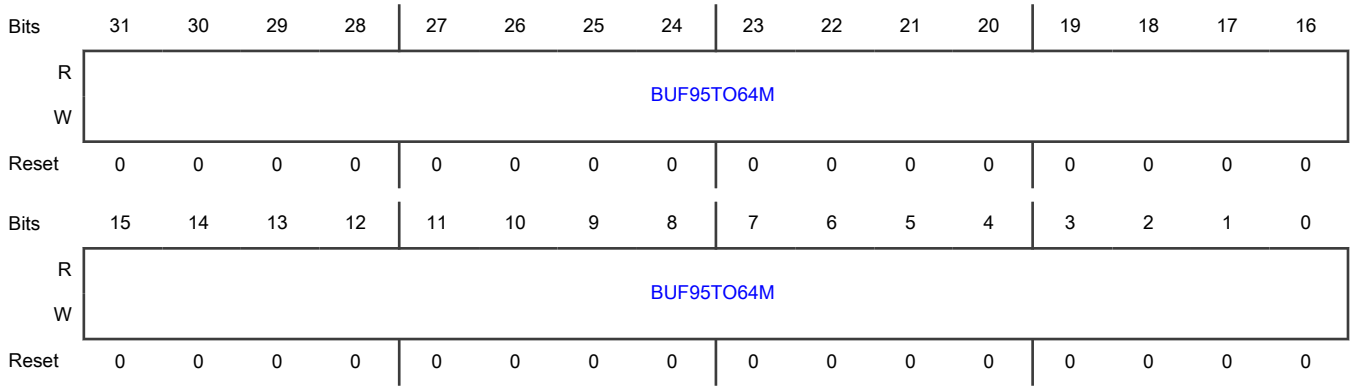
| Instance | Register supported | Register not supported |
|---|---|---|
| FLEXCAN_0 | IMASK3 | — |
| FLEXCAN_1 | — | IMASK3 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Instance | Register supported | Register not supported |
|---|---|---|
| FLEXCAN_2 | — | IMASK3 |
| FLEXCAN_3 | — | IMASK3 |
| FLEXCAN_4 | — | IMASK3 |
| FLEXCAN_5 | — | IMASK3 |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>BUF95TO64M | Buffer MBi Mask<br>Enables or disables the corresponding FlexCAN message buffer interrupt for MB95–MB64. When CAN FD is enabled, the MB range is defined in accordance with FDCTRL[MBDSRn].<br>**NOTE**<br>If the corresponding IFLAG3 flag is set, writing 1 or 0 to a field in IMASK3 can set or clear an interrupt request.<br>0b - The corresponding buffer interrupt is disabled.<br>1b - The corresponding buffer interrupt is enabled. |

## 70.6.2.21 Interrupt Flags 3 (IFLAG3)

### Offset

| Register | Offset |
|---|---|
| IFLAG3 | 74h |

### Function

Defines the flags for the 32 message buffer interrupts for MB95–MB64. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG3 flag. If the corresponding IMASK3 bit is 1, an interrupt is generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.

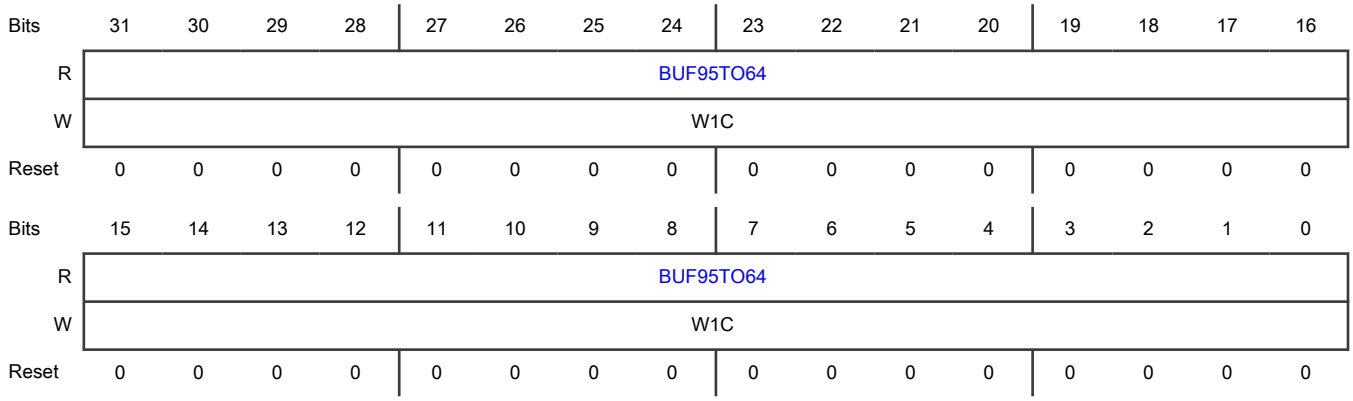
Before updating MCR[MAXMB], the CPU must service the IFLAG3 flags whose MB value is greater than the MAXMB to be updated. Otherwise, they remain set and are inconsistent with the number of message buffers available.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| FLEXCAN_0 | IFLAG3 | — |
| FLEXCAN_1 | — | IFLAG3 |
| FLEXCAN_2 | — | IFLAG3 |
| FLEXCAN_3 | — | IFLAG3 |
| FLEXCAN_4 | — | IFLAG3 |
| FLEXCAN_5 | — | IFLAG3 |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>BUF95TO64 | Buffer MBi Interrupt<br>Flags the corresponding FlexCAN message buffer interrupt for MB95–MB64.When CAN FD is enabled, the MB range is defined in accordance with FDCTRL[MBDSRn].<br>0b - The corresponding buffer has no occurrence of successfully completed transmission or reception.<br>1b - The corresponding buffer has successfully completed transmission or reception. |

### 70.6.2.22 Receive Individual Mask (RXIMR0 - RXIMR95)

#### Offset

For n = 0 to 95:

| Register | Offset |
|---|---|
| RXIMRn | 880h + (n × 4h) |

#### Function

Stores the acceptance masks for ID filtering in RX message buffers and the Legacy RX FIFO.

When the Legacy RX FIFO is disabled (MCR[RFEN] = 0), an individual mask is provided for each available RX message buffer on a one-to-one correspondence. When the Legacy RX FIFO is enabled (MCR[RFEN] = 1), an individual mask is provided for each Legacy RX FIFO ID filter table element on a one-to-one correspondence. This correspondence depends on the setting of CTRL2[RFFN] (see Legacy RX FIFO).

RXIMR0 stores the individual mask associated with either MB0 or ID filter table element 0. RXIMR1 stores the individual mask associated with either MB1 or ID filter table element 1, and so on.

The CPU can only access the RXIMR registers when the module is in Freeze mode; otherwise, the module blocks them. Reset does not affect these registers. They are located in RAM and must be explicitly initialized prior to any reception.

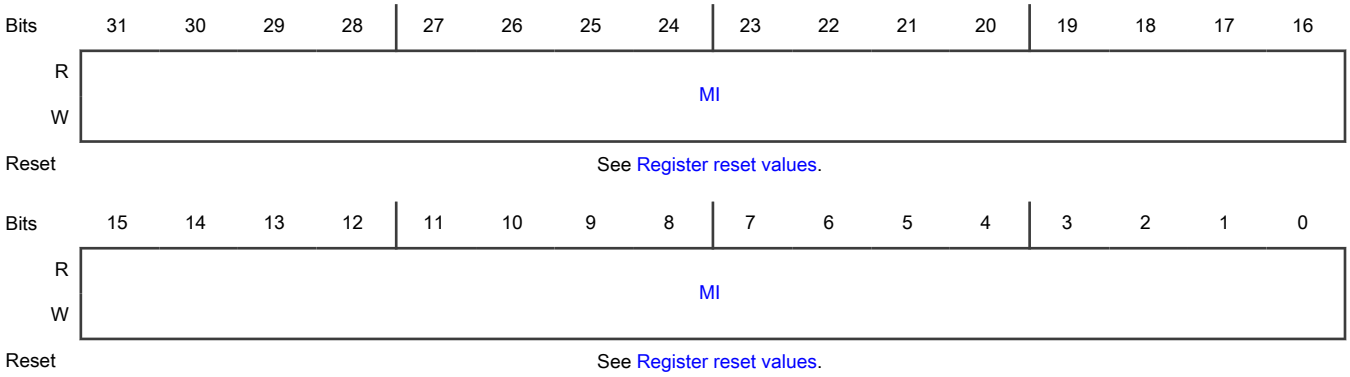
It is possible for the RXIMR memory region to be accessed as general-purpose memory. See Bus interface for more information.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| FLEXCAN_0 | RXIMR0–RXIMR95 | — |
| FLEXCAN_1 | RXIMR0–RXIMR63 | RXIMR64–RXIMR95 |
| FLEXCAN_2 | RXIMR0–RXIMR63 | RXIMR64–RXIMR95 |
| FLEXCAN_3 | RXIMR0–RXIMR31 | RXIMR32–RXIMR95 |
| FLEXCAN_4 | RXIMR0–RXIMR31 | RXIMR32–RXIMR95 |
| FLEXCAN_5 | RXIMR0–RXIMR31 | RXIMR32–RXIMR95 |

### Diagram




### Register reset values

| Register | Reset value |
|---|---|
| RXIMR0–RXIMR31 | FLEXCAN_0–FLEXCAN_5: undefined |
| RXIMR32–RXIMR63 | FLEXCAN_0–FLEXCAN_2: undefined<br>FLEXCAN_3–FLEXCAN_5: Register not supported |
| RXIMR64–RXIMR95 | FLEXCAN_0: undefined<br>FLEXCAN_1–FLEXCAN_5: Register not supported |

### Fields

| Field | Function |
|---|---|
| 31-0<br>MI | Individual Mask Bits<br>Masks the corresponding bit in both the message buffer filter and Legacy RX FIFO ID filter table element in distinct ways.<br>For message buffer filters, see RX Message Buffers Global Mask (RXMGMASK).<br>For Legacy RX FIFO ID filter table elements, see Legacy RX FIFO Global Mask (RXFGMASK).<br>0b - The corresponding bit in the filter is "don't care."<br>1b - The corresponding bit in the filter is checked. |

## 70.6.2.23 Memory Error Control (MECR)

### Offset

| Register | Offset |
|---|---|
| MECR | AE0h |

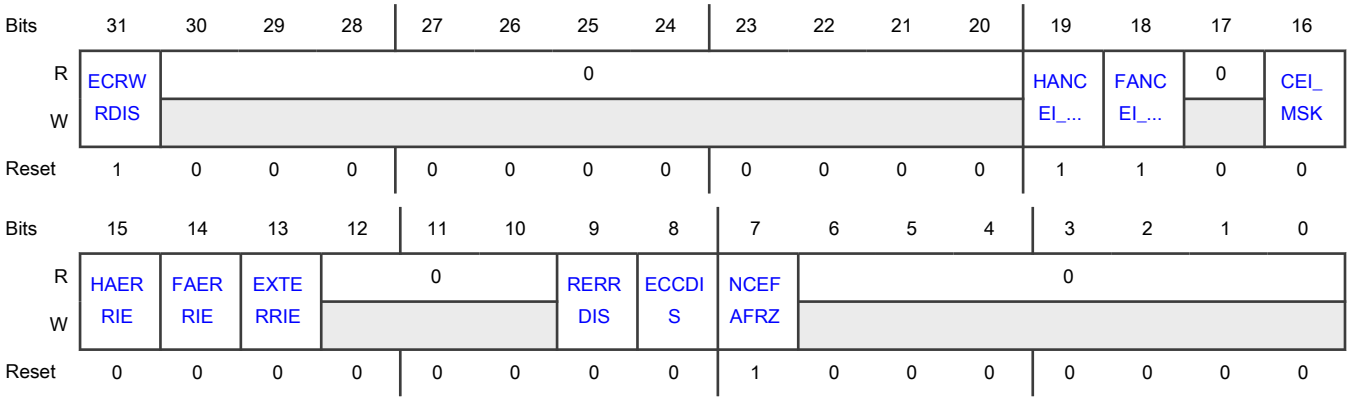
### Function

Contains control bits for memory error detection and correction (ECC).

**NOTE**

When CTRL2[ECRWRE] = 0, writes to this register are blocked, except for MECR[ECRWRDIS].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>ECRWRDIS | Error Configuration Register Write Disable<br>Disables writes on this register.<br>This field automatically becomes 1 (disabled) when CTRL2[ECRWRE] is 1. The protocol described in Detection and correction of memory errors must be followed.<br>0b - Enable<br>1b - Disable |
| 30-20<br>— | Reserved |
| 19<br>HANCEI_MSK | Host Access with Noncorrectable Errors Interrupt Mask<br>Enables the interrupt for noncorrectable errors detected in memory reads issued by the host (CPU).<br>0b - Disable<br>1b - Enable |
| 18<br>FANCEI_MSK | FlexCAN Access with Noncorrectable Errors Interrupt Mask<br>Enables the interrupt for noncorrectable errors detected in memory reads issued by FlexCAN internal processes.<br>0b - Disable<br>1b - Enable |
| 17<br>— | Reserved |
| 16 | Correctable Errors Interrupt Mask |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| CEI_MSK | Enables the interrupt for correctable errors detected in memory reads issued by the host or FlexCAN internal processes.<br>0b - Disable<br>1b - Enable |
| 15<br>HAERRIE | Host Access Error Injection Enable<br>Enables the injection of errors only in memory reads issued by the host (CPU).<br>0b - Disable<br>1b - Enable |
| 14<br>FAERRIE | FlexCAN Access Error Injection Enable<br>Enables the injection of errors only in memory reads issued by FlexCAN internal processes.<br>0b - Disable<br>1b - Enable |
| 13<br>EXTERRIE | Extended Error Injection Enable<br>Extends the error injection on 32-bit memory accesses to the complementary 32-bit word. This feature uses the same 32-bit error injection data and parity words used for 64-bit memory accesses performed by internal FlexCAN processes. See Error Injection Data Pattern (ERRIDPR) and Error Injection Parity Pattern (ERRIPPR).<br>0b - Disable. Apply error injection only to the 32-bit word.<br>1b - Enable. Apply error injection to the 64-bit word. |
| 12-10<br>— | Reserved |
| 9<br>RERRDIS | Error Report Disable<br>Disables the update of the error report registers. The update of error-related flags and the generation of bus transfer errors are still active.<br>**NOTE**<br>When reading the report registers, this field must be 1 to ensure coherence on the consecutive register reads.<br>0b - Enable<br>1b - Disable |
| 8<br>ECCDIS | Error Correction Disable<br>Completely disables the memory detection and correction mechanism. Besides disabling the error report mechanism, it also stops the update of the error-related flags and the generation of bus transfer errors. The parity bits continue to be calculated and written into memory on write transactions. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Enable<br>1b - Disable |
| 7<br>NCEFAFRZ | Noncorrectable Errors in FlexCAN Access Put Chip in Freeze Mode<br>Determines whether to put FlexCAN into Freeze mode when a noncorrectable error is detected in a memory read performed by FlexCAN internal processes. In this case, entering Freeze mode prevents FlexCAN internal processes from treating corrupted data as valid. See Freeze mode for more information.<br>0b - Normal operation<br>1b - Freeze mode |
| 6-0<br>— | Reserved |

### 70.6.2.24 Error Injection Address (ERRIAR)

#### Offset

| Register | Offset |
|---|---|
| ERRIAR | AE4h |

#### Function

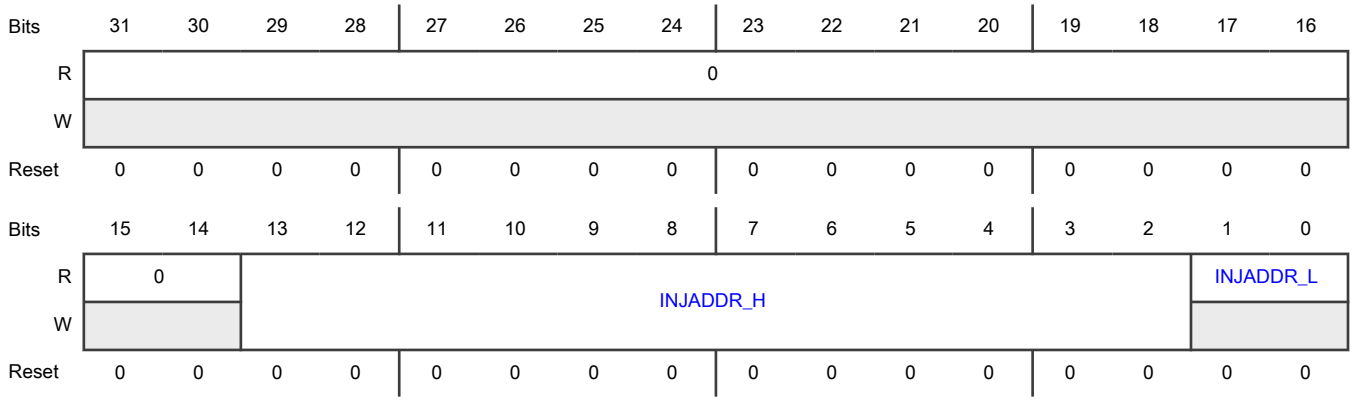
Contains the address where error is to be injected.

See the chip-specific FlexCAN information for the FlexCAN RAM to memory map address mapping table.

**NOTE**

For RXFIR, Enhanced RX FIFO, and HR_TIME_STAMP addresses, you must inject ECC errors in host access only.

#### Diagram



### Fields

| Field | Function |
|---|---|
| 31-14<br>— | Reserved |
| 13-2<br>INJADDR_H | Error Injection Address High<br>Defines the 12 most significant bits of the physical RAM address where error is to be injected (see table above). |
| 1-0<br>INJADDR_L | Error Injection Address Low<br>Defines the two least significant bits of the physical RAM address where error is to be injected. These bits ensure that the address is on a 32-bit boundary. |

## 70.6.2.25 Error Injection Data Pattern (ERRIDPR)

### Offset

| Register | Offset |
|---|---|
| ERRIDPR | AE8h |

### Function
Contains the error pattern to be injected in the data word read from memory.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DFLIP | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | DFLIP | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>DFLIP | Data Flip Pattern<br>Contains the flip pattern. Bits set to 1 in the flip pattern cause the corresponding data bit in the word read from memory to invert. |

### 70.6.2.26 Error Injection Parity Pattern (ERRIPPR)

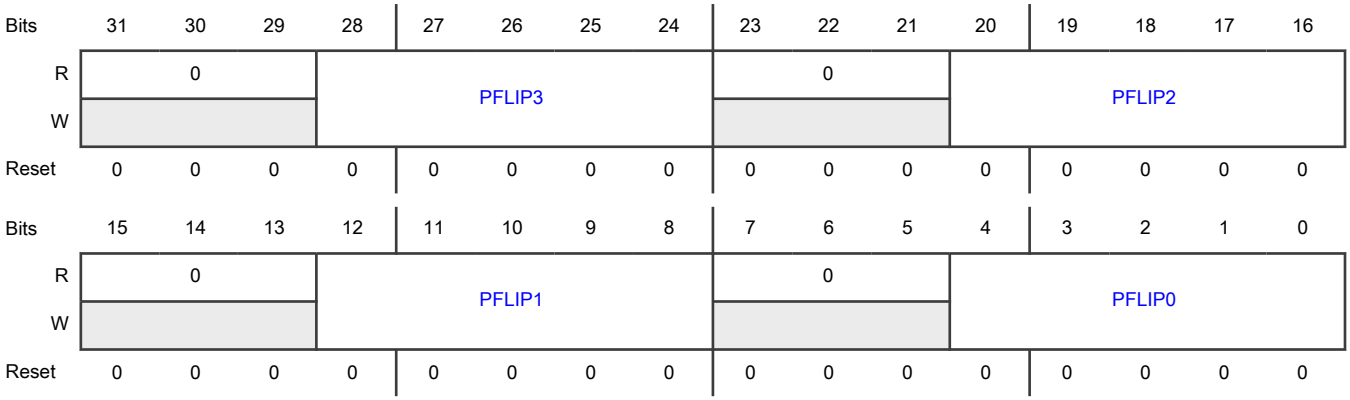
#### Offset

| Register | Offset |
|---|---|
| ERRIPPR | AECh |

#### Function

Contains the error pattern to be injected in parity bits read from memory with the data word. Bits set to 1 in the flip pattern cause the corresponding parity bit in the word read from memory to invert.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |
| 28-24<br>PFLIP3 | Parity Flip Pattern for Byte 3 (Most Significant)<br>Contains the error injection pattern for Byte 3. |
| 23-21<br>— | Reserved |
| 20-16<br>PFLIP2 | Parity Flip Pattern for Byte 2<br>Contains the error injection pattern for Byte 2. |
| 15-13<br>— | Reserved |
| 12-8<br>PFLIP1 | Parity Flip Pattern for Byte 1<br>Contains the error injection pattern for Byte 1. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 7-5<br>— | Reserved |
| 4-0<br>PFLIP0 | Parity Flip Pattern for Byte 0 (Least Significant)<br>Contains the error injection pattern for Byte 0. |

### 70.6.2.27 Error Report Address (RERRAR)

#### Offset

| Register | Offset |
|---|---|
| RERRAR | AF0h |

#### Function

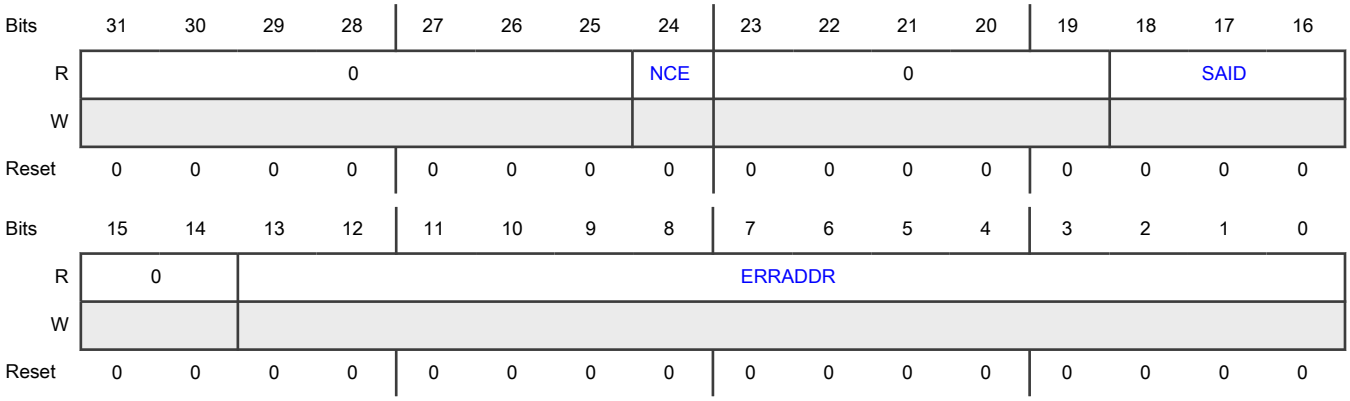
Reports the address used for an access operation in which an error (correctable or noncorrectable) was detected. Also reports the identification of the source of that access.

This address is always reported using a 32-bit alignment. Non-aligned accesses (ERRADDR[1:0] nonzero) are reported with the address aligned, and data is reported in Error Report Data (RERRDR) shifted accordingly. When errors are detected in accesses larger than 32-bit (as performed by FlexCAN internal processes), the address of the 32-bit word where the error was detected is reported. For errors detected in more than one 32-bit word, only the least significant address is reported.

**Table 447. Source of memory access**

| SAID[2:0] | Error during... |
|---|---|
| 0 | Move-out FlexCAN access |
| 1 | Move-in |
| 2 | TX arbitration |
| 3 | RX matching |
| 4 | Move-out host access |
| 5–7 | Reserved |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved |
| 24<br>NCE | Noncorrectable Error<br>Indicates that the report is due to a noncorrectable error.<br>0b - Reporting a correctable error<br>1b - Reporting a noncorrectable error |
| 23-19<br>— | Reserved |
| 18-16<br>SAID | SAID<br>SAID[2] — Identification of the requester of the memory read request:<br>• 0 = Requested by FlexCAN internal processes<br>• 1 = Requested by host (CPU)<br>SAID[1] — Details of FlexCAN operation:<br>• 0 = Move<br>• 1 = Scanning<br>SAID[0] — Operation that requested the memory read:<br>• 0 = Transmission<br>• 1 = Reception<br>For more information, see Table 447. |
| 15-14<br>— | Reserved |
| 13-0 | Address Where Error Detected |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| ERRADDR | See description of Error Injection Address (ERRIAR). |

### 70.6.2.28 Error Report Data (RERRDR)

#### Offset

| Register | Offset |
|---|---|
| RERRDR | AF4h |

#### Function

Reports the raw data (unmodified by the correction performed by ECC logic) read from memory with error. The value reported does not represent the transient values of the BUSY bit (see Table 453) when reading a message buffer.

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-0<br>RDATA | Raw Data Word Read from Memory with Error |

### 70.6.2.29 Error Report Syndrome (RERRSYNR)

#### Offset

| Register | Offset |
|---|---|
| RERRSYNR | AF8h |

### Function

Contains the syndrome detected in a memory read with error. It also reports the bytes which were read in this 32-bit read transaction.

Each SYND*n* field indicates the type of error and which bit in byte (*n*) the error affects. SYND3 corresponds to the most significant byte in the data word read from memory; SYND0 corresponds to the least significant.

**Table 448. Syndrome definition**

| SYND*n* (hex) | Type | Bit affected |
|---|---|---|
| 00 | — | None (no error) |
| 01 | Code | 0 |
| 02 | Code | 1 |
| 04 | Code | 2 |
| 07 | Data | 5 |
| 08 | Code | 3 |
| 0E | Data | 7 |
| 10 | Code | 4 |
| 13 | Data | 2 |
| 15 | Data | 6 |
| 16 | Data | 1 |
| 19 | Data | 3 |
| 1A | Data | 4 |
| 1C | Data | 0 |
| 06 | — | All-zeroes noncorrectable error |
| 1F | — | All-ones noncorrectable error |
| All others | — | Noncorrectable error |

Each BE*n* field indicates which byte in the 32-bit word reported was effectively read. The syndrome bits are calculated for all bytes, including the non-read ones. Errors detected in non-read bytes are indicated (see Error indication) and reported (see Error reporting).

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BE3 | 0 | | | SYND3 | | | | BE2 | 0 | | | SYND2 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | BE1 | 0 | | | SYND1 | | | | BE0 | 0 | | | SYND0 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>BE3 | Byte Enabled for Byte 3 (Most Significant)<br>0b - Byte was not read.<br>1b - Byte was read. |
| 30-29<br>— | Reserved |
| 28-24<br>SYND3 | Error Syndrome for Byte 3 (Most Significant)<br>See Table 448. |
| 23<br>BE2 | Byte Enabled for Byte 2<br>0b - Byte was not read.<br>1b - Byte was read. |
| 22-21<br>— | Reserved |
| 20-16<br>SYND2 | Error Syndrome for Byte 2<br>See Table 448. |
| 15<br>BE1 | Byte Enabled for Byte 1<br>0b - Byte was not read.<br>1b - Byte was read. |
| 14-13<br>— | Reserved |
| 12-8<br>SYND1 | Error Syndrome for Byte 1<br>See Table 448. |
| 7<br>BE0 | Byte Enabled for Byte 0 (Least Significant)<br>0b - Byte was not read.<br>1b - Byte was read. |
| 6-5<br>— | Reserved |
| 4-0<br>SYND0 | Error Syndrome for Byte 0 (Least Significant)<br>See Table 448. |

### 70.6.2.30 Error Status (ERRSR)

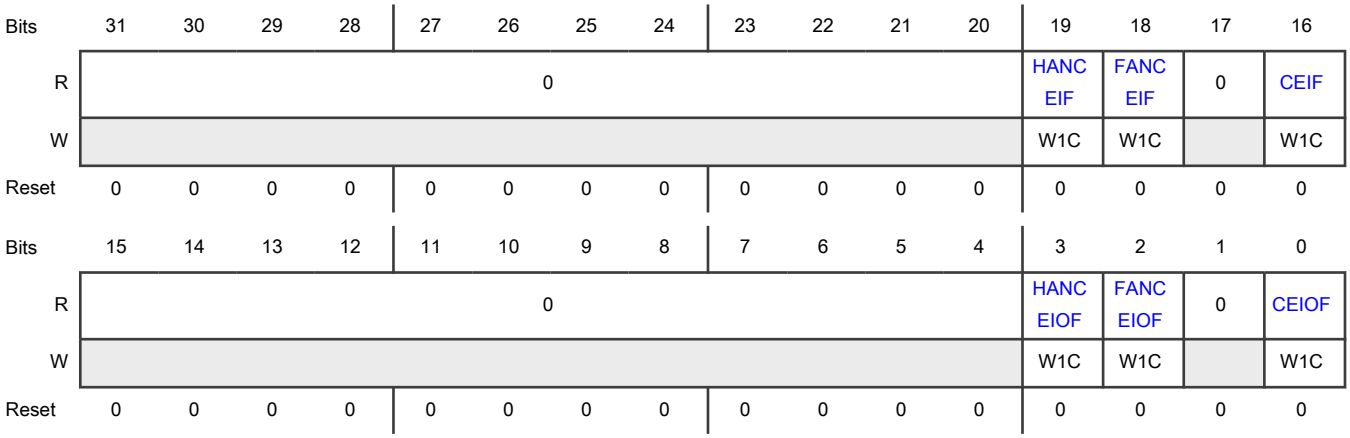
#### Offset

| Register | Offset |
|---|---|
| ERRSR | AFCh |

#### Function

Contains the status bits of the error correction and detection operations. These flags can be cleared by writing 1 to them. Writing 0 has no effect.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19<br>HANCEIF | Host Access with Noncorrectable Error Interrupt Flag<br>Indicates that a noncorrectable error was detected in a memory read initiated by host. A bus transfer error is asserted for that access. If MECR[HANCEI_MSK] = 1, the interrupt is 1.<br>0b - No errors detected<br>1b - Error detected |
| 18<br>FANCEIF | FlexCAN Access with Noncorrectable Error Interrupt Flag<br>Indicates that a noncorrectable error was detected in a memory read initiated by FlexCAN internal processes. If MECR[FANCEI_MSK] = 1, the interrupt is 1.<br>0b - No errors detected<br>1b - Error detected |
| 17 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 16<br>CEIF | Correctable Error Interrupt Flag<br>Indicates that a correctable error was detected in a memory read. If MECR[CEI_MSK] = 1, the interrupt is 1.<br>0b - No errors detected<br>1b - Error detected |
| 15-4<br>— | Reserved |
| 3<br>HANCEIOF | Host Access with Noncorrectable Error Interrupt Overrun Flag<br>Indicates that a noncorrectable error was detected in a memory read initiated by host when ERRSR[HANCEIF] was set. No interrupt is associated with this flag. See Error indication.<br>0b - No errors detected<br>1b - Error detected |
| 2<br>FANCEIOF | FlexCAN Access with Noncorrectable Error Interrupt Overrun Flag<br>Indicates that a noncorrectable error was detected in a memory read initiated by FlexCAN internal processes when ERRSR[FANCEIF] was set. No interrupt is associated with this flag. See Error indication.<br>0b - No errors detected<br>1b - Error detected |
| 1<br>— | Reserved |
| 0<br>CEIOF | Correctable Error Interrupt Overrun Flag<br>Indicates that a correctable error was detected in a memory read when ERRSR[CEIF] was set. No interrupt is associated with this flag. See Error indication.<br>0b - No errors detected<br>1b - Error detected |

### 70.6.2.31 Enhanced CAN Bit Timing Prescalers (EPRS)

#### Offset

| Register | Offset |
|---|---|
| EPRS | BF0h |

#### Function

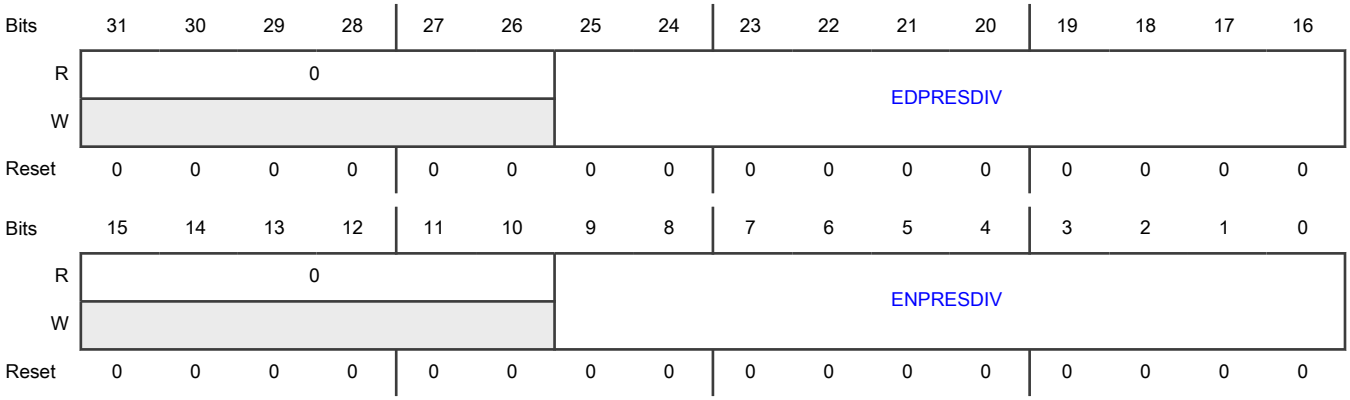
Defines the CAN bit timing prescalers for the nominal phase and data phase when CTRL2[BTE] = 1.

Used by the module only if CTRL2[BTE] = 1; otherwise, a write operation has no effect and all fields are read as zero.

This register can be written only in Freeze mode; the module blocks it in other modes.

Soft reset does not affect the contents of this register.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25-16<br>EDPRESDIV | Extended Data Phase Prescaler Division Factor<br>Defines the ratio between the PE clock frequency and the Sclock frequency in the data phase of a CAN FD message when CTRL2[BTE] = 1.<br>The Sclock period defines the time quantum of the CAN FD protocol for the data bit rate.<br>Sclock frequency = PE clock frequency ÷ (EDPRESDIV + 1)<br>**NOTE**<br>To minimize errors when processing FD frames, use the same value for this field and for EPRS[ENPRESDIV]. See the first note in CAN FD frames for details. |
| 15-10<br>— | Reserved |
| 9-0<br>ENPRESDIV | Extended Nominal Prescaler Division Factor<br>Defines the ratio between the PE clock frequency and the Sclock frequency when CTRL2[BTE] = 1. Otherwise, it reads as 0 and a write operation has no effect.<br>The Sclock period defines the time quantum of the CAN protocol in the nominal phase. For the reset value, the Sclock frequency is equal to the PE clock frequency (see Protocol timing).<br>Sclock frequency = PE clock frequency ÷ (ENPRESDIV + 1) |

### 70.6.2.32 Enhanced Nominal CAN Bit Timing (ENCBT)

#### Offset

| Register | Offset |
|---|---|
| ENCBT | BF4h |

#### Function

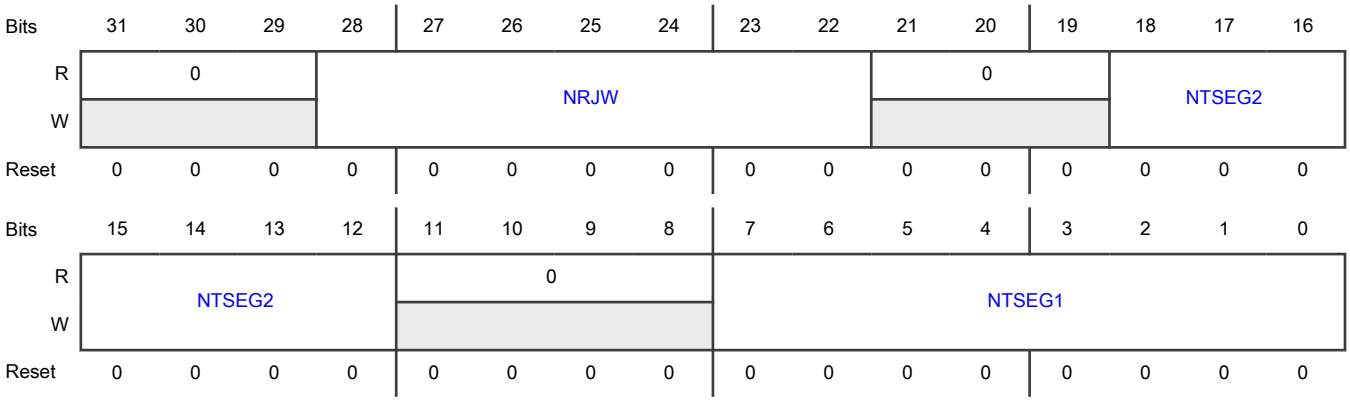
Provides an alternative way to store the CAN bit timing variables described in Control 1 (CTRL1) and CAN Bit Timing (CBT), to get higher CAN bit timing resolution.

This register is used by the module only if CTRL2[BTE] = 1. Otherwise, a write operation has no effect and all fields are read as zero.

Soft reset does not affect the contents of this register.

This register can be written only in Freeze mode; the module blocks it in other modes.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |
| 28-22<br>NRJW | Nominal Resynchronization Jump Width<br>Defines the maximum number of time quanta that one resynchronization can change a nominal bit time when CTRL2[BTE] = 1. Otherwise, it has no effect.<br>One time quantum = one Sclock period<br>Nominal Resync Jump Width = NRJW + 1 |
| 21-19<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 18-12<br>NTSEG2 | Nominal Time Segment 2<br>Defines the length of Time Segment 2 in the nominal bit time when CTRL2[BTE] = 1. Otherwise, it has no effect.<br>Nominal Time Segment 2 = (NTSEG2 + 1) × Time Quanta<br>One time quantum = one Sclock period |
| 11-8<br>— | Reserved |
| 7-0<br>NTSEG1 | Nominal Time Segment 1<br>Defines the length of Time Segment 1 in the bit time when CTRL2[BTE] = 1. Otherwise, it has no effect.<br>Nominal Time Segment 1 = (NTSEG1 + 1) × Time Quanta<br>One time quantum = one Sclock period |

### 70.6.2.33 Enhanced Data Phase CAN Bit Timing (EDCBT)

#### Offset

| Register | Offset |
|---|---|
| EDCBT | BF8h |

#### Function

Provides an alternative way to store the data phase CAN bit timing variables described in CAN FD Bit Timing (FDCBT) to achieve higher CAN bit timing resolution.

This register is used by the module only if CTRL2[BTE] = 1; otherwise, a write operation has no effect and all fields are read as zero.

Soft reset does not affect the contents of this register.

This register can be written only in Freeze mode; the module blocks it in other modes.

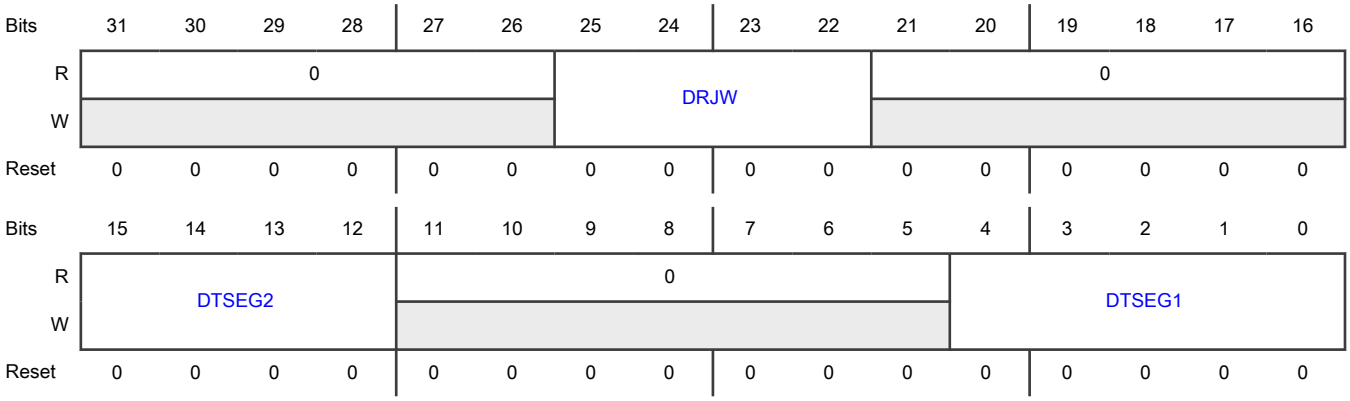
**NOTE**

Ensure that bit time settings and protocol engine tolerance are in compliance with the CAN Protocol standard (ISO 11898-1:2015).

**NOTE**

DTSEG1 must be at least two time quanta.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25-22<br>DRJW | Data Phase Resynchronization Jump Width<br>Defines the maximum number of time quanta that one resynchronization can change a data phase bit time when CTRL2[BTE] = 1. Otherwise, it has no effect.<br>Data Phase Resync Jump Width = DRJW + 1. |
| 21-16<br>— | Reserved |
| 15-12<br>DTSEG2 | Data Phase Time Segment 2<br>Defines the length of time segment 2 in the data phase bit time when CTRL2[BTE] = 1. Otherwise, it has no effect.<br>Data Phase Time Segment 2 = (DTSEG2 + 1) × Time Quanta.<br>One Time Quantum = one Sclock period. |
| 11-5<br>— | Reserved |
| 4-0<br>DTSEG1 | Data Phase Segment 1<br>Defines the length of time segment 1 in the data phase bit time when CTRL2[BTE] = 1. Otherwise, it has no effect.<br>Data Phase Time Segment 1 = (NTSEG1 + 1) × Time Quanta.<br>One Time Quantum = one Sclock period. |

### 70.6.2.34 Enhanced Transceiver Delay Compensation (ETDC)

#### Offset

| Register | Offset |
|---|---|
| ETDC | BFCh |

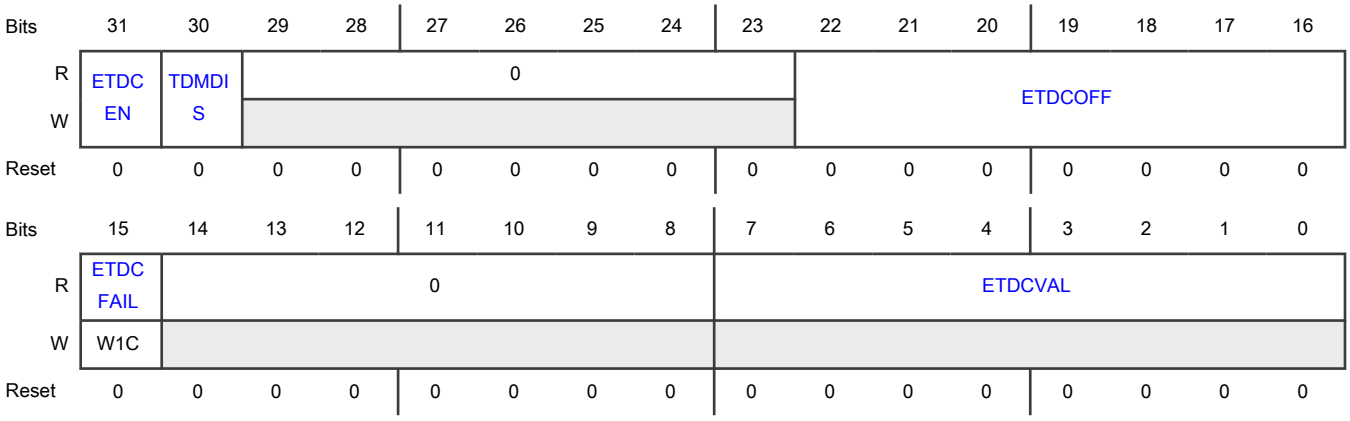
#### Function

Contains extended versions of FDCTRL[TDCOFF] and FDCTRL[TDCVAL]. This register is used by the module only if CTRL2[BTE] = 1. Otherwise, a write operation has no effect and all fields are read as zero.

**NOTE**

See Transmitter delay compensation in the CAN Protocol standard (ISO 11898-1:2015) for details.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31<br>ETDCEN | Transceiver Delay Compensation Enable<br>Enables the TDC feature. It can be written in Freeze mode only.<br>**NOTE**<br>See Transmitter delay compensation in the CAN Protocol standard (ISO 11898-1:2015) for details.<br>**NOTE**<br>TDC must be disabled when the Loop Back Mode is enabled. See CTRL1[LPB].<br>0b - Disable<br>1b - Enable |
| 30<br>TDMDIS | Transceiver Delay Measurement Disable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Disables the transceiver delay measurement. When the TDC measurement is disabled, only ETDC[ETDCOFF] determines the secondary sample point position. If TCD measurement is enabled, the sum of the transceiver delay measurement plus the enhanced TDC offset determines the secondary sample point position.<br>Soft reset does not affect this field.<br>**NOTE**<br>This bit can be enabled only if CTRL2[BTE] = 1.<br>0b - Enable<br>1b - Disable |
| 29-23<br>— | Reserved |
| 22-16<br>ETDCOFF | Enhanced Transceiver Delay Compensation Offset<br>Contains the offset value to be added to the loop delay of the measured transceiver. This value defines the position of the delayed comparison point when bit rate switching is active. See Transceiver delay compensation for details on how the loop delay measurement is performed.<br>This field can be written in Freeze mode only. Its value can be defined in protocol engine (PE) clock periods (CANCLK, see Protocol timing for more details). It must be smaller than the CAN bit duration in the data bit rate for proper operation.<br>Do not write 0 to this field.<br>**NOTE**<br>If CTRL2[BTE] becomes 1 after a chip-level hard reset, this field is read as 1h. |
| 15<br>ETDCFAIL | Transceiver Delay Compensation Fail<br>Indicates whether the transceiver delay compensation (TDC) mechanism is out of range. In this case, it is unable to compensate the loop delay of the transceiver and successfully compare the delayed received bits to the transmitted ones. (See Transceiver delay compensation.) This field becomes 0 the first time FlexCAN detects the out of range condition.<br>0b - In range<br>1b - Out of range |
| 14-8<br>— | Reserved |
| 7-0<br>ETDCVAL | Enhanced Transceiver Delay Compensation Value<br>Contains ETDC[ETDCOFF] added to the measured value of the transceiver loop delay in the latest transmitted CAN FD frame, with BRS = 1.<br>The module only updates this field when ETDC[ETDCEN] = 1.<br>Soft reset affects this field. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>If ETDC[TDMDIS] = 1, this field stores ETDC[ETDCOFF] only. |

### 70.6.2.35 CAN FD Control (FDCTRL)

#### Offset

| Register | Offset |
|---|---|
| FDCTRL | C00h |

#### Function

Contains control bits for CAN FD operation. It also defines the data size of message buffers allocated in different partitions of RAM (memory blocks) as described in Table 449.

When an 8-byte payload is selected:

- Block R0 allocates MB0–MB31.
- Block R1 allocates MB32–MB63.
- Block R2 allocates MB64–MB95.

When a payload larger than eight bytes is selected, the maximum number of message buffers in a block is limited as described below.

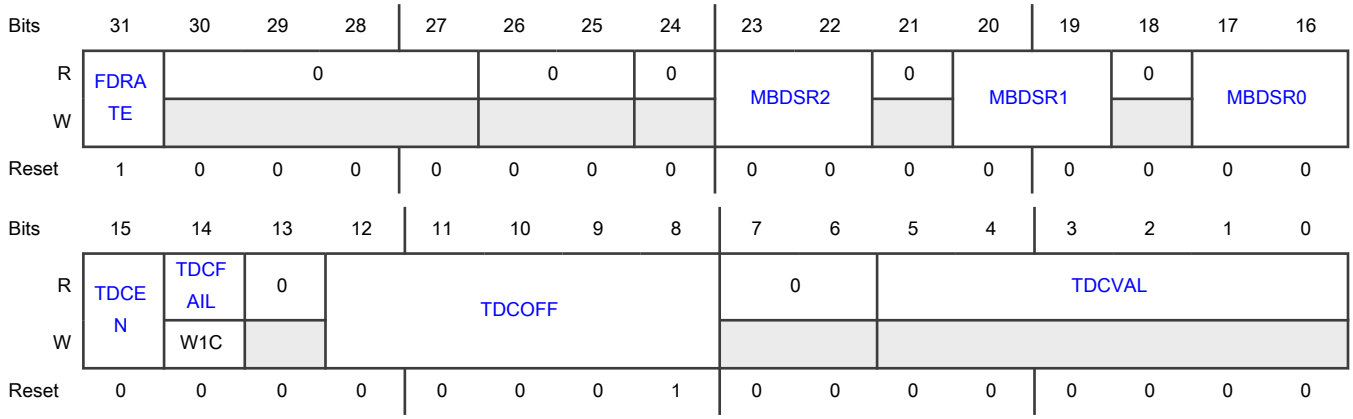
**Table 449. Number of message buffers**

| Payload size | Maximum number of message buffers per RAM block |
|---|---|
| 8 bytes | 32 |
| 16 bytes | 21 |
| 32 bytes | 12 |
| 64 bytes | 7 |

One memory block fits exactly 32 message buffers with an 8-byte payload. For other possible payload sizes, empty memory may exist between the last message buffer in a block and the beginning of the next block. This empty memory corresponds to less than one message buffer, and must not be used.

Soft reset does not affect the contents of this register.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>FDRATE | Bit Rate Switch Enable<br>Enables the effect of the Bit Rate Switch (BRS bit) during the data phase of TX messages. When 1, if BRS = 1 in the TX message buffer, frames are transmitted with bit rate switching. When 0, frames are transmitted at a nominal rate, and the BRS bit in the TX MB has no effect.<br>The CPU can write to this field at any time. However, its effect becomes active only under one of these conditions:<br>• The CAN bus is in the Wait for Bus Idle state.<br>• The CAN bus is in the Bus Idle state.<br>• The CAN bus is in the Bus Off state.<br>• The current frame under reception or transmission reaches the interframe space.<br>By writing 0 to FDCTRL[FDRATE], the CPU can force all bits in CAN FD messages to be transmitted at nominal bit rate. This transmission occurs regardless of the value in the BRS bit of the TX message buffers.<br>0b - Disable<br>1b - Enable |
| 30-27<br>— | Reserved |
| 26-25<br>— | Reserved |
| 24<br>— | Reserved |
| 23-22<br>MBDSR2 | Message Buffer Data Size for Region 2<br>Selects the data size per message buffer for region R2 of message buffers allocated in RAM.<br>This field can be written in Freeze mode only. |

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>FLEXCAN_0</td><td>FDCTRL</td><td>—</td></tr>
<tr><td>FLEXCAN_1</td><td>—</td><td>FDCTRL</td></tr>
<tr><td>FLEXCAN_2</td><td>—</td><td>FDCTRL</td></tr>
<tr><td>FLEXCAN_3</td><td>—</td><td>FDCTRL</td></tr>
<tr><td>FLEXCAN_4</td><td>—</td><td>FDCTRL</td></tr>
<tr><td>FLEXCAN_5</td><td>—</td><td>FDCTRL</td></tr>
</table>
00b - 8 bytes<br>
01b - 16 bytes<br>
10b - 32 bytes<br>
11b - 64 bytes
</td></tr>
<tr><td>21<br>—</td><td>Reserved</td></tr>
<tr><td>20-19<br>MBDSR1</td><td>
Message Buffer Data Size for Region 1<br>
Selects the data size per message buffer for region R1 of message buffers allocated in RAM.<br>
This field can be written in Freeze mode only.
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>FLEXCAN_0</td><td>FDCTRL</td><td>—</td></tr>
<tr><td>FLEXCAN_1</td><td>FDCTRL</td><td>—</td></tr>
<tr><td>FLEXCAN_2</td><td>FDCTRL</td><td>—</td></tr>
<tr><td>FLEXCAN_3</td><td>—</td><td>FDCTRL</td></tr>
</table>
</td></tr>
</table>

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>FLEXCAN_4</td><td>—</td><td>FDCTRL</td></tr>
<tr><td>FLEXCAN_5</td><td>—</td><td>FDCTRL</td></tr>
</table>
00b - 8 bytes<br>
01b - 16 bytes<br>
10b - 32 bytes<br>
11b - 64 bytes</td></tr>
<tr><td>18<br>—</td><td>Reserved</td></tr>
<tr><td>17-16<br>MBDSR0</td><td>Message Buffer Data Size for Region 0<br>Selects the data size per message buffer for region R0 of message buffers allocated in RAM.<br>This field can be written in Freeze mode only.<br>00b - 8 bytes<br>01b - 16 bytes<br>10b - 32 bytes<br>11b - 64 bytes</td></tr>
<tr><td>15<br>TDCEN</td><td>Transceiver Delay Compensation Enable<br>Enables the TDC feature. It can be written in Freeze mode only.<br>See Transmitter delay compensation in the CAN Protocol standard (ISO 11898-1:2015) for details.<br>TDC must be disabled when Loopback mode is enabled (see CTRL1[LPB]).<br>NOTE<br>If CTRL2[BTE] = 1, this field is read as 0 and a write operation has no effect.<br>0b - Disable<br>1b - Enable</td></tr>
<tr><td>14<br>TDCFAIL</td><td>Transceiver Delay Compensation Fail<br>Indicates whether the Transceiver Delay Compensation (TDC) mechanism is out of range. In this case, the mechanism cannot compensate for the loop delay of the transceiver and successfully compare the delayed received bits to the transmitted ones (see Transceiver delay compensation). The first time that FlexCAN detects the out-of-range condition, this field becomes 1.<br>NOTE<br>If CTRL2[BTE] = 1, this field is read as 0 and a write operation has no effect.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - In range<br>1b - Out of range |
| 13<br>— | Reserved |
| 12-8<br>TDCOFF | Transceiver Delay Compensation Offset<br>Contains the offset value to be added to the loop delay of the measured transceiver. This value defines the position of the delayed comparison point when bit rate switching is active. See Transceiver delay compensation for details about loop delay measurement.<br>This field can be written in Freeze mode only. Its value can be defined in Protocol Engine Clock periods (CANCLK, see Protocol timing for more details). The value must be smaller than the CAN bit duration in the data bit rate for proper operation.<br>**NOTE**<br>If CTRL2[BTE] = 1, TDCOFF is read as 0 and a write operation has no effect.<br>Do not write 0 to this field. |
| 7-6<br>— | Reserved |
| 5-0<br>TDCVAL | Transceiver Delay Compensation Value<br>Contains the value of the transceiver loop delay measured from the transmitted EDL-to-R0 transition edge to the respective received one added to FDCTRL[TDCOFF]. This value is an integer multiple of the Protocol Engine Clock period (CANCLK).<br>If CTRL2[BTE] = 1, this field is read as 0.<br>See Protocol timing for details on the loop delay measurement. |

### 70.6.2.36 CAN FD Bit Timing (FDCBT)

#### Offset

| Register | Offset |
|---|---|
| FDCBT | C04h |

#### Function

Stores the CAN bit timing variables used in the data phase of CAN FD messages when the FDCTRL[FDRATE] = 1, compatible with the CAN FD specification. Fields in this register define:

- The time quantum duration
- The number of time quanta per CAN bit
- The sample point position for the data bit rate portion of a CAN FD message with BRS = 1

Soft reset does not affect the contents of this register.

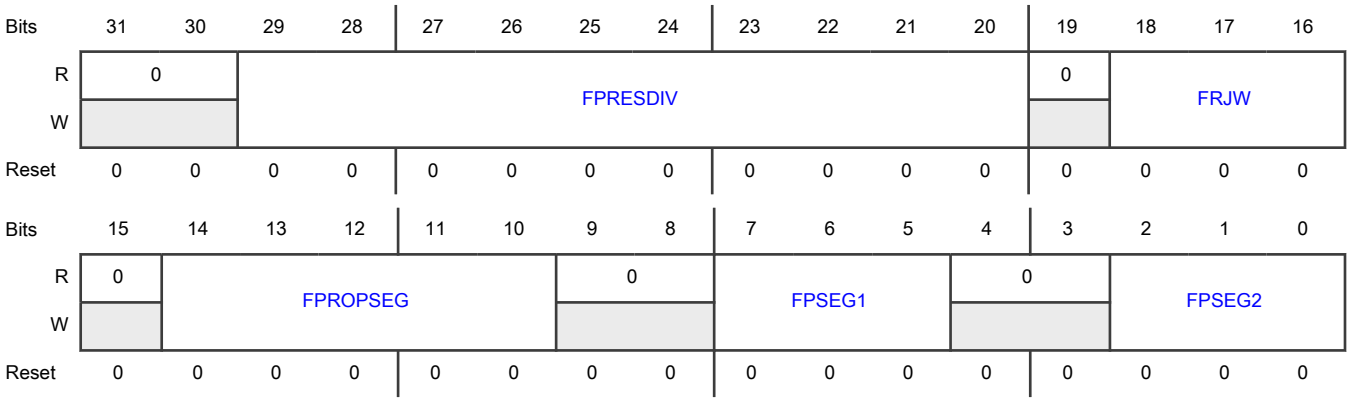
The sum of the Fast Propagation Segment (FPROPSEG) and Fast Phase Segment 1 (FPSEG1) must be at least two time quanta.

Ensure bit time settings and protocol engine tolerance are in compliance with the CAN Protocol standard (ISO 11898-1:2015).

**NOTE**

If CTRL2[BTE] = 1, this register is read as zero and a write operation has no effect.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-20<br>FPRESDIV | Fast Prescaler Division Factor<br>Defines the ratio between the PE clock frequency and the serial clock (Sclock) frequency in the data bit rate portion of a CAN FD message with BRS = 1.<br>The Sclock period defines the time quantum of the CAN FD protocol for the data bit rate. This field can be written only in Freeze mode; the module blocks it in other modes.<br>Sclock frequency = PE clock frequency ÷ (FPRESDIV + 1).<br>**NOTE**<br>To minimize errors when processing FD frames, use the same value for this field and for CTRL1[PRESDIV] or CBT[EPRESDIV]. See the first note in CAN FD frames for details. |
| 19<br>— | Reserved |
| 18-16<br>FRJW | Fast Resync Jump Width<br>Defines the maximum number of time quanta that one resynchronization can change a bit time in the data bit rate portion of a CAN FD message with BRS = 1.<br>This field can be written only in Freeze mode; the module blocks it in other modes.<br>Resync Jump Width = FSJW + 1. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | One Time Quantum = one Sclock period. |
| 15<br>— | Reserved |
| 14-10<br>FPROPSEG | Fast Propagation Segment<br>Defines the length of the propagation segment in the bit time in the data bit rate portion of a CAN FD message with BRS = 1. This field can be written only in Freeze mode; the module blocks it in other modes.<br>Propagation Segment Time = FPROPSEG × Time Quanta.<br>One Time Quantum = one Sclock period. |
| 9-8<br>— | Reserved |
| 7-5<br>FPSEG1 | Fast Phase Segment 1<br>Defines the length of phase segment 1 in the bit time in the data bit rate portion of a CAN FD message with BRS = 1. This field can be written only in Freeze mode; the module blocks it in other modes.<br>Phase Segment 1 = (FPSEG1 + 1) × Time Quanta.<br>One Time Quantum = one Sclock period. |
| 4-3<br>— | Reserved |
| 2-0<br>FPSEG2 | Fast Phase Segment 2<br>Defines the length of phase segment 2 in the data bit rate portion of a CAN FD message with BRS = 1. This field can be written only in Freeze mode; the module blocks it in other modes.<br>Phase Segment 2 = (FPSEG2 + 1) × Time Quanta.<br>One Time Quantum = one Sclock period. |

### 70.6.2.37 CAN FD CRC (FDCRC)

#### Offset

| Register | Offset |
|---|---|
| FDCRC | C08h |

#### Function

Provides information about the cyclic redundancy check (CRC) of transmitted messages.

FlexCAN uses different CRC polynomials for different frame formats.

- The CRC_15 polynomial is used for all frames in CAN format.
- The CRC_17 polynomial is used for frames in CAN FD format with a DATA FIELD up to 16 bytes.

- The CRC_21 polynomial is used for frames in CAN FD format with a DATA FIELD longer than 16 bytes.

Each polynomial shown below results in a Hamming distance of 6. This register is updated at the same time that the TX Interrupt flag is set.

$$CRC_15 = C599h:\ (x^{15} + x^{14} + x^{10} + x^{8} + x^{7} + x^{4} + x^{3} + 1)$$

$$CRC_17 = 3685Bh:\ (x^{17} + x^{16} + x^{14} + x^{13} + x^{11} + x^{6} + x^{4} + x^{3} + x^{1} + 1)$$

$$CRC_21 = 302899h:\ (x^{21} + x^{20} + x^{13} + x^{11} + x^{7} + x^{4} + x^{3} + 1)$$

**Equation 31. CRC polynomial used on CAN frame**

**NOTE**

See CRC sequence calculation in the CAN Protocol standard (ISO 11898-1:2015) for details.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-24<br>FD_MBCRC | CRC Message Buffer Number for FD_TXCRC<br>Indicates the number of the message buffer corresponding to the value in FDCRC[FD_TXCRC], for both FD and non-FD frames.<br>It reports the same information as in CRCR[MBCRC]. |
| 23-21<br>— | Reserved |
| 20-0<br>FD_TXCRC | Extended Transmitted CRC value<br>Contains the CRC value calculated over the most recent transmitted message. Different CRC polynomials are used for different frame formats. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | For CRC_15 and CRC_17, the six most significant bits and the four most significant bits are reported as zeroes, respectively.<br>For CRC_15, this field has the same content as Cyclic Redundancy Check (CRCR). |

### 70.6.2.38 Enhanced RX FIFO Control (ERFCR)

#### Offset

| Register | Offset |
|---|---|
| ERFCR | C0Ch |

#### Function

Defines the Enhanced RX FIFO configuration.

This register can be written only in Freeze mode.

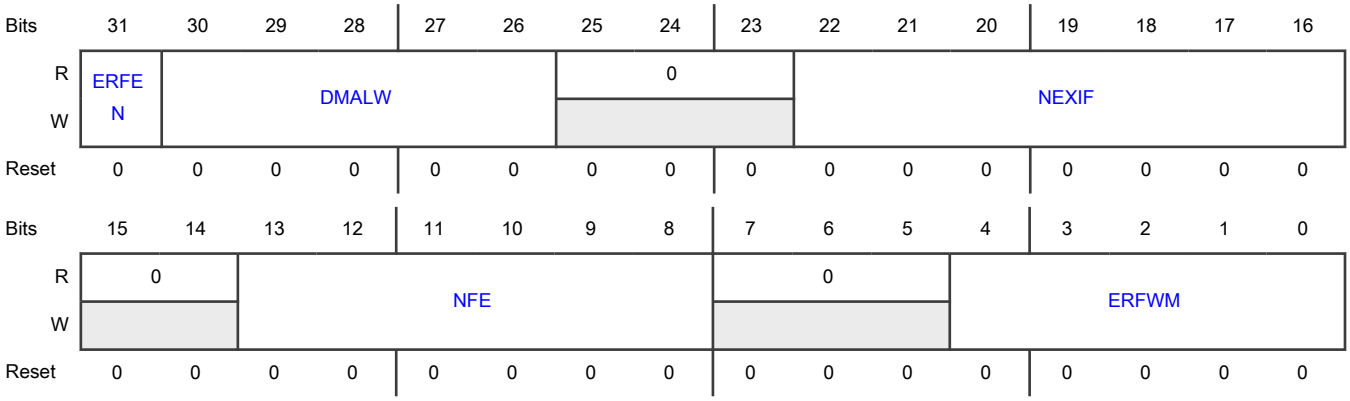
Soft reset does not affect any of the contents of this register.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| FLEXCAN_0 | ERFCR | — |
| FLEXCAN_1 | — | ERFCR |
| FLEXCAN_2 | — | ERFCR |
| FLEXCAN_3 | — | ERFCR |
| FLEXCAN_4 | — | ERFCR |
| FLEXCAN_5 | — | ERFCR |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>ERFEN | Enhanced RX FIFO enable<br>Enables the Enhanced RX FIFO.<br>**NOTE**<br>If MCR[RFEN] = 1, do not write 1 to this field.<br>0b - Disable<br>1b - Enable |
| 30-26<br>DMALW | DMA Last Word<br>Defines the last DMA address for each Enhanced RX FIFO element.<br>This table shows the number of elements and the last address for each Enhanced RX FIFO element according to the value of DMALW. |

| DMALW | Number of 32-bit words transferred | Last FIFO address |
|---|---|---|
| 0 | 1 | 2000h |
| 1 | 2 | 2004h |
| 2 | 3 | 2008h |
| 3 | 4 | 200Ch |
| 4 | 5 | 2010h |
| 5 | 6 | 2014h |
| 6 | 7 | 2018h |
| 7 | 8 | 201Ch |
| 8 | 9 | 2020h |
| 9 | 10 | 2024h |
| 10 | 11 | 2028h |

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>DMALW</th><th>Number of 32-bit words transferred</th><th>Last FIFO address</th></tr>
<tr><td>11</td><td>12</td><td>202Ch</td></tr>
<tr><td>12</td><td>13</td><td>2030h</td></tr>
<tr><td>13</td><td>14</td><td>2034h</td></tr>
<tr><td>14</td><td>15</td><td>2038h</td></tr>
<tr><td>15</td><td>16</td><td>203Ch</td></tr>
<tr><td>16</td><td>17</td><td>2040h</td></tr>
<tr><td>17</td><td>18</td><td>2044h</td></tr>
<tr><td>18</td><td>19</td><td>2048h</td></tr>
<tr><td>19</td><td>20</td><td>204Ch</td></tr>
</table>
NOTE<br>Undefined DMALW values in the table are reserved and must not be used.
</td></tr>
<tr><td>25-23<br>—</td><td>Reserved</td></tr>
<tr><td>22-16<br>NEXIF</td><td>Number of Extended ID Filter Elements<br>Defines the number of extended ID filter elements used during the Enhanced RX FIFO matching process.<br>The value of this field must be less than or equal to NFE + 1.<br>The number of standard ID filter elements is 2 × (NFE - NEXIF + 1).<br>This table shows the number of extended ID filters and standard ID filters available for Enhanced RX FIFO if all filter elements are used.
<table>
<tr><th>NEXIF</th><th>NFE</th><th>Number of Extended ID filter elements</th><th>Number of Standard ID filter elements</th></tr>
<tr><td>0</td><td>63</td><td>0</td><td>128</td></tr>
<tr><td>1</td><td>63</td><td>1</td><td>126</td></tr>
<tr><td>2</td><td>63</td><td>2</td><td>124</td></tr>
<tr><td>3</td><td>63</td><td>3</td><td>122</td></tr>
<tr><td>4</td><td>63</td><td>4</td><td>120</td></tr>
<tr><td>5</td><td>63</td><td>5</td><td>118</td></tr>
<tr><td>6</td><td>63</td><td>6</td><td>116</td></tr>
<tr><td>7</td><td>63</td><td>7</td><td>114</td></tr>
</table>
</td></tr>
</table>

| Field | Function |
|---|---|

| NEXIF | NFE | Number of Extended ID filter elements | Number of Standard ID filter elements |
|---|---|---|---|
| 8 | 63 | 8 | 112 |
| 9 | 63 | 9 | 110 |
| 10 | 63 | 10 | 108 |
| 11 | 63 | 11 | 106 |
| 12 | 63 | 12 | 104 |
| 13 | 63 | 13 | 102 |
| 14 | 63 | 14 | 100 |
| 15 | 63 | 15 | 98 |
| 16 | 63 | 16 | 96 |
| 17 | 63 | 17 | 94 |
| 18 | 63 | 18 | 92 |
| 19 | 63 | 19 | 90 |
| 20 | 63 | 20 | 88 |
| 21 | 63 | 21 | 86 |
| 22 | 63 | 22 | 84 |
| 23 | 63 | 23 | 82 |
| 24 | 63 | 24 | 80 |
| 25 | 63 | 25 | 78 |
| 26 | 63 | 26 | 76 |
| 27 | 63 | 27 | 74 |
| 28 | 63 | 28 | 72 |
| 29 | 63 | 29 | 70 |
| 30 | 63 | 30 | 68 |
| 31 | 63 | 31 | 66 |
| 32 | 63 | 32 | 64 |
| 33 | 63 | 33 | 62 |
| 34 | 63 | 34 | 60 |
| 35 | 63 | 35 | 58 |
| 36 | 63 | 36 | 56 |

*Table continued from the previous page...*

| Field | Function | | | |
|---|---|---|---|---|
| | NEXIF | NFE | Number of Extended ID filter elements | Number of Standard ID filter elements |
| | 37 | 63 | 37 | 54 |
| | 38 | 63 | 38 | 52 |
| | 39 | 63 | 39 | 50 |
| | 40 | 63 | 40 | 48 |
| | 41 | 63 | 41 | 46 |
| | 42 | 63 | 42 | 44 |
| | 43 | 63 | 43 | 42 |
| | 44 | 63 | 44 | 40 |
| | 45 | 63 | 45 | 38 |
| | 46 | 63 | 46 | 36 |
| | 47 | 63 | 47 | 34 |
| | 48 | 63 | 48 | 32 |
| | 49 | 63 | 49 | 30 |
| | 50 | 63 | 50 | 28 |
| | 51 | 63 | 51 | 26 |
| | 52 | 63 | 52 | 24 |
| | 53 | 63 | 53 | 22 |
| | 54 | 63 | 54 | 20 |
| | 55 | 63 | 55 | 18 |
| | 56 | 63 | 56 | 16 |
| | 57 | 63 | 57 | 14 |
| | 58 | 63 | 58 | 12 |
| | 59 | 63 | 59 | 10 |
| | 60 | 63 | 60 | 8 |
| | 61 | 63 | 61 | 6 |
| | 62 | 63 | 62 | 4 |
| | 63 | 63 | 63 | 2 |
| | 64 | 63 | 64 | 0 |

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>15-14<br>—</td><td>Reserved</td></tr>
<tr><td>13-8<br>NFE</td><td>Number of Enhanced RX FIFO Filter Elements<br>Defines the total number of filter elements used during the enhanced RX FIFO matching process according to the table.
<table>
<tr><th>NFE</th><th>Maximum number of extended ID filter elements (NEXIF = NFE + 1)</th><th>Maximum number of standard ID filter elements (NEXIF = 0)</th></tr>
<tr><td>0</td><td>1</td><td>2</td></tr>
<tr><td>1</td><td>2</td><td>4</td></tr>
<tr><td>2</td><td>3</td><td>6</td></tr>
<tr><td>3</td><td>4</td><td>8</td></tr>
<tr><td>4</td><td>5</td><td>10</td></tr>
<tr><td>5</td><td>6</td><td>12</td></tr>
<tr><td>6</td><td>7</td><td>14</td></tr>
<tr><td>7</td><td>8</td><td>16</td></tr>
<tr><td>8</td><td>9</td><td>18</td></tr>
<tr><td>9</td><td>10</td><td>20</td></tr>
<tr><td>10</td><td>11</td><td>22</td></tr>
<tr><td>11</td><td>12</td><td>24</td></tr>
<tr><td>12</td><td>13</td><td>26</td></tr>
<tr><td>13</td><td>14</td><td>28</td></tr>
<tr><td>14</td><td>15</td><td>30</td></tr>
<tr><td>15</td><td>16</td><td>32</td></tr>
<tr><td>16</td><td>17</td><td>34</td></tr>
<tr><td>17</td><td>18</td><td>36</td></tr>
<tr><td>18</td><td>19</td><td>38</td></tr>
<tr><td>19</td><td>20</td><td>40</td></tr>
<tr><td>20</td><td>21</td><td>42</td></tr>
<tr><td>21</td><td>22</td><td>44</td></tr>
<tr><td>22</td><td>23</td><td>46</td></tr>
<tr><td>23</td><td>24</td><td>48</td></tr>
</table>
</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function | | |
|---|---|---|---|
| | NFE | Maximum number of extended ID filter elements (NEXIF = NFE + 1) | Maximum number of standard ID filter elements (NEXIF = 0) |
| | 24 | 25 | 50 |
| | 25 | 26 | 52 |
| | 26 | 27 | 54 |
| | 27 | 28 | 56 |
| | 28 | 29 | 58 |
| | 29 | 30 | 60 |
| | 30 | 31 | 62 |
| | 31 | 32 | 64 |
| | 32 | 33 | 66 |
| | 33 | 34 | 68 |
| | 34 | 35 | 70 |
| | 35 | 36 | 72 |
| | 36 | 37 | 74 |
| | 37 | 38 | 76 |
| | 38 | 39 | 78 |
| | 39 | 40 | 80 |
| | 40 | 41 | 82 |
| | 41 | 42 | 84 |
| | 42 | 43 | 86 |
| | 43 | 44 | 88 |
| | 44 | 45 | 90 |
| | 45 | 46 | 92 |
| | 46 | 47 | 94 |
| | 47 | 48 | 96 |
| | 48 | 49 | 98 |
| | 49 | 50 | 100 |
| | 50 | 51 | 102 |
| | 51 | 52 | 104 |
| | 52 | 53 | 106 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | <table><tr><th>NFE</th><th>Maximum number of extended ID filter elements (NEXIF = NFE + 1)</th><th>Maximum number of standard ID filter elements (NEXIF = 0)</th></tr><tr><td>53</td><td>54</td><td>108</td></tr><tr><td>54</td><td>55</td><td>110</td></tr><tr><td>55</td><td>56</td><td>112</td></tr><tr><td>56</td><td>57</td><td>114</td></tr><tr><td>57</td><td>58</td><td>116</td></tr><tr><td>58</td><td>59</td><td>118</td></tr><tr><td>59</td><td>60</td><td>120</td></tr><tr><td>60</td><td>61</td><td>122</td></tr><tr><td>61</td><td>62</td><td>124</td></tr><tr><td>62</td><td>63</td><td>126</td></tr><tr><td>63</td><td>64</td><td>128</td></tr></table> |
| 7-5<br>— | Reserved |
| 4-0<br>ERFWM | Enhanced RX FIFO Watermark<br>Defines the minimum number of CAN messages stored in the Enhanced RX FIFO. When that number is reached, ERFSR[ERFWMI] becomes 1.<br>Minimum number of CAN messages = ERFWM + 1.<br>**NOTE**<br>If MCR[DMA] = 1, write 0h to this field. |

### 70.6.2.39 Enhanced RX FIFO Interrupt Enable (ERFIER)

**Offset**

| Register | Offset |
|---|---|
| ERFIER | C10h |

**Function**

Contains the interrupt enables for the Enhanced RX FIFO.

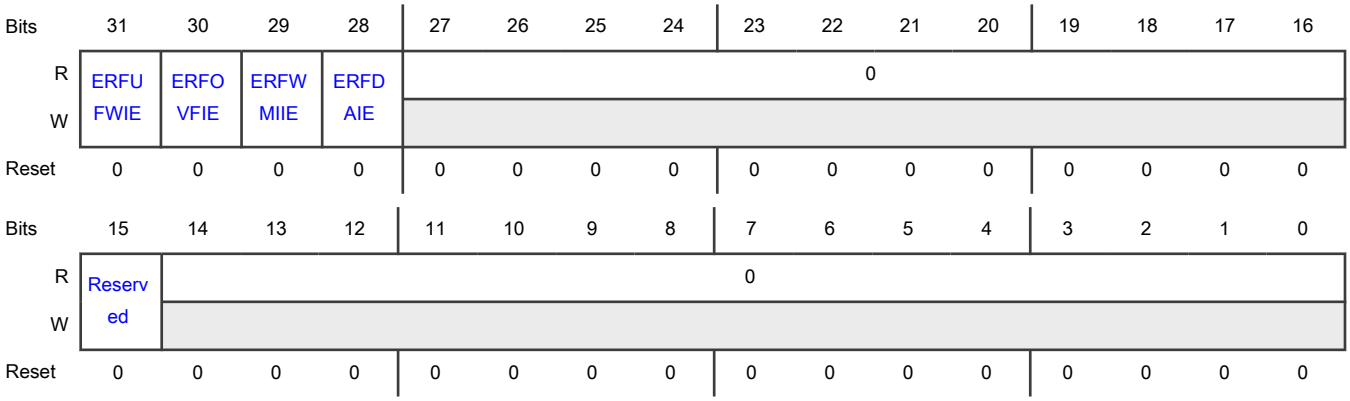
Soft reset does not affect this register.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| FLEXCAN_0 | ERFIER | — |
| FLEXCAN_1 | — | ERFIER |
| FLEXCAN_2 | — | ERFIER |
| FLEXCAN_3 | — | ERFIER |
| FLEXCAN_4 | — | ERFIER |
| FLEXCAN_5 | — | ERFIER |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>ERFUFWIE | Enhanced RX FIFO Underflow Interrupt Enable<br>Enables interrupt for ERFSR[ERFUFW].<br>0b - Disable<br>1b - Enable |
| 30<br>ERFOVFIE | Enhanced RX FIFO Overflow Interrupt Enable<br>Enables interrupt for ERFSR[ERFOVF].<br>0b - Disable<br>1b - Enable |
| 29<br>ERFWMIIE | Enhanced RX FIFO Watermark Indication Interrupt Enable<br>Enables interrupt for ERFSR[ERFWMI]. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disable<br>1b - Enable |
| 28<br>ERFDAIE | Enhanced RX FIFO Data Available Interrupt Enable<br>Enables interrupt for ERFSR[ERFDA].<br>0b - Disable<br>1b - Enable |
| 27-16<br>— | Reserved |
| 15<br>— | Reserved |
| 14-0<br>— | Reserved |

### 70.6.2.40 Enhanced RX FIFO Status (ERFSR)

#### Offset

| Register | Offset |
|---|---|
| ERFSR | C14h |

#### Function

Contains the status fields of the Enhanced RX FIFO including error indications and a clear FIFO field.

Soft reset does not affect this register.

**NOTE**

Each module instance supports a different number of registers.

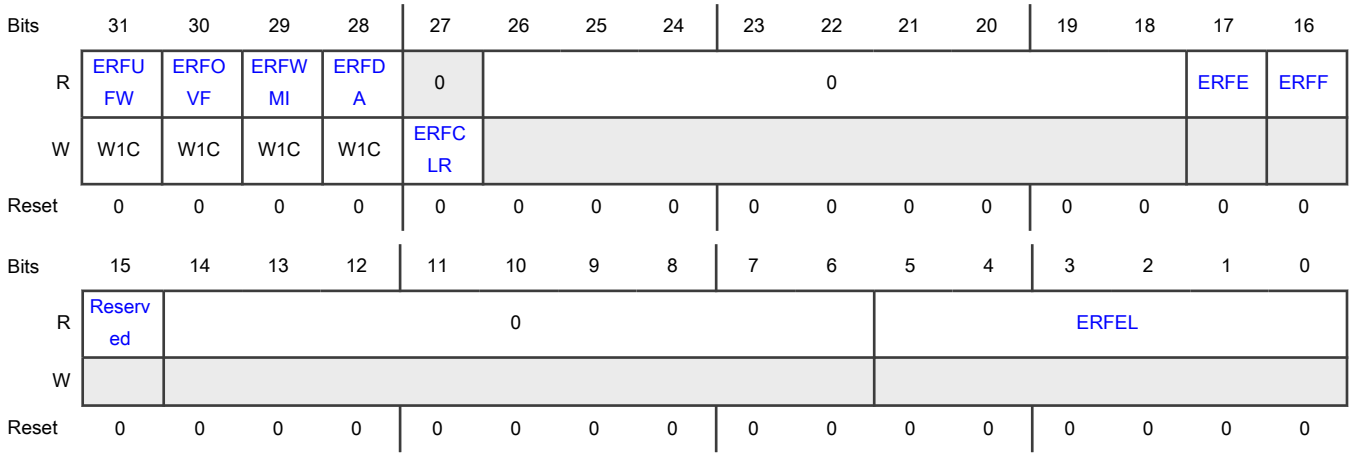
| Instance | Register supported | Register not supported |
|---|---|---|
| FLEXCAN_0 | ERFSR | — |
| FLEXCAN_1 | — | ERFSR |
| FLEXCAN_2 | — | ERFSR |
| FLEXCAN_3 | — | ERFSR |

*Table continues on the next page...*

*Table continued from the previous page...*

| Instance | Register supported | Register not supported |
|---|---|---|
| FLEXCAN_4 | — | ERFSR |
| FLEXCAN_5 | — | ERFSR |

## Diagram




## Fields

| Field | Function |
|---|---|
| 31<br>ERFUFW | Enhanced RX FIFO Underflow Flag<br>Indicates whether an underflow condition occurred in the enhanced RX FIFO.<br>If ERFIER[ERFUFWIE] = 1, this field generates an interrupt.<br>0b - No such occurrence<br>1b - Underflow |
| 30<br>ERFOVF | Enhanced RX FIFO Overflow Flag<br>Indicates whether an overflow condition occurred in the Enhanced RX FIFO.<br>If ERFIER[ERFOVFIE] = 1, this field generates an interrupt.<br>0b - No such occurrence<br>1b - Overflow |
| 29<br>ERFWMI | Enhanced RX FIFO Watermark Indication Flag<br>Indicates whether the number of messages available in the Enhanced RX FIFO is greater than the watermark defined in ERFCR[ERFWM].<br>If ERFIER[ERFWMIIE] = 1, this field generates an interrupt.<br>0b - No such occurrence |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Number of messages in FIFO is greater than the watermark |
| 28<br>ERFDA | Enhanced RX FIFO Data Available Flag<br>Indicates whether there is at least one message stored in the ERX FIFO.<br>If ERFIER[ERFDAIE] = 1, this field generates an interrupt.<br>0b - No such occurrence<br>1b - At least one message stored in Enhanced RX FIFO |
| 27<br>ERFCLR | Enhanced RX FIFO Clear<br>Writing 1 to this field during Freeze mode resets the internal FIFO pointers, but the FIFO content stored in RAM is not changed.<br>Writing to this field outside Freeze mode, or writing 0 to this field, has no effect.<br>0b - No effect<br>1b - Clear enhanced RX FIFO content |
| 26-18<br>— | Reserved |
| 17<br>ERFE | Enhanced RX FIFO Empty Flag<br>Indicates whether Enhanced RX FIFO is empty.<br>0b - Not empty<br>1b - Empty |
| 16<br>ERFF | Enhanced RX FIFO Full Flag<br>Indicates whether enhanced RX FIFO is full.<br>0b - Not full<br>1b - Full |
| 15<br>— | Reserved |
| 14-6<br>— | Reserved |
| 5-0<br>ERFEL | Enhanced RX FIFO Elements<br>Indicates the number of CAN messages stored in the Enhanced RX FIFO. |

### 70.6.2.41 High-Resolution Timestamp (HR_TIME_STAMP0 - HR_TIME_STAMP95)

**Offset**

For n = 0 to 95:

| Register | Offset |
|---|---|
| HR_TIME_STAMPn | C30h + (n × 4h) |

### Function

Stores a copy of a 32-bit timer at the start or end of a CAN frame.

HR_TIME_STAMP0 stores the 32-bit timestamp associated with MB0, HR_TIME_STAMP1 stores the 32-bit timestamp associated with MB1, and so on.

Reset does not affect these registers.

**NOTE**

Do not write to these registers outside Freeze mode.

**Table 450. High-Resolution Timestamp register operation**

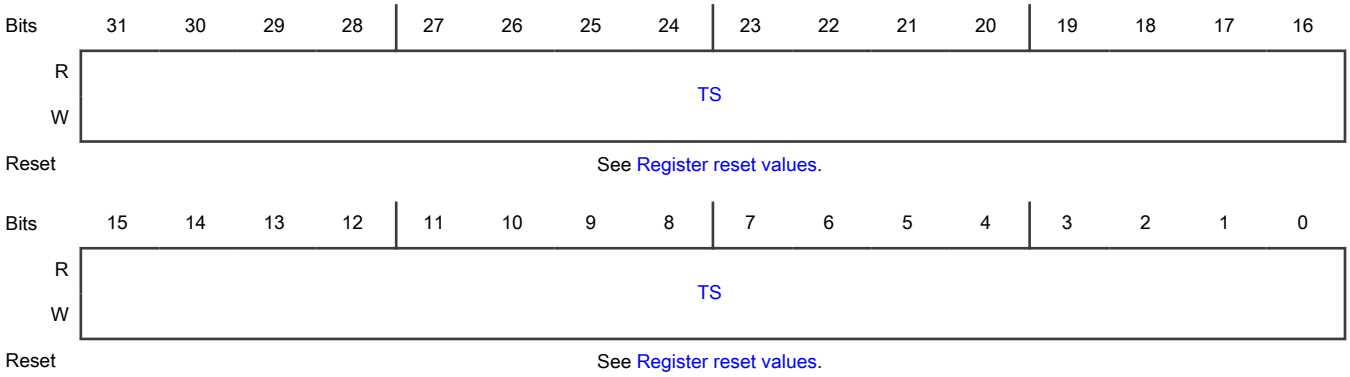
| TSTAMPCAP | Captured timebase | Capture point |
|---|---|---|
| b00 | None | None |
| b01 | 32 bits of high-resolution on-chip timer | Seventh bit of the end-of-frame field for transmission and sixth bit of the end-of-frame field for reception. |
| b10 | 32 bits of high-resolution on-chip timer | Start of frame |
| b11 | 32 bits of high-resolution on-chip timer | Start of frame for classical CAN frame format and res bit for CAN FD frame format |

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| FLEXCAN_0 | HR_TIME_STAMP0–HR_TIME_STAMP95 | — |
| FLEXCAN_1 | HR_TIME_STAMP0–HR_TIME_STAMP63 | HR_TIME_STAMP64–HR_TIME_STAMP95 |
| FLEXCAN_2 | HR_TIME_STAMP0–HR_TIME_STAMP63 | HR_TIME_STAMP64–HR_TIME_STAMP95 |
| FLEXCAN_3 | HR_TIME_STAMP0–HR_TIME_STAMP31 | HR_TIME_STAMP32–HR_TIME_STAMP95 |
| FLEXCAN_4 | HR_TIME_STAMP0–HR_TIME_STAMP31 | HR_TIME_STAMP32–HR_TIME_STAMP95 |
| FLEXCAN_5 | HR_TIME_STAMP0–HR_TIME_STAMP31 | HR_TIME_STAMP32–HR_TIME_STAMP95 |

### Diagram




### Register reset values

| Register | Reset value |
|---|---|
| HR_TIME_STAMP0–HR_TIME_STAMP31 | FLEXCAN_0–FLEXCAN_5: undefined |
| HR_TIME_STAMP32–HR_TIME_STAMP63 | FLEXCAN_0–FLEXCAN_2: undefined<br>FLEXCAN_3–FLEXCAN_5: Register not supported |
| HR_TIME_STAMP64–HR_TIME_STAMP95 | FLEXCAN_0: undefined<br>FLEXCAN_1–FLEXCAN_5: Register not supported |

### Fields

| Field | Function |
|---|---|
| 31-0<br>TS | High-Resolution Timestamp<br>Captures the copy of the timestamp of corresponding message buffer. This field always captures a 32-bit timer value. |

## 70.6.2.42 Enhanced RX FIFO Filter Element (ERFFEL0 - ERFFEL127)

### Offset

For n = 0 to 127:

| Register | Offset |
|---|---|
| ERFFELn | 3000h + (n × 4h) |

### Function

Stores the filter elements of the Enhanced RX FIFO.

For standard ID filtering, each ERFFEL register stores one filter element. For extended ID filtering, each pair of ERFFEL registers stores one filter element.

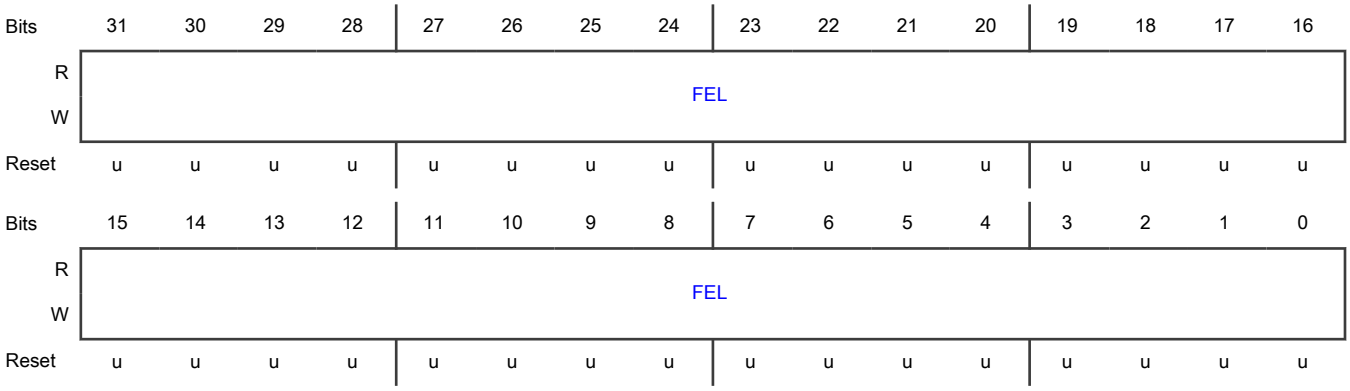
ERFFEL registers can be written only in Freeze mode; otherwise, the module blocks them. Reset does not affect these registers. They are located in RAM and must be explicitly initialized prior to any reception.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| FLEXCAN_0 | ERFFEL0–ERFFEL127 | — |
| FLEXCAN_1 | — | ERFFEL0–ERFFEL127 |
| FLEXCAN_2 | — | ERFFEL0–ERFFEL127 |
| FLEXCAN_3 | — | ERFFEL0–ERFFEL127 |
| FLEXCAN_4 | — | ERFFEL0–ERFFEL127 |
| FLEXCAN_5 | — | ERFFEL0–ERFFEL127 |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>FEL | Filter Element Bits<br>Stores filter elements. Each filter element is used during the match process. If the matching criteria are met, a message is stored in the Enhanced RX FIFO. |

## 70.6.3 Message buffer structure

The message buffer structure used by FlexCAN is represented in the following figure. Both extended (29-bit identifier) and standard (11-bit identifier) frames used in the CAN specification (Version 2.0 Part B) are represented. Each individual message buffer is 16, 24, 40, or 72 bytes, depending on the quantity of data bytes allocated for the message payload: 8, 16, 32, or 64 data bytes, respectively.

The memory area 80h–67Fh is used by the message buffers. When CAN FD is enabled, the exact address for each message buffer depends on the size of its payload. See FlexCAN memory partition for CAN FD.

**Table 451. Message buffer structure example with 64-byte payload**

| | 31 | 30 | 29 | 28 | 27 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 8 | 7 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0h | EDL | BRS | ESI | | CODE | | | SRR | IDE | RTR | DLC | | | | TIMESTAMP | | | |
| 4h | PRIO | | | | ID (standard/extended) | | | | | | | | | | ID (extended) | | | |
| 8h | Data byte 0 | | | | | | Data byte 1 | | | | | | | | Data byte 2 | | Data byte 3 | |
| Ch | Data byte 4 | | | | | | Data byte 5 | | | | | | | | Data byte 6 | | Data byte 7 | |
| 10h | Data byte 8 | | | | | | Data byte 9 | | | | | | | | Data byte 10 | | Data byte 11 | |
| 14h | Data byte 12 | | | | | | Data byte 13 | | | | | | | | Data byte 14 | | Data byte 15 | |
| 18h | Data byte 16 | | | | | | Data byte 17 | | | | | | | | Data byte 18 | | Data byte 19 | |
| 1Ch | Data byte 20 | | | | | | Data byte 21 | | | | | | | | Data byte 22 | | Data byte 23 | |
| 20h | Data byte 24 | | | | | | Data byte 25 | | | | | | | | Data byte 26 | | Data byte 27 | |
| 24h | Data byte 28 | | | | | | Data byte 29 | | | | | | | | Data byte 30 | | Data byte 31 | |
| 28h | Data byte 32 | | | | | | Data byte 33 | | | | | | | | Data byte 34 | | Data byte 35 | |
| 2Ch | Data byte 36 | | | | | | Data byte 37 | | | | | | | | Data byte 38 | | Data byte 39 | |
| 30h | Data byte 40 | | | | | | Data byte 41 | | | | | | | | Data byte 42 | | Data byte 43 | |
| 34h | Data byte 44 | | | | | | Data byte 45 | | | | | | | | Data byte 46 | | Data byte 47 | |
| 38h | Data byte 48 | | | | | | Data byte 49 | | | | | | | | Data byte 50 | | Data byte 51 | |
| 3Ch | Data byte 52 | | | | | | Data byte 53 | | | | | | | | Data byte 54 | | Data byte 55 | |
| 40h | Data byte 56 | | | | | | Data byte 57 | | | | | | | | Data byte 58 | | Data byte 59 | |
| 44h | Data byte 60 | | | | | | Data byte 61 | | | | | | | | Data byte 62 | | Data byte 63 | |

= Unimplemented or reserved

**Table 452. Field descriptions**

| Mnemonic | Field | Description |
|---|---|---|
| EDL | Extended Data Length | Distinguishes between CAN format and CAN FD format frames. EDL must not be 1 for message buffers configured to RANSWER with code field 1010b (see Table 453). |
| BRS | Bit Rate Switch | Defines whether the bit rate is switched inside a CAN FD format frame. |
| ESI | Error State Indicator | Indicates whether the transmitting node is error-active or error-passive. |
| CODE | Message Buffer Code | Can be accessed (read or write) by the CPU and by the FlexCAN module itself, as part of the message buffer matching and arbitration process. The encoding is shown in Table 453 and Table 454. See Functional description. |

Table 453. Message buffer code for RX buffers

| CODE description | RX code BEFORE RX new frame | SRV[1] | RX code AFTER successful reception[2] | RRS[3] | Comment |
|---|---|---|---|---|---|
| 0000b: INACTIVE. Message buffer is not active. | INACTIVE | — | — | — | Message buffer does not participate in the matching process. |
| 0100b: EMPTY. Message buffer is active and empty. | EMPTY | — | FULL | — | When a frame is received successfully (after Move-in), CODE is automatically updated to FULL. |
| 0010b: FULL. Message buffer is full. | FULL | Yes | FULL | — | The act of reading the Control and Status word followed by unlocking the message buffer (SRV) does not make CODE return to EMPTY. It remains FULL. If a new frame is moved to the message buffer after the message buffer is serviced, the code remains FULL. See Matching process for matching details related to FULL code. |
| | | No | OVERRUN | — | If the message buffer is FULL and a new frame is moved to this message buffer before the CPU services it, CODE is automatically updated to OVERRUN. See Matching process for details about overrun behavior. |
| 0110b: OVERRUN. Message buffer is being overwritten into a full buffer. | OVERRUN | Yes | FULL | — | If CODE indicates OVERRUN and the CPU has serviced the message buffer, when a new frame is moved to the message buffer, CODE returns to FULL. |
| | | No | OVERRUN | — | If CODE already indicates OVERRUN, and another new frame must be moved, the message buffer is overwritten |

*Table continues on the next page...*

Table 453. Message buffer code for RX buffers (continued)

| CODE description | RX code BEFORE RX new frame | SRV[1] | RX code AFTER successful reception[2] | RRS[3] | Comment |
|---|---|---|---|---|---|
| | | | | | again, and CODE remains OVERRUN. See Matching process for details about overrun behavior. |
| 1010b: RANSWER[4]. A frame was configured to recognize a Remote Request frame and transmit a Response frame in return.[5] | RANSWER | — | TANSWER (1110b) | 0 | A Remote Answer was configured to recognize a Remote Request frame received. After that, a message buffer is set to transmit a response frame. CODE is automatically changed to TANSWER (1110b). See Matching process for details. If CTRL2[RRS] = 0, transmit a response frame when a remote request frame with the same ID is received. |
| | | — | — | 1 | This code is ignored during matching and arbitration process. See Matching process for details. |
| CODE[0] = 1: BUSY. FlexCAN is updating the contents of the message buffer. The CPU must not access the message buffer. | BUSY[6] | — | FULL | — | Indicates that the message buffer is being updated. It automatically becomes 0 and does not interfere with the next CODE. |
| | | — | OVERRUN | — | |

1. SRV: Serviced message buffer. Message buffer was read and unlocked by reading TIMER or another message buffer.
2. A frame is considered a successful reception after the frame is moved to a message buffer (move-in process). See Move-in.
3. Remote Request Stored field. See Control 2 (CTRL2).
4. Code 1010b is not considered TX and a message buffer with this code should not be aborted.
5. Code 1010b must be used in message buffers configured in CAN FD format, with EDL = 1.
6. For TX message buffers, the BUSY bit should be ignored upon read, except when MCR[AEN] = 1. If this field is 1, the corresponding message buffer does not participate in the matching process.

Table 454. Message buffer code for TX buffers

| CODE Description | TX Code BEFORE TX frame | MB RTR | TX Code AFTER successful transmission | Comment |
|---|---|---|---|---|
| 1000b: INACTIVE. Message buffer is not active. | INACTIVE | — | — | Message buffer does not participate in arbitration process. |
| 1001b: ABORT. Message buffer is aborted. | ABORT | — | — | Message buffer does not participate in arbitration process. |
| 1100b: DATA. Message buffer is a TX data frame (MB RTR must be 0). | DATA | 0 | INACTIVE | Transmit data frame unconditionally once. After transmission, the message buffer automatically returns to the INACTIVE state. |
| 1100b: REMOTE. Message buffer is a Transmit Remote Request frame (MB RTR must be 1). | REMOTE | 1 | EMPTY | Transmit remote request frame unconditionally once. After transmission, the message buffer automatically becomes an RX Empty message buffer with the same ID. |
| 1110b: TANSWER. Message buffer is a Transmit Response frame from an incoming Remote Request frame. | TANSWER | — | RANSWER | This intermediate code is automatically written to the message buffer by the CHI as a result of a match to a Remote Request frame. The Remote Response frame is transmitted unconditionally once, then the code automatically returns to RANSWER (1010b). The CPU can also write this code with the same effect. The Remote Response frame can be a data frame or another remote request frame, depending on the value of RTR. See Matching process and Arbitration process for details. |

Table 455. RX and TX message buffer field descriptions

| Mnemonic | Field | Description |
|---|---|---|
| SRR | Substitute Remote Request | Fixed recessive bit, used only in extended format. Write 1 to SRR for transmission (TX Buffers). SRR is stored with the value received on the CAN |

*Table continues on the next page...*

Table 455. RX and TX message buffer field descriptions (continued)

| Mnemonic | Field | Description |
|---|---|---|
| | | bus for RX receiving buffers. It can be received as either recessive or dominant. If FlexCAN receives this bit as dominant, it is interpreted as an arbitration loss.<br>1: Recessive value is compulsory for transmission in extended format frames.<br>0: Dominant is not a valid value for transmission in extended format frames. |
| IDE | ID Extended Bit | Identifies whether the frame format is standard or extended.<br>1: Frame format is extended<br>0: Frame format is standard |
| RTR | Remote Transmission Request | Affects the behavior of remote frames and is part of the reception filter. See Table 453, Table 454, and CTRL2[RRS].<br>If FlexCAN transmits this field as 1 (recessive) and receives it as 0 (dominant), it is interpreted as an arbitration loss. If this field is transmitted as 0 (dominant) and it is received as 1 (recessive), FlexCAN treats it as a bit error. If the value received matches the value transmitted, it is considered a successful bit transmission.<br>1: If message buffer is TX, indicates that the current message buffer may have a Remote Request frame to be transmitted. If the message buffer is RX, incoming remote request frames may be stored.<br>0: Indicates that the current message buffer has a Data frame to be transmitted. In an RX message buffer, it may be considered in matching processes.<br>**NOTE**<br>When configuring CAN FD frames, this field must be 0. |
| DLC | Data Length Code | Indicates the length (in bytes) of the RX or TX data, which is located in offset 8h–Fh of the message buffer space (see Table 451).<br>In reception, this field is written by FlexCAN, copied from the DLC field of the received frame.<br>In transmission, this field is written by the CPU and corresponds to the DLC field value of the frame to be transmitted.<br>When RTR = 1, the frame to be transmitted is a remote frame and does not include the data field, regardless of the DLC field (see Table 457). |
| TIMESTAMP | Free-Running Counter Timestamp | Provides a copy of the Free-Running Timer, captured for TX and RX frames when the beginning of the Identifier field appears on the CAN bus.<br>See Table 456 for Timestamp operation. |
| PRIO | Local priority | Used only when MCR[LPRIOEN] = 1, and only makes sense for transmit message buffers. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See Arbitration process. |
| ID | Frame Identifier | In standard frame format, only the 11 most significant bits (28 to 18) are used for frame identification in both receive and transmit cases. The 18 least |

*Table continues on the next page...*

Table 455. RX and TX message buffer field descriptions (continued)

| Mnemonic | Field | Description |
|---|---|---|
| | | significant bits are ignored. In extended frame format, all bits are used for frame identification in both receive and transmit cases. |
| DATA BYTE 0–63 | Data Field | Up to 64 bytes can be used for a data frame, depending on the size of payload selected for the message buffers.<br>For RX frames, the data is stored as it is received from the CAN bus. DATA BYTE (*n*) is valid only if *n* is less than DLC, as shown in Table 457. |

Table 456. Timestamp operation

| TSTAMPCAP | MBTSBASE | TIMER_SOURCE | Captured timebase | Capture point |
|---|---|---|---|---|
| b00 | bxx | 0 | CAN_TIMER incremented by CAN bit clock | Second bit of identifier field |
| b00 | bxx | 1 | CAN_TIMER incremented by on-chip timer clock | Second bit of identifier field |
| bxx | b00 | 0 | CAN_TIMER incremented by CAN bit clock | Second bit of identifier field |
| bxx | b00 | 1 | CAN_TIMER incremented by on-chip timer clock | Second bit of identifier field |
| b01 | b01 | x | Lower 16 bits of high-resolution on-chip timer | Seventh bit of the end of frame field for transmission and sixth bit of the end of frame field for reception |
| b01 | b10 | x | Upper 16 bits of high-resolution on-chip timer | Seventh bit of the end of frame field for transmission and sixth bit of the end of frame field for reception |
| b10 | b01 | x | Lower 16 bits of high-resolution on-chip timer | Start of frame |
| b10 | b10 | x | Upper 16 bits of high-resolution on-chip timer | Start of frame |
| b11 | b01 | x | Lower 16 bits of high-resolution on-chip timer | Start of frame for classical CAN frame format and res bit for CAN FD frame format |
| b11 | b10 | x | Upper 16 bits of high-resolution on-chip timer | Start of frame for classical CAN frame format and res bit for CAN FD frame format |

Table 457. DATA BYTE validity

| DLC | Valid data bytes |
|---|---|
| 0 | None |

*Table continues on the next page...*

Table 457. DATA BYTE validity (continued)

| DLC | Valid data bytes |
|---|---|
| 1 | DATA BYTE 0 |
| 2 | DATA BYTE 0–1 |
| 3 | DATA BYTE 0–2 |
| 4 | DATA BYTE 0–3 |
| 5 | DATA BYTE 0–4 |
| 6 | DATA BYTE 0–5 |
| 7 | DATA BYTE 0–6 |
| 8 | DATA BYTE 0–7 |
| 9 | DATA BYTE 0–11 |
| 10 | DATA BYTE 0–15 |
| 11 | DATA BYTE 0–19 |
| 12 | DATA BYTE 0–23 |
| 13 | DATA BYTE 0–31 |
| 14 | DATA BYTE 0–47 |
| 15 | DATA BYTE 0–63 |

### 70.6.4 FlexCAN memory partition for CAN FD

When CAN FD is enabled, FlexCAN RAM can be partitioned into blocks of 512 bytes each. Each block can accommodate a number of message buffers depending on the configuration provided by FDCTRL[MBDSR*n*] as shown in Table 458.

Table 458. RAM partition

| RAM block | Number of MBs with 8 bytes (default range) | Size control field in FDCTRL | Number of MBs of different sizes, per block |
|---|---|---|---|
| 0 | 0 to 31 | MBDSR0 | MBDSR0 = 00, 32 MBs with 8-byte payload<br>MBDSR0 = 01, 21 MBs with 16-byte payload<br>MBDSR0 = 10, 12 MBs with 32-byte payload<br>MBDSR0 = 11, 7 MBs with 64-byte payload |
| 1 | 32 to 63 | MBDSR1 | MBDSR1 = 00, 32 MBs with 8-byte payload<br>MBDSR1 = 01, 21 MBs with 16-byte payload<br>MBDSR1 = 10, 12 MBs with 32-byte payload<br>MBDSR1 = 11, 7 MBs with 64-byte payload |
| 2 | 64 to 95 | MBDSR2 | MBDSR2 = 00, 32 MBs with 8-byte payload<br>MBDSR2 = 01, 21 MBs with 16-byte payload<br>MBDSR2 = 10, 12 MBs with 32-byte payload<br>MBDSR2 = 11, 7 MBs with 64-byte payload |

Payload sizes of 16, 32, or 64 bytes may be configured in some or all of RAM blocks. In those cases, the total number of MBs and their respective number order may differ from the default configuration of 8 bytes. Consider an example where:

- Block0 is configured to an 8-byte payload
- Block1 is configured to a 16-byte payload
- Block2 is configured to 32-byte payload

In this case, Table 459 indicates how the message buffers are arranged in RAM.

**Table 459. RAM partition example**

| RAM block | Payload size | Number of MBs in the RAM block | Message buffer range |
|---|---|---|---|
| 0 | FDCTRL[MBDSR0] = 00, 8-byte payload | 32 | 0 to 31 |
| 1 | FDCTRL[MBDSR1] = 01, 16-byte payload | 21 | 32 to 52 |
| 2 | FDCTRL[MBDSR2] = 10, 32-byte payload | 12 | 53 to 64 |

### 70.6.5 FlexCAN message buffer memory map

The FlexCAN memory buffers are allocated in memory according to the tables below.

**Table 460. 8-byte message buffers**

| Address offset (hex) | MBDSR = b00<br>8-byte payload |
|---|---|
| 0080 | MB0 |
| 0090 | MB1 |
| 00A0 | MB2 |
| 00B0 | MB3 |
| 00C0 | MB4 |
| 00D0 | MB5 |
| 00E0 | MB6 |
| 00F0 | MB7 |
| 0100 | MB8 |
| 0110 | MB9 |
| 0120 | MB10 |
| 0130 | MB11 |
| 0140 | MB12 |
| 0150 | MB13 |
| 0160 | MB14 |

*Table continues on the next page...*

Table 460. 8-byte message buffers (continued)

| Address offset (hex) | MBDSR = b00<br>8-byte payload |
|---|---|
| 0170 | MB15 |
| 0180 | MB16 |
| 0190 | MB17 |
| 01A0 | MB18 |
| 01B0 | MB19 |
| 01C0 | MB20 |
| 01D0 | MB21 |
| 01E0 | MB22 |
| 01F0 | MB23 |
| 0200 | MB24 |
| 0210 | MB25 |
| 0220 | MB26 |
| 0230 | MB27 |
| 0240 | MB28 |
| 0250 | MB29 |
| 0260 | MB30 |
| 0270 | MB31 |
| 0280 | MB32 |
| 0290 | MB33 |
| 02A0 | MB34 |
| 02B0 | MB35 |
| 02C0 | MB36 |
| 02D0 | MB37 |
| 02E0 | MB38 |
| 02F0 | MB39 |
| 0300 | MB40 |
| 0310 | MB41 |
| 0320 | MB42 |
| 0330 | MB43 |
| 0340 | MB44 |
| 0350 | MB45 |

*Table continues on the next page...*

Table 460. 8-byte message buffers (continued)

| Address offset (hex) | MBDSR = b00<br>8-byte payload |
|---|---|
| 0360 | MB46 |
| 0370 | MB47 |
| 0380 | MB48 |
| 0390 | MB49 |
| 03A0 | MB50 |
| 03B0 | MB51 |
| 03C0 | MB52 |
| 03D0 | MB53 |
| 03E0 | MB54 |
| 03F0 | MB55 |
| 0400 | MB56 |
| 0410 | MB57 |
| 0420 | MB58 |
| 0430 | MB59 |
| 0440 | MB60 |
| 0450 | MB61 |
| 0460 | MB62 |
| 0470 | MB63 |
| 0480 | MB64 |
| 0490 | MB65 |
| 04A0 | MB66 |
| 04B0 | MB67 |
| 04C0 | MB68 |
| 04D0 | MB69 |
| 04E0 | MB70 |
| 04F0 | MB71 |
| 0500 | MB72 |
| 0510 | MB73 |
| 0520 | MB74 |
| 0530 | MB75 |
| 0540 | MB76 |

*Table continues on the next page...*

Table 460. 8-byte message buffers (continued)

| Address offset (hex) | MBDSR = b00<br>8-byte payload |
|---|---|
| 0550 | MB77 |
| 0560 | MB78 |
| 0570 | MB79 |
| 0580 | MB80 |
| 0590 | MB81 |
| 05A0 | MB82 |
| 05B0 | MB83 |
| 05C0 | MB84 |
| 05D0 | MB85 |
| 05E0 | MB86 |
| 05F0 | MB87 |
| 0600 | MB88 |
| 0610 | MB89 |
| 0620 | MB90 |
| 0630 | MB91 |
| 0640 | MB92 |
| 0650 | MB93 |
| 0660 | MB94 |
| 0670 | MB95 |

Table 461. 16-byte message buffers

| Address offset (hex) | MBDSR = b01<br>16-byte payload |
|---|---|
| 0080 | MB0 |
| 0098 | MB1 |
| 00B0 | MB2 |
| 00C8 | MB3 |
| 00E0 | MB4 |
| 00F8 | MB5 |
| 0110 | MB6 |
| 0128 | MB7 |

*Table continues on the next page...*

Table 461. 16-byte message buffers (continued)

| Address offset (hex) | MBDSR = b01<br>16-byte payload |
|---|---|
| 0140 | MB8 |
| 0158 | MB9 |
| 0170 | MB10 |
| 0188 | MB11 |
| 01A0 | MB12 |
| 01B8 | MB13 |
| 01D0 | MB14 |
| 01E8 | MB15 |
| 0200 | MB16 |
| 0218 | MB17 |
| 0230 | MB18 |
| 0248 | MB19 |
| 0260 | MB20 |
| 0280 | MB21 |
| 0298 | MB22 |
| 02B0 | MB23 |
| 02C8 | MB24 |
| 02E0 | MB25 |
| 02F8 | MB26 |
| 0310 | MB27 |
| 0328 | MB28 |
| 0340 | MB29 |
| 0358 | MB30 |
| 0370 | MB31 |
| 0388 | MB32 |
| 03A0 | MB33 |
| 03B8 | MB34 |
| 03D0 | MB35 |
| 03E8 | MB36 |
| 0400 | MB37 |
| 0418 | MB38 |

*Table continues on the next page...*

Table 461. 16-byte message buffers (continued)

| Address offset (hex) | MBDSR = b01<br>16-byte payload |
|---|---|
| 0430 | MB39 |
| 0448 | MB40 |
| 0460 | MB41 |
| 0480 | MB42 |
| 0498 | MB43 |
| 04B0 | MB44 |
| 04C8 | MB45 |
| 04E0 | MB46 |
| 04F8 | MB47 |
| 0510 | MB48 |
| 0528 | MB49 |
| 0540 | MB50 |
| 0558 | MB51 |
| 0570 | MB52 |
| 0588 | MB53 |
| 05A0 | MB54 |
| 05B8 | MB55 |
| 05D0 | MB56 |
| 05E8 | MB57 |
| 0600 | MB58 |
| 0618 | MB59 |
| 0630 | MB60 |
| 0648 | MB61 |
| 0660 | MB62 |

Table 462. 32-byte message buffers

| Address offset (hex) | MBDSR = b10<br>32-byte payload |
|---|---|
| 0080 | MB0 |
| 00A8 | MB1 |
| 00D0 | MB2 |

*Table continues on the next page...*

Table 462. 32-byte message buffers (continued)

| Address offset (hex) | MBDSR = b10<br>32-byte payload |
|---|---|
| 00F8 | MB3 |
| 0120 | MB4 |
| 0148 | MB5 |
| 0170 | MB6 |
| 0198 | MB7 |
| 01C0 | MB8 |
| 01E8 | MB9 |
| 0210 | MB10 |
| 0238 | MB11 |
| 0280 | MB12 |
| 02A8 | MB13 |
| 02D0 | MB14 |
| 02F8 | MB15 |
| 0320 | MB16 |
| 0348 | MB17 |
| 0370 | MB18 |
| 0398 | MB19 |
| 03C0 | MB20 |
| 03E8 | MB21 |
| 0410 | MB22 |
| 0438 | MB23 |
| 0480 | MB24 |
| 04A8 | MB25 |
| 04D0 | MB26 |
| 04F8 | MB27 |
| 0520 | MB28 |
| 0548 | MB29 |
| 0570 | MB30 |
| 0598 | MB31 |
| 05C0 | MB32 |
| 05E8 | MB33 |

*Table continues on the next page...*

Table 462. 32-byte message buffers (continued)

| Address offset (hex) | MBDSR = b10<br>32-byte payload |
|---|---|
| 0610 | MB34 |
| 0638 | MB35 |

Table 463. 64-byte message buffers

| Address offset (hex) | MBDSR = b11<br>64-byte payload |
|---|---|
| 0080 | MB0 |
| 00C8 | MB1 |
| 0110 | MB2 |
| 0158 | MB3 |
| 01A0 | MB4 |
| 01E8 | MB5 |
| 0230 | MB6 |
| 0280 | MB7 |
| 02C8 | MB8 |
| 0310 | MB9 |
| 0358 | MB10 |
| 03A0 | MB11 |
| 03E8 | MB12 |
| 0430 | MB13 |
| 0480 | MB14 |
| 04C8 | MB15 |
| 0510 | MB16 |
| 0558 | MB17 |
| 05A0 | MB18 |
| 05E8 | MB19 |
| 0630 | MB20 |

### 70.6.6 legacy RX FIFO structure

When MCR[RFEN] = 1, the memory area 80h–DCh (which is normally occupied by MBs 0–5) is used by the reception Legacy FIFO engine.

The region 80h–8Ch contains the output of the Legacy RX FIFO, which the CPU must read as a message buffer. This output contains the oldest message that has been received but not yet read. The region 90h–DCh is reserved for internal use of the Legacy RX FIFO engine.

An additional memory area, which starts at E0h and may extend up to 2DCh (normally occupied by MBs 6–37) depending on the value of CTRL2[RFFN], contains the ID filter table (configurable from 8 to 128 table elements) that specifies filtering criteria for accepting frames into the Legacy RX FIFO.

Out of reset, the ID filter table flexible memory area defaults to E0h and extends only to FCh, which corresponds to MBs 6 to 7 for RFFN = 0, for backward compatibility with previous versions of FlexCAN.

The following shows the Legacy RX FIFO data structure.

**Table 464. Legacy RX FIFO structure**

| | 31 28 | 24 23 | 22 | 21 | 20 | 19 18 17 16 | 15 8 | 7 0 |
|---|---|---|---|---|---|---|---|---|
| 80h | IDHIT | | SRR | IDE | RTR | DLC | TIMESTAMP | |
| 84h | | ID standard | | | | ID extended | | |
| 88h | Data byte 0 | | Data byte 1 | | | | Data byte 2 | Data byte 3 |
| 8Ch | Data byte 4 | | Data byte 5 | | | | Data byte 6 | Data byte 7 |
| 90h–DCh | Reserved | | | | | | | |
| E0h | ID filter table element 0 | | | | | | | |
| E4h | ID filter table element 1 | | | | | | | |
| E8h–2D4h | ID filter table elements 2 to 125 | | | | | | | |
| 2D8h | ID filter table element 126 | | | | | | | |
| 2DCh | ID filter table element 127 | | | | | | | |
| | | | | | | | | |
| | = Unimplemented or reserved | | | | | | | |

Each ID filter table element occupies an entire 32-bit word. One, two, or four Identifier Acceptance Filters (IDAF) can compound each element, depending on MCR[IDAM]. The following tables show the IDAF indexing.

Table 465 shows the three different formats of the ID table elements. All elements of the table must have the same format. See Legacy RX FIFO for more information.

**Table 465. ID table structure**

| Format | 31 | 30 | 29 24 | 23 16 | 15 | 14 | 13 8 | 7 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| A | RTR | IDE | RXIDA (standard = 29–19, extended = 29–1) | | | | | | |
| | | | | | | | | | |
| B | RTR | IDE | RXIDB_0 (standard = 29–19, extended = 29–16) | | RTR | IDE | RXIDB_1 (standard = 13–3, extended = 13–0) | | |
| | | | | | | | | | |
| C | RXIDC_0 (std and ext = 31–24) | | RXIDC_1 (std and ext = 23–16) | | RXIDC_2 (std and ext = 15–8) | | RXIDC_3 (std and ext = 7–0) | | |
| | | | | | | | | | |
| | = Unimplemented or Reserved | | | | | | | | |

Table 466. Field descriptions

| Mnemonic | Field | Description |
|---|---|---|
| RTR | Remote Frame | Specifies whether remote frames are accepted into the Legacy FIFO if they match the target ID.<br>1: Remote frames can be accepted and data frames are rejected.<br>0: Remote frames are rejected and data frames can be accepted. |
| IDE | Extended Frame | Specifies whether extended or standard frames are accepted into the Legacy FIFO if they match the target ID.<br>1: Extended frames can be accepted and standard frames are rejected.<br>0: Extended frames are rejected and standard frames can be accepted. |
| RXIDA | RX Frame Identifier (Format A) | Specifies an ID to be used as acceptance criteria for the Legacy FIFO. In the standard frame format, only the 11 most significant bits (29 to 19) are used for frame identification. In the extended frame format, all bits are used. |
| RXIDB_0, RXIDB_1 | RX Frame Identifier (Format B) | Specifies an ID to be used as acceptance criteria for the Legacy FIFO. In the standard frame format, the 11 most significant bits (a full standard ID) (29 to 19 and 13 to 3) are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID. |
| RXIDC_0, RXIDC_1, RXIDC_2, RXIDC_3 | RX Frame Identifier (Format C) | Specifies an ID to be used as acceptance criteria for the Legacy FIFO. In both standard and extended frame formats, all 8 bits of the field are compared to the 8 most significant bits of the received ID. |
| IDHIT | Identifier Acceptance Filter Hit Indicator | Indicates which identifier acceptance filter the received message in the output of the Legacy RX FIFO hit. See Legacy RX FIFO for more information. |

## 70.6.7 Enhanced RX FIFO structure

When ERFCR[ERFEN] = 1, the Enhanced RX FIFO is enabled. The region 2000h–204Ch contains the output of the Enhanced RX FIFO, which the CPU must read as a message buffer. This output contains the oldest message that has been received but not yet read.

Table 467. Enhanced RX FIFO structure

| | 31 | 30 | 29 | 28 ... 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 ... 8 | 7 | 6 ... 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2000h | EDL | BRS | ESI | Reserved | | SRR | IDE | RTR | DLC | | | | TIMESTAMP LEGACY | | |
| 2004h | Reserved | | | ID (standard/extended) | | | | | | | | | ID (extended) | | |
| 2008h | Data byte 0 | | | | | Data byte 1 | | | | | | | Data byte 2 | | Data byte 3 |
| 200Ch | Data byte 4 | | | | | Data byte 5 | | | | | | | Data byte 6 | | Data byte 7 |
| 2010h | Data byte 8 | | | | | Data byte 9 | | | | | | | Data byte 10 | | Data byte 11 |
| 2014h | Data byte 12 | | | | | Data byte 13 | | | | | | | Data byte 14 | | Data byte 15 |
| 2018h | Data byte 16 | | | | | Data byte 17 | | | | | | | Data byte 18 | | Data byte 19 |
| 201Ch | Data byte 20 | | | | | Data byte 21 | | | | | | | Data byte 22 | | Data byte 23 |

*Table continues on the next page...*

Table 467. Enhanced RX FIFO structure (continued)

| | | | | |
|---|---|---|---|---|
| 2020h | Data byte 24 | Data byte 25 | Data byte 26 | Data byte 27 |
| 2024h | Data byte 28 | Data byte 29 | Data byte 30 | Data byte 31 |
| 2028h | Data byte 32 | Data byte 33 | Data byte 34 | Data byte 35 |
| 202Ch | Data byte 36 | Data byte 37 | Data byte 38 | Data byte 39 |
| 2030h | Data byte 40 | Data byte 41 | Data byte 42 | Data byte 43 |
| 2034h | Data byte 44 | Data byte 45 | Data byte 46 | Data byte 47 |
| 2038h | Data byte 48 | Data byte 49 | Data byte 50 | Data byte 51 |
| 203Ch | Data byte 52 | Data byte 53 | Data byte 54 | Data byte 55 |
| 2040h | Data byte 56 | Data byte 57 | Data byte 58 | Data byte 59 |
| 2044h | Data byte 60 | Data byte 61 | Data byte 62 | Data byte 63 |
| IH_OFF | Reserved | | | ID HIT |
| TS_OFF | HR TIMESTAMP | | | |
| 2050h | 19 Enhanced FIFO Elements (Reserved) | | | |
| ... | | | | |
| 263Ch | | | | |

**NOTE**

ID HIT offset and high-resolution timestamp offset change dynamically according to data length code (DLC) as shown in Table 468.

Table 468. ID HIT offset and high-resolution timestamp offset

| Data Length Code (DLC) | ID HIT offset (IH_OFF) | High-resolution timestamp offset (TS_OFF) |
|---|---|---|
| 0 | 2008h | 200Ch |
| 1–4 | 200Ch | 2010h |
| 5–8 | 2010h | 2014h |
| 9 | 2014h | 2018h |
| 10 | 2018h | 201Ch |
| 11 | 201Ch | 2020h |
| 12 | 2020h | 2024h |
| 13 | 2028h | 202Ch |
| 14 | 2038h | 203Ch |
| 15 | 2048h | 204Ch |

Table 469. Field descriptions

| Mnemonic | Field | Description |
|---|---|---|
| EDL | Extended Data Length | Distinguishes between classical CAN format and CAN FD format frames.<br>0: Classical CAN frame format<br>1: CAN FD frame format |
| BRS | Bit Rate Switch | Defines whether the bit rate is switched inside a CAN FD format frame.<br>0: Bit rate is not switched in a CAN FD frame.<br>1: Bit rate is switched in a CAN FD frame. |
| ESI | Error State Indicator | Indicates whether the transmitting node is error-active or error-passive. This field is meaningful only if EDL = 1.<br>0: Error-active<br>1: Error-passive |
| SRR | Substitute Remote Request | Fixed recessive bit, used only in extended format. Transmitting nodes always send it as recessive and receiving nodes can receive it as either recessive or dominant. If FlexCAN receives this bit as dominant, it is interpreted as an arbitration loss. |
| IDE | ID Extended Bit | Identifies whether the frame format is standard or extended.<br>0: Standard<br>1: Extended |
| RTR | Remote Frame | Identifies whether the current frame is a data frame or a remote request.<br>0: Data frame<br>1: Remote request |
| DLC | Data Length Code | Defines the number of bytes in the data field of a CAN frame (Data byte 0 to Data byte 63). When RTR = 1, the frame is a remote request and does not include the data field, regardless of the DLC field. See Table 457 for more details. |
| LEGACY TIMESTAMP | 16-bit Timestamp | Provides a copy of the Free-Running Timer, captured during the CAN frame. See Table 456 for details about legacy timestamp operation. |
| ID | Frame Identifier | In base frame format, only the 11 most significant bits are used for frame identification. The 18 least significant bits are ignored. In extended frame format, all bits are used for frame identification. |
| DATA BYTE 0–63 | Data Field | Up to 64 bytes can be stored in the data field. |
| IDHIT | Identifier Acceptance Filter Hit Indicator | Indicates which Enhanced RX FIFO Filter Element (ERFFEL0 - ERFFEL127) the received message in the output of the Enhanced RX FIFO hit. For each filter region, standard-ID filter space, and extended-ID filter space, there is an independent index starting from zero. Table 470 shows how FlexCAN writes IDHIT according to each filter element. |

*Table continues on the next page...*

Table 469. Field descriptions (continued)

| Mnemonic | Field | Description |
|---|---|---|
| HR TIMESTAMP | High-resolution Timestamp | 32-bit timebase captured during the CAN frame. When CTRL2[TSTAMPCAP] is not zero, a 32-bit timebase is captured from a dedicated on-chip timer which operates in free-running mode. See CTRL2[TSTAMPCAP] for details about capture point configuration of the high-resolution timestamp. |

Table 470. IDHIT for Enhanced RX FIFO

| Enhanced RX FIFO filter element - ERFFEL | IDHIT value | Filter element type |
|---|---|---|
| ERFFEL0 | 0 | Extended-ID |
| ERFFEL1 | 1 | Extended-ID |
| . . . | . . . | Extended-ID |
| ERFFEL(m-1) | m-1 | Extended-ID |
| ERFFEL(m) | 0 | Standard-ID |
| ERFFEL(m+1) | 1 | Standard-ID |
| . . . | . . . | Standard-ID |
| ERFFEL(2n-m+1) | 2x(n-m)+1 | Standard-ID |

**NOTE**

Where m = NEXIF and n = NFE. If NEXIF = 0, only standard-ID filter elements exist. If NEXIF > NFE, only extended-ID filter elements exist.

## 70.7 Glossary

**Active message buffer** — A message buffer is active if it can participate in the current matching or arbitration process.

**Bus interface unit** — FlexCAN submodule responsible for the interface to the CPU.

**Bus off** — See CAN Specification.[8]

**CAN** — Controller Area Network, a serial communication protocol defined in CAN Specification[8] and ISO International Standard.[9]

**CHI** — Controller-host interface, a FlexCAN submodule responsible for message buffer matching and arbitration algorithms.

**CRC** — Cyclic redundancy check

[8] Controller Area Network - CAN Specification Version 2.0 Part A, Part B, Robert Bosch GmbH, 1991.
[9] ISO International Standard - ISO 11898 First Edition 1993 Road Vehicles - Interchange of Digital Information - Controller Area Network (CAN) for high-speed Communication.

| | |
|---|---|
| **Dominant bit** | A dominant bit wins the arbitration on the CAN bus. It is transmitted as 0. |
| **Doze mode** | A system low-power mode where the CPU bus remains active and a global Doze mode request is sent to all peripherals asking them to enter low power mode. |
| **Error active** | See CAN Specification.[8] |
| **Error frame** | See CAN Specification.[8] |
| **Error passive** | See CAN Specification.[8] |
| **Form error** | See CAN Specification.[8] |
| **Hard reset** | A reset coming from an external pin and/or following power-on. It resets everything. |
| **Idle** | See CAN Specification.[8] |
| **Information processing time** | See CAN Specification.[8] |
| **Intermission** | See CAN Specification.[8] |
| **Matching elements** | Data used in the matching process, such as ID, filter, mask, etc. |
| **MB** | See Message buffer. |
| **Message buffer** | Internal FlexCAN data structure containing bytes received from, or to be transmitted to, the CAN line, as well as information about this data. |
| **Overload frame** | See CAN Specification.[8] |
| **Phase buffer segment** | See CAN Specification.[8] |
| **Recessive bit** | A recessive bit loses the arbitration on the CAN bus. It is transmitted as 1. |
| **Sclock** | Serial clock. Obtained by dividing the clock feeding the CAN engine (oscillator or bus clock) by a prescaler factor. The Sclock period defines the time quantum for CAN protocol timing. |
| **Soft reset** | Global reset typically used by peripherals to reinitialize some of its registers, but not all of them. |
| **Stop mode** | A system low-power mode in which all chip clocks are stopped for maximum power savings. |
| **Stuffing error** | See CAN Specification.[8] |
| **Time quantum** | This is equal to the Sclock period. It is the minimum time period used to compose the CAN protocol bit timing. |

# Chapter 71
# Synchronous Audio Interface (SAI)

## 71.1 Chip-specific SAI information

### 71.1.1 SAI instances and configuration

**Table 471. SAI instances**

| Instance | MCXE31B | MCXE315/ MCXE316/MCXE317 |
|---|---|---|
| SAI_0 | Yes | No |
| SAI_1 | Yes | No |

**Table 472. SAI configuration**

| Chip | Instances | Synchronous Mode | DMA support | Data Rate/ channel | No. of Channels | Frame Synchronization | Master Clock Frequency | FIFO size(TX/RX)/ Channel | No. of words/ frame |
|---|---|---|---|---|---|---|---|---|---|
| MCXE31B | SAI_0 | No | Yes | 12.288 Mbps | 4 | I2S, AC97, and CODEC/DSP interfaces | 24.576 external audio | 8X32 bit | 8-16 words |
| | SAI_1 | No | Yes | 12.288 Mbps | 1 | I2S, AC97, and CODEC/DSP interfaces | 24.576 external audio | 8x32bit | 8-16 words |

**NOTE**

The SAI_BCLK (SAI_TX_BCLK, SAI_RX_BCLK), SAI_SYNC (SAI_TX_SYNC, SAI_RX_SYNC), and SAI_DATA (SAI_TX_DATA, SAI_RX_DATA) pins are shared between each module instance's (SAI_0, SAI_1) receiver and transmitter. See the IOMUX file attached to this document for details. At any specific time, either the SAI receiver or the SAI transmitter should be active or they should operate synchronously (only one active at a time). The SAI transmitter and SAI receiver should not be configured to operate simultaneously with inconsistent configurations.

**NOTE**

Timing specification of SAI master and SAI slave may limit maximum frequency.

## 71.2 Overview

SAI provides an interface that supports full-duplex serial digital audio interfaces with frame synchronization formats such as $I^2S$, AC97, TDM, and Codec/DSP interfaces.

### 71.2.1 Block diagram

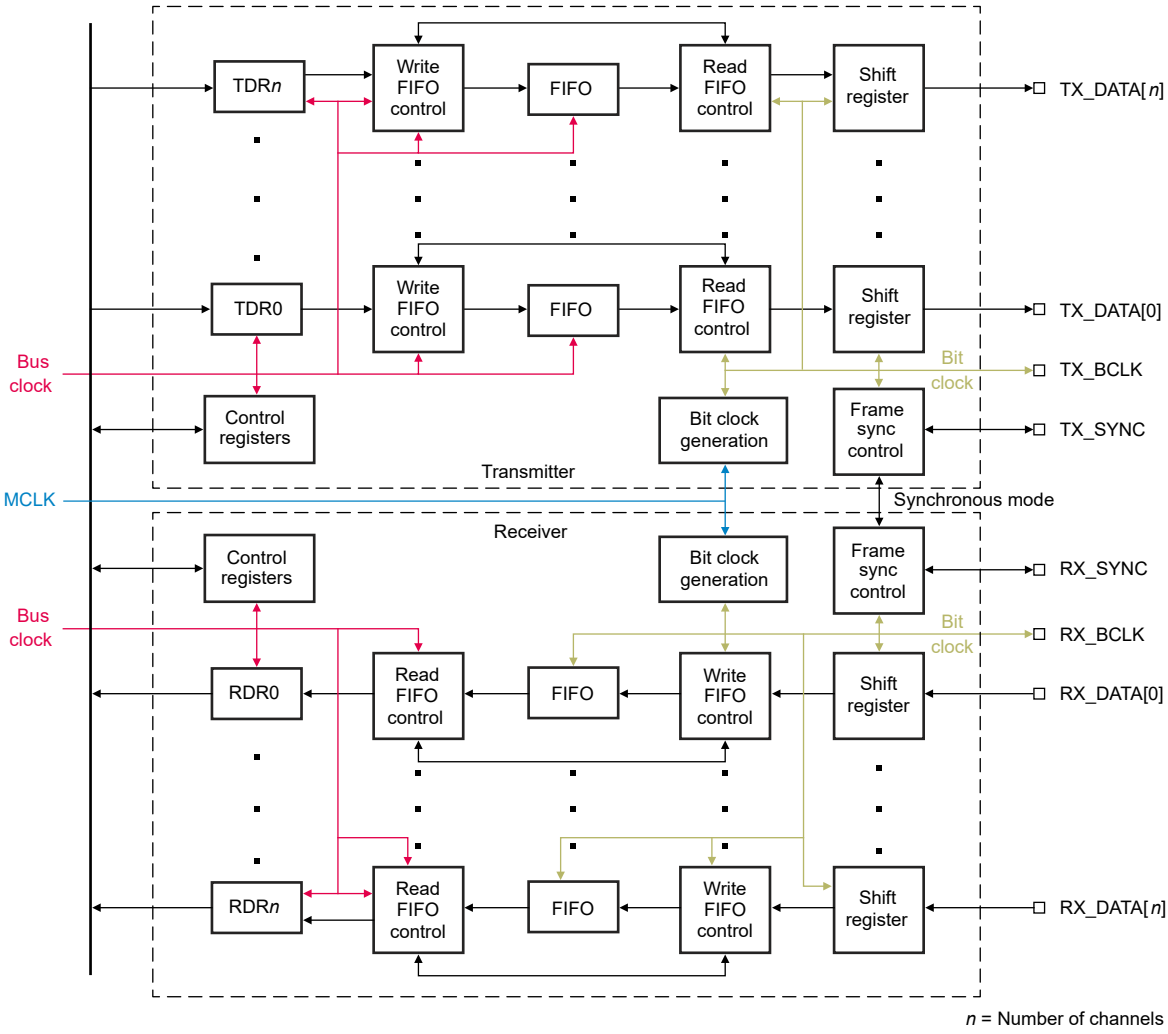



Figure 385. Block diagram

### 71.2.2 Features

Features can vary among chips and among SAI modules on the same chip. See the chip-specific SAI information at the beginning of this chapter.

- Transmitter with independent bit clock and frame synchronization supporting 4 data lines
- Receiver with independent bit clock and frame synchronization supporting 4 data lines
- Maximum frame size of 16 words per data line
- Word length of 8 to 32 bits
- Word length configured separately for the first word and remaining words in a frame
- Asynchronous 8 × 32-bit FIFO for each transmit and receive data line supports:
  - Graceful restart after FIFO error.
  - Automatic restart after FIFO error without software intervention.
  - 8-bit and 16-bit data packing into each 32-bit FIFO word.

— Multiple-data-line FIFOs combining into single-data-line FIFO.

## 71.3 Functional description

This section provides a complete functional description of SAI.

### 71.3.1 Modes of operation

- Run mode
- Debug mode

#### 71.3.1.1 Run mode

In Run mode, the SAI transmitter and receiver operate normally.

#### 71.3.1.2 Debug mode

You can configure either or both the SAI transmitter and receiver to continue operating in Debug mode:

- Write 1 to TCSR[DBGE] to configure the transmitter to run in Debug mode.
- Write 1 to RCSR[DBGE] to configure the receiver to run in Debug mode.

When TCSR[DBGE] and RCSR[DBGE] are 0 and the chip enters Debug mode, SAI is disabled after completing the current transmit or receive frame. Debug mode does not affect the transmitter and receiver bit clocks.

### 71.3.2 Synchronous modes

The SAI transmitter and receiver can operate synchronously to each other. You can configure the SAI transmitter and receiver to operate with a synchronous bit clock and frame sync (see TCR2[SYNC] and RCR2[SYNC]).

If both the transmitter and receiver use the transmitter bit clock and frame sync:

- Configure the transmitter for asynchronous operation and the receiver for synchronous operation.
- Enable the receiver in Synchronous mode only after configuring both the transmitter and receiver.
- Enable the transmitter last and disable the transmitter first.

If both the transmitter and receiver use the receiver bit clock and frame sync:

- Configure the receiver for asynchronous operation and the transmitter for synchronous operation.
- Enable the transmitter in Synchronous mode only after configuring both the receiver and transmitter.
- Enable the receiver last and disable the receiver first.

When operating in Synchronous mode, the transmitter and receiver share only the bit clock, frame sync, and transmitter and receiver enable. Otherwise, the transmitter and receiver operate independently, although the configuration register settings must be consistent across both the transmitter and receiver.

### 71.3.3 Frame sync configuration

When enabled, SAI continuously transmits and receives frames of data. Each frame consists of a fixed number of words, with each word consisting of a fixed number of bits. Within each frame, you can mask any word, causing the receiver to ignore that word and the transmitter to 3-state during that word.

The frame sync signal indicates the start of a frame. A valid frame sync requires the detection of a rising edge (if active high) or falling edge (if active low). The transmitter or receiver cannot be busy with a previous frame. SAI ignores a valid frame sync (in Target mode) or does not generate one (in Controller mode) for the first four bit-clock cycles after enabling the transmitter or receiver.

You can configure the transmitter and receiver frame sync independently by using any of the following options:

- Generate externally or internally
- Configure to be active high or active low
- Assert with the first bit in the frame or one bit early
- Assert for a duration between one bit-clock cycle and the first word length
- Configure the frame length from 1 to 16 words
- Configure the word length from 8 to 32 bits, with separate configurations for the length of the first word and that of the remaining words
- Transmit or receive words MSB first or LSB first

You cannot change these configuration options after enabling the SAI transmitter or receiver.

## 71.3.4 Data FIFO

Each transmit and receive channel includes a 8 × 32-bit FIFO. Use Transmit Data (TDR0 - TDR3) and Receive Data (RDR0 - RDR3) to access FIFO data.

### 71.3.4.1 Data alignment

Data in the FIFO can be aligned anywhere within the 32-bit-wide register via the First Bit Shifted (FBT) configuration field. This field selects the bit index (between 31 and 0) of the first bit shifted.

Examples of supported data alignment and the required FBT configuration are shown in Figure 386 for LSB-first configurations and Figure 387 for MSB-first configurations.

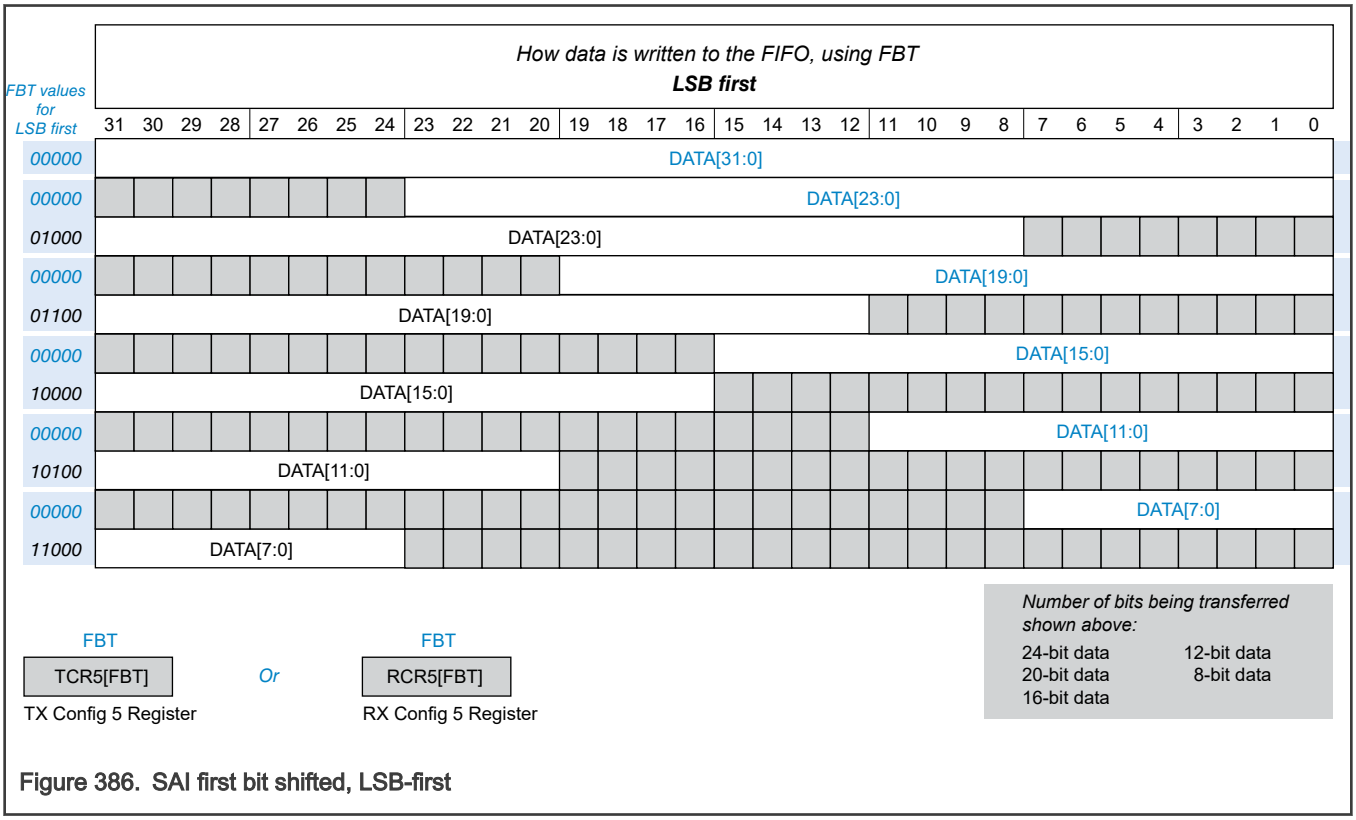



Figure 386. SAI first bit shifted, LSB-first

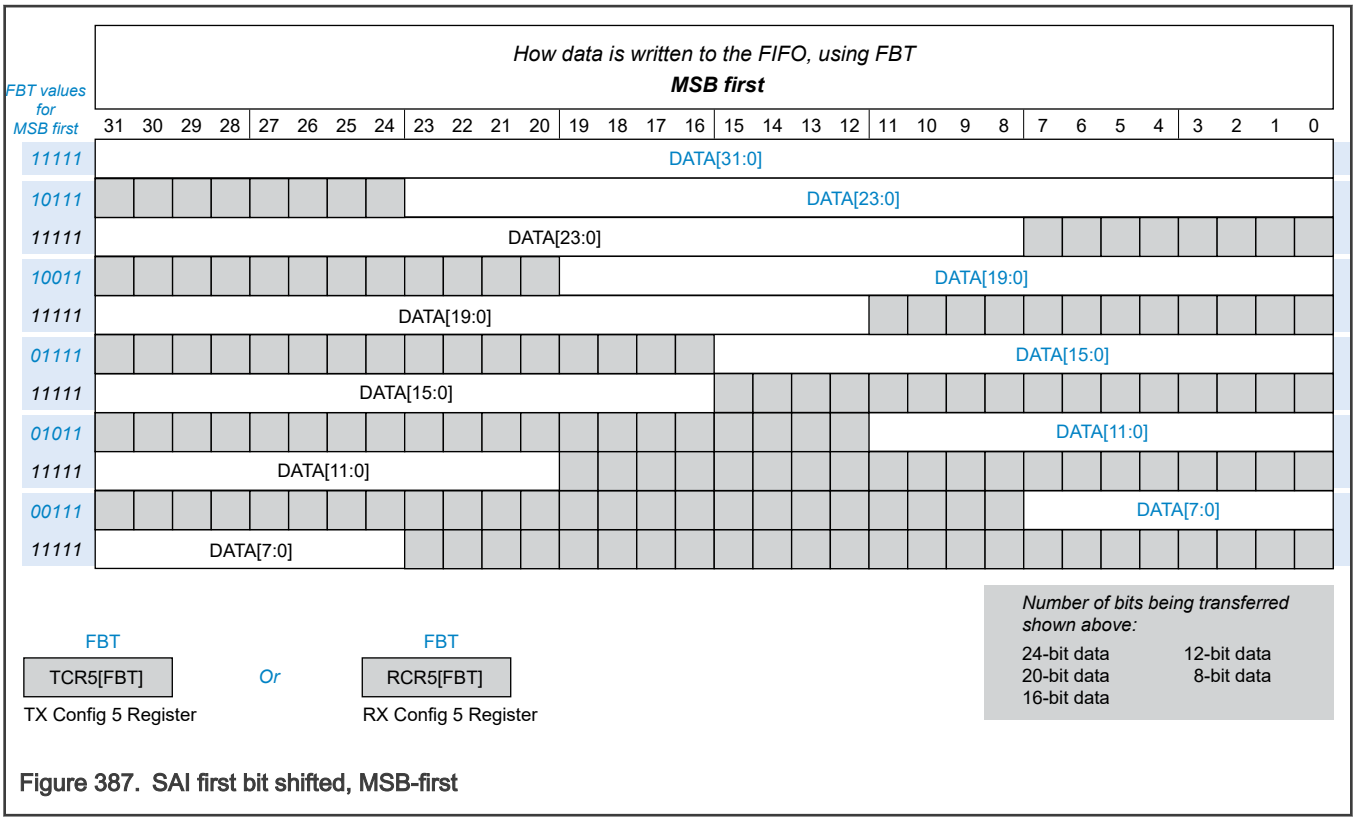



**Figure 387. SAI first bit shifted, MSB-first**

### 71.3.4.2 FIFO pointers

When writing to one of the Transmit Data (TDR0 - TDR3) registers, the write FIFO pointer of the corresponding Transmit FIFO register (TFR*n*[WFP]) increments after each valid write. SAI supports 8-bit, 16-bit, and 32-bit writes to TDR*n*, and the FIFO pointer increments after each individual write. Use only 8-bit writes when transmitting up to 8-bit data, and use only 16-bit writes when transmitting up to 16-bit data:

- If the transmit FIFO is full, SAI ignores writes to TDR*n*.
- If the transmit FIFO is empty, write to TDR*n* at least three bit-clock cycles before the start of the next unmasked word. This write avoids a FIFO underrun.
- Before enabling the transmitter, initialize the transmit FIFO with data. The transmitter starts a new frame after enabling the transmitter. If no data is in the FIFO, the transmitter immediately generates an error.

When reading one of the Receive Data (RDR0 - RDR3) registers, the read FIFO pointer of the corresponding Receive FIFO register (RFR*n*[RFP]) increments after each valid read. SAI supports 8-bit, 16-bit, and 32-bit reads from RDR*n*, and the FIFO pointer increments after each individual read. Use only 8-bit reads when receiving up to 8-bit data, and use only 16-bit reads when receiving up to 16-bit data:

- If the receive FIFO is empty, SAI ignores reads to RDR*n*.
- If the receive FIFO is full, read RDR*n* at least three bit-clock cycles before the end of an unmasked word. This read avoids a FIFO overrun.

### 71.3.4.3 FIFO packing

Using FIFO packing, you can store multiple 8-bit or 16-bit data words in one 32-bit FIFO word for the transmitter or receiver. You could emulate this feature by adjusting the number of bits per word and number of words per frame. FIFO packing, however, does not require even multiples of words per frame, and it fully supports word masking.

When FIFO packing is enabled, the FIFO pointers only increment when the full 32-bit FIFO word has been written (transmit) or read (receive). In this way, FIFO packing supports scenarios where different words within each frame are stored in different areas of memory.

Using 16-bit FIFO packing for transmitting, the transmit shift register loads at the start of each frame and after every second unmasked transmit word. The first transmitted word is taken from the 16-bit word at byte offset 0h. (The first bit is selected by TCR5[FBT], and you must configure it within this 16-bit word.) The second transmitted word is taken from the 16-bit word at byte offset 2h. (The first bit is selected by TCR5[FBT][3:0].) After the 16-bit word has been transmitted, the transmitter transmits logic zeroes until the start of the next word.

Using 16-bit FIFO packing for receiving, the receive shift register is stored after every second unmasked received word. If there is an odd number of unmasked-received words in each frame, the receive register is also stored, at the end of each frame. The first received word is stored in the 16-bit word at byte offset 0h. (The first bit is selected by RCR5[FBT], and you must configure it within this 16-bit word.) The second received word is stored in the 16-bit word at byte offset 2h. (The first bit is selected by RCR5[FBT][3:0].) The receiver ignores received data until the start of the next word after the 16-bit word has been received.

8-bit FIFO packing is similar to 16-bit packing, except four words are loaded or stored into each 32-bit FIFO word. The first word is stored in byte offset 0h, the second word in byte offset 1h, and so on. You must configure TCR5[FBT], RCR5[FBT], or both within byte offset 0h.

#### 71.3.4.4 FIFO Combining mode

This mode enables separate FIFOs for multiple data channels to be used as a single FIFO for software accesses, as a single data channel, or both. The enabled data channels must be contiguous and data channel 0 must be enabled when using FIFO Combining mode.

You can combine FIFOs for software access by writing to transmit FIFO registers and reading from receive FIFO registers. Doing so enables a DMA controller or software to read or write multiple FIFOs without incrementing the address that is accessed. After FIFO Combining mode is enabled (TCR4[FCOMB] > 00b), the first software access to a FIFO register accesses the first enabled channel FIFO. The second access to a FIFO register accesses the second enabled channel FIFO. This process continues until software accesses the last enabled channel FIFO and the pointer resets to the first enabled channel FIFO. To reset the pointer manually, you can reset the FIFOs or disable FIFO combining on software accesses.

Combining FIFOs for transmit data channels enables one data channel to use the FIFOs of all enabled channel FIFOs. In this case, the data output is identical on each enabled data channel. The transmit shift registers for all enabled data channels load at the start of each frame. The registers also load upon every *n*th unmasked word, where *n* is the number of enabled data channels. The first word transmitted loads from the first enabled channel FIFO; the second word transmitted loads from the second enabled channel FIFO. This loading continues until the end of the frame.

Combining FIFOs for receive data channels enables one data channel to use the FIFOs of all enabled channel FIFOs. In this case, the received data from channel 0 is stored in each enabled data channel. The receive shift registers for all enabled data channels are stored after every *n*th unmasked word, where *n* is the number of enabled data channels. The first word received is stored in the first enabled channel FIFO; the second word received is stored in the second enabled channel FIFO. This storage continues until the end of the frame.

**NOTE**

The first word in each frame is always stored in the first enabled data channel. NXP recommends that the number of unmasked words in each frame be evenly divisible by the number of enabled data channels.

Combining FIFOs for data channels loads or stores each channel FIFO at the same time. As a result, FIFO error conditions are only checked every *n*th word, where *n* is the number of enabled data channels. If any enabled data channel meets the warning flag or request flag conditions, SAI asserts the FIFO warning and request flags.

### 71.3.5 Word mask register

The SAI transmitter and receiver each have a word mask register (Transmit Mask (TMR) and Receive Mask (RMR)) that you can use to mask any word in the frame. The word mask register is double buffered. You can update it before the end of each frame to mask a particular word in the next frame.

TMR causes the transmit data pin to be 3-stated for the length of each selected word; the transmit FIFO is not read for masked words.

RMR causes the received data for each selected word to be discarded and not written to the receive FIFO.

## 71.3.6 Clocking

- Audio clock
- Bit clock
- Bus clock

### 71.3.6.1 Audio clock

The audio clock (MCLK) generates the bit clock when you configure the receiver or transmitter for an internally generated bit clock. The transmitter and receiver can independently select between the bus clock and up to three audio clocks to generate the bit clock.

The audio clock generation and selection is chip-specific. See chip-specific clocking information about how the audio clocks are generated.

### 71.3.6.2 Bit clock

The SAI transmitter and receiver support asynchronous free-running bit clocks that an audio clock can generate internally or an external source can supply. SAI can also have synchronous bit clock and frame sync operation between the receiver and transmitter, or between multiple SAI peripherals. The transmitter and receiver configuration affects the bit clock and frame sync in the following ways:

- If you configure both the transmitter and receiver for asynchronous operation, each uses its own bit clock and frame sync.
- If you configure the transmitter for asynchronous operation and the receiver for synchronous operation, both use the transmitter bit clock and frame sync.
- If you configure the receiver for asynchronous operation and the transmitter for synchronous operation, both use the receiver bit clock and frame sync.

Your choice of synchronous or asynchronous operation selects the bit clock and frame sync used.

Externally generated bit clocks must be:

- Enabled before the SAI transmitter or receiver is enabled.
- Disabled after the SAI transmitter or receiver is disabled and completes its current frames.

In asynchronous operation, a SAI transmitter or receiver can use a bit clock externally generated by a SAI module instance that is disabled in Stop mode. In this case, disable the transmitter or receiver before entering Stop mode. This issue does not apply when the transmitter or receiver is in synchronous operation because all synchronous SAI modules are enabled and disabled simultaneously.

### 71.3.6.3 Bus clock

The control and configuration registers use the bus clock to generate synchronous interrupts and DMA requests.

**NOTE**

Although no minimum bus clock frequency is specified, the frequency must be fast enough (relative to the bit clock frequency) to serve the FIFOs. You must meet this requirement without generating a transmitter FIFO underrun or receiver FIFO overflow condition.

## 71.3.7 Reset

SAI is asynchronously reset on system reset. SAI has a Software reset and a FIFO reset.

#### 71.3.7.1 Software reset

The SAI transmitter includes a software reset that resets all transmitter internal logic, including the bit clock generation, status flags, and FIFO pointers. The SAI receiver includes a software reset that resets all receiver internal logic, including bit clock generation, status flags, and FIFO pointers.

These software resets do not reset the configuration registers. The software resets remain asserted until you clear them via software.

#### 71.3.7.2 FIFO reset

The SAI transmitter includes a FIFO reset that synchronizes the FIFO write pointer to the value of the FIFO read pointer. This FIFO reset empties the FIFO contents. Use this reset after TCSR[FEF] becomes 1, and before SAI reinitializes the FIFO and TCSR[FEF] becomes 0. SAI asserts the FIFO reset for one cycle only.

The SAI transmitter can also reset the FIFO of individual data channels by writing 1 to the appropriate bit in TCR3[CFR]. Use this reset only when the corresponding bit in TCR3[TCE] is 0.

The SAI receiver includes a FIFO reset that synchronizes the FIFO read pointer to the value of the FIFO write pointer. This FIFO reset empties the FIFO contents. Use this reset after RCSR[FEF] becomes 1 and SAI reads any remaining data from the FIFO, and before RCSR[FEF] becomes 0. SAI asserts the FIFO reset for one cycle only.

The SAI receiver can also reset the FIFO of individual data channels by writing 1 to the appropriate bit in RCR3[RCE]. Use this reset only when the corresponding bit in RCR3[RCE] is 0.

### 71.3.8 Interrupts and DMA requests

In SAI, the transmitter and receiver generate separate interrupts and DMA requests, but use the same status flags. SAI only generates asynchronous interrupts when the system clock is gated but the corresponding BCLK signal is active.

#### 71.3.8.1 FIFO request flag

SAI sets the transmit FIFO request flag (TCSR[FRF]) when the number of entries in any enabled transmit FIFO is less than or equal to the transmit FIFO watermark configuration (TCR1[TFW]). SAI clears this flag when the number of entries in each enabled transmit FIFO is greater than TCR1[TFW].

SAI sets the receive FIFO request flag (RCSR[FRF]) when the number of entries in any enabled receive FIFO is greater than the receive FIFO watermark configuration (RCR1[RFW]). SAI clears this flag when the number of entries in each enabled receive FIFO is less than or equal to RCR1[RFW].

The FIFO request flag can generate an interrupt when TCSR[FRIE] = 1. It can generate a DMA request when TCSR[FRDE] = 1.

#### 71.3.8.2 FIFO warning flag

SAI sets the transmit FIFO warning flag (TCSR[FWF]) when the number of entries in any of the enabled transmit FIFOs is empty. SAI clears this flag when the number of entries in each enabled transmit FIFO is not empty.

SAI sets the receive FIFO warning flag (RCSR[FWF]) when the number of entries in any of the enabled receive FIFOs is full. SAI clears this flag when the number of entries in each enabled receive FIFO is not full.

The FIFO warning flag can generate an interrupt when TCSR[FWIE] = 1. The flag can generate a DMA request when TCSR[FWDE] = 1.

#### 71.3.8.3 FIFO error flag

SAI sets the transmit FIFO error flag (TCSR[FEF]) when any enabled transmit FIFOs underrun. After the error flag is set, all enabled transmit channels transmit zero data before SAI clears TCSR[FEF].

When you write 1 to TCR4[FCONT], the FIFO continues transmitting data following an underrun without software intervention. To ensure that data transmits in the correct order, the transmitter continues from the same word number in the frame that caused the FIFO to underrun. It continues only after new data is written to the transmit FIFO. In this case, clear TCSR[FEF] without reinitializing the transmit FIFOs.

SAI sets the receive FIFO error flag (RCSR[FEF]) when any enabled receive FIFO overflows. After the flag is set, all enabled receive channels discard their received data until SAI clears RCSR[FEF] and the next receive frame starts. Empty all enabled receive FIFOs before SAI clears RCSR[FEF].

When you write 1 to RCR4[FCONT], the FIFO continues receiving data following an overflow without software intervention. To ensure that data is received in the correct order, the receiver continues from the same word number in the frame that caused the FIFO to overflow. It continues only after data has been read from the receive FIFO. In this case, clear RCSR[FEF] without emptying the receive FIFOs.

TCSR[FEF] and RSCR[FEF] can only generate an interrupt.

#### 71.3.8.4 Sync error flag

SAI sets a sync error flag (TCSR[SEF] or RCSR[SEF]) when both of these conditions are true:

- The frame sync is generated externally.
- The external frame sync asserts when the transmitter or receiver is busy with the previous frame.

SAI ignores the external frame sync assertion and sets the sync error flag. When this flag is set, the transmitter or receiver continues checking for frame sync assertion when idle or at the end of each frame.

TCSR[SEF] and RCSR[SEF] can only generate an interrupt.

#### 71.3.8.5 Word start flag

SAI sets the word start flag (TCSR[WSF]) at the start of the second bit-clock cycle for the selected word. You can select this word via TCR3[WDFL].

TCSR[WSF] can only generate an interrupt.

## 71.4 External signals

| Name | Function | I/O |
|---|---|---|
| TX_BCLK | Transmit bit clock: the bit clock is an input when externally generated and an output when internally generated. | I/O |
| TX_SYNC | Transmit frame sync: the frame sync is an input sampled synchronously by the bit clock when externally generated. It is an output generated synchronously by the bit clock when internally generated. | I/O |
| TX_DATA[3:0] | Transmit data: the bit clock synchronously generates the transmit data; this signal is 3-stated when not transmitting a word. | O |
| RX_BCLK | Receive bit clock: the bit clock is an input when externally generated and an output when internally generated. | I/O |
| RX_SYNC | Receive frame sync: the frame sync is an input sampled synchronously by the bit clock when externally generated. It is an output generated synchronously by the bit clock when internally generated. | I/O |

*Table continues on the next page...*

*Table continued from the previous page...*

| Name | Function | I/O |
|---|---|---|
| RX_DATA[3:0] | Receive data: the bit clock synchronously samples the receive data. | I |

## 71.5 Initialization

To initialize SAI:

1. Enable the clock to SAI.
2. Reset the internal transmitter logic and receiver logic by writing 1 to TCSR[SR] and RCSR[SR], respectively.

## 71.6 Memory map and register definition

A read or write access to an address from offset 100h and above results in a bus error.

### 71.6.1 SAI register descriptions

SAI provides an interface that supports full-duplex serial digital audio interfaces with frame synchronization formats.

#### 71.6.1.1 SAI memory map

SAI_0 base address: 4036_C000h

SAI_1 base address: 404D_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID (VERID) | 32 | R | 0301_0000h |
| 4h | Parameter (PARAM) | 32 | R | See section |
| 8h | Transmit Control (TCSR) | 32 | RW | 0000_0000h |
| Ch | Transmit Configuration 1 (TCR1) | 32 | RW | 0000_0000h |
| 10h | Transmit Configuration 2 (TCR2) | 32 | RW | 0000_0000h |
| 14h | Transmit Configuration 3 (TCR3) | 32 | RW | 0000_0000h |
| 18h | Transmit Configuration 4 (TCR4) | 32 | RW | 0000_0000h |
| 1Ch | Transmit Configuration 5 (TCR5) | 32 | RW | 0000_0000h |
| 20h - 2Ch | Transmit Data (TDR0 - TDR3) | 32 | RW | See section |
| 40h - 4Ch | Transmit FIFO (TFR0 - TFR3) | 32 | R | See section |
| 60h | Transmit Mask (TMR) | 32 | RW | 0000_0000h |
| 88h | Receive Control (RCSR) | 32 | RW | 0000_0000h |
| 8Ch | Receive Configuration 1 (RCR1) | 32 | RW | 0000_0000h |
| 90h | Receive Configuration 2 (RCR2) | 32 | RW | 0000_0000h |
| 94h | Receive Configuration 3 (RCR3) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 98h | Receive Configuration 4 (RCR4) | 32 | RW | 0000_0000h |
| 9Ch | Receive Configuration 5 (RCR5) | 32 | RW | 0000_0000h |
| A0h - ACh | Receive Data (RDR0 - RDR3) | 32 | R | See section |
| C0h - CCh | Receive FIFO (RFR0 - RFR3) | 32 | R | See section |
| E0h | Receive Mask (RMR) | 32 | RW | 0000_0000h |

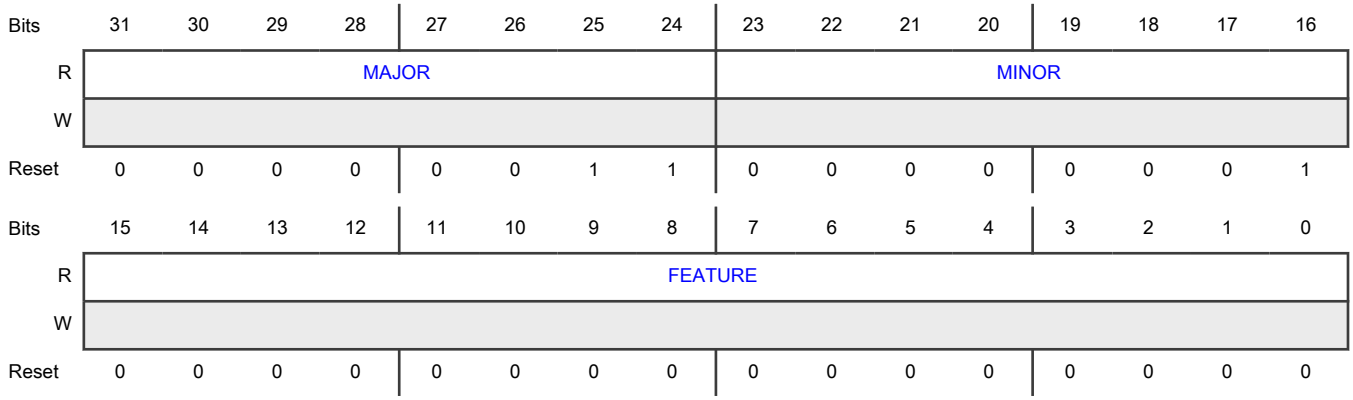
### 71.6.1.2 Version ID (VERID)

#### Offset

| Register | Offset |
|---|---|
| VERID | 0h |

#### Function
Contains version numbers for the module design and feature set.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>Indicates the major version number for the specification. |
| 23-16<br>MINOR | Minor Version Number<br>Indicates the minor version number for the specification. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 15-0<br>FEATURE | Feature Specification Number<br>Indicates the feature set number.<br>0000_0000_0000_0000b - Standard feature set |

### 71.6.1.3 Parameter (PARAM)

#### Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

#### Function

Contains parameter values implemented in the module.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | | | | | FRAME | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | See Register reset values. | | | | | | | | | | | | | | | |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | FIFO | | | | 0 | | | | DATALINE | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | See Register reset values. | | | | | | | | | | | | | | | |

#### Register reset values

| Register | Reset value |
|---|---|
| PARAM | SAI_0: 0004_0304h<br>SAI_1: 0004_0301h |

#### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 19-16<br>FRAME | Frame Size<br>Describes the maximum number of slots per frame, which is 2^FRAME. |
| 15-12<br>— | Reserved |
| 11-8<br>FIFO | FIFO Size<br>Describes the number of words in each FIFO, which is 2^FIFO. |
| 7-4<br>— | Reserved |
| 3-0<br>DATALINE | Number of Data Lines<br>Contains the number of data lines implemented. |

### 71.6.1.4 Transmit Control (TCSR)

#### Offset

| Register | Offset |
|---|---|
| TCSR | 8h |

#### Function

Contains transmitter enable fields including resets, error and interrupt enable fields, and error flag fields.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TE | Reserved | DBGE | BCE | 0 | | 0 | SR | 0 | | | WSF | SEF | FEF | FWF | FRF |
| W | | | | | | | FR | | | | | W1C | W1C | W1C | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | WSIE | SEIE | FEIE | FWIE | FRIE | 0 | | | 0 | | | FWDE | FRDE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>TE | Transmitter Enable<br>Enables the transmitter. When you write 0 to this field, the transmitter remains enabled, and this field remains 1 until the end of the current frame.<br>0b - Disable<br>1b - Enable (or transmitter has been disabled and has not yet reached the end of the frame) |
| 30<br>— | Reserved<br>Only write zero to this field. |
| 29<br>DBGE | Debug Enable<br>Enables transmitter operation in Debug mode, which does not affect the transmit bit clock. If you write 0 to this field, the transmitter operation is disabled after completing the current frame.<br>0b - Disable<br>1b - Enable |
| 28<br>BCE | Bit Clock Enable<br>Enables the transmit bit clock, separately from TCSR[TE]. This field automatically becomes 1 when TCSR[TE] becomes 1. When you write 0 to this field, the transmit bit clock remains enabled, and the field remains 1 until the end of the current frame.<br>0b - Disable<br>1b - Enable |
| 27-26<br>— | Reserved |
| 25<br>FR | FIFO Reset<br>Empties the FIFO and sets the FIFO read and write pointers to the same value, which may or may not be zero.<br>**NOTE**<br>The FIFO reset is asserted for one cycle only.<br>Reading this field always returns zero. SAI resets the FIFO pointers when the transmitter is disabled or the FIFO error flag is set.<br>0b - No effect<br>1b - FIFO reset |
| 24<br>SR | Software Reset<br>Resets the internal transmitter logic, including the FIFO read and write pointers.<br>Software-visible registers are not affected, except for the status registers. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>The software reset remains asserted until software writes 0 to the field.<br>0b - No effect<br>1b - Software reset |
| 23-21<br>— | Reserved |
| 20<br>WSF | Word Start Flag<br>Indicates whether the start of the configured word has been detected.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not detected<br>1b - Detected<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 19<br>SEF | Sync Error Flag<br>Indicates whether an error in the externally generated frame sync has been detected.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not detected<br>1b - Detected<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 18<br>FEF | FIFO Error Flag<br>Indicates whether an enabled transmit FIFO has underrun.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Not detected<br>1b - Detected<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 17<br>FWF | FIFO Warning Flag<br>Indicates whether an enabled transmit FIFO is empty.<br>0b - Not empty<br>1b - Empty |
| 16<br>FRF | FIFO Request Flag<br>Indicates whether the number of words in an enabled transmit channel FIFO is less than or equal to the transmit FIFO watermark.<br>0b - Watermark not reached<br>1b - Watermark reached |
| 15-13<br>— | Reserved |
| 12<br>WSIE | Word Start Interrupt Enable<br>Enables word start interrupts.<br>0b - Disable<br>1b - Enable |
| 11<br>SEIE | Sync Error Interrupt Enable<br>Enables sync error interrupts.<br>0b - Disable<br>1b - Enable |
| 10<br>FEIE | FIFO Error Interrupt Enable<br>Enables FIFO error interrupts.<br>0b - Disable<br>1b - Enable |
| 9<br>FWIE | FIFO Warning Interrupt Enable<br>Enables FIFO warning interrupts.<br>0b - Disable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Enable |
| 8<br>FRIE | FIFO Request Interrupt Enable<br>Enables FIFO request interrupts.<br>0b - Disable<br>1b - Enable |
| 7-5<br>— | Reserved |
| 4-2<br>— | Reserved |
| 1<br>FWDE | FIFO Warning DMA Enable<br>Enables the DMA warning.<br>0b - Disable<br>1b - Enable |
| 0<br>FRDE | FIFO Request DMA Enable<br>Enables DMA requests.<br>0b - Disable<br>1b - Enable |

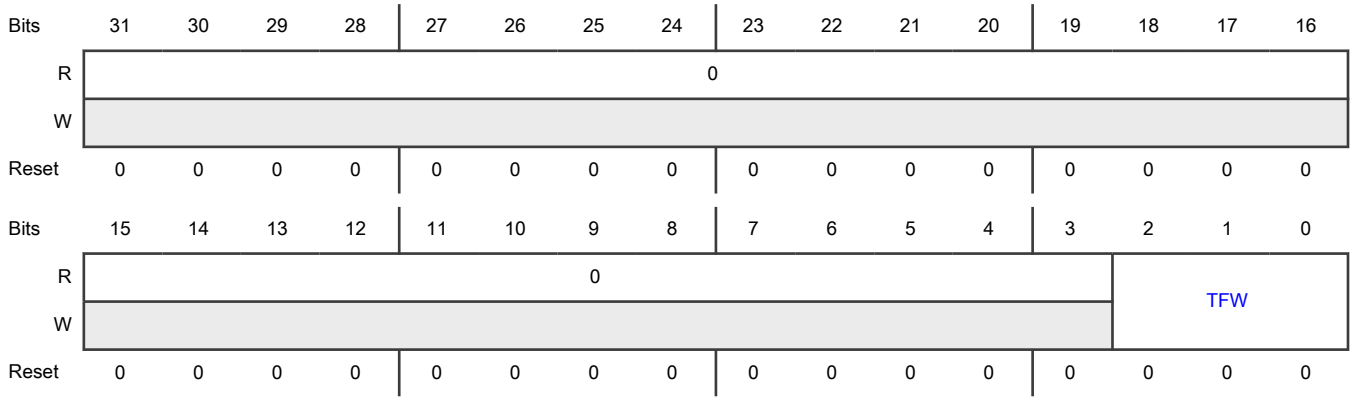
#### 71.6.1.5 Transmit Configuration 1 (TCR1)

**Offset**

| Register | Offset |
|---|---|
| TCR1 | Ch |

**Function**
Configures the watermark level for all enabled transmit channels.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2-0<br>TFW | Transmit FIFO Watermark<br>Indicates the number of 32-bit FIFO words in the watermark level of the transmit FIFO.<br>000b - 0<br>001b - 1<br>010b-110b - (TFW)<br>111b - 7 |

## 71.6.1.6 Transmit Configuration 2 (TCR2)

### Offset

| Register | Offset |
|---|---|
| TCR2 | 10h |

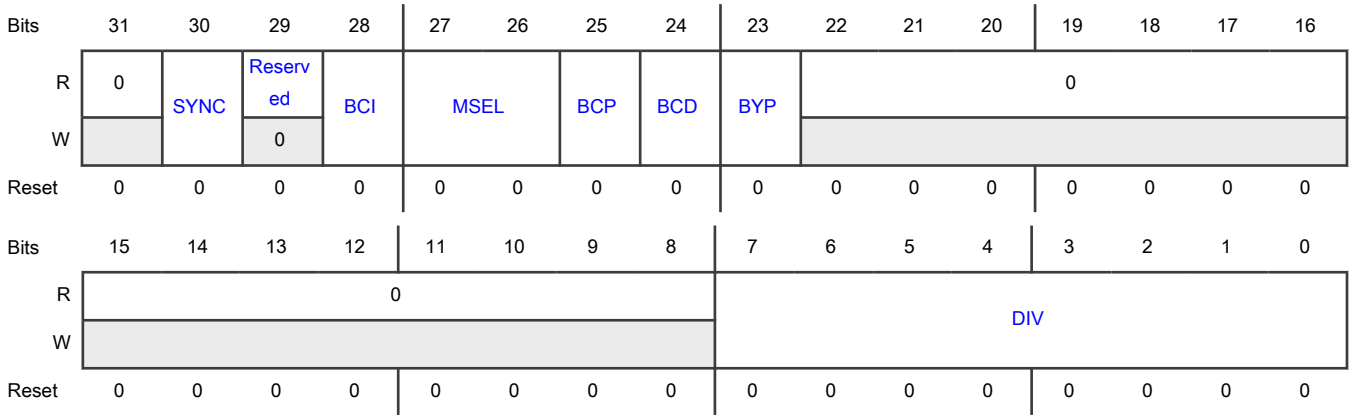
### Function

Contains the SYNC mode and clock setting fields.

**NOTE**

Do not alter this register when TCSR[TE] is 1.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>SYNC | Synchronous Mode<br>Configures Asynchronous and Synchronous modes of operation. When configured for Synchronous mode, you must configure the receiver for asynchronous operations.<br>0b - Asynchronous mode<br>1b - Synchronous with receiver |
| 29<br>— | Reserved |
| 28<br>BCI | Bit Clock Input<br>Enables the internal logic to be clocked as if the bit clock is generated externally.<br>When this field is 1 while using an internally generated bit clock in Synchronous or Asynchronous mode, the bit clock used by the transmitter is delayed by the pad output delay. (The pad input clocks the transmitter as if the clock is externally generated.) This setting decreases the data input setup time, but increases the data output valid time.<br>Use the Target mode timing from the data sheet for the transmitter when this field is 1. In Synchronous mode, this field allows the transmitter to use the Target mode timing from the data sheet, while the receiver uses the Controller mode timing. This field has no effect when configured for an externally generated bit clock.<br>**NOTE**<br>When BCI = 1, both the input buffer and output buffer must be enabled for the BCLK pad.<br>0b - Disable<br>1b - Enable |
| 27-26 | MCLK Select |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| MSEL | Selects the audio clock option used to generate an internally generated bit clock. This field has no effect when SAI is configured for an externally generated bit clock.<br>**NOTE**<br>Depending on the chip, some audio clock options might not be available. See the chip-specific information for the meaning of each option.<br>00b - Bus clock<br>01b - Audio clock (MCLK) option 1<br>10b - Audio clock (MCLK) option 2<br>11b - Audio clock (MCLK) option 3 |
| 25<br>BCP | Bit Clock Polarity<br>Configures the polarity of the bit clock. If you write 0 to this field, the bit clock becomes active high with drive outputs on rising edge and sample inputs on falling edge. If you write 1 to this field, the bit clock becomes active low with drive outputs on falling edge and sample inputs on rising edge.<br>0b - Active high<br>1b - Active low |
| 24<br>BCD | Bit Clock Direction<br>Configures the direction of the bit clock.<br>0b - Generate externally in Target mode<br>1b - Generate internally in Controller mode |
| 23<br>BYP | Bit Clock Bypass<br>Bypasses the bit clock divider. When bypassed, the internal bit clock is divide-by-one of the audio clock. When not bypassed, the internal bit clock is generated from the bit clock divider.<br>0b - Disable<br>1b - Enable |
| 22-8<br>— | Reserved |
| 7-0<br>DIV | Bit Clock Divide<br>Determines the value by which SAI divides down the audio clock to generate the bit clock (when configured for an internal bit clock). The division value is (DIV + 1) × 2. |

### 71.6.1.7 Transmit Configuration 3 (TCR3)

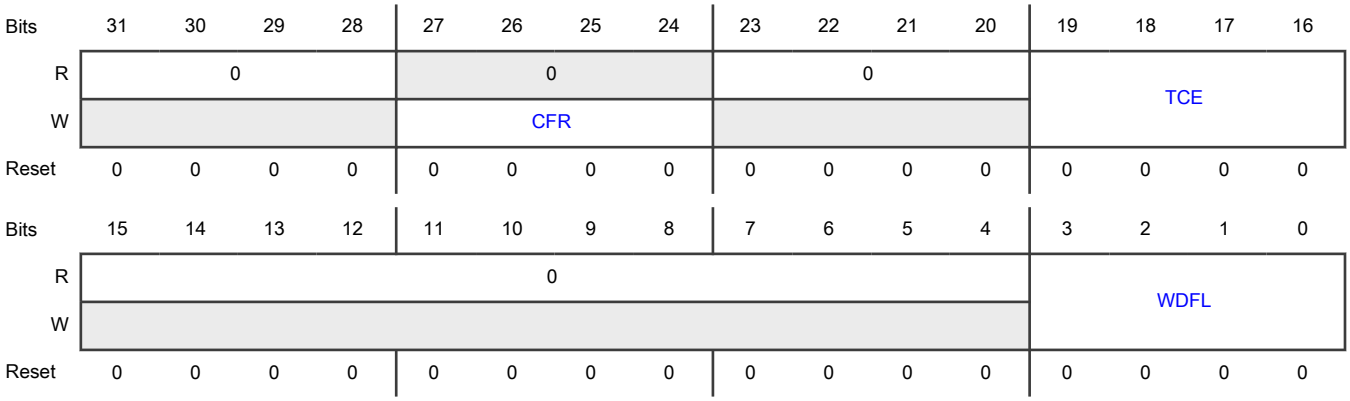
#### Offset

| Register | Offset |
|---|---|
| TCR3 | 14h |

#### Function

Contains the transmit channel settings.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>CFR | Channel FIFO Reset<br>Resets the FIFO pointers for a specific channel. Reading this field always returns zero. You must reset FIFO pointers only when a channel is disabled or the FIFO error flag is set.<br>The width of this field = the number of transmit channels (call it *n*). For example, if this field is 2 bits wide, bit position 24 refers to the transmit channel 1 FIFO pointer and bit position 25 refers to the transmit channel 2 FIFO pointer. Writing 1 to bit 24 resets the transmit channel 1 FIFO pointer, and writing 1 to bit 25 resets the transmit channel 2 FIFO pointer. Writing 1 to bit *n* resets the transmit channel *n* FIFO pointer.<br>**NOTE**<br>When there is only one channel, you do not need the individual channel FIFO reset (TCR3[CFR]). For a single channel, use the global FIFO reset (TCSR[FR]).<br>• 0b – No effect<br>• 1b – Reset transmit data channel *n* FIFO |

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>SAI_0</td><td>TCR3</td><td>—</td></tr>
<tr><td>SAI_1</td><td>—</td><td>TCR3</td></tr>
</table>
</td></tr>
<tr><td>23-20<br>—</td><td>Reserved</td></tr>
<tr><td>19-16<br>TCE</td><td>
Transmit Channel Enable<br>
Enables the corresponding data channel for transmit operations. Changing this field takes effect immediately for generating the FIFO request and warning flags. It takes effect at the end of each frame for transmit operations.<br>
The width of this field = the number of transmit channels (call it n). For example, if this field is two bits wide, bit position 16 refers to transmit channel 1 and bit position 17 refers to transmit channel 2. Writing 1 to bit 16 enables transmit channel 1, and setting bit 17 enables transmit channel 2. Writing 1 to bit n enables transmit channel n. For each transmit data channel:<br>
• 0b – Disable<br>
• 1b – Enable<br>
NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>SAI_0</td><td>TCR3</td><td>—</td></tr>
<tr><td>SAI_1</td><td>TCR3[16]</td><td>TCR3[19–17]</td></tr>
</table>
</td></tr>
<tr><td>15-4<br>—</td><td>Reserved</td></tr>
<tr><td>3-0<br>WDFL</td><td>
Word Flag Configuration<br>
Selects the word that sets the word start flag (TCSR[WSF]). The value must be one less than the word number. For example, writing 0 selects the first word in the frame. When you configure this field with a value greater than TCR4[FRSZ], TCSR[WSF] is never set.
</td></tr>
</table>

### 71.6.1.8 Transmit Configuration 4 (TCR4)

#### Offset

| Register | Offset |
|---|---|
| TCR4 | 18h |

#### Function

Contains the transmit fields for FIFO Combine mode, FIFO Packing mode, and frame sync settings.

**NOTE**

Do not alter this register when TCSR[TE] is 1.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | FCONT | FCOMB | | FPACK | | 0 | | | | FRSZ | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | SYWD | | | | | 0 | | CHMOD | MF | FSE | ONDEM | FSP | FSD |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |
| 28<br>FCONT | FIFO Continue on Error<br>Configures when SAI must continue transmitting after a FIFO error is detected. When this field is 0 and a FIFO error occurs, SAI continues from the start of the next frame after the FIFO error flag clears. When this field is 1 and a FIFO error occurs, SAI continues from the same word that caused the FIFO error after the FIFO warning flag clears.<br>0b - Continue from the start of the next frame<br>1b - Continue from the same word that caused the FIFO error |
| 27-26<br>FCOMB | FIFO Combine Mode<br>Enables FIFO Combine mode for specified operations.<br>When FIFO Combine mode is enabled for FIFO writes, software writing to any FIFO data register alternates the write among the enabled data channel FIFOs. For example, if two data channels are enabled: |

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<ul>
<li>The first write is performed to the first enabled data channel FIFO.</li>
<li>The second write is performed to the second enabled data channel FIFO.</li>
</ul>
Resetting the FIFO or disabling FIFO Combine mode for FIFO writes resets the pointer back to the first enabled data channel.<br><br>
When FIFO Combine mode is enabled for FIFO reads from the transmit shift registers, the transmit data channel output alternates between the enabled data channel FIFOs. For example, if two data channels are enabled:
<ul>
<li>The first unmasked word is transmitted from the first enabled data channel FIFO.</li>
<li>The second unmasked word is transmitted from the second enabled data channel FIFO.</li>
</ul>
Because the first word of the frame is always transmitted from the first enabled data channel FIFO, NXP recommends that the number of unmasked words per frame is evenly divisible by the number of enabled data channels.<br><br>
NOTE<br>
This field is not supported in every instance. The following table includes only supported registers.<br><br>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>SAI_0</td><td>TCR4</td><td>—</td></tr>
<tr><td>SAI_1</td><td>—</td><td>TCR4</td></tr>
</table>
00b - Disable<br>
01b - Enable on FIFO reads (from transmit shift registers)<br>
10b - Enable on FIFO writes (by software)<br>
11b - Enable on FIFO reads (from transmit shift registers) and writes (by software)
</td></tr>
<tr><td>25-24<br>FPACK</td><td>FIFO Packing Mode<br><br>
Enables packing of 8-bit data or 16-bit data into each 32-bit FIFO word. If the word size is greater than 8 bits or 16 bits, only the first 8 bits or 16 bits are loaded from the FIFO. The first word in each frame always starts with a new 32-bit FIFO word and the first bit shifted must be configured within the first packed word. When FIFO packing is enabled, the FIFO write pointer only increments when you write the full 32-bit FIFO word.<br>
00b - Disable FIFO packing<br>
01b - Reserved<br>
10b - Enable 8-bit FIFO packing<br>
11b - Enable 16-bit FIFO packing</td></tr>
<tr><td>23-20<br>—</td><td>Reserved</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 19-16<br>FRSZ | Frame Size<br>Configures the number of words in each frame. The value must be one less than the number of words in the frame. For example, write 0 for one word per frame. The maximum supported frame size is 16 words. |
| 15-13<br>— | Reserved |
| 12-8<br>SYWD | Sync Width<br>Configures the length of the frame sync in number of bit-clock cycles. The value must be one less than the number of bit-clock cycles. For example, write 0 for the frame sync to assert for one bit-clock cycle only. You cannot configure the sync width to be longer than the first word of the frame. |
| 7-6<br>— | Reserved |
| 5<br>CHMOD | Channel Mode<br>Specifies the mode of transmit data pins. In TDM mode, transmit data pins are 3-stated when slots are masked or channels are disabled. In Output mode, transmit data pins are never 3-stated, and they output zero when slots are masked or channels are disabled.<br>0b - TDM mode<br>1b - Output mode |
| 4<br>MF | MSB First<br>Configures what is transmitted first: LSB or MSB.<br>0b - LSB<br>1b - MSB |
| 3<br>FSE | Frame Sync Early<br>Determines when frame sync is asserted.<br>0b - First bit of the frame<br>1b - One bit before the first bit of the frame |
| 2<br>ONDEM | On-Demand Mode<br>Determines when the internal frame sync is generated. When this field is 1, and the frame sync is generated internally, a frame sync is only generated after the FIFO warning flag is cleared.<br>0b - Generated continuously<br>1b - Generated after the FIFO warning flag is cleared |
| 1<br>FSP | Frame Sync Polarity<br>Configures the polarity of the frame sync. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Active high<br>1b - Active low |
| 0<br>FSD | Frame Sync Direction<br>Configures the direction of the frame sync.<br>0b - Generated externally in Target mode<br>1b - Generated internally in Controller mode |

### 71.6.1.9 Transmit Configuration 5 (TCR5)

#### Offset

| Register | Offset |
|---|---|
| TCR5 | 1Ch |

#### Function

Contains transmit word width and bit index settings.

**NOTE**

Do not alter this register when TCSR[TE] is 1.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | WNW | | | | | 0 | | | W0W | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

#### Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 28-24<br>WNW | Word N Width<br>Configures the number of bits in each word, except for the first one in the frame. The value written must be one less than the number of bits per word. SAI does not support word widths of less than 8 bits.<br>SAI does not support bit settings 0–6 (0b–0_0110b).<br>0_0111b - 8<br>0_1000b - 9<br>0_1001b-1_1110b - (WNW value + 1)<br>1_1111b - 32 |
| 23-21<br>— | Reserved |
| 20-16<br>W0W | Word 0 Width<br>Configures the number of bits in the first word of each frame. The value written must be one less than the number of bits in the first word. SAI does not support word widths of less than 8 bits when there is only one word per frame.<br>SAI does not support bit settings 0–6 (0b–0_0110b).<br>0_0111b - 8<br>0_1000b - 9<br>0_1001b-1_1110b - (W0W value + 1)<br>1_1111b - 32 |
| 15-13<br>— | Reserved |
| 12-8<br>FBT | First Bit Shifted<br>Configures the bit index for the first bit transmitted for each word in the frame. If configured for MSB-first, the index of the next bit transmitted is one less than the current bit transmitted. If configured for LSB-first, the index of the next bit transmitted is one more than the current bit transmitted. The value written must be greater than or equal to the word width when configured for MSB-first. The value written must be less than or equal to 31-word width when configured for LSB-first.<br>See Data alignment for details.<br>0_0000b - 0<br>0_0001b-1_1110b - FBT<br>1_1111b - 31 |
| 7-0<br>— | Reserved |

### 71.6.1.10 Transmit Data (TDR0 - TDR3)

#### Offset

| Register | Offset |
|---|---|
| TDR0 | 20h |
| TDR1 | 24h |
| TDR2 | 28h |
| TDR3 | 2Ch |

#### Function

Contains transmit data.

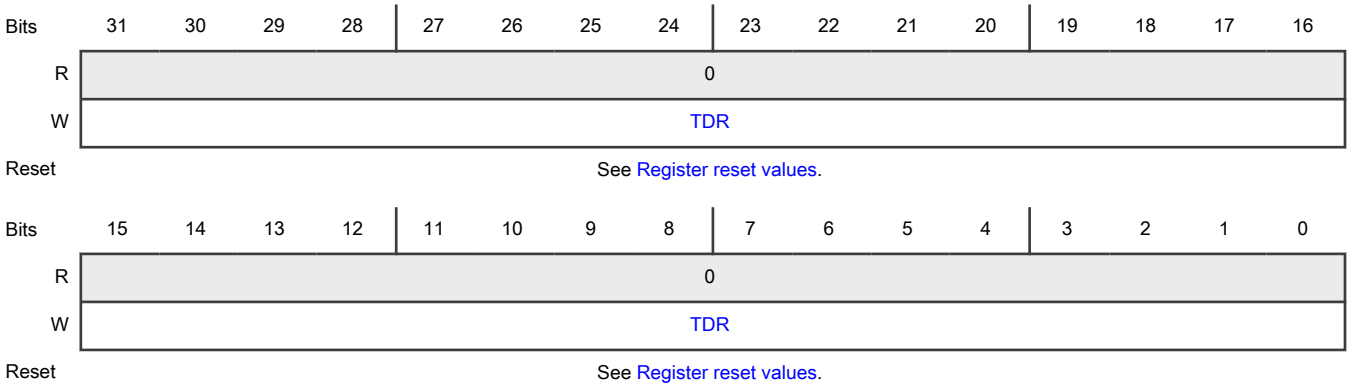
When the transmit FIFO is not full, writes to this register push the data written into the transmit data FIFO. When the transmit FIFO is full, SAI ignores writes to this register.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| SAI_0 | TDR0–TDR3 | — |
| SAI_1 | TDR0 | TDR1–TDR3 |

#### Diagram




#### Register reset values

| Register | Reset value |
|---|---|
| TDR0 | SAI_0,SAI_1: 0000_0000h |
| TDR1–TDR3 | 0000_0000h |

### Fields

| Field | Function |
|---|---|
| 31-0<br>TDR | Transmit Data |

## 71.6.1.11 Transmit FIFO (TFR0 - TFR3)

### Offset

| Register | Offset |
|---|---|
| TFR0 | 40h |
| TFR1 | 44h |
| TFR2 | 48h |
| TFR3 | 4Ch |

### Function

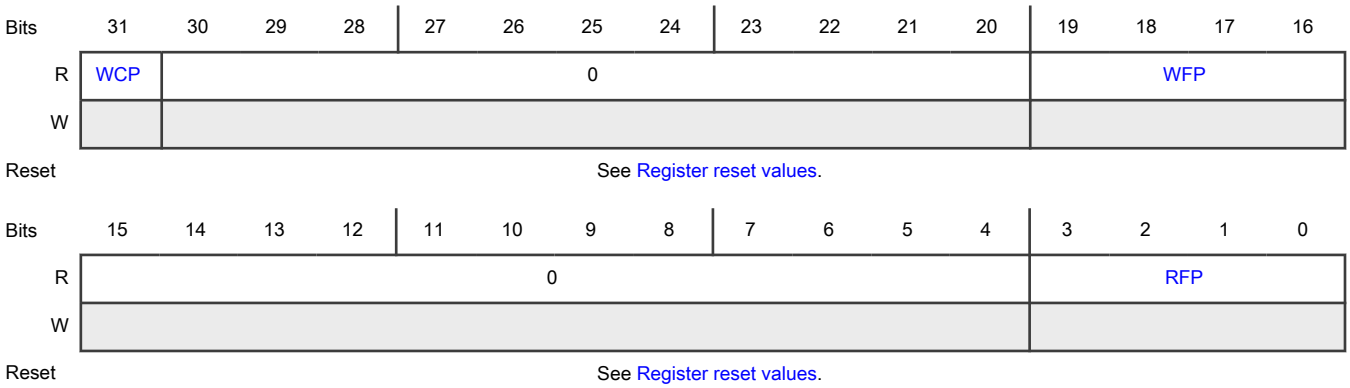
Contains the transmit FIFO pointers. The MSB of the read and write pointers distinguishes between FIFO full and empty conditions. If the read and write pointers are identical, the FIFO is empty. If the read and write pointers are identical except for the MSB, the FIFO is full.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| SAI_0 | TFR0–TFR3 | — |
| SAI_1 | TFR0 | TFR1–TFR3 |

### Diagram



**Register reset values**

| Register | Reset value |
|---|---|
| TFR0 | SAI_0,SAI_1: 0000_0000h |
| TFR1–TFR3 | 0000_0000h |

**Fields**

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>31<br>WCP</td><td>Write Channel Pointer<br>Indicates whether this data channel is the next FIFO to be written when FIFO Combine mode is enabled for write operations.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>SAI_0</td><td>TFR0–TFR3</td><td>—</td></tr><tr><td>SAI_1</td><td>—</td><td>TFR0</td></tr></table>0b - No effect<br>1b - Next FIFO to be written</td></tr>
<tr><td>30-20<br>—</td><td>Reserved</td></tr>
<tr><td>19-16<br>WFP</td><td>Write FIFO Pointer<br>Indicates the FIFO write pointer for the transmit data channel.</td></tr>
<tr><td>15-4<br>—</td><td>Reserved</td></tr>
<tr><td>3-0<br>RFP</td><td>Read FIFO Pointer<br>Indicates the FIFO read pointer for the transmit data channel.</td></tr>
</table>

### 71.6.1.12 Transmit Mask (TMR)

**Offset**

| Register | Offset |
|---|---|
| TMR | 60h |

### Function

Contains the mask for the transmit word. This register is double-buffered and updates:

- When TCSR[TE] first becomes 1.
- At the end of each frame.

This setup allows the masked words in each frame to change from frame to frame.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>TWM | Transmit Word Mask<br>Configures whether the transmit word *n* is masked for the corresponding word in the frame. In this case, being masked means that transmit data pins are 3-stated or driven zero and transmit data is not read from the FIFO.<br>0000_0000_0000_0000b - Enable<br>0000_0000_0000_0001b - Mask |

## 71.6.1.13 Receive Control (RCSR)

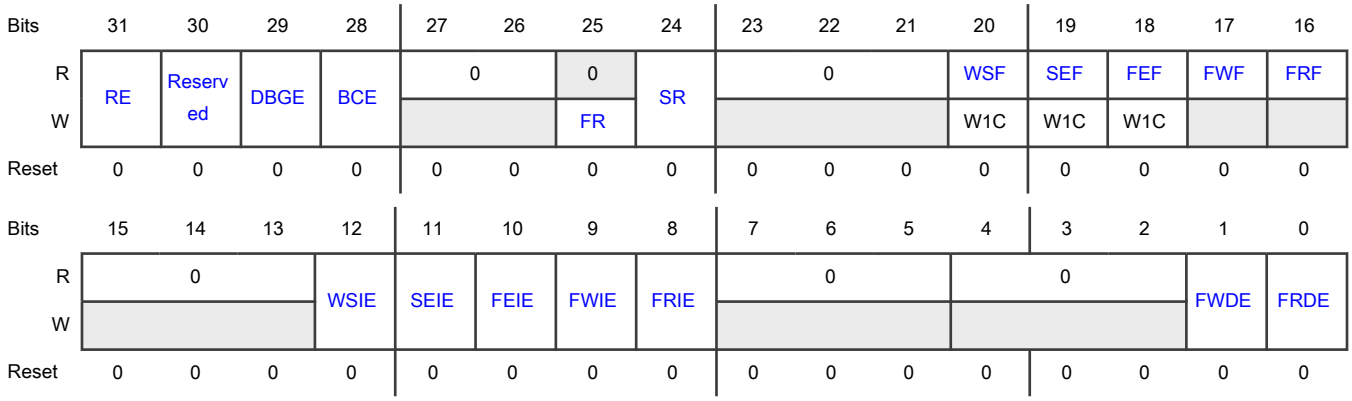
### Offset

| Register | Offset |
|---|---|
| RCSR | 88h |

### Function

Contains receiver enable fields including resets, error and interrupt enable fields, and error flag fields.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>RE | Receiver Enable<br>Enables the receiver. When you write 0 to this field, the receiver remains enabled and the field remains 1 until the end of the current frame.<br>0b - Disable<br>1b - Enable (or receiver disabled and not yet reached end of frame) |
| 30<br>— | Reserved<br>Only write zero to this field. |
| 29<br>DBGE | Debug Enable<br>Enables receiver operation in Debug mode, which does not affect the receive bit clock.<br>0b - Disable after completing the current frame<br>1b - Enable |
| 28<br>BCE | Bit Clock Enable<br>Enables the receive bit clock, separately from RCSR[RE]. This field becomes 1 automatically when RCSR[RE] becomes 1. When you write 0 to this field, the receive bit clock remains enabled, and this field remains 1 until the end of the current frame.<br>0b - Disable<br>1b - Enable |
| 27-26<br>— | Reserved |
| 25<br>FR | FIFO Reset<br>Empties the FIFO, and sets the FIFO read and write pointers to the same value, which can be zero. Reading this field always returns zero. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>The FIFO reset is asserted for one cycle only.<br>You must reset FIFO pointers only when the receiver is disabled or the FIFO error flag is 1.<br>0b - No effect<br>1b - Reset |
| 24<br>SR | Software Reset<br>Resets the internal receiver logic including the FIFO pointers. Software-visible registers are not affected, except for the status registers.<br>**NOTE**<br>The software reset remains asserted until you clear it.<br>0b - No effect<br>1b - Software reset |
| 23-21<br>— | Reserved |
| 20<br>WSF | Word Start Flag<br>Indicates whether SAI has detected the start of the configured word.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not detected<br>1b - Detected<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 19<br>SEF | Sync Error Flag<br>Indicates whether SAI has detected an error in the externally generated frame sync.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not detected<br>1b - Detected<br>When writing |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No effect<br>1b - Clear the flag |
| 18<br>FEF | FIFO Error Flag<br>Indicates whether an enabled receive FIFO has overflowed.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No error<br>1b - Receive overflow detected<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 17<br>FWF | FIFO Warning Flag<br>Indicates whether an enabled receive FIFO is full.<br>0b - Not full<br>1b - Full |
| 16<br>FRF | FIFO Request Flag<br>Indicates whether the number of words in an enabled receive channel FIFO is greater than the receive FIFO watermark.<br>0b - Watermark not reached<br>1b - Watermark reached |
| 15-13<br>— | Reserved |
| 12<br>WSIE | Word Start Interrupt Enable<br>Enables word start interrupts.<br>0b - Disable<br>1b - Enable |
| 11<br>SEIE | Sync Error Interrupt Enable<br>Enables sync error interrupts.<br>0b - Disable<br>1b - Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 10<br>FEIE | FIFO Error Interrupt Enable<br>Enables FIFO error interrupts.<br>0b - Disable<br>1b - Enable |
| 9<br>FWIE | FIFO Warning Interrupt Enable<br>Enables FIFO warning interrupts.<br>0b - Disable<br>1b - Enable |
| 8<br>FRIE | FIFO Request Interrupt Enable<br>Enables FIFO request interrupts.<br>0b - Disable<br>1b - Enable |
| 7-5<br>— | Reserved |
| 4-2<br>— | Reserved |
| 1<br>FWDE | FIFO Warning DMA Enable<br>Enables DMA warnings.<br>0b - Disable<br>1b - Enable |
| 0<br>FRDE | FIFO Request DMA Enable<br>Enables DMA requests.<br>0b - Disable<br>1b - Enable |

### 71.6.1.14 Receive Configuration 1 (RCR1)

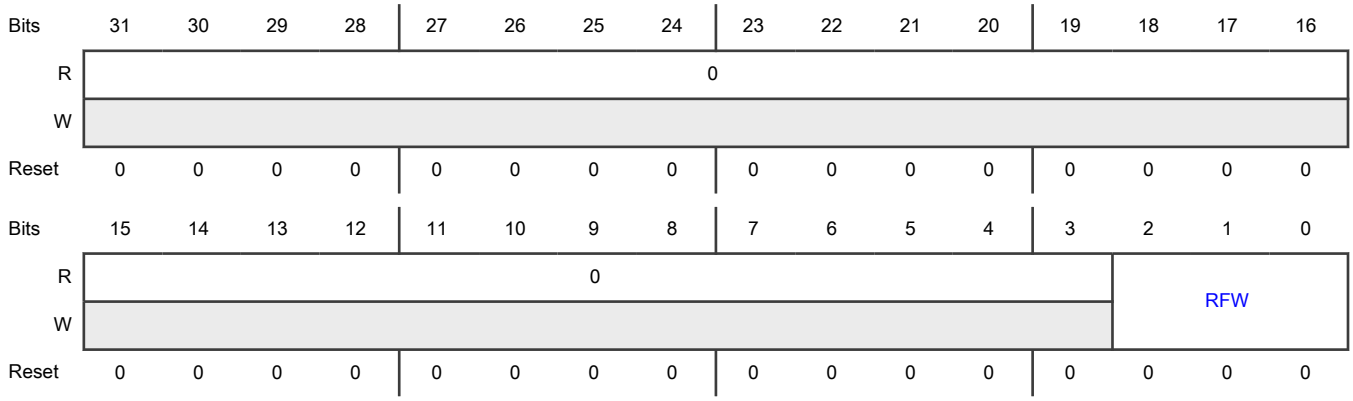
#### Offset

| Register | Offset |
|---|---|
| RCR1 | 8Ch |

#### Function

Configures the watermark level for all enabled receiver channels.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-3<br>— | Reserved |
| 2-0<br>RFW | Receive FIFO Watermark<br>Configures the level of the receive FIFO watermark, in 32-bit FIFO words.<br>000b - 1<br>001b - 2<br>010b-110b - (RFW value + 1)<br>111b - 8 |

## 71.6.1.15 Receive Configuration 2 (RCR2)

### Offset

| Register | Offset |
|---|---|
| RCR2 | 90h |

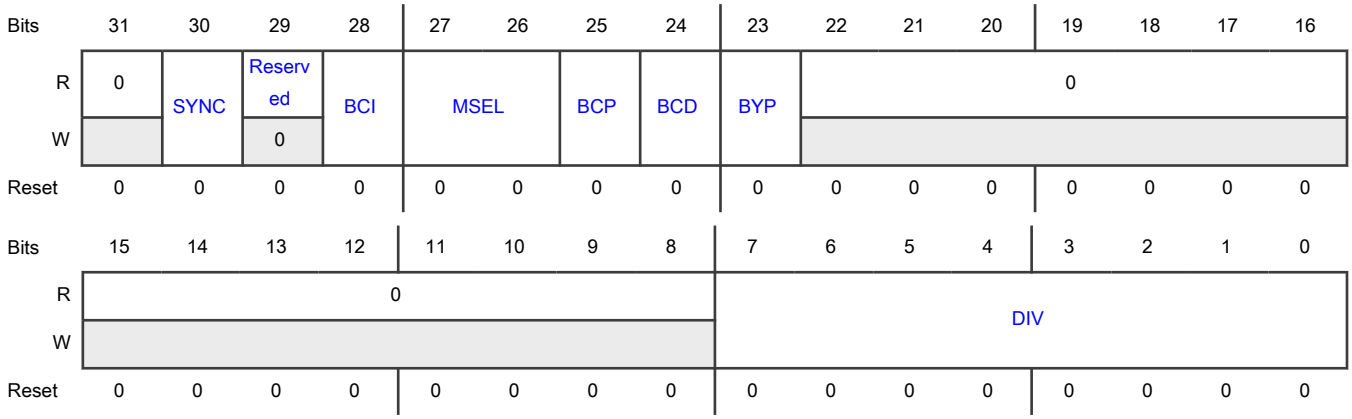
### Function

Contains the SYNC mode and clock setting fields.

**NOTE**

Do not alter this register when RCSR[RE] is 1.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>SYNC | Synchronous Mode<br>Configures Asynchronous and Synchronous modes of operation. When configured for a Synchronous mode of operation, you must configure the transmitter for asynchronous operation.<br>0b - Asynchronous mode<br>1b - Synchronous with transmitter |
| 29<br>— | Reserved |
| 28<br>BCI | Bit Clock Input<br>Enables the internal logic to be clocked as if the bit clock is generated externally.<br>When this field is 1 and when using an internally generated bit clock in Synchronous or Asynchronous mode, the bit clock used by the receiver is delayed by the pad output delay. (The pad input clocks the receiver as if the clock is externally generated.) This setting decreases the data input setup time, but increases the data output valid time.<br>Use the Target mode timing from the data sheet for the receiver when this field is 1. In Synchronous mode, this field allows the receiver to use the Target mode timing from the data sheet, while the transmitter uses the Controller mode timing. This field has no effect when configured for an externally generated bit clock.<br>**NOTE**<br>When this field is 1, both the input buffer and output buffer must be enabled for the BCLK pad.<br>0b - Disable<br>1b - Enable |
| 27-26 | MCLK Select |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| MSEL | Selects the audio clock option used to generate an internally generated bit clock. This field has no effect when configured for an externally generated bit clock.<br>**NOTE**<br>Some audio clock options are not available for some chips. See the chip-specific information for the availability and chip-specific meaning of each option.<br>00b - Bus clock<br>01b - Audio clock (MCLK) option 1<br>10b - Audio clock (MCLK) option 2<br>11b - Audio clock (MCLK) option 3 |
| 25<br>BCP | Bit Clock Polarity<br>Configures the polarity of the bit clock. If you write 0 to this field, the bit clock becomes active high with drive outputs on the rising edge and sample inputs on the falling edge. If you write 1 to this field, the bit clock becomes active low with drive outputs on the falling edge and sample inputs on the rising edge.<br>0b - Active high<br>1b - Active low |
| 24<br>BCD | Bit Clock Direction<br>Configures the direction of the bit clock.<br>0b - Generated externally in Target mode<br>1b - Generated internally in Controller mode |
| 23<br>BYP | Bit Clock Bypass<br>Enables the bypass of the bit clock divider. When enabled, the internal bit clock is divide-by-one of the audio clock. When disabled, the internal bit clock is generated from the bit clock divider.<br>0b - Disable<br>1b - Enable |
| 22-8<br>— | Reserved |
| 7-0<br>DIV | Bit Clock Divide<br>Determines the value by which the audio clock is divided to generate the bit clock, when configured for an internal bit clock. The division value is (DIV + 1) × 2. |

### 71.6.1.16 Receive Configuration 3 (RCR3)

#### Offset

| Register | Offset |
|---|---|
| RCR3 | 94h |

#### Function

Contains the receive channel settings.

#### Diagram

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

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

#### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>CFR | Channel FIFO Reset<br>Resets the FIFO pointers for a specific channel. Reading this field always returns zero. Reset FIFO pointers only when a channel is disabled or the FIFO error flag is set.<br>The width of this field = the number of receive channels (call it *n*). For example, if this field is two bits wide, bit position 24 refers to the receive channel 1 FIFO pointer and bit position 25 refers to the receive channel 2 FIFO pointer. Writing 1 to bit 24 resets the receive channel 1 FIFO pointer, and writing 1 to bit 25 enables the receive channel 2 FIFO pointer. Writing 1 to bit *n* resets receive channel *n* FIFO pointer.<br>• 0b – No effect<br>• 1b – Reset receive data channel *n* FIFO<br>**NOTE**<br>This field is not supported in every instance. The following table includes only supported registers. |

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>SAI_0</td><td>RCR3</td><td>—</td></tr>
<tr><td>SAI_1</td><td>—</td><td>RCR3</td></tr>
</table>
</td></tr>
<tr><td>23-20<br>—</td><td>Reserved</td></tr>
<tr><td>19-16<br>RCE</td><td>Receive Channel Enable<br>Enables the corresponding data channel for receive operation. Changing this field takes effect immediately for generating the FIFO request and warning flags. It takes effect at the end of each frame for receive operations.<br>The width of RCE = the number of receive channels (call it n). For example, if RCE is two bits wide, then bit position 16 refers to receive channel 1 and bit position 17 refers to receive channel 2. Writing 1 to bit 16 enables receive channel 1, and writing 1 to bit 17 enables receive channel 2. Writing 1 to bit n enables receive channel n.<br>NOTE<br>When there is only a single channel, you do not need an individual channel FIFO reset (RCR3[CFR]). In that case, use the global FIFO reset (RCSR[FR]).<br>• 0b – Disable receive data channel n<br>• 1b – Enable receive data channel n<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>SAI_0</td><td>RCR3</td><td>—</td></tr>
<tr><td>SAI_1</td><td>RCR3[16]</td><td>RCR3[19–17]</td></tr>
</table>
</td></tr>
<tr><td>15-4<br>—</td><td>Reserved</td></tr>
<tr><td>3-0<br>WDFL</td><td>Word Flag Configuration<br>Configures which word sets the word start flag (RCSR[WSF]). The value must be one less than the word number (for example, write zero to select the first word in the frame). If you configure this field to a value greater than RCR4[FRSZ], RCSR[WSF] is never set.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0000b - Word 1<br>0001b - Word 2<br>0010b-1110b - Word (WDFL value + 1)<br>1111b - Word 16 |

### 71.6.1.17 Receive Configuration 4 (RCR4)

#### Offset

| Register | Offset |
|---|---|
| RCR4 | 98h |

#### Function

Contains the receive fields for FIFO Combine mode, FIFO Packing mode, and frame sync settings.

**NOTE**

Do not alter this register when RCSR[RE] is 1.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | FCONT | FCOMB | | FPACK | | 0 | | | | FRSZ | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | SYWD | | | | | 0 | | Reserved | MF | FSE | ONDEM | FSP | FSD |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |
| 28<br>FCONT | FIFO Continue on Error<br>Configures when SAI continues receiving data after a FIFO error is detected. |

*Table continues on the next page...*

*Table continued from the previous page...*

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>0b - From the start of the next frame after the FIFO error flag is cleared<br>1b - From the same word that caused the FIFO error to become 1 after the FIFO warning flag is cleared</td></tr>
<tr><td>27-26<br>FCOMB</td><td>FIFO Combine Mode<br>Enables FIFO Combine mode for specified operations.<br>When FIFO Combine mode is enabled for FIFO reads, software reading any FIFO data register alternates the read among the enabled data channel FIFOs. For example, if two data channels are enabled:<br>• The first read is performed to the first enabled data channel FIFO.<br>• The second read is performed to the second enabled data channel FIFO.<br>Resetting the FIFO or disabling FIFO Combine mode for FIFO reads resets the pointer back to the first enabled data channel.<br>When FIFO Combine mode is enabled for FIFO writes from the receive shift registers, the first enabled data channel input alternates between the enabled data channel FIFOs. For example, if two data channels are enabled:<br>• The first unmasked received word is stored in the first enabled data channel FIFO.<br>• The second unmasked received word is stored in the second enabled data channel FIFO.<br>Because the first word of the frame is always stored in the first enabled data channel FIFO, NXP recommends that the number of unmasked words per frame is evenly divisible by the number of enabled data channels.<br>NOTE<br>This field is not supported in every instance. The following table includes only supported registers.<br><table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>SAI_0</td><td>RCR4</td><td>—</td></tr><tr><td>SAI_1</td><td>—</td><td>RCR4</td></tr></table><br>00b - Disable<br>01b - Enable on FIFO writes (from receive shift registers)<br>10b - Enable on FIFO reads (by software)<br>11b - Enable on FIFO writes (from receive shift registers) and reads (by software)</td></tr>
<tr><td>25-24<br>FPACK</td><td>FIFO Packing Mode<br>Enables packing of 8-bit data or 16-bit data into each 32-bit FIFO word. If the word size is greater than 8 bits or 16 bits, only the first 8 bits or 16 bits are stored in the FIFO. The first word in each frame always starts with a new 32-bit FIFO word and the first bit shifted must be configured within the first packed word. When FIFO packing is enabled, the FIFO read pointer only increments when software reads the full 32-bit FIFO word.</td></tr>
</table>

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 00b - Disable<br>01b - Reserved<br>10b - Enable 8-bit FIFO packing<br>11b - Enable 16-bit FIFO packing |
| 23-20<br>— | Reserved |
| 19-16<br>FRSZ | Frame Size<br>Configures the number of words in each frame. The value must be one less than the number of words in the frame. For example, write 0 for one word per frame. The maximum supported frame size is 16 words.<br>0000b - 1<br>0001b - 2<br>0010b-1110b - (FRSZ value + 1)<br>1111b - 16 |
| 15-13<br>— | Reserved |
| 12-8<br>SYWD | Sync Width<br>Configures the length of the frame sync in number of bit-clock cycles. The value must be one less than the number of bit-clock cycles. For example, write 0 for the frame sync to assert for one bit-clock cycle only. You cannot configure this field to be longer than the first word of the frame.<br>0_0000b - 1<br>0_0001b - 2<br>0_0010b-1_1110b - (SYWD value + 1)<br>1_1111b - 32 |
| 7-6<br>— | Reserved |
| 5<br>— | Reserved<br>Write only zero to this field. |
| 4<br>MF | MSB First<br>Configures what is received first: LSB or MSB.<br>0b - LSB<br>1b - MSB |
| 3 | Frame Sync Early |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| FSE | Determines when frame sync is asserted.<br>0b - First bit of the frame<br>1b - One bit before the first bit of the frame |
| 2<br>ONDEM | On-Demand Mode<br>Determines when the internal frame sync is generated. When this field is 1, and the frame sync is generated internally, a frame sync is only generated when the FIFO warning flag is 0.<br>0b - Generated continuously<br>1b - Generated when the FIFO warning flag is 0 |
| 1<br>FSP | Frame Sync Polarity<br>Configures the polarity of the frame sync.<br>0b - Active high<br>1b - Active low |
| 0<br>FSD | Frame Sync Direction<br>Configures the direction of the frame sync.<br>0b - Generated externally in Target mode<br>1b - Generated internally in Controller mode |

### 71.6.1.18 Receive Configuration 5 (RCR5)

#### Offset

| Register | Offset |
|---|---|
| RCR5 | 9Ch |

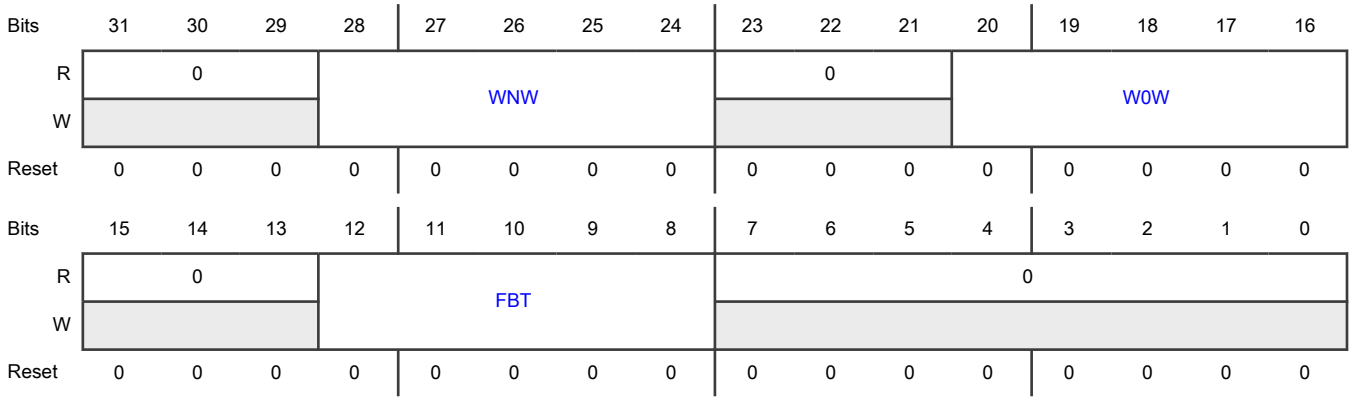
#### Function

Contains receive word and bit index settings.

**NOTE**

Do not alter this register when RCSR[RE] is 1.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |
| 28-24<br>WNW | Word N Width<br>Configures the number of bits in each word except for the first one in the frame. The value must be one less than the number of bits per word. SAI does not support word widths less than 8 bits.<br>SAI does not support bit settings 0–6 (0b–0_0110b).<br>0_0111b - 8<br>0_1000b - 9<br>0_1001b-1_1110b - (WNW value + 1)<br>1_1111b - 32 |
| 23-21<br>— | Reserved |
| 20-16<br>W0W | Word 0 Width<br>Configures the number of bits in the first word of each frame. The value must be one less than the number of bits in the first word. SAI does not support word widths less than 8 bits when there is only one word per frame.<br>0_0000b - 1<br>0_0001b - 2<br>0_0010b-1_1110b - (W0W value + 1)<br>1_1111b - 32 |
| 15-13<br>— | Reserved |
| 12-8 | First Bit Shifted |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| FBT | Configures the bit index for the first bit received for each word in the frame. If configured for MSB-first, the index of the next bit received is one less than the current bit received. If configured for LSB-first, the index of the next bit received is one more than the current bit received. The value must be greater than or equal to the word width when configured for MSB-first. The value must be less than or equal to 31-word width when configured for LSB-first.<br>See Data alignment for details.<br>0_0000b - 0<br>0_0001b-1_1110b - FBT value<br>1_1111b - 31 |
| 7-0<br>— | Reserved |

### 71.6.1.19 Receive Data (RDR0 - RDR3)

#### Offset

| Register | Offset |
|---|---|
| RDR0 | A0h |
| RDR1 | A4h |
| RDR2 | A8h |
| RDR3 | ACh |

#### Function

Contains receive data.

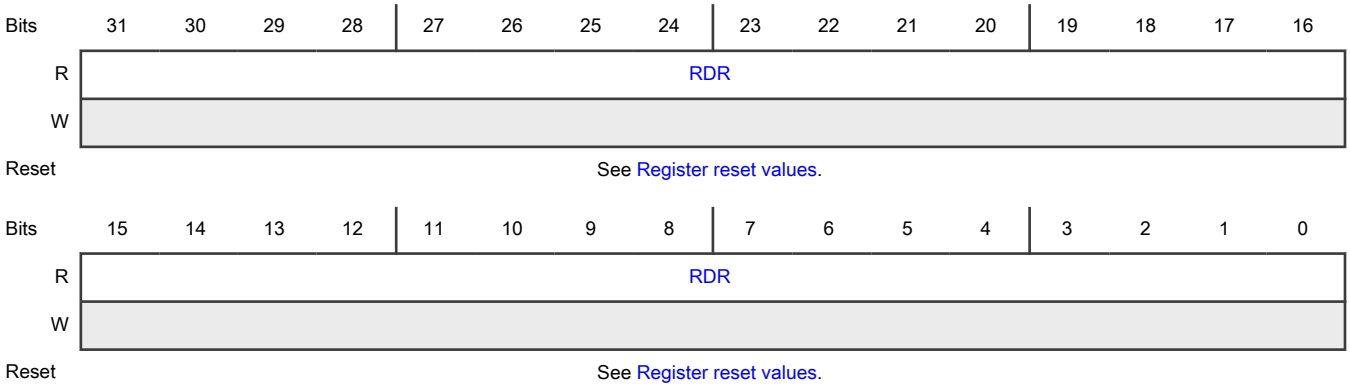
When the receive FIFO is not empty, reads from this register return the data from the top of the receive FIFO. When the receive FIFO is empty, SAI ignores reads from this register.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| SAI_0 | RDR0–RDR3 | — |
| SAI_1 | RDR0 | RDR1–RDR3 |

### Diagram




### Register reset values

| Register | Reset value |
|---|---|
| RDR0 | SAI_0,SAI_1: 0000_0000h |
| RDR1–RDR3 | 0000_0000h |

### Fields

| Field | Function |
|---|---|
| 31-0<br>RDR | Receive Data |

## 71.6.1.20 Receive FIFO (RFR0 - RFR3)

### Offset

| Register | Offset |
|---|---|
| RFR0 | C0h |
| RFR1 | C4h |
| RFR2 | C8h |
| RFR3 | CCh |

### Function

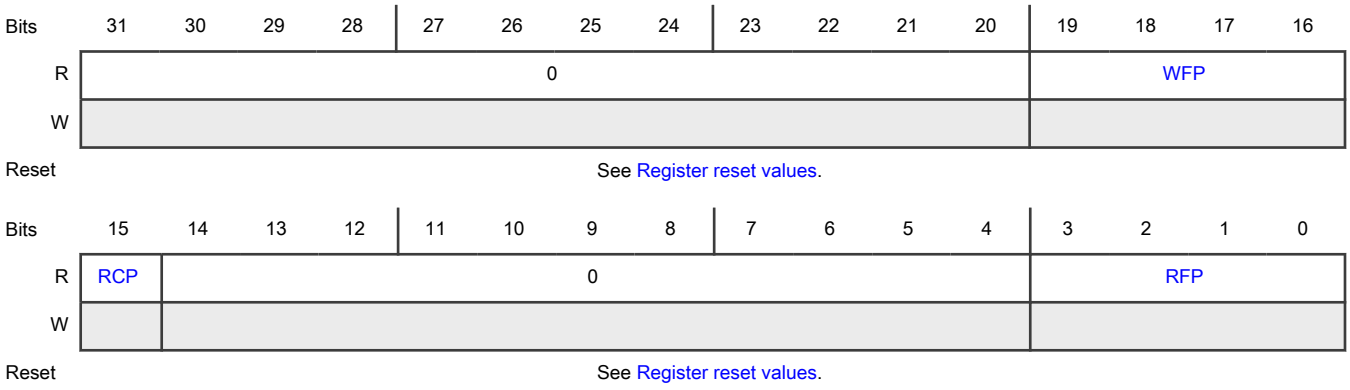
Contains the receive FIFO pointers. The MSB of the read and write pointers distinguishes between FIFO full and empty conditions. If the read and write pointers are identical, the FIFO is empty. If the read and write pointers are identical except for the MSB, the FIFO is full.

**NOTE**

Each module instance supports a different number of registers.

| Instance | Register supported | Register not supported |
|---|---|---|
| SAI_0 | RFR0–RFR3 | — |
| SAI_1 | RFR0 | RFR1–RFR3 |

### Diagram




### Register reset values

| Register | Reset value |
|---|---|
| RFR0 | SAI_0,SAI_1: 0000_0000h |
| RFR1–RFR3 | 0000_0000h |

### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19-16<br>WFP | Write FIFO Pointer<br>Indicates the FIFO write pointer for the receive data channel. |
| 15<br>RCP | Read Channel Pointer<br>Indicates whether this data channel is the next FIFO to be read when FIFO Combine mode is enabled for read operations.<br>**NOTE**<br>This field is not supported in every instance. The following table includes only supported registers. |

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | <table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>SAI_0</td><td>RFR0–RFR3</td><td>—</td></tr><tr><td>SAI_1</td><td>—</td><td>RFR0</td></tr></table><br>0b - No effect<br>1b - Next FIFO to be read |
| 14-4<br>— | Reserved |
| 3-0<br>RFP | Read FIFO Pointer<br>Indicates the FIFO read pointer for the receive data channel. |

### 71.6.1.21 Receive Mask (RMR)

#### Offset

| Register | Offset |
|---|---|
| RMR | E0h |

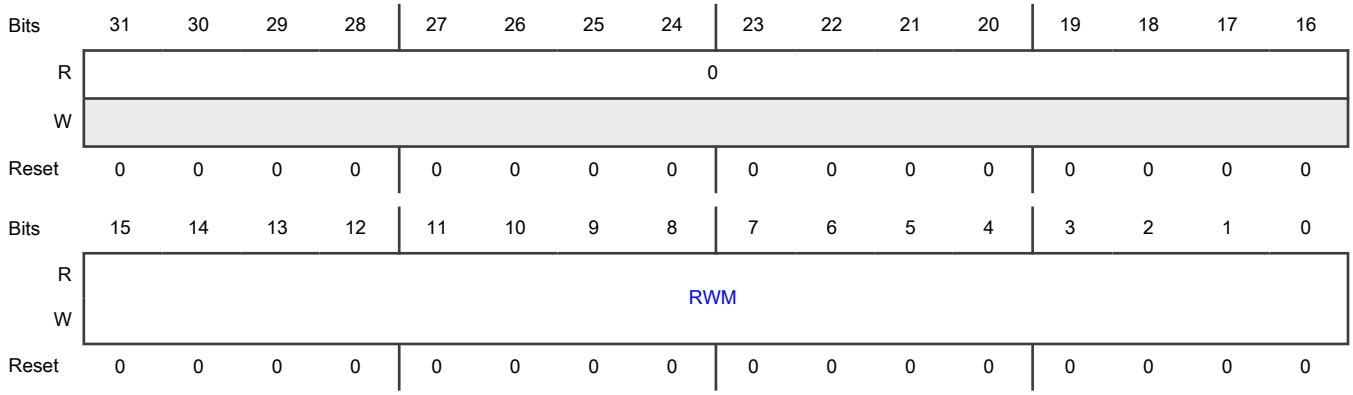
#### Function

Contains the mask for the receive word. This register is double-buffered and updates:

- When RCSR[RE] first becomes 1.
- At the end of each frame.

This setup allows the masked words in each frame to change from frame to frame.

#### Diagram



**Fields**

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>RWM | Receive Word Mask<br>Configures whether the receive word is masked (received data ignored and not written to receive FIFO) for the corresponding word in the frame.<br>0000_0000_0000_0000b - Enable<br>0000_0000_0000_0001b - Mask |

## 71.7 Glossary

| | |
|---|---|
| **FIFO** | First-in first-out |
| **LSB** | Least significant bit |
| **MSB** | Most significant bit |

# Chapter 72
# Ethernet Media Access Controller (EMAC)

## 72.1 Chip-specific EMAC information

### 72.1.1 EMAC instance and configuration

This chip supports up to 1 instance of EMAC IP.

**Table 473. EMAC instance**

| Instance | MCXE31B | MCXE315/MCXE316/ MCXE317 |
|---|---|---|
| EMAC | Yes | No |

**NOTE**

DWC_EQOS refer to the Synopsys Ethernet MAC IP.

**NOTE**

The description for some EMAC registers include references to configurations of the Synopsis Ethernet MAC IP. Refer to MAC_HW_Feature registers for the specific features available in this chip.

**NOTE**

EMAC operates only in Clock options A and B, since the module clock becomes lower than the protocol clock (RMII/MII clocks) in other modes.

Following key features are supported:

- This chip supports only MII/RMII interfaces
- 4 bit MII interface operating at 2.5/25/50Mhz, MII can run at 10/100/200 Mbps.
- 2 bit RMII interface operating at 50Mhz, RMII can run at 10/100Mbps
- 4 bit MII-Lite interface operating at 2.5/25Mhz
- 4 PPS are supported using DMA trigger
- 200Mbps data rate is supported on Ethernet MII interface
- MTL Receive FIFO size 8192bytes
- MTL Transmit FIFO size 8192bytes

**NOTE**

The DMA referred within EMAC chapter refers to the internal EMAC DMA engine and not the device AXBS master DMA.

**NOTE**

Register access beyond 8k address space is not supported and will generate a transfer error.

**NOTE**

Always use store and forward mode if possible. In cut through mode, if multiple masters are active then there is chance (even in round robin arbitration mode) that EMAC DMA may starve for data resulting in insufficient data in TX FIFO leading to Tx underflow. To avoid this, increase MTL_TxQ0_Operation_Mode[TTC] value as needed.

The table below provides the information on the specific configuration utilized for the MCXE31B devices.

Table 474. Specific configuration information

| | | |
|---|---|---|
| **Application Interface** | | |
| | Application interface configuration | EQOS-AHB |
| | Data width | 32 |
| | Endian mode | Little Endian |
| | Address width | 32 |
| | CSR interfcae | APB3 interface |
| **General feature** | | |
| | Mode of operation | 10/100 |
| | Enable double VLAN processing | Yes |
| | Enable Que/channel based VLAN tag inertion on Tx | Yes |
| | Enable SA and VLAN insertion on Tx | Enabled by default |
| **Buffer managemnet** | | |
| | MTL Receive FIFO size in bytes | 8192 |
| | MTL Receive FIFO size in bytes | 8192 |
| | Enable debug memory access | Yes |
| | Use single port memory | Yes |
| **PHY interface** | | |
| | Enable RGMII, RMII only | Enable RMII, GMII/MII default |
| | Enabled signel phy interface | No |
| | Enable MDIO interface | Yes |
| **Filtering** | | |
| | Enable Additional 1-31 MAC address registers | 2 |
| | Enable Address Filter Hash Table | Yes |
| | Hash Table Size | 64 |
| | Enable VLAN Hash table based filtering | Yes |
| | Extended Rx VLAN Filter Enable | Yes |
| | Number of VLAN tag Filters | 4 |
| | Enable Layer3 and Layer4 Packet Filter | Yes |
| | Number of layer3 and Layer 4 packet filters | 4 |
| | Enable Flexible Programmable Receive Parser | Yes |

*Table continues on the next page...*

Table 474. Specific configuration information (continued)

| | | |
|---|---|---|
| | Maximum Packet Header Size for Parsing | 128 |
| | Maximum Entries of Parser Look up table | 64 |
| | Packet duplication support | Yes |
| **IEEE 1588 timestamp** | | |
| | Enable IEEE 1588 Time stamp support | Enabled by default |
| | IEEE 1588 System Time Support | Both option |
| | Add IEEE 1588 Higher word register | Yes |
| | Enable IEEE 1588 sub nanoseconds time stamp support | Yes |
| | Add IEEE 1588 Auxilary snapshot | No |
| | Enable Flexible Pulse Per Seconds Output | Yes |
| | Number of pulse per Second outputs | 4 |
| | Enable PTP Timestamp offload feature | No |
| | Enable One Step timestamp for PTP over UDP/IP feature | No |
| | Enable One Step timestamp feature | Yes |
| **Multiple queue support** | | |
| | Number of transmit Queue | 2 |
| | Number of recieve Queue | 2 |
| | Enable data center Bridging | No |
| | Enable Audo Vedio Bridging | Yes |
| | Enable support for AV in Tx queue 1 | No |
| | Number of DMA Channel on the receive side | 2 |
| **Time sensitive networking** | | |
| | Enable Enhancement to Schedule traffic(EST) | Yes |
| | Width of Time Interval Field in the Gate Control List | 24 |
| | Depth of Gate Control List | 256 |
| | Enable Frame Pre-emption support | Yes |
| **Time based scheduling** | | |
| | Enable time based Scheduling | Yes |
| **TCP/IP offloading features** | | |

*Table continues on the next page...*

Table 474. Specific configuration information (continued)

| | | |
|---|---|---|
| | Enable receive TCP/IP Checksum Check | Enabled by default |
| | Enable transmit TCP/IP Checksum offload | Enabled by default |
| | Enable support for Tx COE in Tx queue 0 | Yes |
| | Enable support for Tx COE in Tx queue 1 | No |
| | Enable TCP Segmentation offloading for TCP/IP | No |
| | Enable Split Header Feature | No |
| | Enable IPv4 ARP offload | No |
| **RMON counters** | | |
| | Enable MAC Management Counters(MMC) | Yes |
| | Enable the MAC Management Counters for Reciever TCP/IP Checksum offload | No |

### 72.1.2 Impact of MAC flush commands on MAC_PPSx_Target_Time_Seconds register

If the software discards the media clock generation request by configuring the PPSCTRL_PPSCMD (or PPSCMD#i) field of MAC_PPS_Control register to 0, the MAC does not flush the captured presentation time, i.e., the MAC_PPSx_Target_Time_Seconds register contents don't get changed.

## 72.2 Overview

This section and all its sub-sections are Synopsys Proprietary. Used with permission.

EMAC:

- Supports the 10/100 Mbps application in compliance with IEEE802.3-2015 specifications
- Can support advanced applications such as time-sensitive networking and AVB
- Supports a MAC core that has prominent features such as a flexible receive parser, media clock recovery and generation, and safety

### 72.2.1 Features

EMAC supports these features in addition to the default feature defined in the IEEE802.3 specification:

- MII (10/100 Mbps), RMII (10/100 Mbps)
- Time-aware shaper (IEEE802.1bv), time synchronization (IEEE802.1AS-rev), and frame preemption (IEEE802.1Qbu) for time-sensitive networking
- Media clock recovery and generation for AVB
- AMBA 2.0 for AHB master and APB3 interface
- RMII specification version 1.2 from RMII consortium
- Separate interface for transmit, receive, and control paths
- Two-buffer ring
- Broadcast and multicast packet duplication
- Full-duplex flow control operations (IEEE 802.3x pause packet and priority flow control)

- Network statistics with MAC management (RMON) counters
- Flexibility to control the pulse per second (PPS) output signal
- MDIO clause 22 and clause 45 interface for configuration and management of the PHY device
- Preamble and SFD insertion and deletion

**NOTE**

Only byte-aligned preamble is supported.

- Automatic CRC and pad generation/stripping option
- Option to disable CRC checking
- Programmable insert packet gap
- Source address insertion or replacement
- VLAN insertion, replacement, and deletion in transmitted packets with per-packet or static-global control, insertion, replacement, and deletion of up to two VLAN tags, insertion, replacement, and deletion of queue or channel-based VLAN tags
- IEEE802.1Q VLAN tag detection and an option to delete the tags in the receive packets
- Programmable safety watchdog timeout limits
- Flexible address filtering modes that allow:
  - Up to two additional 48-bit perfect DA filters with masks for each byte
  - Up to two 48-bit SA comparison checks with masks for each byte
  - 64-bit hash filter for multicast and unicast DA addresses
  - Option to pass all multi-cast addressed packets
  - Promiscuous mode to pass all packets without any filtering for network monitoring
  - Passing of all incoming packets (according to filter) with a status report
- Additional packet filtering that is based on:
  - Virtual local area network (VLAN) tag—perfect match and hash-based filtering, which is based on either outer or inner VLAN tag, is possible
  - Layer 3 and layer 4—TCP or UDP over IPv4 or IPv6
  - Extended VLAN tag—filtering based on four filter selections

**NOTE**

The DCB feature discussed in the following section(s) is not supported.

### 72.2.2 Block diagram

This figure shows the EMAC block diagram. EMAC has four main blocks to perform all the functions.

- The AHB interface is connected to all DMA channels.
- The DMA arbiter helps in the arbitration of all paths (transmit and receive) in DMA channels.
- Each channel has a separate set of control and status registers (CSRs) for managing the transmit and receive functions, descriptor handling, and interrupt handling.

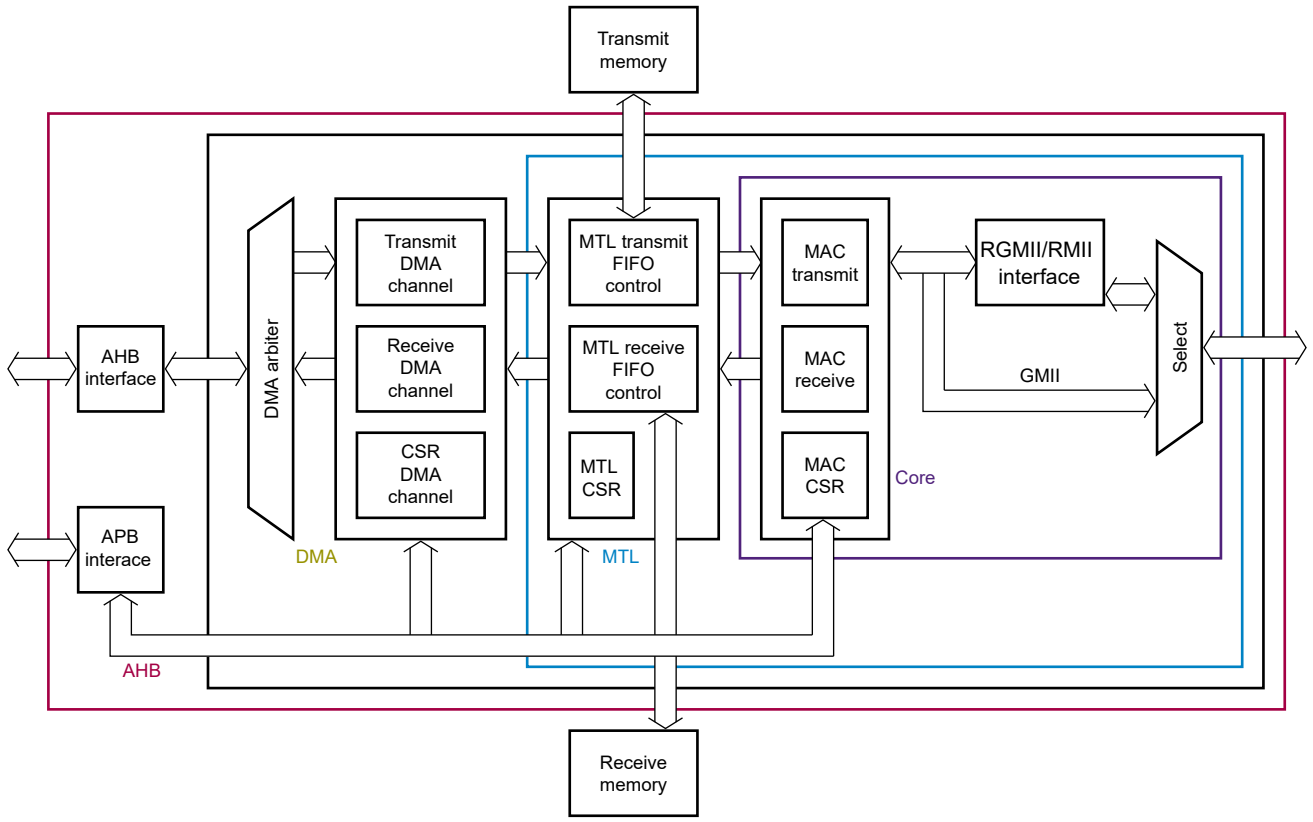



Figure 388. System-level block diagram

## 72.3 Architecture

This section and all its sub-sections are Synopsys Proprietary. Used with permission.

The section describes EMAC interfaces, protocols, and functionality. These are its main blocks:

- AHB master and APB3 slave interface
- DMA controller
- MTL
- MAC
- Interrupts

### 72.3.1 AHB master and CSR slave (APB3) interface

The 32-bit APB3 slave interface:

- Provides access to DMA, MTL, and MAC control and status registers (CSRs)
- Supports the 32-bit write and read transfers to these CSRs
- Supports single- and all-burst transfers, and an OKAY response

The DMA block uses the AHB master interface to interact with the external world. The AHB master interface:

- Controls data transfer
- Is AMBA 2.0-compliant with no restrictions
- Supports 32-bit data transfer

- Supports single and INCR transfers
- Supports burst-length modes such as Fixed, Unspecified, and Mixed

### 72.3.2 DMA inclusions

DMA controller:

- Includes independent transmit (Tx) and receive (Rx) engines, and a CSR space. The transmit engine transfers data from the system memory to the MTL interface, whereas the receive engine transfers data from the MTL interface to the system memory.
- Is designed for packet-oriented data transfers such as packets in Ethernet.
- Communicates with the host through CSR, descriptor lists, and data buffers.

**DMA descriptors**

DMA supports up to two transmit and two receive descriptor lists (or DMA channels). The base address of each list is written to the respective transmit and receive descriptor list address registers. A descriptor list is forward linked and the next descriptor is always considered at a fixed offset to the current one, and DMA_CH0_Control[DSL] controls the offset. The number of descriptors in the list is programmed using the respective DMA Channel 0 Tx Descriptor Ring Length (DMA_CH0_TxDesc_Ring_Length) and DMA Channel 0 Rx Descriptor Ring Length (DMA_CH0_RxDesc_Ring_Length) registers. After the DMA processes the last descriptor in the list, it automatically jumps to the descriptor in the list address register to create a descriptor ring.

EMAC supports the ring structure for DMA descriptors. For more information, see Descriptors .

The DMA engine uses descriptors to efficiently move data from source to destination with minimal host intervention, and the DMA controller can be programmed to interrupt the host in certain situations. These situations may include packet transmit and receive transfer completion and other normal or error conditions.

The descriptor lists reside in the physical memory address space of the application. Each descriptor can point to a maximum of two buffers in the system memory, enabling two buffers to be used and physically addressed rather than contiguous buffers in the memory.

**Data buffers**

A data buffer resides in the application physical memory space and consists of an entire packet or part of a packet but cannot exceed a single packet. Buffers contain only data. The buffer status is maintained in the descriptor. Data chaining refers to packets that span multiple data buffers. However, a single descriptor cannot span multiple packets. The DMA skips to the data buffer of the next packet when an EOP is detected.

The next few sections discuss DMA controller.

#### 72.3.2.1 DMA controller bus burst access

DMA engines attempt to transfer data in a burst of the maximum size programmed using DMA_CH0_Tx_Control[TxPBL] and DMA_CH0_Rx_Control[RxPBL] of the respective DMAs, which always access the receive and transmit descriptors in the maximum possible (limited by PBL or 16 * 8/bus width) burst length for 16 bytes to be read. The burst transfers that the DMA initiates can be split into multiple burst transfers according to the AHB requirements and settings specified in DMA System Bus Mode (DMA_SysBus_Mode).

The transmit DMA initiates a data transfer only when sufficient space is available in the MTL transmit queue to accommodate either of these:

- Bytes corresponding to the configured burst (PBL * bus_width/8)
- Remaining bytes in the transmit buffer without EOP
- Number of bytes till EOP

The receive DMA initiates a data transfer in these conditions:

- Sufficient data is available in the MTL receive queue to accommodate the configured burst
- EOP (when it is less than the configured burst length) is detected in the receive queue

DMA indicates the start address and the number of transfers required to the AHB master interface. When the interface is configured for fixed-length burst, it transfers data by using the best combination of the INCR4, INCR8, or INCR16 and SINGLE transactions. If EOP is reached before the fixed burst ends on the AHB interface, DMA performs dummy transfers to complete the fixed burst. Otherwise, DMA_SysBus_Mode[FB] becomes 0. The DMA transfers the data using undefined length (INCR) and SINGLE transactions.

When the AHB interface is configured for address-aligned beats, both DMA engines ensure that the first-burst transfer that AHB initiates is less than or equal to the size of the configured PBL value, and the subsequent beats start at an address aligned to this value. The DMA can only align the address for beats up to size 16 (for PBL > 16) for the AHB interface because it does not support more than INCR16.

#### 72.3.2.2 DMA application data buffer alignment

The transmit and receive data buffers do not have any restrictions on start address alignment. For example, in systems with 32-bit memory, the start address for buffers can be aligned to any of the four bytes. However, the DMA always initiates write transfers with an address aligned to the bus width and dummy data (old data) in the invalid byte lanes. The process of write transfer typically happens during the beginning or end of an Ethernet packet transfer. The software driver must discard the dummy bytes based on the start address of the buffer and size of the packet.

**Table 475. Data buffer alignment examples**

| Examples | |
|---|---|
| Buffer read | If the transmit buffer address is FF2h (for a 32-bit data bus) and 15 bytes need to be transferred, the DMA reads five full words from address FF0h. However, when DMA transfers data to the MTL transmit queue, the extra bytes (the first two bytes) are dropped or ignored. Similarly, the last 3 bytes of the last transfer are also ignored. The DMA always ensures that it transfers a full 32-bit data to the MTL transmit queue, unless it is the end of the packet. |
| Buffer write | If the receive buffer address is FF2h (for a 64-bit data bus) and 16 bytes of a received packet need to be transferred, the DMA writes 3 full words from address FF0h. However, the first 2 bytes of the first transfer and the last 6 bytes of the third transfer have dummy data. The DMA considers the offset address only if it is the first receive buffer of the packet. It ignores the offset address and performs full-word writes for the middle and last receive buffers of the packet. |

#### 72.3.2.3 DMA buffer size calculations

DMA does not update the size fields in the transmit and receive descriptors. It updates only the status fields (RDES and TDES) of the descriptors. The driver performs the size calculations.

The transmit DMA transfers the exact number of bytes (as the buffer size field of TDES2 indicates) to MAC. If a descriptor is marked as first (TDES3[FD] = 1), the DMA marks the first transfer from the buffer as SOP, and if the descriptor is marked as last (TDES3[LD]), the DMA marks the last transfer from that data buffer as EOP to the MTL.

The receive DMA transfers data to a buffer until the buffer is full or MTL sends the end of packet. When TDES3[FD] = 1, the amount of valid data in a buffer is accurately indicated by the buffer size fields in DMA Channel Rx Control (DMA_CH0_Rx_Control) minus the data buffer pointer offset. The offset is 0 when the data buffer pointer is aligned to the data bus width. If a descriptor is marked as last, the buffer may not be full (as indicated by DMA_CH0_Rx_Control[RBSZ_13_y] and DMA_CH0_Rx_Control[RBSZ_x_0]). To compute the amount of valid data in this final buffer, the driver must read the packet length (PL fields of RDES3[14:0]) and subtract the sum of the buffer sizes of the preceding buffers in this packet. The receive DMA always transfers the start of the next packet with a new descriptor.

#### 72.3.2.4 DMA arbiter

The arbiter inside the DMA module performs the arbitration between the transmit and receive channel accesses to the AHB master interface. These two types of arbitrations are supported:

- Round-robin: If DMA_Mode[DA] = 0 and both transmit and receive DMAs simultaneously request access, the arbiter allocates the data bus in ratio sets defined in DMA_Mode[PR].
- Fixed-priority: If DMA_Mode[DA] = 1, receive DMA is always prioritized over transmit DMA for data access by default. If DMA_Mode[TXPR] = 1, transmit DMA is prioritized over receive DMA as explained in Table 492.

### 72.3.3 MTL

MTL provides the FIFO memory interface to buffer and regulate the packets between system memory and MAC. It also enables data to be transferred between the system clock and MAC clock domains. The MTL layer has two data paths: transmit path and receive path. MTL communicates with the host through ATI on the transmit path and through ARI on the receive path.

#### 72.3.3.1 Transmit path

The internal DMA:

- Handles all transactions for the transmit path through ATI.
- Pushes the Ethernet packet reads from system memory to the corresponding queue.

The Ethernet packet, then, pops out and is transferred to MAC when the queue reaches its threshold (Threshold mode) or if the complete packet is in the queue (Store-and-Forward mode). When EOP is transferred, the status of the transmission is taken from MAC and transferred back to the internal DMA.

##### 72.3.3.1.1 Transmit control word

This control information related to packet transmission is provided as part of the control word through the ATI interface:

- Packet length (if DCB is enabled with WFQ scheduling algorithm)
- CRC pad control
- Source address insertion control
- VLAN insertion and replacement control and VLAN tag for outer and inner VLAN tags
- TCP/IP checksum insertion control
- One-step timestamping control correction
- Transmit timestamp enable

##### 72.3.3.1.2 Transmit operation

These two modes of operation trigger data read towards MAC:

- Threshold (or cut-through) mode: This is the default mode. In this mode, as soon as the number of bytes in the queue crosses the configured threshold level (or when the end of packet is written before the threshold is crossed), the data is ready to be popped out and forwarded to MAC. The threshold level is configured by using MTL_TxQ0_Operation_Mode[TTC].
- Store-and-Forward mode: In this mode, MTL pops the packet out to MAC only when one or more of the following conditions are true:
  - A complete packet is stored in the queue.
  - The transmit FIFO becomes almost full.
  - The ATI watermark becomes low.

  The watermark becomes low when the requested queue is not spacious enough to accommodate the requested burst length on ATI. Therefore, MTL, when operating in Store-and-Forward mode, allows packet transmission even if the packet length is bigger than the transmit queue size.

You can flush the complete content of the transmit queue by writing 1 to MTL_TxQ0_Operation_Mode[FTQ] or MTL_TxQ1_Operation_Mode[FTQ], depending on the queue. Doing so initializes the queue pointers to the default state. The FTQ

field returns to 0 by itself. If you write 1 to the FTQ field during a packet transfer from MTL to MAC, MTL stops further transfers because MTL considers the queue to be empty. Therefore, an underflow event occurs at the MAC transmitter.

#### 72.3.3.1.3 Initialization flow

After reset, MTL is ready to manage the data flow between DMA and MAC.

- With single-transmit queue configurations, there are no initialization requirements for enabling MTL.
- With multiple-transmit queue configurations, initialize the queue size for each of the queues by programming MTL_TxQ0_Operation_Mode[TQS] corresponding to a transmit queue. Also, initialize the MAC block. Internal DMA controllers must be individually enabled through their respective CSRs.

### 72.3.3.2 Receive path

MAC sends packets to the MTL receive module and pushes them into the receive queue. MTL indicates the status (fill level) of the queue to the application or DMA when it crosses the configured receive threshold (MTL_RxQ0_Operation_Mode[RTC]), or when the MTL receive module receives the complete packet in Store-and-Forward mode. MTL also indicates the fill level of the queue so that DMA can initiate preconfigured burst transfers to the AHB interface.

#### 72.3.3.2.1 Receive operation

During a receive operation, MTL is MAC's slave. This is the general sequence of events:

1. When MAC receives a packet, it indicates the availability of receive data.
2. MAC indicates the SOP and EOP delimiters.
3. MTL accepts the data and pushes it into the corresponding receive queue.
4. When MAC transfers EOP to the MTL receive module, MAC also drives the status word that MTL pushes into the corresponding receive queue.
5. MTL takes the data out of the queue and sends it to DMA depending on these modes:
   - Threshold mode
   - Store-and-Forward mode

   The sections that follow discuss these modes.

##### 72.3.3.2.1.1 Threshold mode

In Threshold (default) mode, MTL reads the data and indicates its availability to the application or DMA when one of these occurs:

- Data bytes equal to the threshold amount are written to the receive queue (to MTL_RxQ0_Operation_Mode[RTC] and MTL_RxQ1_Operation_Mode[RTC]).
- A full packet of data is received into the queue.

##### 72.3.3.2.1.2 Store-and-Forward mode

In this mode (when MTL_RxQ0_Operation_Mode[RSF] = 1), the initial receive queue locations are reserved for the status words before writing the SOP. A packet is read out only after it is completely written into the receive queue. In this mode, all error packets are dropped (if configured through MTL_RxQ0_Operation_Mode[FEP]) in such a way that only valid packets are read and forwarded to the application.

#### 72.3.3.2.2 Multi-packet receive operation

In Threshold mode, the packet status is available immediately after the packet data. In Store-and-Forward mode, the packet data is available after the packet status. MTL is capable of storing any number of packets in the queue as long as it is not full.

If MAC receives a packet when the corresponding receive queue is full, MTL ignores that packet and overflow pulse generates on the corresponding receive queue. In addition, MTL increments the overflow counter in MTL Rx Queue Missed Packet Overflow Count (MTL_RxQ0_Missed_Packet_Overflow_Cnt) for the corresponding queue.

#### 72.3.3.2.3 Error handling in receive operation

MTL performs these actions if the MTL receive queue is full before it receives the EOP data from MAC:

1. Declares an overflow
2. Drops the whole packet (including the status word)
3. Increments the overflow counter in DMA (MTL Rx Queue Missed Packet Overflow Count (MTL_RxQ0_Missed_Packet_Overflow_Cnt))

This is true even if MTL_RxQ0_Operation_Mode[FEP] is 1.

If the start address of such a packet is already transferred to the read controller, the rest of the packet is dropped and a dummy EOP is written to the queue along with the status word with overflow status. The status indicates a partial packet because of overflow. In such packets, the Packet Length field is invalid. If the MTL receive queue is configured to operate in the Store-and-Forward mode and the length of the received packet is more than the queue size, overflow occurs and all such packets are dropped.

The MTL receive control logic can filter errors and undersized packets, if enabled by using MTL_RxQ0_Operation_Mode[FEP] and MTL_RxQ0_Operation_Mode[FUP]. If the start address of one such packet is already transferred to the receive queue read controller, the packet is not filtered. The start address of the packet is transferred to the read controller after the packet crosses the receive threshold defined using MTL_RxQ0_Operation_Mode[RTC].

If the MTL receive queue is configured to operate in Store-and-Forward mode, all error packets can be filtered and dropped. For the application or DMA to flush the error packet being read from the queue, it must assert the flush signal. MTL then stops transferring data to the application (DMA). It internally reads the rest of the packet and drops it. MTL then starts transferring the next packet, if it's available.

### 72.3.4 MAC

MAC can support the MII and RMII PHY interface and it consists of:

- MTI
- MRI
- MCI

#### 72.3.4.1 MAC transmission

The MAC transmission process is as follows:

1. The transmission initiates when MTL pushes in data with the SOP signal asserted.
2. After the SOP signal is detected, MAC accepts the data and begins transmitting to RMII or MII.
3. After the EOP signal is transferred to MAC, it performs one of these steps:
   - Completes normal packet transmission and sends the transmission status to MTL
   - Sends retry requests if a normal collision (in Half-Duplex mode) occurs during transmission, and until one of the following is true:
     - Packet is successfully transmitted.
     - Maximum retry requests have expired. When this happens, MAC aborts the packet transmission with "excessive collision transmit" status. MAC accepts and drops all further data until it receives the next SOP. The MTL block must retransmit the same packet from SOP on observing a retry request (in the status) from MAC.
   - If any one of the following happens, MAC aborts the packet transmission:

- No carrier (Half-Duplex mode)
- Loss of carrier (Half-Duplex mode)
- Excessive deferral (Half-Duplex mode)
- Late collisions (Half-Duplex mode)
- Jabber
- MAC accepts and drops all further data until it receives the next SOP

4. MAC issues an underflow status if MTL is not able to provide data continuously during transmission. Until the next SOP is received, MAC accepts and drops all further data.
5. During the normal transfer of a packet from MTL, if MAC receives an SOP without getting an EOP for the previous packet, it ignores the SOP and considers the new packet as a continuation of the previous packet.

#### 72.3.4.2 MAC reception

The receive operation initiates as follows:

1. MAC detects a state-of-frame data on RMII or MII.
2. MAC strips the preamble and SFD before processing an Ethernet packet.
3. The MAC AFM:
   - Checks the header fields (SA and DA) of an incoming packet for filtering
   - Verifies the CRC contained in the packet's FCS field
4. MAC's AFM checks the header fields (SA and DA) of an incoming packet for filtering.
5. FCS verifies the CRC of the received packet.
6. MAC stores the received packet in a shallow buffer until address filtering is performed.
7. AFM drops the packet if it fails the address filter.

### 72.3.5 Interrupts

EMAC supports interrupt coalescing, which reduces the number of interrupts generated by the module and reduces the CPU load. MAC Interrupt Status (MAC_Interrupt_Status) captures various interrupt events. If the interrupt enable field of an interrupt is 1, the corresponding interrupt generates based on the event status in the Status registers. The Interrupt mode (INTM) field decides whether the interrupt signal is a level signal or pulse signal. See the following table and interrupt map file attached to this document for details.

**Table 476. Interrupt request**

| Interrupt request | Interrupt Enable / Mask Register | Interrupt Flag Register |
|---|---|---|
| Common interrupt | MAC Interrupt Enable (MAC_Interrupt_Enable) | MAC Interrupt Status (MAC_Interrupt_Status) |
| | MMC Receive Interrupt Mask (MMC_Rx_Interrupt_Mask) | MMC Receive Interrupt (MMC_Rx_Interrupt) |
| | MMC Transmit Interrupt Mask (MMC_Tx_Interrupt_Mask) | MMC Transmit Interrupt (MMC_Tx_Interrupt) |
| | MMC FPE Transmit Interrupt Mask (MMC_FPE_Tx_Interrupt_Mask) | MMC Transmit FPE Fragment Counter Interrupt Status (MMC_FPE_Tx_Interrupt) |

*Table continues on the next page...*

Table 476. Interrupt request (continued)

| Interrupt request | Interrupt Enable / Mask Register | Interrupt Flag Register |
|---|---|---|
| | MMC FPE Receive Interrupt Mask (MMC_FPE_Rx_Interrupt_Mask) | MMC Receive Packet Assembly Error Counter Interrupt Status (MMC_FPE_Rx_Interrupt) |
| | MTL Debug Control (MTL_DBG_CTL) | MTL Debug Status (MTL_DBG_STS) |
| | MTL EST Interrupt Enable (MTL_EST_Intr_Enable) | MTL EST Status (MTL_EST_Status) |
| | MTL Rx Parser Interrupt Control Status (MTL_RXP_Interrupt_Control_Status) | MTL Rx Parser Interrupt Control Status (MTL_RXP_Interrupt_Control_Status) |
| | MTL Queue 0 Interrupt Control Status (MTL_Q0_Interrupt_Control_Status) | MTL Queue 0 Interrupt Control Status (MTL_Q0_Interrupt_Control_Status) |
| | MTL Queue 1 Interrupt Control Status (MTL_Q1_Interrupt_Control_Status) | MTL Queue 1 Interrupt Control Status (MTL_Q1_Interrupt_Control_Status) |
| | DMA Channel 0 Interrupt Enable (DMA_CH0_Interrupt_Enable)(transfer complete interrupts only) | DMA_CH0_Interrupt_Status |
| | DMA Channel 1 Interrupt Enable (DMA_CH1_Interrupt_Enable)(transfer complete interrupts only) | DMA_CH1_Interrupt_Status |
| | | DMA Interrupt Status (DMA_Interrupt_Status)<br>(First level status. Indicates major block event source: DMA channel, MTL or MAC) |
| | | MTL Interrupt Status (MTL_Interrupt_Status)<br>(Status summary for MTL block interrupts: Rx Parser, EST, FIFO Debug, Queue 1/Queue 0) |
| Tx interrupt 0/1 | DMA Channel Tx Control (DMA_CH0_Tx_Control) | DMA Channel 0 Status (DMA_CH0_Status) |
| | DMA Channel 0 Interrupt Enable (DMA_CH0_Interrupt_Enable) | |
| | DMA Channel 1 Tx Control (DMA_CH1_Tx_Control) | DMA Channel 1 Status (DMA_CH1_Status) |
| | DMA Channel 1 Interrupt Enable (DMA_CH1_Interrupt_Enable) | |
| Rx interrupt 0/1 | DMA Channel Rx Control (DMA_CH0_Rx_Control) | DMA Channel 0 Status (DMA_CH0_Status) |
| | DMA Channel 0 Interrupt Enable (DMA_CH0_Interrupt_Enable) | |
| | DMA Channel 1 Rx Control (DMA_CH1_Rx_Control) | DMA Channel 1 Status (DMA_CH1_Status) |
| | DMA Channel 1 Interrupt Enable (DMA_CH1_Interrupt_Enable) | |

*Table continues on the next page...*

Table 476. Interrupt request (continued)

| Interrupt request | Interrupt Enable / Mask Register | Interrupt Flag Register |
|---|---|---|
| Safety interrupt | MTL ECC Interrupt Enable (MTL_ECC_Interrupt_Enable) | MTL ECC Interrupt Status (MTL_ECC_Interrupt_Status) |
| | | MTL Safety Interrupt Status (MTL_Safety_Interrupt_Status) |
| | MTL DPP Control (MTL_DPP_Control) | MAC DPP FSM Interrupt Status (MAC_DPP_FSM_Interrupt_Status) |
| | MAC FSM Control (MAC_FSM_Control) | |

These are the interrupt output signals that are synchronous to the CSR clock.

- Sbd_intr_o – common interrupt
- Sbd_perch_tx_intr_o[max_dma_ch] – interrupt per transmit channel
- Sbd_perch_rx_intr_o [max_dma_ch] – interrupt per receive channel

**NOTE**

max_dma_ch = number of transmit/receive queues, which is 2

The sbd_intr_o common interrupt is a level signal. When it is asserted, a corresponding interrupt event source can be found in DMA Interrupt Status (DMA_Interrupt_Status), which is a read-only register that contains the event source fields corresponding to each DMA channel (transmit and receive queue pair), MAC transaction layer, and MAC blocks. You must then read the following registers and look for the fields that are 1:

- MAC Interrupt Status (MAC_Interrupt_Status)
- MTL Interrupt Status (MTL_Interrupt_Status)
- DMA Channel 0 Status (DMA_CH0_Status)

The sbd_intr_o interrupt deasserts only when all the enabled interrupt events are clear in their respective status registers, and correspondingly, all the fields in DMA Interrupt Status (DMA_Interrupt_Status) are 0. DMA Channel 0 Status (DMA_CH0_Status) captures all the interrupt events of that transmit DMA and receive DMA channel pair.

DMA Channel 0 Interrupt Enable (DMA_CH0_Interrupt_Enable) contains the corresponding enable fields for each of the interrupt events. These are the two groups of interrupts in the DMA channel:

- Normal—DMA_CH0_Status[NIS] indicates this interrupt, which is used for events that occur during the normal transfer of packets (TI, RI, TBU).
- Abnormal—DMA_CH0_Status[AIS] indicates this interrupt, which is used for error events.

Interrupts are not queued. If the same interrupt event occurs again before the driver responds to the previous one, no additional interrupts generate.

The common sbd_intr_o output signal asserts for the transfer complete interrupts only when the corresponding interrupts are enabled in DMA Channel 0 Interrupt Enable (DMA_CH0_Interrupt_Enable).

EMAC also supports these per-channel transfer-complete interrupt signals:

- sbd_perch_tx_intr_o[max_dma_ch] (transmit per channel interrupts)
- sbd_perch_rx_intr_o[max_dma_ch] (receive per channel interrupts)

The behavior of the RI/TI/sbd_perch_tx_intr_o[]/sbd_perch_rx_intr_o[] changes depends on the configuration specified in DMA_Mode[INTM]. This table explains the transfer complete interrupt behavior.

Table 477. Transfer complete interrupt behavior

| Interrupt mode | Behavior of sbd_perch_tx_intr_o[] and sbd_perch_rx_intr_o[] | Behavior of the TI/RI interrupts and sbd_intr_o |
|---|---|---|
| INTM = 1 | In this mode, these signals indicate the values of the corresponding DMA_CH0_Status[RI] and DMA_CH0_Status[TI] fields when DMA_CH0_Status[RI] and DMA_CH0_Status[TI] are 1, respectively. Therefore, they are level signals that you clear by writing 1 to these fields. The signals do not assert when DMA_CH0_Status[RI] or DMA_CH0_Status[TI] is 0. | DMA_CH0_Status[RI] and DMA_CH0_Status[TI] are configured as explained.<br>For any RI/TI events:<br>• The sbd_intr_o signal does not assert.<br>• DMA_CH0_Status[NIS] remains 0. |
| INTM = 2 | In this mode, RI/TI are queued and indicate the values of the corresponding DMA_CH0_Status[RI] and DMA_CH0_Status[TI] fields when any of these fields is 1. RI and TI are level signals that you clear by writing 1 to these fields. However, the fields become 1 again if another TI/RI event is detected before DMA_CH0_Status[RI] and DMA_CH0_Status[TI] become 1 for the previous event. | DMA_CH0_Status[RI] and DMA_CH0_Status[TI] become 1 whenever the transfer complete event is detected and are reset whenever the software driver changes them to 0 by writing 1. However, if another transfer complete event is detected before the software changes it to 0 (services it), EMAC automatically writes 1 to these fields again. The generation of the sbd_intr_o signal; however, is not based on DMA_CH0_Status[RI] or DMA_CH0_Status[TI]. |

## 72.4 External signals

This section and all its sub-sections are Synopsys Proprietary. Used with permission.

### 72.4.1 Module signals

This table provides port/signal names and their descriptions. See the IOMUX file attached to this document for details.

Table 478. Module signals

| Port name (clock/signal) | I/O | Description |
|---|---|---|
| MII_RMII_TXCLK (Clock) | I | MII: The external PHY provides this transmission clock, which operates at a frequency of 25 MHz in 100 Mbps mode and at 2.5 MHz in 10 Mbps mode. All transmission signals that MAC generates are synchronous to this clock, which is required for all PHY interfaces. |

*Table continues on the next page...*

**Table 478. Module signals (continued)**

| Port name (clock/signal) | I/O | Description |
|---|---|---|
| | | RMII: The RMII interface uses this 50 MHz clock. If you select RMII mode, MII_RX_CLK (25 MHz or 2.5 MHz) must be derived from the RMII reference clock. |
| MII_RX_CLK (Clock) | I | The external PHY provides this receive clock for the MII and RMII interfaces. The clock operates at a frequency of 25 MHz in 100 Mbps mode and at 2.5 MHz in 10 Mbps mode. All MII receive signals that MAC receives are synchronous to MII_RX_CLK. The clock's input is required for all PHY interfaces. |
| EMAC_PPS[3:0] (Signals) | I/O | This group of signals is used as pulse per second in Output mode and as media clock generation trigger in Input mode. The signals trigger input to DUT to capture presentation time. Based on the presentation control value of MAC PPS Control (MAC_PPS_Control), the signals can be defined as pulse or level signals. |
| MII_RMII_TX_EN (Signal) | O | MAC drives this signal, which performs multiple functions depending on the selected PHY interface as described in this list:<br>• MII: When high, indicates that valid data is being transmitted to the PHY_TXD_O bus. MII_RMII_TX_EN is synchronous to MII_RMII_TX_CLK.<br>• RMII: When high, indicates that valid data is being transmitted to the PHY_TXD_O bus. MII_RMII_TX_EN is synchronous to MII_RMII_RX_CLK. |
| MII_RMII_TXD[3:0] (Signals) | O | This is a group of transmit data signals driven by MAC. These signals perform multiple functions depending on the selected PHY interface as described in this list:<br>• MII: Bits[3:0] provide the MII transmit data nibble. The data is valid only when the MII_RMII_TX_EN signal is high. MII_RMII_TXD[3:0] is synchronous to MII_MII_TX_CLK. |

*Table continues on the next page...*

**Table 478. Module signals (continued)**

| Port name (clock/signal) | I/O | Description |
|---|---|---|
| | | • RMII: Bits[1:0] provide the RMII transmit data. The data is valid only when the MII_RMII_TX_EN signal is high. MII_RMII_TXD[3:0] is synchronous to MII_RMII_TX_CLK. |
| MII_CRS (Signal) | I | This signal is valid only in MII mode. The PHY drives this signal high when the transmit or receive medium is not idle, and drives the signal low when both these mediums are idle. The signal is not synchronous to any clock. |
| MII_COL (Signal) | I | This signal is valid only in MII mode. The PHY drives this signal high when a collision is detected on the medium. The signal is not synchronous to any clock. |
| MII_RMII_RX_DV (Signal) | I | The PHY drives this signal. It performs multiple functions depending on the selected PHY interface as described in this list:<br>• MII: Indicates that the data on the MII_RXD bus is valid. It remains high continuously from the first recovered byte or nibble of the packet through the final recovered byte or nibble of the packet. MII_RMII_RX_DV is synchronous to MII_RX_CLK.<br>• RMII: Contains the CRS and data valid information of the receive interface. MII_RMII_RX_DV is synchronous to MII_RMII_TX_CLK. |
| MII_RMII_RX_ER (Signal) | I | The PHY drives this signal. It performs multiple functions depending on the selected PHY interface as described in this list:<br>• MII: Indicates an error or carrier extension in the received packet of the MII_RXD[3:0] bus. MII_RMII_RX_ER is synchronous to MII_RX_CLK.<br>• RMII: Is not used. |
| MII_RMII_RXD[3:0] (Signals) | I | This is a group of data signals received from the PHY. These signals perform multiple functions depending on the |

*Table continues on the next page...*

**Table 478. Module signals (continued)**

| Port name (clock/signal) | I/O | Description |
|---|---|---|
| | | selected PHY interface as described in this list:<br>• MII: Bits [3:0] provide the MII receive data nibble. The data is valid only when the MII_RMII_RX_DV signals are high. MII_RMII_RXD[3:0] is synchronous to MII_RX_CLK.<br>• RMII: Bits [1:0] provide the RMII receive data. The data is valid only when the MII_RMII_RX_DV signal is high. MII_RMII_RXD[3:0] is synchronous to MII_RMII_TX_CLK. |
| MII_RMII_MDC | O | MAC provides timing reference for MII_RMII_MDIO or MII through this periodic clock. The application clock generates this clock through a clock divider that MAC_MDIO_Address[CR] controls. |
| MII_RMII_MDIO (Signal) | I/O | MDIO uses this signal to transfer control and data information to PHY. |

## 72.5 Using PHY interfaces

This section is Synopsys Proprietary. Used with permission.

This module support the following modes:

- RMII 10/100 Mbps interface
- MII 10/100 Mbps interface

Phy_intf_sel input signal decides which mode is selected. Samples the Phy_intf_sel signal at reset. See External signals for more information. MAC_Configuration[PS] and MAC_Configuration[FES] selects the mode's speed.

The module supports the IEEE802.3-2015 specification for MII and RMII specification version 1.2 from RMII consortium.

**NOTE**

RMII reference clock (50 MHz) can be fed to IP from an external source or internally from PLL on SoC.

The module supports the access of the phy registers through SMA. It is a two wire Station management interface (MIM):

1. MDC
2. MDIO

According to IEEE 802.3 specification, maximum operating frequency of MDC is 2.5 Mhz which system clock derives using a divider. MAC_MDIO_Address[CR] programs to generate different MDC clock frequency.

SMA supports MDIO clause 45 and clause 22 frame structure per the IEEE802.3 specification. Writing 1 to MAC_MDIO_Address[C45E] enables the clause 45 frame structure.

The MII interface reduces the accuracy of the 1588 timestamp if it is overclocked to run at 200 Mbps. This happens because the MAC logic uses the MII interface speed,to adjust or compensate for the timestamps taken at MII, as compared to the time generated in the PTP clock domain as specified in MAC_Configuration[FES].

## 72.6 VLAN and double VLAN insertion, deletion, replacement and tagging

This section and all its sub-sections are Synopsys proprietary. These are used with permission.

The following sections describe the VLAN and double VLAN features.

### 72.6.1 Double VLAN processing

In the double VLAN tagging processing, the module supports the processing of two VLAN tags. The two VLAN tags are:

1. Inner VLAN tag (C-VLAN)
2. Outer VLAN tag (S-VLAN)

If there is only one tag in a packet then it is considered as an outer tag.

The module uses MAC_VLAN_Incl and MAC_Inner_VLAN_Incl and MAC_VLAN_Tag registers to support the following functions in a double VLAN processing feature.

- Insertion, replacement, or deletion of up to two VLAN tags in the transmit path.
- Packet filtering and stripping on the basis of any one of the two VLAN tags in the receive path. Stripping and providing up to two VLAN tags in the receive path as a part of the receive status.

#### 72.6.1.1 Transmit path

**Table 479. Double VLAN processing features in transmit path**

| Feature | Description |
|---|---|
| Support for C-VLAN and S-VLAN tag types | The inner or outer VLAN tag can be of C-VLAN and S-VLAN type. MAC_VLAN_Incl[CSVL] and MAC_Inner_VLAN_Incl[CSVL] specifies the VLAN type. The module supports the processing of any sequence of outer and inner VLAN tags.<br>**NOTE**<br>The module does not support the C-VLAN and S-VLAN sequence.<br>MAC does not examine whether the packet, which the application provides has a valid sequence of the VLAN tag types or the insertion or replacement operation results in an invalid sequence of VLAN tag type. Therefore, the application must provide a correct sequence of VLAN tag types and program the MAC in such a way that it results in correct sequence of VLAN tag types in the transmitted packet. The application must ensure the following:<br>• The inner tag should not be S-VLAN when outer C-VLAN Tag insertion is enabled.<br>• The outer tag should not be C-VLAN when inner S-VLAN Tag insertion is enabled.<br>• The inner tag should not be S-VLAN when C-VLAN replaces the outer tag.<br>• The outer tag should not be C-VLAN when S-VLAN replaces the inner tag. |
| VLAN tag deletion | MAC_VLAN_Incl[VLC] or MAC_Inner_VLAN_Incl[VLC] respectively enables the VLAN tag deletion for an outer or inner tag. When you enable the VLAN deletion, MAC deletes the tag present at the corresponding position. When a packet has only one tag, it is considered as the outer tag. If you enable the inner tag deletion and if there is only one tag in the packet, then the MAC does not delete the tag. |
| VLAN tag insertion or replacement | MAC_VLAN_Incl[VLC] or MAC_Inner_VLAN_Incl[VLC] respectively enables the VLAN tag insertion or replacement for an outer or inner tag. When you enable the VLAN tag insertion or replacement, the MAC_VLAN_Incl[VLTI] or<br>MAC_Inner_VLAN_Incl[VLTI] determines whether the VLAN tag must be fetched from the register or from the control word. |

#### 72.6.1.2 Receive path

Table 480. Double VLAN processing in receive path

| Feature | Description |
|---|---|
| Outer or inner VLAN tag-based filtering | Mac can filter packets through the ERIVLT bit on the basis of the outer or inner VLAN tag. |
| C-VLAN or S-VLAN tag-based filtering | Mac can filter packets through the ERSVLM bit on the basis of the C-VLAN or S-VLAN type. |
| Outer and inner VLAN tag stripping | Mac can strip the outer and inner VLAN tags from received frame on the basis of the EVLS and EIVLS bits. |
| 16-bit outer and inner VLAN tag and type in Rx status | Mac can provide the 16-bit outer and inner VLAN tag and type in the Rx status on the basis of the EVLRXS and EIVLRXS bits, respectively. |
| Disabling or skipping checking of outer VLAN tag type | Mac can disable or skip checking of an outer VLAN tag type to match C-VLAN or S-VLAN on the basis of the DOVLTC bit. |

#### 72.6.1.3 Source address and VLAN insertion, replacement, or deletion

This module supports the SA, and VLAN insertion, replacement and deletion feature. The SA and VLAN fields are Tx packet-related control information that interfaces provide as part of the control word. IP supports the feature to insert or replace the source address on the basis of the information in the MAC address registers, and it also supports the feature to insert, replace, or delete the VLAN fields (VLAN type and VLAN tag) on the basis of the setting of the MAC_VLAN_Incl[VLTI]. SA insertion or replacement feature is enabled for all the transmit packets or selective packets. Similarly, VLAN insertion, replacement, or deletion feature is enabled for all the Tx packets or selective packets.

#### 72.6.1.4 Programming source address insertion or replacement

You can use the SA insertion or replacement feature to instruct MAC to perform the following actions for Tx packets:

- Insert the content of the MAC address registers in the SA field
- Replace the content of the SA field with the content of the MAC address registers

When SA insertion is enabled, the application must ensure that the packets sent to MAC do not have the SA field. The MAC does not check whether the SA field is present in the transmit packet and it inserts the content of MAC address registers in the SA field. Similarly, when the SA replacement is enabled, the application must ensure that the SA field is present in the packets sent to the MAC.

MAC replaces the six bytes following the DA field in the transmit packet with the content of the MAC address registers.

- Configure MAC_Configuration[SARC] to enable the SA insertion or replacement for all Tx packets.
- Enable the SA insertion for selective packet by configuring the SA insertion control field (bits [25:23] of TDES3) in the first transmit descriptor of the packet. When you write 1 to bit 25 of TDES3, the SA insertion control field indicates insertion or replacement by MAC Address1 registers. When bit 25 of TDES3 resets, it indicates insertion or replacement by MAC Address 0 registers.

If you do not enable the MAC Address1 registers, then the MAC Address0 registers are used for insertion or replacement irrespective of the value of the most-significant bit of the SA insertion control field.

#### 72.6.1.5 Programming VLAN insertion, replacement, or deletion

You can use the VLAN insertion, replacement, or deletion feature to instruct MAC to perform the following actions for the Tx packets:

- Delete the VLAN type and VLAN tag fields.
- Insert or replace the VLAN type and VLAN tag fields.

Insertion or replacement is done on the basis of the setting of the MAC_VLAN_Incl[VLTI] as described in theTable 481:

**Table 481. VLAN insertion or replacement based on VLTI bit**

| Condition | Description |
|---|---|
| VLTI bit=1 | MAC inserts or replaces the following:<br>• VLAN type field (C-VLAN or S-VLAN as indicated by the MAC_VLAN_Incl[CSVL])<br>• VLAN tag field depending on the content of the VT field of transmit context descriptor of the packet |
| VLTI bit resets | MAC inserts or replaces the following:<br>• VLAN type field (C-VLAN or S-VLAN as indicated by the MAC_VLAN_Incl[CSVL])<br>• VLAN tag field with the MAC_VLAN_Incl[VLT] |

When the VLAN replacement or deletion is enabled, MAC checks whether the VLAN type field (0x8100 or 0x88a8) is present after the DA and SA fields in the transmit packet. The replace or delete operation does not occur if the VLAN type field is not detected in two bytes following the DA and SA fields. However, when the VLAN insertion is enabled, MAC does not check the presence of VLAN type field in the transmit packet and just inserts the VLAN type and VLAN tag fields.

Configuring MAC_VLAN_Incl[VLC] and MAC_VLAN_Incl[VLP] enables the VLAN insertion, replacement, or deletion feature for all Tx packets.

Configuring the VTIR field of TDES2 normal descriptor enables the VLAN insertion, replacement, or deletion for selective packets.

In addition, the VLP (VLAN Priority control) field must reset in MAC_VLAN_Incl (for outer VLAN) and MAC_Inner_VLAN_Incl register (in inner VLAN) for MAC to take the control inputs from the host, depending on the configuration.

## 72.7 Packet Filtering

This section and all its sub-sections are Synopsys proprietary. Used with permission.

MAC receiver contains various packet filtering scheme. Figure 389 shows the filtering sequence in their precedence order of received packet.

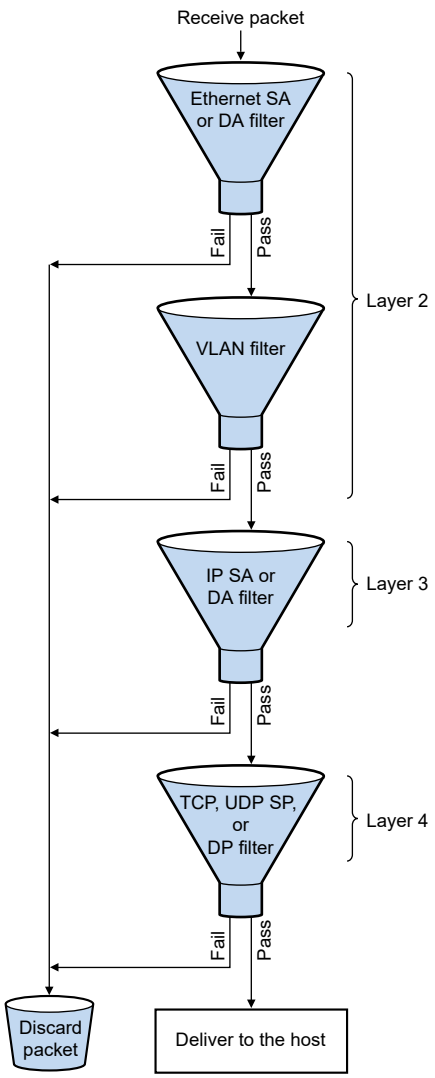



Figure 389. Packet filtering sequence

If you do not enable any of the layer filters, then that filter is bypassed and the subsequent filter is applied. Discard the packet that fails any of the filters. However, you can forward the discarded packet to the host on the basis of the register control.

## 72.7.1 Source address or destination address filtering

The address filtering module of MAC checks the SA and DA fields of each incoming packet. Programming of different types of address filtering is described in the sections that follows.

### 72.7.1.1 Unicast destination address filtering

MAC supports up to three MAC addresses for unicast perfect filtering. If perfect filtering is selected (MAC_Packet_Filter[HUC] resets), MAC compares all 48 bits of received unicast address with the programmed MAC address for any match. Default MacAddr0 is always enabled. MAC selects the MacAddr1 to MacAddr2 addresses with an individual enable field. Each byte of MacAddr1 to MacAddr2 can be masked when you compare them with corresponding received DA byte by writing 1 to the corresponding Mask byte control field in the register. This enables group address filtering for the DA.

In hash filtering mode (when you write 1 to HUC bit), MAC performs imperfect filtering by using a 64-bit hash table for unicast addresses . For hash filtering, MAC uses the upper 6 bits CRC of the received destination address to index the content of the hash table. A value of 00000 selects bit 0 of selected register, and a value of 11111 selects bit 63 of hash table register. If the value of

the corresponding bit (indicated by the 6-bit CRC) is 1, the unicast packet is considered to have passed the hash filter, otherwise, the packet is considered to have failed the hash filter.

#### 72.7.1.2 Multicast destination address filtering

Enable MAC_Packet_Filter[PM] to pass all the multicast packets or disable to do multicast addresses filtering on the basis of MAC_Packet_Filter[HMC]. Compare the multicast address with the configured MAC destination address registers (1–2), also supports the group address filtering.

In Hash filtering mode, MAC performs imperfect filtering using a 64-bit hash table. MAC uses the upper 6-bits CRC of received multicast address to index the content of the hash table. A value of 000000 selects bit 0 of selected register and a value of 111111 selects bit 63 of the hash table register. The multicast packet is considered to have passed the hash filter if the value of the corresponding bit is equal to 1. Otherwise, the packet is considered to have failed the hash filter.

#### 72.7.1.3 Hash or perfect address filtering

Configure MAC_Packet_Filter[HPF] to enable or disable the destination address filtering either by hash filter or perfect filter. Also configure MAC_Packet_Filter[HUC] and MAC_Packet_Filter[HMC] to select the hash filtering or perfect filtering for unicast or multicast packet.

#### 72.7.1.4 Broadcast address filtering

MAC does not filter any broadcast packets by default. You must write 1 to MAC_Packet_Filter[DBF] to reject the broadcast packets.

#### 72.7.1.5 Unicast source address filtering

MAC by default compares the source address field with the value configured in the source address registers. Configure bit 30 of MAC address register [1-2] to use it for source address instead of destination address comparison. MAC also supports group filtering with SA. You can filter a group of addresses by masking one or more bytes of the address. MAC drop the packets that fail the SA filter if you write 1 to MAC_Packet_Filter[SAF] field. Otherwise, the result of the SA filter is given as a status bit in the receive status word. When you write 1 to the SAF field, the SA filter and DA filter results ends to decide whether you can forward the packets. This means that the packet is dropped if either filter fails. The packet is forwarded to the application only if the packet passes both filters in-order.

#### 72.7.1.6 Inverse filtering

For DA and SA filtering, you can invert the filter-match result at the final output by writing 1 to the DAIF and SAIF fields of MAC_Packet_Filter register. The DAIF field is applicable for both the unicast and multicast DA packets. The unicast or multicast destination address filter result is inverted in this mode. Similarly, writing 1 to the SAIF field reverses the result of unicast SA filter.

Table 482 and Table 483 summarize the DA and SA filtering on the basis of the type of packets received.

**Table 482. Destination address filtering**

| Packet type | PR | HPF | HUC | DAIF | HMC | PM | DBF | DA filter operation |
|---|---|---|---|---|---|---|---|---|
| Broadcast | 1 | X | X | X | X | X | X | Pass |
| | 0 | X | X | X | X | X | 0 | Pass |
| | 0 | X | X | X | X | X | 1 | Fail |
| Unicast | 1 | X | X | X | X | X | X | All packets pass |
| | 0 | X | 0 | 0 | X | X | X | Pass on perfect or group filter match |

*Table continues on the next page...*

Table 482. Destination address filtering (continued)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 0 | X | 0 | 1 | X | X | X | Fail on perfect or group filter match |
| | 0 | 0 | 1 | 0 | X | X | X | Pass on hash filter match |
| | 0 | 0 | 1 | 1 | X | X | X | Fail on hash filter match |
| | 0 | 1 | 1 | 0 | X | X | X | Pass on hash or perfect or group filter match |
| | 0 | 1 | 1 | 1 | X | X | X | Fail on hash or perfect or group filter match |
| Multicast | 1 | X | X | X | X | X | X | Pass all packets |
| | X | X | X | X | X | 1 | X | Pass all packets |
| | 0 | X | X | 0 | 0 | 0 | X | Pass on perfect or group filter match and drop pause packets if PCF = 0x |
| | 0 | 0 | X | 0 | 1 | 0 | X | Pass on hash filter match and drop pause packets if PCF = 0x |
| | 0 | 1 | X | 0 | 1 | 0 | X | Pass on hash or perfect or group filter match and drop pause packets if PCF = 0x |
| | 0 | X | X | 1 | 0 | 0 | X | Fail on perfect or group filter match and drop pause packets if PCF = 0x |
| | 0 | 0 | X | 1 | 1 | 0 | X | Fail on hash filter match and drop pause packets if PCF = 0x |
| | 0 | 1 | X | 1 | 1 | 0 | X | Fail on hash or perfect/group filter match and drop pause packets if PCF = 0x |

Table 483. Source address filtering

| Packet type | PR | SAIF | SAF | SA filter operation |
|---|---|---|---|---|
| Unicast | 1 | X | X | Pass all packets. |
| | 0 | 0 | 0 | Pass status on perfect or group filter match but do not drop packets that fail |
| | 0 | 1 | 0 | Fail status on perfect or group filter match but do not drop packet |
| | 0 | 0 | 1 | Pass status on perfect or group filter match and drop packets that fail |
| | 0 | 1 | 1 | Fail status on perfect or group filter match and drop packets that fail |

**NOTE**

When you write 1 to the MAC_Packet_Filter[RA] field, all packets are forwarded to the system along with the correct result of the address filtering in the Rx Status.

### 72.7.2 VLAN filtering

VLAN tag filter can be done either by perfect match or hash table. Depending upon the configuration, MAC can compare either lower 12 bits or all 16 bits of received VLAN tag for perfect match. MAC forward the VLAN tagged packet with match status when it drop all the packets which do not match.

MAC uses 16-bit VLAN hash table for group address filtering on the basis of the VLAN tag. If MAC enables the hash filtering then the most significant 4-bit of CRC-32 of VLAN tag are used to index the content of the MAC_VLAN_Hash_Table. A value of 1 in the VLAN hash table register, corresponding to the index indicates that the VLAN tag of the packet has matched and the packet should be forwarded. A value of 0 indicates that VLAN-tagged packet should be dropped.

MAC supports the inverse matching for VLAN packets. In the Inverse matching mode, the packet must be dropped when the VLAN tag of a packet matches the perfect or hash filter. If MAC enables the VLAN perfect and VLAN hash match, then a packet is considered as matched if either the VLAN hash or the VLAN perfect filter matches. When inverse match is set, MAC forwards a packet only when both the perfect and hash filters indicate mismatch.

Table 484 specifies the different possibilities for VLAN matching and the final VLAN match status. When you write 1 to MAC_Packet_Filter[RA] all packets are received and it indicates the VLAN match status in the VF field of RDES2 normal descriptor (write-back format). When the RA field is not set and when you write 1 to MAC_Packet_Filter[VTFE] the packet drops if the final VLAN match status fails

When VLAN VID programs to 0 in the VL field of MAC_VLAN_Tag register, consider that all the VLAN-tagged packets are perfectly matched but the status of the VLAN hash match depends on the VTHM and VTIM fields in MAC_VLAN_Tag register.

**Table 484. VLAN match status**

| VID | VLAN perfect filter match result | VTHM bit | VLAN hash filter match result | VTIM bit | Final VLAN match status |
|---|---|---|---|---|---|
| VID = 0 | Pass | 0 | X | X | Pass |
| | Pass | 1 | X | 0 | Pass |
| | Pass | 1 | Fail | 1 | Pass |
| | Pass | 1 | Pass | 1 | Fail |
| VID1= 0 | Pass | X | X | 0 | Pass |
| | Fail | 0 | X | 0 | Fail |
| | Fail | 1 | Fail | 0 | Fail |
| | Fail | 1 | Pass | 0 | Pass |
| | Fail | 0 | X | 1 | Pass |
| | Pass | X | X | 1 | Fail |
| | Fail | 1 | Pass | 1 | Fail |
| | Fail | 1 | Fail | 1 | Pass |

In Table 484, X represents any value.

### 72.7.3 VLAN filter fail packets queue

When you enable VLAN filtering, route the VLAN filter fail packets to a programmable queue (VFFQ) when the value of MAC_Packet_Filter[RA] = 1 or MAC_Packet_Filter[VTFE] = 0 and write 1 to MAC_RxQ_Ctrl4[VFFQE] for the queue.

Route the packets that passes the VLAN filtering to Rx queues on the basis of the VLAN tag priority field by configuring the PSRQ field in the corresponding MAC_RxQ_Ctrl2 and MAC_RxQ_Ctrl3 registers. Discard the packets that fail the VLAN filter if RA=0 or VTFE=1. However, when RA=1 or VTFE=0, forward the VLAN filter fail packets to the application. In such case, when you write 1 to the VLAN filter fail queue enable (VFFQE) field, the VLAN filter fail packets forward to the Rx queue number programmed in the VFFQ field. If the value of VFFQE=0, the VLAN priority mapping determines the Rx queue number per the PSRQ fields.

Table 485 shows the Rx queue routing table for unicast tagged packets, with DA or SA filter enabled.

**Table 485. Rx queue routing table for unicast tagged packets**

| RA | VFTE | SA or DA filter result | VLAN filter result | VFFQE | Queue routing |
|---|---|---|---|---|---|
| X | X | Pass | Pass | X | PSRQ |
| 0 | 0 | Pass | Fail | 0 | PSRQ |
| 0 | 0 | Pass | Fail | 1 | VFFQ |
| 0 | X | Fail | X | X | Dropped |
| 0 | 1 | Pass | Fail | X | Dropped |
| 1 | X | Fail | X | 0 | UFFQ*/PSRQ |
| 1 | X | Fail | X | 1 | UFFQ*/VFFQ |
| 1 | X | Pass | Fail | 0 | PSRQ |
| 1 | X | Pass | Fail | 1 | VFFQ |

X : Don't care condition * : When UFFQE is enabled else PSRQ.

### 72.7.4 Extended receive VLAN filtering and routing

When the extended Rx VLAN filtering and routing is enabled then both the perfect filtering and hash filtering can be enabled. The overall VLAN filter result is based on the perfect filter result and hash filter result (if enabled). Filter result is passed to application as part of the status bit. Extended routing will take place only if VLAN filter has passed. Routing is only based on perfect filter result. Each perfect filter has a DMA channel enable and DMA channel number filed which must be programmed for routing.

See the extended VLAN based DMA selection in Dynamic (per packet) mapping for more information about routing.

#### 72.7.4.1 Comparison mode

Application has these comparison option for each VLAN tag filter:

- Programs MAC to compare the inner or outer VLAN tag either with 12 bits or 16 bits programmed VID.
- Selects whether the VID comparison is for SVLAN or CVLAN type frames, if type check is enabled for a filter.

#### 72.7.4.2 Filtering

Perfect filtering is done on the basis of the MAC_VLAN_Tag_Filter registers. MAC compares the relevant VLAN tag ID and gives a result for each VLAN tag filter.

The results for each VLAN tag filter are:

- Pass- If any one of the VLAN tag filters gives a match.
- Fail- If the frame mismatches all the filters.

This behavior is applicable only when the inverse filtering is not enabled in MAC_VLAN_Tag_Ctrl.

If inverse filtering is enabled and the frame mis-matches all the relevant filters then it is considered to have passed the VLAN filter. If the frame matches any one of the relevant filters then it is considered to have failed. The frame is bypassed to the application if none of the enabled filters can perform a comparison or if none of the filters are enabled.

The overall filter result and the programming on MAC_Packet_Filter[VTFE] and MAC_Packet_Filter[RA] determines if the frame will drop or it is forwarded to the application. If the value of RA = 1 or VTFE = 0, then the frame is always forwarded whether the filter result is a pass or fail, . If the value of RA = 0 and VTFE = 1, only then, if the VLAN tag filter result is a pass the MAC forwards the frame. If the frame is forwarded to the application, then the relevant filter result is indicated through the status bits.

#### 72.7.4.3 Filter status

The extended receive VLAN filtering and routing feature provides two status fields to show the comparison result of the VLAN tags.

By default, MAC indicates the VLAN filter status through one bit in the status VF field in RDES2. When you enable the extended RX VLAN filtering and routing, two status fields will show the comparison result of the VLAN tags. The outer VLAN tag filter pass and inner VLAN tag filter pass bits are defined in the following positions. The status indicated through these fields depends on the programming as described below.

In RDES2:

- Bit 15 – Outer VLAN tag filter status
- Bit 14 – Inner VLAN tag filter status

In ARI status: MAC filter status:

- Bit 15 – Outer VLAN tag filter status
- Bit 14 – Inner VLAN tag filter status

In MRI status:

- Bit 47 – Outer VLAN tag filter status
- Bit 46 – Inner VLAN tag filter status

Outer VLAN tag filter status (OTS)

- In perfect filtering, without enabling inverse filtering, if you write 1 to this field, it indicates that the frame's outer VLAN tag has matched one of the VLAN tag filters.
- If this field resets, it indicates that the frame's outer VLAN tag has either failed the relevant outer VLAN tag filters or bypassed them.
- This field resets if none of the filters are enabled for outer VLAN tag comparison.
- If inverse filtering is enabled and if you write 1 to this field, then the frame's VLAN tag has passed all the relevant VLAN tag filters. If it is resets, then it has failed at least one filter or bypassed all the filters programmed for outer VLAN tag comparison.
- This bit is valid for both single and double VLAN tagged frames.

Inner VLAN tag filter status (ITS)

- In perfect Filtering, without enabling inverse filtering, if you write 1 to this field, it indicates that the frame's inner VLAN tag has matched one of the VLAN tag filters.
- If this field resets, it indicates that the frame's inner VLAN tag has either failed the relevant inner VLAN tag filters or bypassed them.
- This field resets if none of the filters are enabled for inner VLAN tag comparison.
- If Inverse Filtering is enabled and if you write 1 to this field, then the frame's VLAN tag has passed all the relevant VLAN tag filters. If it resets, then it has failed at least one filter or bypassed all the filters programmed for inner VLAN tag comparison.

- This bit is valid for only double VLAN tagged frames, when double VLAN processing is enabled.

The application must observe the status fields and the program to determine if the frame has passed or failed the VLAN filter.

Table 486 and Table 487 describes the possible filter combinations and the corresponding filter results and they also explain the scenarios when the double VLAN processing and the hash VLAN filter are enabled in the design.

Legend for the table OTS and ITS Bit Values with at least 1 perfect filter enabled

- VTIM: VLAN tag inverse match enable – bit 17 in VLAN_Tag_Ctrl register.
- HFO: Hash filter enabled for outer VLAN tag comparison - bit 25 and bit 27 in VLAN_Tag_Ctrl register.
- HFI: Hash filter enabled for inner VLAN tag comparison – bit 25 and bit 27 in VLAN_Tag_Ctrl register.
- PFO – Perfect filter comparison enabled for outer VLAN tag - Any of the MAC_VLAN_Tag_Filter registers is enabled (write 1 to bit 16) and programmed for outer VLAN tag comparison (write 0 to bit 20).
- PFI – Perfect filter comparison enabled for Inner VLAN Tag - Any of the MAC_VLAN_Tag_Filter registers is enabled (write 1 to bit 16) and programmed for inner VLAN tag comparison (write 1 to bit 20).
- OTS – Outer VLAN tag filter status
- ITS – Inner VLAN tag filter status

Table 486 shows the possible values of status bits (OTS and ITS) when at least one perfect filter is enabled.

**Table 486. OTS and ITS bit values with at least 1 perfect filter enabled**

| VTIM | HFO | HFI | PFO | PFI | OTS | ITS |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 | 1/0 |
| 0 | 0 | 0 | 1 | 0 | 1/0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1/0 | 1/0 |
| 0 | 1 | 0 | 1 | 1 | 1/0 | 1/0 |
| 0 | 1 | 0 | 1 | 0 | 1/0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1/0 | 1/0 |
| 0 | 0 | 1 | 1 | 1 | 1/0 | 1/0 |
| 0 | 0 | 1 | 1 | 0 | 1/0 | 1/0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1/0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1/0 |
| 1 | 0 | 0 | 1 | 0 | 1/0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1/0 | 1/0 |
| 1 | 1 | 0 | 1 | 1 | 1/0 | 1/0 |
| 1 | 1 | 0 | 1 | 0 | 1/0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1/0 | 1/0 |

*Table continues on the next page...*

Table 486. OTS and ITS bit values with at least 1 perfect filter enabled (continued)

| VTIM | HFO | HFI | PFO | PFI | OTS | ITS |
|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 | 1/0 | 1/0 |
| 1 | 0 | 1 | 1 | 0 | 1/0 | 1/0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1/0 |

Table 487 shows the possible values of status bits (OTS and ITS) when no perfect filters are enabled except the VLAN hash filter is enabled.

Table 487. OTS and ITS bit values with only VLAN hash filter enabled

| VTIM | HFO | HFI | OTS | ITS |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1/0 | 0 |
| 0 | 0 | 1 | 1/0 | 1/0 |
| 1 | 0 | 0 | 1/0 | 0 |
| 1 | 1 | 0 | 1/0 | 0 |
| 1 | 0 | 1 | 1/0 | 1/0 |

With no perfect filters enabled, any VLAN packet is considered to have bypassed the perfect filter. If Hash Filter is enabled for one of the Tags, then the respective Status bit depends on the Filter's result. The Status bits are set to 0 if VLAN Hash Filter is also not enabled.

If the value of ITS/OTS is shown as 1/0; then it indicates that the final result is dependent on the enabled relevant filter's result.

Example 1: The second row of OTS and ITS bit values with only VLAN hash filter enabled indicates that at least one Perfect Filter is enabled for Outer VLAN tag comparison and none of the filters are enabled for Inner VLAN tag comparison. Inverse VLAN Filtering is not enabled. The bit OTS is given as 1/0. If the received frame passes at least one of the enabled Outer VLAN Tag filters then the bit is set to 1. If the frame doesn't pass any of the enabled Outer VLAN Tag filters, then the bit is set to 0.

Example 2: The last row of table "OTS and ITS bit values with only VLAN hash filter enabled" indicates that the inverse filtering is enabled, hash filter and at least one perfect filter is enabled for Inner VLAN Tag comparison, then if the received frame's Inner VLAN tag mismatches with both the Hash Filter and all the enabled Perfect filters, then the frame will have the ITS bit set to 1 else it is set to 0. OTS will be set to 0 as no comparison is performed.

#### 72.7.4.4 Stripping

Each VLAN tags has individual control over stripping. The programming options of always strip, never strip, strip on pass and strip on fail are available. Inner or outer VLAN tag stripping is based on the pass or fail results of the individual tag. If all the relevant filters bypasses a tag, stripping is not applicable for the tag.

- If strip on pass is enabled for the outer VLAN tag, then the stripping occurs only if the outer VLAN tag has passed the relevant filters. The outer VLAN tag filter result field will be 1.
- If strip on fail is enabled for the outer VLAN tag, then the stripping occurs only if the outer VLAN tag has failed relevant filters. The outer VLAN tag filter result field will reset.
- If the outer VLAN tag of the received frame is bypassed by the entire filter (no comparison has been made), then the tag is not stripped, though the status bit is 0.

- As multiple filters are enabled, it is possible that the received VLAN frame could have matched any one or more of the filters. The VLAN Tag's value is not always deterministic from the filter status bits.

If an application strips the VLAN tag on the basis of the filter result, it might lose the VID. You can use it, if stripping is enabled for any of the tags, place the tag in the status. For this the software or application will enable the respective VLAN tag in status bit - 24 or 31 in the MAC VLAN tag control register.

See section Programming guidelines for extended VLAN filtering and routing on receive for more information.

### 72.7.5 Layer 3 and layer 4 filtering

IP supports the layer 3 based packet filtering which is an IP source or destination address filtering in the IPv4 or IPv6 packets, and layer 4 based packet filtering which is a source or destination port number filtering in TCP or UDP. When you enable layer 3 and layer 4 packet filtering then packets are filtered in the following manner.

- Matched packets: MAC forward the packets that match all enabled fields to the application along with the status. It gives the matched field status only if MAC_Configuration[IPC] = 1 and if one of the following conditions is true:
  - All enabled layer 3 and layer 4 fields match
  - At least one enabled field matches and other fields are bypassed or disabled

When you enable multiple layer 3 and layer 4 filters, any filter match is considered as a match. If more than one filter matches, MAC provides the status of the lowest filter where filter 0 is the lowest filter and filter 3 is the highest filter.

- Unmatched packets: MAC drop the packets that do not match any enabled fields. You can use the inverse match feature to block or drop a packet with specific TCP or UDP over IP fields and forward all other packets.
- Non-TCP or UDP IP packets: By default, all the non-TCP or UDP IP packets bypasses the layer 3 and layer 4 filters. You can optionally program MAC to drop all the non-TCP or UDP packets over IP packets.
- IP has a control register MAC_L3_L4_Control to control the layer L3 and L4 packet filtering along with address registers to program the layer L3 and L4 fields to be matched.

### 72.7.6 Flexible receive parser

When you enable this function, all incoming packets are parsed per the programmable instruction stored in memory. It perform these functions:

- Packet filter (Accept or reject)
- DMA channel selection

Flexible receive parser block operates over the first 128 bytes data of receive packet on the basis of 64 bytes of instruction on each packet. Figure 390 shows the functional block diagram of flexible receive parser.

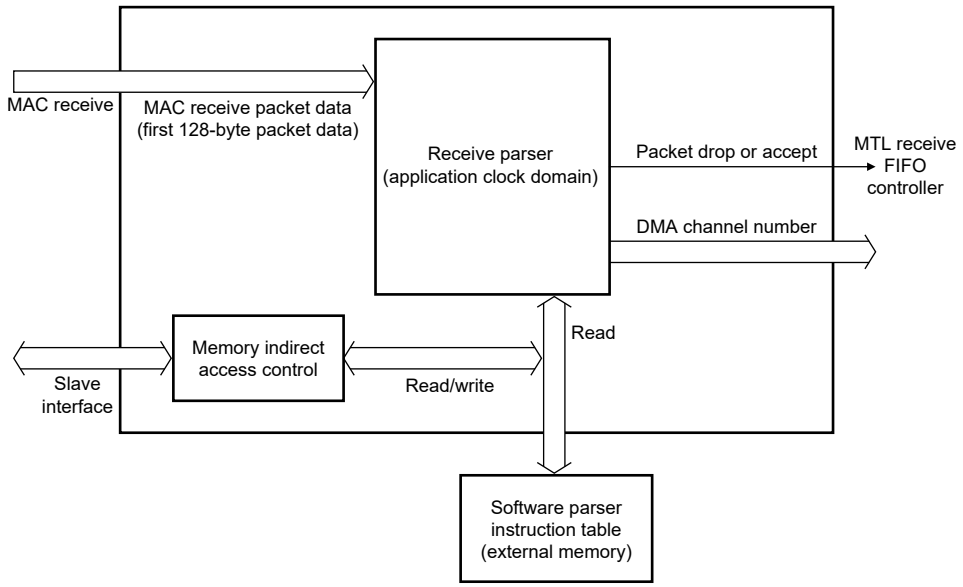



**Figure 390. Functional block diagram of flexible receive parser**

### 72.7.6.1 Instruction table format of the flexible receive parser

You must build the 96-bit wide instruction before enabling the receive parser feature and program each instruction as an entry in the instruction memory.

Table 488 shows the format of each entry in the instruction table .

**Table 488. Rx parser instruction entry format**

| Field | Field Name | Description |
|---|---|---|
| 31:0 | MATCH_DATA | This is a 4-byte data. You can use it to compare with incoming packet data which starts at the frame offset as defined in [79:72] of the entry. The comparison is done only on those bits whose corresponding mask bits are 1 (MATCH_EN bits of [63:32]). See the notes (below this table) on the byte order relative to Ethernet arrival data and endianness which the QoS IP supports on the slave interface. The programming of this instruction table must adhere to the same. |
| 62:32 | MATCH_EN | When the MATCH_EN = 1, you can use the corresponding packet data bit for comparing. Otherwise, corresponding data bit is don’t care. |
| 64 | AF | Accept frame |
| 65 | RF | Reject frame |
| 66 | IM | Inverse match |
| 67 | NC | Next instruction control |
| 71:68 | Reserved | - |

*Table continues on the next page...*

Table 488. Rx parser instruction entry format (continued)

| Field | Field Name | Description |
|---|---|---|
| 79:72 | Frame offset | [77:72] indicates the frame offset in terms of 4 bytes. (Here [79:78] always 0) Compare the frame offset in the packet data for Match. This is in terms of 4 bytes. The value of actual frame offset in bytes are 0 01 42 8… ..63 252<br>The max value programmed in this instruction table is 128 B. The 'Actual frame offset' must not cross this limit. |
| 87:80 | OK index | If (NIC==0) the memory index you can use next when ENTRY_MATCH==1 and neither AF or RF = 1. If (NIC==1) the memory index you can use next when ENTRY_MATCH==0 |
| 95:88 | DMA Ch No | Indicates the DMA channel number (1-bit for each). You can use this when ENTRY_MATCH ==1 and AF = 1 (Accept frame). The following bits give the encoding: bit[0]- DMA channel number o, bit[1]- DMA channel number , 1bit[2]- DMA channel number 2… bit[7]- DMA channel number 7.<br>**NOTE**<br>You can encode the DMA channel number using bit wise as QoS IP support multicasting or broadcast across DMA channels. See section 3.3.12 for more information and proper usage. |
| 128:96 | Reserved | 0 (This field is only for software view and reserved for future enhancements. The memory width remains as 96-bits). |

The parser begins parsing from the 0th entry, for each received packet. The subsequent parsing entry location (next entry or OK_INDEX[]) depends on the current entry parser result. Flexible receive parser flow chart shows the detailed flow.

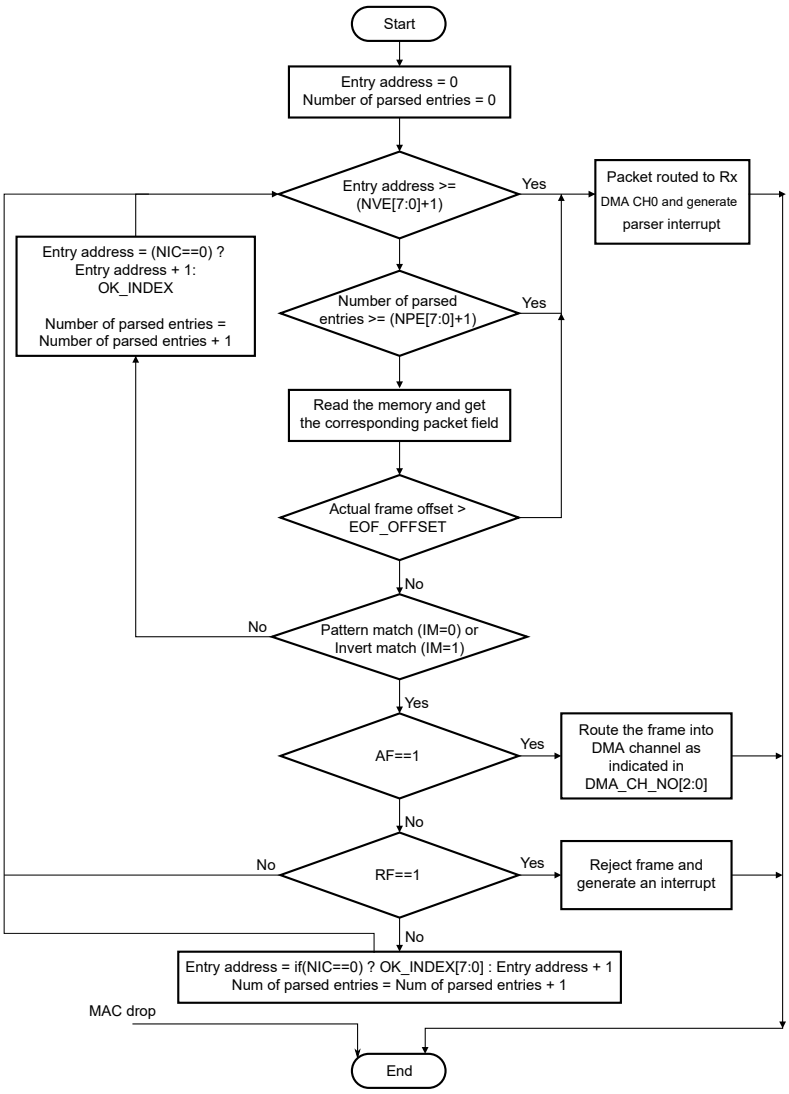



**Figure 391. Flexible receive parser flow chart**

The EOF_OFFSET shown in the figure is the least among the following:

- Actual packet size
- 128 Byte when frame pre-emption feature is not enabled or configured
- 128 Byte when frame pre-emption feature is enabled and packet is pre-emptive
- 64 Byte when frame pre-emption feature is enabled and packet is express packet

### 72.7.6.2 Number of valid entries (NVE)

NVE indicates the number of valid entries in the instruction table. The default value is 64. You can change this value by writing into MTL_RXP_Control_Status[NVE].

When parsing, an error is flagged in MTL_RXP_Interrupt_Control_Status[NVEOVIS], if the entry address is greater than MTL_RXP_Control_Status[NVE] + 1. IP generates an interrupt status (MTL_Interrupt_Status[MTLPIS]).

#### 72.7.6.3 Number of parsable entries (NPE)

NPE indicates the number of entries that you can parse on an incoming packet. The default value is 64. Write into MTL_RXP_Control_Status[NPE] to change this value.

When parsing, an error is flagged in MTL_RXP_Interrupt_Control_Status[NPEOVIS], if the number of parsed of parsed entries is more than MTL_RXP_Control_Status[NPE] + 1. IP generates an interrupt status (MTL_Interrupt_Status[MTLPIS]).

NVE[] indicates the number of valid entries in the receive parser memory which you have built and NPE[] indicates the worstcase parsable entries considering the various possible paths that the parser can take. (NVE[] >= NPE[]).

#### 72.7.6.4 Frame offset

It indicates the frame offset, in terms of 4 bytes which you use when comparing against the current table entry.

In an entry, an error is flagged in MTL_RXP_Interrupt_Control_Status[FOOVIS] in any of the following cases:

When frame offset is more than:

- Packet size
- 128B if pre-emption is disabled
- 128B if pre-emption is enabled and packet is pre-empitble
- 64B if pre-emption is enabled and packet is an express packet

IP generates an interrupt status (MTL_Interrupt_Status[MTLPIS]).

**NOTE**

If the frame ends after the (frame_offset *4) bytes but before ((frame_offset *4)+4) bytes, the receive parser declares a frame offset error if you enable comparison (via MASK bits) for the non-received bytes; from (frame_offset *4) to ((frame_offset *4)+4).

#### 72.7.6.5 Frame reject

You can drop the frame in cases, where an entry matches (considering the inverse match, IM bit in the instruction table) and frame accept bit is not 1, and frame reject bit (RF) = 1.

When a frame drops because RF = 1, IP generates an interrupt status (MTL_Interrupt_Status[MTLPIS]).

#### 72.7.6.6 Out of order processing

The Rx parser flow can be out of order. The decision tree is not based on the order in which packet arrives. This implies that the next entry and next frame offset (OK_INDEX[] and FRAME_OFFSET[]) can be less than the current entry address and current frame offset.

#### 72.7.6.7 DMA channel selection

When you enables the receive parser (MTL_Operation_Mode[FRPE] = 1) it overrides the MAC DMA selection criteria.

When you disables the receive parser (MTL_Operation_Mode[FRPE] = 0) the MAC/MTL determines the DMA channel number.

#### 72.7.6.8 Receive queue selection

The existing MAC functionality determines the receive queue, irrespective of whether the receive parser is enabled or disabled. This is because the receive parser is mainly designed to select the DMA channel and the MAC decides the receive queuing to enable the proper functioning of the pause or PFC feature.

**NOTE**

When there are multiple queues and multiple DMA Channels are enabled, a particular DMA channel might receive out of order packets because queue selection is based on MAC criteria (VLAN Priority and other criteria) and Rx DMA channel selection is based on receive parser.

#### 72.7.6.9 MAC packet filtering/drop/error handling

Packet filtering is done on the basis of the received packet fields. You must disable the packet filtering features inside the MAC when the receive parser is enabled. Follow these steps to disable the packet filtering features inside the MAC :

1. Write 1 to the Promiscuous mode (MAC_Packet_Filter[PR]).
2. Write 1 to all other fields in MAC_Packet_Filter to its default values.

When MAC decides to drop the packet due to the receive MAC dependent error such as:

- GMII error
- Receive watchdog error
- CRC error
- Giant frame error

the packet drops irrespective of the receive parser decision.

#### 72.7.6.10 Pad strip or CRC strip handling

Software controls the pad strip and CRC strip and it is applicable to all the received packets. Also,software must build the receive parser instructions accordingly, when you enable these features.

**NOTE**

Enable pad and CRC stripping in real use case.

#### 72.7.6.11 VLAN strip handling

MAC supports stripping the outer VLAN as well as inner VLAN. When you program MAC to strip the VLAN, the receive parser considers those VLAN tags to be part of the incoming packet.

#### 72.7.6.12 Multicast and broadcast support

IP supports multicast and broadcast packet and the packet route to the highest queue number.

When you enables the receive parser, the DMA channel numbers (DMA CH NO, Bits [95:88]) in the instruction table, decide the packet routing.

When you disables the receive parser, DCS (DMA Channel Select) field of the MAC_AddressX_High register determines the packet routing. See the Broadcast/multicast packet duplication.

**NOTE**

You can select multiple DMA channels for the packet routing because the DCS/DMA CH NO field is per the DMA channel control.

#### 72.7.6.13 Pre-emption support

IP supports the frame preemption feature when the flexible receive parser feature is enabled. It has the following limitations:

- Number of bytes would be 64, so that the receive parser can operate on an express traffic
- Number of bytes would be 128, so that receive parser can operate on a pre-emptive traffic.

#### 72.7.6.14 Software access to the flexible receive parser memory

Software can read and write into the receive parser memory. It uses the following registers via indirect addressing.

- MTL_RXP_Indirect_Acc_Control_Status
- MTL_RXP_Indirect_Acc_Data

For more details, see EMAC register descriptions.

#### 72.7.6.15 Statistical counters

IP supports these statistics counters for flexible receive parser.

1. MTL_RXP_Drop_Cnt
2. DMA_RXP_Error_Cnt

DMA_CH(#i)_Rxp_Accept_cnt (i=0; i<2)

For more details, see EMAC register descriptions.

#### 72.7.6.16 Changing the instruction table by software

Software can update the instruction table in the memory in the following ways:

- Disables the MAC receiver and receive parser by writing 0 to MAC_Configuration[RE] and MTL_Operation_Mode[FRPE]. It waits for receive parser to become inactive (RXPIA, MTL_RXP_Control_Status register bit[31]).
- (Optional) Programs in the entry address 0 to unconditionally (all MATCH_EN bit 0) skip to OK_INDEX[] (to certain location in the memory), where you can program it to reject or accept all the packets.

#### 72.7.6.17 Receive cut-through functionality

In the Cut-Through mode, the receive controller transfer the received packets to DMA just after it receives the cut-through threshold amount of packet data (RTC field in the MTL_RxQx_Operation_Mode register ).

However due to Rx parser's result worst case arrival time can be more than the programmed cut-through threshold. So, the receive controller delays the packet transfer to DMA accordingly, and the cut-through functionality is re-defined as follows:

The receive controller transfers packet to DMA after at least RTC number of bytes have been received and receive parser results are available.

#### 72.7.6.18 Receive packet drop indication

In most cases, the packet drops internally in the receive queue. However, back-to-back packets can arrive for the same queue and the first packet's receive parser result is not available when the second packet arrives. In such cases, the packet cannot be flushed internally and forwarded to DMA or the application interface and indicates the status accordingly. See RDES2, bit 16(RXPD) in Receive normal descriptor (write-back format) for more information.

#### 72.7.6.19 Runt packet handling

The receive parser can performs the operation with 64B minimum size packet. If the runt packet is received (MAC must indicate it), the receive parser assumes that it is dropped in the MAC, this is because, the back-to-back runt packets (< 64B) cannot be handled in the receive parser.

#### 72.7.6.20 Uncorrectable ECC error handling

The packet does not drops when the parser detects an uncorrectable ECC error while parsing the parser memory. The packet is sent to the application with an error indication. The RXPI (RX parser incomplete) sets in the packet status along with error summary (ES) bit.

See the Receive normal descriptor (write-back format) for more information.

## 72.8 IEEE 1588 timestamp support

This section and all its sub-sections are Synopsys proprietary. Used with permission.

IP supports the IEEE 1588 precision time protocol (PTP). It is also able to do precise time stamping and captures incoming and outgoing frame because of the PTP protocol implementation in the IP. It supports all the clock types defined in IEEE 1588-2008. The typical frequency of PTP timestamp clock is 125 MHz.

IP supports these features:

- Provides an option to take snapshot of all packets or only PTP type packets
- Provides an option to take snapshot of only event messages
- Identifies the PTP message type, version, and PTP payload in packets sent directly over Ethernet and sends the status
- Provides an option to measure sub-second time in digital or binary format

### 72.8.1 Delay request-response mechanism

The system or network is classified into the master and slave nodes for distributing the timing and clock information. Figure 392 shows the process that PTP uses for synchronizing a slave node to a master node by exchanging PTP messages.

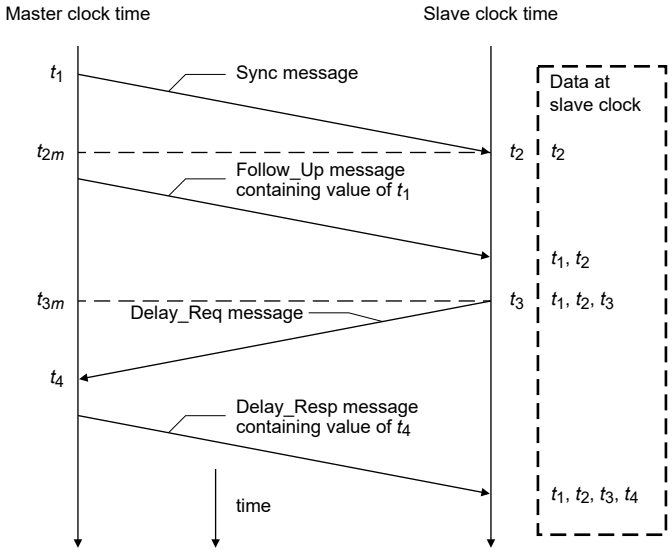



**Figure 392. Networked time synchronization**

As shown in the figure, PTP uses this process:

1. The master broadcast the PTP sync messages to all its nodes. The sync message contains the reference time information of the master. This message leaves the system of the master at t1. You must capture this time for Ethernet ports at GMII or MII.
2. The slave receives the sync message and it also captures the exact time, t2, using its timing reference.
3. The master sends a Follow_Up message to the slave, which contains t1 information for later use.
4. The slave sends a Delay_Req message to the master and note the exact time, t3, at which this packet leaves the GMII or MII interface.
5. The master receives the message, capturing the exact time t4, at which the message enters its system.
6. The master sends the t4 information to the slave in the Delay_Resp message.
7. The slave use the four values of t1, t2, t3, and t4 to synchronize its local timing reference to the master's timing reference.

Most of the PTP implementation is done in the software above the Ethernet layer. However, the hardware captures the exact time when specific PTP packets enter into or leave the Ethernet port at the MII interface. This timing information must be captured and returned to the software for proper implementation of PTP with high accuracy.

### 72.8.2 Peer-to-peer PTP transparent clock (P2P TC) message support

The IEEE 1588-2008 supports the peer-to-peer PTP (Pdelay) message in addition to the sync, delay request, follow-up, and delay response messages. Figure 393 shows the method to calculate the propagation delay in clocks supporting the peer-to-peer path correction.

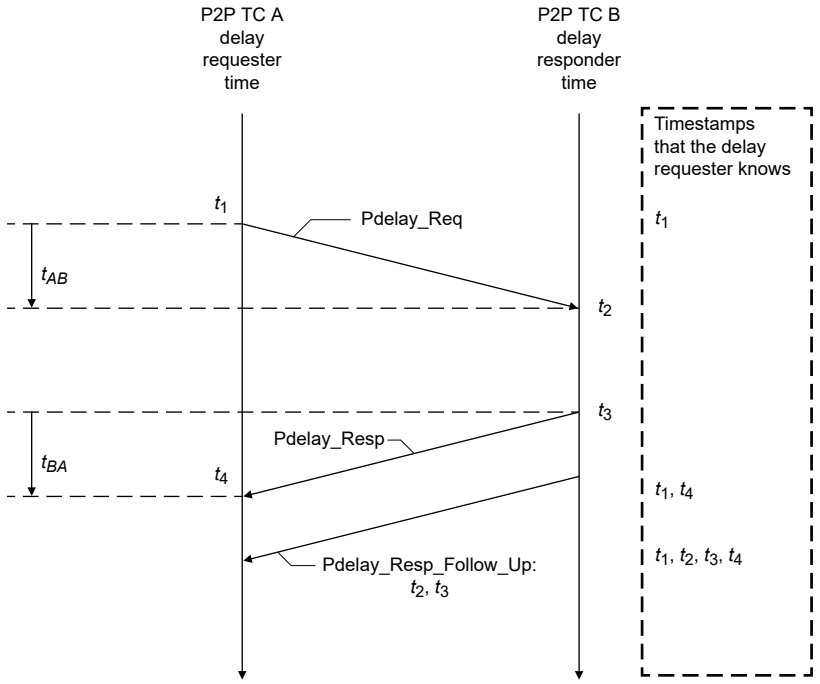



**Figure 393. Propagation delay calculation in clocks supporting peer-to-peer path correction**

As shown in the above figure, the propagation delay is calculated in the following way:

1. Port 1 issues a Pdelay_Req message and generates a timestamp (t1) for the Pdelay_Req message.
2. Port 2 receives a Pdelay_Req message and generates a timestamp (t2) for this message.
3. Port 2 returns a Pdelay_Resp message and generates a timestamp (t3) for this message. Port 2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message, to minimize the errors because of any frequency offset between the two ports. Port 2 returns any one of the following:
   - Difference between the timestamps t2 and t3 in the Pdelay_Resp message
   - Difference between the timestamps t2 and t3 in the Pdelay_Resp_Follow_Up message
   - Timestamps t2 and t3 in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages, respectively
4. Port 1 generates a timestamp (t4) after receiving the Pdelay_Resp message.
5. Port 1 uses all the four timestamps t1, t2, t3, and t4 to compute the mean link delay.

### 72.8.3 Timestamp correction

According to the IEEE 1588 specification, you must capture a timestamp when the PTP message timestamp point (leading edge of the first bit of the octet immediately following the start frame delimiter octet) crosses the boundary between the node and the network. As MAC takes the timestamp at an internal point far from the actual boundary of the node and network, you must correct or update the captured timestamp for the ingress or egress path latency (including the delay in the PHY layers). Further correction

is done for the inaccuracies or errors introduced because the clock (MII Tx, Rx clock) is different at the capture point as compared to the PTP clock (CLK_PTP_REF_I) that generates the time. The resultant CDC (Clock domain crossing) circuits add an error depending on the clock period of the MII and PTP clocks.

#### 72.8.3.1 Ingress correction

The timestamp captured at the internal snapshot point In the receive side is delayed (later in time) as compared to the time at which the packet's SFD bit is received at the port's boundary. Therefore, the ingress latency and the errors in CDC sampling reduces the captured timestamp. You must determine or calculate the correction value and write into the MAC_Timestamp_Ingress_Corr_* registers.

The correction value consists of these three components:

1. External latency in the PHY layer between boundary point and the input of the core.

   If the PHY is compliant with the IEEE 802.3 Clause 45 MMD registers, it has a register which indicates the maximum and minimum ingress latency. You can read these registers and determines the average ingress latency in the PHY. Alternatively (if the PHY does not support these registers), you must determine the ingress latency from its datasheet or timing characteristics.

2. Internal latency from the core's input to the internal capture point.

   You can read the internal ingress latency from MAC_Timestamp_Ingress_Latency. This is a read-only register and provides the latency in scaled nanoseconds format as defined in IEEE 1588 Clause 5.3.2. The latency differs on the basis of the active PHY interface (MII, RMII, so on) and the operating speed. Therefore, you must read this register after any speed change in MAC, to determine the current internal latency.

3. CDC synchronization

   The CDC synchronization error is almost equal to twice the clock-period of the PTP clock (clk_pt-p_ref_i).

You must add the values that these three components determine and must write into the TSIC and TSICSNS fields of the MAC_Timestamp_Ingress_Corr_* registers.

**NOTE**

The value written into the register must be negative (two's complement as explained below), because it is subtracted from the captured timestamp. The MAC receiver adds the value in this register to the captured timestamp and then gives the resultant value as the timestamp of the received packet.

When MAC_Timestamp_Control[TSCTRLSSR] is 1, it indicates that the nanoseconds field of the captured timestamp is in decimal format with a granularity of 1ns. So the Bit31 of TSIC must be 1 (for negative value) and bits[30:0] must write with "10^9 - total ingress_correction_value[nanosecond part]" represented in binary. For example, if the required correction value is -5 ns, then the value is 0xBB9A_C9FB.

When MAC_Timestamp_Control[TSCTRLSSR] becomes 0, it indicates that the nanoseconds field of the captured timestamp is in binary format with a granularity of ~0.466ns. Therefore, bits[30:0] must write with "2^31 - total ingress_correction_value" represented in binary with bit[31] = 1.

#### 72.8.3.2 Egress correction

The timestamp captured at the internal snapshot point in the transmit side is earlier (advanced in time) as compared to the time at which the packet's SFD bit is output at the port's boundary. Therefore, the egress latency and the errors in CDC sampling must compensate the captured timestamp. You must determine or calculate this correction value and write into the MAC_Timestamp_Egress_Corr_* registers.

The correction value consists of these three components:

1. External latency in the PHY layer between the core output and the port and the network boundary

   If the PHY is compliant with the IEEE 802.3 Clause 45 MMD registers, it has a register which indicates the maximum and minimum egress latency. You can read these registers and determine the average egress latency in the PHY. Alternatively (if the PHY does not support these registers), you must determine the egress latency from its datasheet or timing characteristics.

2. Internal latency from the internal capture point and the core output of the c

   You can read this internal egress latency from MAC_Timestamp_Egress_Latency. This is a read-only register and gives the latency in scaled nanoseconds format as defined in IEEE 1588 Clause 5.3.2. The latency differs on the basis of the active PHY interface (MII, RMII, so on) and the operating speed. Therefore, you must read this register after any speed change in MAC to determine the current internal latency.

3. CDC synchronization error

   The CDC synchronization error value for one-step timestamping = (1 * period of CLK_PTP_REF_I + 4 * period of CLK_TX_I). Otherwise (Two-step timestamping mode), the value = -(2 * period of CLK_PTP_REF_I).

#### 72.8.3.3 Frequency range of reference timing clock

The timestamp information is transferred across asynchronous clock domains that is from the MAC clock domain to the application/system clock domain. Therefore, a minimum delay is required between the two consecutive timestamp captures. This delay is 4 clock cycles of MII and 3 clock cycles of PTP clocks. If the delay between the two captured timestamp is less than this delay, MAC does not take a timestamp snapshot for the second packet.

#### 72.8.3.4 PTP processing and control

Table 489 shows the common message header for the PTP messages. This format is defined in the IEEE 1588-2008.

**Table 489. Message format defined in IEEE 1588-2008**

| Bits | | | | | | | | Octet | Offset |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| transportSpecific | | | | messageType | | | | 1 | 0 |
| Reserved | | | | versionPTP | | | | 1 | 1 |
| messageLength | | | | | | | | 2 | 2 |
| domainNumber | | | | | | | | 1 | 4 |
| Reserved | | | | | | | | 1 | 5 |
| flagField | | | | | | | | 2 | 6 |
| correctionField | | | | | | | | 8 | 8 |
| Reserved | | | | | | | | 4 | 16 |
| sourcePortIdentity | | | | | | | | 10 | 20 |
| sequenceId | | | | | | | | 2 | 30 |
| controlField (*) | | | | | | | | 1 | 32 |
| logMessageInterval | | | | | | | | 1 | 33 |

(*) – control Field is used in version 1. In version 2 message type field is used for detecting different message types.

There are some fields in the Ethernet payload that detects the PTP packet type and controls the snapshot to be taken. These fields are specified in Table 490 for PTP packets over Ethernet.

Following table provides the information about the fields that match to control the snapshots for the PTP packets sent over Ethernet for IEEE 1588 version 1 and version 2. The octet positions are offset by 4 for the tagged packets. This is based on the IEEE 1588-2008, Annex D, and the message format.

**Table 490. Ethernet PTP packet fields required for control And status**

| Field matched | Octet position | Matched value | Description |
|---|---|---|---|
| MAC destination multicast address[1] | 0–5 | 01-1B-19-00-00-00<br>01-80-C2-00-00-0E | All PTP messages can use any of the following multicast addresses[2]:<br>• 01-1B-19-00-00-00<br>• 01-80-C2-00-00-0E[3] |
| MAC packet type | 12, 13 | 0x88F7 | PTP Ethernet packet |
| PTP control field<br>(IEEE 1588 version 1) | 46 | 0x00, 0x01, 0x02, 0x03, or 0x04 | • 0x00: SYNC<br>• 0x01: Delay_Req<br>• 0x02: Follow_Up<br>• 0x03: Delay_Resp<br>• 0x04: Management |
| PTP message type field (IEEE 1588 version 2) | 14 (nibble) | 0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, or 0xD | • 0x0: SYNC<br>• 0x1: Delay_Req<br>• 0x2: Pdelay_Req<br>• 0x3: Pdelay_Resp<br>• 0x8: Follow_Up<br>• 0x9: Delay_Resp<br>• 0xA: Pdelay_Resp_Follow_Up<br>• 0xB: Announce<br>• 0xC: Signaling<br>• 0xD: Management |
| PTP version | 15 (nibble) | 0x1 or 0x2 | • 0x1: Supports PTP version 1<br>• 0x2: Supports PTP version 2 |

1. The unicast address match of destination addresses (DA), which is programmed in MAC address 0 to 31, is used if MAC_Timestamp_Control[TSENMACADDR] = 1.
2. IEEE 1588-2008, Annex F
3. MAC does not consider the PTP version 1 messages with peer delay multicast address (01-80-C2-00-00-0E) as valid PTP messages.

### 72.8.4 Transmit path functions

MAC captures a timestamp when the start packet delimiter (SFD) of a packet is sent on the MII interface. You can control the packets, for which the timestamps are captured on a per-packet basis and mark each transmit packet to indicate whether to capture a timestamp for it. MAC does not process the transmitted packets to identify the PTP packets. You can use the control bits in the transmit descriptor to specify the packets. MAC returns the timestamp to the software inside the corresponding transmit descriptor and therefore connects the timestamp automatically to the specific PTP packet. You must write the 64-bit timestamp information to TDES0 and TDES1 fields. The TDES0 field holds the 32 least significant bits of the timestamp.

### 72.8.5 Receive path functions

You can program MAC to capture the timestamp of all packets received on the MII interface or to process the packets to identify the valid PTP messages.

You can use these options of MAC_Timestamp_Control to control the snapshot of the time sent to the application:

- Enable snapshot for all packets
- Enable snapshot for IEEE 1588 version 1 or version 2 timestamp
- Enable snapshot for PTP packets transmitted directly over Ethernet
- Enable timestamp snapshot for the received packet for IPv4 or IPv6
- Enable timestamp snapshot only for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ, or PDELAY_RESP)
- Enable the node to be a master or slave and select the snapshot type

**Table 491. Timestamp snapshot dependency on register bits**

| SNAPTYPSEL | TSMSTRENA | TSEVNTENA | PTP messages |
|---|---|---|---|
| 00 | x | 0 | SYNC, Follow_Up, Delay_Req, Delay_Resp |
| 00 | 0 | 1 | SYNC |
| 00 | 1 | 1 | Delay_Req |
| 01 | x | 0 | SYNC, Follow_Up, Delay_Req, Delay_Resp, Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up |
| 01 | 0 | 1 | SYNC, Pdelay_Req, Pdelay_Resp |
| 01 | 1 | 1 | Delay_Req, Pdelay_Req, Pdelay_Resp |
| 10 | x | x | SYNC, Delay_Req |
| 11 | x | x | Pdelay_Req, Pdelay_Resp |

DMA returns the timestamp to the software inside the corresponding receive descriptor. The extended status, contains the timestamp message status and the IPC status. You must write the extended status in the normal descriptor RDES1 and the snapshot of the timestamp in RDES0 and RDES1 fields of context descriptor. The RDES0 field holds the 32 least significant bits of the timestamp.

See the System time correction for programming guidelines for IEEE 1588 timestamping (System time correction).

### 72.8.6 IEEE 1588 system time source

IP supports both the external and internal time source for reference timestamping.

- External time source uses the 64-bit external time reference and clock as an input in IP. The clock input synchronizes the external timing reference into the MAC clock domain. Upper 32-bit indicates the time in seconds and the lower 32-bit indicates the time in nanoseconds.
- Internal time source uses the only clock input to generate timing reference internally for snapshot and capture timestamps. Internal time reference has two fields.
  - UInteger48 seconds field: The seconds field is the integer portion of the timestamp in seconds units. It is 48-bit wide.

    For example, 2.000000001 seconds are represented as seconds field = 0x0000_0000_0002.

— UInteger32 nanoseconds field The nanoseconds field is the fractional portion of the timestamp in nanoseconds units.

For example, 2.000000001 seconds are represented as nanoseconds field = 0x0000_0001. The nanoseconds field supports the following two modes:

- Digital rollover mode: In this mode, the maximum value in the nanoseconds field is 0x3B9A_C9FF, that is, (10e9-1) nanoseconds.
- Binary rollover mode: In this mode, the nanoseconds field rolls over and increments the seconds field after 0x7FFF_FFFF value . Accuracy is ~0.466 ns per bit.

There is a system time register module, it is used when you use an internal time reference. The 80-bit time is maintained in this module and updated using the input reference clock (CLK_PTP_REF_I). This time is the source for taking snapshots (timestamps) of Ethernet packets which is transmitted or received at the MII interface.

Initialize or correct the system time counter using the coarse correction method. In this method, write the initial value or the offset value to the timestamp update register. For initialization, write the system time counter with the value in the timestamp update register. For system time correction, the offset value is added to or subtracted from the system time.

In the fine correction method, correct the frequency offset and/or frequency drift of a slave clock (CLK_PTP_REF_I) with respect to the master clock (as defined in IEEE 1588-2002) over a period of time instead of in one clock, as in coarse correction. This maintains linear time and does not introduce drastic changes (or a large jitter) in the reference time between the PTP sync message intervals.

See Initialization guidelines for system time generation for programming guidelines for IEEE 1588 timestamping (For internal timestamp source configuration).

### 72.8.7 IEEE 1588 higher word register

You can invoke the higher word register if system time source is internal. MAC timestamp is 64-bit wide. The values of the upper 16-bits of the seconds field are read from the CSR register.

### 72.8.8 Flexible pulse-per-second output

IP supports the flexibility of programming the start or stop time, generates pulse width and interval on pulse-per-second output. It also support four such output signals. By default, IP is in fixed pulse-per-second output mode with interval of 1 second.

Initially you must program the start time in target time registers "MAC_PPSx_Target_Time_Seconds and MAC_PPSx_Target_Time_Nanoseconds" for all desired or enabled pps output. If you re-programs start or stop time then you must do it after the synchronization of earlier programmed value. Bit 31 of MAC_PPS#_Target_Time_Nanoseconds register indicates that the synchronization is complete. If the application programs a start or stop time that has already elapsed, MAC sets an error status bit which indicates the programming error. If enabled, MAC also sets the target time reached interrupt event. The application can cancel the start or stop request only if the corresponding start or stop time has not elapsed. If the time has elapsed, the cancel command has no effect.

Program the PPS width and interval in terms of granularity of system time, that is, the number of units of sub-second increment value.

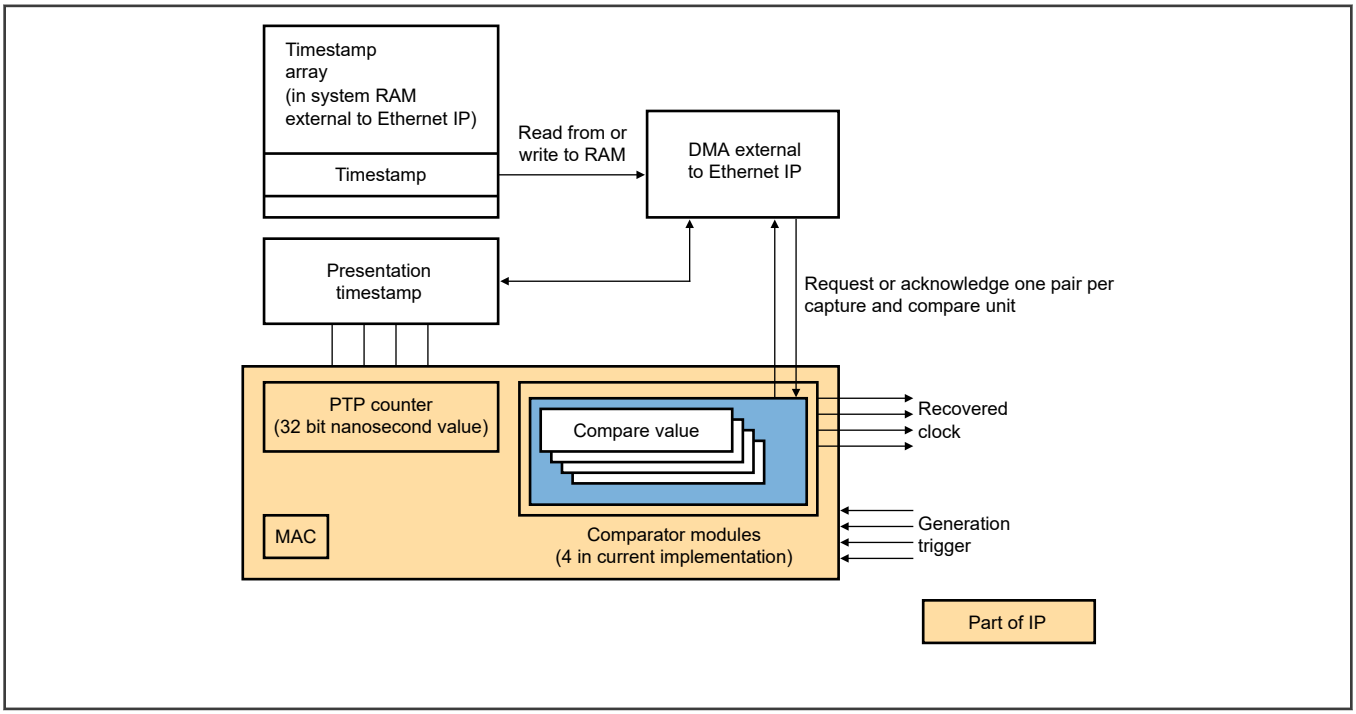
See Programming guidelines for flexible pulse-per-second output for more information.

### 72.8.9 Media clock generation and recovery

Media clock recovery and generation requires a dedicated hardware and software to complete the operation. IP generates a reference clock with the help of a dedicated hardware which multiplies to get the desired media clock. Software extracts the presentation time from the incoming PTP 1722 frames and program these values into CSR registers so that IP can read them for media clock recovery.

Figure 394 shows that the timestamp array and the DMA module are external to IP. IP supports a 32-bit presentation time counter in nanoseconds that completes at the full 32-bit value to match with 1722 presentation time format and an handshake mechanism with an external application (such as DMA) to program the presentation time into CSR register.

**Figure 394. Media clock generation and recovery block diagram**




### 72.8.9.1 Presentation time counter

Presentation time counter provides an another perspective of the PTP system time (1588 timer). For a given PTP system time (PTP[63:32] represents time in seconds Comparator modules and PTP[31:0] represents time in nanoseconds), there exists a corresponding presentation time (32-bit value in nanosecond). The presentation time computed is referred to as current presentation time (CPT).

CPT derives from 64-bit PTP system time. PTPNS[63:0] = (PTP[63:32] * 32'd1,000,000,000) + PTP[31:0] where, PTPNS is the PTP system time converted into a 64-bit nanosecond value.

Current presentation time[31:0] = PTPNS[31:0]

Media clock recovery is possible when the system time is internally generated in IP and compute the CPT in the same way as the internal PTP system time generation. The increment cycle of CPT and system time are same because the values of both the timers are in nanoseconds and they are synchronous. The timer updates at the same instance, but the updated value could be different. You must compute a separate 32-bit update value in nanoseconds for the CPT update. CPT sampled at different edges of a triggering input generates media clock timestamps that are inserted in 1722 based AVBTP packets, to recover the clock at the destination.

### 72.8.9.2 Comparator modules

Figure 394 shows that the module supports four comparator modules to handle multiple media clocks that it may require. When MAC_Timestamp_Control[PTGE] is 1, it indicates that the comparator modules handshake with the application to program the timestamps into the MAC_PPS(#i)_Target_Time_Seconds register.

Write 1 to presentation time control fields of the particular instance for recovery mode.

The timestamps received from the application are referred as target presentation time (TPT).

When MCGREN#i field of MAC_PPS_Control is 1, it indicates that the IP operates in MCGR mode. The comparator module sends up to two requests to the application for TPT write, when MAC_Timestamp_Control[PTGE] and the presentation time control bits of the particular instance are 1 for MCGR mode. Subsequent requests are generated each time a presentation time match occurs. CPT transitions from a value less than TPT, to a value greater than or equal to CPT. The first request asserts when a specific comparator instance sets to MCGR mode with a non-zero presentation time control and an additional request is made when the comparator receives the first data. This allows the application to write the next TPT value when IP processes the previous TPT value for a match.

TPT read from the MAC_PPS(#i)_Target_Time_Second is considered as a future time. A toggle or pulse is generated in the next cycle when a match is detected. Also,the mcgr_dma_req_o#i request for that comparator asserts (to obtain next TPT) until the corresponding application acknowledgment is set, when a match is detected.

The presentation control fields (PPSCMD#i field of MAC_PPS_Control) determine the shape of the generated waveform. You can program these fields to either toggle or generate a high/low pulse for one PTP clock cycle, when matched.

**Media clock recovery:** A match occurs, when the free-running CPT value matches the received TPT value. An output signal, EMAC_TMR signal as shown in the IO mux sheet asserts (toggle, low pulse, or high pulse) on the basis of the programmed presentation control value.

**Media clock generation**: On the basis of the presentation control value programmed in MAC_PPS_Control, the particular comparator captures the presentation time and programs it into MAC_PPS(#i)_Target_Time_Second register. The captured timestamp is read when a request is raised to the application. No new timestamps are captured, until the read operation is complete acknowledgment is received from the application.

### 72.8.9.3 Media clock generation and recovery flow

Figure 395 shows the media clock generation and recovery flow. Write 1 to MAC_Timestamp_Control[PTGE] for CPT generation and write 1 to MAC_PPS_Control MCGREN#i field to enable the corresponding instance in the MCGR mode, for both media clock generation and recovery.

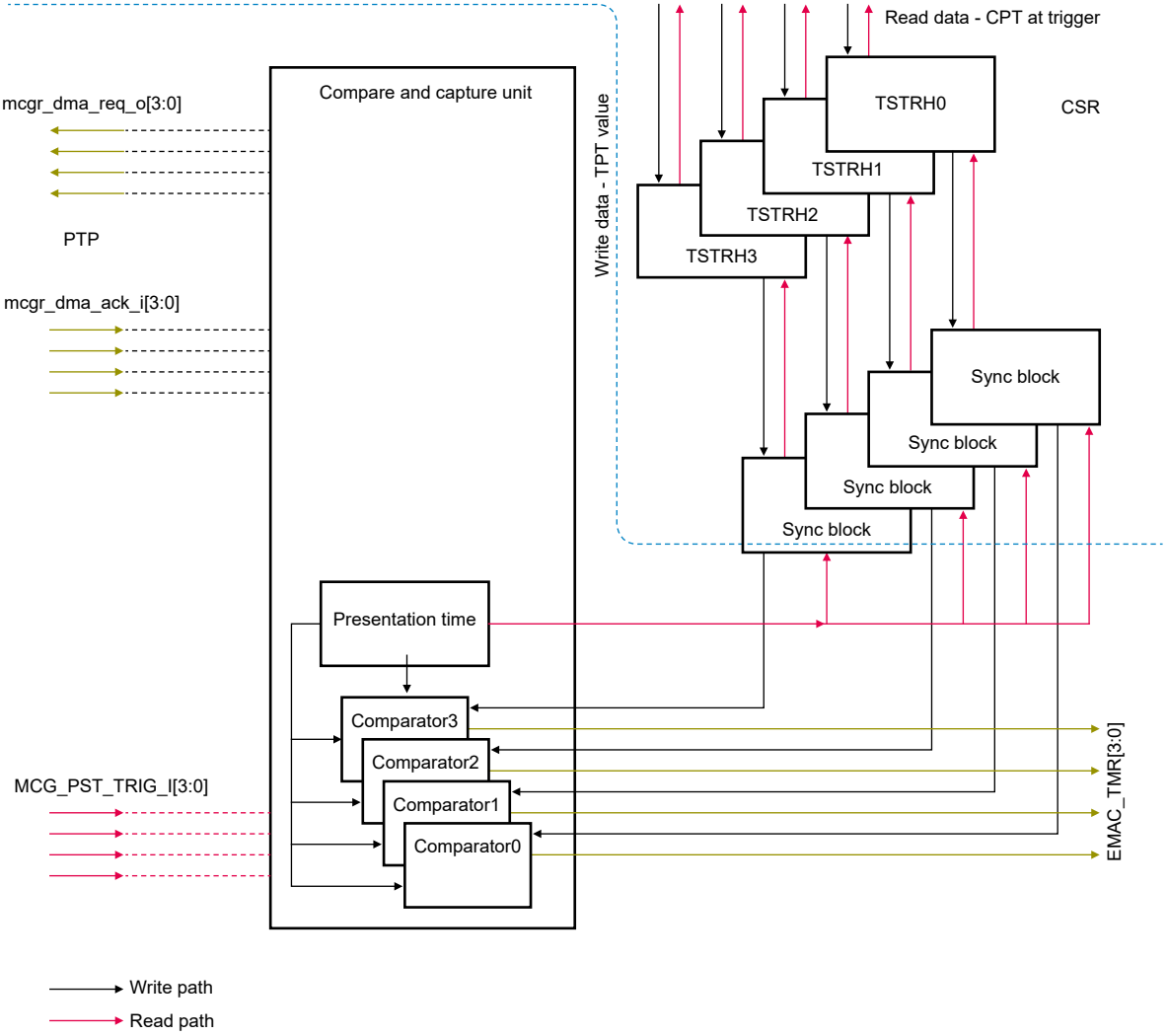



Figure 395. Media clock generation and recovery flow

**NOTE**

1. The consecutive triggers to sample the presentation time must assert after a few PTP clock cycles so as to allow synchronization delays. (There is no such issue when the input trigger maximum frequency is 8 KHz and the PTP clock runs at least at 1MHz)
2. In the Media Clock Recovery mode, the DMA acknowledgment is both posted as well as non-posted.

See Programming guidelines for media clock generation and recovery for more information.

### 72.8.10 One step timestamp

IP supports the one step timestamp feature. Writing 1 to bit 20 (OSTC) in the control word, enables the one step timestamp feature for a packet.

See the Programming guidelines for IEEE 1588 timestamping for more information.

### 72.8.11 IEEE 1588 sub nanoseconds timestamp

IP supports the ingress and egress correction accuracy in sub nanoseconds. It is programmed in MAC_Timestamp_Ingress_Corr_Subnanosec[TSICSNS] and MAC_Timestamp_Egress_Corr_Subnanosec[TSECSNS],

respectively. In this case, the correction has an unit of nanosecond, multiplied by 2^8. You must program the least significant 8 bits of the value in the sub-nanoseconds register.

For example, if the required egress correction is 3.6 nS and MAC_Timestamp_Control[TSCTRLSSR] is 1, then MAC_Timestamp_Egress_Corr_Nanosecond[TSEC] must be 0x3 and MAC_Timestamp_Egress_Corr_Subnanosec[TSECSNS] must be 0x99 (0.6 * 256). Similarly, if the required ingress correction is -3.6 nS and MAC_Timestamp_Control[TSCTRLSSR] is 1, then MAC_Timestamp_Ingress_Corr_Nanosecond[TSIC] must be 0xBB9A_C9FC and MAC_Timestamp_Ingress_Corr_Subnanosec[TSICSNS] must be 0x66 (fractional part of (10^9 - 3.6) * 256).

## 72.9 Multiple channels and queues support

This section and all its sub-sections are Synopsys Proprietary. Used with permission.

IP support two queues and channel on Tx and Rx paths. Figure 396 shows the architecture of IP with two queues and channel.

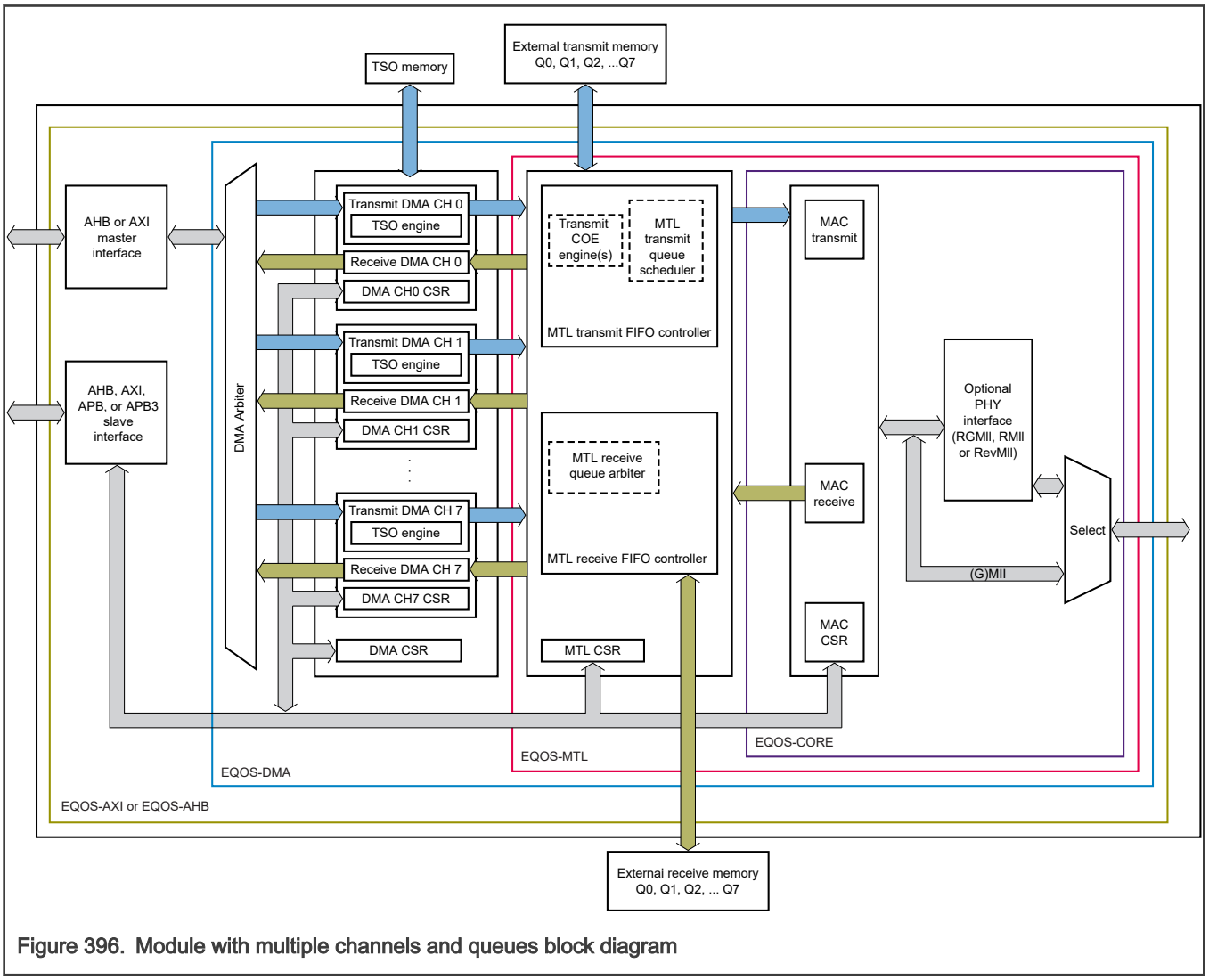



**Figure 396. Module with multiple channels and queues block diagram**

The above block diagram, support only two queues.

IP DMA arbiter support two types of arbitration scheme, fixed priority and weighted round-robin. The DMA arbiter performs the arbitration between the Tx and Rx paths of DMA channels to access descriptors and data buffers. DMA_Mode[DA] specifies the arbitration scheme (fixed or weighted round-robin) between the channel Tx and Rx DMA.

DMA_Mode[TXPR] sets the priority between the corresponding Tx DMA and Rx DMA. Writing 1 to DMA_Mode[PR] specifies the weighted priority between the Tx DMA and Rx DMA in round robin arbitration scheme.

Table 492 provides an information about the priority scheme between Tx DMA and Rx DMA.

**Table 492. Priority scheme for Tx DMA and Rx DMA**

| Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Priority schemes |
|---|---|---|---|---|---|
| x | x | x | 0 | 1 | Rx always has priority over Tx |
| 0 | 0 | 0 | 0 | 0 | Tx and Rx have equal priority. Rx gets the access first on simultaneous requests |
| 0 | 0 | 1 | 0 | 0 | Rx has priority over Tx in ratio 2:1 |
| 0 | 1 | 0 | 0 | 0 | Rx has priority over Tx in ratio 3:1 |
| 0 | 1 | 1 | 0 | 0 | Rx has priority over Tx in ratio 4:1 |
| 1 | 0 | 0 | 0 | 0 | Rx has priority over Tx in ratio 5:1 |
| 1 | 0 | 1 | 0 | 0 | Rx has priority over Tx in ratio 6:1 |
| 1 | 1 | 0 | 0 | 0 | Rx has priority over Tx in ratio 7:1 |
| 1 | 1 | 1 | 0 | 0 | Rx has priority over Tx in ratio 8:1 |
| x | x | x | 1 | 1 | Tx always has priority over Rx |
| 0 | 0 | 0 | 1 | 0 | Tx and Rx have equal priority. Tx gets the access first on simultaneous requests |
| 0 | 0 | 1 | 1 | 0 | Tx has priority over Rx in ratio 2:1 |
| 0 | 1 | 0 | 1 | 0 | Tx has priority over Rx in ratio 3:1 |
| 0 | 1 | 1 | 1 | 0 | Tx has priority over Rx in ratio 4:1 |
| 1 | 0 | 0 | 1 | 0 | Tx has priority over Rx in ratio 5:1 |
| 1 | 0 | 1 | 1 | 0 | Tx has priority over Rx in ratio 6:1 |
| 1 | 1 | 0 | 1 | 0 | Tx has priority over Rx in ratio 7:1 |
| 1 | 1 | 1 | 1 | 0 | Tx has priority over Rx in ratio 8:1 |

### 72.9.1 Multiple queues and channels support in the transmit path

IP support two transmit queues and two channels.

Fixed priority scheme is the default priority scheme for the DMA channels and channel with highest priority always wins the arbitration when it requests for the bus. Channel 0 is always of lowest priority and channel 1 is always of highest priority.

In weighted strict priority (WSP), the weight corresponds to the number of burst transfers given to a channel in one arbitration cycle. Reallocate the unused burst transfers of one or more channels on the basis of the priority. The channel with highest priority receives the unused burst transfer time before it is allocated to a channel with the next highest priority. You can process the next lower priority, when a channel uses the allocated burst transfers. After processing the allocated bandwidth of all the channels that have packets to transmit, allocate any unused burst transfer time to the channel of the highest priority (if required), and then next highest priority (if required), and so on.

In weighted round robin (WRR) priority scheme, program the weight through TCW field of DMA_CH#_Tx_Control register. IN WRR scheme, service all channels in round-robin order according to the weights settings. Table 493 shows that the TCW field of DMA_CH#_Tx_Control register provides the weight for each transmit channel.

**Table 493. Weight for DMA channels**

| TCW Field | Transmit Channel Weight |
|---|---|
| 000 | 1 |
| 001 | 2 |
| 010 | 3 |
| 011 | 4 |
| 100 | 5 |
| 101 | 6 |
| 110 | 7 |
| 111 | 8 |

The configured weights correspond to the number of burst transfers given to a channel in one arbitration cycle. The unused or excess burst transfers are distributed equally to all channels.

See Programming guidelines for multi-channel multi-queuing for more information.

### 72.9.2 Multiple queues and channels support in the receive path

IP support two receive channels and two queues.

In the receive direction, the MTL Rx controller selects the Rx DMA for which it transfers or reads the data from the receive FIFO memory. This scheduling is based on the programming done in the respective MTL_RxQ[x]_Control register.

Each Rx DMA indicates when it is ready to transfer data and the size of the burst-length (number of beats) that it will transfer. The scheduler checks whether sufficient data (of requested burst length) is available to transfer to these DMAs and then selects the receive DMA that is serviced using the programmed priorities.

See Programming guidelines for multi-channel multi-queuing for more information.

### 72.9.3 Rx queue to DMA mapping

You can program MTL_RxQ_DMA_Map0 (for queues 0 and 1) to route the packets in the MTL receive queues to any one of the multiple DMA channels.

The following types of receive queue to DMA mapping is possible through programming.

#### 72.9.3.1 Static mapping

In this mode, you can connect all the packets of the receive queue to a specific DMA channel. For example, all the packets from the receive queue 0 program MTL_RxQ_DMA_Map0[Q0MDMACH] (bit[3:0]) and MTL_RxQ_DMA_Map0[Q0DDMACH] (bit[7] = 0) to route to a DMA channel.

Similarly, packets from other receive queues program register fields corresponding to each queue to route to any DMA channel.

#### 72.9.3.2 Dynamic (per packet) mapping

In this mode, the destination DMA channel of a packet read from a Rx queue is not constant but it is decided independently for each packet. For example, if you write 1 to MTL_RxQ_DMA_Map0[Q1DDMACH], it indicates that the static mapping disables for receive queue 1 and ignores the value in MTL_RxQ_DMA_Map0[Q1MDMACH]. The MAC receiver decides the destination DMA channel for each packet, depending on the following in decreasing order of priority:

1. **L3-L4 filter based DMA selection**: The TCP/UDP and IP header fields of the received packet are matched against the corresponding values programmed and enabled for comparison in the MAC_L3_L4_Address_Control register. If the match is successful, the DMA channel number programmed in the DMCHN field of the MAC_L3_L4_Address_Control register is selected as the destination DMA channel number, if DMCHEN field of the same register is 1. If none of the L3-L4 registers give a comparison match, then the module proceeds to the next step below.
2. **Extended VLAN based DMA selection**: Extended routing is applicable only if the VLAN filter has passed. Routing is done only on the basis of the perfect filter result. Each perfect filter has a DMA channel enable and a DMA channel number field which you can program. Routing is applicable for a filter, only if the DMA channel enable field is 1. The frame routes to the smallest matching filter's DMA channel, if it is enabled. If the filter's DMA channel number is not enabled, the frame route to channel 0. For example, if a frame's VLAN tag matches filters 7, 3, and 1, then the MAC checks if DMA channel number of filter 1 is enabled through programming. If yes, the frame route to the programmed value; otherwise it route to DMA. When the inverse filter is enabled; is routed to the least mismatched filter's DMA channel number provided it is enabled. If the DMA Channel enable bit is not set, then the frame routes on the basis of the DA based addressing or to channel 0.

   If hash filter is also enabled, it is determines the filter result only. Routing will still depends on the enabled perfect filters. Routing based on the VLAN filter will not occur if none of the perfect filters are enabled or if all of them are bypassed. The frame will route via DA based addressing or to channel 0. If all the perfect filters fails and the hash filter has passed, then the VLAN filter result is a pass but routing will be based on DA based addressing or to channel 0. Similar behavior is seen when you enables inverse filtering as well.
3. **Ethernet DA-based DMA selection**: The DA address of the received packet is compared against the programmed DA values in MAC address registers. If the address matches any of the programmed values, the corresponding DCS field (when enabled) determines the destination DMA channel number.

If none of the above operations make a successful match or decision, then the packet routes to the DMA channel 0 by default.

##### 72.9.3.2.1 Broadcast/multicast packet duplication

This feature provides a mechanism to send the received broadcast or multicast packets to multiple DMA channels.

MAC_Address0_High[DCS] and other optional MAC_Address(#i)_High (i=1 to 2) registers determines the DMA channel number to which the received packet (that matches the MAC address present in that register) must route.

DMA channel routing mechanisms such as extended VLAN, or L3-L4 based routing does not support the packet duplication.

The packet duplication feature is supported only on the highest MTL queue configured. Therefore, you must program MAC_RxQ_Ctrl1[MCBCQ] to the highest RxQ present in the configuration for the packet duplication for broadcast or multicast packets.

### 72.9.4 Selection of tag priorities assigned to receive queues

Program the PSRQ field in the corresponding MAC_RxQ_Ctrl2 to assign the VLAN tag priorities to receive queues. The bit corresponding to the VLAN tag priority can be set in the PSRQ field for assigning that priority to the receive queue. For example, if you want to assign VLAN tag priority of 3 to receive queue 0, write 1 to bit [3] in PSRQ field of MAC_RxQ_Ctrl2. The VLAN tag priority assigned to particular receive queue must be unique, that is, you cannot assign more than one receive queue to the same VLAN tag priority. However, more than one VLAN tag priorities can be assigned to same receive queue.

The settings in the PSRQ field is used for VLAN tagged receive packet routing to receive queues as well as for PFC based transit flow control. The received VLAN tagged receive packet route to receive queue that has the VLAN tag priority match. In PFC based transit flow control, PSRQ field corresponding to a particular receive queue enables VLAN tag priorities in the PFC packet transmitted when corresponding receive queue reach the threshold level.

### 72.9.5 Receive side routing from MAC to queues

IP supports receive side routing from MAC to queues. MAC route the receive packets to the receive queues on the basis of the these packet types in that order

- Unicast/Multicast destination address packets that fail the destination address filter
- Multicast/Broadcast destination address packets that pass the destination address filter
- VLAN tag priority field in VLAN tagged AV data packets
- AV control packets
- VLAN tagged IEEE 1588 PTP over Ethernet packets
- Untagged IEEE1588 PTP over Ethernet packets
- VLAN tag priority field in VLAN tagged packets
- Untagged packets

You can route the outer C-VLAN tagged AV data receive packets on the basis of the priorities assigned to receive queues through PSRQ field in the corresponding MAC_RxQ_Ctrl2 and MAC_RxQ_Ctrl3 and the corresponding receive queue is enabled for AV through RXQ#EN field in MAC_RxQ_Ctrl0. These packets may be single VLAN tagged with C-VLAN type or double VLAN tagged with outer VLAN tag of C-VLAN type when double VLAN feature enables (MAC_VLAN_Tag_Ctrl[EDVLP] = 1) with an inner C-VLAN tagged or inner S-VLAN tagged when SVLAN processing is enabled (MAC_VLAN_Tag_Ctrl[ESVL] = 1). This type of Rx packet routing is available when you select the AV feature and multiple receive queues in the configuration.

You can route the AV control receive packets on the basis of the receive queue number specified in MAC_RxQ_Ctrl1[AVCPQ] and the corresponding receive queue is enabled for AV through RXQ#EN field in MAC_RxQ_Ctrl0. These packets may be single VLAN tagged with C-VLAN type or double VLAN tagged with outer VLAN tag of C-VLAN type when double VLAN feature is enabled (MAC_VLAN_Tag_Ctrl[EDVLP] = 1) with inner C-VLAN tagged or inner S-VLAN tagged when SVLAN processing is enabled (MAC_VLAN_Tag_Ctrl[ESVL] = 1). This type of receive packet routing is available when you select the AV feature and multiple receive queues in the configuration.

You can route the VLAN tagged receive packets on the basis of the priorities assigned to receive queues through PSRQ field in corresponding MAC_RxQ_Ctrl2 and MAC_RxQ_Ctrl3 and RXQ#EN field in MAC_RxQ_Ctrl0 enables the corresponding receive queue for DCB/generic. This type of receive packet routing is available when you select the multiple receive queues in the configuration.

You can route the untagged IEEE 1588 PTP over Ethernet receive packets on the basis of the receive queue number specified in MAC_RxQ_Ctrl1[PTPQ] and the corresponding receive queue is enabled through RXQ#EN field in MAC_RxQ_Ctrl0. Also, you can route the VLAN tagged IEEE 1588 PTP over Ethernet receive packets on the basis of the priorities assigned to receive queues through PSRQ field in corresponding MAC_RxQ_Ctrl2 and MAC_RxQ_Ctrl3 or the receive queue number specified in the MAC_RxQ_Ctrl1[PTPQ] and RXQ#EN field in MAC_RxQ_Ctrl0 register enables the corresponding Rx queue. Programming MAC_RxQ_Ctrl1[TPQC] determines this. This type of receive packet routing is available when IEEE 1588 timestamp feature supports and the multiple receive queues are selected in the configuration. The VLAN tagged IEEE 1588 PTP over Ethernet receive packets are detected only when you disables 802.1AS mode (MAC_Timestamp_Control[AV8021ASMEN] = 0), otherwise this type of receive packets are routed as generic VLAN tagged Rx packets.

You can route the multicast or broadcast destination address receive packets that passes the destination address filter on the basis of the receive queue number specified in MAC_RxQ_Ctrl1[MCBCQ]the MCBCQ field of MAC_RxQ_Ctrl1 register when enabled through MAC_RxQ_Ctrl1[MCBCQEN] and the corresponding receive queue is enabled through RXQ#EN field in MAC_RxQ_Ctrl0. This type of Rx packet routing is available when you select the multiple receive queues in the configuration.

You can route the untagged receive packets on the basis of the receive queue number specified in MAC_RxQ_Ctrl1[UPQ] and RXQ#EN field in MAC_RxQ_Ctrl0 enables the corresponding receive queue. This type of receive packet routing is available when you select the multiple receive queues in the configuration.

The unicast destination address receive packets that fail the destination address filter can be routed based on the receive queue number specified in MAC_RxQ_Ctrl4[UFFQ] when enabled through MAC_RxQ_Ctrl4[UFFQE]UFFQE, MAC_Packet_Filter[RA] = 1 and the corresponding receive queue is enabled through RXQ#EN field in MAC_RxQ_Ctrl0. This type of receive packet routing is available when you select the multiple receive queues in the configuration.

You can route the multicast destination address receive packets that fails the destination address filter on the basis of the receive queue number specified in MAC_RxQ_Ctrl4[MFFQ] when MAC_RxQ_Ctrl4[MFFQE] enables it. MAC_Packet_Filter[RA] = 1 and RXQ#EN field in MAC_RxQ_Ctrl0 enables the corresponding receive queue. This type of receive packet routing is available when you select the multiple receive queues in the configuration.

The receive packet will route through the receive queue 0, if it is not classified in any of the defined packet type for routing.

### 72.9.6 Receive side arbitration between DMA and MTL

IP controller supports receive side arbitration between DMA and MTL.

After the current packet processing completes, the DMA channel controller fetches the next descriptor and after the descriptor fetching completes, the DMA channel controller evaluates the amount of data to transfer to the Rx buffer on the basis of the programmed PBL and receive buffer length. Accordingly, it requests the MTL to transfer the data.

After servicing the current request, the MTL receive queue arbitration scheme selects receive queue on the basis of the arbitration scheme (MTL_Operation_Mode[RAA]) and the weights programmed in queue <n> receive control register. The arbitration is done among queues for which DMA is ready to service. After the receive queue is selected, PBL amount of data is read out from that queue and route to the receive DMA channel on the basis of the receive channel selection criteria.

The arbitration occur after every PBL data transfer completes and descriptors are ready for processing from at least one DMA channel.

### 72.9.7 Transit side arbitration between DMA and MTL

IP supports transit side arbitration between DMA and MTL. Transit DMA channels and transit queues are always mapped directly because the number of transit DMA channels are always equal to the number of transit queues in MTL. For instance, each transit DMA pushes data into its respective transit queue assigned to it.

The data inside each transit queue is stored in packets. Therefore, if two DMAs are allowed to transfer data into the same queue, when a transit DMA starts a packet transfer, the other DMA cannot transfer data unless the previous packet is completely pushed-in. This means that the second DMA remains idle until the first packet is transferred. Hence, each DMA is always connected directly to its corresponding transit queue.

### 72.9.8 Audio video bridging

IP supports the audio video bridging in 100 Mbps mode only, which allows the transmission of time sensitive traffic over network. IP implements these protocol for supporting the AVB feature:

- IEEE 802.1Qbv-2015 (Enhancements to scheduling traffic)
- IEEE 802.3br (Interspersing traffic)
- IEEE 802.1Qbu (Frame preemption)

The queue 0 transmit path supports the strict priority algorithm, and it is used for best-effort traffic when transmit paths of additional queues use the credit-based shaper algorithm to support traffic management. For a queue, the credit-based shaper algorithm determines that a queue is available for transmission if these conditions are true:

- The queue contains one or more packets.
- The credit for the queue is positive per the algorithm
- AV traffic is received on all queues. IP can also receive untagged PTP packets in addition to AV traffic on any queue.

### 72.9.9 Queue modes

Transit and receive queues both handles the AV traffic.

Queueing is based on WRR (Weighted round robin) or WSP (Weighted strict priority) algorithm for generic traffic. It is based on CBS (Credit Base Shaper) and SP (Strict Priority) algorithm for AV traffic.

The receive queue is configured for generic or AV based routing, which depends on the RXQ#EN field of corresponding queue. Queueing is done on the basis of the VLAN tag priority. The VLAN tag priority must match the PSRQ field of MAC_RxQ_Ctrl2. The receive packets can route to a particular DMA channel on the basis of the DCS field of perfectly-matched MAC address register.

By default, the untagged packets in a generic traffic route to the receive queue specified in MAC_RxQ_Ctrl1[UPQ]. Queue 0 is the default value of MAC_RxQ_Ctrl1[UPQ]. You can override the default value with any other value, for the MAC_RxQ_Ctrl1[UPQ].

The AV control packets (tagged or untagged) route on the basis of MAC_RxQ_Ctrl1[AVCPQ]. The PTP over Ethernet packets route on the basis of MAC_RxQ_Ctrl1[PTPQ].

### 72.9.10 Queue priorities

You can program the priority of an receive queue in the corresponding field of MAC_RxQ_Ctrl2. Also, you must program the AV queue (high priority) as an higher priority than the best-effort queue (low priority).

### 72.9.11 Enhancements to scheduled traffic (EST)

The IEEE 802.1Qbv-2015 defines the schedule for each queues on every egress port which makes the implementation aware of traffic arrival schedule. This information blocks the lower priority traffic from transmission in this time window or slot. This ensures that the sender forward the scheduled traffic to receiver through all the network nodes with a deterministic delay.

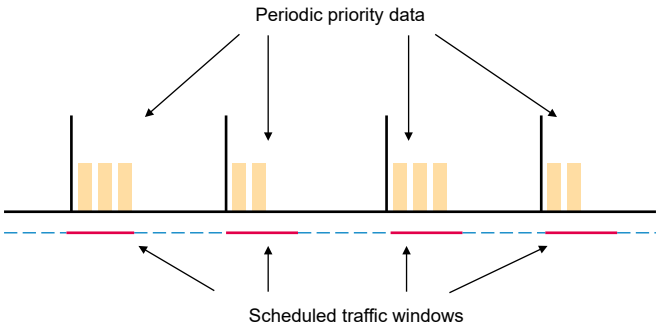



**Figure 397. Time aware shaper implementation for traffic scheduling**

An important requirement to achieve a low latency is to ensure that there are no interfering frames during the scheduled windows that are reserved for high priority traffic. The use of scheduled traffic imposes limitations when a transmission starts.

As shown in Figure 398 , if an interfering frame begins transmission just before the reserved time period starts, it can extend transmission into the reserved window, and potentially interfere with higher priority traffic. Therefore, a guard band whose size is equal to the largest possible interfering frame is required before the window starts.

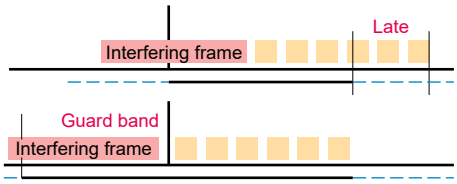



**Figure 398. Implementing a guard band to avoid delays due to interfering frames**

A larger guard band equates to a less efficient use of network bandwidth. However, this issue is addressed with the implementation of IEEE802.1Qbu (frame preemption). Frame preemption breaks the interfering frame into smaller fragments. Therefore, the guard band must be as large as the largest possible interfering fragment instead of the largest possible interfering frame.

During the guard band, only the frames that can complete the transmission of the entire frame before the next gate close event are permitted. This ensures that the high priority traffic can always start at the beginning of the window reserved for it.

#### 72.9.11.1 Frequently used terms in EST

These are some of the frequently used terms in the EST support and their definitions.

- Gate control list: The list in the hardware memory that hold the gate controls and the associated time intervals.
- Gate controls: For a given schedule (row in gate control list) there is a gate open (O) and gate close (C) state associated with each TC. The set of O or C values, whose width is same as configured TCs is called the gate controls. Example: CCOCOOCO means TC7=C, TC6=C, TC5=O and so on.
- Time interval: Time interval (in nanoseconds) is a 16, 20, or 24-bit configurable field in the gate control list that indicates the time for which the associated gate controls are valid.
- Base time register: Each gate control list is associated with a 64-bit base time register that holds the start time (in PTP format) for the list.

#### 72.9.11.2 Updates to the transmit scheduling to support EST

To support EST, these updates are required to transmit scheduling:

- Implementation of gate control list (GCL)
- Enforcing gate controls in the scheduler
- Utilizing gate open (O) duty cycle in the computation of idleSlope (CBS)

##### 72.9.11.2.1 Implementation of gate control list (GCL)

Figure 399 shows how the gate control list governs the gate close(C) and open(O) events on the basis of the schedule provided for each event. GCL has two parts:

- Time interval: Defines the time in nanoseconds for which the gate controls are valid and must apply before reading the next gate controls from the list. The configurable width of 16, 20 or 24-bit represents a maximum of 64us, 1ms, or 16ms schedule interval respectively. It supports left shift of the time interval up to 7 bits to be able to apply a multiplication factor from 1 to 128ns (in steps of 2(power)n). The maximum value (post shifting) of this field must be 999,999,999 ns.
- Gate control: Defines the open (O) represented by logic 1 or close (C) represented by logic 0 state for the gate of each TC.

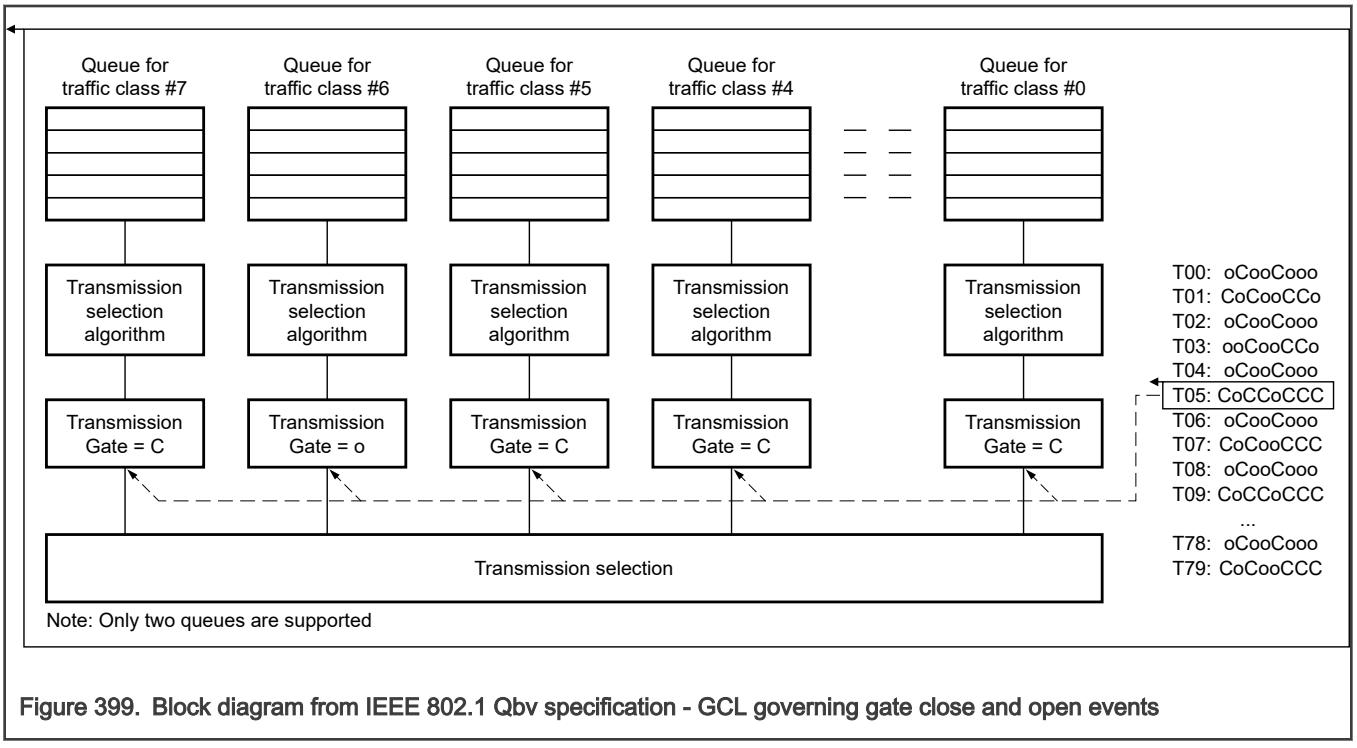



**Figure 399. Block diagram from IEEE 802.1 Qbv specification - GCL governing gate close and open events**

The implementation of GCL consists of these two gate control lists:

- HWOL - Hardware owned list which is a list for hardware access.
- SWOL - Software owned list which is a list for software access.

The access to these lists is mutually exclusive. Hardware sets the ownership to the list in MTL_EST_Status[SWOL]. Table 494 provides the implementation details of GCL.

**Table 494. External memory used for holding the two gate-control lists**

| Gate control (up to 8 bits) | Time interval (ns) 16, 20, or 24 bits | |
|---|---|---|
| OOCCCCCC | T0 | HWOL or SWOL |
| OOOOCCCC | T1 | |
| CCCCOOCC | T2 | |
| CCCCCCOO | T3 | |
| I | I | |
| I | I | |
| OCOCOCOO | | |
| OOOOCCCC | T last | |
| OOOOCCCC | T0 | SWOL or HWOL |
| OOOOCCCC | T1 | |
| OOCCCCCC | T2 | |
| OOOOCCCC | T3 | |
| I | I | |
| I | I | |
| OCOCCCCC | | |
| OOCCCCCC | T last | |

### 72.9.11.2.2 Registers related to gate control list

The following are the set of four registers (one for each GCL) related to GCL. These registers are implemented through Indirect addressing using MTL_EST_GCL_Control and MTL_EST_GCL_Data registers.

1. 64-bit Base time register (BTR)

2. 40-bit Cycle time register (CTR)

3. m-bit Time extension register (TER)

4. n-bit [Gate control] List length register (LLR) (n depends of the configured GCL depth)

#### 72.9.11.2.2.1 Base time register (BTR)

This is a 64-bit register that specifies the start time to execute the GCL. The format of the BTR is same as the PTP format (upper 32-bits holds time in seconds and lower 32 bits hold time in nanoseconds). After the execution of a given list starts, the implementation can update the value in BTR to indicate the next list execution start time.

#### 72.9.11.2.2.2 Cycle time register (CTR)

This is a 40-bit register that specifies the time at which the execution of the GCL must repeat. This register consists of an 8-bit value in seconds, and a 32-bit value in nanoseconds (similar to the PTP time format with truncated seconds register). For a given GCL, the start time is "Base Time" + N * "Cycle Time" where N is an integer value that represents the iteration number starting with 0 for first iteration. If the GCL execution takes longer than the cycle time, then the list is truncated at the cycle time and the subsequent loop begins at cycle time.

#### 72.9.11.2.2.3 Time extension register (TER)

This is a m-bit (where m = Configured time interval width + 7) register that specifies the amount of time (in nano seconds) the current GCL can extend before switching to the new GCL. This helps to avoid the execution of the small fragments of the current list before switching to a new list.

#### 72.9.11.2.2.4 List length register (LLR)

This is an n-bit register (when n is 7, 8, 9, 10, or 11 for a GCL configured depth of 64, 128, 256, 512, or 1024 respectively) that specifies the integer value of the length of the GCL (that is the number of valid rows in each GCL). The processing of the GCL stops after the number of rows read equals to the LLR value.

### 72.9.11.2.3 Transmission gating implementation

A bridge or an end station can enhance to allow transmission from each TC that is yet to be scheduled relative to a known timescale. To achieve this, a transmission gate is associated with each TC; the state of the transmission gate determines if you can select the queued frames for transmission.

For a TC, the transmission gate can be in one of these two states:

1. Open: Select queued frames for transmission, in accordance with the definition of the transmission selection algorithm associated with the TC.
2. Closed: Does not select queued frames for transmission.

A frame on a traffic class queue is not available for transmission if the transmission gate is in the closed state or if there is insufficient time available to transmit the entirety of that frame before the next gate-close event associated with that queue.

The implementation has visibility to the current schedule of gate controls and the immediate next schedule of the gate controls. So the maximum Gate Open period does not exceed the sum of the two Time Intervals. This is because, a frame is selected for transmission only if the gate is currently Open and the duration of gate open interval is greater than the time taken to transfer the entire frame.

The implementation must know the frame size before the transmission, so that you can avoid the transmission overruns and only the frames that can complete are scheduled at all times.

The implementation adequately compensates for the implementation delays in the data transfer from the buffer to the line by offsetting the current time with all the relevant delays (provided by MTL_EST_Control[CTOV]). This ensures that the schedule provided is always accurately implemented at the line.

You must ensure that the GCL slot interval is always greater than the expected packet size and overhead (scheduler delay, inter frame gap (IFG), and preamble, all combined).

### 72.9.11.3 Idle slope computation updates

When EST is enabled, credit accumulates only when the gate is open therefore, the effective data rate of the idleSlope must increase to reflect the duty cycle for the transmission gate associated with the queue.

The idleSlope is computed on the basis of the gate open time and oper cycle time values. Program the idleSlope registers (implemented one per CBS enabled TC) based on the following equation. The existing MTL register has sufficient field width to accommodate the new values for idleSlope.

idleSlope = (operIdleSlope(N) * OperCycle/GateOpenTime)

#### 72.9.11.3.1 Operational details of GCL

Write 1 to MTL_EST_Control[SSWL], so that the hardware can access the programmed gate control list. The first set of gate controls are applied when the current time is equal to the value in the base time register (BTR) and is held until the programmed "time interval" value.

One additional gate control event is always read ahead from the list, to avoid the transmission overruns. This enables the GCL to determine the next gate close events (if any) for the open TCs.

The scheduling is done on the basis of the gate open state and time interval of only the current and subsequent schedule. An internal accumulator adds the time intervals when gate controls are applied. BTR + Accumulator specifies the time at which the next set of gate controls are applied.

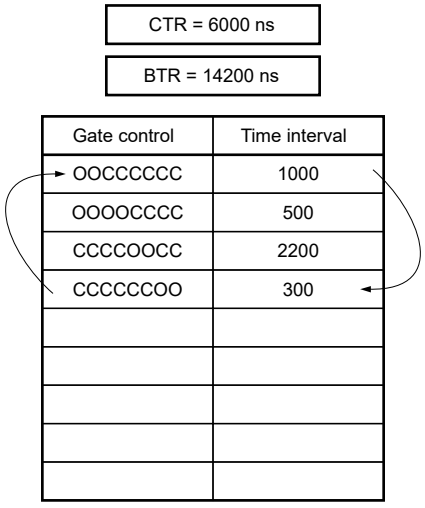



**Figure 400. GCL and associated registers - basetime and cycletime**

GCL is read progressively from the first row adhering to the schedule. The read operations continue until the list length (from LLR register) is reached and the execution of the list restarts at BTR + CTR time. At this point the value in CTR increments BTR to mark the beginning of a new cycle. In the absence of any gate controls, all the gates are in open state, during the execution of the list.

In cases where the execution time of the list is greater than the cycletime, the list is truncated and the next iteration starts when the current time equals BTR + CTR.

**Table 495. GCL and associated registers - BTR and CTR**

| Current time | Gate control applied | Accumulator value | BTR (with updates) |
|---|---|---|---|
| 14200 | OOCCCCCC | 1000 | 14200 |
| 15200 | OOOOCCCC | 1500 | 14200 |
| 15700 | CCCCOOCC | 3700 | 14200 |
| 17900 | CCCCCCOO | 4000 | 14200 |

*Table continues on the next page...*

Table 495. GCL and associated registers - BTR and CTR (continued)

| Current time | Gate control applied | Accumulator value | BTR (with updates) |
|---|---|---|---|
| 18200 | OOOOOOOO | 0 | 20200 |
| 20200 | OOCCCCCC | 1000 | 20200 |
| 21200 | OOOOCCCC | 1500 | 20200 |

Table 495 describes an example in which the execution starts at 14200 and the first set of gate controls "OOCCCCCC" are applied immediately. The time interval is 1000 ns, so the next set of gate controls are applied at 14200 (BTR) + 1000 (Accumulator) = 15200 ns . The above table shows that there are no gate controls available after the execution of the last gate control and before the next iteration of the loop. The gates are deemed in open state during that time period as depicted at 18200 current time.

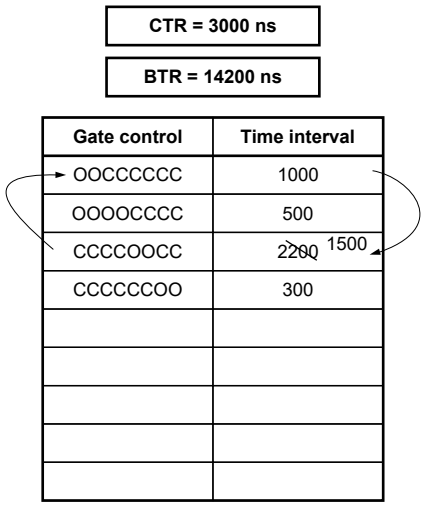



Figure 401. GCL and associated registers - BaseTime and CycleTime list execution time > CycleTime

Table 496 indicates that the list execution takes longer than the allocated CycleTime, so the list is truncated and the list starts from the BTR+CTR.

Table 496. GCL and associated registers - BTR and CTR, execution time > Cycle Time

| Current Time | Gate Control Applied | Accumulator Value | BTR (with updates) |
|---|---|---|---|
| 14200 | OOCCCCCC | 1000 | 14200 |
| 15200 | OOOOCCCC | 1500 | 14200 |
| 15700 | CCCCOOCC | 3700 | 14200 |
| 17200 | OOCCCCCC | 1000 | 17200 |
| 18200 | OOOOCCCC | 1500 | 17200 |
| 18700 | CCCCOOCC | 3700 | 17200 |

When you apply the third set of gate controls, BTR + CycleTime (17200) < BTR + Accumulator (17900), so the list is truncated and execution switches to a new iteration at 17200.

#### 72.9.11.3.2 Installing a new GCL

The switch to the new GCL can happen in one of the following ways when a new software programmed GCL is available and executed at the new BTR value:

- New base time aligned with current schedule
- New base time unaligned with current schedule

##### 72.9.11.3.2.1 New base time aligned with current schedule

If the choice of cycle time for the new gating cycle is unchanged from the cycle time for the current gating cycle, and if the BTR chosen for the new gating cycle (new BTR) is an integer multiple of the current cycle time (+ current BTR), then the new gating cycle exactly lines up with the old gating cycle, that is, the cycle start times for the new gating cycle is same as they would have been for the old configuration. This could be considered to be the ideal case and allows the new gating cycle to be installed and executed with no timing issues. The implementation completes the execution of an iteration of the current list and switches to the new list at the beginning of the BaseTime listed in the new list.

If (New base time > = Current time)

ConfigChangeTime = New BaseTime

Else If (New base time < Current time)

1. Set the BTRError

2. ConfigChangeTime = (New BaseTime + N* New CycleTime)

where N is the smallest integer for which the relation ConfigChangeTime >= CurrentTime and (N =< 8) is true.

When N > 8 the hardware cannot auto recover and the loop count value in BTRError reporting is set to 1111 requires the software to reprogram the new base time.

Figure 402 illustrates the installation of the new GCL along with the timelines.

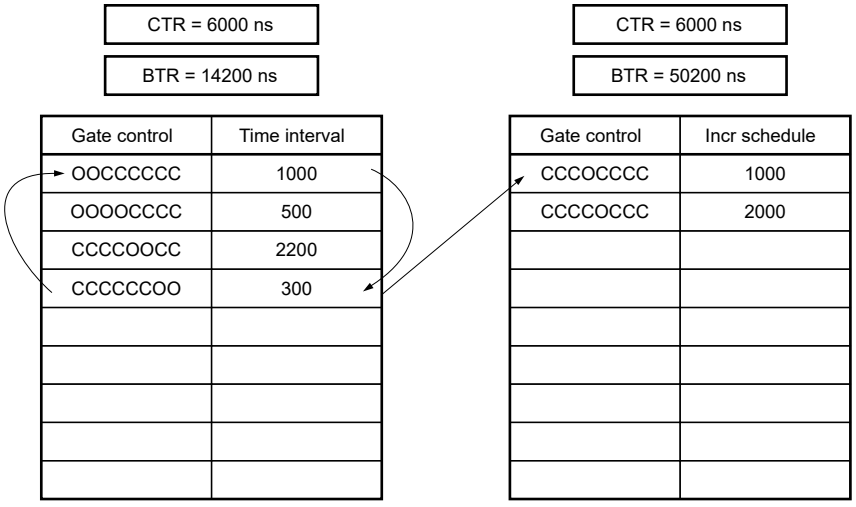



**Figure 402. Switching to a new configuration that is aligned with the existing configuration**

In the above example, after the sixth iteration of the first GCL, the BTR values of the old and new GCL are equal. At that point the new GCL is processed as a natural extension to the existing GCL.

Table 497. GCL and associated registers - BTR and CTR

| Current time | Gate control applied | Accumulator value | BTR (with updates) |
|---|---|---|---|
| 44200 | OOCCCCCC | 1000 | 44200 |
| 45200 | OOOOCCCC | 1500 | 44200 |
| 45700 | CCCCOOCC | 3700 | 44200 |
| 47900 | CCCCCCOO | 4000 | 44200 |
| 48200 | OOOOOOOO | 0 | 50200 |
| 50200 | CCCOCCCC | 1000 | 50200 |
| 51200 | CCCCOCCC | 3000 | 50200 |
| 53200 | OOOOOOOO | 0 | 56200 |

#### 72.9.11.3.2.2 New base time unaligned with current schedule

If the new cycletime differs from the current cycletime or new basetime in the future and is not an integer multiple of current cycletime, then the old and new cycles do not line up. When new basetime is reached ( when the new configuration is installed and starts to execute), the last old cycle is normally truncated to start the first new cycle. This could be undesirable if it results in a very short last old cycle; arguably it would be better to simply extend the penultimate old cycle by that small amount, rather than starting a very short cycle. The Cycle Time Extension Register (related to the current config list) allows this extension of the last old cycle to be done in a defined way; if the last complete old cycle ends normally in less than current Cycle Time Extension (TER) ns before the new base time, then the last complete cycle before new BaseTime is reached is extended so that it ends at new BaseTime.

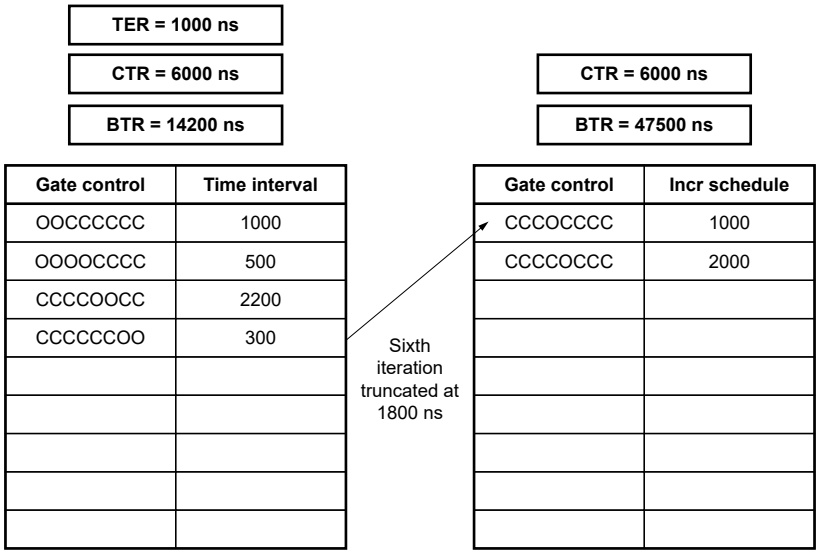



Figure 403. Switching to new list truncating the current list

At the end of the fifth iteration the Current time + Cycle time extension (TER) < New BTR so the sixth iteration of current configuration is started. During the sixth iteration of the current list when the new BTR value is smaller than the next schedule in the current list, it switches to the new list.

Table 498. Extending to new list by truncating the current list

| Current time | Gate control applied | Accumulator value | BTR (with updates) |
|---|---|---|---|
| 44200 | OOCCCCCC | 1000 | 44200 |
| 45200 | OOOOCCCC | 1500 | 44200 |
| 45700 | CCCCOOCC | 3700 | 44200 |
| 47500 | CCCOCCCC | 4000 | 44200 |
| 48500 | CCCCOCCC | 0 | 50200 |
| 50500 | OOOOOOOO | 1000 | 50200 |

Below is an example where the current config list is extended instead of starting a new iteration as the extension time of 800 ns is less than the allowed cycle extension time (TER) of 1000 ns.

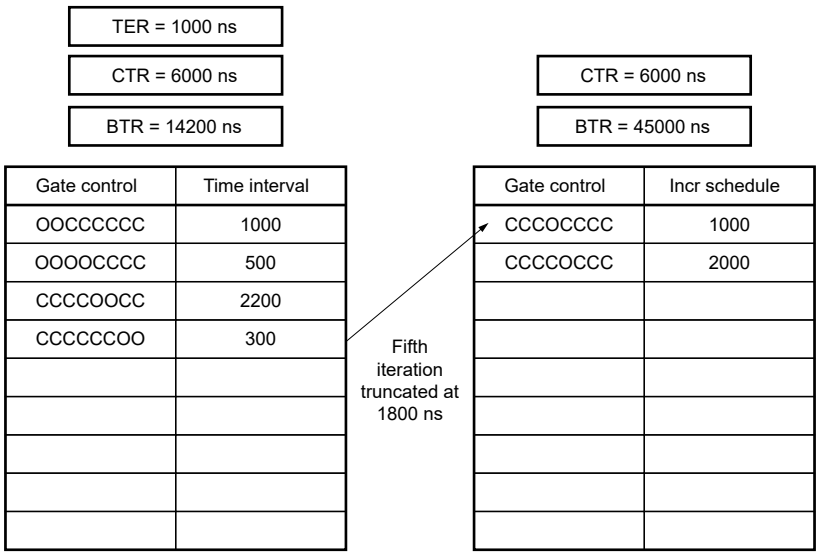



Figure 404. Switching to new list by extending the current list

Table 499. Switching to new list by extending the current list

| Current time | Gate control applied | Accumulator value | BTR (with updates) |
|---|---|---|---|
| 38200 | OOCCCCCC | 1000 | 38200 |
| 39200 | OOOOCCCC | 1500 | 38200 |
| 39700 | CCCCOOCC | 3700 | 38200 |
| 41900 | CCCCCCOO | 4000 | 38200 |
| 42200 | OOOOOOOO | 0 | 44200 |

*Table continues on the next page...*

Table 499. Switching to new list by extending the current list (continued)

| Current time | Gate control applied | Accumulator value | BTR (with updates) |
|---|---|---|---|
| 45000 | CCCOCCCC | 1000 | 45000 |
| 46000 | CCCCOCCC | 3000 | 45000 |
| 48000 | OOOOOOOO | 0 | 51000 |

#### 72.9.11.4 Impact of Transmit Scheduling Algorithms on EST

When EST is used in isolation, the Gate Control List manages the final open/close state of the Queues along with the checks carried out by the Transmission Selection Algorithm in MTL. As the Gate Controls operate on a predefined repetitive schedule, it is recommended to use Strict Priority or Credit Based Shaper (CBS) scheduling algorithms.

Other algorithms such as the Weighted Round Robin (WRR), Deficit Weighted Round Robin (DWRR) and Weighted Fair Queuing (WFQ) implement masking of the queues based on the current winning queue. The algorithm is applied only among the group of queues that open simultaneously. To ensure Queues whose gates are "Open" get priority, these algorithms are modified to treat "gate open" queues and "gate closed" queues as separate groups giving priority to gate open queues.

For example, consider 4 queues (Q3, Q2, Q1, Q0) with weights 4:3:2:1; Q3 and Q2 are in Open state in slot one slot, while Q1 and Q0 are in Open state in another slot. In this case, the scheduler works as follows:

1. In the first slot, the Q3 and Q2 are serviced in the ratio of 4:3 for the duration the slot is open.
2. In the second slot, the queues Q1 and Q0 are serviced in the ratio of 2:1.
3. Fresh arbitration is started every time a slot is opened.

In other words, the traffic does not get distributed in the intended ratio of 4:3:2:1; but as two groups with different ratios and only for the duration of the slot when the gates are open continuously

**NOTE**

See Programming guidelines for EST

### 72.9.12 Frame preemption (FPE)

Frame preemption breaks the interfering frame into smaller fragments. Therefore, the guard band needs to be only as large as the largest possible interfering fragment instead of the largest possible interfering frame. During the guard band, only the frames that can complete the transmission of the entire frame before the next gate close event are permitted. This ensures that the high priority traffic can always start at the beginning of the window reserved for it.

Preemption allows one or more higher priority (express) frames to interrupt the transmission of a lower priority (preemptable) frame; the preemptable frame transmission is resumed and completed after the express frame transmission is complete. To support frame preemption, the following two abstractions of the MAC are used:

- A preemptable MAC, called pMAC, which carries the preemptable traffic.
- An express MAC, called eMAC, which carries the express traffic.

In the implementation, only parts of the MAC that holds the state during preemption is replicated and represented as pMAC and eMAC. The MAC uses the following two ways to puts on hold, the transmission of the preemtable traffic, in the presence of express traffic:

- The MTL scheduler interrupts the preemptable traffic that is currently being transmitted. When the preemption capability is active, the MAC interrupts the transmission and reception of preemptable frames. A preempted fragment can be followed by zero or more express frames, before the continuation fragments. The preemptable frame can be fragmented any number of times, however, the minimum final and non-final fragment length criterion must be is met. However, interleaving of more than 1 preemptable packet is not permitted. This implies that if a preemptable packet is fragmented by an express

packet, another preemptable packet cannot be transferred until all the remaining fragments of the first preempted packet are transferred.

- The MTL scheduler prevents starting the transmission of preemptable traffic. When the preemption capability is inactive, the pMAC entity is disabled and only express traffic is transmitted or received. Frame preemption feature can be enabled by setting EFPE field of the MAC_FPE_CTRL_STS register.

If you are unable to schedule fragmented packets after a certain number of attempts because of some reason, there is a possibility that the remaining fragmented packets are dropped. For this, software can program MTL_EST_Control[LCSE] to increase number of iterations or MTL_EST_Control[DFBS] to disable dropping of packet.

**NOTE**

EST and FPE cannot be used at the same time.

### 72.9.12.1 GCL Modification to Support FPE

In the EST-only configuration, the GCL entry has up to 24 bits of Time Interval and up to 8 high order bits representing the Gate Open/Close state requirements as shown in the following figure.

| Gate control (up to 8 bits) | Time interval (ns) 16, 20 or 24 bits |
|---|---|
| OOCCCCCC | T0 |
| OOOOCCCC | T1 |
| CCCCOOCC | T2 |
| CCCCCCOO | T3 |
| \| | \| |
| \| | \| |
| OCOCOCOO | T126 |
| OOOOCCCC | T127 |

**Figure 405. Gate Control List When FPE is Disabled**

EST only supports SetGate operation, which implies that the gates are set to either open or close at a given time interval. However, when both Frame Preemption (FPE) and EST are enabled, the GCL also supports Set-and-Hold-MAC and Set-and-Release-MAC operations, in addition to the SetGate operation. To enable the support of hold and release operations, the format of the GCL is slightly changed while configuring and enabling the FPE. The first Queue (Q0) is always programmed to carry preemption traffic and therefore it is always Open. The bit corresponding to Q0 is renamed as Release/Hold MAC control. The TX Queues whose packets are preemptable are indicated by setting the PEC field of the MTL_FPE_CTRL_STS register. The GCL bit of the corresponding Queue are ignored and considered as always "Open". So, even if the software writes a "0" ("C"), it is ignored for such queues.

| Gate control (up to 7 bits) | Release or hold indication | Time interval (ns) 16, 20 or 24 bits |
|---|---|---|
| CCCOOOO | 0 | T0 |
| CCCCOOO | 0 | T1 |
| OCCCOOO | 1 | T2 |
| COCCOOO | 1 | T3 |
| CCOCOOO | 1 | T4 |
| CCCOOOO | 0 | T5 |
| CCCCOOO | 0 | T6 |
| | | |

**Figure 406. Gate Control List When FPE is Enabled**

When the Release/Hold bit transitions from a '0' to '1', a Set-and-Hold-MAC operation is performed. This marks the cease of the preemptable traffic. This is achieved by sending a Hold request to MTL "ha" ns in advance (where ha is the time interval mentioned in the Hold Advance (HADV) field of the MTL_FPE_Advance register). When the Release/Hold bit transitions from a '1' to '0' a Set-and-Release-MAC operation is performed. This marks the resuming of the preemptable traffic. This is achieved by sending a Release request to MTL "ra" ns in advance (where ra is the time interval mentioned in the Release Advance (RADV) field of the MTL_FPE_Advance register). The preemptable traffic is blocked for the time duration the Release/Hold bit is set to a '1' in the Gate Control List.

#### 72.9.12.2 Impact of Preemption on CBS

In Credit Base Shaper(CBS), the definition of "Transmit" is as follows:

- TRUE for the duration of frame transmission from the queue.
- FALSE when frame transmission from the queue is complete.
- When CBS algorithm is used along with frame preemption, the value of "Transmit" is TRUE only while the frame is being transmitted by the MAC. If the frame transmission is delayed or interrupted (for instance, a preemptable frame transmission is interrupted to allow the transmission of an express frame from a different queue, or the start of express frame is delayed because a preemptable frame is being transmitted) the value of "Transmit" is FALSE until transmission of the frame commences or is resumed.

Also, the value of "Transmit" is FALSE during the transmission of any overhead that is a consequence of frame preemption. For example, additional frame overhead (mCRC, Fragment Count) that is added to the preemptable frame.

At any given time, if there are no frames in the queue, and the value of Transmit is FALSE, and credit is positive value, the credit value is set to zero if here is no preemptable frame from the queue for which transmission is in progress but has been interrupted.

#### 72.9.12.3 mPacket Format

When the preemption capability is active, MAC sends mPackets to the PHY. An mPacket can be one of the following:

1. A express packet
2. A preemptable packet
3. An initial fragment of a preemptable packet
4. A continuation fragment of a preemptable packet

**Start mPacket Delimiter (SMD)**

The value of the SMD indicates whether the mPacket contains an express packet, the start of a preemptable packet (initial fragment or complete packet), or any of continuation fragments of a preemptable packet. Following table shows the valid SMD values.

Table 500. Possible SMD Values of mPacket

| mPacket Type | Notation | Frame Count | Value |
|---|---|---|---|
| verify packet | SMD-V | - | 0x07 |
| respond packet | SMD-R | - | 0x19 |
| express packet | SMD-E | - | 0xD5 |
| preemptable packet start | SMD-S0 | 0 | 0xE6 |
| | SMD-S1 | 1 | 0x4C |
| | SMD-S2 | 2 | 0x7F |
| | SMD-S3 | 3 | 0xB3 |
| continuation fragment | SMD-C0 | 0 | 0x61 |
| | SMD-C1 | 1 | 0x52 |
| | SMD-C2 | 2 | 0x9E |
| | SMD-C3 | 3 | 0x2A |

### frag_count

A frag_count is a modulo-4 counter that increments for each continuation fragment of the preemptable packet. The frag_count protects against mPacket reassembly errors by enabling detection of the loss of up to 3 packet fragments.

The frag_count field is present only in mPackets with SMD-C notation (continuation fragment). The frag_count is zero in the first continuation fragment of each preemptable packet.

Table 501. Possible frag_count Values

| frag_count | Value |
|---|---|
| 0 | 0xE6 |
| 1 | 0x4C |
| 2 | 0x7F |
| 3 | 0xB3 |

### mData

The contents of the packet from the MAC, starting with the first byte after the SFD to the last byte before the FCS are sent in the mData fields of one or more mPackets for that frame. The minimum size of the mData field is 60 bytes.

**CRC** The CRC field contains a cyclic redundancy check (CRC) and has an indication of the final mPacket of a frame. In the final mPacket of a frame, the CRC field contains the last 4 octets of the MAC frame (the FCS field).

For other mPackets, the CRC field contains an mCRC value. The mCRC is calculated on the octets of the packet from the first octet of the frame (the octet following the SFD of preemption frames) to the last octet of the packet transmitted in that mPacket by performing an XOR of the calculated 32 bit CRC value of the fragment and a value of 0x0000FFFF.

### Summary of Packet Formats

- Express Packet: 7bytes of PREAMBLE, SMD-E, Data, and CRC
- Complete Preemptable Packet: 7 bytes of PREAMBLE, <Current Preemptable packet SMD>, Data, CRC
- Initial Fragment (non-final) of Preemptable Packet: 7 bytes of PREAMBLE, <Current Preemptable packet SMD>, Data, mCRC
- Continuation fragments (non-final) of Preemptable packet: 6 bytes of PREAMBLE, <Current Preemptable continuation fragment SMD>, <Current Preemptable continuation fragment FC>, Data, mCRC
- Final fragment of Preemptable packet: 6 bytes of PREAMBLE, <Current Preemptable continuation fragment SMD>, <Current Preemptable continuation fragment FC>, Data, CRC

**Table 502. Current and Previous SMD Values**

| Previous Preemptable packet SMD | Current Preemptable packet SMD |
|---|---|
| SMD-S0 | SMD-S1 |
| SMD-S1 | SMD-S2 |
| SMD-S2 | SMD-S3 |
| SMD-S3 | SMD-S0 |

**Table 503. Current and Previous SMD Values**

| Previous Preemptable fragment SMD | Previous Preemptable fragment FC | Current Preemptable continuation fragment SMD | Current Preemptable continuation fragment FC |
|---|---|---|---|
| SMD-S0 | NA | SMD-C0 | FC0 |
| SMD-S1 | NA | SMD-C1 | FC0 |
| SMD-S2 | NA | SMD-C2 | FC0 |
| SMD-S3 | NA | SMD-C3 | FC0 |
| SMD-C0 | FC0 | SMD-C0 | FC1 |
| SMD-C0 | FC1 | SMD-C0 | FC2 |
| SMD-C0 | FC2 | SMD-C0 | FC3 |
| SMD-C0 | FC3 | SMD-C0 | FC0 |
| SMD-C1 | FC0 | SMD-C1 | FC1 |
| SMD-C1 | FC1 | SMD-C1 | FC2 |
| SMD-C1 | FC2 | SMD-C1 | FC3 |
| SMD-C1 | FC3 | SMD-C1 | FC0 |
| SMD-C2 | FC0 | SMD-C2 | FC1 |

*Table continues on the next page...*

Table 503. Current and Previous SMD Values (continued)

| Previous Preemptable fragment SMD | Previous Preemptable fragment FC | Current Preemptable continuation fragment SMD | Current Preemptable continuation fragment FC |
|---|---|---|---|
| SMD-C2 | FC1 | SMD-C2 | FC2 |
| SMD-C2 | FC2 | SMD-C2 | FC3 |
| SMD-C2 | FC3 | SMD-C2 | FC0 |
| SMD-C3 | FC0 | SMD-C3 | FC1 |
| SMD-C3 | FC1 | SMD-C3 | FC2 |
| SMD-C3 | FC2 | SMD-C3 | FC3 |
| SMD-C3 | FC3 | SMD-C3 | FC0 |

#### 72.9.12.4 Transmit Preemption

When FPE is enabled (setting EFPE=1 in MAC_FPE_CTRL_STS register), the MTL preempts a preemptable frame, when a "hold" request is asserted (EST/Qbv configured and enabled) express frames are available for transmission, that is, frame is present in MTL FIFO and is qualified for arbitration, after ensuring that the minimum mPacket mData field size is met. Therefore, preemption occurs only if at least 60 bytes of the preemptable frame have been transmitted and at least 64 bytes (including the frame CRC) remain to be transmitted.

The earliest starting position of preemption is controlled by the AFSZ field of the MTL_ FPE_CTRL_STS Register. Preemption does not occur until at least 64 * (1+ AFSZ) - 4 bytes of the preemptable frame have been sent.

When preemption occurs, all the preemptable queues are blocked and only the express queues are allowed to arbitrate (if more than one express queue has traffic) and transmit.

Continuation fragment of the preempted frame is the first frame to be transmitted after "release" request is asserted (EST/Qbv configured and enabled) and all the express traffic transmission completes.

**NOTE**

All the PTP packets should be transmitted as express packets

MTL communicates the following frame-type information to the MAC using a 2-bit preemption control signal on the MTI interface qualified by SoF and EoF.

- Express Frame
- Preemption Frame (Full or Fragment)
- Continuation Fragment (non-Final or Final)

Table 504. Preemption Control Values on MTI Interface for Various Frame Types

| Qualifier | Preemption Control Value | Frame Type |
|---|---|---|
| SoF | 00 | Start Express |
| SoF | 01 | Start Preemption |

*Table continues on the next page...*

Table 504. Preemption Control Values on MTI Interface for Various Frame Types (continued)

| Qualifier | Preemption Control Value | Frame Type |
|---|---|---|
| SoF | 10 | Continuation Fragment |
| SoF | 11 | Reserved |
| EoF | 00 | End of Frame |
| EoF | 01 | End of Fragment |

MTL should wait for the previous fragment status to be received before resuming the continuation fragments of a preempted frame. Fragment status is described in detail in the Tx Fragment Status section.

#### 72.9.12.4.1 MAC Tx Preemption

MAC supports preemption by implementing the functionality needed to generate the mPackets as described in “mPacket Format”.

Based on the Preemption Control value received on the MTI interface (qualified with SoF and EoF), MAC determines the frame type (shown in the above table 'Preemption Control Values on MTI Interface for Various Frame Types') and generates mPackets accordingly.

When the preemption capability is active, MAC replaces the SFD of a preemption packet with an SMD-S value. A 2-bit rolling frame count is encoded in the SMD-S value.

The SMD-E value is the same as the SFD value so the SFD of an express packet does not need to be replaced.

##### 72.9.12.4.1.1 Tx Fragment Status

MAC sends Tx fragment status to indicate successful transmission of fragmented mPackets.

In case of a transmission error (underflow, jabber, and so on) the frame status is sent (and not a fragment status) with an error indication along with all other relevant status fields. In case of receiving an error status for a transmitted fragment, the MTL drops the remaining fragments and does not send any more continuation fragments.

### 72.9.12.5 Receive Preemption

#### 72.9.12.5.1 MAC Receive Preemption

When FPE is enabled, the MAC Receiver passes the incoming packets and differentiates between Express packets and preemptable packets. An SMD containing an SMD-E indicates express packet, and SMD containing an SMD-S indicates the first mPacket of a preemptable packet.

If an mPacket containing an SMD-S is received when MAC has not completed receiving the previous preempted packet, MAC sets a CRC Error status for the previously received partial packet.

When an SMD-S is detected, MAC records the frame count indicated by the SMD and then begins sending data on the MRI interface.

The MAC checks the last four bytes of the mPacket . If the last four bytes of the mPacket do not match CRC, that indicates the end of the packet with or without a CRC error as per the CRC check result. If the last four octets of the mPacket match, that indicates that the packet was preempted.

An SMD containing an SMD-C indicates an mPacket that continues the data for a preempted packet. Upon receiving an SMD value of SMD-C, MAC checks the following:

1. A preempted packet is in progress
2. The frame count indicated by the SMD matches the frame count of the packet in progress
3. The frag_count value indicates the next fragment count.

If any of the above check fails, the mPacket is discarded and MAC sets a CRC Error status for the partially received packet.

If all the checks pass, the next fragment count is incremented modulo 4.

When a packet is preempted, the MAC saves the state of the partially received packet (filter check status, timestamp, length fields etc.) and will be able to process any received Express packets before the continuation fragment is received.

The MAC Receiver sends a "dummy status" for all the mPacket fragments successfully received and sends the Rx status with the final fragment. If an error is detected during any of the fragments the Rx status is sent and the fragment is marked as final fragment. All subsequent continuation fragments received for this packet are dropped in the MAC.

The MAC communicates the following frame type information to the MTL using a 2-bit preemption control signal.

1. Express Frame
2. Preemption Frame (Full or Fragment)
3. Continuation Fragment (non-Final or Final)

**Table 505. Preemption Control Values on MRI Interface for Various Frame Types**

| Qualifier | Preemption Control Value | Frame Type |
|---|---|---|
| SoF | 00 | Start Express |
| SoF | 01 | Start Preemption |
| SoF | 10 | Continuation Fragment |
| SoF | 11 | Reserved |
| EoF | 00 | End of Frame |
| EoF | 01 | End of Fragment |

#### 72.9.12.5.1.1 Data Alignment

When a received frame cannot be fragmented on any byte boundary, MAC retains the unaligned bytes of data in the previous fragment and resends them with the next fragment as shown in the following figure.

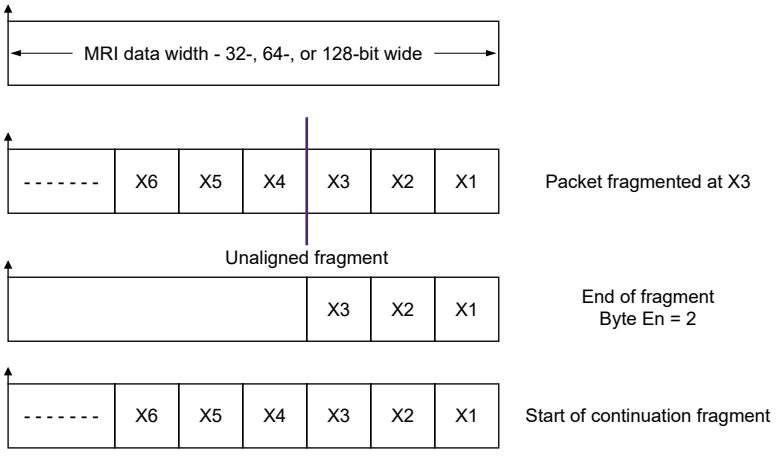



**Figure 407. Data Alignment Feature of MAC**

#### 72.9.12.5.1.2 MTL Receive Preemption

The MTL Rx must have at least 2 Rx queues to support FPE function as the preemptable packets and the express packets must be routed to separate Rx queues. The destination Rx queue of a received packet is controlled by MAC_RxQ_Ctrl[n] registers. Program these registers such that Preemptable traffic and express traffic are not routed to the same Rx queue. As the queue mapping for tagged packets is based on VLAN user-priority field, this implies that priority of preemptable and express packets are mutually exclusive. In other words, packets of a certain priority (traffic class) are either express or preemptable but cannot be both.

For the preemptable frames, in addition to the PSRQ (Priority Selected in Rx queue) based queue routing, a programmable "Frame Preemption Residue Queue" (FPRQ) is supported to route all other Preemption Packets received (Untagged, SA/ DA or VLAN Filter Fails forwarded due to RA being set or VTFE being reset).

**Table 506. Preemption Control Values on MRI Interface for Various Frame Types**

| Preemption Packet Type | Queue Routing |
|---|---|
| AV Tagged Packets passing the SA/DA and VLAN filters | PSRQ* |
| DCB/Generic Tagged Packets passing the SA/DA and VLAN filters | PSRQ** |
| Tagged Packets failing the SA/DA and VLAN filters without setting the Receive All (RA = 0) or setting VTFE = 1 | Dropped |
| All other packets OR when RA = 1 | FPRQ |

#### 72.9.12.5.1.3 MTL Receive Arbitration

On the ARI Interface, data is fetched from the MTL Rx FIFO based on the arbitration selected in the MTL_Operation_Mode register. Frame based arbitration can be used only when all the MTL Preemption Queues are operating in Store and Forward mode, else there will be loss of bandwidth on the ARI interface because all the fragments of a preemptable packet is not available. Therefore, any express packets received between the fragments are blocked until all the fragments are received and transferred; this defeats the purpose of express traffic.

When operating in either Threshold (cut through) or Store and forward modes of operation, PBL based arbitration is recommended over Frame based arbitration. In case of PBL based arbitration, the watermark check is always performed and the arbitration/ transfer of data in terms of chunks of "PBL" size of data which is almost similar to the concepts of "fragments". Therefore the express queue packets get blocked for less time as well as the ARI interface transfers the data without loss of efficiency.

### 72.9.12.6 Verify and Respond mPackets

When FPE function is present, the MAC can receive and detect the Verify and Respond mPackets, even when FPE is not enabled by software. When MAC detects valid Verify/Respond mPackets, it notifies the software by setting the RVER and RRSP fields of MAC_FPE_CTRL_STS register respectively. Optionally an interrupt can be generated. As such packets have empty (all-zero) data payload, they are dropped inside the MAC and not forwarded to the MRI.

Software can set the SVER and SRSP fields of MAC_FPE_CTRL_STS register to request MAC to transmit Verify and Respond mPackets respectively. Upon successful transmission of these frames, MAC clears the SVER/SRSP bits and sets the TVER & TRSP fields of MAC_FPE_CTRL_STS register. Optionally an interrupt can be generated when these events occur.

Software must ensure that frame preemption verify and response packet are not triggered at the same time. Trigger either of the packets and then wait for its completion before triggering another packet.

### 72.9.12.7 Frame Preemption and MMC Counter and Interrupt Registers

The following MMC counters and associated Interrupt registers are instantiated/present in the MAC when Frame Preemption feature is enabled.

Table 507. MMC Counters and Associated Interrupt Registers

| Frame Assembly Error Counter | Description | Associated MMC Counter |
|---|---|---|
| Frame Assembly Error Counter | A 32-bit counter that provides the number of MAC frames with reassembly errors on the Receiver, due to mismatch in the Fragment Count value. | MMC_Rx_Packet_Assembly_Err_Cntr |
| Frame SMD Error Counter | A 32-bit counter that provides the number of received MAC frames rejected due to arriving with an SMD-C when there was no preceding preempted frame. | MMC_Rx_Packet_SMD_Err_Cntr |
| Frame Assembly OK Counter | A 32-bit counter that provides the number of MAC frames that were successfully reassembled. | MMC_Rx_Packet_Assembly_Ok_Cntr |
| MAC Rx Fragment Counter | A 32-bit counter that provides the number of additional mPackets received due to preemption. | MMC_Rx_FPE_Fragment_Cntr |
| MAC Tx Fragment Counter | A 32-bit counter that provides the number of additional mPackets transmitted due to preemption. | MMC_Tx_FPE_Fragment_Cntr |
| Hold Request Counter | A 32-bit counter that maintains the count of number of times a hold request is given to MAC. | MMC_Tx_Hold_Req_Cntr |

#### 72.9.12.7.1 Additional Registers Associated With MMC Interrupts

Following are the additional registers associated with MMC interrupts for the MMC error counters:

- MMC_FPE_Tx_Interrupt
- MMC_FPE_Tx_Interrupt_Mask
- MMC_Rx_Interrupt
- MMC_FPE_Rx_Interrupt_Mask

### 72.9.13 Time-Based Scheduling

Time-based scheduling feature is suitable for traffic whose periodicity and rate are predictable. To improve the quality-of-service of such traffic,

- The transmit DMA fetches the packet from the host memory for transmission at designated time. This helps the software to setup the Transmit descriptors in advance even before packet is ready/available. It reduces the overhead on the software and avoids constant monitoring of the time and preparing descriptors just in time when the packet is targeted to be transmitted.
- The MAC transmits the packet only at the designated/pre-determined time even if the packets are fetched in advance. This helps in maintaining a constant transmission rate that can be consumed by the receiver station; therefore avoiding congestion and excessive buffering in the network.

The time-based scheduling feature supports fetching and launching an Ethernet packet at (or after) a pre-determined time. The time-based scheduling is supported only in the following modes/configurations:

- Full duplex mode
- Link speed is 100Mbps or higher

The following figure shows the time-based scheduling flow.

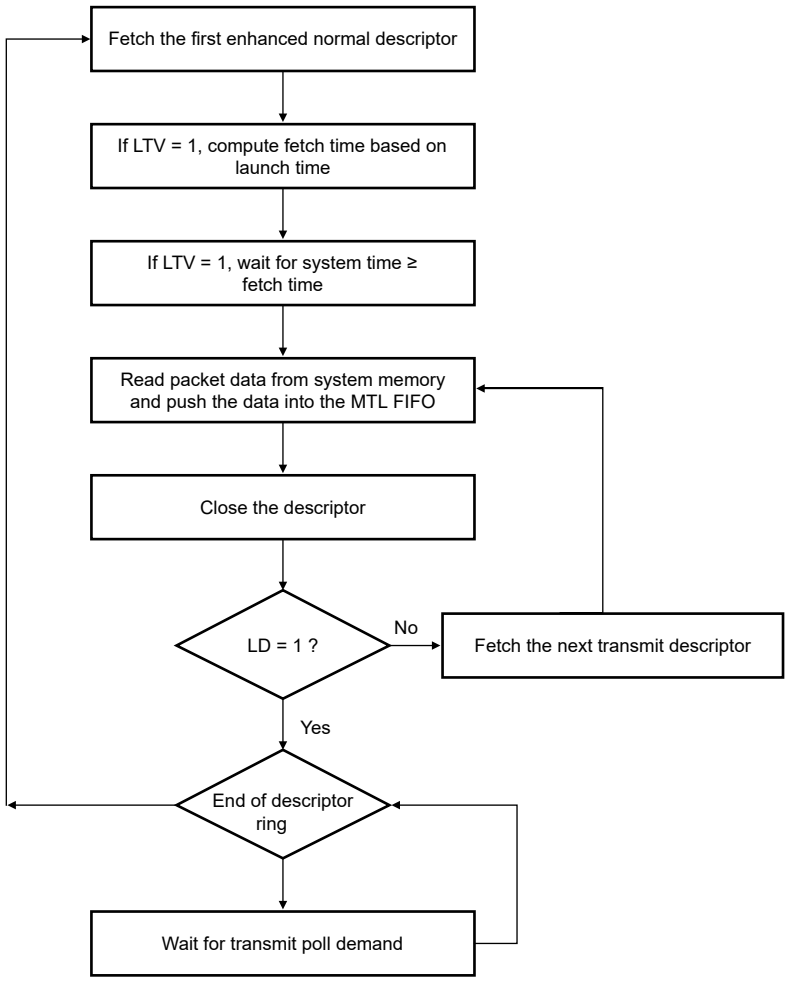



**Figure 408. Time-Based Scheduling Flow**

Following are the definitions specific to the time-based scheduling feature:

### 72.9.13.1 Launch time

The time beyond which MTL can schedule the packet for transmission. The launch time resets after the transmission of one frame. These are the two formats of the launch time.

**Normal or absolute format**:

In this format, the launch time is interpreted in the normal or absolute format if the ESTM bit of the MTL_TBS_CTRL register is set to 0.

The launch time is a 32-bit value, where most-significant 8-bits represent the time in seconds and the rest 24-bits represent the time in 256 ns. The launch time is compared against the IEEE 1588 based system or PTP time (bits[39:8]) and rolls over after 256 seconds.

The maximum value of the lower 32-bits of system time is 999,999,999 decimal (0x3B9AC9FF) and it wraps to 0 when reaching this value (representing a full second). Therefore, the maximum value of the lower 24-bits of the launch time (after multiplying by 256) must be 0x3B9AC9.

As the maximum value of launch time is 256 seconds,

- Launch time is greater than current system time when its value is between system time[39:8] and System time[39:8] + 128 sec.

- Launch time is less than current system time when its value is between System Time[39:8] + 128sec and System Time[39:8] + 256 sec, because this is a modulo 256 computation.

**EST or offset format**

In this format, the launch time is a offset value relative to the time which the BTR of the GCL list provided in the GCL slot number (GSN) indicates. The value in the BTR is always updated to the start time of the current loop of the GCL.

For each packet, the GSN value and the launch time value are specified in the enhanced transmit descriptor in the DMA configurations, or as control word in the MTL configurations.

The launch time offset is a 32-bit value; the upper 8-bits represent the time in seconds and the rest 24 bits represent the time in 256 ns, which is added to the BTR value corresponding to GCL slot number. The value of the launch time offset must be smaller than the value of the cycle time (specified in the CTR register that is implemented when EST is enabled). If the CTR is greater than or equal to 1s, the maximum value of the lower-24 bits of launch time offset must be 999,999,999 decimal (0x3B9AC9FF).

In this format, the launch time is a 64-bit value, which is interpreted as, Launch time = Launch GSN BTR[63:0] + (Launch time offset[31:0] << 8), which is compared with the system time [63:0].

GSN BTR is the Base Time value at which the Launch GSN loop started execution.

**GCL slot number (GSN)** A modulo 16 count of the GCL loop count is implemented and it is known as GSN. The count is incremented for every new GCL loop. Installation of new GCL list does not impact the count. Read MTL_EST_Status[CGSN] to obtain the current GCL slot number.

The maximum value of GSN is 15. Therefore the GSN values between MTL_EST_Status[CGSN] and MTL_EST_Status[CGSN]+8 represent current or future slots and all other GSN values are interpreted as elapsed slots or past slots. So, for the correct interpretation of time, GSN value must be between MTL_EST_Status[CGSN] and MTL_EST_Status[CGSN]+8.

### 72.9.13.2 Launch expiry time

**Normal or Absolute Mode**

In this mode, when MTL_TBS_CTRL[LEOV] = 1, MTL_TBS_CTRL[LEOS] determines the maximum amount of time a frame is eligible for launch, starting from the time the frame becomes eligible for launch.

The launch expiry offset is a 24-bit value defined in 256 ns units, with a maximum possible value of 999,999,999 ns (0x3B9AC9FF).

Launch expiry time = (Launch time[39:8] + LEOS[32:8]) * 256 ns

The packet with a specific launch time is considered as eligible for transmission when the launch time is less than the system time and (if MTL_TBS_CTRL[LEOV] = 1) the system time is less than the launch expiry time.

When the system time is greater than the launch expiry time, the frame is categorized as expired and it drops from the MTL FIFO.

**NOTE**

- For correct interpretation and meaningful operation, the fetch, launch, and launch expiry time must never set to a value larger than the Current system time + 128 sec; such a value is interpreted as time that has already elapsed.
- In Full Duplex mode, the frames dropped from the MTL FIFO have error summary (Bit 15) and excessive deferral (bit 3) of TxStatus set.

**EST/Offset Mode**

In EST mode, when MTL_TBS_CTRL[LEOV] = 1, MTL_TBS_CTRL[LEGOS] and MTL_TBS_CTRL[LEOS] determines the maximum amount of time a frame remains eligible for launch, starting from the time the frame becomes eligible for launch.

Launch Expiry Offset = (LEGOS: LEOS)

LEGOS holds the GSN offset (multiples of CTR time) and LEOS holds maximum value of CTR (sub CTR values) in ns.

Launch expiry offset is a 24-bit value defined in the units of 256 ns, with a maximum possible value of the smaller of 999,999,999 ns or CTR-1 ns.

The Launch expiry GSN is computed as follows:

Launch expiry GSN = (Launch GSN + LEGOS + CCMA) where, CCMA is the CTR carry due to modulo addition. This value is 1 if ((Launch time offset + LEOS) << 8) is equal to or greater than CTR.

When CCMA = 1, Launch expiry offset = (Launch time offset+ LEOS) - CTR

When CCMA = 0,

Launch expiry offset = (Launch time offset + LEOS).

Launch expiry time = Launch expiry GSN BTR [63:0] + Launch expiry time offset.

When MTL_TBS_CTRL[LEOV] is not 1, the launch expiry time is not checked.

When MTL_TBS_CTRL[LEOV] = 1, and

- the system time is greater than the launch expiry time, the frame drops from MTL FIFO. Then the frame is considered as expired.
- the launch time is smaller than the system time and the launch expiry time is greater than the system time, then the frame is considered eligible for launch.

**NOTE**

- Max value of MTL_TBS_CTRL[LEGOS] is 7. This indicates that when MTL_TBS_CTRL[LEOV] = 1, the frame has a maximum life time of <8 GCL loop iterations after it becomes eligible for launch.
- The slot number of the first GCL list that executes each time after EST is enabled, is zero.

### 72.9.13.3 Fetch time

This accounts for all possible delays in the DMA fetch operation and ensures that the frame is present in the MTL FIFO before the launch time.

If DMA_TBS_CTRL[FTOV] is 1, it indicates the fetch time for each packet. If DMA_TBS_CTRL[FTOV] is not 1, it indicates that the fetch time offset is not valid and DMA fetches packets without any time constraints.

#### Normal/Absolute mode

In this mode fetch time is derived or calculated by reducing the time specified in DMA_TBS_CTRL[FTOS] from the given launch time.

In Normal mode, the fetch launch time is computed as:

Fetch Time[39:8] = (Launch Time[39:8] - FTOS[31:8])

The fetch time is 32-bits and is compared against the system time[39:8] to determine whether it is eligible for fetching the frame:

- The fetch time is defined as greater than system time if the fetch time is in the range of system time[39:8]" and system time[39:8] + 128 sec. The frame is considered as not-eligible for fetch.
- The fetch time is defined as smaller than the system time if the fetch time is in the range of system time[39:8] + 128 sec and system time[39:8] + 256 sec. The frame is considered as eligible for fetch.

This is a modulo 256 computation.

#### EST/Offset mode

In this mode, the fetch GSN offset (DMA_TBS_CTRL[FGOS]) provides the slot number offset to deduct from the launch GSN. In this case the FTOS value must be smaller than 999,999,999 ns or CTR-1 ns.

If (Launch time offset >= FTOS):

Fetch time offset = ((Launch time offset - FTOS) * 256ns)

CBFS(CTR borrow for fetch subtraction) = 0

If (Launch time offset < FTOS) Fetch time offset = CTR + ((Launch time offset - FTOS) * 256ns)

CBFS (CTR borrow for fetch subtraction) = 1

The fetch GSN is computed as follows:

Fetch GSN = Launch GSN - FGOS - CBFS

Fetch time = Fetch GSN BTR[63:0] + Fetch time offset

The frame is eligible for DMA fetch when the fetch time is smaller than the system time.

#### DMA operations sequence

This is the sequence of operation when FTOV = 1:

1. Fetch the first enhanced normal descriptor (FD = 1).
2. Compute the fetch time on the basis of the launch time and wait for the system time to be greater than fetch time, if LTV = 1 and fetch is enabled.
3. Read the frame (data) from the host memory and transfer to MTL FIFO.
4. Close the normal descriptor.
5. Fetch the next normal descriptor (if the previous descriptor was not the last).
6. Repeat steps 4 to 6, until the last descriptor of the frame (LD = 1).

After the last descriptor of a frame, program the next enhanced normal descriptor with a new launch time and with LTV = 1. Otherwise, process the subsequent frames without any time restrictions.

See Programming the launch time in time-based scheduling for more information.

## 72.10 TCP/IP Offloading Features

This section and all other sections, along with respective sub-sections are Synopsys Proprietary. Used with permission.

### 72.10.1 Transmit Checksum Offload Engine

The MAC has a Checksum Offload Engine (COE) to support checksum calculation and insertion in the Transmit path, using which, the software can offload the task of checksum insertion to the hardware. In the transmit path MAC calculates the checksum and inserts it in the Tx packet. This feature helps in reducing the load on the software and can improve the overall throughput of the system. This feature is supported for only Q0 queue.

The checksum offload engine module supports two types of checksum calculation and insertion. The checksum engine can be controlled for each packet by setting the CIC bits (TDES3 Bits[17:16]).

Note: The checksum for TCP, UDP, or ICMP is calculated over a complete packet, and then inserted into its corresponding header field. Because of this requirement, when this function is enabled, the Tx FIFO automatically operates in the store-and-forward mode even if IP is configured for Threshold (cut-through) mode.

#### 72.10.1.1 IP Header Checksum Engine

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit Header Checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the Type field of Ethernet packet has the value 0x0800 and the Version field of IP datagram has the value 0x4. The checksum field of the input packet is ignored during calculation and replaced with the calculated value.

**NOTE**

IPv6 headers do not have a checksum field. Therefore, the COE does not modify the IPv6 header fields.

The result of this IP header checksum calculation is indicated by the IP Header Error status bit in the Transmit status (Bit 0 in Table 19-12 on page 1330). This status bit is set whenever the values of the Ethernet Type field and the Version field of IP header are not consistent, or when the Ethernet packet does not have enough data, as indicated by the IP header Length field. In other words, this bit is set when an IP header error is asserted under the following circumstances:

- For IPv4 datagrams:
  - The received Ethernet type is 0x0800, but the Version field of IP header is not equal to 0x4.
  - The IPv4 Header Length field indicates a value less than 0x5 (20 bytes).
  - The total packet length is less than the value given in the IPv4 Header Length field.
- For IPv6 datagrams:
  - The Ethernet type is 0x86dd but the IP header Version field is not equal to 0x6.
  - The packet ends before the IPv6 header (40 bytes) or extension header (as given in the corresponding Header Length field in an extension header) is completely received.

#### 72.10.1.2 TCP/UDP/ICMP Checksum Engine

The TCP/UDP/ICMP Checksum Engine processes the IPv4 or IPv6 header (including extension headers) and determines whether the encapsulated payload is TCP, UDP, or ICMP. The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. The Tx COE can work in the following two modes:

- The TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the Checksum field of the input packet. This engine includes the Checksum field in the checksum calculation, and then replaces the Checksum field with the final calculated checksum.
- The engine ignores the Checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation, and overwrites the checksum field with the final calculated value.

**NOTE**

For ICMP-over-IPv4 packets, the Checksum field in the ICMP packet must always be 16'h0000 in both modes, because pseudo-headers are not defined for such packets. If it does not equal 16'h0000, an incorrect checksum may be inserted into the packet.

The result of this operation is indicated by the Payload Checksum Error status bit in the Transmit Status vector (Bit 12 in Table 19-3 on page 1324). This engine sets the Payload Checksum Error status bit when it detects that the packet has been forwarded to the MAC Transmitter engine in the store-and-forward mode without the end of packet (EOP) being written to the FIFO, or when the packet ends before the number of bytes indicated by the Payload Length field in the IP Header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field. Following table describes the functions supported by Transmit Checksum Offload engine based on the packet type. When the MAC does not insert the checksum, it is indicated as "No" in the table.

**NOTE**

You should not enable checksum insertion for IPv4 or IPv6 packets that are greater than the frame size constraint specified in Description of Transmit Checksum Offload Engine because it may result in incorrect checksum insertion or unexpected behavior.

**Table 508. Transmit Checksum Offload Engine Functions for Different Packet Types**

| Packet Type | Hardware IP headerchecksum insertion | Hardware TCP/ UDPchecksum insertion |
|---|---|---|
| Non-IPv4 or IPv6 packet | No | No |
| IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K Packets is enabled in MAC) but less than or equal to the frame size constraint specified in Transmit Checksum Offload Engine. | Yes | Yes |

*Table continues on the next page...*

Table 508. Transmit Checksum Offload Engine Functions for Different Packet Types (continued)

| Packet Type | Hardware IP headerchecksum insertion | Hardware TCP/ UDPchecksum insertion |
|---|---|---|
| IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K Packets is enabled in MAC) but less than or equal to the frame size constraint specified in Transmit Checksum Offload Engine. | Not Applicable | Yes |
| IPv4 with TCP, UDP, or ICMP | Yes | Yes |
| IPv4 packet has IP options (IP header is longer than 20 bytes) | Yes | Yes |
| Packet is an IPv4 fragment | Yes | No |
| IPv6 packet with the following next header fields in main or extension headers:<br>• Hop-by-hop options (in IPv6 main header)<br>• Hop-by-hop options (in IPv6 extension header)<br>• Destinations options<br>• Routing (with segment left 0)<br>• Routing (with segment left > 0)<br>• TCP, UDP, or ICMP<br>• Authentication<br>• Any other next header field in main or extension headers | • Not Applicable<br>• Not Applicable<br>• Not Applicable<br>• Not Applicable<br>• Not Applicable<br>• Not Applicable<br>• Not Applicable<br>• Not Applicable | • Yes<br>• No<br>• Yes<br>• No<br>• No<br>• Yes<br>• No<br>• No |
| IPv4 packet has TCP header with Options fields | Yes | Yes |
| IPv4 Tunnels:<br>• IPv4 packet in an IPv4 tunnel<br>• IPv6 packet in an IPv4 tunnel | • Yes (IPv4 tunnel header)<br>• Yes (IPv4 tunnel header) | • No<br>• No |
| IPv6 Tunnels:<br>• IPv4 packet in an IPv6 tunnel<br>• IPv6 packet in an IPv6 tunnel | • Not applicable<br>• Not applicable | • No<br>• No |
| IPv4 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled). | Yes | Yes |
| IPv6 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled). | Not applicable | Yes |
| IPv4 frames with security features (such as encapsulated security payload) | Yes | No |
| IPv6 frames with security features (such as encapsulated security payload) | Not applicable | No |

### 72.10.2 Receive Checksum Offload Engine

IP provides the Checksum Offload Engine that is used to detect any error in an IPv4 or IPv6 packet in the receive path. The MAC verifies the checksum field of the received packet with the internally calculated checksum and provides the status.

The Receive Checksum Offload engine is used to detect errors in IP packets by calculating the header checksum and further matching it with the received header checksum. This engine also identifies a TCP, UDP, or ICMP payload in received IP packets and calculates the checksum of such payloads appropriately.

Here, both IPv4 and IPv6 packet in the received Ethernet packets are detected and processed for data integrity. The MAC receiver identifies IPv4 or IPv6 packets by checking for value 0x0800 or 0x86DD, respectively, in the Type field of the received Ethernet packet. This identification is applicable to single VLAN-tagged packets. It is also applicable to double VLAN-tagged packets when the Enable Double VLAN Processing option is selected and the EDVLP bit of the MAC_VLAN_Tag register is set.

The Rx COE calculates the IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of this operation (pass or fail) is given to the RFC module for insertion into the receive status word. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received packet does not have enough bytes, as indicated by the Length field of the IPv4 header (or when fewer than 20 bytes are available in an IPv4 or IPv6 header).

This engine also identifies a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP, UDP, or ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a Payload Checksum Error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not match the expected payload length given in the IP header.

Following table describes the functions supported by the Rx COE based on the packet type. When the payload of an IP packet is not processed (indicated as "No" in the table), the information (whether the checksum engine is bypassed or not) is given in the receive status.

**NOTE**

The MAC does not append any payload checksum bytes to the received Ethernet packets.

**Table 509. Receive Checksum Offload Engine Functions for Different Packet Types**

| Packet type | Hardware IP header checksum checking | Hardware TCP/UDP/ICMP checksum checking |
|---|---|---|
| Non-IPv4 or IPv6 | No | No |
| IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K Packets is enabled in the MAC) | Yes | Yes |
| IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K Packets is enabled in the MAC) | Not Applicable | Yes |
| IPv4 with TCP, UDP, or ICMP | Yes | Yes |
| IPv4 header's protocol field contains a protocol other than TCP, UDP, or ICMP | Yes | No |

*Table continues on the next page...*

**Table 509. Receive Checksum Offload Engine Functions for Different Packet Types (continued)**

| Packet type | Hardware IP header checksum checking | Hardware TCP/UDP/ICMP checksum checking |
|---|---|---|
| IPv4 packet has IP options (IP header is longer than 20 bytes) | Yes | Yes |
| Packet is an IPv4 fragment | Yes | No |
| IPv6 packet with the following next header fields in main or extension headers:<br>• Hop-by-hop options (in IPv6 main header)<br>• Hop-by-hop options (in IPv6 extension header)<br>• Destinations options<br>• Routing (with segment left 0)<br>• Routing (with segment left > 0)<br>• TCP, UDP, or ICMP<br>• Any other next header field in main or extension headers | • Not Applicable<br>• Not Applicable<br>• Not Applicable<br>• Not Applicable<br>• Not Applicable<br>• Not Applicable<br>• Not Applicable | • Yes<br>• No<br>• Yes<br>• Yes<br>• No<br>• Yes<br>• No |
| IPv4 packet has TCP header with Options fields | Yes | Yes |
| IPv4 Tunnels:<br>• IPv4 packet in an IPv4 tunnel<br>• IPv6 packet in an IPv4 tunnel | • Yes (IPv4 tunnel header)<br>• Yes (IPv4 tunnel header) | • No<br>• No |
| IPv4 Tunnels:<br>• IPv4 packet in an IPv6 tunnel<br>• IPv6 packet in an IPv6 tunnel | • Not Applicable<br>• Not Applicable | • No<br>• No |
| IPv4 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled) | Yes | Yes |
| IPv6 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled) | Not Applicable | Yes |
| IPv4 frames with security features (such as encapsulated security payload) | Yes | No |

*Table continues on the next page...*

Table 509. Receive Checksum Offload Engine Functions for Different Packet Types (continued)

| Packet type | Hardware IP header checksum checking | Hardware TCP/UDP/ICMP checksum checking |
|---|---|---|
| IPv6 frames with security features (such as encapsulated security payload) | Not Applicable | No |

Set the IPC bit of the MAC_Configuration register for enabling Receive Checksum offload.

## 72.11 MAC Management Counters

This section is Synopsys Proprietary. Used with permission.

IP supports storing the statistics about the received and transmitted packets in registers that are accessible through the application/software. The MMC module maintains a set of registers for gathering statistics on the received and transmitted packets. The MMC counters are free running.

For MMC register details, see registers "MMC_Control" to "MMC_Rx_FPE_Fragment_Cntr" in EMAC register descriptions.

## 72.12 Flow Control

This section and its sub-sections are Synopsys Proprietary. Used with permission.

### 72.12.1 Transmit Flow Control

The Transmit Flow Control involves transmitting Pause packets in full-duplex mode and backpressure in half-duplex mode to control the flow of packets from the remote end.

#### 72.12.1.1 Flow control in Full duplex mode

IP supports the IEEE802.3x Pause Packets.

Pause packet structure is shown in the following table.

Table 510. Pause Packet Fields

| Field | Description |
|---|---|
| DA | Contains the special multicast address |
| SA | Contains the MAC address 0 |
| Type | Contains 8808 |
| MAC Control opcode | Contains 0001 for IEEE 802.3x Pause Control packets; 0101 for PFC packets |
| PT | Contains Pause time specified in the PT field of the MAC_Q#_Tx_Flow_Ctrl register |

#### 72.12.1.2 Pause Packet Control

When the FCB bit is set, the MAC generates and transmits a single Pause packet. If the FCB bit is set again after the Pause packet transmission is complete, the MAC sends another Pause packet irrespective of whether the pause time is complete or not. To extend the pause or terminate the pause prior to the time specified in the previously-transmitted Pause packet, the application should program the Pause Time register with appropriate value and then again set the FCB bit.

#### 72.12.1.3 Flow Control in Half-Duplex Mode

In half-duplex mode, the MAC uses the deferral mechanism for the flow control (backpressure). When the application requests to stop receiving packets, the MAC sends a JAM pattern of 32 bytes when it senses a packet reception, provided the transmit flow control is enabled. This results in a collision and the remote station backs off. If the application requests a packet to be transmitted, it is scheduled and transmitted even when the backpressure is activated. If the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur), the remote stations abort the transmission because of excessive collisions.

Following table describes the flow control in the Tx path for Queue 0 based on the setting of the following bits:

- EHFC bit of MTL_RxQ0_Operation_Mode register
- TFE bit of MAC_Q0_Tx_Flow_Ctrl register
- DM bit of MAC_Configuration register

Flow control is similar for all queues.

**Table 511. Tx MAC Flow Control**

| EHFC | TFE | DM | Description |
|---|---|---|---|
| x | 0 | x | The MAC transmitter does not perform the flow control or backpressure operation. |
| 0 | 1 | 0 | The MAC transmitter performs back-pressure when Bit 0 of MAC_Q0_Tx_Flow_Ctrl register is set or the sideband signal sbd_flowctrl_i is 1. |
| 1 | 1 | 0 | The MAC transmitter performs back-pressure when Bit 0 of MAC_Q0_Tx_Flow_Ctrl register is set or the sideband signal sbd_flowctrl_i is 1. In addition, the MAC Tx performs back-pressure when Rx Queue level crosses the threshold set by Bits[10:8] of MTL_RxQ0_Operation_Mode register. |
| 0 | 1 | 1 | The MAC transmitter sends the Pause packet when Bit 0 of MAC_Q0_Tx_Flow_Ctrl register is set or the sideband signal sbd_flowctrl_i is 1. |
| 1 | 1 | 1 | The MAC transmitter sends the Pause packet when Bit 0 of MAC_Q0_Tx_Flow_Ctrl register is set or the sideband signal sbd_flowctrl_i is 1. In addition, the MAC Tx sends a Pause packet when Rx Queue level crosses the threshold set by Bits[10:8] of MTL_RxQ0_Operation_Mode register. |

#### 72.12.1.4 Enabling Transmit Flow Control

To independently enable Transmit Flow Control for each Tx queue, set the TFE bit in the MAC_Q#_Tx_Flow_Ctrl register.

### 72.12.2 Receive Flow Control

In the Receive path, the Flow Control is functional only in the full-duplex mode. If any Pause packet is received in the half-duplex mode, the packet is considered as a normal control packet.

**NOTE**

Receive pause packets should have a frame size of 64 bytes.

The Receive Flow Control is implemented by the MAC based on the bit value of the respective register, and the destination address and different fields of the received packet.

Following table describes the flow control in the Rx path based on the setting of the following bits:

- RFE bit of MAC_Rx_Flow_Ctrl register
- DM bit of MAC_Configuration register

Table 512. Rx MAC Flow Control

| RFE | DM | Description |
|---|---|---|
| 0 | x | The MAC receiver does not detect the received Pause packets. |
| 1 | 0 | The MAC receiver does not detect the received Pause packets but recognizes such packets as Control packets. |
| 1 | 1 | The MAC receiver detects or processes the Pause packets and responds to such packets by stopping the MAC transmitter. |

#### 72.12.2.1 Enabling Receive Flow Control

To enable Pause Flow Control, set the RFE bit in the MAC_Rx_Flow_Ctrl register.

## 72.13 Loopback Mode

This section is Synopsys Proprietary. Used with permission.

The MAC supports Loopback of transmitted packets to its receiver.

The following are some guidelines for using the loopback mode:

- Enable loopback only with the full-duplex mode. In half-duplex mode, the carrier sense signal (crs) or collision (col) signal inputs get sampled which may result into issues such as packet dropping.
- If the loopback mode is enabled without connecting a PHY chip (for example, in FPGA setup), you should externally generate the Tx and Rx clocks and provide these clocks to the MAC.
- Do not loop back big packets. Big packets (>1522 bytes) may get corrupted in the loopback FIFO.

To enable this feature, program the LM bit of the MAC_Configuration register.

You can enable loopback for all PHY interfaces. The data is always looped back through internal asynchronous FIFO on to the internal Receive MII or GMII interface, irrespective of which PHY interface is selected. The loopback data is also passed through the corresponding transmit PHY interface block, on to the Ethernet line.

## 72.14 Automotive Safety Features

This section and all its sub-sections are Synopsys Proprietary. Used with permission.

IP supports the automotive safety feature.

### 72.14.1 Error Correction Code (ECC) Protection for Memories

The Error Correction Code (ECC) block can correct single-bit error and detect double-bit error.

At the write interface, ECC checkbits are generated by computing ECC on the contents of the data bus and respective address is appended with the data that is written to the memory.

At the read interface, ECC checkbits are recomputed on the content of the read data and the respective address is compared with the received checkbits in the memory.

Following figure shows the block diagram of ECC protection for memories

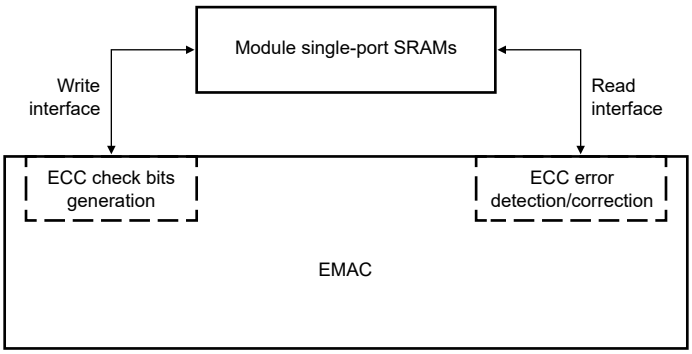



Figure 409. Block Diagram of ECC Protection for Memories

#### 72.14.1.1 Handling the Address Mismatch

The ECC is calculated over the sum of memory data and memory location address. However only data is written in the memory along with the ECC(address is excluded).

On the read interface, the ECC is checked over the sum of memory DATA, memory ECC, and Internally generated address. When an ECC correctable error is detected over the range of the address bit position, it is treated as an uncorrectable error. This is because data might have been read from a different location.

#### 72.14.1.2 Diagnostic Support for the Error Management

Following statistics are provided for monitoring the error behavior on each of the memory blocks.

- Error status provided to management
  - A separate status for correctable, uncorrectable, and address mismatch is specified in the MTL_ECC_Interrupt_Status register.
  - Memory locations at which correctable and uncorrectable errors are detected is specified in the MTL_ECC_Err_Addr_Status register. In addition, a control bit (MEEAO field of MTL_ECC_Control register) decides, whether the first errored memory address is reported or the latest errored memory address is reported.
  - MTL_ECC_Err_Cntr_Status register has separate counters to count the number of correctable and uncorrectable errors
- Interrupts provided to management
  - Separate interrupts are generated for correctable and uncorrectable errors.
  - sbd_sfty_ue_intr_o interrupt is generated when uncorrectable errors are detected and these errors cannot be masked.
  - sbd_sfty_ce_intr_o interrupt is generated when correctable errors are detected. The module sets the respective interrupt enable bits (in the DMA/MTL_ECC_Interrupt_enable register).
  - To find the root cause of the error, read the DMA/MTL_Safety_Interrupt_Status and DMA/MTL_ECC_Interrupt_Status registers.
  - To clear the interrupt, write 1 in the respective interrupt status bit in DMA/MTL_ECC_Interrupt_Status registers.

**NOTE**

- The status/counters/interrupts are generated per memory.
- MTL Tx and Rx memory is logically divided into multiple queues. But, ECC diagnostics (status/counter/interrupts) are common to all queues.

#### 72.14.1.3 ECC Error Injection Capabilities

IP supports error injection capabilities for each memory as a static configuration. The position where the errors are injected in the data word is random. For each memory, 3 control bits are provided to inject errors. The 3 bits are:

- A single bit to enable error injection
- Two bits to indicate the type of error to be injected ❑ 00: 1 bit error
  - 01: 2 bit error
  - 10: 3 bit error
  - 11: 1 bit error in address field

The control bit descriptions for

- MTL Tx/Rx and DMA TSO memory are specified in the MTL_DBG_CTL register, for and for R.
- MTL EST memory are specified in the MTL_EST_GCL_Control register
- Rx parser memory are specified in the MTL_RXP_Indirect_Acc_Control_Status register

**NOTE**

While using debug mode for ECC error injection:

- There should be no traffic in the module
- When multiple CSR writes are required for writing single data word into the memory, the application should ensure that all the CSR writes corresponding to one memory write maintains the same value for the error injection control word.
- See Programming guidelines for ECC protection for memories

### 72.14.2 Finite State Machine(FSM) Parity and Timeout Protection

The FSM protection feature supports FSM parity and time out protection.

#### 72.14.2.1 FSM State Parity Protection

Odd parity is implemented on all the FSM state register bits. Parity is monitored for every clock, after the reset is de-asserted. When a bit flips due to transient errors or permanent faults, the erroneous FSM is set to its default state and an uncorrectable error is indicated.

When the FSM state parity protection is enabled, by setting PRTYEN field of MAC_FSM_Control Register, the FSM state error is indicated by setting the FSMPES field of the MAC_DPP_FSM_Interrupt_Status register. Also, the Safety Interrupt (sbd_sfty_ue_intr_o) is asserted when an FSM Error is detected.

Error Injection mode is also supported for FSM parity error check. Program the Error Injection enable for the respective clock domain denoted by [23:16] bits of MAC_FSM_Control Register.

#### 72.14.2.2 FSM Timeout Protection

The FSM Timeout feature provides a mechanism to ensure that all FSMs in the module can complete a transaction and return to a known completion state (IDLE or any other state that indicates completion of a transaction/transfer) within the programmed timeout value.

Program the TMR field of MAC_FSM_ACT_Timer register with a value that indicates the number of CSR cycles required to generate a 1 microsecond tic. This microsecond tic is internally used to generate the programmed timeout duration. Two timeout tics (normal mode timeout and large mode timeouts) are generated to provide flexibility to choose one per clock domain, by using the bits [31:24] of MAC_FSM_Control Register. Supported values for Timeout duration are 1us, 1.024ms, 16.384ms, 65.5536ms, 262.144ms, 1048.576ms (~1sec), 4194.304ms (~4s), 8.388s, 16.777s, and 33.554s which is based on the programmable bits NTMRMD, LTMRMD fields of MAC_FSM_ACT_Timer register. Interface timeouts are based on the normal mode tic generation; only the FSM timeouts depend on either normal or large tick selection.

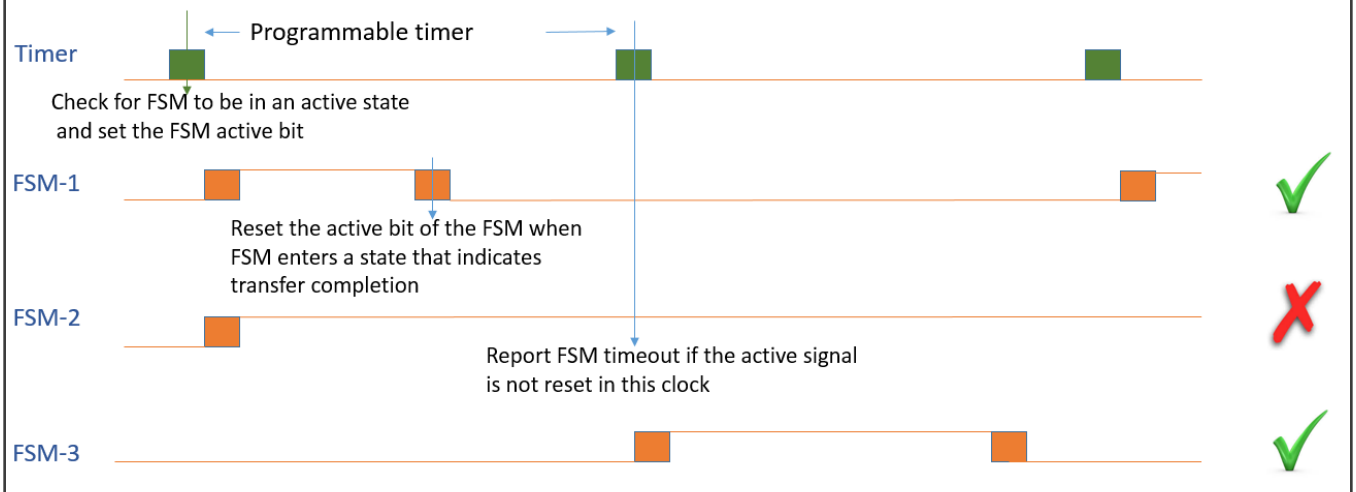



**Figure 410. FSM Timeout Timing Diagram**

In the above figure, the timer ticks are generated based on the programmable timeout value. The timer ticks are synchronized and made available in all the clock domains that have the FSMs.

At every timer tick, all the FSMs are monitored for being in active state (non-IDLE state, which indicates the FSM is actively processing transactions or hand-shakes) and an FSM active flag bit is set. The active flag bit is implemented for every FSM that is monitored for timeout. When the FSM reaches an IDLE or transaction completion state, the active flag bit of that FSM is reset. At the subsequent timer tick, all the FSMs' active flag bits are monitored and any flag that is set indicates a timeout for that FSM. This process of setting the flag and checking at subsequent timer tick is repeated at every tick.

Timeout Error Injection per clock domain is enabled by programming [15:8] fields of MAC_FSM_Control register. When Timeout error injection enable is set for a clock domain, the FSMs in that clock domain automatically timeout and generate an interrupt even without traffic. As error injection is a debug mode, TMR value need not indicate 1us, but can be programmed to a smaller value at which debug mode ticks are needed. FSM Timeout and Parity Error Injection modes can be used for testing at key-on/key-off.

The FSM timeout status is specified in the [15:8] field of MAC_DPP_FSM_Interrupt_Status register as per the respective clock domains. One FSM timeout status bit is made available per clock domain. The safety interrupt (sbd_sfty_ue_intr_o) is asserted when the timeout error is set for any of the clock domains. IP does not attempt to recover from a FSM timeout condition and relies on the application to take the corrective action (resetting IP).

#### 72.14.2.3 Enabling FSM Parity and Timeout Protection

FSM timeout feature is enabled by setting the TMOUTEN field of MAC_FSM_Control register.

**NOTE**

See Programming guidelines for FSM parity and timeout

### 72.14.3 Application/CSR Interface Timeout Protection

This feature provides timeout protection to the application/CSR Interface. All the interfaces which has the handshake mechanism monitored for potential hangs due to the external agent (Master/Slave/Interconnect/Application client) not responding to the requests/transfers initiated by the QoS. After the request is initiated the response arrival interval is monitored. If the response does not arrive within a programmed time (TMR field of MAC_FSM_ACT_Timer) the timeout is triggered, using similar trigger generation as explained in the Section FSM protection.

The timeout is triggered when IP initiates SEQ, NON-SEQ and BUSY transfers on HTRANS[1:0]; and HREADY is not asserted by Slave within the programmed time.

The Timeout status for the AHB master interface is set in the MSTTES field of the MAC_DPP_FSM_Interrupt_Status register. The Safety interrupt (sbd_sfty_ue_intr_o) is asserted when application timeout status is set.

The hardware does not attempt to recover from Application/CSR Interface hangs and relies on software to take appropriate corrective action.

**NOTE**

- Protection is not needed for APB3 Slave interfaces because potential hangs can be only when IP does not respond to the requests. In such cases the IP internal protection logic (FSM timeout) detects such defects.
- Based on the time at which the transfer is initiated relative to the timer ticks, the timeout period could be any value between the programmed timeout and 2x times the programmed timeout.
- When an uncorrectable safety interrupt is issued and the read of the Interrupt status register returns all zeros, it implies that the CSR read is not functional. As soft reset of IP is not possible without the CSR access, and therefore, a hard reset of IP is recommended.
- See Programming guidelines for FSM parity and timeout .

## 72.15 Descriptors

This section and all its sub-sections are Synopsys Proprietary. Used with permission.

### 72.15.1 Overview

The DMA in the Ethernet subsystem transfers data based on a linked list of descriptors. The application creates the descriptors in the system memory. The module supports the following two types of descriptors:

- Normal Descriptor: Normal descriptors are used for packet data and to provide control information applicable to the packets to be transmitted.
- Context Descriptor: Context descriptors are used to provide control information applicable to the packet to be transmitted.

Each normal descriptor contains two buffers and two address pointers. These buffers enable the adapter port to be compatible with various types of memory management schemes.

**NOTE**

There is no limit for the number of descriptors that can be used for a single packet.

### 72.15.2 Descriptor Structure

The module supports the ring structure for DMA descriptor as shown in the following figure.

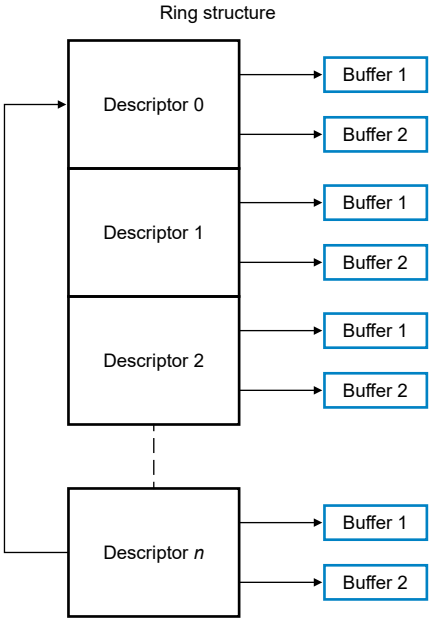



**Figure 411. Descriptor Ring Structure**

In Ring structure, descriptors are separated by the Word, DWord, or LWord number programmed in the DSL field of the DMA_CH#_Control register. The application needs to program the total ring length, that is, the total number of descriptors in ring span in the following registers of a DMA channel:

- Transmit Descriptor Ring Length Register (DMA_CH#_TxDesc_Ring_Length)
- Receive Descriptor Ring Length Register (DMA_CH#_RxDesc_Ring_Length)

The Descriptor Tail Pointer Register contains the pointer to the descriptor address (N). The base address and the current descriptor pointer decide the address of the current descriptor that the DMA can process. The descriptors up to one location less than the one indicated by the descriptor tail pointer (N – 1) are owned by the DMA. The DMA continues to process the descriptors until the following condition occurs:

Current Descriptor Pointer == Descriptor Tail Pointer;

The DMA goes into the Suspend mode when this condition occurs. The application must perform a write to the Descriptor Tail pointer register and update the tail pointer so that the following condition is true:

Current Descriptor Pointer < Descriptor Tail Pointer;

The DMA automatically wraps around the base address when the end of ring is reached, as shown in the following figure.

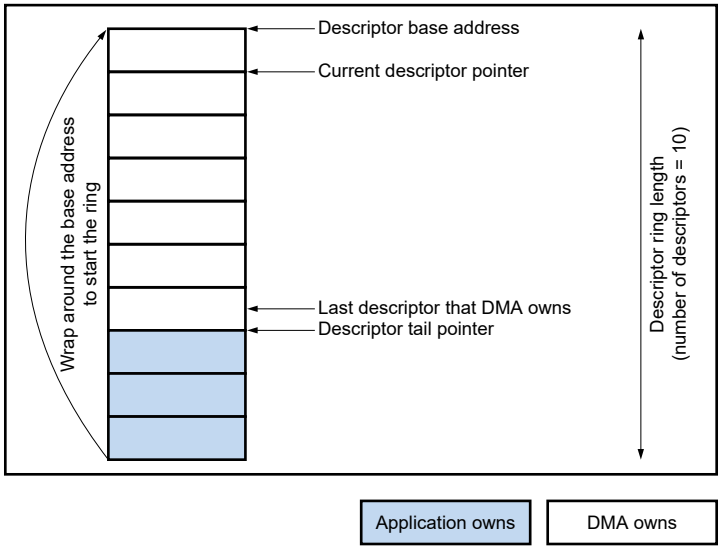



Figure 412. DMA Descriptor Ring

For descriptors owned by the application, the OWN bit of DES3 is reset to 0. For descriptors owned by the DMA, the OWN bit is set to 1. If the application has only one descriptor in the beginning, the application sets the last descriptor address (tail pointer) to Descriptor Base Address + 1. The DMA processes the first descriptor and then waits for the application to advance the tail pointer.

## 72.15.3 Transmit Descriptor

The DMA in the module requires at least one descriptor for a transmit packet. In addition to two buffers, two byte-count buffers, and two address pointers, the transmit descriptor has control fields which can be used to control the MAC operation on per-transmit packet basis. The Transmit Normal descriptor has two formats: Read format and Write-Back format

**NOTE**

TSO Split header is not supported.

Software must not write same tail pointer addresses when DMA is in Suspended state before setting new descriptor(s) for further processing.

### 72.15.3.1 Transmit Normal Descriptor (Read Format)

Following figure shows the Read Format for a Transmit normal descriptor. Table: 'TDES0 Normal Descriptor (Read Format)' through Table: 'TDES3 Normal Descriptor (Read Format)' describe the read format for the Transmit Normal Descriptors: TDES0, TDES1, TDES2, and TDES3.

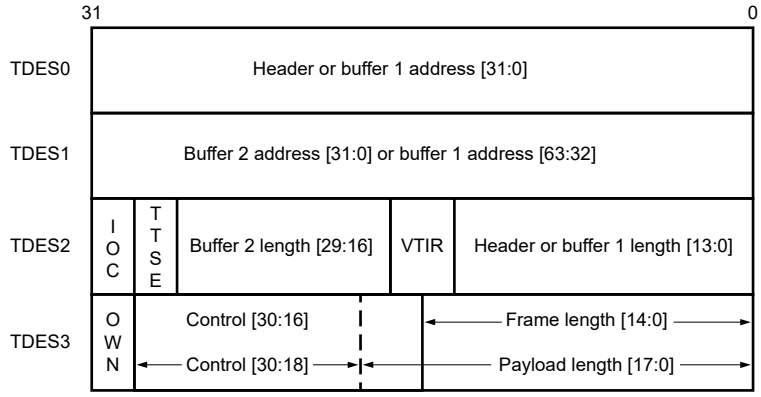



**Figure 413. Transmit Descriptor Read Format**

### 72.15.3.1.1 TDES0 Normal Descriptor (Read Format)

**Table 513. TDES0 Normal Descriptor (Read Format)**

| Bit | Name | Description |
|---|---|---|
| 31:0 | BUF1AP | Buffer 1 Address Pointer or TSO Header Address Pointer<br>These bits indicate the physical address of Buffer 1. These bits indicate the TSO Header Address pointer when the following bits are set:<br>• TSE bit of TDES3<br>• FD bit of TDES3 |

### 72.15.3.1.2 TDES1 Normal Descriptor (Read Format)

**Table 514. TDES1 Normal Descriptor (Read Format)**

| Bit | Name | Description |
|---|---|---|
| 31:0 | BUF2AP | Buffer 2 or Buffer 1 Address Pointer<br>This bit indicates the physical address of Buffer 2 when a descriptor ring structure is used. There is no limitation for the buffer address alignment.<br>In 40- or 48-bit addressing mode, these bits indicate the most-significant 8- or 16- bits of the Buffer 1 Address Pointer. |

### 72.15.3.1.3 TDES2 Normal Descriptor (Read Format)

**Table 515. TDES2 Normal Descriptor (Read Format)**

| 31 | 30 | 29-16 | 15:14 | 13:0 |
|---|---|---|---|---|
| IOC | TTSE/ TMWD | B2L | VTIR | HL or B1L |

Table 516. TDES2 Normal Descriptor (Read Format)

| Bits | Name | Description |
|---|---|---|
| 31 | IOC | Interrupt on Completion<br>This bit controls the setting of TI and ETI status bits in the DMA_CH#_Status register. When ETIC = 1 and TDES2[LD] = 0, this bit sets the ETI bit. When TDES3[LD] = 1, this bit sets the TI status bit. |
| 30 | TTSE/ TMWD | Transmit Timestamp Enable or External TSO Memory Write Enable<br>This bit enables the IEEE1588 time stamping for Transmit packet referenced by the descriptor, if TSE bit is not set.<br>If TSE bit is set and external TSO memory is enabled, setting this bit disables external TSO memory writing for this packet. |
| 29:16 | B2L | Buffer 2 Length<br>The driver sets this field. When set, this field indicates Buffer 2 length. |
| 15:14 | VTIR | VLAN Tag Insertion or Replacement<br>These bits request the MAC to perform VLAN tagging or untagging before transmitting the packets. The application must set the CRC Pad Control bits appropriately when VLAN Tag Insertion, Replacement, or Deletion is enabled for the packet. The following list describes the values of these bits:<br>• 2'b00: Do not add a VLAN tag.<br>• 2'b01: Remove the VLAN tag from the packets before transmission. This option should be used only with the VLAN packets.<br>• 2'b10: Insert a VLAN tag with the tag value programmed in the MAC_VLAN_Incl register or context descriptor.<br>• 2'b11: Replace the VLAN tag in packets with the tag value programmed in the MAC_VLAN_Incl register or context descriptor. This option should be used only with the VLAN packets.<br>These bits are valid when the Enable SA and VLAN Insertion on Tx option is selected while configuring the core. |
| 13:0 | HL or B1L | Header Length or Buffer 1 Length<br>For Header length only bits [9:0] are taken. The size 13:0 is applicable only when interpreting buffer 1 length.<br>If the TCP Segmentation Offload feature is enabled through the TSE bit of TDES3, this field is equal to the header length. When the TSE bit is set in TDES3, the header length includes the length in bytes from Ethernet Source address till the end of the TCP header. The maximum header length supported for TSO feature is 1023 bytes. The maximum header length supported for TSO feature is 1023 bytes.<br>If the TCP Segmentation Offload feature is not enabled, this field is equal to Buffer 1 length. |

#### 72.15.3.1.4 TDES3 Normal Descriptor (Read Format)

Table 517. TDES2 Normal Descriptor (Read Format)

| 31 | 30 | 29 | 28 | 27:26 | 25:23 | 22:19 | 18 | 17:16 | 15 | 14:0 |
|---|---|---|---|---|---|---|---|---|---|---|
| OWN | CTXT | FD | LD | CPC | SAIC | SLOTNUM or THL | TSE | CIC/TPL | TPL | FL/TPL |

Table 518. TDES3 Normal Descriptor (Read Format)

| Bits | Name | Description |
|---|---|---|
| 31 | OWN | Own Bit<br>When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit after it completes the transfer of data given in the associated buffer(s). |
| 30 | CTXT | Context Type<br>This bit should be set to 1'b0 for normal descriptor. |
| 29 | FD | First Descriptor<br>When this bit is set, it indicates that the buffer contains the first segment of a packet. |
| 28 | LD | Last Descriptor<br>When this bit is set, it indicates that the buffer contains the last segment of the packet. When this bit is set, the B1L or B2L field should have a non-zero value. |
| 27:26 | CPC | CRC Pad Control<br>This field controls the CRC and Pad Insertion for Tx packet. This field is valid only when the first descriptor bit (TDES3[29]) is set. The following list describes the values of Bits[27:26]:<br>• 2'b00: CRC and Pad Insertion<br>The MAC appends the cyclic redundancy check (CRC) at the end of the transmitted packet of length greater than or equal to 60 bytes. The MAC automatically appends padding and CRC to a packet with length less than 60 bytes.<br>• 2'b01: CRC Insertion (Disable Pad Insertion)<br>The MAC appends the CRC at the end of the transmitted packet but it does not append padding. The application should ensure that the padding bytes are present in the packet being transferred from the Transmit Buffer, that is, the packet being transferred from the Transmit Buffer is of length greater than or equal to 60 bytes.<br>• 2'b10: Disable CRC Insertion<br>The MAC does not append the CRC at the end of the transmitted packet. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer.<br>• 2'b11: CRC Replacement<br>The MAC replaces the last four bytes of the transmitted packet with recalculated CRC bytes. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer.<br>This field is valid only for the first descriptor.<br>Note: When the TSE bit is set, the MAC ignores this field because the CRC and pad insertion is always done for segmentation. |
| 25:23 | SAIC | SA Insertion Control<br>These bits request the MAC to add or replace the Source Address field in the Ethernet packet with the value given in the MAC Address 0 register. The application must set the CRC Pad Control bits appropriately when SA Insertion Control is enabled for the packet. |

*Table continues on the next page...*

**Table 518. TDES3 Normal Descriptor (Read Format) (continued)**

| Bits | Name | Description |
|---|---|---|
| | | Bit 25 specifies the MAC Address Register (1 or 0) value that is used for Source Address insertion or replacement.<br>The following list describes the values of Bits[24:23]:<br>• 2'b00: Do not include the source address<br>• 2'b01: Include or insert the source address. For reliable transmission, the application must provide frames without source addresses.<br>• 2'b10: Replace the source address. For reliable transmission, the application must provide frames with source addresses.<br>• 2'b11: Reserved<br>These bits are valid in the EQOS-DMA, EQOS-AXI, and EQOS-AHB configurations when the Enable SA and VLAN Insertion on Tx option is selected while configuring the core and when the First Segment control bit (TDES3 [29]) is set.<br>This field is valid only for the first descriptor. |
| 22:19 | SLOTNUM or THL | SLOTNUM: Slot Number Control Bits in AV Mode<br>These bits indicate the slot interval in which the data should be fetched from the corresponding buffers addressed by TDES0 or TDES1.<br>When the Transmit descriptor is fetched, the DMA compares the slot number value in this field with the slot interval maintained in the RSN field DMA_CH#_Slot_Function_Control_Status. It fetches the data from the buffers only if a value matches. These bits are valid only for the AV channels.<br>THL: TCP/UDP Header Length<br>If the TSE bit is set, this field contains the length of the TCP/UDP header. The minimum value of this field must be 5 for TCP header. The value must be equal to 2 for UDP header.<br>This field is valid only for the first descriptor. |
| 18 | TSE | TSO Split header is not supported. The value of this bit is always zero. |
| 17:16 | CIC/TPL | Checksum Insertion Control or TCP Payload Length<br>These bits control the checksum calculation and insertion. The following list describes the bit encoding:<br>• 2'b00: Checksum Insertion Disabled.<br>• 2'b01: Only IP header checksum calculation and insertion are enabled.<br>• 2'b10: IP header checksum and payload checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated in hardware.<br>• 2'b11: IP Header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware.<br>This field is valid when the Enable Transmit TCP/IP Checksum Offload option is selected and the TSE bit is reset.<br>When the TSE bit is set, this field contains the upper bits [17:16] of the TCP Payload (or IP Payload for UDP fragmentation). This allows the TCP/UDP packet length field to be spanned across TDES3[17:0] to provide 256 KB packet length support. |

*Table continues on the next page...*

Table 518. TDES3 Normal Descriptor (Read Format) (continued)

| Bits | Name | Description |
|---|---|---|
| | | This field is valid only for the first descriptor. |
| 15 | TPL | Reserved or TCP Payload Length<br>When the TSE bit is reset, this bit is reserved. When the TSE bit is set, this is Bit 15 of the TCP payload length [17:0].<br>This field is valid only when the Enable TCP Segmentation Offloading for TCP/IP Packets option is selected while configuring the core. |
| 14:0 | FL/TPL | Frame Length or TCP Payload Length<br>This field is equal to the length of the packet to be transmitted in bytes. When the TSE bit is not set, this field is equal to the total length of the packet to be transmitted:<br>Ethernet Header Length + TCP /IP Header Length – Preamble Length – SFD Length + Ethernet Payload Length<br>When the TSE bit is set, this field is equal to the lower 15 bits of the TCP payload length in case of segmentation and IP payload in case of UDP fragmentation.<br>In case of segmentation, this length does not include Ethernet header or TCP/UDP/IP header length. In case of fragmentation, this length does not include Ethernet header and IP header.<br>When DWRR/WFQ algorithm is NOT enabled, value written into this field is not used when TSE = 0. |

#### 72.15.3.1.5 Transmit normal descriptor (write-back format)

The transmit descriptor write-back format includes timestamp low, timestamp high, OWN, and Status bits.

The write-back format is applicable only for the last descriptor of the corresponding packet. Write 1 to the LD bit (TDES3[28]) in the descriptor where DMA writes back the status and timestamp information for the corresponding transmit packet.

Figure 414 shows the transmit descriptor write-back format.

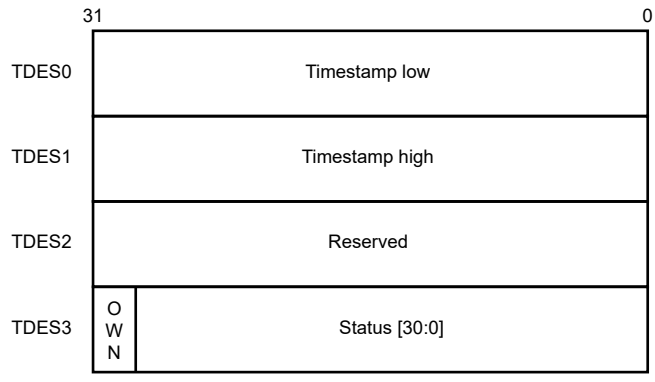



Figure 414. Transmit descriptor write-back format

TDES0 normal descriptor (write-back format)

Table 519 shows that this format is only applicable to the last descriptor of a packet.

**Table 519. TDES0 normal descriptor (write-back format)TDES0 normal descriptor (write-back format)**

| Bit | Name | Description |
|---|---|---|
| 31:0 | TTSL | Transmit Packet Timestamp Low<br>DMA updates this field with least significant 32 bits of the timestamp captured for the corresponding transmit packet. DMA writes the timestamp only if TTSE bit of TDES2 is 1 in the first descriptor of the packet. This field has the timestamp only if you write 1 to the Last Segment bit (LS) in the descriptor and the Timestamp status (TTSS) bit. |

TDES1 normal descriptor (write-back format)

Table 520 shows that this format is only applicable to the last descriptor of a packet.

**Table 520. TDES1 normal descriptor (write-back format)**

| Bit | Name | Description |
|---|---|---|
| 31:0 | TTSH | Transmit Packet Timestamp High<br>DMA updates this field with the most significant 32 bits of the timestamp captured for a corresponding transmit packet. DMA writes the timestamp only if the TTSE bit of TDES2 is 1 in the first descriptor of the packet. This field has the timestamp only if you write 1 to the Last Segment bit (LS) in the descriptor and Timestamp status (TTSS) bit. |

TDES2 Normal Descriptor (Write-Back Format)

This format is applicable only to the last descriptor of a packet.

**Table 521. TDES2 normal descriptor (write-back format)**

| Bit | Description |
|---|---|
| 31:0 | Reserved |

TDES3 Normal Descriptor (Write-Back Format)

This format is applicable only to the last descriptor of a packet.

**Table 522. TDES3 normal descriptor (write-back format)**

| 31 | 30 | 29 | 28 | 27:18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7:4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OWN | CTXT | FD | LD | Rsvd | TTSS | EUE | ES | JT | FF | PCE | LoC | NC | LC | EC | CC | ED | UF | DB | IHE |

**Table 523. TDES3 normal descriptor (write-back format)**

| Bit | Name | Description |
|---|---|---|
| 31 | OWN | Own Bit<br>When this field is 1, it indicates that the module DMA owns the descriptor. DMA write 0 to this field when it completes the packet transmission. After the write-back is complete, this field is set to 'b0. |
| 30 | CTXT | Context Type |

*Table continues on the next page...*

Table 523. TDES3 normal descriptor (write-back format) (continued)

| Bit | Name | Description |
|---|---|---|
| | | This field must be set to 'b0 for normal descriptor. |
| 29 | FD | First Descriptor<br>This field indicates that the buffer contains the first segment of a packet. |
| 28 | LD | Last Descriptor<br>This field is set to 'b1 for last descriptor of a packet. DMA writes the status fields only in the last descriptor of the packet. |
| 27:24 | Rsvd | Reserved |
| 23 | DE | Descriptor Error<br>When this field is 1, it indicates that the descriptor content is incorrect. DMA writes 1 to this field during write-back when the descriptor is closed.<br>Descriptor errors can be:<br>• Incorrect sequence from the context descriptor. For example, a location after the first descriptor for a packet.<br>• All 1s<br>• CTXT is 1 along with LD or FD bits as 1.<br>**NOTE**<br>When a descriptor error occurs due to all ones or CTXT, LD, and FD bits are 1, the transmit DMA closes the transmit descriptor with DE and LD bits set to 1. When you write 1 to IOC bit in TDES2 of corresponding first descriptor, the transmit DMA write 1 to TI bit in the DMA_CH#_Status register<br>**NOTE**<br>Based on CTXT, LD, and FD bits of the transmit descriptor, the subsequent descriptor might consider as the first descriptor (even if FD bit is not 1) and partial packet is sent. |
| 22:18 | Rsvd | Reserved |
| 17 | TTSS | Tx Timestamp Status<br>This field indicates that a timestamp has been captured for the corresponding transmit packet. When this field is 1, it indicates that TDES0 and TDES1 have timestamp values that were captured for the transmit packet. This field is valid only when the Last Segment control bit (TDES3 [28]) in a descriptor is 1. This field is valid only when IEEE1588 timestamping feature is enabled; otherwise, it is reserved. |
| 16 | EUE | ECC Uncorrectable Error Status<br>Indicates the ECC uncorrectable error in the TSO memory.<br>**NOTE**<br>An uncorrectable error in the transmit FIFO memory is reported with (Bit 13) FF = 1. This is because the module flushes all such packets. |

*Table continues on the next page...*

Table 523. TDES3 normal descriptor (write-back format) (continued)

| Bit | Name | Description |
|---|---|---|
| 15 | ES | Error Summary<br>This field indicates the logical OR of the following bits:<br>• TDES3[0]: IP header error<br>• TDES3[14]: Jabber timeout<br>• TDES3[13]: Packet flush<br>• TDES3[12]: Payload checksum error<br>• TDES3[11]: Loss of carrier<br>• TDES3[10]: No carrier<br>• TDES3[9]: Late collision<br>• TDES3[8]: Excessive collision<br>• TDES3[3]: Excessive deferral<br>• TDES3[2]: Underflow error<br>This field is 1 when EUE (bit 16) = 1. |
| 14 | JT | Jabber Timeout<br>This field indicates that the MAC transmitter experiences a jabber time-out. This field is 1 only when MAC_Configuration[JD] is not 1. |
| 13 | FF | Packet Flushed<br>This field indicates that DMA or MTL flushes the packet because the CPU gives a software flush command. |
| 12 | PCE | Payload Checksum Error<br>This field indicates that the checksum offload engine had a failure and does not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can either because of insufficient bytes, as the payload length field of the IP header indicates or the MTL starting to forward the packet to the MAC transmitter in Store-and-forward mode without calculating the checksum. This second error condition only occurs when the transmit FIFO depth is less than the length of the Ethernet packet that is transmitted to avoid deadlock. The MTL starts forwarding the packet when the FIFO is full, even in the Store-and-forward mode.<br>This error can also occur when you detects a bus error during packet transfer.<br>When the full checksum offload engine is not enabled, this bit is reserved. |
| 11 | LoC | Loss of Carrier<br>This field indicates that a loss of carrier occurred during packet transmission (that is, the gmii_crs_i signal was inactive for one or more transmit clock periods during packet transmission). This field is valid only for the packets transmitted without collision and when MAC operates in the Half-duplex mode. |
| 10 | NC | No Carrier<br>This field indicates that the carrier sense signal from the PHY was not asserted during transmission. |

*Table continues on the next page...*

Table 523. TDES3 normal descriptor (write-back format) (continued)

| Bit | Name | Description |
|---|---|---|
| 9 | LC | Late Collision<br>This bit indicates that packet transmission was aborted because a collision occurred after the collision window (64 byte times including Preamble in MII mode and 512 byte times including Preamble and Carrier Extension in GMII mode). This bit is not valid if Underflow Error is set. |
| 8 | EC | Excessive Collision<br>This field indicates that the transmission was aborted after 16 successive collisions when you transmits the current packet. If MAC_Configuration[DR] = 1, this field is 1 after first collision and abort the packet transmission. |
| 7:4 | CC | Collision Count<br>This 4-bit counter value indicates the number of collisions occurred before transmitting the packet. The count is not valid when the EC bit = 1. |
| 3 | ED | Excessive Deferral<br>This field indicates that the transmission ends because of an excessive deferral of over 24,288 bit times (155,680 bits times in 1000 Mbit/s mode or Jumbo packet enabled mode) if MAC_Configuration[DC] = 1.<br>When TBS is enabled in Full duplex mode and this field is 1, it indicates that the frame drops after the expiry time is reached. |
| 2 | UF | Underflow Error<br>This field indicates that MAC aborts the packet because the data arrived late from the system memory. The underflow error can occur because of either of the following conditions:<br>• DMA encounters an empty transmit buffer when it transmits the packet<br>• The application filled the MTL Tx FIFO slower than the MAC transmit rate<br>The transmission process enters the Suspended state and sets the underflow bit corresponding to a queue in MTL_Interrupt_Status. |
| 1 | DB | Deferred Bit<br>This field indicates that MAC defers before transmitting because of presence of carrier. This field is valid only in the Half-Duplex mode. |
| 0 | IHE | IP Header Error<br>When IP Header Error is 1, this field indicates that the checksum offload engine detects an IP header error. This field is valid only when transit checksum offload is enabled. Otherwise, it is reserved. If COE detects an IP header error, it inserts an IPv4 header checksum if the Ethernet type field indicates an IPv4 payload.<br>In Full-Duplex mode, when EST/Qbv is enabled and this field is 1, it indicates the frame drop status due to the frame size error or schedule error. |

### 72.15.3.2 Transmit context descriptor

The transmit context descriptor can provide any time before a packet descriptor. The context is valid for the current packet and subsequent packets. The context descriptor provide the timestamps for one-step timestamp correction and VLAN tag ID for VLAN insertion feature. Write back is done on a context descriptor only to write 0 to OWN bit.

**NOTE**

DMA internally store the VLAN tag IDs and MSS values which the application provides in a context descriptor with their corresponding valid bits set. DMA always passes the last valid VLAN tag to the MTL, when the outer or inner VLAN tag is provided with the valid bit set. The application cannot invalidate the valid VLAN tag which DMA stores. The VLAN tag is inserted or replaced based on the control inputs provided for the packet.

The inner VLAN tag control input is used only for the next packet that immediately follows the context descriptor. The application must provide a context descriptor before the normal descriptor of each packet for which DMA must use the inner VLAN tag control input.

Figure 415 shows the transmit context descriptor format.

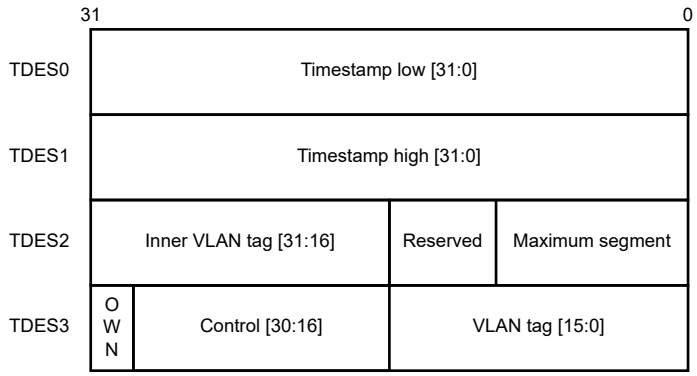



**Figure 415. Transmit context descriptor format**

### 72.15.3.2.1 TDES0 context descriptor

Table 524 describes the TDES0 context descriptor format.

**Table 524. TDES0 context descriptor**

| Bit | Name | Description |
|---|---|---|
| 31:0 | TTSL | Transmit Packet Timestamp Low<br>For one-step correction, the driver can provide the lower 32 bits of timestamp in this descriptor word. DMA uses this value as the low word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are 1. |

### 72.15.3.2.2 TDES1 context descriptor

Table 525 describes the TDES1 context descriptor format.

**Table 525. TDES1 context descriptor**

| Bit | Name | Description |
|---|---|---|
| 31:0 | TTSH | Transmit Packet Timestamp High<br>For one-step correction, the driver can provide the upper 32 bits of timestamp in this descriptor. DMA uses this value as the high word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are 1. |

#### 72.15.3.2.3 TDES2 context descriptor

Table 526 shows the TDES2 context descriptor format.

**Table 526. TDES2 context descriptor**

| Bit | Name | Description |
|---|---|---|
| 31:16 | IVT | Inner VLAN Tag<br>When the IVLTV bit of TDES3 context descriptor is 1 and the TCMSSV and OSTC bits of TDES3 context descriptor becomes 0, it indicates that the TDES2[31:16] contains the inner VLAN tag to insert in the subsequent transmit packets. |
| 15:14 | Rsvd | Reserved |
| 13:0 | MSS | TSO split header is not supported. The value of this bit is always zero. |

#### 72.15.3.2.4 TDES3 context descriptor

**Table 527. TDES3 context descriptor**

| 31 | 30 | 29:28 | 27 | 26 | 25:24 | 23 | 22:18 | 17 | 16 | 15:0 |
|---|---|---|---|---|---|---|---|---|---|---|
| OWN | CTXT | Rsvd | OSTC | TCMSSV | Rsvd | CDE | Rsvd | IVLTV | VLTV | VT |

Table 528 shows the TDES3 context descriptor format.

**Table 528. TDES3 context descriptor**

| Bit | Name | Description |
|---|---|---|
| 31 | OWN | Own Bit<br>When this field is 1, it indicates that the module DMA owns the descriptor. When this field becomes 0, it indicates that the application owns the descriptor. DMA writes 0 to this field immediately after the read. |
| 30 | CTXT | Context Type<br>This field must set to 1'b1 for context descriptor. |
| 29:28 | Rsvd | Reserved |
| 27 | OSTC | One-Step Timestamp Correction Enable<br>When this field is 1, it indicates that DMA performs a one-step timestamp correction with reference to the timestamp values provided in TDES0 and TDES1. |
| 26 | TCMSSV | One-Step Timestamp Correction Input or MSS Valid<br>When this field and the OSTC field are 1, it indicates that the timestamp correction input provided in TDES0 and TDES1 is valid.<br>When the OSTC field becomes 0 and this field and the TSE bit of TDES3 are 1 in subsequent normal descriptor, it indicates that the MSS input in TDES2 is valid. |
| 25:24 | Rsvd | Reserved |

*Table continues on the next page...*

**Table 528. TDES3 context descriptor (continued)**

| Bit | Name | Description |
|---|---|---|
| 23 | DE | Descriptor Error<br>When this field is 1, it indicates that the descriptor content is incorrect. DMA writes 1 to this field during write-back when it closes the context descriptor.<br>Descriptor errors can be:<br>• Incorrect sequence from the context descriptor. For example, a location before the first descriptor for a packet.<br>• All ones.<br>• CD, LD, and FD bits = 1.<br>**NOTE**<br>When a descriptor error occurs due to all ones or CTXT, LD, and FD bits are 1, the transmit DMA closes the transmit descriptor with DE and LD bits are 1. When you write 1 to IOC bit in TDES2 of corresponding first descriptor, the transmit DMA writes 1 to TI bit in the DMA_CH#_Status register<br>**NOTE**<br>On the basis of CTXT, LD, and FD bits of the transmit descriptor, the subsequent descriptor might considered as the first descriptor (even if FD bit is not 1) and partial packet is sent.<br>**NOTE**<br>This error is categorized as an abnormal event. Recovery from this event is possible by issuing a software reset only. DMA stopping, reconfiguring, and restarting recovery mechanism is not supported. |
| 22:20 | Rsvd | Reserved |
| 19:18 | IVTIR | Inner VLAN Tag Insert or Replace<br>When this field is 1, it requests MAC to perform inner VLAN tagging or un-tagging before transmitting the packets. If the packet is modified for VLAN tags, MAC automatically recalculates and replaces the CRC bytes.<br>The following list describes the values of these bits:<br>• 2'b00: Do not add the inner VLAN tag.<br>• 2'b01: Remove the inner VLAN tag from the packets before transmission. You must use this option only with the VLAN frames.<br>• 2'b10: Insert an inner VLAN tag with the tag value programmed in MAC_Inner_VLAN_Incl or context descriptor.<br>• 2'b11: Replace the inner VLAN tag in packets with the tag value programmed in MAC_Inner_VLAN_Incl or context descriptor. You must use this option only with the VLAN frames..<br>These fields are valid when you select the enable SA and VLAN insertion on transit and enable double VLAN processing options. |
| 17 | IVLTV | Inner VLAN Tag Valid<br>When this field is 1, it indicates that IVT field of TDES2 is valid. |

*Table continues on the next page...*

Table 528. TDES3 context descriptor (continued)

| Bit | Name | Description |
|---|---|---|
| 16 | VLTV | VLAN Tag Valid<br>When this field is 1, it indicates that VT field of TDES3 is valid. |
| 15:0 | VT | VLAN Tag<br>Contains the VLAN tag to insert or replace in the packet. This field is used as VLAN tag only when MAC_VLAN_Incl[VLTI] becomes 0. |

### 72.15.4 Receive descriptor

DMA in the module reads a descriptor only if the tail pointer is different from the base pointer or current pointer. It is recommended to have a descriptor ring with a length that accommodates at least two complete packets which MAC receives. Otherwise, the performance of DMA is impacted because of the unavailability of the descriptors. In such situations, the RxFIFO in MTL becomes full and starts dropping packets.

The following receive descriptors are present:

- Normal descriptors
- Context descriptors

You can prepare all the receive descriptors and give to DMA as normal descriptors with the content as shown in receive normal descriptor (read format). DMA reads this descriptor and the receive DMA closes the descriptor with the corresponding packet status after transferring a received packet (or part of) to the buffers indicated by the descriptor. The receive normal descriptor (write-back format) describes the format of this status.

For some packets, only the normal descriptor bits cannot write the complete status. For such packets, the receive DMA writes the extended status to the next descriptor (without processing or using the buffers or pointers embedded in that descriptor). Receive context descriptor describes the descriptor write back format and content.

#### 72.15.4.1 Receive normal descriptor (read format)

The read format for a receive normal descriptor is made up of the following:

- A header or buffer 1 address
- Reserved field
- Payload or buffer 2 or next descriptor address
- A 30-bit reserved field
- OWN bit
- An interrupt field

Figure 416 shows the read format for a receive normal descriptor.

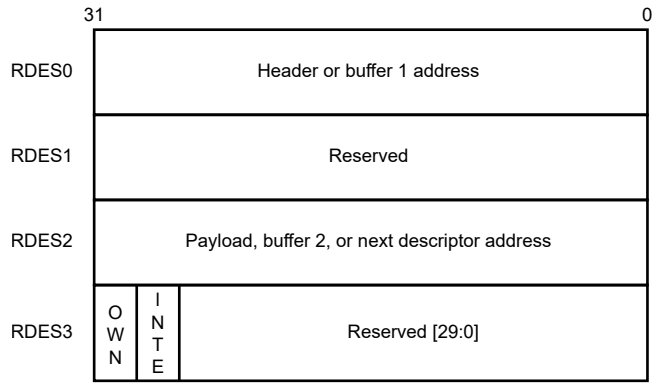



**Figure 416. Receive normal descriptor read format**

**NOTE**

In the receive descriptor (read format), if the buffer address field is all zeros, the module does not transfer data to that buffer and skips to the next buffer or next descriptor.

#### 72.15.4.1.1 RDES0 normal descriptor (read format)

Table 529 shows the RDES0 normal descriptor read format.

**Table 529. RDES0 normal descriptor (read format)**

| Bit | Name | Description |
|---|---|---|
| 31:0 | BUF1AP | Header or Buffer 1 Address Pointer<br>When the SPH field of control register of a channel becomes 0, these field indicates the physical address of buffer 1. When the SPH field is 1, these field indicates the physical address of header buffer where the receive DMA writes the L2/L3/L4 header bytes of the received packet.<br>The application can program a byte-aligned address for this buffer which means that the LS bits of this field can be non-zero. However, DMA performs a write operation with RDES0[1:0] (or RDES0[2:0]/[3:0] in case of 64-/128-bit configuration) as zero, when the start of packet is transferred. However, the packet data shifts per the actual offset which the buffer address pointer provides.<br>If the address pointer points to a buffer where the middle or last part of the packet is stored, DMA ignores the offset address and writes to the full location which the data-width indicates . |

#### 72.15.4.1.2 RDES1 normal descriptor (read format)

Table 530 shows the RDES1 normal descriptor read format.

**Table 530. RDES1 normal descriptor (read format)**

| Bit | Name | Description |
|---|---|---|
| 31:0 | Reserved or BUF1AP | Contains the most-significant 32 bits of the buffer 1 address pointer in 64-bit Addressing mode. Otherwise, this field is reserved. |

#### 72.15.4.1.3 RDES2 normal descriptor (read format)

Table 531 shows the RDES2 normal descriptor read format.

Table 531. RDES2 normal descriptor (read format)

| Bit | Name | Description |
|---|---|---|
| 31:0 | BUF2AP | Buffer 2 Address Pointer<br>Indicates the physical address of buffer 2.<br>When the SPH bit of the DMA_CH#_Control register = 1, it indicates that the buffer address pointer must be bus width-aligned, that is, RDES2[3:0, 2:0, or 1:0] = 0 corresponding to 128, 64, or 32 bus width. LSBs are ignored internally.<br>When the SPH bit of the DMA_CH#_Control register becomes 0, it indicates that there is no limitations on the RDES2 value. However, the RxDMA uses the LS fields of the pointer address only when it transfers the start bytes of a packet. If the BUF2AP provides the address of a buffer in which the middle or last part of a packet is stored, DMA ignores BUF2AP[3:0 or 2:0 or 1:0] (corresponding to 128- or 64- or 32-bit data-bus) and writes to the complete location. |

#### 72.15.4.1.4 RDES3 normal descriptor (read format)

Following table describes the RDES3 normal descriptor read format.

| 31 | 30 | 19-26 | 25 | 24 | 23:0 |
|---|---|---|---|---|---|
| OWN | IOC | Rsvd | BUF2V | BUF1V | Rsvd |

Table 532. RDES3 normal descriptor (read format)

| Bit | Name | Description |
|---|---|---|
| 31 | OWN | Own Bit<br>When this field is 1, it indicates that the module DMA owns the descriptor. When this field becomes 0, it indicates that the application owns the descriptor. DMA write 0 this field when either of the following conditions is true:<br>• DMA completes the packet reception.<br>• The buffers associated with the descriptor are full. |
| 30 | IOC | Interrupt Enabled on Completion<br>When this field is 1, it indicates that an interrupt is issued to the application when DMA closes this descriptor. |
| 29:26 | Rsvd | Reserved |
| 25 | BUF2V | Buffer 2 Address Valid<br>When this field is 1, it indicates to the DMA that the buffer 2 address specified in RDES2 is valid.<br>The application must write 1 to this field so that DMA can use the address, to which the buffer 2 address in RDES2 points to write received packet data. |
| 24 | BUF1V | Buffer 1 Address Valid<br>When this field is 1, it indicates to the DMA that the buffer 1 address specified in RDES1 is valid. |

*Table continues on the next page...*

Table 532. RDES3 normal descriptor (read format) (continued)

| Bit | Name | Description |
|---|---|---|
| | | The application must write 1 to this field, if DMA uses the address to which the buffer 1 address in RDES1 points to<br>write the received packet data. |
| 23:0 | Rsvd | Reserved |

### 72.15.4.1.5 Receive normal descriptor (write-back format)

Following figure shows the write-back format for a receive normal descriptor.

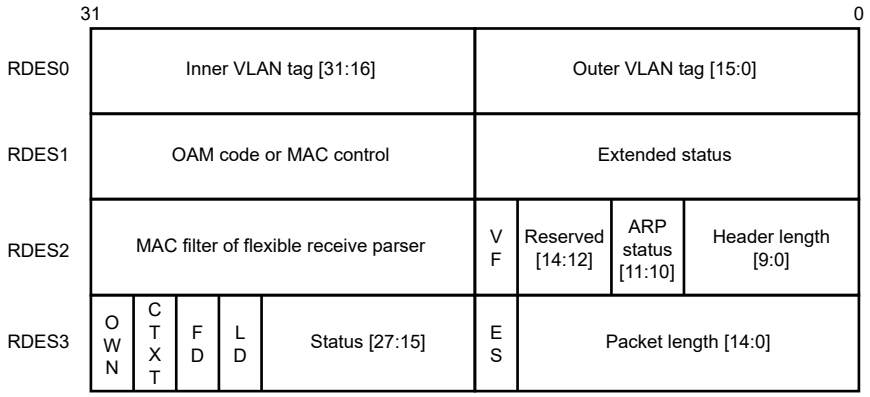



Figure 417. Receive normal descriptor (write-back format)

**NOTE**

When you enables the flexible receive parser, RDES2[31:16] indicates the parser status, and not the MAC filter status. The MAC filter status is not available when flexible receive parser is enabled.

#### 72.15.4.1.5.1 RDES0 normal descriptor (write-back format)

Table 533 shows the write-back format for the RDES0 normal descriptor.

Table 533. RDES0 normal descriptor (write-back format)

| Bit | Name | Description |
|---|---|---|
| 31:16 | IVT | Inner VLAN Tag<br>Contains the inner VLAN tag of the received packet if the RS0V bit of RDES3 = 1.<br>This field is valid only when you enables the double VLAN tag processing and VLAN tag stripping. |
| 15:0 | OVT | Outer VLAN Tag<br>Contains the outer VLAN tag of the received packet if the RS0V bit of RDES3 = 1. |

#### 72.15.4.1.5.2 RDES1 normal descriptor (write-back format)

The status fields in the write-back format are valid only for the last descriptor (RDES3[28] = 1). Following table provide the details of the write-back format for RDES1 normal descriptor.

| 31:16 | 15 | 14 | 13 | 12 | 11:8 | 7 | 6 | 5 | 4 | 3 | 2:0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| OPC | TD | TSA | PV | PFT | PMT | IPCE | IPCB | IPV6 | IPV4 | IPHE | PT |

**Table 534. RDES1 normal descriptor (write-back format)**

| Bit | Name | Description |
|---|---|---|
| 31:16 | OPC | Indicates any one of the following:<br>• OAM sub-type code: If bits [18:16] of RDES3 as described in RDES3 normal descriptor (read format) are 111b, this field contains the OAM sub-type and code fields.<br>• MAC control packet opcode: Bits 15:8 of RDES3 as described in RDES3 normal descriptor (read format) contain the subtype and bits 7:0 contain the code. |
| 15 | TD | Timestamp Dropped<br>Indicates that the timestamp was captured for this packet but it dropped in the MTL Rx FIFO because of overflow.<br>This field is available only when you select the timestamp feature. Otherwise, this field is reserved. |
| 14 | TSA | Timestamp Available<br>Indicates that the timestamp value is available in a context descriptor word 2 (RDES2) and word 1(RDES1) when timestamp is present. This is valid only when the last descriptor field (RDES3 [28]) = 1<br>You can write the context descriptor in the next descriptor just after the last normal descriptor for a packet. |
| 13 | PV | PTP Version<br>Indicates that the received PTP message has the IEEE 1588 version 2 format. When this field becomes 0, it indicates the IEEE 1588 version 1 format.<br>This field is available only when you select the timestamp feature. Otherwise, this field is reserved. |
| 12 | PFT | PTP Packet Type<br>Indicates that the PTP message is sent directly over Ethernet. This field is available only when you select the timestamp feature. Otherwise, this field is reserved. |
| 11:8 | PMT | PTP Message Type<br>Encodes to give the type of the message received:<br>• 0000: No PTP message received<br>• 0001: SYNC (all clock types)<br>• 0010: Follow_Up (all clock types)<br>• 0011: Delay_Req (all clock types)<br>• 0100: Delay_Resp (all clock types)<br>• 0101: Pdelay_Req (in peer-to-peer transparent clock)<br>• 0110: Pdelay_Resp (in peer-to-peer transparent clock)<br>• 0111: Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) |

*Table continues on the next page...*

Table 534. RDES1 normal descriptor (write-back format) (continued)

| Bit | Name | Description |
|---|---|---|
| | | • 1000: Announce<br>• 1001: Management<br>• 1010: Signaling<br>• 1011–1110: Reserved<br>• 1111: PTP packet with reserved message type<br>These fields are available only when you select the timestamp feature. |
| 7 | IPCE | IP Payload Error<br>When this field is 1, it indicates either of the following:<br>• The 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) which MAC calculates does not match the corresponding checksum field in the received segment.<br>• The TCP, UDP, or ICMP segment length does not match the payload length value in the IP header field.<br>• The TCP, UDP, or ICMP segment length is less than minimum allowed segment length for TCP, UDP, or ICMP.<br>Bit 15 (ES) of RDES3 is not set when this field is 1. |
| 6 | IPCB | IP Checksum Bypassed<br>Indicates that the checksum offload engine is bypassed. This field is available when you select the enable receive TCP/IP checksum check feature. |
| 5 | IPV6 | IPv6 header Present<br>Indicates that an IPV6 header is detected. When you select the enable split header feature option and the SPH bit of control register of a channel is 1, the IPV6 header is available in the header buffer area to which RDES0 points. |
| 4 | IPV4 | IPV4 Header Present<br>Indicates that an IPV4 header is detected. When the SPH bit of RDES3 is 1, the IPV4 header is available in the header buffer area to which RDES0 points. |
| 3 | IPHE | IP Header Error<br>When this field is 1, it indicates either of the following:<br>• The 16-bit IPv4 header checksum which MAC calculates does not match the received checksum bytes.<br>• The IP datagram version is not consistent with the Ethernet type value.<br>• Ethernet packet does not have the expected number of IP header bytes.<br>This field is valid when either bit 5 or bit 4 is 1. This field is available when you select the enable receive TCP/IP checksum check feature. |
| 2:0 | PT | Payload Type<br>Indicates the type of payload encapsulated in the IP datagram that the receive checksum offload engine (COE) process. |

*Table continues on the next page...*

Table 534. RDES1 normal descriptor (write-back format) (continued)

| Bit | Name | Description |
|---|---|---|
| | | • 3'b000: Unknown type or IP/AV payload not processed<br>• 3'b001: UDP<br>• 3'b010: TCP<br>• 3'b011: ICMP<br>• 3'b110: AV tagged data packet<br>• 3'b111: AV tagged control packet<br>• 3'b101: AV untagged control packet<br>• 3'b100: IGMP, if IPV4 header present field = 1, else DCB (LLDP) control packet<br>If the COE does not process the payload of an IP datagram because there is an IP header error or fragmented IP, it sets these fields to 3'b000. |

#### 72.15.4.1.5.3 RDES2 normal descriptor (Write-back format)

| 31:29 | 28 | 27 | 26:19 | 18 | 17 | 16 | 15 | 14 | 13:11 | 10 | 9:0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| L3L4FM | L4FM | L3FM | MADRM | HF | DAF | SAF | OTS | ITS | Rsvd | ARPNR | HL |

Table 535. RDES2 normal descriptor (Write-back format)

| Bit | Name | Description |
|---|---|---|
| 31:29 | L3L4FM | Layer 3 and Layer 4 Filter Number Matched<br>Indicates the number of the layer 3 and layer 4 filter that matches the received packet:<br>• 000: Filter 0<br>• 001: Filter 1<br>• 010: Filter 2<br>• 011: Filter 3<br>• 100: Filter 4<br>• 101: Filter 5<br>• 110: Filter 6<br>• 111: Filter 7<br>This field is valid only when bit 28 or bit 27 = 1. When more than one filter matches, these fields provide the number of lowest filter.<br>**NOTE**<br>This status is not available when Flexible RX Parser is enabled. |
| 28 | L4FM | Layer 4 Filter Match |

*Table continues on the next page...*

Table 535. RDES2 normal descriptor (Write-back format) (continued)

| Bit | Name | Description |
|---|---|---|
| | | When this field is 1, it indicates that the received packet matches one of the enabled layer 4 port number fields. This status is given only when one of these conditions is true:<br>• Layer 3 fields are not enabled and all enabled layer 4 fields match<br>• All enabled layer 3 and layer 4 filter fields match<br>When more than one filter matches, this bit gives the layer 4 filter status of filter indicated by Bits[31:29].<br>**NOTE**<br>This status is not available when the flexible receive parser is enabled. |
| 27 | L3FM | Layer 3 Filter Match<br>When this field is 1, it indicates that the received packet matches one of the enabled layer 3 IP address fields. This status is given only when one of these conditions is true:<br>• All enabled layer 3 fields match and bypasses all enabled layer 4 fields.<br>• All enabled filter fields match.<br>When more than one filter matches, this field gives the layer 3 filter status of filter which bits[31:29] indicate.<br>**NOTE**<br>This status is not available when the flexible receive parser is enabled. |
| 26:19 | MADRM | MAC Address Match or Hash Value<br>When HF becomes 0, it indicates that this field contains the MAC address register number that matches the received packet destination address. This field is valid only if the DAF field becomes 0.<br>When HF = 1, it indicates that this field contains the hash value that MAC computes. A packet passes the hash filter when the field corresponding to the hash value is 1 in the hash filter register.<br>**NOTE**<br>This status is not available when the flexible receive parser is enabled. |
| 18 | HF | Hash Filter Status<br>When this field is 1, it indicates that the packet passes the MAC address hash filter. Bits[26:19] indicate the hash value.<br>**NOTE**<br>This status is not available when the flexible receive parser is enabled. |
| 17 | DAF/RXPI | Destination Address Filter Fail<br>When flexible receive parser is disabled, and this field is 1, it indicates that the packet fails the DA filter in MAC.<br>When flexible receive parser is enabled, and this field is 1, it indicates that the packet parsing is incomplete (RXPI) due to ECC error. |

*Table continues on the next page...*

Table 535. RDES2 normal descriptor (Write-back format) (continued)

| Bit | Name | Description |
|---|---|---|
| | | **NOTE**<br>When this field is 1, RDES3 ES field is also 1. |
| 16 | SAF/RXPD | SA Address Filter Fail<br>When flexible receive parser is disabled, and this field is 1, it indicates that the packet fails the SA filter in MAC.<br>When flexible receive parser is enabled and this field is 1, it indicates that the parser drops the RXPD packet.<br>**NOTE**<br>When this field is 1, RDES3 ES field is also 1. |
| 15 | OTS | VLAN Filter Status<br>When this field is 1, it indicates that the received packet VLAN tag passes the VLAN filter.<br>This field redefines as an outer VLAN tag filter status (OTS). This field is valid for both single and double VLAN tagged frames. |
| 14 | ITS | Inner VLAN Tag Filter Status (ITS)<br>This field is valid only for double VLAN tagged frames, when double VLAN processing is enabled.<br>See Filter status for more information. |
| 13:11 | Rsvd | Reserved |
| 10 | ARPNR | ARP Reply Not Generated<br>When this field is 1, it indicates that MAC did not generate the ARP reply for the ARP request packet it receives.. This field is 1 when MAC is busy transmitting ARP reply to earlier ARP request (only one ARP request is processed at a time).<br>This field is reserved when you do not select the enable IPv4 ARP offload option. |
| 9:0 | HL | L3/L4 Header Length<br>Contains the length of the packet header which MAC splits at L3 or L4 header boundary which the MAC receiver identifies. This field is valid only when the first descriptor bit = 1 (FD = 1).<br>You can write the header data to the buffer 1 address of corresponding descriptor. If the header length is zero, this field is not valid. It implies that MAC does not identify and split the header.<br>This field is valid when you select the enable split header feature option. |

### 72.15.4.1.5.4 RDES3 normal descriptor (Write-back format)

Following table describes the write-back format for the RDES3 normal descriptor.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18:16 | 15 | 14:0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OWN | CTXT | FD | LD | RS2V | RS1V | RS0V | CE | GP | RWT | OE | RE | DE | LT | ES | PL |

Table 536. RDES3 normal descriptor (Write-back format)

| Bit | Name | Description |
|---|---|---|
| 31 | OWN | Own Bit<br>When this field is 1, it indicates that the module DMA owns the descriptor. If this field becomes 0, it indicates that the application owns the descriptor. DMA writes 0 to this field when either of these conditions is true:<br>• DMA completes the packet reception.<br>• The buffers associated with the descriptor are full. |
| 30 | CTXT | Receive Context Descriptor<br>When this field is 1, it indicates that the current descriptor is a context type descriptor. DMA writes 1'b0 to this field for a normal receive descriptor.<br>When CTXT and FD bits are used together, {CTXT, FD}<br>• 2’b00: Intermediate descriptor<br>• 2’b01: First descriptor<br>• 2’b10: Reserved<br>• 2’b11: Descriptor error (due to all 1s)<br>**NOTE**<br>When descriptor error occurs, the receive DMA closes the receive descriptor indicating a descriptor error. This receive descriptor is skipped and the buffer addresses are not used to write the packet data. Receive DMA will write 1 to the CDE bit in DMA_CH#_Status register but not to the RI bit even when IOC = 1, as this is not marked as the last receive descriptor for the packet. The subsequent valid receive descriptor writes the packet data. |
| 29 | FD | First Descriptor<br>When this field is 1, it indicates that this descriptor contains the first buffer of the packet. If the size of the first buffer is 0, the second buffer contains the beginning of the packet and if the size of the second buffer is also 0, the next descriptor will contain the beginning of the packet.<br>See the CTXT bit description for more information on how to use the CTXT bit and FD bit together. |
| 28 | LD | Last Descriptor<br>When this field is 1, it indicates that this descriptor buffers points to the last buffers of the packet. |
| 27 | RS2V | Receive Status RDES2 Valid<br>When this field is 1, it indicates that the status in RDES2 is valid and DMA writes it. This field is valid only when the LD bit of RDES3 = 1. |
| 26 | RS1V | Receive Status RDES1 Valid<br>When this field is 1, it indicates that the status in RDES1 is valid and DMA writes it. This field is valid only when the LD bit of RDES3 = 1. |
| 25 | RS0V | Receive Status RDES0 Valid<br>When this field is 1, it indicates that the status in RDES0 is valid and DMA writes it. This field is valid only when the LD bit of RDES3 = 1. |

*Table continues on the next page...*

Table 536. RDES3 normal descriptor (Write-back format) (continued)

| Bit | Name | Description |
|---|---|---|
| 24 | CE | CRC Error<br>When this field is 1, it indicates that a cyclic redundancy check (CRC) error occurs on the received packet. This field is valid only when the LD bit of RDES3 = 1. |
| 23 | GP | Giant Packet<br>When this field is 1, it indicates that the packet length exceeds the specified maximum Ethernet size of 1518, 1522, or 2000 bytes (9018 or 9022 bytes if jumbo packet enable = 1).<br>**NOTE**<br>Giant packet indicates only the packet length. It does not cause any packet truncation. |
| 22 | RWT | Receive Watchdog Timeout<br>When this field is 1, it indicates that the receive watchdog timer expires when it receives the current packet. The current packet truncates after watchdog timeout. |
| 21 | OE | Overflow Error<br>When this field is 1, it indicates that the received packet is damaged because of buffer overflow in receive FIFO.<br>**NOTE**<br>This field is 1 only when DMA transfers a partial packet to the application. This happens only when the receive FIFO operates in the Threshold mode. In the Store-and-Forward mode, all the partial packets drops completely in the receive FIFO. |
| 20 | RE | Receive Error<br>When this field is 1, it indicates that the gmii_rxer_i signal asserts when the gmii_rxdv_i signal asserts during the packet reception. This error also includes carrier extension error in the GMII and Half-duplex mode. Error can be of less or no extension, or error (rxd!= 0f) during extension. |
| 19 | DE | Dribble Bit Error<br>When this field is 1, it indicates that the received packet has a non-integer multiple of bytes (odd nibbles). This field is valid only in the MII mode. |
| 18:16 | LT | Length/Type Field<br>Indicates if the packet received is a length packet or a type packet. The encoding of the 3 bits is as follows:<br>• 3'b000: The packet is a length packet.<br>• 3'b001: The packet is a type packet.<br>• 3'b011: The packet is a ARP request packet type<br>• 3'b100: The packet is a type packet with VLAN tag<br>• 3'b101: The packet is a type packet with double VLAN tag<br>• 3'b110: The packet is a MAC control packet type<br>• 3'b111: The packet is a OAM packet type |

*Table continues on the next page...*

Table 536. RDES3 normal descriptor (Write-back format) (continued)

| Bit | Name | Description |
|---|---|---|
| | | • 3'b010: Reserved |
| 15 | ES | Error Summary<br>When this field is 1, it indicates the logical OR of these bits, :<br>• RDES3[24]: CRC error<br>• RDES3[19]: Dribble error<br>• RDES3[20]: Receive error<br>• RDES3[22]: Watchdog timeout<br>• RDES3[21]: Overflow error<br>• RDES3[23]: Giant packet<br>• RDES2[17]: Destination address filter fail, when the flexible receive parser is enabled<br>• RDES2[16]: SA address filter fail, when flexible receive parser is enabled<br>This field is valid only when the LD bit of RDES3 = 1. |
| 14:0 | PL | Packet Length<br>Indicates the byte length of the received packet that was transferred to system memory (including CRC).<br>This field is valid when the LD bit of RDES3 = 1 and overflow error bits becomes 0. The packet length also includes the two bytes that appends to the Ethernet packet when IP checksum calculation is enabled and the received packet is not a MAC control packet.<br>This field is valid when the LD bit of RDES3 = 1. When the last descriptor and error summary bits are not 1, this field indicates the accumulated number of bytes that are transferred for the current packet. |

### 72.15.4.1.6 Receive context descriptor

This descriptor is read-only for the application. Only DMA can write to this descriptor. The context descriptor provides information about the extended status related to the last received packet. Bit 30 of RDES3 indicates the context type descriptor.

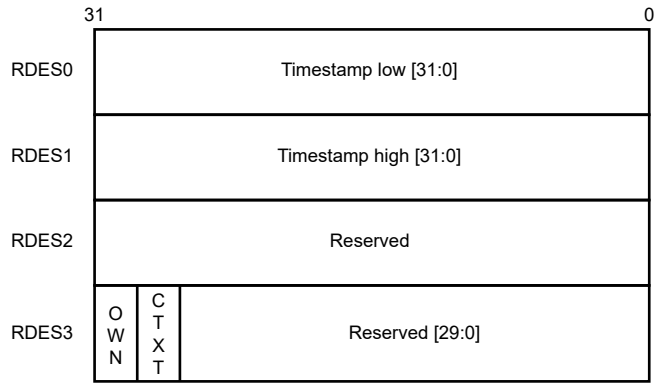



Figure 418. Receive context descriptor format

#### 72.15.4.1.6.1 RDES0 context descriptor

**Table 537. RDES0 context descriptor**

| Bit | Name | Description |
|---|---|---|
| 31:0 | RTSL | Receive Packet Timestamp Low<br>DMA updates this field with least significant 32-bits of the timestamp captured for the corresponding receive packet. When this field and the RTSH field of RDES1 show all-ones value, you must consider the timestamp as corrupt. |

#### 72.15.4.1.6.2 RDES1 context descriptor

**Table 538. RDES1 context descriptor**

| Bit | Field | Description |
|---|---|---|
| 31:0 | RTSH | Receive Packet Timestamp High<br>DMA updates this field with the most significant 32-bits of the timestamp captured for the corresponding receive packet. When this field and the RTSL field of RDES0 show all-ones value, you must consider the timestamp as corrupt. |

#### 72.15.4.1.6.3 RDES2 context descriptor

**Table 539. RDES2 context descriptor**

| Bit | Description |
|---|---|
| 31:0 | Reserved |

#### 72.15.4.1.6.4 RDES3 context descriptor

| 31 | 30 | 29 | 28:0 |
|---|---|---|---|
| OWN | CTXT | DE | Rsvd |

**Table 540. RDES3 context descriptor**

| Bit | Name | Description |
|---|---|---|
| 31 | OWN | Own Bit<br>When this field is 1, it indicates that DMA owns the descriptor. When this field becomes 0, it indicates that the application owns the descriptor. DMA clears this field when either of these conditions is true:<br>• DMA completes the packet reception.<br>• The buffers associated with the descriptor are full. |
| 30 | CTXT | Receive Context Descriptor |

*Table continues on the next page...*

Table 540. RDES3 context descriptor (continued)

| Bit | Name | Description |
|---|---|---|
| | | When this field is 1, it indicates that the current descriptor is a context descriptor. DMA writes 1'b1 to this field for context descriptor.<br>DMA writes 2'b11 to indicate a descriptor error due to all = 1.<br>When CTXT and DE bits are used together {CTXT, DE}<br>• 2'b00: Reserved<br>• 2'b01: Reserved<br>• 2'b10: Context descriptor<br>• 2'b11: Descriptor error<br>Note: When descriptor error occurs, the receive DMA closes the receive descriptor indicating descriptor error. This receive descriptor is skipped and the buffer addresses does not write the packet data. The receive DMA writes 1 to the CDE bit in DMA_CH#_Status register but not to the RI bit even when IOC = 1, because this is not marked as last receive descriptor for the packet. The subsequent valid receive descriptor writes the packet data. |
| 29 | DE | Descriptor Error<br>See the CTXT bit description for more information about how to use the DE bit along with CTXT bit. |
| 28:0 | Rsvd | Reserved |

## 72.15.5 Enhanced descriptor for time-based scheduling

This feature needs 32 bytes enhanced descriptors to enable on all the DMA channels that uses this feature (write 1 to EDSE bit of DMA_CH(#i)_TX_Control register).

The structure of 32-byte descriptor for the context and the normal descriptor in read and write formats are described in the upcoming sections.

### 72.15.5.1 Enhanced normal descriptor - Read (32-bit mode)

These fields are present in the first 16 bytes of the enhanced descriptor format of normal descriptor:

- LTV- Indicates that the launch time (LT) and GSN fields present in the descriptor are valid. The LTV must be = 1 only if the FD bit of the descriptor is not.
- GSN- Indicates the GCL slot number associated with the packet.
- LT- Indicates the launch time associated with the packet.

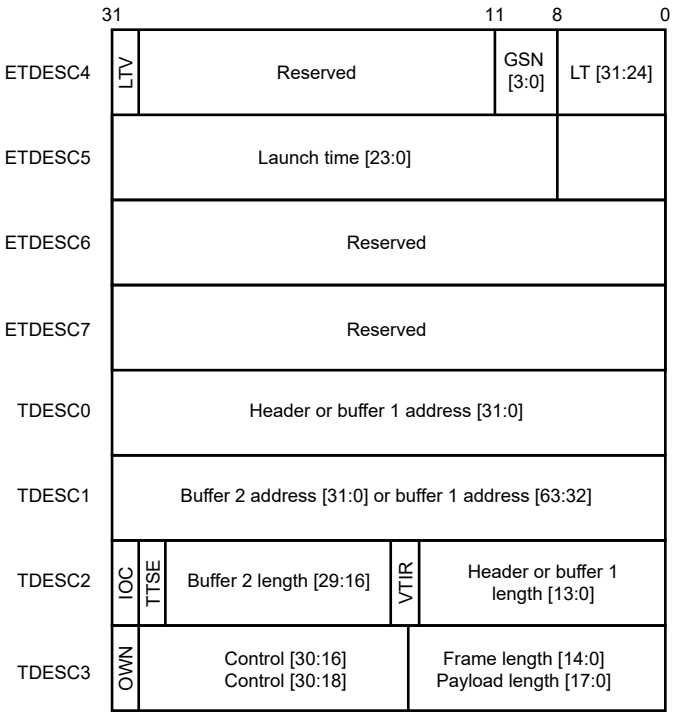



Figure 419. Enhanced normal descriptor - Read (32-bit mode)

### 72.15.5.2 Enhanced normal descriptor (Write, 32-bit mode)

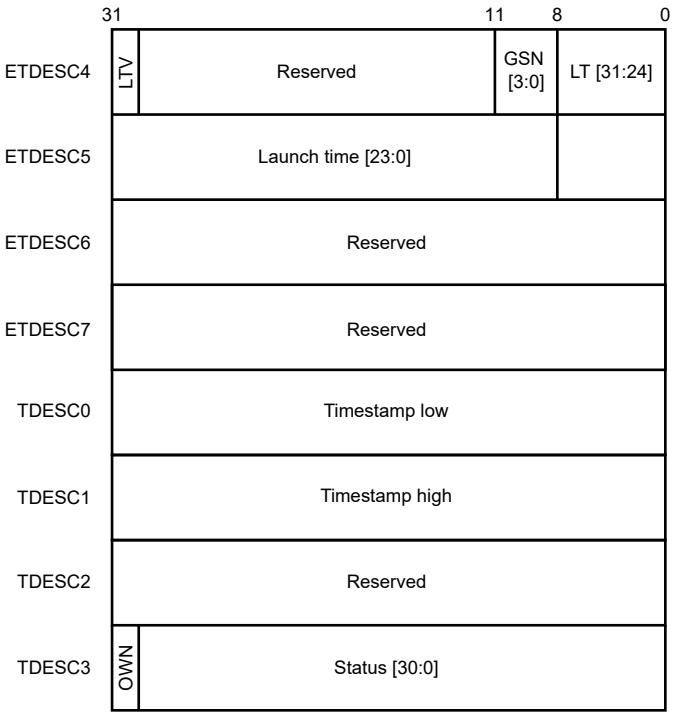



Figure 420. Enhanced normal descriptor write (32-bit mode)

**NOTE**

- All the enhanced descriptors, example, assumes a 32-bit data width configurations. However, the same definitions are valid in 64-bit and 128-bit configurations.
- In both the write back formats, you cannot modify the extended 16 bytes. The remaining 16 bytes (TDESC0 to TDESC3) are written back per the previous 16 bytes descriptor format.
- When you enable the fetch time it overrides the AV slot function
- When an unaligned new GCL list (with a different CTR) is installed it is recommended not to have traffic during the installation of the new list. Any traffic during the switching of the lists might have unpredictable behavior regarding fetch, launch, and launch expiry because the CTR and BTR values get updated when the frame is processed

### 72.15.5.3 Enhanced context descriptor (Read, 32-bit mode)

The first 16 bytes of the enhanced context descriptor (ETDESC4 to ETDESC7) are reserved and must be = 0. The fields present in the last bytes of the descriptor (TDESC0 to TDESC3) are same as the context descriptor (TDESC0 to TDESC3) in 16 byte format.

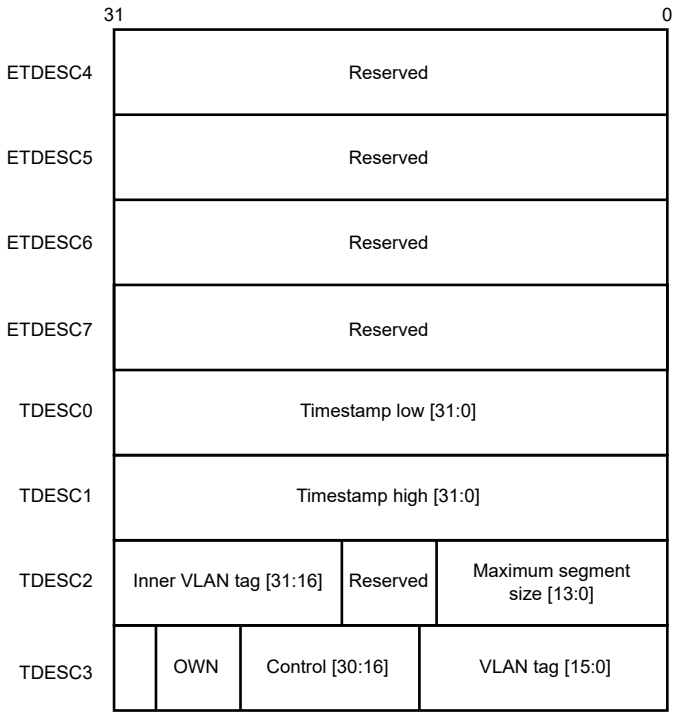



**Figure 421. Enhanced context descriptor - Read**

#### 72.15.5.4 Enhanced context descriptor (Write, 32-bit mode)

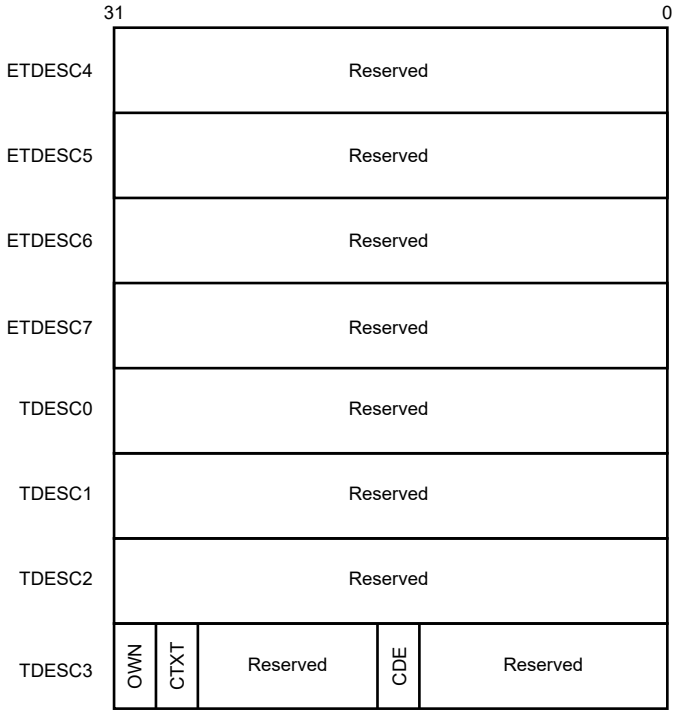



**Figure 422. Enhanced context descriptor - Write**

## 72.16 Programming

This section and all its sub-sections are Synopsys Proprietary. Used with permission.

This section provides the instructions for initializing the DMA or MAC registers in the proper sequence.

### 72.16.1 Initializing DMA

Perform these steps to initialize DMA:

1. Provide a software reset. This resets all the MAC internal registers and logic (bit-0 of DMA_Mode).
2. Wait for the reset process (poll bit 0 of DMA_Mode which is only clears after the reset operation completes) to complete.
3. Program these fields to initialize DMA_SysBus_Mode:
    a. DMA_SysBus_Mode[AAL]
    b. Fixed burst or undefined burst
    c. Burst mode values in case of AHB bus interface, OSR_LMT in case of AXI bus interface.
    d. Select the maximum burst length possible on the AXI bus (bits [7:1]), if fixed length value is enabled.
4. Create a descriptor list for a transmit and receive. Also, ensure that DMA (write 1 to bit 31 of descriptor TDES3/RDES3) owns the descriptors. See Descriptors for more information about descriptors.
5. Program the transmit and receive ring length registers (DMA_CH(#i)_TxDesc_Ring_Length (for i = 0; i <= 1) and DMA_CH(#i)_RxDesc_Ring_Length (for i = 0; i <= 1)). The programmed ring length must be at least 4.

**NOTE**

The descriptor address from the start to the end of the ring must not cross the 4 GB boundary.

6. Initialize the receive and transmit descriptor list address with the base address of the transmit and receive descriptor (DMA_CH(#i)_TxDesc_List_Address (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1), DMA_CH(#i)_RxDesc_List_Address (for i = 0; i <= DWC_EQOS_NUM_DMA_RX_CH-1)). Also, program the transmit and receive tail pointer registers that indicates DMA about the available descriptors (DMA_CH(#i)_TxDesc_Tail_Pointer (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) and DMA_CH(#i)_RxDesc_Tail_Pointer (for i = 0; i <= DWC_EQOS_NUM_DMA_RX_CH-1)).

   **NOTE**

   For 40-bit or 48-bit addressing mode, program the higher address List registers (DMA_CH[n]_TxDesc_List_HAddress, DMA_CH[n]_RxDesc_List_HAddress).

   The tailpointer registers must advance to the location immediately after the descriptors that are set so that DMA is aware that the additional descriptors are available.

7. Program the settings of these registers for the parameters like maximum burst-length (PBL) which DMA initiates, descriptor skip lengths, OSP in case of TxDMA, RBSZ in case of RxDMA, and so on:
   - DMA_CH(#i)_Control (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1)
   - DMA_CH(#i)_TX_Control (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1)
   - DMA_CH(#i)_RX_Control (for i = 0; i <= DWC_EQOS_NUM_DMA_RX_CH-1)
8. Program the DMA_CH(#i)_Interrupt_Enable (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) register to enable the interrupts.
9. Write 1 to SR (bit 0) of the DMA_CH(#i)_RX_Control (for i = 0; i <= DWC_EQOS_NUM_DMA_RX_CH-1) and ST (bit 0) of the DMA_CH(#i)_TX_Control (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) register to start the receive and transmit DMAs.
10. Repeat steps 4 to 9 for all the transit DMA and receive DMA channels selected in the hardware.

### 72.16.2 Initializing MTL registers

The transaction layer (MTL) registers must initialize to establish the transmit and receive operating modes and commands.

Perform these steps to initialize the MTL registers:

1. Program MTL_Operation_Mode[SCHALG] and MTL_Operation_Mode[RAA] to initialize the MTL operation in case of multiple transit and receive queues.
2. Program the receive queue to DMA mapping in MTL_RxQ_DMA_Map0.
3. Program these fields to initialize the mode of operation in MTL_TxQ0_Operation_Mode.
   a. MTL_TxQ0_Operation_Mode[TSF] or MTL_TxQ0_Operation_Mode[TTC] in case of Threshold mode.
   b. MTL_TxQ0_Operation_Mode[TXQEN] to value 2'b10 to enable transmit queue0.
   c. MTL_TxQ0_Operation_Mode[TQS]
4. Program these fields to initialize the mode of operation in MTL_RxQ0_Operation_Mode:
   a. MTL_RxQ0_Operation_Mode[RSF] or MTL_RxQ0_Operation_Mode[RTC] in case of Threshold mode.
   b. Flow control activation and de-activation thresholds for MTL receive FIFO (MTL_RxQ0_Operation_Mode[RFA] and MTL_RxQ0_Operation_Mode[RFD]).
   c. Error Packet and undersized good Packet forwarding enable (MTL_RxQ0_Operation_Mode[FEP] and MTL_RxQ0_Operation_Mode[FUP]).
   d. MTL_RxQ0_Operation_Mode[RQS]
5. Repeat previous two steps for all MTL transit and receive queues selected in the configuration.

### 72.16.3 Initializing MAC

The MAC configuration registers establish the operating mode of MAC. These registers must initialize before initializing DMA.

You can perform these MAC initialization operations after DMA initialization. If the MAC initialization completes before DMA is configured, enable the MAC receiver (last step in the following sequence) only after the DMA is active. Otherwise, the received frames fill the receive FIFO and overflow.

1. Provide the MAC address registers: MAC_Address0_High and MAC_Address0_Low. If more than one MAC address is enabled, , program the MAC addresses appropriately.

2. Program these fields to set the appropriate filters for the incoming frames in MAC_Packet_Filter:

a. Receive all

b. Promiscuous mode

c. Hash or perfect filter

d. Unicast, multicast, broadcast, and control frames filter settings

3. Program these fields for proper flow control in MAC_Q0_Tx_Flow_Ctrl:

a. Pause time and other Pause frame control bits

b. Transmit flow control bits

c. Flow control busy

4. Program MAC_Interrupt_Enable as required, and if applicable, for your configuration.

5. Program the appropriate fields in MAC_Configuration. For example Inter-packet gap when transmission and jabber disable.

6. Write 1 to MAC_Configuration[RE] and MAC_Configuration[TE] to start the MAC transmitter and receiver.

### 72.16.4 Performing normal receive and transmit operation

During normal operation of the module, read the normal and transmit interrupts, poll the descriptors, suspend the DMA (if it does not own descriptors), and read the values of the current host transmitter or receiver descriptor pointers for debugging.

For normal operation, perform these steps:

1. Read the interrupt status for normal transmit and receive interrupts. Then poll the descriptors, read the status of the descriptor which the host owns (either transmit or receive).

2. Set appropriate values for the descriptors, ensure that the DMA owns the transmit and receive descriptors to resume the data transmission and reception.

3. If DMA does not own the descriptors (or no descriptor is available), DMA enters into SUSPEND state. The transmission or reception can resume by freeing the descriptors and writing the descriptor tail pointer to Tx/Rx tail pointer register (DMA_CH[n]_TxDesc_Tail_Pointer and DMA_CH[n]_RxDesc_Tail_Pointer).

4. Read the current host transmitter or receiver descriptor address pointer values for the debug process (DMA_CH[n]_Current_App_TxDesc and DMA_CH[n]_Current_App_RxDesc register).

5. Read the current host transmit buffer address pointer and receive buffer address pointer values for the debug process (Register DMA_CH[n]_Current_App_TxBuffer and DMA_CH[n]_Cur-rent_App_RxBuffer).

### 72.16.5 Stopping and starting transmission

Perform these steps to pause the transmission for some time. The steps are provided for channel 0.

1. Write 0 to bit 0 (ST) of DMA_CH(#i)_TX_Control (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) register to disable the transmit DMA (if applicable) .

2. Wait for any previous frame transmissions to complete. Read the appropriate fields of MTL_TxQ0_Debug (MTL_TxQ0_Debug[TRCSTS] is not 01 and MTL_TxQ0_Debug[TXQSTS] = 0) to check this.

3. Write 0 to MAC_Configuration[RE] and MAC_Configuration[TE] to disable the MAC transmitter and MAC receiver.

4. Disable the receive DMA (if applicable), after you ensure that the data in the receive FIFO transfers to the system memory (read the appropriate fields MTL_RxQ0_Debug[PRXQ] = 0 and MTL_RxQ0_Debug[RXQSTS] = 00).

5. Ensure that both the transit queue and receive queue are empty (MTL_TxQ0_Debug[TXQSTS] = 0 and MTL_RxQ0_Debug[RXQSTS] = 0).

6. Restart the operation by first starting the DMAs, and then enable the MAC transmitter and receiver.

**NOTE**

- Do not change the configuration (such as duplex mode, speed, port, or loop back) when the MAC is actively transmitting or receiving. You can change these parameters only when the MAC transmitter and receiver are not active.
- Similarly, do not change the DMA related configuration when transmit and receive DMA are active.

### 72.16.6 Programming guidelines for switching to new descriptor list in RxDMA

Switching to a new descriptor list is different in the receive DMA as compared to the transit DMA. It is permitted when the RxDMA is in SUSPEND state as described below:

- RxDMA prepares the descriptors in advance.
- If the RxDMA goes to SUSPEND state because the descriptors are not available, a major failure occurs (you are not able to free the filled-up descriptors or buffers). If this issue does not rectify immediately, frames will be lost because of an RxFIFO overflow. Therefore, you can create a new descriptor list and program the RxDMA to start using it immediately, without going into STOP state.

### 72.16.7 Programming guidelines for multi-channel multi-queuing

#### 72.16.7.1 Transmit

1. Program the transmit queue size in TQS field of MTL_TxQ[n]_Operation_Mode register. Based on the value programmed in TQS field, you can determine the queue size.

In the transmit operation, the number of channels is equal to the number of the queues, because of this reason, the channel-to-queue mapping is fixed.

2. Enable the queue in TXQEN in the corresponding MTL_TxQ[n]_Operation_Mode register to use the queue. In DMA configurations, enable the ST field of DMA_CH[n]_Tx_Con-trol register and corresponding TXQEN in MTL_TxQ[n]_Operation Mode register.

3. Program the scheduling method in MTL_Operation_Mode[SCHALG].

4. Program MTL_TxQ[n]_Quantum_Weight register for DCB queue per the selected algorithm. In case of CBS algorithm in AVB queues program the MTL_TxQ[n]_ETS_Control, MTL_TxQ[n]_SendSlopeCredit, MTL_TxQ[n]_HiCredit, and MTL_TxQ[n]_LoCredit registers as required.

5. Program the MAC_TxQ_Prty_Map0 register to assign a fixed priority to the queue, if DCB is enabled and PFC function is required. You can use this assigned priority to determine if the corresponding queue must stop transmitting packet on the basis of the received PFC packet.

#### 72.16.7.2 Receive

1. Program the receive queue size in RQS field of MTL_RxQ[n]_Operation_Mode register. Determine the size of the queue, based on the value programmed in RQS field.
2. Enable the receive queues 0 to 7 in the fields RXQ0EN to RXQ7EN in MAC_RxQ_Ctrl0 for AV or DCB. In DMA configurations, SR bit of statically or dynamically map DMA_CH[n]_Rx_Control register and the corresponding RXQ[n]_EN in MAC_RxQ_Ctrl0 is enabled.

3. Based on these packet types MAC routes the receive packets to the receive queues :
    a. AV PTP packets: Based on the programming of MAC_RxQ_Ctrl1[PTPQ].
    b. AV untagged control packets: Based on the programming of MAC_RxQ_Ctrl1[AVCPQ].
    c. Data center bridging (DCB) related link layer discovery protocol (LLDP) packets. Program DCBCPQ in MAC_RxQ_Ctrl1 to indicate MAC which queue should get the DCB packets.
    d. VLAN tag priority field in VLAN tagged packets: Program PSRQ7-0 of the MAC_RxQ_Ctrl2 and MAC_RxQ_Ctrl3 register for routing the tagged packets based on the received packets USP (user priority) field to the receive queues 0 to 7.
    e. Route the AV tagged control and data packets based on PSRQ field of MAC_RxQ_Ctrl2 and MAC_RxQ_Ctrl3 registers.

**NOTE**

The priorities set in PSRQ7-0 must be unique.

4. If multiple receive DMA channels are enabled, you must perform the following programming for proper arbitration and mapping:
    a. Program MTL_Operation_Mode[RAA] to select the arbitration algorithm to decide which RxQ is read out from the RxFIFO memory.
    b. Program the MTL_RxQ[n]_Control register to decide the weights and the packet arbitration for each RxQ.
    c. If you program the static mapping in MTL_RxQ_DMA_Map[n] register (RXQ[n]DADMACH becomes 0), fields RXQx2DMA and others are programmed to select the channel to map each queue.
    d. Write 1 to RXQ[n]DADMACH field in MTL_RxQ_DMA_Map0 to select the dynamic mapping of packets in each receive queue.
    e. In dynamic channel mapping, the value of DCS field in the lowest MAC address register the decides the routing of a packet to a specific RxDMA channel.

#### 72.16.7.2.1 Programming guidelines for recovering from DMA channel failure

When the DMA channel issues a bus error, follow these steps to recover from the failure.

Recovering from the receive DMA channel failure.

Perform these steps if you get bus error in the receive DMA channel:

1. Write 1 to DMA_CH0_Rx_Control[RPF]. This flushes all the packets one after the other.

This step is optional. However, writing 1 to this field prevents head-of-line blocking in the receive queues when packets sent to the RXDMA are stopped due to the bus error.

2. Re-program the specific registers of the DMA channel.

3. Start the DMA channel.

Recovering from the transmit DMA channel failure.

1. Stop the specific DMA channel, even if it is in active state.

2. Flush the corresponding MTL queue.

3. Re-program the specific registers of the DMA channel.

4. Start the DMA channel.

**NOTE**

Due to the known limitations in the design, reprogramming the DMA channel registers might not be always successful in recovering from a bus error. If the module is not fully functional after reprogramming the DMA, you can issue a soft reset to recover from the bus error.

## 72.16.8 Programming guidelines for GMII link state transitions

### 72.16.8.1 Transmit and receive clocks running when link down

Perform these steps when the link is down but the transmit and receive clocks are running:

**NOTE**

The steps are provided for channel 0.

1. Write 0 to bit 0 (ST) of DMA_CH(#i)_TX_Control (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) register to disable the transmit DMA (if applicable).

2. Write 0 to MAC_Configuration[RE] to disable the MAC receiver.

3. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate fields of MTL_TxQ0_Debug (MTL_TxQ0_Debug[TRCSTS] is not 01) or, flush the Tx FIFO for faster empty operation.

4. Write 0 to MAC_Configuration[TE] to disable the MAC transmitter.

5. Make sure that both the transit queue and receive queue are empty (MTL_TxQ0_Debug[TXQSTS] and MTL_RxQ0_Debug[RXQSTS] are 0).

6. Read the PHY registers to know the latest configuration and accordingly program the MAC registers, after the link is up.

7. Start the transit DMA to restart the operation, and then enable the MAC transmitter and receiver.

You do not disable the receive DMA because the receiver is disabled and the FIFO does not get any data in the receive FIFO.

### 72.16.8.2 Transmit and receive clocks stopped when link down

Perform these steps when the link is down and the transmit and receive clocks are stopped:

**NOTE**

The steps are provided for channel 0.

1. Write 0 to MAC_Configuration[RE] and MAC_Configuration[TE] to disable the MAC transmitter and receiver. This does not take effect immediately as the clocks are absent.

2. Wait until the link is up and the clocks are restored.

3. Wait for the transfer of any partial frame, if any to complete at time of stopping of the transmit or receive clock. To check this, read MAC_Debug (should be all-zero). Some old packets may still remain in the TXFIFO because the MAC transmitter is stopped.

4. Read the PHY registers to know the latest operating mode and accordingly program the MAC registers.

5. Write 1 to MAC_Configuration[RE] and MAC_Configuration[TE] to restart the MAC transmitter and receiver.

## 72.16.9 Programming guidelines for IEEE 1588 timestamping

### 72.16.9.1 Initialization guidelines for system time generation

Write 1 MAC_Timestamp_Control[TSENA] to enable the timestamp feature. However, it is essential to initialize the timestamp counter after MAC_Timestamp_Control[TSENA] = 1. Perform these steps during the module initialization:

1. Write 0 to the bit 16 of MAC_Interrupt_Enable to mask the timestamp trigger interrupt.
2. Write 1 to MAC_Timestamp_Control[TSENA] to enable timestamping.
3. Program MAC_Sub_Second_Increment based on the PTP clock frequency.
4. Program MAC_Timestamp_Addend and write 1 to MAC_Timestamp_Control[TSADDREG], if you are using the fine correction approach.
5. Poll MAC_Timestamp_Control until MAC_Timestamp_Control[TSADDREG] = 0.
6. Program MAC_Timestamp_Control[TSCFUPDT] to select the fine update method (if required).

7. Program MAC_System_Time_Seconds_Update and MAC_System_Time_Nanoseconds_Update with the appropriate time value.
8. Write 1 to MAC_Timestamp_Control[TSINIT].
9. Initialize the timestamp counter with the value written in the timestamp update registers so that the timestamp counter can start operation. If one-step timestamping is enabled
    a. Program Bit 27 of the TDES3 context descriptor, to enable one-step timestamping.
    b. Program MAC_Timestamp_Ingress_Asym_Corr and MAC_Timestamp_Egress_Asym_Corr to update the correction field in PDelay_Req PTP messages.
10. Enable the MAC receiver and transmitter for proper timestamping.

**NOTE**

If the timestamp operation is disabled by writing 0 to MAC_Timestamp_Control[TSENA], repeat all these steps to restart the timestamp operation.

### 72.16.9.2 System time correction

#### 72.16.9.2.1 Coarse correction method

Perform these steps to synchronize or update the system time in one process (coarse correction method):

1. Set the offset (positive or negative) in the timestamp registers that is MAC_System_Time_Seconds_Update and MAC_System_Time_Nanoseconds_Update.

2. Write 1 to MAC_Timestamp_Control[TSUPDT].

The value in the timestamp update registers is added to or subtracted from the system time when MAC_Timestamp_Control[TSUPDT] = 0.

#### 72.16.9.2.2 Fine correction method

Perform these steps to synchronize or update the system time to reduce system-time jitter (fine correction method):

1. Calculate the rate by which you want to increment the system time slower or faster, with the help of the algorithm.

2. Update MAC_Timestamp_Addend with the new value and write 1 to MAC_Timestamp_Control[TSADDREG].

3. Wait for the time for which you want the new value of the addend register to be active. You can do this by enabling the timestamp trigger interrupt after the system time reaches the target value.

4. Program the required target time in MAC_PPS[n]_Target_Time_Seconds register and MAC_PPS[n]_Target_Time_Nanoseconds register.

5. Enable the timestamp interrupt in MAC_Interrupt_Enable[TSIE].

6. Write 1 to bit 4 in MAC_Timestamp_Control.

7. Read MAC_Interrupt_Status when this trigger causes an interrupt.

8. Reprogram MAC_Timestamp_Addend with the old value and write 1 to MAC_Timestamp_Control[TSADDREG] again.

## 72.16.10 Programming guidelines for AV feature

After you enable the AV feature in the module controller, follow these programming tasks:

- Initializing the DMA for QOS-AHB configurations only
- Enabling slot number checking
- Enabling average bits per slot reporting
- Disabling flow control for AV-enabled queues (transmit and receive flow control)

### 72.16.10.1 Initializing the DMA in audio video feature

The first step to program the AV feature in a QOS-AHB configuration is to initialize the DMA.

Use this initialization sequence for QOS-AHB/QOS-AXI/QOS-AXI4 configurations with AV feature.

1. Provide a software reset to reset all the QOS internal registers and logic (DMA_Mode[SWR]).

2. Wait for the reset process to complete. Poll DMA_Mode[SWR], which clears only after the reset operation completes.

3. Set the values in DMA_Mode to program the fields to initialize the DMA register.

4. Create a descriptor list for transmit and receive. In addition, ensure that the DMA owns the transmit and receive descriptors. When you use OSF mode, at least two transmit descriptors are required.

See Descriptors for more information about descriptors.

5. Ensure that you create three or more different transmit or receive descriptors in the list before reusing any descriptors.

6. Program the transmit and receive ring length registers (DMA_CH(#i)_TxDesc_Ring_Length (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) and DMA_CH(#i)_RxDesc_Ring_Length (for i = 0; i <= DWC_EQOS_NUM_DMA_RX_CH-1)). The ring length programmed must be at least 4.

7. Initialize receive and transmit descriptor list address with the base address of the transmit and receive descriptor (DMA_CH(#i)_TxDesc_List_Address (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1), DMA_CH(#i)_TxDesc_List_Address (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1)). In addition, you must program the transmit and receive tail pointer registers to indicate the DMA about the available descriptors (DMA_CH(#i)_TxDesc_Tail_Pointer (for i = 0; i <= DWC_EQOS_NUM_D-MA_TX_CH-1) and DMA_CH(#i)_RxDesc_Tail_Pointer (for i = 0; i <= DWC_EQOS_NUM_D-MA_RX_CH-1)).

8. Program these fields to initialize the mode of operation in MTL_TxQ0_Operation_Mode:

a. MTL_TxQ0_Operation_Mode[TSF]

b. MTL_TxQ0_Operation_Mode[TTC]

c. MTL_TxQ0_Operation_Mode[TXQEN] to value 2'b10 to enable transmit queue0

d. MTL_TxQ0_Operation_Mode[TQS]

9. Enable the interrupts by programming DMA_CH(#i)_Interrupt_Enable (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) register.

10. Repeat steps 4 through 9 for all the additional channels of AV feature.

11. Program the AV queues CBS control register, idleSlope, sendSlope, hiCredit, and loCredit registers.

12. Write 1 to bit 0 of DMA_CH(#i)_TX_Control (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) and DMA_CH(#i)_RX_Control (for i = 0; i <= DWC_EQOS_NUM_DMA_RX_CH-1) register to start the receive and transmit DMA.

### 72.16.10.2 Enabling slot number checking

You can enable slot number checking feature if you specifies the intervals at which the DMA channels map to AV queues fetch the frames from the AHB/AXI system bus.

You must complete these steps after step 11 and before step 12 of Initializing the DMA in audio video feature .

You can use the slot number check feature to specify the intervals at which the DMA channels map to AV queues fetch the frames from the AHB/AXI system bus. This feature is useful for an uniform and periodic transfer of the AV traffic from the host memory and it is available only when you enable timestamping and programs MAC_Sub_Second_Increment. Perform these steps to enable the slot number checking:

1. Follow the steps described in Initialization guidelines for system time generation to enable timestamping.

2. Ensure that the SLOTNUM field (bits 22:19) of TDES3 normal descriptor (Read format) contains a valid slot number. You can read the current reference slot number from the DMA_CH(#i)_Slot_Func-tion_Control_Status (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) register.

3. Write 1 to bit 0 (ESC) of the slot function control and status register of a channel to enable the slot number checking.

#### 72.16.10.3 Enabling average bits per slot reporting

You can enable the reporting of average bits transmitted in a slot.

The CBS status register of the additional AV channels provide information about the average bits that are transmitted in a slot. You can asynchronously read this register to retrieve information about the average bits transmitted per slot. Perform these steps to enable average bits per slot reporting:

1. Follow the steps as described in the Initialization guidelines for system time generation to enable timestamping.

2. Program SLC bits [6:4], of the MTL_TxQ[n]_ETS_Control register of a channel with the number of slots over which the average transmitted bits per slot are computed.

3. Enable bit 9(ABPSSIE) of the MTL_Q[n]_Interrupt_Control_Status register of a channel to generate the average bits per slot interrupt.

**NOTE**

- The frequency of this interrupt depends on the value programmed in step 2. For example, when you program value 0 in the SLC field, the interrupt generates at every 125 microsecond.
- You can disable this interrupt to stop the interrupt flooding, when it is not required.

4. Read ABS bits [16:0], of the MTL_TxQ[n]_ETS_Status register of a channel on each interrupt.

**NOTE**

You can read the ABS fields in the polling mode even if ABPSIE bit is not enabled. When bit 1 (ABPSIS) of the MTL_TxQ[n]_ETS_Status register is 1, it indicates that a new value is updated in the ABS field.

#### 72.16.10.4 Disabling flow control for AV enabled queues

##### 72.16.10.4.1 Transmit flow

Program the EHFC (Enable Hardware Flow Control) field of the corresponding receive queue's MTL_RxQ[n]_Operation_Mode register to 0.

##### 72.16.10.4.2 Receive flow control

Program PSTQ[n] field, corresponding to AV enabled transit queue in MAC_TxQ_Prty_Map0/1 register to 0.

#### 72.16.10.5 Programming guidelines for flexible pulse-per-second output

After you enable the flexible pulse-per-second output feature in the module controller, you can perform these tasks:

- Generating single pulse on PPS
- Generating next pulse on PPS
- Generating a pulse train on PPS
- Generating an interrupt without affecting the PPS

##### 72.16.10.5.1 Generating single pulse on PPS

Perform these steps to generate single pulse on PPS:

1. Program 11 or 10 (for interrupt) in TRGTMODSEL bits [6:5], of MAC_PPS_Control. This instructs MAC to use the target time registers (MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds) for a start time of the PPS signal output.
2. Program the start time value in the target time registers (MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds).
3. Program the PPS signal output width in MAC_PPS(#i)_Width (for i = 0; i <= 3) register.

4. Program PPSCMD bits [3:0], of MAC_PPS_Control to 0001. This instructs MAC to generate single pulse on the PPS signal output at the time programmed in the target time registers.

#### 72.16.10.5.2 Generating next pulse on PPS

When the PPSCMD is executed (PPSCMD bits = 0), you can give the cancel start command (PPSCMD=0011) before the programmed start time elapses to cancel the pulse generation. You can also program the behavior of the next pulse in advance.

Follow these steps to program the next pulse:

1. Program the start time for the next pulse in the target time registers. This time must be more than the time at which the falling edge occurs for the previous pulse.

2. Program the next PPS signal output width in MAC_PPS(#i)_Width (for i = 0; i <= DWC_EQOS_PPS_OUT_NUM-1) register.

3. Program PPSCMD bits [3:0], of MAC_PPS_Control to generate a single pulse after the time at which the previous pulse is de-asserted. This instructs MAC to generate single pulse on the PPS signal output, at the time programmed in target time registers.

If you give this command before the previous pulse becomes low, then the new command overwrites the previous command and the QOS may generate only 1 extended pulse.

#### 72.16.10.5.3 Generating a pulse train on PPS

Perform these steps to generate a pulse train on PPS:

1. Program 11 or 10 (for interrupt) in TRGTMODSEL bits [6:5] , of MAC_PPS_Control. This instructs MAC to use the target time registers for the start time of the PPS signal output.

2. Program the start time value in the target time registers.

3. Program the interval value between the train of pulses on the PPS signal output in MAC_PPS(#i)_Interval (for i = 0; i <= DWC_EQOS_PPS_OUT_NUM-1) register.

4. Program the PPS signal output width in MAC_PPS(#i)_Width (for i = 0; i <= DWC_EQOS_PPS_OUT_NUM-1) register.

5. Program PPSCMD bits [3:0], of MAC_PPS_Control to 0010. This instructs MAC to generate train of pulses on the PPS signal output with start time programmed in target time registers.

By default, the PPS pulse train is free-running unless the STOP pulse train at time or STOP pulse train immediately commands stop it.

6. Program the stop value in the target time registers. Ensure that bit 31 (TSTRBUSY) of MAC_PPS(#i)_Target_Time_Nanoseconds (for i = 0; i <= DWC_EQOS_PPS_OUT_NUM-1) register becomes 0 before programming the target time registers again.

7. Program PPSCMD field [bit 3:0] of MAC_PPS_Control to 0100. This stops the train of pulses on the PPS signal output after the programmed stop time specified in step 6 elapses.

You can program 0101 in PPSCMD field [bit 3:0] of MAC_PPS_Control to stop the pulse train at any time. Similarly, you can program 0110 in PPSCMD field [bit 3:0] of MAC_PPS_Control before the time (programmed in step 6) elapses to cancel the stop pulse train command (given in Step 7). Also, program 0011 in PPSCMD field [bit 3:0] of MAC_PPS_Control before the programmed start time (in Step 2) elapses to cancel the pulse train generation.

#### 72.16.10.5.4 Generating an interrupt without affecting the PPS

MAC_PPS_Control TRGTMODSEL bits [6:5] enables you to program the target time registers to perform any one of these actions:

- Generate only interrupts.
- Generate interrupts and the PPS start and stop time.
- Generate only PPS start and stop time.

Perform these steps to program the target time registers to generate only interrupt event:

1. Program 00 (for interrupt) in MAC_PPS_Control TRGTMODSEL bits [6:5]. This instructs the MAC to use the target time registers for target time interrupt.

2. Program a target time value in the target time registers. This instructs the MAC to generate an interrupt when the target time elapses.

If TRGTMODSEL bits [6:5], are changed (for example, to control the PPS) then over-write the interrupt generation with the new mode and new programmed target time register value.

### 72.16.10.6 Programming guidelines for VLAN filtering on receive

Perform these steps to program VLAN filtering on receive:

1. Program MAC_VLAN_Tag register for the following fields to select the filtering method:

a. ETV: Enables 12-Bit VLAN tag comparison or 16-bit VLAN tag comparison.

b. VTHM: VLAN tag hash table match enable.

c. ERIVLT: Enables inner VLAN tag or outer VLAN Tag (to enable the inner or outer VLAN tag filtering, writing 1 to MAC_VLAN_Tag_Ctrl[EDVLP] enables the double VLAN processing).

d. ERSVLM: Enables receive S-VLAN match or C-VLAN match (write 1 to MAC_VLAN_Tag_Ctrl[ESVL] to enable S-VLAN processing).

e. DOVLTC: Ignores VLAN type for tag match

f. VTIM: Enables VLAN tag inverse match instead of the normal VLAN tag matching

2. Program VL of MAC_VLAN_Tag register for the 12-bit or 16-bit VLAN tag.

3. Program MAC_VLAN_Hash_Table, if hash filtering of VLAN tag is enabled. When MAC_VLAN_Tag_Ctrl[ETV] becomes 0, it indicates that the upper 4 bits of the calculated CRC-32 of VLAN tag are inverted and index the content of MAC_VLAN_Hash_Table. When MAC_VLAN_Tag_Ctrl[ETV] is 1, it indicates that the upper 4 bits of calculated CRC-32 of VLAN Tag index the content of MAC_VLAN_Hash_Table. For example, when MAC_VLAN_Tag_Ctrl[ETV] is 1, a hash value of 4b'1000 selects bit 8 of MAC_VLAN_Hash_Table. When MAC_VLAN_Tag_Ctrl[ETV] becomes 0 a hash value of 4'b1000 selects bit 7 of MAC_VLAN_Hash_Table.

### 72.16.10.7 Programming guidelines for extended VLAN filtering and routing on receive

Perform these steps, for the indirect access of the per VLAN tag registers:

- Write
  - Write the required data into MAC_VLAN_Tag_Data.
  - Program MAC_VLAN_Tag_Ctrl[OFS] with the required filter register's offset and command type to MAC_VLAN_Tag_Ctrl[CT]. For a write command, set this field to 0.
  - Write 1 to MAC_VLAN_Tag_Ctrl[OB] and wait till it becomes 0 to do the next write. This will guarantee that you have programmed the appropriate VLAN Tag Filter register.
- Read
  - Program MAC_VLAN_Tag_Ctrl[OFS] with the required register's offset and command type to MAC_VLAN_Tag_Ctrl[CT]. For a read command, set this field to 1.
  - Write 1 to MAC_VLAN_Tag_Ctrl[OB] and wait till it becomes 0. The appropriate value of the VLAN Tag Filter register is available in MAC_VLAN_Tag_Data.

### 72.16.10.8 Programming sequence for queue/channel based VLAN inclusion register

Perform these steps to program the queue/channel-based VLAN inclusion register:

**NOTE**

When MAC_VLAN_Incl[CBTI] = 1 you cannot access the indirect VLAN include registers.

1. Write 1 to MAC_VLAN_Incl[CBTI], to enable queue or channel based VLAN tag insertion on all the transmitted packets. This field must be 1 before any indirect access to the queue or channel specific MAC_VLAN_Incl(#i) register.

2. Program the VLAN tag and VLAN type to insert in packets from a particular queue or channel in MAC_VLAN_Incl[VLT] and MAC_VLAN_Incl[CSVL], corresponding offset address in ADDR field (0 for queue/channel 0, 1 for queue/channel 1, and so on) must be set. Write 0 to MAC_VLAN_Incl[RDWR] to indicate write access. The write to byte 0 (byte 3 in Big Endian mode) of MAC_VLAN_Incl initiates access to indirect access MAC_VLAN_Incl(#i) register.

3. Write 1 to MAC_VLAN_Incl[BUSY] via the module to indicate the progress of access to indirect access MAC_VLAN_Incl(#i) register. After the access completes, MAC_VLAN_Incl[BUSY] = 0. The application must not attempt subsequent access to MAC_VLAN_Incl(#i) register when the MAC_VLAN_Incl[BUSY] is 1.

4. Repeat step 2 and step 3 to program VLAN tag and VLAN type to insert in packets from the remaining queues or channels. The application must ensure that the required VLAN tag and VLAN type for all the queues or channels are programmed, otherwise an unintended VLAN tag and VLAN type might be inserted.

## 72.16.11 Programming guidelines for EST

Program the gate control values and time intervals in MTL_EST_Status[SWOL] along with the other EST related registers that are described in EMAC register descriptions to appropriate values. The upcoming sub-sections provide step by step details for programming the GCL and the other EST related registers.

### 72.16.11.1 Programming the GCL and GCL linked registers

Perform these steps to program the GCL and the four other registers implemented per GCL:

1. Access the GCL and the four other GCL-linked registers via indirect addressing using MTL_EST_GCL_Control and MTL_EST_GCL_Data. MTL_EST_Status[SWOL] indicates whether the software owns GCL0 or GCL1.
2. Write the 32-bit write data to MTL_EST_GCL_Data, to program the GCL. Then program MTL_EST_GCL_Control to write the write address and other control information.
3. In MTL_EST_GCL_Data, write data consists of up to 8 bits (configurable) of gate controls and up to 24 bits (configurable) of time interval. Programming a 0 indicates gate close and programming a 1 indicates gate open. For a 4-TC and 20-bit time interval configuration, the data width is 24-bits and the remaining 8-bits are reserved/read-only. You must write the data in the following format.

   {8'h0, TC3, TC2, TC1, TC0, 20-bit Time Interval} where TCx = 0 or 1.
4. Program MTL_EST_GCL_Control[SRWO] to 1 (to start a Write Op) and program the address and R/W fields appropriately.
5. Poll and check that the hardware clears MTL_EST_GCL_Control[SRWO] to indicate that the previous operation completes before initiating a new read-write operation via the same indirect addressing mode.
6. Repeat steps 3, 4, 5 until the programming of the GCL completes.
7. Program the BTR, CTR, TER and LLR registers, using the same indirect addressing method as described above. Write 1 to MTL_EST_GCL_Control[GCRR] appropriately. MTL_EST_GCL_Control[GCRR] interprets the address field as belonging to these registers (instead of the GCL).

After the GCL and the related registers are programmed, program MTL_EST_GCL_Control to allow hardware to own and process the GCL. When the list length (as indicated in LLR) is 1, the associated time interval must be smaller than the cycle time register value. Otherwise, an error is reported (as described in the Error handling section) as a single set of gate controls add no value in the TSN context.

**NOTE**

The time unit in all the GCL related registers is seconds and nanoseconds. In cases where internally generated PTP system time is used, you must program the nanoseconds field to use the Digital Rollover mode. (The value of MAC_Timestamp_Control[TSCTRLSSR]must be 1)

### 72.16.11.2 Programming the EST registers

After the steps mentioned in "Programming the GCL and GCL Linked Registers" completes, program MTL_EST_Control

1. Write 1 to MTL_EST_Control[CTOV] and MTL_EST_Control[TILS]. Also, write 1 to MTL_EST_Control[EEST] and MTL_EST_Control[SWOL].

   **NOTE**

   The CTOV recommendations for GMII for SPRAM configurations are:

   - 96 * Tx clock period, for 32 and 64 bit data width configurations.
   - 128 * Tx clock period for 128 bit data width configurations.

   The CTOV recommendations for GMII for non-SPRAM configurations are:

   - 30 * Tx clock period, when SA/VLAN insertion is enabled.
   - 22 * Tx clock period, when SA/VLAN insertion is not enabled.

2. Enables the hardware to own and process the new GCL and make a switch to the new GCL at the BTR value. The hardware provides an interrupt (if enabled) when the switch to the new list happens.
3. Performs an appropriate action to address any other interrupts (explained in the Interrupts section) received during the hardware execution of the GCL.

Write 1 to MTL_EST_Control[SSWL] to handoff to hardware. Software is not allowed to write to the GCL and GCL linked registers when MTL_EST_Control[SSWL] = 1, because the hardware might be using the new GCL.

The hardware resets or clears MTL_EST_Control[SSWL] when it successfully switches to the new list. The hardware also flips MTL_EST_Status[SWOL] to indicate the new GCL that the software owns.

Program the GCL that software owns (MTL_EST_Status[SWOL] indicates) as described in "Programming the GCL and GCL Linked Registers" and then program MTL_EST_Control as described above, to install a new GCL. Ensure that the new BTR is set to an appropriate value to avoid BTR Error that may require software intervention in some cases.

The packet length (frame size) information must be available at all times, to avoid transmission overruns. Therefore, in the DMA configurations, program the packet length in the first descriptor of every transmit frame. Similarly, in the MTL configuration, provide the packet length in the control word.

**NOTE**

The packet length provided in the transmit descriptor must account for the SA and VLAN insertion, to avoid transmission overruns, if applicable, that is, packet length must represent the actual packet length on the Ethernet line.

## 72.16.12 Programming the launch time in time-based scheduling

Configure the launch time in the enhanced normal transmit descriptors in DMA configurations and is driven as a control word in MTL configurations as follows:

The OSTC and launch time features are mutually exclusive and must not be used together. In case of a conflict (if OSTC = LTV = 1 in MTL configuration), give priority to OSTC and ignore the launch time.

In DMA configuration, if a context descriptor is received with a valid OSTC values immediately before receiving a first normal descriptor with LTV = 1, then the LTV is ignored.

## 72.16.13 Programming guidelines for media clock generation and recovery

### 72.16.13.1 Programming guidelines for media clock generation

These are the guidelines for the media clock generation:

1. Program the appropriate presentation time control (supported generation modes "1001-1011" ) to PPSCTRL_PPSCMD ( for 0th instance)/PPSCMD#i (for 1,2,3 instances) of MAC_PPS_Control, to define the PPS instance in Media clock generation mode.

2. Based on the selected PPS instance, application must drive the appropriate trigger signal to the corresponding MCG_PST_TRIG_I[#i].

3. Based on the programmed mode, DUT captures the timestamp and programs it in the MAC_PPS(#i)_Target_Time_Seconds register. Then, mcgr_dma_req_o[#i] is set.

4. For every request which DUT generates, the module reads the corresponding MAC_PPS(#i)_Target_Time_Seconds register and asserts the corresponding mcgr_d-ma_ack_i[#i] to acknowledge the read request.

### 72.16.13.2 Programming guidelines for media clock recovery

These are the guidelines for the media clock recovery:

- Writing 1 to MAC_Timestamp_Control[PTGE] enables the CPT counter. In addition to configuring the initialization values for the system time, update MAC_Presn_Time_Updt with the equivalent presentation time initialization value, then MAC_Timestamp_Control[TSINIT] = 1.
- Use the increment values for the system time and also for the CPT because the increment value is in sub-seconds and sub-nanoseconds.
- Update MAC_Presn_Time_Updt with the equivalent presentation time update value (32 bit ns), when an update is required, and also configure the updated values for the system time. Finally, MAC_Timestamp_Control[TSUPDT] must be 1 for the update.
- Program the MCGREN#i field of MAC_PPS_Control to enable the PPS instance to operate in MCGR mode.
- Program the appropriate presentation time control (supported recovery modes "0001 - 0011") to PPSCTRL_PPSCMD (for 0th instance) or PPSCMD#i (for 1,2,3 instances) of MAC_PPS_Control register, to set the PPS instance in Media clock recovery mode.
- After the PPS instance is set in Recovery mode, DUT sets the corresponding mcgr_dma_req_o[#i]. For every request which DUT generates, program the target presentation time into the MAC_PPS(#i)_Target_Time_Seconds register and assert the corresponding mcgr_dma_ack_i[#i] to acknowledge the request. Acknowledgment can be given before/after programming the Target presentation time.
- Observe the recovered media clock on the corresponding PTP_PPS_O[#i].

## 72.16.14 Programming guidelines for ECC protection for memories

These are the guidelines for the ECC protection for memories:

- Write 1 to the appropriate field of MTL_ECC_Control to enable the ECC features for the respective memory.
- Generate a correctable interrupt (sbd_sfty_ce_intr_o) or an uncorrectable interrupt (sbd_sfty_ue_intr_o) to indicate the application, if any correctable, uncorrectable, or address errors are detected. DMA_Interrupt_Status or MTL_ECC_Interrupt_Status indicates an appropriate status.
- Provide debug mode for each memory to specify errors.

**NOTE**

Enable the ECC feature before the traffic is online (or after the reset), otherwise the false interrupts may trigger.

### 72.16.14.1 Programming guidelines for ECC hardware error injection (Debug mode)

Follow these steps for ECC hardware error injection:

- Write 1 to the appropriate field in MTL_DBG_CTL to enable the ECC error injection feature for MTL TX/RX and DMA TSO.
- Write 1 to the appropriate field in MTL_EST_GCL_Control to enable the ECC error injection feature for the EST memory.
- Write 1 to the appropriate field in MTL_RXP_Indirect_Acc_Control_Status to enable the ECC error injection for the receive parser memory. To access the receive parser memory in Debug mode, write 0 to MTL_RXP_Indirect_Acc_Control_Status[FRPE] to disable the receive parser feature.
- Ensure that all the CSR writes corresponding to one memory write must have the same value for the error injection control word, where multiple CSR writes are required for writing single data word into the memory.

### 72.16.15 Programming guidelines for on-chip datapath parity protection

Follow these steps for on-chip datapath parity protection:

- Writing 1 to MTL_DPP_Control[EDPP] enables the parity generation and parity detection mismatch on datapath.
- Generates an uncorrectable interrupt (sbd_sfty_ue_intr_o) to indicate the application if any parity mismatch is detected. DMA_Safety_Interrupt_Status or MTL_Safety_Interrupt_Status and MAC_DPP_FSM_Interrupt_Status indicates the appropriate status.
- Supports Debug mode for each parity generator to insert an error.

**NOTE**

Enable the datapath parity protection before the receive or transit traffic starts, to avoid the false safety interrupts.

### 72.16.16 Programming guidelines for FSM parity and timeout

Follow these steps to configure the FSM parity and timeout:

- Write 1 to MAC_FSM_Control[PRTYEN] and MAC_FSM_Control[TMOUTEN] to enable the FSM parity and timeout feature respectively.
- Force the FSMs to enter into a state with even number of 1s and wait for the safety uncorrectable interrupt to define with MAC_DPP_FSM_Interrupt_Status[FSMPES] for a parity error detection.
- Write 1 to the parity error injection using the [23:16] bits of MAC_FSM_Control to select the appropriate clock domain, for Error injection mode in FSM parity. Wait for the safety uncorrectable interrupt to be defined with MAC_DPP_FSM_Interrupt_Status[FSMPES].
- Configure the large and normal mode values which MAC_FSM_ACT_Timer[LTMRMD] and MAC_FSM_ACT_Timer[NTMRMD] indicates , for FSM timeouts.
- Use the [31:24] bits of MAC_FSM_Control to select the large or normal mode tic generation per clock domain.
- Write 1 to MAC_FSM_ACT_Timer[TMR] with the appropriate number of CSR clock cycles to generate 1us tic.
- Force the FSM in a particular clock domain to remain in an active state for timeout duration (for example, T-2T). You can check the safety interrupt and the status field for the corresponding clock domain that sets in the [15:8] bits of the MAC_DPP_FSM_Interrupt_Status .
- Select the appropriate clock domain for which timeout error injection must be set using the [15:8] bits of MAC_FSM_Control, for Error injection mode in FSM timeout. Wait for the safety uncorrectable interrupt to define with the corresponding error status in the MAC_DPP_FSM_Interrupt_Status .
- Use only normal mode tic generation, for application or CSR interface timeout. On the basis of the time-outs of configured interface, appropriate fields (MSTTES, SLVTES) of MAC_DPP_FSM_Interrupt_Status is defined with the safety interrupt.

## 72.17 EMAC register descriptions

This section and all its sub-sections are Synopsys Proprietary. Used with permission.

**NOTE**

This chip has 16 KB allocated per Ethernet controller for configuration space. EMAC uses only 8 KB of this memory and the remaining 8 KB is reserved for future use. Writing to the reserved memory may lead to unintended behavior.

### 72.17.1 EMAC memory map

EMAC base address: 4048_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | MAC Configuration (MAC_Configuration) | 32 | RW | 0000_8000h |
| 4h | MAC Extended Configuration (MAC_Ext_Configuration) | 32 | RW | 0000_0000h |
| 8h | MAC Packet Filter (MAC_Packet_Filter) | 32 | RW | 0000_0000h |
| Ch | MAC Watchdog Timeout (MAC_Watchdog_Timeout) | 32 | RW | 0000_0000h |
| 10h | MAC Hash Table First 32 Bits (MAC_Hash_Table_Reg0) | 32 | RW | 0000_0000h |
| 14h | MAC Hash Table Second 32 Bits (MAC_Hash_Table_Reg1) | 32 | RW | 0000_0000h |
| 50h | MAC VLAN Tag (MAC_VLAN_Tag) | 32 | RW | 0000_0000h |
| 50h | MAC VLAN Tag Control (MAC_VLAN_Tag_Ctrl) | 32 | RW | 0000_0000h |
| 54h | MAC VLAN Tag Data (MAC_VLAN_Tag_Data) | 32 | RW | 0000_0000h |
| 54h | MAC VLAN Tag Filter 0 (MAC_VLAN_Tag_Filter0) | 32 | RW | 0000_0000h |
| 54h | MAC VLAN Tag Filter 1 (MAC_VLAN_Tag_Filter1) | 32 | RW | 0000_0000h |
| 54h | MAC VLAN Tag Filter 2 (MAC_VLAN_Tag_Filter2) | 32 | RW | 0000_0000h |
| 54h | MAC VLAN Tag Filter 3 (MAC_VLAN_Tag_Filter3) | 32 | RW | 0000_0000h |
| 58h | MAC VLAN Hash Table (MAC_VLAN_Hash_Table) | 32 | RW | 0000_0000h |
| 60h | MAC VLAN Inclusion Or Replacement (MAC_VLAN_Incl) | 32 | RW | 0000_0000h |
| 60h | MAC VLAN Inclusion 0 (MAC_VLAN_Incl0) | 32 | RW | 0000_0000h |
| 60h | MAC VLAN Inclusion 1 (MAC_VLAN_Incl1) | 32 | RW | 0000_0000h |
| 60h | MAC VLAN Inclusion 2 (MAC_VLAN_Incl2) | 32 | RW | 0000_0000h |
| 60h | MAC VLAN Inclusion 3 (MAC_VLAN_Incl3) | 32 | RW | 0000_0000h |
| 60h | MAC VLAN Inclusion 4 (MAC_VLAN_Incl4) | 32 | RW | 0000_0000h |
| 60h | MAC VLAN Inclusion 5 (MAC_VLAN_Incl5) | 32 | RW | 0000_0000h |
| 60h | MAC VLAN Inclusion 6 (MAC_VLAN_Incl6) | 32 | RW | 0000_0000h |
| 60h | MAC VLAN Inclusion 7 (MAC_VLAN_Incl7) | 32 | RW | 0000_0000h |
| 64h | Inner VLAN Tag Inclusion Or Replacement (MAC_Inner_VLAN_Incl) | 32 | RW | 0000_0000h |
| 70h | MAC Q0 Tx Flow Control (MAC_Q0_Tx_Flow_Ctrl) | 32 | RW | 0000_0000h |
| 90h | MAC Receive Flow Control (MAC_Rx_Flow_Ctrl) | 32 | RW | 0000_0000h |
| 94h | MAC RxQ Control 4 (MAC_RxQ_Ctrl4) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| A0h | MAC RxQ Control 0 (MAC_RxQ_Ctrl0) | 32 | RW | 0000_0000h |
| A4h | Receive Queue Control 1 (MAC_RxQ_Ctrl1) | 32 | RW | 0000_0000h |
| A8h | MAC RxQ Control 2 (MAC_RxQ_Ctrl2) | 32 | RW | 0000_0000h |
| B0h | MAC Interrupt Status (MAC_Interrupt_Status) | 32 | R | 0000_0000h |
| B4h | MAC Interrupt Enable (MAC_Interrupt_Enable) | 32 | RW | 0000_0000h |
| B8h | MAC Rx Transmit Status (MAC_Rx_Tx_Status) | 32 | R | 0000_0000h |
| 110h | MAC Version (MAC_Version) | 32 | R | 0000_1051h |
| 114h | MAC Debug (MAC_Debug) | 32 | R | 0000_0000h |
| 11Ch | MAC Hardware Feature 0 (MAC_HW_Feature0) | 32 | R | 0E09_5135h |
| 120h | MAC Hardware Feature 1 (MAC_HW_Feature1) | 32 | R | 2118_29A6h |
| 124h | MAC Hardware Feature 2 (MAC_HW_Feature2) | 32 | R | 0404_1041h |
| 128h | MAC Hardware Feature 3 (MAC_HW_Feature3) | 32 | R | 2C37_0E31h |
| 140h | MAC DPP FSM Interrupt Status (MAC_DPP_FSM_Interrupt_Status) | 32 | R | 0000_0000h |
| 148h | MAC FSM Control (MAC_FSM_Control) | 32 | RW | 0000_0000h |
| 14Ch | MAC FSM ACT Timer (MAC_FSM_ACT_Timer) | 32 | RW | 0000_0000h |
| 150h | SCS_REG 1 (SCS_REG1) | 32 | RW | 0000_0000h |
| 200h | MAC MDIO Address (MAC_MDIO_Address) | 32 | RW | 0000_0000h |
| 204h | MAC MDIO Data (MAC_MDIO_Data) | 32 | RW | 0000_0000h |
| 230h | MAC CSR Software Control (MAC_CSR_SW_Ctrl) | 32 | RW | 0000_0000h |
| 234h | MAC FPE Control STS (MAC_FPE_CTRL_STS) | 32 | RW | 0000_0000h |
| 240h | MAC Presentation Time (MAC_Presn_Time_ns) | 32 | R | 0000_0000h |
| 244h | MAC Presentation Time Update (MAC_Presn_Time_Updt) | 32 | RW | 0000_0000h |
| 300h | MAC Address 0 High (MAC_Address0_High) | 32 | RW | 8000_FFFFh |
| 304h | MAC Address 0 Low (MAC_Address0_Low) | 32 | RW | FFFF_FFFFh |
| 308h | MAC Address 1 High (MAC_Address1_High) | 32 | RW | 0000_FFFFh |
| 30Ch | MAC Address 1 Low (MAC_Address1_Low) | 32 | RW | FFFF_FFFFh |
| 310h | MAC Address 2 High (MAC_Address2_High) | 32 | RW | 0000_FFFFh |
| 314h | MAC Address 2 Low (MAC_Address2_Low) | 32 | RW | FFFF_FFFFh |
| 700h | MMC Control (MMC_Control) | 32 | RW | 0000_0000h |
| 704h | MMC Receive Interrupt (MMC_Rx_Interrupt) | 32 | R | 0000_0000h |
| 708h | MMC Transmit Interrupt (MMC_Tx_Interrupt) | 32 | R | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 70Ch | MMC Receive Interrupt Mask (MMC_Rx_Interrupt_Mask) | 32 | RW | 0000_0000h |
| 710h | MMC Transmit Interrupt Mask (MMC_Tx_Interrupt_Mask) | 32 | RW | 0000_0000h |
| 714h | Transmit Octet Count Good Bad (Tx_Octet_Count_Good_Bad) | 32 | R | 0000_0000h |
| 718h | Transmit Packet Count Good Bad (Tx_Packet_Count_Good_Bad) | 32 | R | 0000_0000h |
| 71Ch | Transmit Broadcast Packets Good (Tx_Broadcast_Packets_Good) | 32 | R | 0000_0000h |
| 720h | Transmit Multicast Packets Good (Tx_Multicast_Packets_Good) | 32 | R | 0000_0000h |
| 724h | Transmit 64-Octet Packets Good Bad (Tx_64Octets_Packets_Good_Bad) | 32 | R | 0000_0000h |
| 728h | Transmit 65 To 127 Octet Packets Good Bad (Tx_65To127Octets_Packets_Good_Bad) | 32 | R | 0000_0000h |
| 72Ch | Transmit 128 To 255 Octet Packets Good Bad (Tx_128To255Octets_Packets_Good_Bad) | 32 | R | 0000_0000h |
| 730h | Transmit 256 To 511 Octet Packets Good Bad (Tx_256To511Octets_Packets_Good_Bad) | 32 | R | 0000_0000h |
| 734h | Transmit 512 To 1023 Octet Packets Good Bad (Tx_512To1023Octets_Packets_Good_Bad) | 32 | R | 0000_0000h |
| 738h | Transmit 1024 To Max Octet Packets Good Bad (Tx_1024ToMaxOctets_Packets_Good_Bad) | 32 | R | 0000_0000h |
| 73Ch | Transmit Unicast Packets Good Bad (Tx_Unicast_Packets_Good_Bad) | 32 | R | 0000_0000h |
| 740h | Transmit Multicast Packets Good Bad (Tx_Multicast_Packets_Good_Bad) | 32 | R | 0000_0000h |
| 744h | Transmit Broadcast Packets Good Bad (Tx_Broadcast_Packets_Good_Bad) | 32 | R | 0000_0000h |
| 748h | Transmit Underflow Error Packets (Tx_Underflow_Error_Packets) | 32 | R | 0000_0000h |
| 74Ch | Transmit Single Collision Good Packets (Tx_Single_Collision_Good_Packets) | 32 | R | 0000_0000h |
| 750h | Transmit Multiple Collision Good Packets (Tx_Multiple_Collision_Good_Packets) | 32 | R | 0000_0000h |
| 754h | Transmit Deferred Packets (Tx_Deferred_Packets) | 32 | R | 0000_0000h |
| 758h | Transmit Late Collision Packets (Tx_Late_Collision_Packets) | 32 | R | 0000_0000h |
| 75Ch | Transmit Excessive Collision Packets (Tx_Excessive_Collision_Packets) | 32 | R | 0000_0000h |
| 760h | Transmit Carrier Error Packets (Tx_Carrier_Error_Packets) | 32 | R | 0000_0000h |
| 764h | Transmit Octet Count Good (Tx_Octet_Count_Good) | 32 | R | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 768h | Transmit Packet Count Good (Tx_Packet_Count_Good) | 32 | R | 0000_0000h |
| 76Ch | Transmit Excessive Deferral Error (Tx_Excessive_Deferral_Error) | 32 | R | 0000_0000h |
| 770h | Transmit Pause Packets (Tx_Pause_Packets) | 32 | R | 0000_0000h |
| 774h | Transmit VLAN Packets Good (Tx_VLAN_Packets_Good) | 32 | R | 0000_0000h |
| 778h | Transmit O Size Packets Good (Tx_OSize_Packets_Good) | 32 | R | 0000_0000h |
| 780h | Receive Packets Count Good Bad (Rx_Packets_Count_Good_Bad) | 32 | R | 0000_0000h |
| 784h | Receive Octet Count Good Bad (Rx_Octet_Count_Good_Bad) | 32 | R | 0000_0000h |
| 788h | Receive Octet Count Good (Rx_Octet_Count_Good) | 32 | R | 0000_0000h |
| 78Ch | Receive Broadcast Packets Good (Rx_Broadcast_Packets_Good) | 32 | R | 0000_0000h |
| 790h | Receive Multicast Packets Good (Rx_Multicast_Packets_Good) | 32 | R | 0000_0000h |
| 794h | Receive CRC Error Packets (Rx_CRC_Error_Packets) | 32 | R | 0000_0000h |
| 798h | Receive Alignment Error Packets (Rx_Alignment_Error_Packets) | 32 | R | 0000_0000h |
| 79Ch | Receive Runt Error Packets (Rx_Runt_Error_Packets) | 32 | R | 0000_0000h |
| 7A0h | Receive Jabber Error Packets (Rx_Jabber_Error_Packets) | 32 | R | 0000_0000h |
| 7A4h | Receive Undersize Packets Good (Rx_Undersize_Packets_Good) | 32 | R | 0000_0000h |
| 7A8h | Receive Oversize Packets Good (Rx_Oversize_Packets_Good) | 32 | R | 0000_0000h |
| 7ACh | Receive 64 Octets Packets Good Bad (Rx_64Octets_Packets_Good_Bad) | 32 | R | 0000_0000h |
| 7B0h | Receive 65-127 Octets Packets Good Bad (Rx_65To127Octets_Packets_Good_Bad) | 32 | R | 0000_0000h |
| 7B4h | Receive 128-255 Octets Packets Good Bad (Rx_128To255Octets_Packets_Good_Bad) | 32 | R | 0000_0000h |
| 7B8h | Receive 256-511 Octets Packets Good Bad (Rx_256To511Octets_Packets_Good_Bad) | 32 | R | 0000_0000h |
| 7BCh | Receive 512-1023 Octets Packets Good Bad (Rx_512To1023Octets_Packets_Good_Bad) | 32 | R | 0000_0000h |
| 7C0h | Receive 1024 To Max Octets Good Bad (Rx_1024ToMaxOctets_Packets_Good_Bad) | 32 | R | 0000_0000h |
| 7C4h | Receive Unicast Packets Good (Rx_Unicast_Packets_Good) | 32 | R | 0000_0000h |
| 7C8h | Receive Length Error Packets (Rx_Length_Error_Packets) | 32 | R | 0000_0000h |
| 7CCh | Receive Out of Range Type Packet (Rx_Out_Of_Range_Type_Packets) | 32 | R | 0000_0000h |
| 7D0h | Receive Pause Packets (Rx_Pause_Packets) | 32 | R | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 7D4h | Receive FIFO Overflow Packets (Rx_FIFO_Overflow_Packets) | 32 | R | 0000_0000h |
| 7D8h | Receive VLAN Packets Good Bad (Rx_VLAN_Packets_Good_Bad) | 32 | R | 0000_0000h |
| 7DCh | Receive Watchdog Error Packets (Rx_Watchdog_Error_Packets) | 32 | R | 0000_0000h |
| 7E0h | Receive Error Packets (Rx_Receive_Error_Packets) | 32 | R | 0000_0000h |
| 7E4h | Receive Control Packets Good (Rx_Control_Packets_Good) | 32 | R | 0000_0000h |
| 8A0h | MMC Transmit FPE Fragment Counter Interrupt Status (MMC_FPE_Tx_Interrupt) | 32 | R | 0000_0000h |
| 8A4h | MMC FPE Transmit Interrupt Mask (MMC_FPE_Tx_Interrupt_Mask) | 32 | RW | 0000_0000h |
| 8A8h | Transmit FPE Fragment Counter (MMC_Tx_FPE_Fragment_Cntr) | 32 | R | 0000_0000h |
| 8ACh | Transmit Hold Request Counter (MMC_Tx_Hold_Req_Cntr) | 32 | R | 0000_0000h |
| 8C0h | MMC Receive Packet Assembly Error Counter Interrupt Status (MMC_FPE_Rx_Interrupt) | 32 | R | 0000_0000h |
| 8C4h | MMC FPE Receive Interrupt Mask (MMC_FPE_Rx_Interrupt_Mask) | 32 | RW | 0000_0000h |
| 8C8h | MMC Receive Packet Assembly Error Counter (MMC_Rx_Packet_Assembly_Err_Cntr) | 32 | R | 0000_0000h |
| 8CCh | MMC Receive Packet SMD Error Counter (MMC_Rx_Packet_SMD_Err_Cntr) | 32 | R | 0000_0000h |
| 8D0h | MMC Receive Packet Assembly OK Counter (MMC_Rx_Packet_Assembly_OK_Cntr) | 32 | R | 0000_0000h |
| 8D4h | MMC Receive FPE Fragment Counter (MMC_Rx_FPE_Fragment_Cntr) | 32 | R | 0000_0000h |
| 900h | MAC Layer 3 Layer 4 Control 0 (MAC_L3_L4_Control0) | 32 | RW | 0000_0000h |
| 904h | MAC Layer 4 Address 0 (MAC_Layer4_Address0) | 32 | RW | 0000_0000h |
| 910h | MAC Layer 3 Address 0 Reg 0 (MAC_Layer3_Addr0_Reg0) | 32 | RW | 0000_0000h |
| 914h | MAC Layer 3 Address 1 Reg 0 (MAC_Layer3_Addr1_Reg0) | 32 | RW | 0000_0000h |
| 918h | MAC Layer 3 Address 2 Reg 0 (MAC_Layer3_Addr2_Reg0) | 32 | RW | 0000_0000h |
| 91Ch | MAC Layer 3 Address 3 Reg 0 (MAC_Layer3_Addr3_Reg0) | 32 | RW | 0000_0000h |
| 930h | MAC L3 L4 Control 1 (MAC_L3_L4_Control1) | 32 | RW | 0000_0000h |
| 934h | MAC Layer 4 Address 1 (MAC_Layer4_Address1) | 32 | RW | 0000_0000h |
| 940h | MAC Layer 3 Address 0 Reg 1 (MAC_Layer3_Addr0_Reg1) | 32 | RW | 0000_0000h |
| 944h | MAC Layer 3 Address 1 Reg 1 (MAC_Layer3_Addr1_Reg1) | 32 | RW | 0000_0000h |
| 948h | MAC Layer 3 Address 2 Reg 1 (MAC_Layer3_Addr2_Reg1) | 32 | RW | 0000_0000h |
| 94Ch | MAC Layer 3 Address 3 Reg 1 (MAC_Layer3_Addr3_Reg1) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 960h | MAC L3 L4 Control 2 (MAC_L3_L4_Control2) | 32 | RW | 0000_0000h |
| 964h | MAC Layer 4 Address 2 (MAC_Layer4_Address2) | 32 | RW | 0000_0000h |
| 970h | MAC Layer 3 Address 0 Reg 2 (MAC_Layer3_Addr0_Reg2) | 32 | RW | 0000_0000h |
| 974h | MAC Layer 3 Address 1 Reg 2 (MAC_Layer3_Addr1_Reg2) | 32 | RW | 0000_0000h |
| 978h | MAC Layer 3 Address 2 Reg 2 (MAC_Layer3_Addr2_Reg2) | 32 | RW | 0000_0000h |
| 97Ch | MAC Layer 3 Address 3 Reg 2 (MAC_Layer3_Addr3_Reg2) | 32 | RW | 0000_0000h |
| 990h | MAC L3 L4 Control 3 (MAC_L3_L4_Control3) | 32 | RW | 0000_0000h |
| 994h | MAC Layer 4 Address 3 (MAC_Layer4_Address3) | 32 | RW | 0000_0000h |
| 9A0h | MAC Layer 3 Address 0 Reg 3 (MAC_Layer3_Addr0_Reg3) | 32 | RW | 0000_0000h |
| 9A4h | MAC Layer 3 Address 1 Reg 3 (MAC_Layer3_Addr1_Reg3) | 32 | RW | 0000_0000h |
| 9A8h | MAC Layer 3 Address 2 Reg 3 (MAC_Layer3_Addr2_Reg3) | 32 | RW | 0000_0000h |
| 9ACh | MAC Layer 3 Address 3 Reg 3 (MAC_Layer3_Addr3_Reg3) | 32 | RW | 0000_0000h |
| B00h | MAC Timestamp Control (MAC_Timestamp_Control) | 32 | RW | 0000_2000h |
| B04h | MAC Sub Second Increment (MAC_Sub_Second_Increment) | 32 | RW | 0000_0000h |
| B08h | MAC System Time In Seconds (MAC_System_Time_Seconds) | 32 | R | 0000_0000h |
| B0Ch | MAC System Time In Nanoseconds (MAC_System_Time_Nanoseconds) | 32 | R | 0000_0000h |
| B10h | MAC System Time Seconds Update (MAC_System_Time_Seconds_Update) | 32 | RW | 0000_0000h |
| B14h | MAC System Time Nanoseconds Update (MAC_System_Time_Nanoseconds_Update) | 32 | RW | 0000_0000h |
| B18h | MAC Timestamp Addend (MAC_Timestamp_Addend) | 32 | RW | 0000_0000h |
| B1Ch | MAC System Time Higher Word In Seconds (MAC_System_Time_Higher_Word_Seconds) | 32 | RW | 0000_0000h |
| B20h | MAC Timestamp Status (MAC_Timestamp_Status) | 32 | R | 0000_0000h |
| B30h | MAC Transmit Timestamp Status In Nanoseconds (MAC_Tx_Timestamp_Status_Nanoseconds) | 32 | R | 0000_0000h |
| B34h | MAC Transmit Timestamp Status In Seconds (MAC_Tx_Timestamp_Status_Seconds) | 32 | R | 0000_0000h |
| B50h | MAC Timestamp Ingress Asymmetry Correction (MAC_Timestamp_Ingress_Asym_Corr) | 32 | RW | 0000_0000h |
| B54h | MAC Timestamp Egress Asymmetry Correction (MAC_Timestamp_Egress_Asym_Corr) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| B58h | MAC Timestamp Ingress Correction In Nanoseconds (MAC_Timestamp_Ingress_Corr_Nanosecond) | 32 | RW | 0000_0000h |
| B5Ch | MAC Timestamp Egress Correction In Nanoseconds (MAC_Timestamp_Egress_Corr_Nanosecond) | 32 | RW | 0000_0000h |
| B60h | MAC Timestamp Ingress Correction In Subnanoseconds (MAC_Timestamp_Ingress_Corr_Subnanosec) | 32 | RW | 0000_0000h |
| B64h | MAC Timestamp Engress Correction In Subnanoseconds (MAC_Timestamp_Egress_Corr_Subnanosec) | 32 | RW | 0000_0000h |
| B68h | MAC Timestamp Ingress Latency (MAC_Timestamp_Ingress_Latency) | 32 | R | 0000_0000h |
| B6Ch | MAC Timestamp Egress Latency (MAC_Timestamp_Egress_Latency) | 32 | R | 0000_0000h |
| B70h | MAC PPS Control (MAC_PPS_Control) | 32 | RW | 0000_0000h |
| B80h | MAC PPS0 Target Time In Seconds (MAC_PPS0_Target_Time_Seconds) | 32 | RW | 0000_0000h |
| B84h | MAC PPS0 Target Time In Nanoseconds (MAC_PPS0_Target_Time_Nanoseconds) | 32 | RW | 0000_0000h |
| B88h | MAC PPS0 Interval (MAC_PPS0_Interval) | 32 | RW | 0000_0000h |
| B8Ch | MAC PPS0 Width (MAC_PPS0_Width) | 32 | RW | 0000_0000h |
| B90h | MAC PPS1 Target Time In Seconds (MAC_PPS1_Target_Time_Seconds) | 32 | RW | 0000_0000h |
| B94h | MAC PPS1 Target Time In Nanoseconds (MAC_PPS1_Target_Time_Nanoseconds) | 32 | RW | 0000_0000h |
| B98h | MAC PPS1 Interval (MAC_PPS1_Interval) | 32 | RW | 0000_0000h |
| B9Ch | MAC PPS1 Width (MAC_PPS1_Width) | 32 | RW | 0000_0000h |
| BA0h | MAC PPS2 Target Time In Seconds (MAC_PPS2_Target_Time_Seconds) | 32 | RW | 0000_0000h |
| BA4h | MAC PPS2 Target Time In Nanoseconds (MAC_PPS2_Target_Time_Nanoseconds) | 32 | RW | 0000_0000h |
| BA8h | MAC PPS2 Interval (MAC_PPS2_Interval) | 32 | RW | 0000_0000h |
| BACh | MAC PPS2 Width (MAC_PPS2_Width) | 32 | RW | 0000_0000h |
| BB0h | MAC PPS3 Target Time In Seconds (MAC_PPS3_Target_Time_Seconds) | 32 | RW | 0000_0000h |
| BB4h | MAC PPS3 Target Time In Nanoseconds (MAC_PPS3_Target_Time_Nanoseconds) | 32 | RW | 0000_0000h |
| BB8h | MAC PPS3 Interval (MAC_PPS3_Interval) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| BBCh | MAC PPS3 Width (MAC_PPS3_Width) | 32 | RW | 0000_0000h |
| C00h | MTL Operation Mode (MTL_Operation_Mode) | 32 | RW | 0000_0000h |
| C08h | MTL Debug Control (MTL_DBG_CTL) | 32 | RW | 0000_0000h |
| C0Ch | MTL Debug Status (MTL_DBG_STS) | 32 | RW | 0000_0018h |
| C10h | MTL FIFO Debug Data (MTL_FIFO_Debug_Data) | 32 | RW | 0000_0000h |
| C20h | MTL Interrupt Status (MTL_Interrupt_Status) | 32 | R | 0000_0000h |
| C30h | MTL Receive Queue DMA Map 0 (MTL_RxQ_DMA_Map0) | 32 | RW | 0000_0000h |
| C40h | MTL TBS Control (MTL_TBS_CTRL) | 32 | RW | 0000_0000h |
| C50h | MTL EST Control (MTL_EST_Control) | 32 | RW | 0000_0000h |
| C58h | MTL EST Status (MTL_EST_Status) | 32 | RW | 0000_0000h |
| C60h | MTL EST Scheduling Error (MTL_EST_Sch_Error) | 32 | RW | 0000_0000h |
| C64h | MTL EST Frame Size Error (MTL_EST_Frm_Size_Error) | 32 | RW | 0000_0000h |
| C68h | MTL EST Frame Size Capture (MTL_EST_Frm_Size_Capture) | 32 | R | 0000_0000h |
| C70h | MTL EST Interrupt Enable (MTL_EST_Intr_Enable) | 32 | RW | 0000_0000h |
| C80h | MTL EST GCL Control (MTL_EST_GCL_Control) | 32 | RW | 0000_0000h |
| C84h | MTL EST GCL Data (MTL_EST_GCL_Data) | 32 | RW | 0000_0000h |
| C90h | MTL FPE Control Status (MTL_FPE_CTRL_STS) | 32 | RW | 0000_0000h |
| C94h | MTL FPE Advance (MTL_FPE_Advance) | 32 | RW | 0000_0000h |
| CA0h | MTL Rx Parser Control Status (MTL_RXP_Control_Status) | 32 | RW | 803F_003Fh |
| CA4h | MTL Rx Parser Interrupt Control Status (MTL_RXP_Interrupt_Control_Status) | 32 | RW | 0000_0000h |
| CA8h | MTL Rx Parser Drop Count (MTL_RXP_Drop_Cnt) | 32 | R | 0000_0000h |
| CACh | MTL Rx Parser Error Count (MTL_RXP_Error_Cnt) | 32 | R | 0000_0000h |
| CB0h | MTL Rx Parser Indirect Access Control Status (MTL_RXP_Indirect_Acc_Control_Status) | 32 | RW | 0000_0000h |
| CB4h | MTL Rx Parser Indirect Access Data (MTL_RXP_Indirect_Acc_Data) | 32 | R | 0000_0000h |
| CC0h | MTL ECC Control (MTL_ECC_Control) | 32 | RW | 0000_000Fh |
| CC4h | MTL Safety Interrupt Status (MTL_Safety_Interrupt_Status) | 32 | R | 0000_0000h |
| CC8h | MTL ECC Interrupt Enable (MTL_ECC_Interrupt_Enable) | 32 | RW | 0000_1111h |
| CCCh | MTL ECC Interrupt Status (MTL_ECC_Interrupt_Status) | 32 | RW | 0000_0000h |
| CD0h | MTL ECC Error Status (MTL_ECC_Err_Sts_Rctl) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| CD4h | MTL ECC Error Address Status (MTL_ECC_Err_Addr_Status) | 32 | R | 0000_0000h |
| CD8h | MTL ECC Error Control Status (MTL_ECC_Err_Cntr_Status) | 32 | R | 0000_0000h |
| CE0h | MTL DPP Control (MTL_DPP_Control) | 32 | RW | 0000_0003h |
| D00h | MTL Tx Queue 0 Operation Mode (MTL_TxQ0_Operation_Mode) | 32 | RW | 0000_0000h |
| D04h | MTL Tx Queue 0 Underflow (MTL_TxQ0_Underflow) | 32 | R | 0000_0000h |
| D08h | MTL Tx Queue 0 Debug (MTL_TxQ0_Debug) | 32 | R | 0000_0000h |
| D14h | MTL Tx Queue 0 ETS Status (MTL_TxQ0_ETS_Status) | 32 | R | 0000_0000h |
| D18h | MTL Tx Queue Quantum Weight (MTL_TxQ0_Quantum_Weight) | 32 | RW | 0000_0000h |
| D2Ch | MTL Queue 0 Interrupt Control Status (MTL_Q0_Interrupt_Control_Status) | 32 | RW | 0000_0000h |
| D30h | MTL Rx Queue 0 Operation Mode (MTL_RxQ0_Operation_Mode) | 32 | RW | 0000_0000h |
| D34h | MTL Rx Queue Missed Packet Overflow Count (MTL_RxQ0_Missed_Packet_Overflow_Cnt) | 32 | R | 0000_0000h |
| D38h | MTL Rx Queue 0 Debug (MTL_RxQ0_Debug) | 32 | R | 0000_0000h |
| D3Ch | MTL Rx Queue 0 Control 0 (MTL_RxQ0_Control) | 32 | RW | 0000_0000h |
| D40h | MTL Tx Queue 1 Operation Mode (MTL_TxQ1_Operation_Mode) | 32 | RW | 0000_0000h |
| D44h | MTL Tx Queue 1 Underflow (MTL_TxQ1_Underflow) | 32 | R | 0000_0000h |
| D48h | MTL Tx Queue 1 Debug (MTL_TxQ1_Debug) | 32 | R | 0000_0000h |
| D50h | MTL Tx Queue 1 ETS Control (MTL_TxQ1_ETS_Control) | 32 | RW | 0000_0000h |
| D54h | MTL Tx Queue 1 ETS Status (MTL_TxQ1_ETS_Status) | 32 | R | 0000_0000h |
| D58h | MTL Tx Queue 1 Quantum Weight (MTL_TxQ1_Quantum_Weight) | 32 | RW | 0000_0000h |
| D5Ch | MTL Tx Queue 1 Sendslope Credit (MTL_TxQ1_SendSlopeCredit) | 32 | RW | 0000_0000h |
| D60h | MTL Tx Queue 1 HiCredit (MTL_TxQ1_HiCredit) | 32 | RW | 0000_0000h |
| D64h | MTL Tx Queue 1 LoCredit (MTL_TxQ1_LoCredit) | 32 | RW | 0000_0000h |
| D6Ch | MTL Queue 1 Interrupt Control Status (MTL_Q1_Interrupt_Control_Status) | 32 | RW | 0000_0000h |
| D70h | MTL Rx Queue 1 Operation Mode (MTL_RxQ1_Operation_Mode) | 32 | RW | 0000_0000h |
| D74h | MTL Rx Queue 1 Missed Packet Overflow Counter (MTL_RxQ1_Missed_Packet_Overflow_Cnt) | 32 | R | 0000_0000h |
| D78h | MTL Rx Queue 1 Debug (MTL_RxQ1_Debug) | 32 | R | 0000_0000h |
| D7Ch | MTL Rx Queue 1 Control (MTL_RxQ1_Control) | 32 | RW | 0000_0000h |
| 1000h | DMA Mode (DMA_Mode) | 32 | RW | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1004h | DMA System Bus Mode (DMA_SysBus_Mode) | 32 | RW | 0000_0000h |
| 1008h | DMA Interrupt Status (DMA_Interrupt_Status) | 32 | R | 0000_0000h |
| 100Ch | DMA Debug Status 0 (DMA_Debug_Status0) | 32 | R | 0000_0000h |
| 1050h | DMA TBS Control (DMA_TBS_CTRL) | 32 | RW | 0000_0000h |
| 1080h | DMA Safety Interrupt Status (DMA_Safety_Interrupt_Status) | 32 | R | 0000_0000h |
| 1100h | DMA Channel 0 Control (DMA_CH0_Control) | 32 | RW | 0000_0000h |
| 1104h | DMA Channel Tx Control (DMA_CH0_Tx_Control) | 32 | RW | 0000_0000h |
| 1108h | DMA Channel Rx Control (DMA_CH0_Rx_Control) | 32 | RW | 0000_0000h |
| 1114h | DMA Channel 0 Tx Descriptor List Address (DMA_CH0_TxDesc_List_Address) | 32 | RW | 0000_0000h |
| 111Ch | DMA Channel 0 Rx Descriptor List Address (DMA_CH0_RxDesc_List_Address) | 32 | RW | 0000_0000h |
| 1120h | DMA Channel 0 Tx Descriptor Tail Pointer (DMA_CH0_TxDesc_Tail_Pointer) | 32 | RW | 0000_0000h |
| 1128h | DMA Channel 0 Rx Descriptor List Pointer (DMA_CH0_RxDesc_Tail_Pointer) | 32 | RW | 0000_0000h |
| 112Ch | DMA Channel 0 Tx Descriptor Ring Length (DMA_CH0_TxDesc_Ring_Length) | 32 | RW | 0000_0000h |
| 1130h | DMA Channel 0 Rx Descriptor Ring Length (DMA_CH0_RxDesc_Ring_Length) | 32 | RW | 0000_0000h |
| 1134h | DMA Channel 0 Interrupt Enable (DMA_CH0_Interrupt_Enable) | 32 | RW | 0000_0000h |
| 1138h | DMA Channel 0 Rx Interrupt Watchdog Timer (DMA_CH0_Rx_Interrupt_Watchdog_Timer) | 32 | RW | 0000_0000h |
| 113Ch | DMA Channel 0 Slot Function Control Status (DMA_CH0_Slot_Function_Control_Status) | 32 | RW | 0000_07C0h |
| 1144h | DMA Channel 0 Current Application Transmit Descriptor (DMA_CH0_Current_App_TxDesc) | 32 | R | 0000_0000h |
| 114Ch | DMA Channel 0 Current Application Receive Descriptor (DMA_CH0_Current_App_RxDesc) | 32 | R | 0000_0000h |
| 1154h | DMA Channel 0 Current Application Transmit Descriptor (DMA_CH0_Current_App_TxBuffer) | 32 | R | 0000_0000h |
| 115Ch | DMA Channel 0 Current Application Receive Buffer (DMA_CH0_Current_App_RxBuffer) | 32 | R | 0000_0000h |
| 1160h | DMA Channel 0 Status (DMA_CH0_Status) | 32 | RW | 0000_0000h |
| 1164h | DMA Channel 0 Miss Frame Counter (DMA_CH0_Miss_Frame_Cnt) | 32 | R | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 1168h | DMA Channel 0 Rx Parser Accept Count (DMA_CH0_RXP_Accept_Cnt) | 32 | R | 0000_0000h |
| 116Ch | DMA Channel 0 Rx ERI Count (DMA_CH0_RX_ERI_Cnt) | 32 | R | 0000_0000h |
| 1180h | DMA Channel 1 Control (DMA_CH1_Control) | 32 | RW | 0000_0000h |
| 1184h | DMA Channel 1 Tx Control (DMA_CH1_Tx_Control) | 32 | RW | 0000_0000h |
| 1188h | DMA Channel 1 Rx Control (DMA_CH1_Rx_Control) | 32 | RW | 0000_0000h |
| 1194h | DMA Channel 1 Tx Descriptor List Address (DMA_CH1_TxDesc_List_Address) | 32 | RW | 0000_0000h |
| 119Ch | DMA Channel 1 Rx Descriptor List Address (DMA_CH1_RxDesc_List_Address) | 32 | RW | 0000_0000h |
| 11A0h | DMA Channel 1 Tx Descriptor Tail Pointer (DMA_CH1_TxDesc_Tail_Pointer) | 32 | RW | 0000_0000h |
| 11A8h | DMA Channel 1 Rx Descriptor Tail Pointer (DMA_CH1_RxDesc_Tail_Pointer) | 32 | RW | 0000_0000h |
| 11ACh | DMA Channel 1 Tx Descriptor Ring Length (DMA_CH1_TxDesc_Ring_Length) | 32 | RW | 0000_0000h |
| 11B0h | DMA Channel 1 Rx Descriptor Ring Length (DMA_CH1_RxDesc_Ring_Length) | 32 | RW | 0000_0000h |
| 11B4h | DMA Channel 1 Interrupt Enable (DMA_CH1_Interrupt_Enable) | 32 | RW | 0000_0000h |
| 11B8h | DMA Channel 1 Rx Interrupt Watchdog Timer (DMA_CH1_Rx_Interrupt_Watchdog_Timer) | 32 | RW | 0000_0000h |
| 11BCh | DMA Channel 1 Slot Function Control Status (DMA_CH1_Slot_Function_Control_Status) | 32 | RW | 0000_07C0h |
| 11C4h | DMA Channel 1 Current Application Transmit Descriptor (DMA_CH1_Current_App_TxDesc) | 32 | R | 0000_0000h |
| 11CCh | DMA Channel 1 Current Application Receive Descriptor (DMA_CH1_Current_App_RxDesc) | 32 | R | 0000_0000h |
| 11D4h | DMA Channel 1 Current Application Transmit Buffer (DMA_CH1_Current_App_TxBuffer) | 32 | R | 0000_0000h |
| 11DCh | DMA Channel 1 Current Application Receive Buffer (DMA_CH1_Current_App_RxBuffer) | 32 | R | 0000_0000h |
| 11E0h | DMA Channel 1 Status (DMA_CH1_Status) | 32 | RW | 0000_0000h |
| 11E4h | DMA Channel 1 Miss Frame Counter (DMA_CH1_Miss_Frame_Cnt) | 32 | R | 0000_0000h |
| 11E8h | DMA Channel 1 Rx Parser Accept Count (DMA_CH1_RXP_Accept_Cnt) | 32 | R | 0000_0000h |
| 11ECh | DMA Channel 1 Rx ERI Count (DMA_CH1_RX_ERI_Cnt) | 32 | R | 0000_0000h |

*Table continues on the next page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|

## 72.17.2 MAC Configuration (MAC_Configuration)

### Offset

| Register | Offset |
|---|---|
| MAC_Configuration | 0h |

### Function

Establishes MAC's operating mode.

**Table 541. Packet length based on CST and ACS bits**

| Received packet length | CST | ACS | FCS stripping done |
|---|---|---|---|
| <1536 | — | 0 | No |
| | — | 1 | Yes (for Ethernet packets) |
| >=1536 | 0 | — | No |
| | 1 | — | Yes (for Type packets) |
| <1536 | — | 0 | No |
| | — | 1 | Yes (for Ethernet packets) |

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | SARC | | | IPC | IPG | | | GPSLCE | S2KP | CST | ACS | WD | Reserved | JD | JE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | PS | FES | DM | LM | ECRSFD | DO | DCRS | DR | Reserved | BL | | DC | PRELEN | | TE | RE |
| W | | | | | | | | | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-28<br>SARC | Source Address Insertion Or Replacement Control<br>Controls the source address insertion or replacement for all the transmitted packets.<br>Bit 30 of this field specifies which MAC Address register (0 or 1) is used for source address insertion or replacement based on the values of bits [29:28]:<br>• 2'b0x: The mti_sa_ctrl_i and ati_sa_ctrl_i input signals control the SA field generation.<br>• 2'b10: If bit 30 = 0, MAC inserts the content of the MAC Address 0 registers in the SA field of all the transmitted packets. If bit 30 = 1 and the Enable MAC Address Register 1 option is selected while configuring the core, MAC inserts the content of the MAC Address 1 registers in the SA field of all the transmitted packets.<br>• 2'b11: If bit 30 = 0, MAC replaces the content of the MAC Address 0 registers in the SA field of all the transmitted packets. If bit 30 = 1 and the MAC Address Register 1 is enabled, MAC replaces the content of the MAC Address 1 registers in the SA field of all the transmitted packets.<br>**NOTE**<br>Changes to this field take effect only on the start of a packet. If you write to the field when a packet is being transmitted, only the subsequent packet can use the updated value. This means that the current packet does not use the updated value.<br>000b - mti_sa_ctrl_i and ati_sa_ctrl_i input signals control the SA field generation<br>010b - Contents of are inserted in the SA field<br>011b - Contents of replace the SA field<br>110b - Contents of are inserted in the SA field<br>111b - Contents of replace the SA field |
| 27<br>IPC | Checksum Offload<br>Indicates the status of IP header or payload checksum checking.<br>If this field is 1, the field enables IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When the field becomes 0, the COE function in the receiver is disabled.<br>The Layer 3 and Layer 4 packet filter and enable split header features automatically select the IPC full checksum offload engine on the receive side. When any of these features are enabled, you must write 1 to this field.<br>0b - Disabled<br>1b - Enabled |
| 26-24<br>IPG | Inter-Packet Gap<br>Controls the minimum IPG between packets during transmission.<br>The range of minimum IPG is valid in Full-Duplex mode, and in this mode, the minimum IPG can be configured only for 64-bit times (IPG = 100). Lower values are not considered. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When a JAM pattern is being transmitted because of backpressure activation, MAC does not consider the minimum IPG.<br>This function (IPG less than 96-bit times) is valid only when MAC_Ext_Configuration[EIPGEN] is 0. If this field is 1, the minimum IPG (greater than 96-bit times) is controlled according to the field's description.<br>000b - 96-bit times IPG<br>001b - 88-bit times IPG<br>010b - 80-bit times IPG<br>011b - 72-bit times IPG<br>100b - 64-bit times IPG<br>101b - 56-bit times IPG<br>110b - 48-bit times IPG<br>111b - 40-bit times IPG |
| 23<br>GPSLCE | Giant Packet Size Limit Control Enable<br>Enables and disables giant packet size limit control.<br>If this field = 1, MAC considers the value of MAC_Ext_Configuration[GPSL] to declare a received packet as a giant packet. This field must be programmed to have a size of more than 1,518 bytes. Otherwise, MAC considers 1,518 bytes as the giant packet limit.<br>When this field becomes 0, MAC considers the received packet as a giant packet if its size is greater than 1,518 bytes (1522 bytes for a tagged packet).<br>The WD, JE, and S2KP fields have a higher precedence over this field, which means MAC considers a received packet as a giant packet when its size is greater than 9,018 bytes (9,022 bytes for a tagged packet) with JE = 1 and the size of the jumbo packet greater than 2,000 bytes and S2KP = 1. If the WD field is 1, the field terminates the received packet if it reaches the watchdog limit. Therefore, the programmed giant packet limit must be less than the watchdog limit to get the giant packet status.<br>0b - Disabled<br>1b - Enabled |
| 22<br>S2KP | IEEE 802.3 Support For 2K Packets<br>Indicates the status of IEEE 802.3 support for 2K packets.<br>If this field is 1, MAC considers all the packets with up to 2,000-byte length as normal. When the JE field is not 1, MAC considers all the received packets of size more than 2K bytes as giant packets.<br>When this field becomes 0 and the JE field is not 1, MAC considers all the received packets of size more than 1,518 bytes (1,522 bytes for tagged) as giant packets. For more information about how the setting of this field and the JE field impacts the giant packet status, see this status based on the S2KP and JE fields.<br>If the JE field is 1, writing 1 to the S2KP field has no effect on the giant packet status.<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 21<br>CST | CRC Stripping For Type Packets<br>Indicates the status of CRC stripping for Type packets.<br>If this field is 1, the last four bytes (FCS) of all the Ether-type packets are stripped and dropped before forwarding the packet to the application.<br>This field is valid only when the receive packet size is more than or equal to 1536 bytes.<br>For information about how the settings of this field and the ACS field impact the packet length, see MAC Configuration (MAC_Configuration).<br>0b - Disabled<br>1b - Enabled |
| 20<br>ACS | Automatic Pad Or CRC Stripping<br>Indicates the status of automatic pad or CRC stripping.<br>If this field is 1, MAC strips the pad or the FCS field on the incoming packets only if the size of the packets is less than 1,536 bytes. All the received packets with a size greater than or equal to 1,536 bytes are passed to the application without stripping the pad or the FCS field.<br>When this field becomes 0, MAC passes all the incoming packets to the application, without any modification.<br>For information about how the settings of this field and the CST field impact the packet length, see MAC Configuration (MAC_Configuration).<br>0b - Disabled<br>1b - Enabled |
| 19<br>WD | Watchdog Disable<br>Indicates the status of watchdog.<br>If this field is 1, MAC disables the watchdog timer on the receiver. MAC can receive packets of up to 16,383 bytes.<br>When this field becomes 0, MAC does not allow more than 2,048 bytes (10,240 if the JE field = 1) of the packet being received. MAC cuts off bytes received after 2,048.<br>0b - Enabled<br>1b - Disabled |
| 18<br>— | Reserved |
| 17<br>JD | Jabber Disable<br>Indicates the status of jabber.<br>If this field is 1, MAC disables the jabber timer on the transmitter. It can transfer packets of up to 16,383 bytes. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | If this field is 0 and the application sends more than 2,048 bytes of data (10,240 if the JE field = 1) during transmission, MAC does not send the rest of the bytes in that packet.<br>0b - Enabled<br>1b - Disabled |
| 16<br>JE | Jumbo Packet Enable<br>Indicates the status of jumbo packets.<br>If this field is 1, MAC allows jumbo packets of 9,018 bytes (9,022 bytes for VLAN-tagged packets) without reporting a giant packet error in the receive packet status.<br>0b - Disabled<br>1b - Enabled |
| 15<br>PS | Port Select<br>Selects the Ethernet line speed.<br>This field, along with the FES field, selects the exact line speed. In the 10/100 Mbit/s-only (always 1) or 1000 Mbit/s-only (always 0) configurations, this field is read-only (RO) with an appropriate value. By default, with 10/100/1000 Mbit/s configurations, this field is read-write (R/W). The mac_speed_o[1] signal reflects the value of this field.<br>0b - For 1000 or 2500 Mbit/s operations<br>1b - For 10 or 100 Mbit/s operations |
| 14<br>FES | Speed<br>Indicates the speed.<br>This field selects the speed mode.<br>The mac_speed_o[0] signal indicates the value of this field.<br>0b - 10 Mbit/s if PS = 1 and 1 Gbps if PS = 0<br>1b - 100 Mbit/s if PS = 1 and 2.5 Gbps if PS = 0 |
| 13<br>DM | Duplex Mode<br>Indicates the mode in which MAC operates.<br>If this field is 1, MAC operates in Full-Duplex mode in which it can transmit and receive simultaneously. The field can only be read, with 1 as its default value, in full-duplex-only configurations.<br>0b - Half-duplex mode<br>1b - Full-duplex mode |
| 12<br>LM | Loopback Mode<br>Indicates the status of Loopback mode<br>If this field is 1, MAC operates in Loopback mode at GMII or MII. This mode requires the (G)MII Rx clock input (CLK_RX_I) signal to function properly because the transmit clock is not internally looped back. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |
| 11<br>ECRSFD | Enable Carrier Sense In Full-Duplex Mode<br>Indicates whether ECRSFD is enabled or disabled.<br>If this field is 1, the MAC transmitter checks the CRS signal before packet transmission in Full-Duplex mode. MAC starts transmitting only when the CRS signal is low.<br>If this field becomes 0, the MAC transmitter ignores the status of the CRS signal.<br>0b - Disabled<br>1b - Enabled |
| 10<br>DO | Disable Receive Own<br>Enables or disables receive own.<br>If this field is 1, MAC disables the reception of packets when the gmii_txen_o signal asserts in Half-Duplex mode. When the field is 0, MAC receives all the packets that PHY sent.<br>This field is not applicable in Full-Duplex mode.<br>0b - Enabled<br>1b - Disabled |
| 9<br>DCRS | Disable Carrier Sense During Transmission<br>Enables or disables carrier sense during transmission.<br>If this field is 1, the MAC transmitter ignores the (G)MII CRS signal during packet transmission in Half-Duplex mode. As a result, no errors generate because of loss of carrier or no carrier during transmission.<br>When this field becomes 0, the MAC transmitter generates errors because of carrier sense. MAC can even abort the transmission.<br>0b - Enabled<br>1b - Disabled |
| 8<br>DR | Disable Retry<br>Enables or disables retry attempts.<br>If this field is 1, MAC attempts only one transmission. When a collision occurs on GMII or MII, MAC ignores the current packet transmission and reports a packet abort with excessive collision errors in the transmit packet status.<br>If this field becomes 0, MAC retries based on the settings of the BL field of this register.<br>The settings of this field apply only in Half-Duplex mode.<br>0b - Enabled<br>1b - Disabled |
| 7 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 6-5<br>BL | Back-Off Limit<br>Determines the random integer number (r) of slot time delays (4,096 bit times for 1000/2500 Mbit/s; 512 bit times for 10/100 Mbit/s) for which MAC waits before rescheduling a transmission attempt during retry attempts after a collision.<br>• *n* = retransmission attempt<br>• *r* = random integer that takes the value in the range 0 <= r < 2^k<br>This field is applicable only in Half-Duplex mode.<br>00b - k = min(n,10)<br>01b - k = min(n,8)<br>10b - k = min(n,4)<br>11b - k = min(n,1) |
| 4<br>DC | Deferral Check<br>Indicates the status of the deferral check function.<br>If this field is 1, the deferral check function is enabled in MAC, which issues a packet abort status, with Tx_Excessive_Deferral_Error[TXEXSDEF] = 1 in the transmit packet status, when the transmit state machine is deferred for more than 24,288 bit times in 10 Mbit/s or 100 Mbit/s mode.<br>If MAC is configured for a 1000/2500 Mbit/s operation, the threshold for deferral is 155,680 bit times. Deferral begins when the transmitter is ready to transmit, but it is prevented because of an active carrier sense signal (CRS) on GMII or MII.<br>The defer time is not cumulative. For example, if the transmitter defers for 10,000 bit times because the CRS signal is active, and then the CRS signal becomes inactive, the transmitter transmits and collision happens. Because of this collision, the transmitter needs to back off and then defer again after back off completion. In such a scenario, the deferral timer resets to 0 and restarts.<br>When this field becomes 0, the deferral check function disables and MAC defers until the CRS signal becomes inactive.<br>This field is applicable only in Half-Duplex mode.<br>0b - Disabled<br>1b - Enabled |
| 3-2<br>PRELEN | Preamble Length for Transmit Packets<br>Controls the number of preamble bytes that are added to the beginning of every transmit packet.<br>Preamble reduction occurs only when MAC operates in Full-Duplex mode.<br>00b - 7 bytes<br>01b - 5 bytes<br>10b - 3 bytes<br>11b - Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 1<br>TE | Transmitter Enable<br>Indicates the status of the transmitter.<br>• If this field is 1, the MAC transmit state machine is enabled for transmission on GMII or MII.<br>• When the field becomes 0, the MAC transmit state machine is disabled after it completes the transmission of the current packet. The transmit state machine does not transmit any more packets.<br>0b - Disabled<br>1b - Enabled |
| 0<br>RE | Receiver Enable<br>Indicates the status of the receiver.<br>If this field is 1, MAC's receive state machine is enabled for receiving packets from GMII or MII. When this field becomes 0, the MAC receive state machine is disabled after it completes the reception of the current packet. The receive state machine does not receive any more packets from GMII or MII.<br>0b - Disabled<br>1b - Enabled |

### 72.17.3 MAC Extended Configuration (MAC_Ext_Configuration)

#### Offset

| Register | Offset |
|---|---|
| MAC_Ext_Configuration | 4h |

#### Function

Establishes MAC's operating mode.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | Reserved | EIPG | | | | | EIPGEN | Reserved | Reserved | | | PDC | USP | SPEN | DCRCC |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | GPSL | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>— | Reserved |
| 29-25<br>EIPG | Extended Inter-Packet Gap<br>Indicates the value of the extended inter-packet gap.<br>The value of this field is applicable when EIPGEN = 1. The most-significant bits of this field, along with those of MAC_Configuration[IPG], render the minimum IPG greater than 96-bit times in steps of 8-bit times.<br>8'h00: 104-bit times<br>8'h01: 112-bit times<br>8'h02: 120-bit times<br>----------<br>8'hFF: 2144-bit times |
| 24<br>EIPGEN | Extended Inter-Packet Gap Enable<br>Indicates the status of the extended inter-packet gap.<br>• If this field is 1, MAC interprets the EIPG field of this register and the IPG field of MAC Configuration (MAC_Configuration) together as minimum IPG greater than 96-bit times in steps of 8-bit times.<br>• If this field is 0, MAC ignores the EIPG field of this register and interprets the IPG field of MAC Configuration (MAC_Configuration) as minimum IPG less than or equal to 96-bit times in steps of 8-bit times.<br>**NOTE**<br>You must enable the extended inter-packet gap feature when operating in Full-Duplex mode only. There could be undesirable effects on the back-pressure function and frame transmission if the feature is enabled in Half-Duplex mode.<br>0b - Disabled<br>1b - Enabled |
| 23<br>— | Reserved |
| 22-20<br>— | Reserved |
| 19<br>PDC | Packet Duplication Control<br>Indicates the packet duplication control status. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • If this field is 1, the received packet with multicast or broadcast destination address is routed to multiple receive DMA channels. MAC_Address0_High[DCS] identifies the receive DMA channels (corresponding to the MAC Address register) that match the multicast or broadcast destination addresses in the received packet. The DCS field is interpreted to be a one-hot value, each bit corresponding to the receive DMA channel.<br>• If this field is 0, the received packet routes to a single receive DMA channel. MAC_Address0_High[DCS] identifies the receive DMA channel corresponding to the MAC Address register that matches the destination address in the received packet. The DCS field is interpreted as a binary value.<br>0b - Disabled<br>1b - Enabled |
| 18<br>USP | Unicast Slow Protocol Packet Detect<br>Indicates the status of unicast slow-protocol packet detection.<br>• If this field is 1, MAC detects the slow-protocol packets with unicast address of the station specified in MAC Address 0 High (MAC_Address0_High) and MAC Address 0 Low (MAC_Address0_Low). MAC also detects the slow-protocol packets with the slow-protocol multicast address (01-80-C2-00-00-02).<br>• If this field is 0, MAC detects only slow-protocol packets with the slow protocol multicast address specified in IEEE 802.3-2015, section 5.<br>0b - Disabled<br>1b - Enabled |
| 17<br>SPEN | Slow Protocol Detection Enable<br>Enables or disables slow protocol detection.<br>• If this field is 0, EMAC forwards all error-free, slow-protocol packets to the application. EMAC considers such packets as normal-type packets.<br>• If this field is 1, EMAC processes the slow-protocol packets (Ether type 8809h) and provides the slow protocol sub-type and code fields in receive status.<br>0b - Disabled<br>1b - Enabled |
| 16<br>DCRCC | Disable CRC Checking For Received Packets<br>Indicates the status of CRC checking.<br>If this field is 1, MAC receiver does not check the CRC field in the received packets. When this field becomes 0, the MAC receiver always checks the CRC field in the received packets.<br>0b - Enabled<br>1b - Disabled |
| 15-14 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 13-0<br>GPSL | Giant Packet Size Limit<br>Declares the status of a packet based on its size.<br>If the received packet size (in bytes) is greater than the value programmed in this field, MAC declares the received packet as a giant packet. The value programmed in this field must be greater than or equal to 1,518 bytes. Also, any other programmed value is considered as 1,518 bytes.<br>For VLAN-tagged packets, MAC adds 4 bytes to the programmed value. If MAC_VLAN_Tag_Ctrl[EDVLP] = 1, MAC adds 8 bytes to the programmed value for double VLAN-tagged packets.<br>The value of GPSL is valid if MAC_Configuration[GPSLCE] = 1. |

### 72.17.4 MAC Packet Filter (MAC_Packet_Filter)

#### Offset

| Register | Offset |
|---|---|
| MAC_Packet_Filter | 8h |

#### Function

Contains the filter controls for receiving packets.

Some of the controls from this register go to MAC's address check block that performs the first level of address filtering. The second level of filtering is performed on the incoming packet based on other controls such as pass bad packets and pass control packets.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RA | Reserved | | | | | | | | | DNTU | IPFE | Reserved | | | VTFE |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | HPF | SAF | SAIF | PCF | | | DBF | PM | DAIF | HMC | HUC | PR |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>RA | Receive All<br>Indicates whether the received packets are enabled or disabled.<br>• If this field is 1, the MAC Receiver module passes all the received packets to the application, irrespective of their passing the address filter. The result (pass or fail) of the SA or DA filtering is updated in the corresponding field of the receive status word.<br>• If this field is 0, the module passes only those packets to the application that pass the SA or DA address filter.<br>0b - Receive All is disabled<br>1b - Receive All is enabled |
| 30-22<br>— | Reserved |
| 21<br>DNTU | Drop Non-TCP/UDP Over IP Packets<br>Indicates whether MAC drops or forwards non-TCP/UDP protocols over IP packets.<br>• If this field is 1, MAC drops these protocols over IP packets. MAC forwards only those packets that the layer 4 filter processes.<br>• If this field is 0, MAC forwards all these protocols over IP packets.<br>0b - Forwards<br>1b - Drops |
| 20<br>IPFE | Layer 3 and Layer 4 Filter Enable<br>Indicates the status of layer 3 and layer 4 filters.<br>• If this field is 1, MAC drops packets that do not match the enabled layer 3 and layer 4 filters. If these two filters are not enabled for matching, this field does not have any effect.<br>• If this field is 0, MAC forwards all the packets irrespective of the match status of the layer 3 and layer 4 fields.<br>0b - Disabled<br>1b - Enabled |
| 19-17<br>— | Reserved |
| 16<br>VTFE | VLAN Tag Filter Enable<br>Indicates the status of VLAN tag filter.<br>• If this field is 1, MAC drops the VLAN-tagged packets that do not match the VLAN tag.<br>• If this field is 0, MAC forwards all the packets irrespective of the match status of the VLAN tag.<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 15-11<br>— | Reserved |
| 10<br>HPF | Hash Or Perfect Filter<br>Indicates the status of hash or perfect filter.<br>• If this field is 1, the address filter passes a packet if it matches either perfect filtering or hash filtering as defined by the HMC or HUC fields of this register.<br>• If this field is 0 and HUC = 1 or HMC = 1, the packet passes only if it matches the hash filter.<br>0b - Disabled<br>1b - Enabled |
| 9<br>SAF | Source Address Filter Enable<br>Indicates the status of SA filtering.<br>• If this field is 1, MAC compares the SA field of the received packets with the values programmed in the enabled SA registers. If the comparison fails, MAC drops the packet.<br>• When this field becomes 0, MAC forwards the received packet to the application with an updated field value, depending on the SA address comparison.<br>**NOTE**<br>According to the IEEE specification, bit 47 of the SA field is reserved. However, MAC compares all the 48 bits. You must consider this when programming the MAC address registers for SA.<br>0b - Disabled<br>1b - Enabled |
| 8<br>SAIF | SA Inverse Filtering<br>Indicates the status of SA inverse filtering.<br>• If this field is 1, the address check block operates in the inverse filtering mode for SA address comparison. If the SA of a packet matches the values programmed in the SA registers, it is marked as failing the SA address filter.<br>• If this field becomes 0 and the SA of a packet does not match the values programmed in the SA registers, the SA is marked as failing the SA address filter.<br>0b - Disabled<br>1b - Enabled |
| 7-6<br>PCF | Pass Control Packets<br>Controls the forwarding of all the control packets (including unicast and multicast pause packets).<br>00b - MAC filters all the control packets from reaching the application<br>01b - MAC forwards all the control packets, except pause packets, to the application even if they fail the address filter |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 10b - MAC forwards all the control packets to the application even if they fail the address filter<br>11b - MAC forwards all the control packets that pass the address filter |
| 5<br>DBF | Disable Broadcast Packets<br>Enables or disables broadcast packets.<br>• If this field is 1, AFM blocks all the incoming broadcast packets. In addition, it overrides all the other filter settings.<br>• If this field is 0, AFM passes all the received broadcast packets.<br>0b - Enabled<br>1b - Disabled |
| 4<br>PM | Pass All Multicast<br>Enables or disables the passing of received packets.<br>• If this field is 1, it indicates that all the received packets with a multicast destination address (first bit in the destination address field is 1) are passed.<br>• If this field is 0, filtering of multicast packets depends on the settings of the HMC field.<br>0b - Disabled<br>1b - Enabled |
| 3<br>DAIF | DA Inverse Filtering<br>Indicates the status of DA inverse filtering.<br>• If this field is 1, the Address Check block operates in Inverse Filtering mode for the DA address comparison of both unicast and multicast packets.<br>• If this field is 0, normal filtering of packets is performed.<br>0b - Disabled<br>1b - Enabled |
| 2<br>HMC | Hash Multicast<br>Indicates the status of hast multicast.<br>• If this field is 1, MAC performs the destination address filtering of received multicast packets according to the hash table.<br>• If this field is 0, MAC performs the perfect destination address filtering for multicast packets, that is, it compares the DA field with the values programmed in DA registers.<br>0b - Disabled<br>1b - Enabled |
| 1<br>HUC | Hash Unicast<br>Indicates the status of hash unicast. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • If this field is 1, MAC performs the destination address filtering of unicast packets according to the hash table.<br>• When this field becomes 0, MAC performs a perfect destination address filtering for unicast packets, that is, it compares the DA field with the values you program in the MAC Address registers.<br>0b - Disabled<br>1b - Enabled |
| 0<br>PR | Promiscuous Mode<br>Indicates the status of Promiscuous mode.<br>If this field is 1, the address filtering module passes all the incoming packets irrespective of the destination or source address. MAC clears the SA or DA filter fail status fields of the receive status word when PR = 1.<br>0b - Disabled<br>1b - Enabled |

## 72.17.5 MAC Watchdog Timeout (MAC_Watchdog_Timeout)

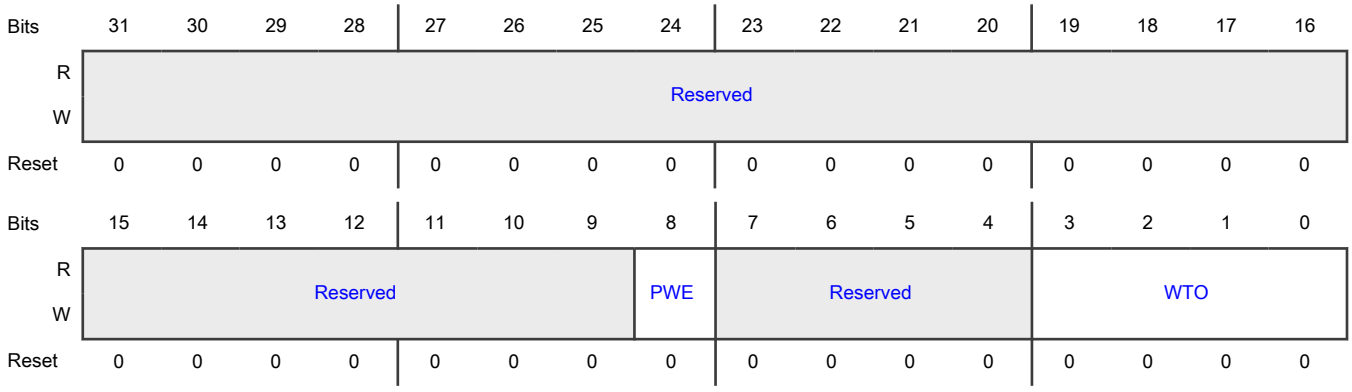
### Offset

| Register | Offset |
|---|---|
| MAC_Watchdog_Timeout | Ch |

### Function

Controls the watchdog timeout for received packets.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>PWE | Programmable Watchdog Enable<br>Indicates the status of programmable watchdog.<br>• If this field is 1 and MAC_Configuration[WD] is 0, the WTO field of this register is used as watchdog timeout for a received packet.<br>• If this field is 0, MAC_Configuration[WD] and MAC_Configuration[JE] control the watchdog timeout for a received packet.<br>0b - Disabled<br>1b - Enabled |
| 7-4<br>— | Reserved |
| 3-0<br>WTO | Watchdog Timeout<br>Indicates the size of watchdog timer.<br>If PWE = 1 and MAC_Configuration[WD] = 0, this field is used as watchdog timeout for a received packet. If the length of a received packet exceeds the value of this field, the packet terminates as an error packet.<br>**NOTE**<br>If PWE = 1, the value in this field must exceed 1,522 (05F2h). Otherwise, the IEEE 802.3-specified valid tagged packets are declared as error packets and dropped.<br>0000b - 2 KB<br>0001b - 3 KB<br>0010b - 4 KB<br>0011b - 5 KB<br>0100b - 6 KB<br>0101b - 7 KB<br>0110b - 8 KB<br>0111b - 9 KB<br>1000b - 10 KB<br>1001b - 11 KB<br>1010b - 12 KB<br>1011b - 13 KB<br>1100b - 14 KB<br>1101b - 15 KB |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1110b - 16383 bytes<br>1111b - Reserved |

## 72.17.6 MAC Hash Table First 32 Bits (MAC_Hash_Table_Reg0)

### Offset

| Register | Offset |
|---|---|
| MAC_Hash_Table_Reg0 | 10h |

### Function

Contains the first 32 bits of the hash table, when the total width of the hash table is 128 bits or 256 bits.

**NOTE**

In this chip, upper 6 bits should be used for 64-bit Hash Table.

The hash table is used for group address filtering, for which the content of the destination address in the incoming packet is passed through the CRC logic and the bits of Receive CRC Error Packets (Rx_CRC_Error_Packets) are used to index the contents of the hash table as follows:

- 6 bits when you have a 64-bit hash
- 7 bits when you have a 128-bit hash
- 8 bits when you have a 256-bit hash

The most-significant bits determine the Hash Table register to be used, and the least-significant 5 bits determine the bit within the register. For example, a hash value of 10_0000b (in 64-bit hash) selects bit 0 of MAC Hash Table Second 32 Bits (MAC_Hash_Table_Reg1).

Perform these steps to calculate the hash value of the destination address:

1. Calculate the 32-bit CRC for DA (see section 3.2.8 in IEEE 802.3 for the steps to calculate CRC32).
2. Perform bit-wise reversal for the value obtained in step 1.
3. Use the upper 6, 7, or 8 bits from the value obtained in step 2.

If this field is 1, the packet is accepted. Otherwise, it is rejected. If MAC_Packet_Filter[PM] = 1, all multicast packets are accepted regardless of the multicast hash values.

If this register is configured to be double-synchronized with the (G)MII clock domain, the synchronization is triggered only when bits [31:24] (in Little-Endian mode) or bits [7:0] (in Big-Endian mode) of the Hash Table registers are written to.

If double-synchronization is enabled, consecutive writes to this register must be performed after at least four clock cycles in the destination clock domain.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | HT31T0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | HT31T0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>HT31T0 | MAC Hash Table First 32 Bits<br>Contains the first 32 bits [31:0] of the hash table. |

## 72.17.7 MAC Hash Table Second 32 Bits (MAC_Hash_Table_Reg1)

### Offset

| Register | Offset |
|---|---|
| MAC_Hash_Table_Reg1 | 14h |

### Function

Contains the second 32 bits of the hash table, when the width of the hash table is 128 or 256 bits.

**NOTE**

In this chip, upper 6 bits should be used for 64-bit Hash Table.

The hash table is used for group address filtering, for which the content of the destination address in the incoming packet is passed through the CRC logic and the bits of Receive CRC Error Packets (Rx_CRC_Error_Packets) are used to index the contents of the hash table as follows:

- 6 bits when you have a 64-bit hash
- 7 bits when you have a 128-bit hash
- 8 bits when you have a 256-bit hash

The most-significant bits determine the Hash Table register to be used, and the least-significant 5 bits determine the bit within the register. For example, a hash value of 10_0000b (in 64-bit hash) selects bit 0 of MAC Hash Table Second 32 Bits (MAC_Hash_Table_Reg1).

Perform these steps to calculate the hash value of the destination address:

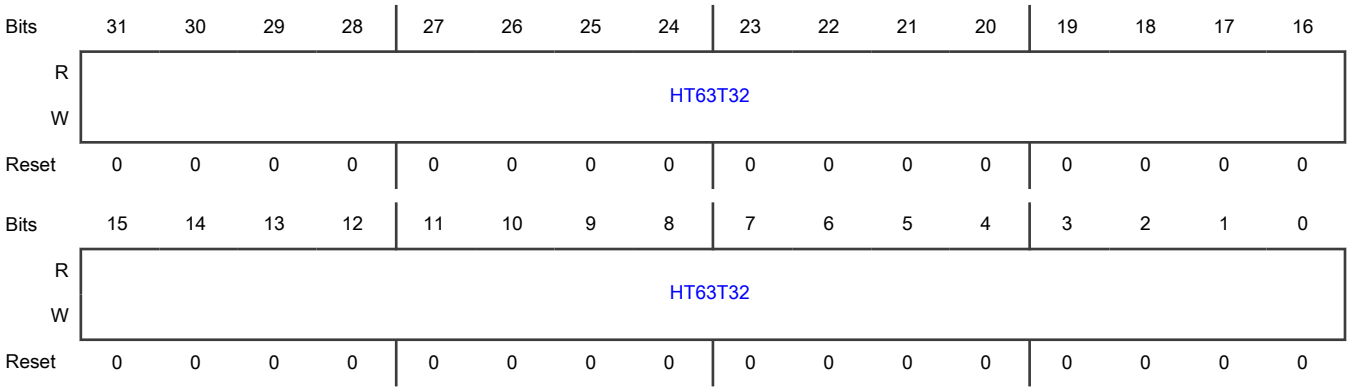
1. Calculate the 32-bit CRC for DA (see section 3.2.8 in IEEE 802.3 for the steps to calculate CRC32).
2. Perform bit-wise reversal for the value obtained in step 1.
3. Take the upper 6, 7, or 8 bits from the value obtained in step 2.

If the corresponding bit value of this register is 1'b1, the packet is accepted. Otherwise, it is rejected. If MAC_Packet_Filter[PM] = 1, all multicast packets are accepted regardless of the multicast hash values.

If this register is configured to be double-synchronized with the (G)MII clock domain, the synchronization is triggered only when bits [31:24] (in Little-Endian mode) or bits [7:0] (in Big-Endian mode) of the Hash Table registers are written to.

If double-synchronization is enabled, consecutive writes to this register must be performed after at least four clock cycles in the destination clock domain.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>HT63T32 | MAC Hash Table Second 32 Bits<br>Contains the second 32 bits [63:32] of the hash table. |

## 72.17.8 MAC VLAN Tag (MAC_VLAN_Tag)

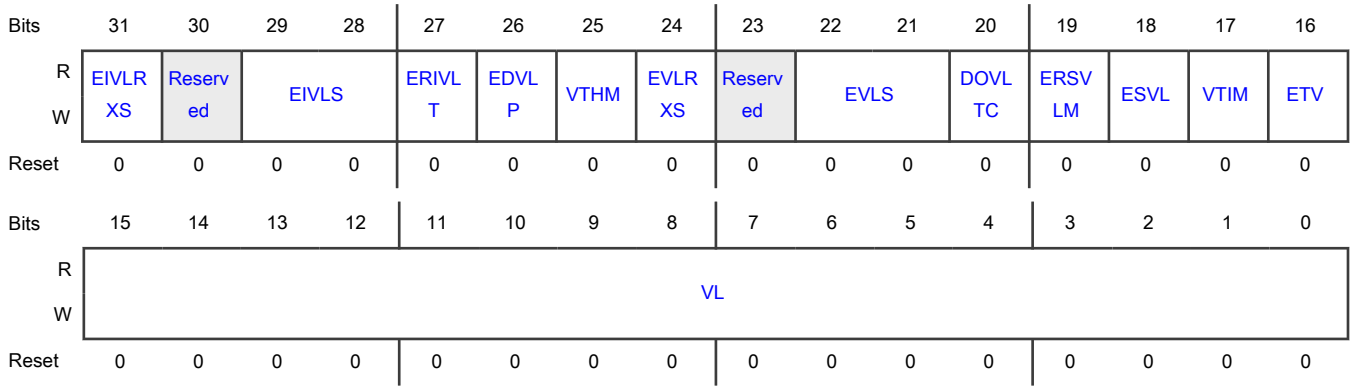
### Offset

| Register | Offset |
|---|---|
| MAC_VLAN_Tag | 50h |

### Function

Identifies the IEEE 802.1Q VLAN type packets.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>EIVLRXS | Enable Inner VLAN Tag In Receive Status<br>Indicates whether the inner VLAN tag in receive status is enabled or disabled.<br>• If this field is 1, MAC provides the inner VLAN tag in the receive status.<br>• If this field is 0, MAC does not provide the inner VLAN tag in the receive status.<br>0b - Disabled<br>1b - Enabled |
| 30<br>— | Reserved |
| 29-28<br>EIVLS | Enable Inner VLAN Tag Stripping<br>Indicates the stripping operation on the inner VLAN tag in a received packet. The field enables or disables inner VLAN tag stripping on receive.<br>00b - Do not strip<br>01b - Strip if VLAN filter passes<br>10b - Strip if VLAN filter fails<br>11b - Always strip |
| 27<br>ERIVLT | Enable Inner VLAN Tag Comparison<br>Enables or disables the inner VLAN tag.<br>• If ERIVLT = VTHM = EDVLP = 1, the EMAC receiver enables the VLAN hash filtering operation on the inner VLAN tag (if present).<br>• If ERIVLT = 0 and VTHM = 1, the EMAC receiver enables the VLAN hash filtering operation on the outer VLAN tag (if present). ERSVLM and DOVLTC determine which VLAN type is enabled for filtering.<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 26<br>EDVLP | Enable Double VLAN Processing<br>Enables or disables double VLAN processing.<br>• If this field is 1, MAC enables processing of up to two VLAN tags on transmit and receive (if present).<br>• If this field is 0, MAC enables processing of up to one VLAN tag on transmit and receive (if present).<br>0b - Disabled<br>1b - Enabled |
| 25<br>VTHM | VLAN Tag Hash Table Match<br>Indicates the status of VLAN tag hash table match.<br>• If this field is 1, the most-significant four bits of CRC-32 of VLAN tag are used to index the content of MAC VLAN Hash Table (MAC_VLAN_Hash_Table). See VLAN filtering for details.<br>• If MAC_VLAN_Hash_Table[VLHT] = 1, corresponding to the index, it indicates that the Ethernet packet matches the VLAN hash table. See VLAN filtering for details.<br>• If ETV = 1, the CRC of the 12-bit VLAN identifier (VID) is used for comparison. When the ETV field becomes 0, the CRC of the 16-bit VLAN tag is used for comparison.<br>• If VTHM = 1, the VLAN hash match operation is not performed.<br>0b - Disabled<br>1b - Enabled |
| 24<br>EVLRXS | Enable VLAN Tag In Receive Status<br>Indicates whether the VLAN tag in receive status is enabled.<br>• If this field is 1, MAC provides the outer VLAN tag in the receive status.<br>• If this field is 0, MAC does not provide the outer VLAN tag in receive status.<br>0b - Disabled<br>1b - Enabled |
| 23<br>— | Reserved |
| 22-21<br>EVLS | Enable VLAN Tag Stripping<br>Indicates the stripping operation on the outer VLAN tag in received packets. The field enables or disables VLAN tag stripping on receive.<br>00b - Do not strip<br>01b - Strip if VLAN filter passes<br>10b - Strip if VLAN filter fails<br>11b - Always strip |
| 20 | Disable VLAN Type Check |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| DOVLTC | Enables or disables VLAN type check for VLAN hash filtering.<br>• If this field is 1, the MAC VLAN hash filter matches or filters the VLAN tag that the ERIVLT field specifies only when the VLAN tag type is similar to the one that the ERSVLM field specifies.<br>• If this field is 0, the MAC VLAN hash filter does not check whether the VLAN tag that the ERIVLT field specifies is of type S-VLAN or C-VLAN.<br>• The VLAN filter is bypassed when VLAN type of received packet do not match the programmed VLAN type in the VLAN filter.<br>0b - Enabled<br>1b - Disabled |
| 19<br>ERSVLM | Enable Receive S-VLAN Match<br>Enables or disables receive S-VLAN match for VLAN hash filtering.<br>• If this field is 1, the MAC receiver enables VLAN hash filtering or S-VLAN matching (type = 88A8h) packets.<br>• If this field is 0, the MAC receiver enables VLAN hash filtering or matching for C-VLAN (type = 8100h) packets.<br>The ERIVLT field determines the VLAN tag position considered for VLAN hash filtering or matching.<br>0b - Disabled<br>1b - Enabled |
| 18<br>ESVL | Enable S-VLAN<br>Enables or disables S-VLAN.<br>If this field is 1, the MAC transmitter and receiver consider the S-VLAN packets (type = 88A8h) as valid VLAN-tagged packets.<br>0b - Disabled<br>1b - Enabled |
| 17<br>VTIM | VLAN Tag Inverse Match Enable<br>Enables or disables VLAN tag inverse matching.<br>• If this field is 1, it enables the VLAN tag inverse matching. The packets without matching VLAN tag are marked as matched.<br>• If this field is 0, it enables the VLAN tag perfect matching. The packets with matched VLAN tag are marked as matched.<br>0b - Disabled<br>1b - Enabled |
| 16<br>ETV | Enable Tag For VLAN<br>Enables or disables 12-bit VLAN tag comparison for VLAN hash filtering. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • If this field is 1, a 12-bit VLAN identifier is used for VLAN hash filtering instead of the complete 16-bit VLAN tag. Bits [11:0] of the VLAN tag in the received VLAN-tagged packet are used for hash-based VLAN filtering.<br>• When this field becomes 0, all 16 bits of the 15th and 16th bytes of the received VLAN packet are used for VLAN hash filtering.<br>0b - Disabled<br>1b - Enabled |
| 15-0<br>VL | VLAN Tag Identifier for Receive Packets<br>Contains the 802.1Q VLAN tag to identify the VLAN packets.<br>This VLAN tag identifier is compared to the 15th and 16th bytes of the packets being received for VLAN packets.<br>The following list describes the bits of this field:<br>• 15:13 — User priority<br>• 12 — Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)<br>• 11:0 — VID field of VLAN tag<br>When ETV = 1, only the VID is used for comparison.<br>When VL = 0, and ETV = 1, MAC does not check the 15th and 16th bytes for VLAN tag comparison and declares all packets with Type field value of 8100h or 88a8h as VLAN packets. |

### 72.17.9 MAC VLAN Tag Control (MAC_VLAN_Tag_Ctrl)

#### Offset

| Register | Offset |
|---|---|
| MAC_VLAN_Tag_Ctrl | 50h |

#### Function

Is the redefined version of the MAC_VLAN_Tag register.

This register provides the control and addressing fields required for indirect access to the MAC_VLAN_Tag_Filter registers. It also contains the address offset, command type, and busy bit for CSR access of the MAC VLAN Hash Filter registers.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | EIVLRXS | Reserved | EIVLS | | ERIVLT | EDVLP | VTHM | EVLRXS | Reserved | EVLS | | DOVLTC | ERSVLM | ESVL | VTIM | ETV |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | | | | | OFS | | CT | OB |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>EIVLRXS | Enable Inner VLAN Tag In Receive Status<br>Indicates whether the inner VLAN tag in receive status is enabled or disabled.<br>• If this field is 1, MAC provides the inner VLAN tag in the receive status.<br>• If this field is 0, MAC does not provide the inner VLAN tag in the receive status.<br>0b - Disabled<br>1b - Enabled |
| 30<br>— | Reserved |
| 29-28<br>EIVLS | Enable Inner VLAN Tag Stripping<br>Indicates the stripping operation on the inner VLAN tag in a received packet. The field enables or disables inner VLAN tag stripping on receive.<br>00b - Do not strip<br>01b - Strip if VLAN filter passes<br>10b - Strip if VLAN filter fails<br>11b - Always strip |
| 27<br>ERIVLT | Enable Inner VLAN Tag Comparison<br>Enables or disables the inner VLAN tag.<br>• If ERIVLT = VTHM = EDVLP = 1, the EMAC receiver enables the VLAN hash filtering operation on the inner VLAN tag (if present).<br>• If ERIVLT = 0 and VTHM = 1, the EMAC receiver enables the VLAN hash filtering operation on the outer VLAN tag (if present). ERSVLM and DOVLTC determine which VLAN type is enabled for filtering.<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 26<br>EDVLP | Enable Double VLAN Processing<br>Enables or disables double VLAN processing.<br>• If this field is 1, MAC enables processing of up to two VLAN tags on transmit and receive (if present).<br>• If this field is 0, MAC enables processing of up to one VLAN tag on transmit and receive (if present).<br>0b - Disabled<br>1b - Enabled |
| 25<br>VTHM | VLAN Tag Hash Table Match<br>Indicates the status of VLAN tag hash table match.<br>• If this field is 1, the most-significant four bits of CRC-32 of VLAN tag are used to index the content of MAC VLAN Hash Table (MAC_VLAN_Hash_Table). See VLAN filtering for details.<br>• If MAC_VLAN_Hash_Table[VLHT] = 1, corresponding to the index, it indicates that the Ethernet packet matches the VLAN hash table. See VLAN filtering for details.<br>• If ETV = 1, the CRC of the 12-bit VLAN identifier (VID) is used for comparison. When the ETV field becomes 0, the CRC of the 16-bit VLAN tag is used for comparison.<br>• If VTHM = 1, the VLAN hash match operation is not performed.<br>0b - Disabled<br>1b - Enabled |
| 24<br>EVLRXS | Enable VLAN Tag In Receive Status<br>Indicates whether the VLAN tag in receive status is enabled.<br>• If this field is 1, MAC provides the outer VLAN tag in the receive status.<br>• If this field is 0, MAC does not provide the outer VLAN tag in receive status.<br>0b - Disabled<br>1b - Enabled |
| 23<br>— | Reserved |
| 22-21<br>EVLS | Enable VLAN Tag Stripping<br>Indicates the stripping operation on the outer VLAN tag in received packets. The field enables or disables VLAN tag stripping on receive.<br>00b - Do not strip<br>01b - Strip if VLAN filter passes<br>10b - Strip if VLAN filter fails<br>11b - Always strip |
| 20 | Disable VLAN Type Check |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| DOVLTC | Enables or disables VLAN type check for VLAN hash filtering.<br>• If this field is 1, the MAC VLAN hash filter matches or filters the VLAN tag that the ERIVLT field specifies only when the VLAN tag type is similar to the one that the ERSVLM field specifies.<br>• If this field is 0, the MAC VLAN hash filter does not check whether the VLAN tag that the ERIVLT field specifies is of type S-VLAN or C-VLAN.<br>0b - Enabled<br>1b - Disabled |
| 19<br>ERSVLM | Enable Receive S-VLAN Match<br>Enables or disables receive S-VLAN match for VLAN hash filtering.<br>• If this field is 1, the MAC receiver enables VLAN hash filtering or S-VLAN matching (type = 88A8h) packets.<br>• If this field is 0, the MAC receiver enables VLAN hash filtering or matching for C-VLAN (type = 8100h) packets.<br>The ERIVLT field determines the VLAN tag position considered for VLAN hash filtering or matching.<br>0b - Disabled<br>1b - Enabled |
| 18<br>ESVL | Enable S-VLAN<br>Enables or disables S-VLAN.<br>If this field is 1, the MAC transmitter and receiver consider the S-VLAN packets (type = 88A8h) as valid VLAN-tagged packets.<br>0b - Disabled<br>1b - Enabled |
| 17<br>VTIM | VLAN Tag Inverse Match Enable<br>Enables or disables VLAN tag inverse matching.<br>• If this field is 1, it enables the VLAN tag inverse matching. The packets without matching VLAN tag are marked as matched.<br>• If this field is 0, it enables the VLAN tag perfect matching. The packets with matched VLAN tag are marked as matched.<br>0b - Disabled<br>1b - Enabled |
| 16<br>ETV | Enable Tag For VLAN<br>Enables or disables 12-bit VLAN tag comparison for VLAN hash filtering.<br>• If this field is 1, a 12-bit VLAN identifier is used for VLAN hash filtering instead of the complete 16-bit VLAN tag. Bits [11:0] of the VLAN tag in the received VLAN-tagged packet are used for hash-based VLAN filtering. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • When this field becomes 0, all 16 bits of the 15th and 16th bytes of the received VLAN packet are used for VLAN hash filtering.<br>0b - Disabled<br>1b - Enabled |
| 15-4<br>— | Reserved |
| 3-2<br>OFS | Offset<br>Holds the address offset of the MAC VLAN Tag Filter register that the application tries to access.<br>The width of this field depends on the number of enabled MAC VLAN Tag registers. |
| 1<br>CT | Command Type<br>Specifies whether the current register access indicates a read or a write operation. It indicates a read operation if this field is 1 and a write operation when the field is 0.<br>0b - Write operation<br>1b - Read operation |
| 0<br>OB | Operation Busy<br>Indicates the operation busy status.<br>• This field becomes 1 to initiate a read or write command for an indirect access to the Per VLAN Tag Filter register.<br>• The field becomes 0 when the read or write command to Per VLAN Tag Filter indirect access register completes. The next indirect register access can be initiated only after the field resets.<br>During a write operation, the field becomes 0 only after the data is written into a MAC_VLAN_Tag_Filter register.<br>During a read operation, the data must be read from MAC VLAN Tag Data (MAC_VLAN_Tag_Data) only after this field becomes 0.<br>0b - Disabled<br>1b - Enabled |

### 72.17.10 MAC VLAN Tag Data (MAC_VLAN_Tag_Data)

#### Offset

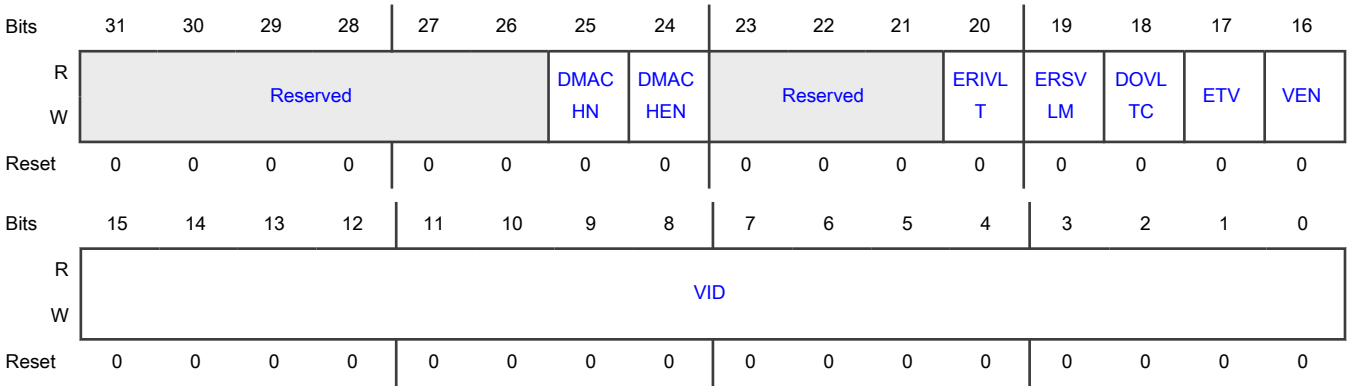
| Register | Offset |
|---|---|
| MAC_VLAN_Tag_Data | 54h |

#### Function

Holds the read and write data for indirect access to the Per VLAN Tag registers.

- During read access, this register contains valid read data only after MAC_VLAN_Tag_Ctrl[OB] becomes 0.
- During write access, this register must become valid before you write 1 to MAC_VLAN_Tag_Ctrl[OB].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25<br>DMACHN | DMA Channel Number<br>Indicates the DMA channel number to which the VLAN-tagged frame is to be routed if it passes the VLAN tag filter programmed using this field.<br>If the routing based on VLAN tag filter is not necessary, this field does not need to be programmed. |
| 24<br>DMACHEN | DMA Channel Number Enable<br>Enables or disables the DMA channel number value programmed using the DMACHN field.<br>If this field is 0, the routing does not occur based on the VLAN filter result. The frame is routed on the basis of DA-based DMA channel routing.<br>0b - Disabled<br>1b - Enabled |
| 23-21<br>— | Reserved |
| 20<br>ERIVLT | Enable Inner VLAN Tag<br>Enables or disables inner VLAN tag comparison.<br>• This field is valid only when the double VLAN tag enable of the filter is set.<br>• If this field and the EDVLP field is 1, the MAC receiver enables operation on the inner VLAN tag (if present).<br>• If this field is 0, the MAC receiver enables operation on the outer VLAN tag (if present). |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |
| 19<br>ERSVLM | Enable S-VLAN Match<br>Enables or disables S-VLAN match for received frames.<br>This field is valid only when the VLAN tag enable of the filter is set.<br>• If this field is 1, the MAC receiver enables filtering or matching for S-VLAN (type = 88A8h) packets.<br>• If this field is 0, the MAC receiver enables filtering or matching for C-VLAN (type = 8100h) packets.<br>0b - Disabled<br>1b - Enabled |
| 18<br>DOVLTC | Disable VLAN Type Comparison<br>Enables or disables VLAN type comparison.<br>This field is valid only when the VEN field is 1.<br>• If this field is 1, MAC does not check whether the VLAN tag that ERIVLT specifies is of type S-VLAN or C-VLAN.<br>• If this field is 0, MAC filters or matches the VLAN tag that the ERIVLT field specifies only when the VLAN tag type is similar to the one that the ERSVLM field specifies.<br>0b - Enabled<br>1b - Disabled |
| 17<br>ETV | VLAN Comparison<br>Indicates 12-bit or 16-bit VLAN comparison.<br>• This field is valid only when the VEN field is 1.<br>• When the value of this field is 1, a 12-bit VLAN identifier is used for filter and comparison instead of the complete 16-bit VLAN tag. Bits [11:0] of the VLAN tag are compared with the corresponding field in the received VLAN-tagged packet.<br>0b - 16-bit VLAN comparison<br>1b - 12-bit VLAN comparison |
| 16<br>VEN | VLAN Tag Enable<br>Enables or disables the VLAN tag.<br>• If this field is 1, MAC compares the VLAN tag of the received packet with the VLAN tag ID.<br>• If this field is 0, no comparison is performed irrespective of the programming of the other fields.<br>0b - Disabled<br>1b - Enabled |
| 15-0 | VLAN Tag ID |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| VID | Holds the VLAN tag value that MAC uses for perfect comparison. The field is valid when VEN = 1. |

## 72.17.11 MAC VLAN Tag Filter 0 (MAC_VLAN_Tag_Filter0)

### Offset

| Register | Offset |
|---|---|
| MAC_VLAN_Tag_Filter0 | 54h |

### Function

Contains VLAN tag filter 0 control information.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | DMACHN | DMACHEN | Reserved | | | ERIVLT | ERSVLM | DOVLTC | ETV | VEN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25<br>DMACHN | DMA Channel Number<br>Indicates the DMA channel number to which the VLAN-tagged frame is to be routed if it passes the VLAN tag filter programmed using this field.<br>If the routing based on VLAN tag filter is not necessary, this field does not need to be programmed. |
| 24<br>DMACHEN | DMA Channel Number Enable<br>Enables or disables the DMA channel number value programmed using the DMACHN field.<br>If this field is 0, the routing does not occur based on the VLAN filter result. The frame is routed on the basis of DA-based DMA channel routing. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |
| 23-21<br>— | Reserved |
| 20<br>ERIVLT | Enable Inner VLAN Tag<br>Enables or disables inner VLAN tag comparison.<br>• This field is valid only when the double VLAN tag enable of the filter is set.<br>• If this field and the EDVLP field is 1, the MAC receiver enables operation on the inner VLAN tag (if present).<br>• If this field is 0, the MAC receiver enables operation on the outer VLAN tag (if present).<br>0b - Disabled<br>1b - Enabled |
| 19<br>ERSVLM | Enable S-VLAN Match<br>Enables or disables S-VLAN match for received frames.<br>This field is valid only when the VLAN tag enable of the filter is set.<br>• If this field is 1, the MAC receiver enables filtering or matching for S-VLAN (type = 88A8h) packets.<br>• If this field is 0, the MAC receiver enables filtering or matching for C-VLAN (type = 8100h) packets.<br>0b - Disabled<br>1b - Enabled |
| 18<br>DOVLTC | Disable VLAN Type Comparison<br>Enables or disables VLAN type comparison.<br>This field is valid only when the VEN field is 1.<br>• If this field is 1, MAC does not check whether the VLAN tag that ERIVLT specifies is of type S-VLAN or C-VLAN.<br>• If this field is 0, MAC filters or matches the VLAN tag that the ERIVLT field specifies only when the VLAN tag type is similar to the one that the ERSVLM field specifies.<br>0b - Enabled<br>1b - Disabled |
| 17<br>ETV | VLAN Comparison<br>Indicates 12-bit or 16-bit VLAN comparison.<br>• This field is valid only when the VEN field is 1.<br>• When the value of this field is 1, a 12-bit VLAN identifier is used for filter and comparison instead of the complete 16-bit VLAN tag. Bits [11:0] of the VLAN tag are compared with the corresponding field in the received VLAN-tagged packet. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - 16-bit VLAN comparison<br>1b - 12-bit VLAN comparison |
| 16<br>VEN | VLAN Tag Enable<br>Enables or disables the VLAN tag.<br>• If this field is 1, MAC compares the VLAN tag of the received packet with the VLAN tag ID.<br>• If this field is 0, no comparison is performed irrespective of the programming of the other fields.<br>0b - Disabled<br>1b - Enabled |
| 15-0<br>VID | VLAN Tag ID<br>Holds the VLAN tag value that MAC uses for perfect comparison. The field is valid when VEN = 1. |

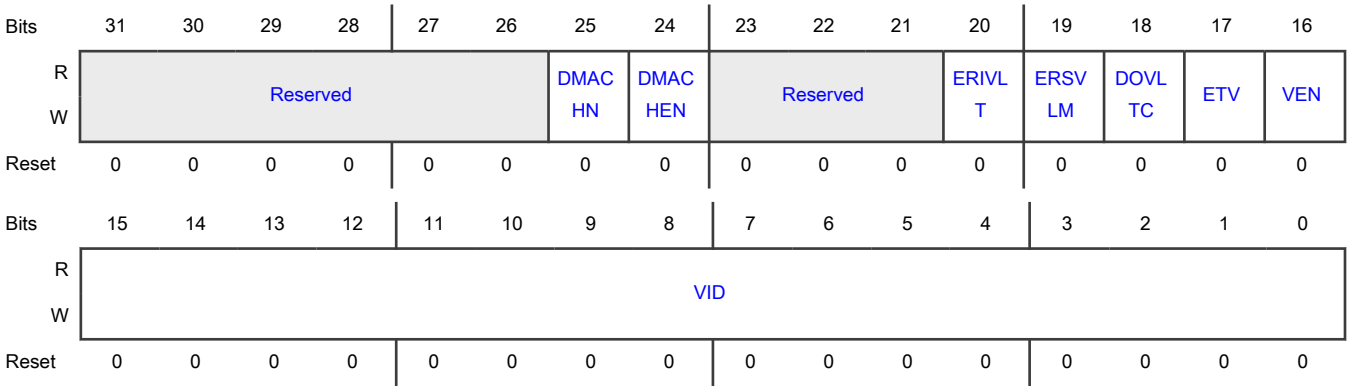
## 72.17.12 MAC VLAN Tag Filter 1 (MAC_VLAN_Tag_Filter1)

### Offset

| Register | Offset |
|---|---|
| MAC_VLAN_Tag_Filter1 | 54h |

### Function
Contains VLAN tag filter 1 control information.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25<br>DMACHN | DMA Channel Number<br>Indicates the DMA channel number to which the VLAN-tagged frame is to be routed if it passes the VLAN tag filter programmed using this field.<br>If the routing based on VLAN tag filter is not necessary, this field does not need to be programmed. |
| 24<br>DMACHEN | DMA Channel Number Enable<br>Enables or disables the DMA channel number value programmed using the DMACHN field.<br>If this field is 0, the routing does not occur based on the VLAN filter result. The frame is routed on the basis of DA-based DMA channel routing.<br>0b - Disabled<br>1b - Enabled |
| 23-21<br>— | Reserved |
| 20<br>ERIVLT | Enable Inner VLAN Tag<br>Enables or disables inner VLAN tag comparison.<br>• This field is valid only when the double VLAN tag enable of the filter is set.<br>• If this field and the EDVLP field is 1, the MAC receiver enables operation on the inner VLAN tag (if present).<br>• If this field is 0, the MAC receiver enables operation on the outer VLAN tag (if present).<br>0b - Disabled<br>1b - Enabled |
| 19<br>ERSVLM | Enable S-VLAN Match<br>Enables or disables S-VLAN match for received frames.<br>This field is valid only when the VLAN tag enable of the filter is set.<br>• If this field is 1, the MAC receiver enables filtering or matching for S-VLAN (type = 88A8h) packets.<br>• If this field is 0, the MAC receiver enables filtering or matching for C-VLAN (type = 8100h) packets.<br>0b - Disabled<br>1b - Enabled |
| 18<br>DOVLTC | Disable VLAN Type Comparison<br>Enables or disables VLAN type comparison.<br>This field is valid only when the VEN field is 1. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • If this field is 1, MAC does not check whether the VLAN tag that ERIVLT specifies is of type S-VLAN or C-VLAN.<br>• If this field is 0, MAC filters or matches the VLAN tag that the ERIVLT field specifies only when the VLAN tag type is similar to the one that the ERSVLM field specifies.<br>0b - Enabled<br>1b - Disabled |
| 17<br>ETV | VLAN Comparison<br>Indicates 12-bit or 16-bit VLAN comparison.<br>• This field is valid only when the VEN field is 1.<br>• When the value of this field is 1, a 12-bit VLAN identifier is used for filter and comparison instead of the complete 16-bit VLAN tag. Bits [11:0] of the VLAN tag are compared with the corresponding field in the received VLAN-tagged packet.<br>0b - 16-bit VLAN comparison<br>1b - 12-bit VLAN comparison |
| 16<br>VEN | VLAN Tag Enable<br>Enables or disables the VLAN tag.<br>• If this field is 1, MAC compares the VLAN tag of the received packet with the VLAN tag ID.<br>• If this field is 0, no comparison is performed irrespective of the programming of the other fields.<br>0b - Disabled<br>1b - Enabled |
| 15-0<br>VID | VLAN Tag ID<br>Holds the VLAN tag value that MAC uses for perfect comparison. The field is valid when VEN = 1. |

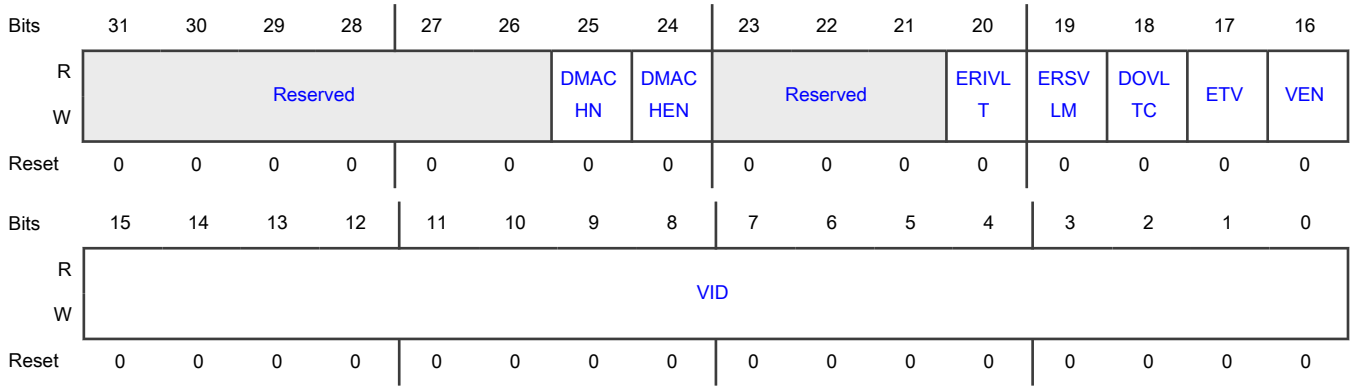
### 72.17.13 MAC VLAN Tag Filter 2 (MAC_VLAN_Tag_Filter2)

**Offset**

| Register | Offset |
|---|---|
| MAC_VLAN_Tag_Filter2 | 54h |

**Function**
Contains VLAN tag filter 2 control information.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25<br>DMACHN | DMA Channel Number<br>Indicates the DMA channel number to which the VLAN-tagged frame is to be routed if it passes the VLAN tag filter programmed using this field.<br>If the routing based on VLAN tag filter is not necessary, this field does not need to be programmed. |
| 24<br>DMACHEN | DMA Channel Number Enable<br>Enables or disables the DMA channel number value programmed using the DMACHN field.<br>If this field is 0, the routing does not occur based on the VLAN filter result. The frame is routed on the basis of DA-based DMA channel routing.<br>0b - Disabled<br>1b - Enabled |
| 23-21<br>— | Reserved |
| 20<br>ERIVLT | Enable Inner VLAN Tag<br>Enables or disables inner VLAN tag comparison.<br>• This field is valid only when the double VLAN tag enable of the filter is set.<br>• If this field and the EDVLP field is 1, the MAC receiver enables operation on the inner VLAN tag (if present).<br>• If this field is 0, the MAC receiver enables operation on the outer VLAN tag (if present).<br>0b - Disabled<br>1b - Enabled |
| 19 | Enable S-VLAN Match |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| ERSVLM | Enables or disables S-VLAN match for received frames.<br>This field is valid only when the VLAN tag enable of the filter is set.<br>• If this field is 1, the MAC receiver enables filtering or matching for S-VLAN (type = 88A8h) packets.<br>• If this field is 0, the MAC receiver enables filtering or matching for C-VLAN (type = 8100h) packets.<br>0b - Disabled<br>1b - Enabled |
| 18<br>DOVLTC | Disable VLAN Type Comparison<br>Enables or disables VLAN type comparison.<br>This field is valid only when the VEN field is 1.<br>• If this field is 1, MAC does not check whether the VLAN tag that ERIVLT specifies is of type S-VLAN or C-VLAN.<br>• If this field is 0, MAC filters or matches the VLAN tag that the ERIVLT field specifies only when the VLAN tag type is similar to the one that the ERSVLM field specifies.<br>0b - Enabled<br>1b - Disabled |
| 17<br>ETV | VLAN Comparison<br>Indicates 12-bit or 16-bit VLAN comparison.<br>• This field is valid only when the VEN field is 1.<br>• When the value of this field is 1, a 12-bit VLAN identifier is used for filter and comparison instead of the complete 16-bit VLAN tag. Bits [11:0] of the VLAN tag are compared with the corresponding field in the received VLAN-tagged packet.<br>0b - 16-bit VLAN comparison<br>1b - 12-bit VLAN comparison |
| 16<br>VEN | VLAN Tag Enable<br>Enables or disables the VLAN tag.<br>• If this field is 1, MAC compares the VLAN tag of the received packet with the VLAN tag ID.<br>• If this field is 0, no comparison is performed irrespective of the programming of the other fields.<br>0b - Disabled<br>1b - Enabled |
| 15-0<br>VID | VLAN Tag ID<br>Holds the VLAN tag value that MAC uses for perfect comparison. The field is valid when VEN = 1. |

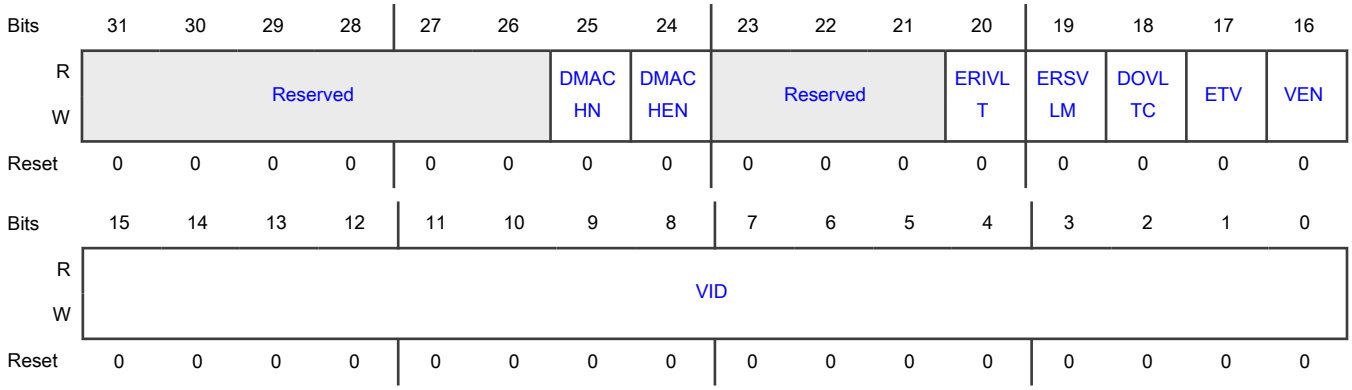
## 72.17.14 MAC VLAN Tag Filter 3 (MAC_VLAN_Tag_Filter3)

### Offset

| Register | Offset |
|---|---|
| MAC_VLAN_Tag_Filter3 | 54h |

### Function
Contains VLAN tag filter 3 control information.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25<br>DMACHN | DMA Channel Number<br>Indicates the DMA channel number to which the VLAN-tagged frame is to be routed if it passes the VLAN tag filter programmed using this field.<br>If the routing based on VLAN tag filter is not necessary, this field does not need to be programmed. |
| 24<br>DMACHEN | DMA Channel Number Enable<br>Enables or disables the DMA channel number value programmed using the DMACHN field.<br>If this field is 0, the routing does not occur based on the VLAN filter result. The frame is routed on the basis of DA-based DMA channel routing.<br>0b - Disabled<br>1b - Enabled |
| 23-21<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 20<br>ERIVLT | Enable Inner VLAN Tag<br>Enables or disables inner VLAN tag comparison.<br>• This field is valid only when the double VLAN tag enable of the filter is set.<br>• If this field and the EDVLP field is 1, the MAC receiver enables operation on the inner VLAN tag (if present).<br>• If this field is 0, the MAC receiver enables operation on the outer VLAN tag (if present).<br>0b - Disabled<br>1b - Enabled |
| 19<br>ERSVLM | Enable S-VLAN Match<br>Enables or disables S-VLAN match for received frames.<br>This field is valid only when the VLAN tag enable of the filter is set.<br>• If this field is 1, the MAC receiver enables filtering or matching for S-VLAN (type = 88A8h) packets.<br>• If this field is 0, the MAC receiver enables filtering or matching for C-VLAN (type = 8100h) packets.<br>0b - Disabled<br>1b - Enabled |
| 18<br>DOVLTC | Disable VLAN Type Comparison<br>Enables or disables VLAN type comparison.<br>This field is valid only when the VEN field is 1.<br>• If this field is 1, MAC does not check whether the VLAN tag that ERIVLT specifies is of type S-VLAN or C-VLAN.<br>• If this field is 0, MAC filters or matches the VLAN tag that the ERIVLT field specifies only when the VLAN tag type is similar to the one that the ERSVLM field specifies.<br>0b - Enabled<br>1b - Disabled |
| 17<br>ETV | VLAN Comparison<br>Indicates 12-bit or 16-bit VLAN comparison.<br>• This field is valid only when the VEN field is 1.<br>• When the value of this field is 1, a 12-bit VLAN identifier is used for filter and comparison instead of the complete 16-bit VLAN tag. Bits [11:0] of the VLAN tag are compared with the corresponding field in the received VLAN-tagged packet.<br>0b - 16-bit VLAN comparison<br>1b - 12-bit VLAN comparison |
| 16<br>VEN | VLAN Tag Enable<br>Enables or disables the VLAN tag. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • If this field is 1, MAC compares the VLAN tag of the received packet with the VLAN tag ID.<br>• If this field is 0, no comparison is performed irrespective of the programming of the other fields.<br>0b - Disabled<br>1b - Enabled |
| 15-0<br>VID | VLAN Tag ID<br>Holds the VLAN tag value that MAC uses for perfect comparison. The field is valid when VEN = 1. |

## 72.17.15 MAC VLAN Hash Table (MAC_VLAN_Hash_Table)

### Offset

| Register | Offset |
|---|---|
| MAC_VLAN_Hash_Table | 58h |

### Function

If MAC_VLAN_Tag_Ctrl[VTHM] = 1, the 16-bit VLAN hash table register is used for group address filtering based on the VLAN tag. For hash filtering, the content of the 16-bit VLAN tag or 12-bit VLAN ID (based on MAC_VLAN_Tag_Ctrl[ETV]) in the incoming packet is passed through the CRC logic. The upper four bits of the calculated CRC value are used to index the contents of the VLAN hash table. For example, hash value of 1000b selects bit 8 of the VLAN hash table.

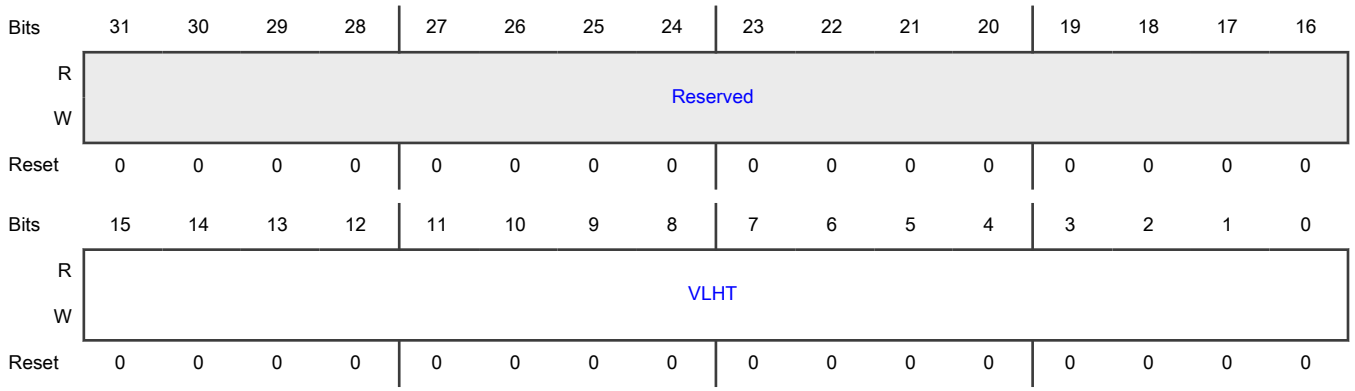
Perform these steps to calculate the hash value of the destination address:

1. Calculate the 32-bit CRC for DA (see section 3.2.8 of IEEE 802.3 for steps to calculate CRC32).
2. Perform bit-wise reversal for the value obtained in step 1.
3. Take the upper 4 bits from the value obtained in step 2.

If this register is configured to be double-synchronized with the (G)MII clock domain, the synchronization is triggered only when bits [15:8] (in Little-Endian mode) or bits [7:0] (in Big-Endian mode) of the register are written to.

If double-synchronization is enabled, consecutive writes to this register must be performed after at least four clock cycles in the destination clock domain.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>VLHT | VLAN Hash Table<br>Contains the 16-bit VLAN hash table. |

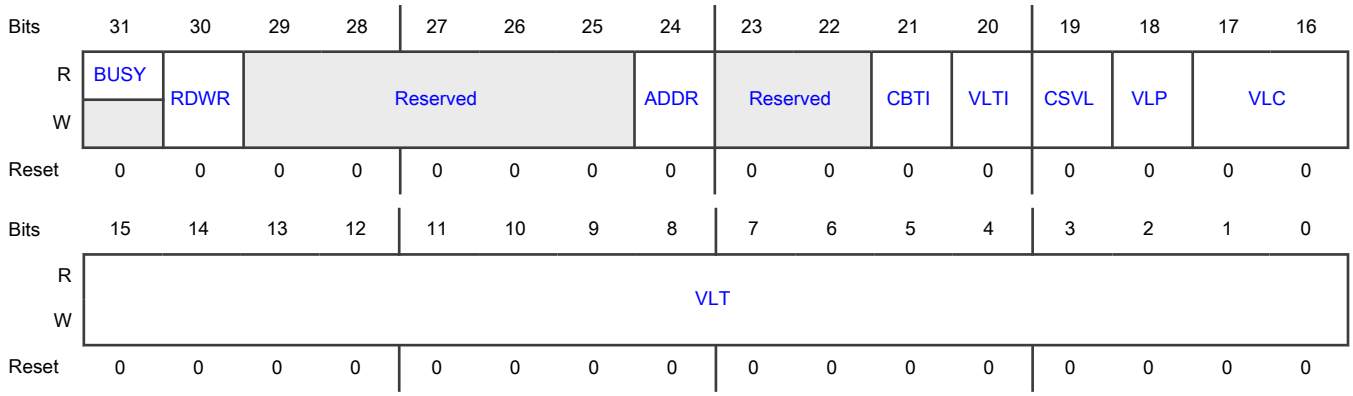
## 72.17.16 MAC VLAN Inclusion Or Replacement (MAC_VLAN_Incl)

### Offset

| Register | Offset |
|---|---|
| MAC_VLAN_Incl | 60h |

### Function
Contains the VLAN tag for insertion or replacement in the transmit packets and the VLAN tag insertion controls.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>BUSY | Busy<br>Indicates the status of the read or write operation of indirect access to the queue- or channel-specific VLAN inclusion register.<br>The write operation to a register completes when this field becomes 0. For a read operation, the read data is valid when the field becomes 0.<br>The application must make sure that this field becomes 0 before attempting subsequent accesses to this register. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Busy status not detected<br>1b - Busy status detected |
| 30<br>RDWR | Read Write Control<br>Controls the read or write operation for indirectly accessing the queue- or channel-specific VLAN inclusion register.<br>• If this field is 1, it indicates the write operation.<br>• If this field is 0, it indicates the read operation.<br>The value of this field is not impacted when the CBTI field becomes 0.<br>0b - Read operation of indirect access<br>1b - Write operation of indirect access |
| 29-25<br>— | Reserved |
| 24<br>ADDR | Address<br>Selects one of the queue- or channel-specific VLAN inclusion registers for read or write access.<br>The value of this field is not impacted when the CBTI field is 0. |
| 23-22<br>— | Reserved |
| 21<br>CBTI | Channel-Based Tag Insertion<br>Indicates the status of channel-based tag insertion.<br>If this field is 1:<br>• The outer VLAN tag is inserted for all the packets that MAC transmits.<br>• The tag value is taken from the queue- or channel-specific VLAN tag register. Also, the VLTI, VLP, VLC, and VLT fields of the register are ignored.<br>• A write operation to byte 3 of this register initiates the read or write access to the indirect register.<br>If this field is 0, the outer VLAN operation is based on the settings of the VLTI, VLP, VLC, and VLT fields of this register.<br>0b - Disabled<br>1b - Enabled |
| 20<br>VLTI | VLAN Tag Input<br>Indicates the status of the VLAN tag input.<br>The value of this field being 1 indicates that the VLAN tag to be inserted or replaced in the transmit packet must be taken from the transmit descriptor. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |
| 19<br>CSVL | C-VLAN Or S-VLAN<br>Indicates whether C-VLAN or S-VLAN is inserted or replaced in the transmitted packets.<br>• If this field is 1, S-VLAN type (88A8h) is inserted or replaced in the 13th and 14th bytes of the transmitted packets.<br>• If this field is 0, C-VLAN type (8100h) is inserted or replaced in the 13th and 14th bytes of the transmitted packets.<br>0b - C-VLAN<br>1b - S-VLAN |
| 18<br>VLP | VLAN Priority Control<br>Indicates the status of VLAN priority control.<br>• If this field is 1, the control bits [17:16] are used for VLAN deletion, insertion, or replacement.<br>• If this field is 0, the mti_vlan_ctrl_i control input is used and bits [17:16] are ignored.<br>0b - Disabled<br>1b - Enabled |
| 17-16<br>VLC | VLAN Tag Control<br>Contains values for VLAN tag control in transmit packets.<br>MAC:<br>• Removes the VLAN type (bytes 13 and 14) and VLAN tag (bytes 15 and 16) of all the transmitted packets with VLAN tags.<br>• Replaces VLT in bytes 15 and 16 of all the VLAN-type transmitted packets (bytes 13 and 14 are 8100h or 88a8h).<br>MAC inserts the following into the packet in the order shown:<br>1. Type value (8100h or 88A8h) into bytes 13 and 14<br>2. MAC_VLAN_Tag_Data[VLT] into bytes 15 and 16<br>This operation is performed on all the transmitted packets, irrespective of whether they already have a VLAN tag.<br>**NOTE**<br>Changes to this field take effect only on the start of a packet. If you write to this field when a packet is being transmitted, only the subsequent packets can use the updated value. That is, the current packet does not use the updated value.<br>00b - No VLAN tag deletion, insertion, or replacement<br>01b - VLAN tag deletion |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 10b - VLAN tag insertion<br>11b - VLAN tag replacement |
| 15-0<br>VLT | VLAN Tag For Transmit Packets<br>Contains the value of the VLAN tag to be inserted or replaced. This value must only be changed during the initialization phase or when the transmit lines are inactive.<br>The following list describes the bits of this field:<br>• Bits [15:13]: User priority<br>• Bit 12: Canonical format indicator (CFI) or drop eligible indicator (DEI)<br>• Bits [11:0]: MAC_VLAN_Tag_Data[VID] |

## 72.17.17 MAC VLAN Inclusion 0 (MAC_VLAN_Incl0)

### Offset

| Register | Offset |
|---|---|
| MAC_VLAN_Incl0 | 60h |

### Function

Contains the VLAN tag for insertion in the transmit packets from transmit queue 0. The register also contains the VLAN tag insertion controls.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | CSVL | Reserved | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | VLT | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-20 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 19<br>CSVL | C-VLAN or S-VLAN<br>When this field is 1, S-VLAN type (0x88A8) is inserted in the 13th and 14th bytes of transmitted packets. When this field is reset, C-VLAN type (0x8100) is inserted in the 13th and 14th bytes of transmitted packets.<br>0b - C-VLAN type (0x8100) is inserted<br>1b - S-VLAN type (0x88A8) is inserted |
| 18-16<br>— | Reserved |
| 15-0<br>VLT | VLAN Tag for Transmit Packets<br>Contains the value of the VLAN tag to be inserted.<br>The value of this field must only be changed when the transmit lines are inactive or during the initialization phase.<br>The following list describes the bits of this field:<br>• Bits[15:13]: User Priority<br>• Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)<br>• Bits[11:0]: VLAN Identifier (VID) field of VLAN tag |

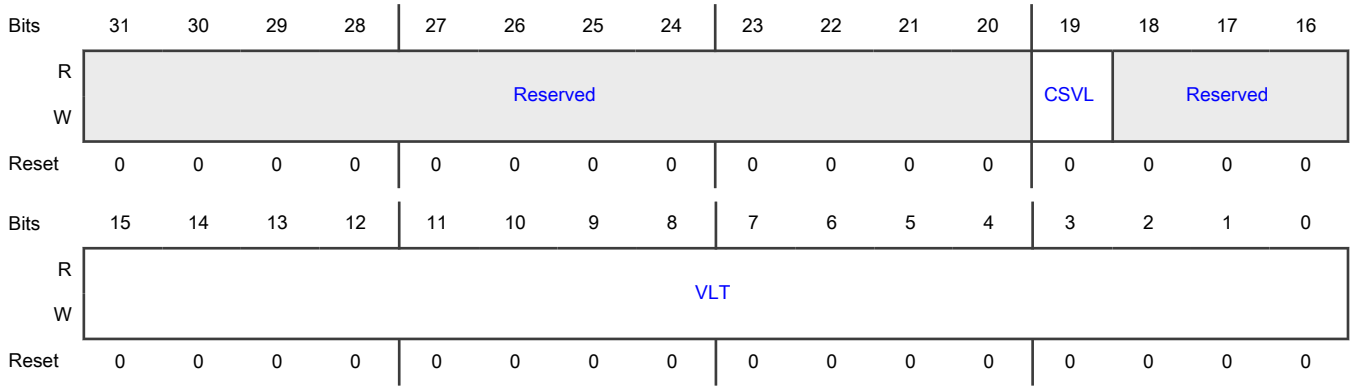
### 72.17.18 MAC VLAN Inclusion 1 (MAC_VLAN_Incl1)

**Offset**

| Register | Offset |
|---|---|
| MAC_VLAN_Incl1 | 60h |

**Function**
Contains the VLAN tag for insertion in the transmit packets from transmit queue 1. The register also contains the VLAN tag insertion controls.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19<br>CSVL | C-VLAN or S-VLAN<br>When this field is 1, S-VLAN type (0x88A8) is inserted in the 13th and 14th bytes of transmitted packets. When this field is reset, C-VLAN type (0x8100) is inserted in the 13th and 14th bytes of transmitted packets.<br>0b - C-VLAN type (0x8100) is inserted<br>1b - S-VLAN type (0x88A8) is inserted |
| 18-16<br>— | Reserved |
| 15-0<br>VLT | VLAN Tag for Transmit Packets<br>Contains the value of the VLAN tag to be inserted.<br>The value of this field must only be changed when the transmit lines are inactive or during the initialization phase.<br>The following list describes the bits of this field:<br>• Bits[15:13]: User Priority<br>• Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)<br>• Bits[11:0]: VLAN Identifier (VID) field of VLAN tag |

## 72.17.19 MAC VLAN Inclusion 2 (MAC_VLAN_Incl2)

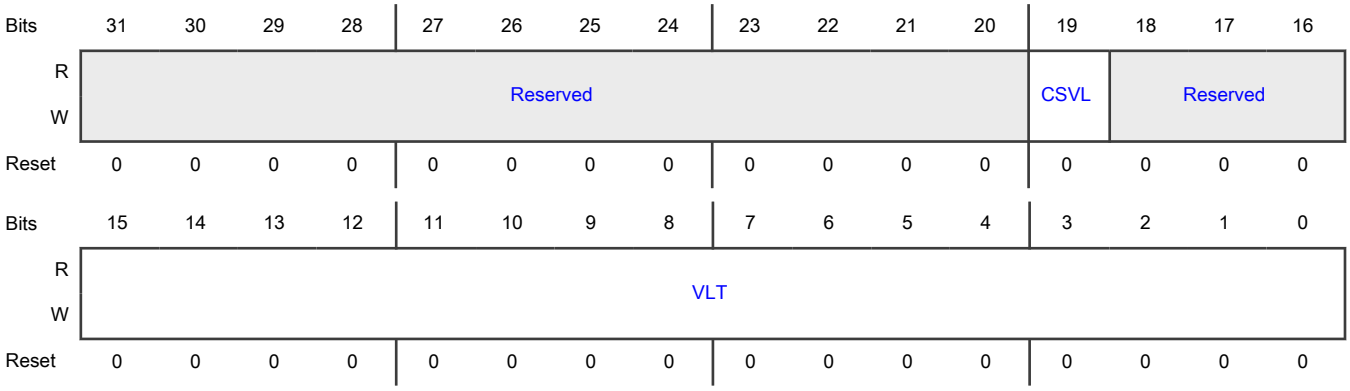
### Offset

| Register | Offset |
|---|---|
| MAC_VLAN_Incl2 | 60h |

### Function

Contains the VLAN tag for insertion in the transmit packets from transmit queue 2. The register also contains the VLAN tag insertion controls.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19<br>CSVL | C-VLAN or S-VLAN<br>When this field is 1, S-VLAN type (0x88A8) is inserted in the 13th and 14th bytes of transmitted packets. When this field is reset, C-VLAN type (0x8100) is inserted in the 13th and 14th bytes of transmitted packets.<br>0b - C-VLAN type (0x8100) is inserted<br>1b - S-VLAN type (0x88A8) is inserted |
| 18-16<br>— | Reserved |
| 15-0<br>VLT | VLAN Tag for Transmit Packets<br>Contains the value of the VLAN tag to be inserted.<br>The value of this field must only be changed when the transmit lines are inactive or during the initialization phase.<br>The following list describes the bits of this field:<br>• Bits[15:13]: User Priority<br>• Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)<br>• Bits[11:0]: VLAN Identifier (VID) field of VLAN tag |

## 72.17.20 MAC VLAN Inclusion 3 (MAC_VLAN_Incl3)

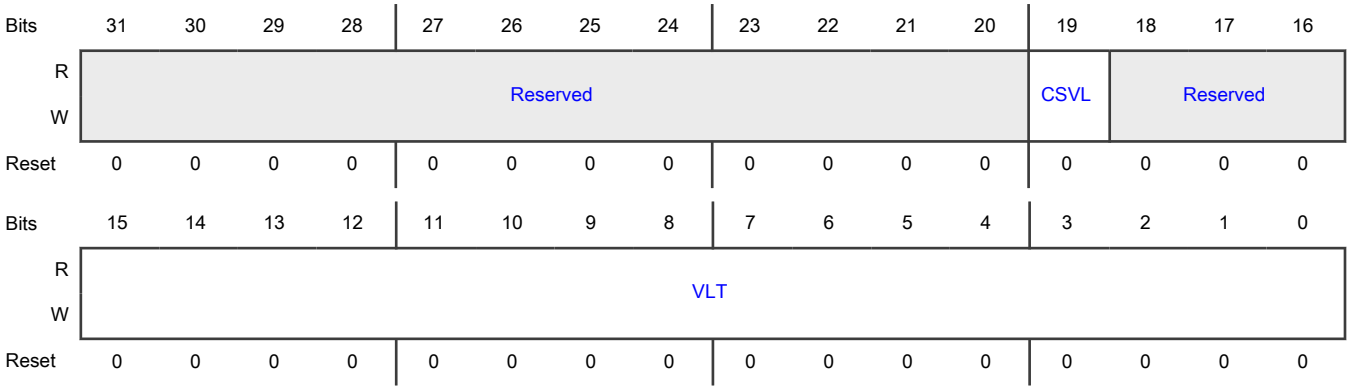
### Offset

| Register | Offset |
|---|---|
| MAC_VLAN_Incl3 | 60h |

### Function

Contains the VLAN tag for insertion in the transmit packets from transmit queue 3. The register also contains the VLAN tag insertion controls.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19<br>CSVL | C-VLAN or S-VLAN<br>When this field is 1, S-VLAN type (0x88A8) is inserted in the 13th and 14th bytes of transmitted packets. When this field is reset, C-VLAN type (0x8100) is inserted in the 13th and 14th bytes of transmitted packets.<br>0b - C-VLAN type (0x8100) is inserted<br>1b - S-VLAN type (0x88A8) is inserted |
| 18-16<br>— | Reserved |
| 15-0<br>VLT | VLAN Tag for Transmit Packets<br>Contains the value of the VLAN tag to be inserted.<br>The value of this field must only be changed when the transmit lines are inactive or during the initialization phase.<br>The following list describes the bits of this field: |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • Bits[15:13]: User Priority<br>• Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)<br>• Bits[11:0]: VLAN Identifier (VID) field of VLAN tag |

## 72.17.21 MAC VLAN Inclusion 4 (MAC_VLAN_Incl4)

### Offset

| Register | Offset |
|---|---|
| MAC_VLAN_Incl4 | 60h |

### Function

Contains the VLAN tag for insertion in the transmit packets from transmit queue 4. The register also contains the VLAN tag insertion controls.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | CSVL | Reserved | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | VLT | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19<br>CSVL | C-VLAN or S-VLAN<br>When this field is 1, S-VLAN type (0x88A8) is inserted in the 13th and 14th bytes of transmitted packets. When this field is reset, C-VLAN type (0x8100) is inserted in the 13th and 14th bytes of transmitted packets.<br>0b - C-VLAN type (0x8100) is inserted<br>1b - S-VLAN type (0x88A8) is inserted |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 18-16<br>— | Reserved |
| 15-0<br>VLT | VLAN Tag for Transmit Packets<br>Contains the value of the VLAN tag to be inserted.<br>The value of this field must only be changed when the transmit lines are inactive or during the initialization phase.<br>The following list describes the bits of this field:<br>• Bits[15:13]: User Priority<br>• Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)<br>• Bits[11:0]: VLAN Identifier (VID) field of VLAN tag |

## 72.17.22 MAC VLAN Inclusion 5 (MAC_VLAN_Incl5)

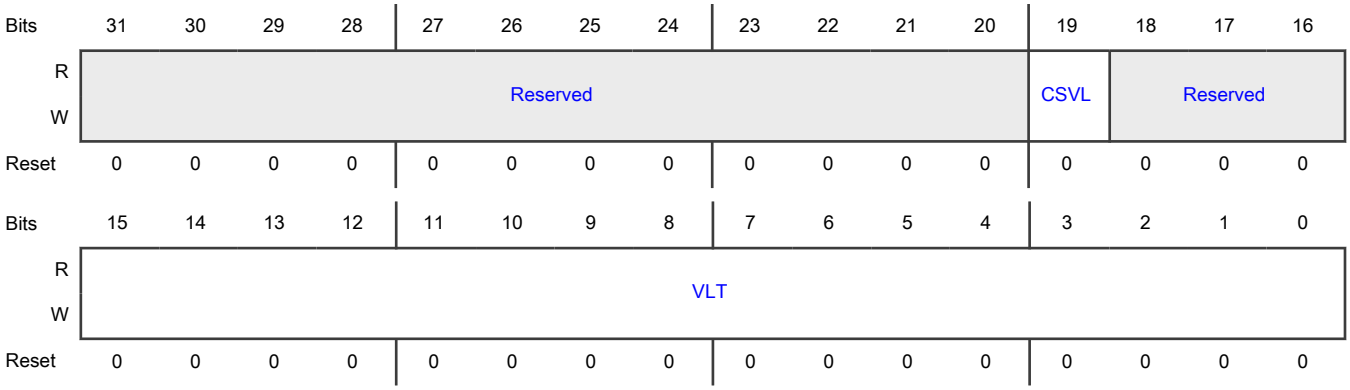
### Offset

| Register | Offset |
|---|---|
| MAC_VLAN_Incl5 | 60h |

### Function

Contains the VLAN tag for insertion in the transmit packets from transmit queue 5. The register also contains the VLAN tag insertion controls.

### Diagram



**Fields**

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19<br>CSVL | C-VLAN or S-VLAN<br>When this field is 1, S-VLAN type (0x88A8) is inserted in the 13th and 14th bytes of transmitted packets. When this field is reset, C-VLAN type (0x8100) is inserted in the 13th and 14th bytes of transmitted packets.<br>0b - C-VLAN type (0x8100) is inserted<br>1b - S-VLAN type (0x88A8) is inserted |
| 18-16<br>— | Reserved |
| 15-0<br>VLT | VLAN Tag for Transmit Packets<br>Contains the value of the VLAN tag to be inserted.<br>The value of this field must only be changed when the transmit lines are inactive or during the initialization phase.<br>The following list describes the bits of this field:<br>• Bits[15:13]: User Priority<br>• Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)<br>• Bits[11:0]: VLAN Identifier (VID) field of VLAN tag |

## 72.17.23 MAC VLAN Inclusion 6 (MAC_VLAN_Incl6)

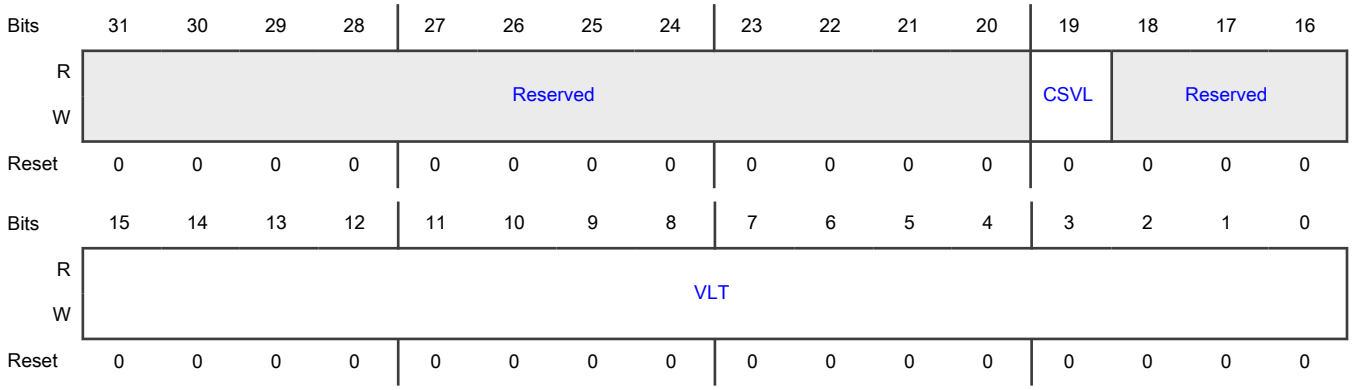
**Offset**

| Register | Offset |
|---|---|
| MAC_VLAN_Incl6 | 60h |

**Function**

Contains the VLAN tag for insertion in the transmit packets from transmit queue 6. The register also contains the VLAN tag insertion controls.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19<br>CSVL | C-VLAN or S-VLAN<br>When this field is 1, S-VLAN type (0x88A8) is inserted in the 13th and 14th bytes of transmitted packets. When this field is reset, C-VLAN type (0x8100) is inserted in the 13th and 14th bytes of transmitted packets.<br>0b - C-VLAN type (0x8100) is inserted<br>1b - S-VLAN type (0x88A8) is inserted |
| 18-16<br>— | Reserved |
| 15-0<br>VLT | VLAN Tag for Transmit Packets<br>Contains the value of the VLAN tag to be inserted.<br>The value of this field must only be changed when the transmit lines are inactive or during the initialization phase.<br>The following list describes the bits of this field:<br>• Bits[15:13]: User Priority<br>• Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)<br>• Bits[11:0]: VLAN Identifier (VID) field of VLAN tag |

## 72.17.24 MAC VLAN Inclusion 7 (MAC_VLAN_Incl7)

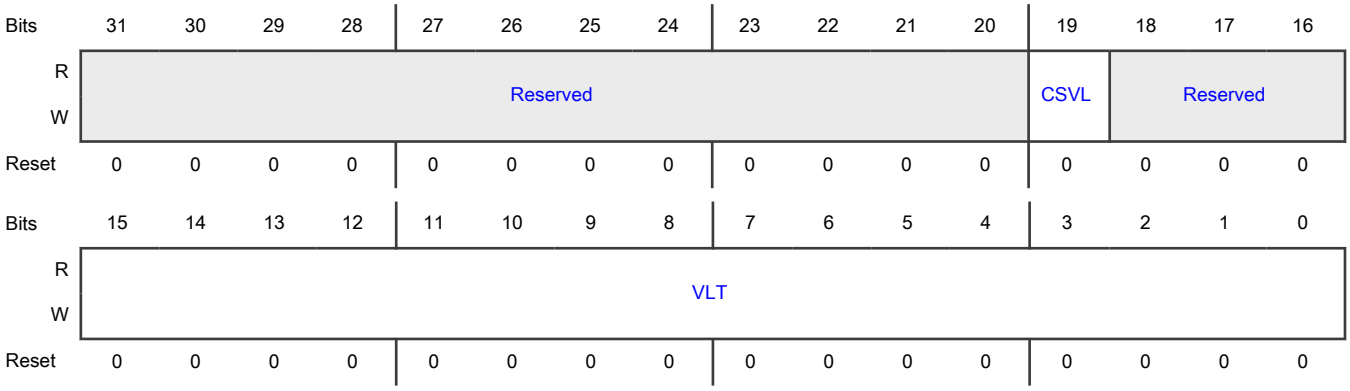
### Offset

| Register | Offset |
|---|---|
| MAC_VLAN_Incl7 | 60h |

### Function

Contains the VLAN tag for insertion in the transmit packets from transmit queue 7. The register also contains the VLAN tag insertion controls.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19<br>CSVL | C-VLAN or S-VLAN<br>When this field is 1, S-VLAN type (0x88A8) is inserted in the 13th and 14th bytes of transmitted packets. When this field is reset, C-VLAN type (0x8100) is inserted in the 13th and 14th bytes of transmitted packets.<br>0b - C-VLAN type (0x8100) is inserted<br>1b - S-VLAN type (0x88A8) is inserted |
| 18-16<br>— | Reserved |
| 15-0<br>VLT | VLAN Tag for Transmit Packets<br>Contains the value of the VLAN tag to be inserted.<br>The value of this field must only be changed when the transmit lines are inactive or during the initialization phase.<br>The following list describes the bits of this field:<br>• Bits[15:13]: User Priority<br>• Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)<br>• Bits[11:0]: VLAN Identifier (VID) field of VLAN tag |

## 72.17.25 Inner VLAN Tag Inclusion Or Replacement (MAC_Inner_VLAN_Incl)

### Offset

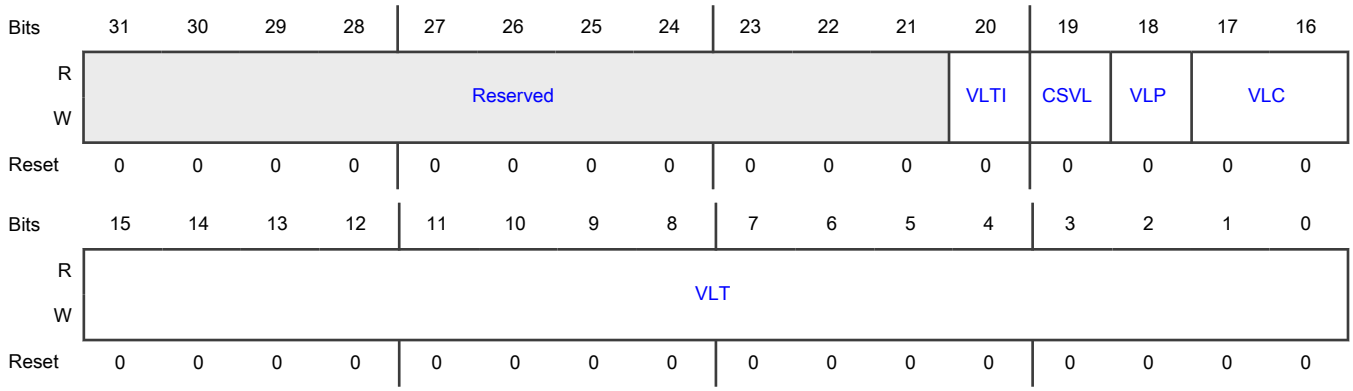
| Register | Offset |
|---|---|
| MAC_Inner_VLAN_Incl | 64h |

### Function

Contains:

- The inner VLAN tag to be inserted or replaced in the transmit packet
- The inner VLAN tag insertion controls

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-21<br>— | Reserved |
| 20<br>VLTI | VLAN Tag Input<br>Indicates the status of VLAN tag input.<br>If this field is 1, it indicates that the VLAN tag to be inserted or replaced in the transmit packet must be taken from the transmit descriptor.<br>0b - Disabled<br>1b - Enabled |
| 19<br>CSVL | C-VLAN Or S-VLAN<br>Controls the insertion or replacement type in the 17th and 18th bytes of the transmitted packets.<br>0b - C-VLAN type (8100h)<br>1b - S-VLAN type (88A8h) |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 18<br>VLP | VLAN Priority Control<br>Indicates the status of VLAN priority control.<br>• If this field is 1, it's used for VLAN deletion, insertion, or replacement.<br>• If this field is 0, the mti_vlan_ctrl_i control input is used and MAC_VLAN_Incl[VLC] is ignored.<br>0b - Disabled<br>1b - Enabled |
| 17-16<br>VLC | VLAN Tag Control in Transmit Packets<br>Indicates the value of VLAN tag control in transmit packets.<br>The following list specifies these values:<br>• 2'b00: No VLAN tag deletion, insertion, or replacement<br>• 2'b01: VLAN tag deletion<br>• 2'b10: VLAN tag insertion<br>• 2'b11: VLAN tag replacement<br>MAC:<br>• Removes the VLAN type (bytes 17 and 18) and VLAN tag (bytes 19 and 20) of all the transmitted packets with VLAN tags<br>• Inserts VLT in bytes 19 and 20 of the packet after inserting the Type value (8100h or 88a8h) in bytes 17 and 18. This operation is performed on all the transmitted packets, irrespective of whether they already have a VLAN tag.<br>• Replaces VLT in bytes 19 and 20 for all the VLAN-type transmitted packets (bytes 17 and 18 are 8100h or 88a8h).<br>**NOTE**<br>Changes to this field take effect only on the start of a packet. If you write to the field when a packet is being transmitted, only the subsequent packets can use the updated value. That is, the current packet does not use the updated value.<br>00b - No VLAN tag deletion, insertion, or replacement<br>01b - VLAN tag deletion<br>10b - VLAN tag insertion<br>11b - VLAN tag replacement |
| 15-0<br>VLT | VLAN Tag For Transmit Packets<br>Contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase.<br>The following list describes the bits of this field:<br>• Bits [15:13]: User priority |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • Bit 12: CFI or DEI<br>• Bits [11:0]: MAC_VLAN_Tag_Data[VID] |

## 72.17.26 MAC Q0 Tx Flow Control (MAC_Q0_Tx_Flow_Ctrl)

### Offset

| Register | Offset |
|---|---|
| MAC_Q0_Tx_Flow_Ctrl | 70h |

### Function

Controls the generation and reception of the control (pause command) packets by MAC's flow control module. A write to the register with the Busy field = 1 triggers the flow control block to generate a pause packet. The 802.3x specification indicates the way the fields of the control packet are selected, and the pause time value from this register is used in the PT field of the control packet. The Busy field remains 1 until the control packet is transferred onto the cable. You must make sure that the Busy field is 0 before writing to this register.

If the value of the PFCE field in the MAC_Rx_Flow_Ctrl register is 1, this register controls the generation of priority flow control (PFC) frames with priorities mapped according to MAC_RxQ_Ctrl2[PSRQ0].

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PT | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | DZPQ | PLT | | | Reserved | | TFE | FCB_<br>BPA |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-16<br>PT | Pause Time<br>Holds the value to be used in this field in the transmit control packet. If the bits of this field are configured to be double-synchronized with the (G)MII clock domain, consecutive writes to this register must be performed only after at least four clock cycles in the destination clock domain. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 15-8<br>— | Reserved |
| 7<br>DZPQ | Disable Zero-Quanta Pause<br>Enables or disables zero-quanta pause packet generation.<br>• If this field is 1, it disables the automatic generation of the zero-quanta pause packets that the flow-control signal from the FIFO layer (MTL or external sideband flow control signal sbd_flowctrl_i or mti_flowctrl_i) deasserts.<br>• If this field is 0, normal operations with automatic zero-quanta pause packet generation are enabled.<br>0b - Enabled<br>1b - Disabled |
| 6-4<br>PLT | Pause Low Threshold<br>Configures the threshold of the pause timer at which the input flow control signal mti_flowctrl_i (or sbd_flowctrl_i) is checked for automatic retransmission of the pause packet.<br>The threshold values must always be less than the pause time configured in bits [31:16]. For example, if the PT field = 100H (256 slot times), and the PLT field = 001, a second pause packet automatically transmits if the mti_flowctrl_i signal is asserted at 228 (256-28) slot times after the first pause packet transmits.<br>This list provides the threshold values for different values:<br>• The slot time is defined as the time taken to transmit 512 bits (64 bytes) on GMII or MII.<br>• This (approximate) computation is based on the packet size (64, 1518, 2000, 9018, 16384, or 32768) + two pause packet sizes + IPG in slot times.<br>000b - Pause time minus 4 slot times (PT is 4 slot times)<br>001b - Pause time minus 28 slot times (PT is 28 slot times)<br>010b - Pause time minus 36 slot times (PT is 36 slot times)<br>011b - Pause time minus 144 slot times (PT is 144 slot times)<br>100b - Pause time minus 256 slot times (PT is 256 slot times)<br>101b - Pause time minus 512 slot times (PT is 512 slot times)<br>110b - Reserved |
| 3-2<br>— | Reserved |
| 1<br>TFE | Transmit Flow Control Enable<br>Indicates whether transmit flow control is enabled or disabled.<br>• Full-Duplex mode: If this field is 1, MAC enables the flow control operation to the transmit pause packets. When the field becomes 0, MAC's flow control operation disables and MAC does not transmit any pause packets. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • Half-Duplex mode: If this field is 1, MAC enables the backpressure operation. When the field becomes 0, the backpressure operation disables.<br>0b - Disabled<br>1b - Enabled |
| 0<br>FCB_BPA | Flow Control Busy Or Backpressure Activate<br>Initiates if a pause packet is in Full-Duplex mode and activates the backpressure function in Half-Duplex mode if the TFE field is 1.<br>• Full-Duplex mode: In this mode, FCB_BPA must be read as 1'b0 before writing to this register. To initiate a pause packet, the application must set this field to 1'b1. During control packet transfer, the field's value continues to be 1 to indicate that a packet transmission is in progress. When pause packet transmission is complete, MAC resets this field to 1'b0. You must not write to this register until this field becomes 0.<br>• Half-Duplex mode: If FCB_BPA = 1 and TFE = 1 too, MAC asserts backpressure in this mode. During the backpressure function, when MAC receives a new packet, the transmitter starts sending a JAM pattern resulting in a collision. This control register field is logically ORed with the mti_flowctrl_i input signal for the backpressure function. When you configure MAC using Full-Duplex mode, this field automatically becomes 0.<br>Access restrictions apply to this field: writing 1 sets it, writing 0 has no effect, and the field is self-clearing.<br>0b - Flow control busy or backpressure activate is disabled<br>1b - Flow control busy or backpressure activate is enabled |

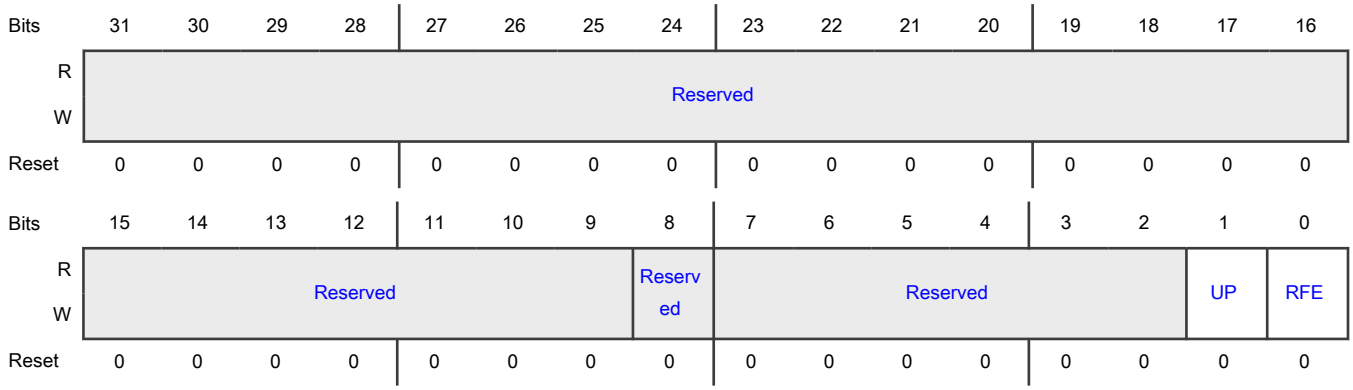
### 72.17.27 MAC Receive Flow Control (MAC_Rx_Flow_Ctrl)

**Offset**

| Register | Offset |
|---|---|
| MAC_Rx_Flow_Ctrl | 90h |

**Function**
Controls the pausing of MAC transmit based on the received pause packet.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>— | Reserved |
| 7-2<br>— | Reserved |
| 1<br>UP | Unicast Pause Packet Detect<br>Indicates whether the unicast pause packet is enabled or disabled.<br>A pause packet is processed when it has the unique multicast address specified in IEEE 802.3.<br>• If this field is 1, MAC can detect pause packets with a unicast address of the station. This address must comply with the specifications in MAC Address 0 High (MAC_Address0_High) and MAC Address 0 Low (MAC_Address0_Low).<br>• If this field is 0, MAC only detects pause packets with a unique multicast address.<br>**NOTE**<br>MAC does not process a pause packet if the multicast address is different from the unique multicast address. This applies to the received PFC packet if PFC is enabled. The unique multicast address (01_80_C2_00_00_01h) complies with the IEEE 802.1 Qbb-2011 specifications.<br>0b - Disabled<br>1b - Enabled |
| 0<br>RFE | Receive Flow Control Enable<br>Indicates whether the receive flow control is enabled or disabled.<br>• If this field is 1 and MAC is operating in Full-Duplex mode, MAC decodes the received pause packet and disables its transmitter for a specified (pause) time. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • If this field is 0 or if MAC is operating in Half-Duplex mode, the decode function of the pause packet disables.<br>When PFC is enabled, the flow control for PFC packets is enabled too. MAC decodes the received PFC packet and disables the transmit queue, with matching priorities, for a duration of the received pause time.<br>0b - Disabled<br>1b - Enabled |

## 72.17.28 MAC RxQ Control 4 (MAC_RxQ_Ctrl4)

### Offset

| Register | Offset |
|---|---|
| MAC_RxQ_Ctrl4 | 94h |

### Function

Controls the routing of unicast and multicast packets that fail the destination or source address filter to the receive queues.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | VFFQ | VFFQE |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | MFFQ | MFFQE | Reserved | | | | | | UFFQ | UFFQE |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-18<br>— | Reserved |
| 17<br>VFFQ | VLAN Tag Filter Fail Packets Queue<br>Holds the receive queue number to which the tagged packets failing the destination or source address filter (and UFFQE and MFFQE not being 1) or failing the VLAN tag filter must be routed to. This field is valid only when the VFFQE field = 1. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 16<br>VFFQE | VLAN Tag Filter Fail Packets Queuing Enable<br>Indicates whether VLAN tag filter fail packet queuing is enabled or disabled.<br>• If this field is 1, the tagged packets that fail the destination or source address filter or fail the VLAN tag filter are routed to the receive queue number programmed using the VFFQ field.<br>• If this field is 0, the tagged packets that fail the destination or source address filter or fail the VLAN tag filter are routed based on the other routing options. This field is valid only when MAC_Packet_Filter[RA] = 1.<br>0b - Disabled<br>1b - Enabled |
| 15-10<br>— | Reserved |
| 9<br>MFFQ | Multicast Address Filter Fail Packets Queue<br>Holds the receive queue number to which the multicast packets failing the destination or source address filter are routed to. This field is valid only when the MFFQE field = 1. |
| 8<br>MFFQE | Multicast Address Filter Fail Packets Queuing Enable<br>Indicates whether multicast address filter fail packet queuing is enabled or disabled.<br>• If this field is 1, the multicast packets that fail the destination or source address filter are routed to the receive queue number programmed using the MFFQ field.<br>• If this field is 0, the multicast packets that fail the destination or source address filter are routed based on the other routing options.<br>This field is valid only when MAC_Packet_Filter[RA] = 1.<br>0b - Disabled<br>1b - Enabled |
| 7-2<br>— | Reserved |
| 1<br>UFFQ | Unicast Address Filter Fail Packets Queue<br>Holds the receive queue number to which the unicast packets failing the destination or source address filter are routed to. This field is valid only when the UFFQE field = 1. |
| 0<br>UFFQE | Unicast Address Filter Fail Packets Queuing Enable<br>Indicates whether unicast address filter fail packet queuing is enabled or disabled.<br>• When the value of this field is 1, the unicast packets that fail the destination or source address filter are routed to the receive queue number programmed using the UFFQ field.<br>• If this field is 0, the unicast packets that fail the destination or source address filter are routed based on the other routing options. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | This field is valid only when MAC_Packet_Filter[RA] = 1.<br>0b - Disabled<br>1b - Enabled |

## 72.17.29 MAC RxQ Control 0 (MAC_RxQ_Ctrl0)

### Offset

| Register | Offset |
|---|---|
| MAC_RxQ_Ctrl0 | A0h |

### Function

Controls queue management in the MAC receiver.

**NOTE**

In multiple receive queues configuration, all the queues are disabled by default. Enable the receive queue by programming the corresponding field in this register.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | Reserved | | Reserved | | Reserved | | Reserved | | Reserved | | RXQ1EN | | RXQ0EN | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-14<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 13-12<br>— | Reserved |
| 11-10<br>— | Reserved |
| 9-8<br>— | Reserved |
| 7-6<br>— | Reserved |
| 5-4<br>— | Reserved |
| 3-2<br>RXQ1EN | Receive Queue 1 Enable<br>Performs in a manner similar to the RXQ0EN field and indicates if receive queue 1 is enabled for AV or DCB.<br>00b - Queue not enabled<br>01b - Queue enabled for AV<br>10b - Queue enabled for DCB/generic<br>11b - Reserved |
| 1-0<br>RXQ0EN | Receive Queue 0 Enable<br>Indicates whether receive queue 0 is enabled for AV or DCB.<br>00b - Queue not enabled<br>01b - Queue enabled for AV<br>10b - Queue enabled for DCB/generic<br>11b - Reserved |

## 72.17.30 Receive Queue Control 1 (MAC_RxQ_Ctrl1)

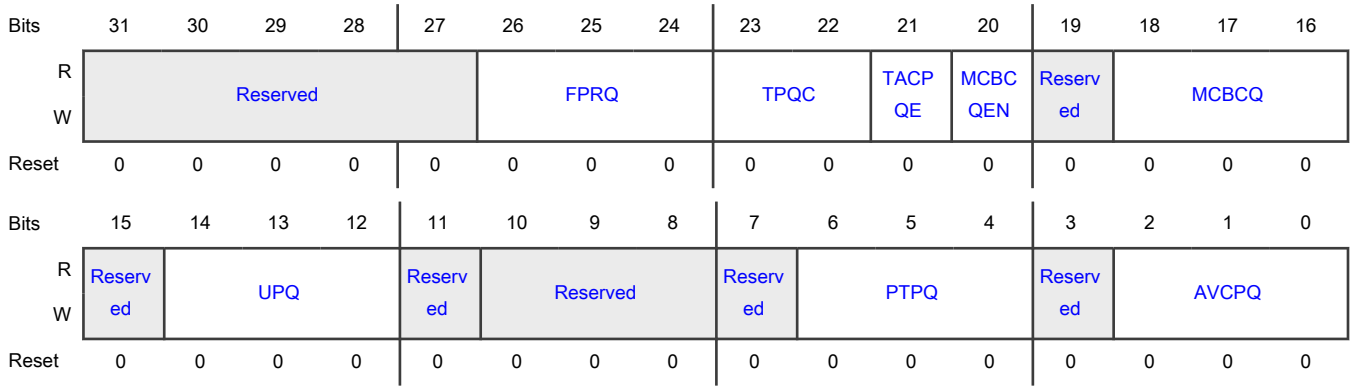
### Offset

| Register | Offset |
|---|---|
| MAC_RxQ_Ctrl1 | A4h |

### Function

Controls the routing of multicast, broadcast, AV, DCB, and untagged packets to the receive queues.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26-24<br>FPRQ | Frame Preemption Residue Queue<br>Holds the receive queue number to which the residual preemption frames must be forwarded.<br>Preemption frames that are tagged and pass the SA, DA, or VLAN filtering are routed based on the settings of the PSRQ fields in MAC RxQ Control 2 (MAC_RxQ_Ctrl2). All other frames are treated as residual frames and are routed to the queue number mentioned in this field. Queue 0 is used as the default queue for express frames, so you cannot write 0 to this field. |
| 23-22<br>TPQC | Tagged PTP Over Ethernet Packets Queuing Control<br>Controls the routing of the VLAN-tagged PTPoE packets.<br>These are the allowed programmable options:<br>• 2'b00: VLAN-tagged PTPoE packets are routed as generic VLAN-tagged packets (based on the PSRQ fields in MAC RxQ Control 2 (MAC_RxQ_Ctrl2) for only non-AV enabled receive queues).<br>• 2'b01: VLAN-tagged PTPoE packets are routed to the receive queue that the PTPQ field of this register specifies (this Rx queue can be enabled for AV or non-AV traffic).<br>• 2'b10: VLAN-tagged PTPoE packets are routed to only AV-enabled receive queues based on the settings of the PSRQ fields. |
| 21<br>TACPQE | Tagged AV Control Packets Queuing Enable<br>Indicates the status of tagged AV control packet queuing.<br>• If this field is 1, MAC routes the received tagged AV control packets to the receive queue that the AVCPQ field specifies.<br>• If this field is 0, MAC routes the received tagged AV control packets based on the tag priority matching the PSRQ fields in MAC RxQ Control 2 (MAC_RxQ_Ctrl2).<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 20<br>MCBCQEN | Multicast And Broadcast Queue Enable<br>Specifies whether multicast or broadcast packet routing to the receive queue is enabled or disabled. The multicast or broadcast packets must be routed to the receive queue specified in the MCBCQ field.<br>0b - Disabled<br>1b - Enabled |
| 19<br>— | Reserved |
| 18-16<br>MCBCQ | Multicast And Broadcast Queue<br>Specifies the receive queue onto which multicast or broadcast packets are routed. Any receive queue enabled for Generic, DCB, or AV traffic can be used to route the multicast or broadcast packets.<br>000b - Receive queue 0<br>001b - Receive queue 1<br>010b - Receive queue 2<br>011b - Receive queue 3<br>100b - Receive queue 4<br>101b - Receive queue 5<br>110b - Receive queue 6<br>111b - Receive queue 7 |
| 15<br>— | Reserved |
| 14-12<br>UPQ | Untagged Packet Queue<br>Indicates the receive queue to which the untagged packets need to be routed. Any receive queue enabled for generic, DCB, or AV traffic can be used to route the untagged packets.<br>000b - Receive queue 0<br>001b - Receive queue 1<br>010b - Receive queue 2<br>011b - Receive queue 3<br>100b - Receive queue 4<br>101b - Receive queue 5<br>110b - Receive queue 6<br>111b - Receive queue 7 |
| 11<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 10-8<br>— | Reserved |
| 7<br>— | Reserved |
| 6-4<br>PTPQ | PTP Packets Queue<br>Specifies the receive queue on which the PTP packets sent over the Ethernet payload (not over IPv4 or IPv6) are routed.<br>When MAC_Timestamp_Control[AV8021ASMEN] = 1, only untagged PTP over Ethernet packets are routed to a receive queue. If this field is not 1, then based on the settings of the TPQC field, both tagged and untagged PTPoE packets can be routed to this receive queue.<br>000b - Receive queue 0<br>001b - Receive queue 1<br>010b - Receive queue 2<br>011b - Receive queue 3<br>100b - Receive queue 4<br>101b - Receive queue 5<br>110b - Receive queue 6<br>111b - Receive queue 7 |
| 3<br>— | Reserved |
| 2-0<br>AVCPQ | AV Untagged Control Packets Queue<br>Specifies the receive queue on which the received AV tagged and untagged control packets are routed.<br>The AV tagged (when the TACPQE field = 1) and untagged control packets are routed to the receive queue that this field specifies.<br>000b - Receive queue 0<br>001b - Receive queue 1<br>010b - Receive queue 2<br>011b - Receive queue 3<br>100b - Receive queue 4<br>101b - Receive queue 5<br>110b - Receive queue 6<br>111b - Receive queue 7 |

## 72.17.31 MAC RxQ Control 2 (MAC_RxQ_Ctrl2)

### Offset

| Register | Offset |
|---|---|
| MAC_RxQ_Ctrl2 | A8h |

### Function

Controls the routing of tagged packets based on the settings of MAC_Ext_Configuration[USP] for packets received in receive queues 0-3.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | Reserved | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PSRQ1 | | | | | | | | PSRQ0 | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16<br>— | Reserved |
| 15-8<br>PSRQ1 | Priorities Selected In Receive Queue 1<br>Determines the priorities assigned to receive queue 1. All the packets with priorities that match the values defined in this field are routed to receive queue 1.<br>For example, if PSRQ1[4] = 1, packets with MAC_Ext_Configuration[USP] equal to 4 are routed to receive queue 1. You must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues. This means that the same priority is not mapped to multiple receive queues. |
| 7-0<br>PSRQ0 | Priorities Selected In Receive Queue 0<br>Determines the priorities assigned to receive queue 0. All the packets with priorities that match the values defined in this field are routed to receive queue 0.<br>For example, if PSRQ0[5] = 1, packets with MAC_Ext_Configuration[USP] equal to 5 are routed to receive queue 0. You must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues. This means that the same priority is not mapped to multiple receive queues. |

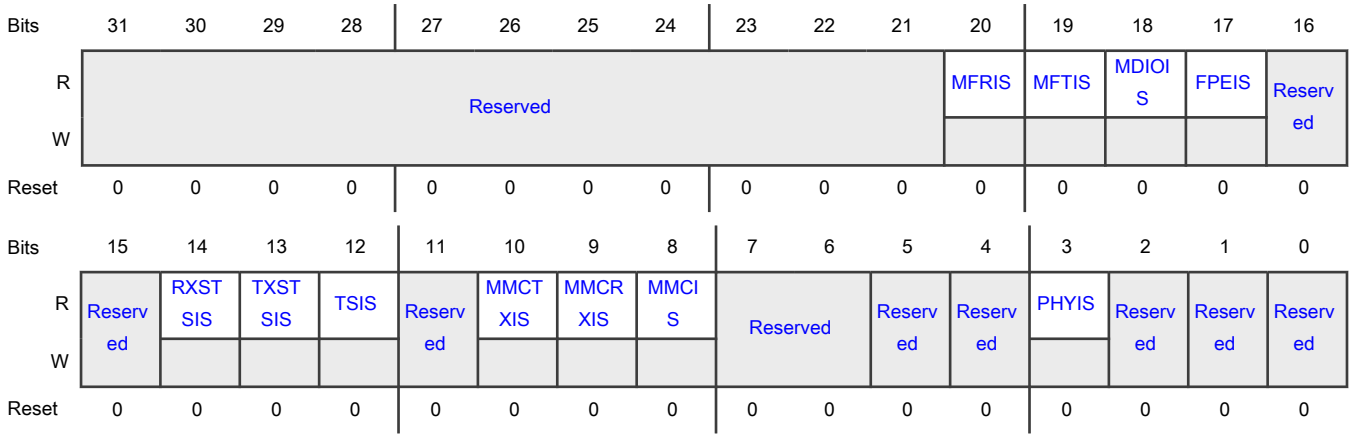
## 72.17.32 MAC Interrupt Status (MAC_Interrupt_Status)

### Offset

| Register | Offset |
|---|---|
| MAC_Interrupt_Status | B0h |

### Function
Contains the status of interrupts.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-21<br>— | Reserved |
| 20<br>MFRIS | MMC FPE Receive Interrupt Status<br>Indicates the status of MMC FPE receive interrupt.<br>This field becomes 1 when any of the fields in MMC Receive Packet Assembly Error Counter Interrupt Status (MMC_FPE_Rx_Interrupt) is 1. The field becomes 0 when all the other fields in MMC Receive Packet Assembly Error Counter Interrupt Status (MMC_FPE_Rx_Interrupt) become 0 too.<br>0b - Inactive<br>1b - Active |
| 19<br>MFTIS | MMC FPE Transmit Interrupt Status<br>Indicates the status of MMC FPE transmit interrupt.<br>This field becomes 1 when an interrupt generates in MMC Transmit FPE Fragment Counter Interrupt Status (MMC_FPE_Tx_Interrupt). The field becomes 0 when all the other fields in this register become 0 too.<br>This field is valid only when you select the Enable MAC Management Counters (MMC) option along with FPE support. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Inactive<br>1b - Active |
| 18<br>MDIOIS | MDIO Interrupt Status<br>Indicates the status of an interrupt event after the completion of an MDIO operation.<br>To reset this field, you must read this field or write 1 to it when MAC_CSR_SW_Ctrl[RCWE] = 1.<br>Access restrictions apply to this field. It clears on a read or write of 1 when MAC_CSR_SW_Ctrl[RCWE] = 1. The field automatically becomes 1 on an internal event.<br>0b - Inactive<br>1b - Active |
| 17<br>FPEIS | Frame Preemption Interrupt Status<br>Indicates the status of an interrupt event during the frame preemption operation (the value of RVER, RRSP, TVER, and TRSP fields of MAC FPE Control STS (MAC_FPE_CTRL_STS) = 1).<br>To reset this field, you must clear the event in MAC FPE Control STS (MAC_FPE_CTRL_STS) that caused the interrupt.<br>0b - Inactive<br>1b - Active |
| 16<br>— | Reserved |
| 15<br>— | Reserved |
| 14<br>RXSTSIS | Receive Status Interrupt<br>Indicates the status of received packets.<br>This field becomes 1 when MAC_Rx_Tx_Status[RWT] = 1. The field becomes 0 when you read the corresponding interrupt source field in MAC Rx Transmit Status (MAC_Rx_Tx_Status), or if you write to this interrupt source field when MAC_CSR_SW_Ctrl[RCWE] = 1.<br>The field becomes 0 in either of these circumstances:<br>• You read the corresponding interrupt-source field in MAC Rx Transmit Status (MAC_Rx_Tx_Status).<br>• You write 1 to the corresponding interrupt-source field in MAC Rx Transmit Status (MAC_Rx_Tx_Status) when MAC_CSR_SW_Ctrl[RCWE] = 1.<br>0b - Inactive<br>1b - Active |
| 13<br>TXSTSIS | Transmit Status Interrupt<br>Indicates the status of transmitted packets. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | This field becomes 1 when any of the following fields is 1 in MAC Rx Transmit Status (MAC_Rx_Tx_Status):<br>• Excessive collision (EXCOL)<br>• Late collision (LCOL)<br>• Excessive deferral (EXDEF)<br>• Loss of carrier (LCARR)<br>• No carrier (NCARR)<br>• Jabber timeout (TJT)<br>This field becomes 0 in either of these circumstances:<br>• You read the corresponding interrupt-source field in MAC Rx Transmit Status (MAC_Rx_Tx_Status).<br>• You write 1 to the corresponding interrupt-source field in MAC Rx Transmit Status (MAC_Rx_Tx_Status) when MAC_CSR_SW_Ctrl[RCWE] = 1.<br>0b - Inactive<br>1b - Active |
| 12<br>TSIS | Timestamp Interrupt Status<br>Indicates the status of timestamp interrupt.<br>If the timestamp feature is enabled, this field becomes 1 when any of the following conditions is true:<br>• The system time value is equal to or exceeds the value specified in MAC PPS1 Target Time In Seconds (MAC_PPS1_Target_Time_Seconds) and MAC PPS1 Target Time In Nanoseconds (MAC_PPS1_Target_Time_Nanoseconds).<br>• There is an overflow in MAC System Time In Seconds (MAC_System_Time_Seconds).<br>• The target time error occurred, which means, the programmed target time already elapsed.<br>In configurations other than EQOS_CORE, when the drop transmit status is enabled in MTL, this field becomes 1 when the captured transmit timestamp is updated in MAC Transmit Timestamp Status In Nanoseconds (MAC_Tx_Timestamp_Status_Nanoseconds) and MAC Transmit Timestamp Status In Seconds (MAC_Tx_Timestamp_Status_Seconds).<br>The field becomes 0 in either of these circumstances:<br>• You read the corresponding interrupt-source field in MAC Rx Transmit Status (MAC_Rx_Tx_Status).<br>• You write 1 to the corresponding interrupt-source field in MAC Rx Transmit Status (MAC_Rx_Tx_Status) when MAC_CSR_SW_Ctrl[RCWE] = 1.<br>0b - Inactive<br>1b - Active |
| 11<br>— | Reserved |
| 10<br>MMCTXIS | MMC Transmit Interrupt Status<br>Indicates the status of MMC transmit interrupt. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | This field:<br>• Becomes 1 when any field in MMC Transmit Interrupt (MMC_Tx_Interrupt) becomes 1.<br>• Becomes 0 when all the fields in MMC Transmit Interrupt (MMC_Tx_Interrupt)return to 0.<br>This field is valid only when you select the Enable MAC Management Counters (MMC) option.<br>0b - Inactive<br>1b - Active |
| 9<br>MMCRXIS | MMC Receive Interrupt Status<br>Indicates the status of MMC receive interrupt.<br>This field:<br>• Becomes 1 when any field in MMC Receive Interrupt (MMC_Rx_Interrupt) becomes 1.<br>• Becomes 0 when all the fields in MMC Receive Interrupt (MMC_Rx_Interrupt)return to 0.<br>This field is valid only when you select the Enable MAC Management Counters (MMC) option.<br>0b - Inactive<br>1b - Active |
| 8<br>MMCIS | MMC Interrupt Status<br>Indicates the status of MMC interrupt.<br>This field becomes 1 when MMCRXIS = MMCTXIS = 1, and the field becomes 0 only when MMCRXIS = MMCTXIS = 0.<br>MMCIS is valid only when you select the Enable MAC Management Counters (MMC) option.<br>0b - Inactive<br>1b - Active |
| 7-6<br>— | Reserved |
| 5<br>— | Reserved |
| 4<br>— | Reserved |
| 3<br>PHYIS | PHY Interrupt<br>Indicates whether the PHY interrupt is detected.<br>This field becomes 1 when rising edge is detected on the phy_intr_i input signal. The field becomes 0 when you read this register (or if you write 1 to this field when MAC_CSR_SW_Ctrl[RCWE] = 1).<br>0b - Not detected |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Detected |
| 2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>— | Reserved |

## 72.17.33 MAC Interrupt Enable (MAC_Interrupt_Enable)

### Offset

| Register | Offset |
|---|---|
| MAC_Interrupt_Enable | B4h |

### Function
Contains masks for generating MAC interrupts.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | MDIOIE | FPEIE | Reserved |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | RXSTSIE | TXSTSIE | TSIE | Reserved | | | | | | Reserved | Reserved | PHYIE | Reserved | Reserved | Reserved |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18 | MDIO Interrupt Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| MDIOIE | Indicates whether the MDIO interrupt is enabled or disabled.<br>If this field is 1, it enables the assertion of the interrupt when MAC_Interrupt_Status[MDIOIS] = 1.<br>0b - Disabled<br>1b - Enabled |
| 17<br>FPEIE | Frame Preemption Interrupt Enable<br>Indicates whether the frame preemption interrupt is enabled or disabled.<br>If this field is 1, it enables the assertion of the interrupt when MAC_Interrupt_Status[FPEIS] = 1.<br>0b - Disabled<br>1b - Enabled |
| 16<br>— | Reserved |
| 15<br>— | Reserved |
| 14<br>RXSTSIE | Receive Status Interrupt Enable<br>Indicates whether the receive status interrupt is enabled or disabled.<br>If this field is 1, it enables the assertion of the interrupt signal because of the setting of MAC_Interrupt_Status[RXSTSIS].<br>0b - Disabled<br>1b - Enabled |
| 13<br>TXSTSIE | Transmit Status Interrupt Enable<br>Indicates whether the timestamp status interrupt is enabled or disabled.<br>If this field is 1, it enables the assertion of the interrupt signal because of the setting of MAC_Interrupt_Status[TXSTSIS].<br>0b - Disabled<br>1b - Enabled |
| 12<br>TSIE | Timestamp Interrupt Enable<br>Indicates whether the timestamp interrupt is enabled or disabled.<br>If this field is 1, it enables the assertion of the interrupt signal because of the setting MAC_Interrupt_Status[TSIS].<br>0b - Disabled<br>1b - Enabled |
| 11-6 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 5<br>— | Reserved |
| 4<br>— | Reserved |
| 3<br>PHYIE | PHY Interrupt Enable<br>Indicates whether the assertion of the interrupt signal is enabled or disabled.<br>If this field is 1, it enables the assertion of the interrupt signal because of the setting of MAC_Interrupt_Status[PHYIS].<br>0b - Disabled<br>1b - Enabled |
| 2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>— | Reserved |

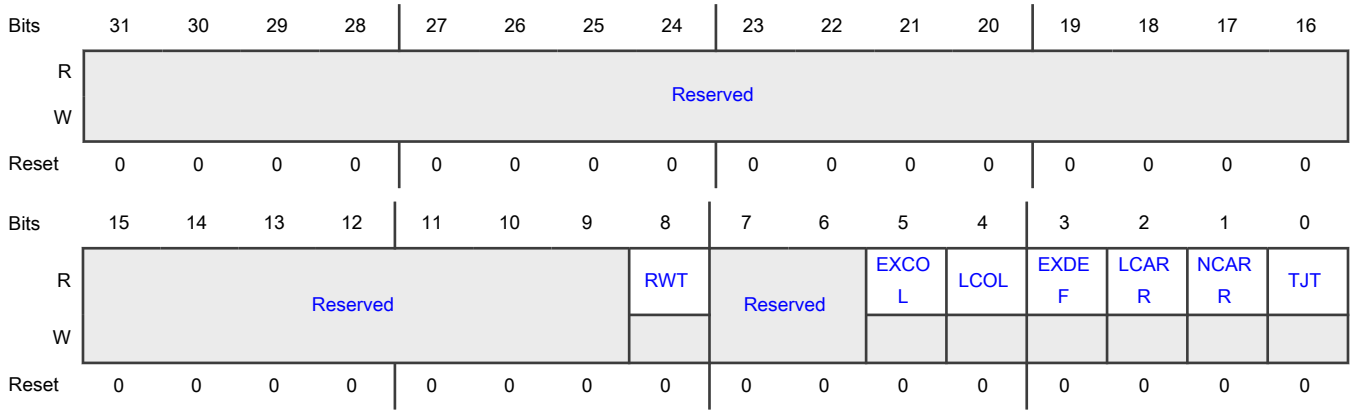
## 72.17.34 MAC Rx Transmit Status (MAC_Rx_Tx_Status)

**Offset**

| Register | Offset |
|---|---|
| MAC_Rx_Tx_Status | B8h |

**Function**
Contains the receive and transmit error status.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>RWT | Receive Watchdog Timeout<br>Indicates the status of receive watchdog.<br>This field becomes 1 when a packet with a length greater than 2,048 bytes is received (10,240 bytes when Jumbo Packet mode is enabled) and MAC_Configuration[WD] = 0. RWT becomes 1 when a packet with length greater than 16,383 bytes is received and MAC_Configuration[WD] = 1.<br>Access restrictions apply to RWT. It becomes 0 on read or write of 1 when MAC_CSR_SW_Ctrl[RCWE] = 1. RWT automatically becomes 1 on an internal event occurrence.<br>0b - No receive watchdog timed out<br>1b - Receive watchdog timed out |
| 7-6<br>— | Reserved |
| 5<br>EXCOL | Excessive Collisions<br>Indicates the status of excessive collision.<br>If MTL_Operation_Mode[DTXSTS] = 1, this field, EXCOL, indicates that the transmission aborted after 16 successive collisions while attempting to transmit the current packet.<br>If MAC_Configuration[DR] = 1, EXCOL becomes 1 after the first collision and the packet transmission is aborted.<br>Access restrictions apply to EXCOL. It becomes 0 on read or write of 1 when MAC_CSR_SW_Ctrl[RCWE] = 1. EXCOL automatically becomes 1 on an internal event occurrence.<br>0b - No collision<br>1b - Excessive collision is sensed |
| 4 | Late Collision |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| LCOL | Indicates the status of late collision.<br>If MTL_Operation_Mode[DTXSTS] = 1, this field, LCOL, indicates that the packet transmission aborted because a collision occurred after the collision window (64 bytes including preamble in MII mode; 512 bytes including preamble and carrier extension in GMII mode).<br>This field is invalid if an underflow error occurs.<br>Access restrictions apply to LCOL. It becomes 0 on read or write of 1 when MAC_CSR_SW_Ctrl[RCWE] = 1. LCOL automatically becomes 1 on an internal event occurrence.<br>0b - No collision<br>1b - Late collision is sensed |
| 3<br>EXDEF | Excessive Deferral<br>Indicates whether the transmission ended because of excessive deferral.<br>If MTL_Operation_Mode[DTXSTS] = 1 and MAC_Configuration[DC] = 1 too, this field, EXDEF, indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 in 1000/2500 Mbit/s mode or when the jumbo packet is enabled).<br>Access restrictions apply to EXDEF. It becomes 0 on read or write of 1 when MAC_CSR_SW_Ctrl[RCWE] = 1. EXDEF automatically becomes 1 on an internal event occurrence.<br>0b - No excessive deferral<br>1b - Excessive deferral |
| 2<br>LCARR | Loss of Carrier<br>Indicates the status of carrier signal.<br>If MTL_Operation_Mode[DTXSTS] = 1, this field, LCARR, indicates that the loss of carrier occurred during packet transmission, which means, the PHY_CRS_I signal was inactive for one or more transmission clock periods during packet transmission. The field is valid only for packets transmitted without collision.<br>Access restrictions apply to LCARR. It becomes 0 on read or write of 1 when MAC_CSR_SW_Ctrl[RCWE] = 1. LCARR automatically becomes 1 on an internal event occurrence.<br>0b - Carrier is present<br>1b - Loss of carrier |
| 1<br>NCARR | No Carrier<br>Indicates whether the carrier signal is present from the PHY at the end of preamble transmission.<br>If MTL_Operation_Mode[DTXSTS] = 1, NCARR indicates that the carrier signal from the PHY is absent at the end of preamble transmission.<br>Access restrictions apply to NCARR. It becomes 0 after either of these actions:<br>• You read NCARR.<br>• You write 1 to NCARR when MAC_CSR_SW_Ctrl[RCWE] = 1.<br>NCARR automatically becomes 1 after an internal event occurs. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Present<br>1b - Absent |
| 0<br>TJT | Transmit Jabber Timeout<br>Indicates whether a transmit-jabber timer timeout occurred. The timeout occurs after any of these events:<br>• MAC_Configuration[JD] = 0 and the normal packet size exceeds 2,048 bytes.<br>• MAC_Configuration[JD] = 0 and the jumbo packet size exceeds 10240 bytes.<br>• MAC_Configuration[JD] = 1 and the normal packet size exceeds 16383 bytes.<br>Access restrictions apply to TJT. It becomes 0 after either of these actions:<br>• You read TJT.<br>• You write 1 to TJT when MAC_CSR_SW_Ctrl[RCWE] = 1.<br>TJT automatically becomes 1 after an internal event occurs.<br>0b - No transmit jabber timeout occurred<br>1b - Transmit jabber timeout occurred |

## 72.17.35 MAC Version (MAC_Version)

### Offset

| Register | Offset |
|---|---|
| MAC_Version | 110h |

### Function

Identifies the version of the module.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>CFGVER | IP Configuration Version |
| 7-0<br>IPVER | IP Version |

## 72.17.36 MAC Debug (MAC_Debug)

### Offset

| Register | Offset |
|---|---|
| MAC_Debug | 114h |

### Function
Provides the debug status of the various MAC blocks.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | | | | | | TFCSTS | | TPESTS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | | | | | | RFCFCSTS | | RPESTS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18-17 | MAC Transmit Packet Controller Status |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| TFCSTS | Indicates the state of the MAC transmit packet controller module.<br>00b - Idle state<br>01b - Waiting for one of these: status of the previous packet or IPG, or for the back-off period to be over<br>10b - Generating and transmitting a pause control packet (in Full-Duplex mode)<br>11b - Transferring input packet for transmission |
| 16<br>TPESTS | MAC GMII Or MII Transmit Protocol Engine Status<br>Indicates whether MAC GMII or MII transmit protocol engine is detected.<br>If this field is 1, it indicates that the MAC GMII or MII transmit protocol engine is actively transmitting data and is not in an idle state.<br>0b - Not detected<br>1b - Detected |
| 15-3<br>— | Reserved |
| 2-1<br>RFCFCSTS | MAC Receive Packet Controller FIFO Status<br>Indicates the status of the small FIFO read and write controllers of the MAC receive packet controller module.<br>If this field is 1, it indicates that the small FIFO read and write controllers of the MAC receive packet controller module are in an active state.<br>00b - Inactive<br>01b - Active |
| 0<br>RPESTS | Receive Protocol Engine Status<br>Indicates whether MAC GMII or MII receive protocol engine is detected.<br>If this field is 1, it indicates that the MAC GMII or MII receive protocol engine is actively receiving data and is not in an idle state.<br>0b - Not detected<br>1b - Detected |

## 72.17.37 MAC Hardware Feature 0 (MAC_HW_Feature0)

**Offset**

| Register | Offset |
|---|---|
| MAC_HW_Feature0 | 11Ch |

### Function

Indicates the presence of the first set of EMAC optional features or functions. You can use this register to dynamically enable or disable the programs related to these features.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | ACTPHYSEL | | | SAVLANI... | TSSTSSEL | | MACADR6... | MACADR3... | ADDMACADRSEL | | | | | Reserved | RXCOESEL |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | TXCOESEL | EEESEL | TSSEL | Reserved | | ARPOFFS... | MMCSEL | MGKSEL | RWKSEL | SMASEL | VLHASH | PCSSEL | HDSEL | GMIISEL | MIISEL |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-28<br>ACTPHYSEL | Active PHY Feature<br>Indicates the selected active PHY interface.<br>If you have multiple PHY interfaces (GMII, MII, RevMII, RGMII, RMII, RTBI, SGMII, SGMII, or TBI) in your configuration, this field indicates the sampled value of the PHY_INTF_SEL_I signal during reset deassertion.<br>000b - GMII or MII<br>001b - RGMII<br>010b - SGMII<br>011b - TBI<br>100b - RMII<br>101b - RTBI<br>110b - SMII<br>111b - RevMII |
| 27<br>SAVLANINS | SA or VLAN Insertion Feature<br>Indicates whether the feature that enables SA or VLAN insertion on transmit is selected.<br>This field becomes 1 if you select the "enable SA and VLAN insertion on Tx" option.<br>0b - Not selected<br>1b - Selected |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 26-25<br>TSTSSEL | Timestamp System Time Source Feature<br>Indicates the source of the timestamp system time.<br>This field becomes 1 if you select the "enable IEEE 1588 timestamp support" option.<br>00b - Internal<br>01b - External<br>10b - Both internal and external<br>11b - Reserved |
| 24<br>MACADR64SEL | MAC Addresses 64-127<br>Indicates whether the feature that enables additional 64 MAC address registers (64-127) is selected.<br>This field becomes 1 if you enable the "MAC addresses (64-127) select" option.<br>0b - Not selected<br>1b - Selected |
| 23<br>MACADR32SEL | MAC Addresses 32-63<br>Indicates whether the feature that enables additional 32 MAC address registers (32-63) is selected.<br>This field becomes 1 if you select the "MAC addresses 32-63 select" option.<br>0b - Not selected<br>1b - Selected |
| 22-18<br>ADDMACADRSEL | MAC Addresses 1-31<br>Indicates whether the feature that enables additional 1-31 MAC address registers is selected.<br>This field becomes 1 if you select a non-zero value for enabling the "MAC addresses 1-31 select" option.<br>0_0000b - Not selected<br>0_0001b - Selected |
| 17<br>— | Reserved |
| 16<br>RXCOESEL | Receive Checksum Offload Feature<br>Indicates whether the feature that enables receive TCP/IP checksum offload is selected.<br>This field becomes 1 if you select the "enable receive TCP/IP checksum check" option.<br>0b - Not selected<br>1b - Selected |
| 15<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 14<br>TXCOESEL | Transmit Checksum Offload Feature<br>Indicates whether the feature that enables transmit TCP/IP checksum insertion is selected.<br>This field becomes 1 if you select the "enable transmit TCP/IP checksum insertion" option.<br>0b - Not selected<br>1b - Selected |
| 13<br>EEESEL | Energy Efficient Ethernet (EEE) Feature<br>Indicates whether the feature that enables EEE is selected.<br>This field becomes 1 if you select the "enable EEE" option.<br>0b - Not selected<br>1b - Selected |
| 12<br>TSSEL | IEEE 1588-2008 Timestamp Feature<br>Indicates whether the option that enables IEEE 1588-2008 timestamp is selected.<br>This field becomes 1 if you select the "enable IEEE 1588 timestamp support" option.<br>0b - Not selected<br>1b - Selected |
| 11-10<br>— | Reserved |
| 9<br>ARPOFFSEL | ARP Offload Feature<br>Indicates whether the feature that enables ARP offload is selected.<br>This field becomes 1 if you select the "enable IPv4 ARP offload" option.<br>0b - Not selected<br>1b - Selected |
| 8<br>MMCSEL | MAC Management Counters (MMC) Feature<br>Indicates whether the feature that enables MMC is selected.<br>This field becomes 1 if you select the "enable MMC" option.<br>0b - Not selected<br>1b - Selected |
| 7<br>MGKSEL | PMT Magic Packet Feature<br>Indicates whether the feature that enables PMT magic detection is selected.<br>This field becomes 1 if you select the "enable magic packet detection" option.<br>0b - Not selected |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Selected |
| 6<br>RWKSEL | PMT Remote Wake-Up Packet Feature<br>Indicates whether the feature that enables PMT remote wake-up packet detection is selected.<br>This field becomes 1 if you select the "enable remote wake-up packet detection" option.<br>0b - Not selected<br>1b - Selected |
| 5<br>SMASEL | SMA (MDIO) Interface Feature<br>Indicates whether the feature that enables station management (MDIO interface) is selected.<br>This field becomes 1 if you select the "enable station management (MDIO interface)" option.<br>0b - Not selected<br>1b - Selected |
| 4<br>VLHASH | VLAN Hash Filter Feature<br>Indicates whether the filtering option based on VLAN hash table is selected.<br>This field becomes 1 if you select the "enable VLAN hash table based filtering" option.<br>0b - Not selected<br>1b - Selected |
| 3<br>PCSSEL | PCS Select<br>Indicates if either of the TBI, SGMII, or RTBI PHY interface options is selected.<br>This field becomes 1 if you select any one of the TBI, SGMII, or RTBI PHY interface options.<br>0b - No<br>1b - Yes |
| 2<br>HDSEL | Half-Duplex Support Feature<br>Indicates whether half-duplex support is available as the mode of operation.<br>This field becomes 1 if you select Half-Duplex as the mode of operation.<br>0b - Unavailable<br>1b - Available |
| 1<br>GMIISEL | 1000 Mbit/s Support Feature<br>Indicates whether 1000 Mbit/s support is available as the mode of operation.<br>This field becomes 1 if you select 1000 Mbit/s as the mode of operation.<br>0b - Unavailable<br>1b - Available |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 0<br>MIISEL | 10 or 100 Mbit/s Support Feature<br>Indicates whether 10 or 100 Mbit/s support is available as the mode of operation.<br>This field becomes 1 if you select 10/100 Mbit/s as the mode of operation.<br>0b - Unavailable<br>1b - Available |

## 72.17.38 MAC Hardware Feature 1 (MAC_HW_Feature1)

### Offset

| Register | Offset |
|---|---|
| MAC_HW_Feature1 | 120h |

### Function

Indicates the presence of the second set of optional features or functions of the module. You can use this register to dynamically enable or disable the programs related to these features.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | L3L4FNUM | | | Reserved | HASHTBLSZ | | | POUOST | Reserved | RAVSEL | AVSEL | DBGMEMA | TSOEN | SPHEN | DCBEN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | ADDR64 | | ADVTHWO... | PTOEN | OSTEN | TXFIFOSIZE | | | | | SPRAM | RXFIFOSIZE | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-27 | L3 Or L4 Filter Number |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| L3L4FNUM | Indicates the total number of L3 or L4 filters.<br>0000b - No filters (0)<br>0001b - 1<br>0010b - 2<br>0011b - 3<br>0100b - 4<br>0101b - 5<br>0110b - 6<br>0111b - 7<br>1000b - 8 |
| 26<br>— | Reserved |
| 25-24<br>HASHTBLSZ | Hash Table Size<br>Indicates the size of the hash table.<br>00b - No hash table<br>01b - 64<br>10b - 128<br>11b - 256 |
| 23<br>POUOST | One Step For PTP Over UDP/IP Feature<br>Indicates whether the feature that enables one step for PTP over UDP/IP is selected.<br>This field becomes 1 if you select the "enable one-step timestamp for PTP over UDP/IP" option.<br>0b - Not selected<br>1b - Selected |
| 22<br>— | Reserved |
| 21<br>RAVSEL | Receive Side-Only AV Feature<br>Indicates whether the feature that enables audio-video bridging only on the receive side is selected.<br>This field becomes 1 if you select the "enable audio-video bridging on receive side only" option.<br>0b - Not selected<br>1b - Selected |
| 20 | AV Feature<br>Indicates whether the feature that enables audio-video bridging is selected. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| AVSEL | This field becomes 1 if you select the "enable audio video bridging" option.<br>0b - Not selected<br>1b - Selected |
| 19<br>DBGMEMA | DMA Debug Registers Enable Feature<br>Indicates whether the feature that enables DMA debug registers is selected.<br>This field becomes 1 if you select the "enable DMA debug mode" option.<br>0b - Not selected<br>1b - Selected |
| 18<br>TSOEN | TCP Segmentation Offload Enable Feature<br>Indicates whether the feature that enables TCP segmentation offload for TCP/IP packets is selected.<br>This field becomes 1 if you select the "enable TCP segmentation offloading for TCP/IP packets" option.<br>0b - Not selected<br>1b - Selected |
| 17<br>SPHEN | Split Header Enable Feature<br>Indicates whether the feature that enables a split header structure is selected.<br>This field becomes 1 if you select the "enable split header structure" option.<br>0b - Not selected<br>1b - Selected |
| 16<br>DCBEN | DCB Enable Feature<br>Indicates whether the feature that enables data center bridging is selected.<br>This field becomes 1 if you select the "enable data center bridging" option.<br>0b - Not selected<br>1b - Selected |
| 15-14<br>ADDR64 | Address Width Feature<br>Indicates the configured address width.<br>00b - 32<br>01b - 40<br>10b - 48<br>11b - Reserved |
| 13<br>ADVTHWORD | IEEE 1588 High-Word Feature<br>Indicates whether the feature that enables the IEEE 1588 high-word register is selected.<br>This field becomes 1 if you select the "enable/add IEEE 1588 higher-word register" option. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Not selected<br>1b - Selected |
| 12<br>PTOEN | PTP Offload Enable Feature<br>Indicates whether the feature that enables PTP offload is selected.<br>This field becomes 1 if you select the "enable PTP timestamp offload" option.<br>0b - Not selected<br>1b - Selected |
| 11<br>OSTEN | One-Step Timestamping Enable Feature<br>Indicates whether the feature that enables one-step timestamping is selected.<br>This field becomes 1 if you select the "enable one-step timestamp" option.<br>0b - Not selected<br>1b - Selected |
| 10-6<br>TXFIFOSIZE | MTL Transmit FIFO Size Feature<br>Contains the configured value of MTL transmit FIFO (in bytes) expressed as log to base 2 minus 7, which means, Log 2 (TXFIFO_SIZE) - 7.<br>0_0000b - 128 bytes<br>0_0001b - 256 bytes<br>0_0010b - 512 bytes<br>0_0011b - 1024 bytes<br>0_0100b - 2048 bytes<br>0_0101b - 4096 bytes<br>0_0110b - 8192 bytes<br>0_0111b - 16384 bytes<br>0_1000b - 32 KB<br>0_1001b - 64 KB<br>0_1010b - 128 KB<br>0_1011b - Reserved |
| 5<br>SPRAM | Single Port RAM Feature<br>Indicates whether the single-port RAM feature is selected.<br>This field becomes 1 if you select the option that enables the "use single port RAM" feature.<br>0b - Not selected<br>1b - Selected |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 4-0<br>RXFIFOSIZE | MTL Receive FIFO Size Feature<br>Contains the configured value of MTL receive FIFO (in bytes) expressed as log to base 2 minus 7, which means, Log 2 (RXFIFO_SIZE) - 7.<br>0_0000b - 128 bytes<br>0_0001b - 256 bytes<br>0_0010b - 512 bytes<br>0_0011b - 1024 bytes<br>0_0100b - 2048 bytes<br>0_0101b - 4096 bytes<br>0_0110b - 8192 bytes<br>0_0111b - 16384 bytes<br>0_1000b - 32 KB<br>0_1001b - 64 KB<br>0_1010b - 128 KB<br>0_1011b - 256 KB<br>0_1100b - Reserved |

## 72.17.39 MAC Hardware Feature 2 (MAC_HW_Feature2)

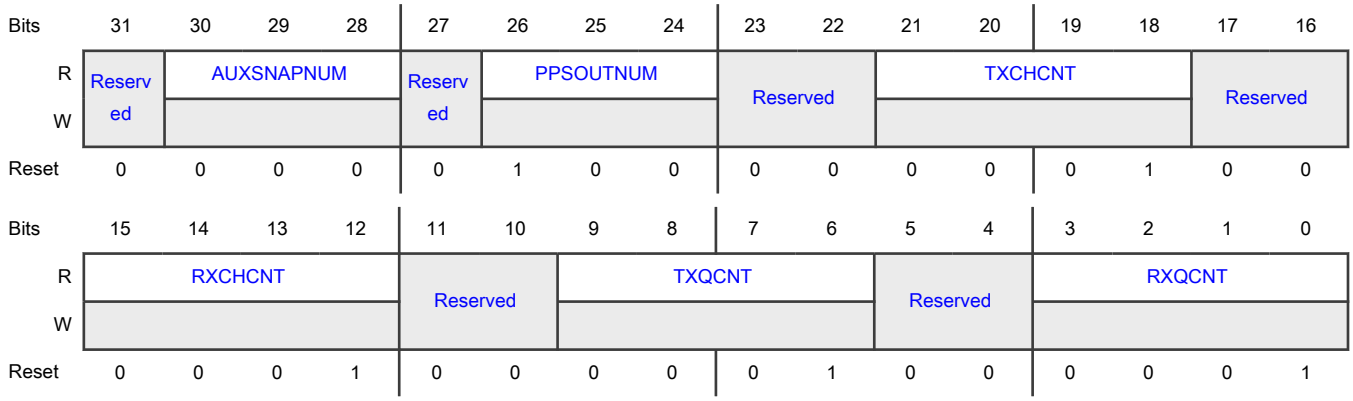
### Offset

| Register | Offset |
|---|---|
| MAC_HW_Feature2 | 124h |

### Function

Indicates the presence of the third set of optional features or functions of the module. You can use this register to dynamically enable or disable the programs related to these features.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-28<br>AUXSNAPNUM | Number Of Auxiliary Snapshot Inputs<br>Indicates the number of auxiliary snapshot inputs.<br>000b - No auxiliary input (0)<br>001b - 1<br>010b - 2<br>011b - 3<br>100b - 4<br>101b - Reserved |
| 27<br>— | Reserved |
| 26-24<br>PPSOUTNUM | Number Of PPS Outputs<br>Indicates the number of PPS outputs.<br>000b - No PPS output (0)<br>001b - 1<br>010b - 2<br>011b - 3<br>100b - 4<br>101b - Reserved |
| 23-22<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 21-18<br>TXCHCNT | Number Of DMA Transmit Channels<br>Indicates the number of DMA transmit channels.<br>0000b - 1<br>0001b - 2<br>0010b - 3<br>0011b - 4<br>0100b - 5<br>0101b - 6<br>0110b - 7<br>0111b - 8 |
| 17-16<br>— | Reserved |
| 15-12<br>RXCHCNT | Number Of DMA Receive Channels<br>Indicates the number of DMA receive channels.<br>0000b - 1<br>0001b - 2<br>0010b - 3<br>0011b - 4<br>0100b - 5<br>0101b - 6<br>0110b - 7<br>0111b - 8 |
| 11-10<br>— | Reserved |
| 9-6<br>TXQCNT | Number Of MTL Transmit Queues<br>Indicates the number of MTL transmit queues.<br>0000b - 1<br>0001b - 2<br>0010b - 3<br>0011b - 4<br>0100b - 5<br>0101b - 6 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0110b - 7<br>0111b - 8 |
| 5-4<br>— | Reserved |
| 3-0<br>RXQCNT | Number Of MTL Receive Queues<br>Indicates the number of MTL receive queues.<br>0000b - 1<br>0001b - 2<br>0010b - 3<br>0011b - 4<br>0100b - 5<br>0101b - 6<br>0110b - 7<br>0111b - 8 |

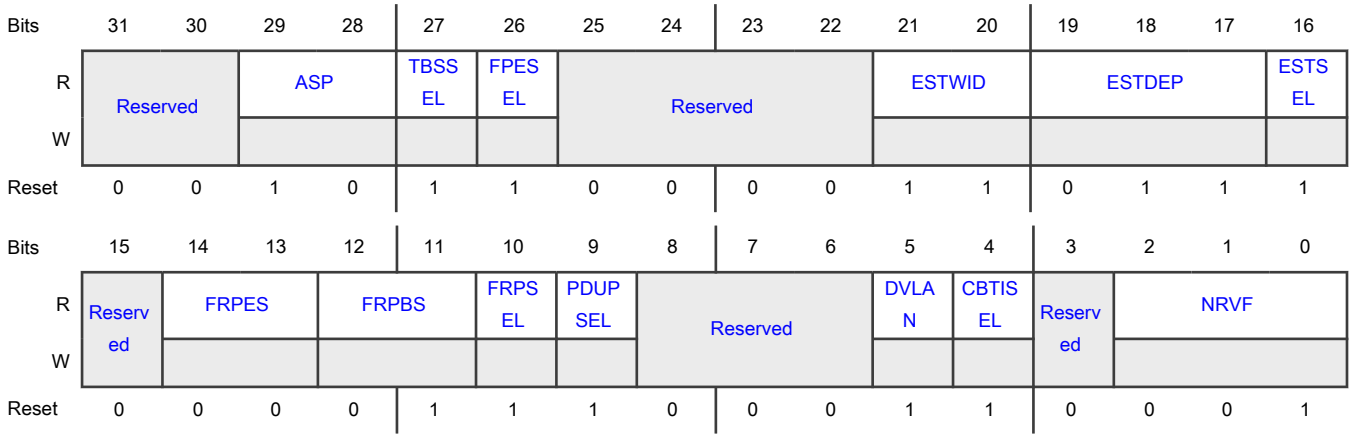
## 72.17.40 MAC Hardware Feature 3 (MAC_HW_Feature3)

**Offset**

| Register | Offset |
|---|---|
| MAC_HW_Feature3 | 128h |

**Function**
Indicates the presence of the fourth set of optional features or functions of the module. You can use this register to dynamically enable or disable the programs related to these features.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-28<br>ASP | Automotive Safety Package<br>Indicates the encoding for the different automotive safety features.<br>00b - No safety features selected<br>01b - Only "ECC protection for external memory" feature is selected<br>10b - All the safety features are selected without the "parity port enable for external interface" feature<br>11b - All the safety features are selected with the "parity port enable for external interface" feature |
| 27<br>TBSSEL | Time-Based Scheduling Feature<br>Indicates whether the time-based scheduling feature is selected.<br>This field becomes 1 if you select the option that enables the time-based scheduling feature.<br>0b - Selected<br>1b - Selected |
| 26<br>FPESEL | Frame Preemption Feature<br>Indicates whether the feature that enables frame preemption is selected.<br>This field becomes 1 if you select the "enable frame preemption" option.<br>0b - Not selected<br>1b - Selected |
| 25-22<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 21-20<br>ESTWID | Estimated Time Interval Width<br>Indicates the width configured for the time interval field in the gate control list.<br>00b - Width not configured<br>01b - 16 bits<br>10b - 20 bits<br>11b - 24 bits |
| 19-17<br>ESTDEP | Depth Of Gate Control List<br>Indicates the depth of gate control list expressed as Log 2 (DWC_EQOS_EST_DEP) - 5.<br>000b - No depth configured<br>001b - 64 bytes<br>010b - 128 bytes<br>011b - 256 bytes<br>100b - 512 bytes<br>101b - 1024 bytes<br>110b - Reserved |
| 16<br>ESTSEL | Enhancements To Scheduling Traffic Feature<br>Indicates whether the "enhancements to scheduling traffic" feature is selected.<br>This field becomes 1 if you select the "enable enhancements to scheduling traffic" option.<br>0b - Not selected<br>1b - Selected |
| 15<br>— | Reserved |
| 14-13<br>FRPES | Flexible Receive Parser Table Entry Size<br>Indicates the maximum number of parser entries that the flexible receive parser supports.<br>00b - 64<br>01b - 128<br>10b - 256<br>11b - Reserved |
| 12-11<br>FRPBS | Flexible Receive Parser Buffer Size<br>Indicates the maximum number of packet data bytes that the flexible receive parser supports.<br>00b - 64 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 01b - 128<br>10b - 256<br>11b - Reserved |
| 10<br>FRPSEL | Flexible Receive Parser Feature<br>Indicates whether the flexible receive parser feature is selected.<br>This field becomes 1 if you select the "enable flexible programmable receive parser" option.<br>0b - Not selected<br>1b - Selected |
| 9<br>PDUPSEL | Broadcast/Multicast Packet Duplication Feature<br>Indicates whether the broadcast or multicast duplication feature is selected.<br>This field becomes 1 if you select the "broadcast or multicast packet duplication" option.<br>0b - Not selected<br>1b - Selected |
| 8-6<br>— | Reserved |
| 5<br>DVLAN | Double VLAN Tag Processing Feature<br>Indicates whether the double VLAN processing feature is selected.<br>0b - Not selected<br>1b - Selected |
| 4<br>CBTISEL | Queue/Channel Based VLAN Tag Insertion On Transmit Feature<br>Indicates whether the "queue- or channel-based VLAN tag insertion on Tx" feature is selected.<br>This field becomes 1 if you select the "enable queue- or channel-based VLAN tag insertion on Tx" option.<br>0b - Not selected<br>1b - Selected |
| 3<br>— | Reserved |
| 2-0<br>NRVF | Number Of Extended VLAN Tag Filters<br>Indicates the number of selected extended VLAN tag filters.<br>000b - No filters (0)<br>001b - 4<br>010b - 8 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 011b - 16<br>100b - 24<br>101b - 32<br>110b - Reserved |

## 72.17.41 MAC DPP FSM Interrupt Status (MAC_DPP_FSM_Interrupt_Status)

### Offset

| Register | Offset |
|---|---|
| MAC_DPP_FSM_Interrupt_Status | 140h |

### Function

Contains the status of automative-safety related data path parity errors, interface timeout errors, FSM state parity errors, and FSM state timeout errors.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | FSMPES | Reserved | | | | | | Reserved | MSTTES |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | Reserved | Reserved | PTES | ATES | Reserved | RTES | TTES | Reserved | Reserved | ARPES | MTSPES | MPES | RDPES | Reserved | Reserved |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved |
| 24<br>FSMPES | FSM State Parity Error Status<br>Indicates whether a parity error is detected on one of the FSM state registers. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | After a read operation, the field's value clears to 0.<br>0b - Not detected<br>1b - Detected |
| 23-18<br>— | Reserved |
| 17<br>— | Reserved |
| 16<br>MSTTES | Master Read Or Write Timeout Error Status<br>Indicates whether an application or CSR timeout error is detected on the master (AXI, AHB, ARI, or ATI) interface.<br>After a read operation, the field's value clears to 0.<br>0b - Not detected<br>1b - Detected |
| 15<br>— | Reserved |
| 14<br>— | Reserved |
| 13<br>— | Reserved |
| 12<br>PTES | PTP FSM Timeout Error Status<br>Indicates whether one of the PTP FSM timeout errors is detected.<br>After a read operation, the field's value clears to 0.<br>0b - Not detected<br>1b - Detected |
| 11<br>ATES | APP FSM Timeout Error Status<br>Indicates whether one of the APP FSM timeout errors is detected.<br>After a read operation, the field's value clears to 0.<br>0b - Not detected<br>1b - Detected |
| 10<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 9<br>RTES | Receive FSM Timeout Error Status<br>Indicates whether one of the receive FSM timeout errors is detected.<br>After a read operation, the field's value clears to 0.<br>0b - Not detected<br>1b - Detected |
| 8<br>TTES | Transmit FSM Timeout Error Status<br>Indicates whether one of the transmit FSM timeout errors is detected.<br>After a read operation, the field's value clears to 0.<br>0b - Not detected<br>1b - Detected |
| 7<br>— | Reserved |
| 6<br>— | Reserved |
| 5<br>ARPES | Application Receive Interface Data Path Parity Error Status<br>Indicates whether the parity checker detected a parity error on the application receive interface (or on PC6 as shown in the transmit data path parity protection diagram).<br>After a read operation, the field's value clears to 0.<br>0b - Not detected<br>1b - Detected |
| 4<br>MTSPES | MTL Transmit Status Data Path Parity Checker Error Status<br>Indicates whether the parity checker detected a parity error for the MTL transmit status data on ATI (or on PC5 as shown in the transmit data path parity protection diagram).<br>After a read operation, the field's value clears to 0.<br>0b - Not detected<br>1b - Detected |
| 3<br>MPES | MTL Data Path Parity Checker Error Status<br>Indicates whether the parity checker detected a parity error on the MTL transmit write controller interface (or on PC4 as shown in the transmit data path parity protection diagram).<br>After a read operation, the field's value clears to 0.<br>0b - Not detected<br>1b - Detected |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 2<br>RDPES | Read Descriptor Parity Checker Error Status<br>Indicates whether the parity checker detected a parity error on the DMA read descriptor interface (or on PC3 as shown in the transmit data path parity protection diagram).<br>After a read operation, the field's value clears to 0.<br>0b - Not detected<br>1b - Detected |
| 1<br>— | Reserved |
| 0<br>— | Reserved |

## 72.17.42 MAC FSM Control (MAC_FSM_Control)

### Offset

| Register | Offset |
|---|---|
| MAC_FSM_Control | 148h |

### Function

Is used to control the FSM state parity and timeout error injection in Debug mode.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | Reserved | Reserved | PLGRNML | ALGRNML | Reserved | RLGRNML | TLGRNML | Reserved | Reserved | Reserved | PPEIN | APEIN | Reserved | RPEIN | TPEIN |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | Reserved | Reserved | PTEIN | ATEIN | Reserved | RTEIN | TTEIN | Reserved | | | | | | PRTYEN | TMOUTEN |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 30<br>— | Reserved |
| 29<br>— | Reserved |
| 28<br>PLGRNML | PTP Large Or Normal Mode Select<br>Indicates which mode of tic generation is used for the PTP domain.<br>0b - Normal mode<br>1b - Large mode |
| 27<br>ALGRNML | APP Large Or Normal Mode Select<br>Indicates which mode of tic generation is used for the APP domain.<br>0b - Normal mode<br>1b - Large mode |
| 26<br>— | Reserved |
| 25<br>RLGRNML | Receive Large Or Normal Mode Select<br>Indicates which mode of tic generation is used for the receive domain.<br>0b - Normal mode<br>1b - Large mode |
| 24<br>TLGRNML | Transmit Large Or Normal Mode Select<br>Indicates which mode of tic generation is used for the transmit domain.<br>0b - Normal mode<br>1b - Large mode |
| 23<br>— | Reserved |
| 22<br>— | Reserved |
| 21<br>— | Reserved |
| 20 | PTP FSM Parity Error Injection |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| PPEIN | Indicates the status of error injection for PTP FSM parity.<br>0b - Disabled<br>1b - Enabled |
| 19<br>APEIN | APP FSM Parity Error Injection<br>Indicates the status of error injection for APP FSM parity.<br>0b - Disabled<br>1b - Enabled |
| 18<br>— | Reserved |
| 17<br>RPEIN | Receive FSM Parity Error Injection<br>Indicates the status of error injection for receive FSM parity.<br>0b - Disabled<br>1b - Enabled |
| 16<br>TPEIN | Transmit FSM Parity Error Injection<br>Indicates the status of error injection for transmit FSM parity.<br>0b - Disabled<br>1b - Enabled |
| 15<br>— | Reserved |
| 14<br>— | Reserved |
| 13<br>— | Reserved |
| 12<br>PTEIN | PTP FSM Timeout Error Injection<br>Indicates the status of error injection for PTP FSM timeout.<br>0b - Disabled<br>1b - Enabled |
| 11<br>ATEIN | APP FSM Timeout Error Injection<br>Indicates the status of APP FSM timeout error injection.<br>0b - Disabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Enabled |
| 10<br>— | Reserved |
| 9<br>RTEIN | Receive FSM Timeout Error Injection<br>Indicates the status of receive FSM timeout error injection.<br>0b - Disabled<br>1b - Enabled |
| 8<br>TTEIN | Transmit FSM Timeout Error Injection<br>Indicates the status of transmit FSM timeout eror injection.<br>0b - Disabled<br>1b - Enabled |
| 7-2<br>— | Reserved |
| 1<br>PRTYEN | Parity Enable<br>Indicates whether the FSM parity feature is enabled.<br>0b - Disabled<br>1b - Enabled |
| 0<br>TMOUTEN | Time Out Enable<br>Indicates the status of the FSM timeout feature.<br>0b - Disabled<br>1b - Enabled |

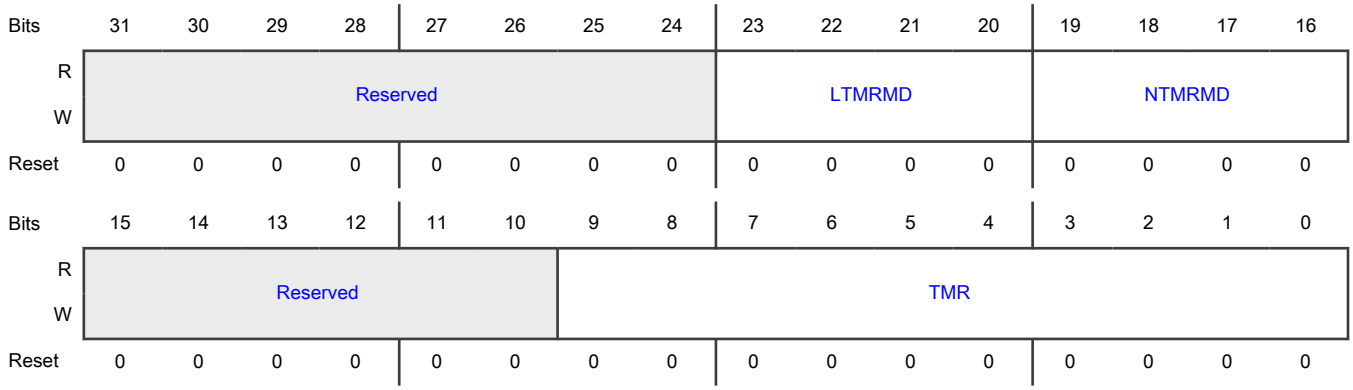
## 72.17.43 MAC FSM ACT Timer (MAC_FSM_ACT_Timer)

**Offset**

| Register | Offset |
|---|---|
| MAC_FSM_ACT_Timer | 14Ch |

**Function**
Is used to select the FSM and interface timeout values.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-20<br>LTMRMD | Large Mode Timeout Value<br>Provides the timeout duration value to be used for large mode FSM and other interface timeouts.<br>0000b - Timer disabled<br>0001b - 1 us<br>0010b - 1.024 ms (~4 ms)<br>0011b - 16.384 ms (~16 ms)<br>0100b - 65.536 ms (~64 ms)<br>0101b - 262.144 ms (~256 ms)<br>0110b - 1.048 sec (~1 sec)<br>0111b - 4.194 sec (~4sec)<br>1000b - 16.777 sec (~16 sec)<br>1001b - 33.554 sec (~32 sec)<br>1010b - 67.108 sec (~64 sec)<br>1011b - Reserved |
| 19-16<br>NTMRMD | Normal Mode Timeout Value<br>Provides the timeout duration value to be used for normal mode FSM and other interface timeouts.<br>0000b - Timer disabled<br>0001b - 1 us<br>0010b - 1.024 ms (~4 ms)<br>0011b - 16.384 ms (~16 ms) |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0100b - 65.536 ms (~64 ms)<br>0101b - 262.144 ms (~256 ms)<br>0110b - 1.048 sec (~1 sec)<br>0111b - 4.194 sec (~4 sec)<br>1000b - 16.777 sec (~16 sec)<br>1001b - 33.554 sec (~32 sec)<br>1010b - 67.108 sec (~64 sec)<br>1011b - Reserved |
| 15-10<br>— | Reserved |
| 9-0<br>TMR | CSR Clocks For 1 µs Tic<br>Indicates the number of CSR clocks required to generate 1 µs tic. |

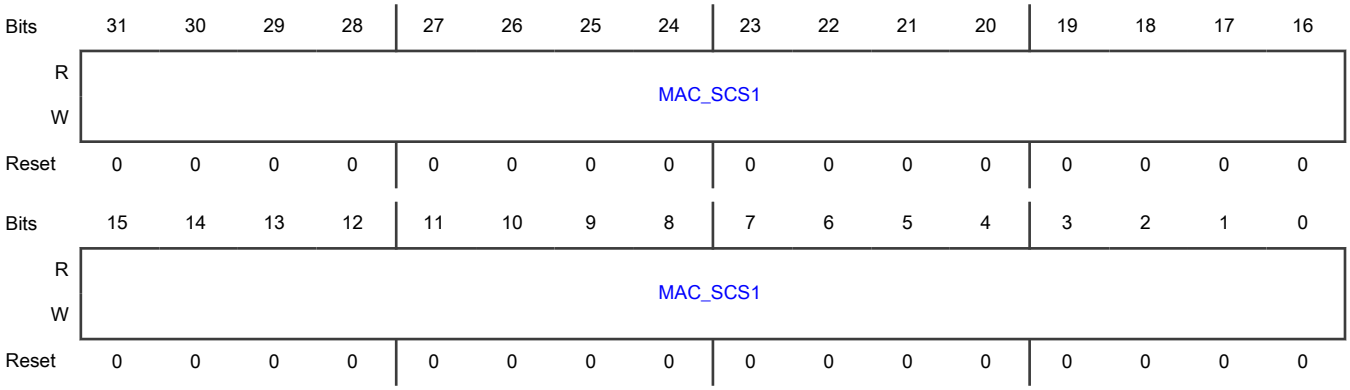
## 72.17.44 SCS_REG 1 (SCS_REG1)

### Offset

| Register | Offset |
|---|---|
| SCS_REG1 | 150h |

### Function
Is an NXP-reserved register for internal use.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-0<br>MAC_SCS1 | MAC SCS 1<br>Is reserved for NXP internal use.<br>The value of this field must always be 0 unless NXP instructs you otherwise. Writing 1 to any of the bits of this field might cause expected chip behavior. |

## 72.17.45 MAC MDIO Address (MAC_MDIO_Address)

### Offset

| Register | Offset |
|---|---|
| MAC_MDIO_Address | 200h |

### Function
Controls the management cycles to external PHY through a management interface.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | PSE | BTB | PA | | | | | | RDA | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | NTC | | | CR | | | | Reserved | | | SKAP | GOC_1 | GOC_0 | C45E | GB |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27<br>PSE | Preamble Suppression Enable<br>Indicates the status of preamble suppression.<br>• If this field is 1, SMA suppresses the 32-bit preamble and transmits MDIO frames with only 1 preamble bit.<br>• If this field is 0, the MDIO frame always includes 32 bits of preamble as defined in the IEEE specification. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |
| 26<br>BTB | Back-To-Back Transactions<br>Indicates the status of back-to-back transactions.<br>When BTB = 1 and NTC > 0, MAC indicates the completion of a read or write command at the end of a frame transfer (before the trailing clocks are transmitted). Therefore, you can initiate the next command, and that command executes immediately irrespective of the number of trailing clocks generated for the previous frame.<br>When BTB = 0, the read or write command completes (GB = 0) only after the trailing clocks generate. In this mode, you must ensure that:<br>• NTC is always generated after each frame.<br>• BTB must not be equal to 1 when NTC = 0.<br>0b - Disabled<br>1b - Enabled |
| 25-21<br>PA | Physical Layer Address<br>Indicates whether MAC accesses clause 22 PHY devices (out of 32) or clause 45 capable PHYs (out of 32). |
| 20-16<br>RDA | Register Or Device Address<br>Indicates whether MAC accesses clause 22 PHY devices (out of 32) or clause 45 capable PHYs (out of 32). |
| 15<br>— | Reserved |
| 14-12<br>NTC | Number Of Trailing Clocks<br>Controls the number of trailing clock cycles generated on the GMII_MDC_O (MDC) signal after the end of MDIO frame transmission. The valid values can range from 0 to 7.<br>Writing 3h to this field indicates that there are additional three clock cycles on the MDC line after the end of MDIO frame transfer. |
| 11-8<br>CR | CSR Clock Range<br>Determines the frequency of the MDC clock according to the CSR clock frequencies used in your design:<br>• 0000: CSR clock = 60-100 MHz; MDC clock = CSR clock/42<br>• 0001: CSR clock = 100-150 MHz; MDC clock = CSR clock/62<br>• 0010: CSR clock = 20-35 MHz; MDC clock = CSR clock/16<br>• 0011: CSR clock = 35-60 MHz; MDC clock = CSR clock/26<br>• 0100: CSR clock = 150-250 MHz; MDC clock = CSR clock/102<br>• 0101: CSR clock = 250-300 MHz; MDC clock = CSR clock/124 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • 0110: CSR clock = 300-500 MHz; MDC clock = CSR clock/204<br>• 0111: CSR clock = 500-800 MHz; MDC clock = CSR clock/324<br>When bit 3 of this field is 0, the suggested CSR clock-frequency range applicable to each value ensures that the MDC clock frequency is approximately between 1.0 MHz and 2.5 MHz.<br>When bit 3 of this field is 1, you can achieve a higher MDC clock frequency than the IEEE 802.3 frequency limit of 2.5 MHz, and program a clock divider of a lower value. For example, if the CSR clock frequency is 100 MHz and you write 1010 to this field, the resultant MDC clock frequency equals 12.5 MHz, which is above the range specified in IEEE 802.3. Program these values only if the interfacing chips support faster MDC clocks:<br>• 1000: CSR clock/4<br>• 1001: CSR clock/6<br>• 1010: CSR clock/8<br>• 1011: CSR clock/10<br>• 1100: CSR clock/12<br>• 1101: CSR clock/14<br>• 1110: CSR clock/16<br>• 1111: CSR clock/18<br>These bits are not used for accessing RevMII. They are read-only if the RevMII interface is selected as a single PHY interface. |
| 7-5<br>— | Reserved |
| 4<br>SKAP | Skip Address Packet<br>Indicates the status of the skip address packet.<br>If this field is 1, SMA does not send the address packets before read, write, or post-read increment address packets. This field is valid only when C45E = 1.<br>0b - Disabled<br>1b - Enabled |
| 3<br>GOC_1 | GMII Operation Command 1<br>Indicates the status of GMII operation command 1.<br>GMII operation command 1 represents the higher bit of the operation command to PHY or RevMII. The fields GOC_1 and GOC_O indicate these values:<br>• 00: Reserved<br>• 01: Write<br>• 10: Post-read increment address for clause 45 PHY<br>• 11: Read |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When clause 22 PHY or RevMII is enabled, only write and read commands are valid.<br>0b - Disabled<br>1b - Enabled |
| 2<br>GOC_0 | GMII Operation Command 0<br>Indicates the status of GMII operation command 0.<br>GMII operation command 0 represents the lower bit of the operation command to PHY or RevMII. When in SMA mode (MDIO master), this field, along with the GOC_1 field, determines the operation to be performed to PHY.<br>This field is read-only and tied to 1 when only RevMII is selected in configuration.<br>0b - Disabled<br>1b - Enabled |
| 1<br>C45E | Clause 45 PHY Enable<br>Indicates the status of clause 45 PHY.<br>• If this field is 1, clause 45 capable PHY is connected to MDIO.<br>• If this field is 0, clause 22 capable PHY is connected to MDIO.<br>0b - Disabled<br>1b - Enabled |
| 0<br>GB | GMII Busy<br>Indicates the status of GMII busy.<br>Writing 1 to this field instructs SMA to initiate a read or write access to the MDIO slave. MAC writes 0 to the field after the MDIO frame transfer is complete. Therefore, you must not write or change any of the fields in the MAC_MDIO_Address and MAC_MDIO_Data registers as long as the value of this field is 1.<br>For write transfers, before writing 1 to this field, you must write a 16-bit data to MAC_MDIO_Data[GD] and MAC_MDIO_Data[RA] if C45E = 1.<br>If C45E = 1, you must also write to MAC_MDIO_Data[RA] before initiating a read transfer.<br>After a read transfer completes (GB = 0), a data read from the PHY register is valid in MAC_MDIO_Data[GD].<br>**NOTE**<br>Even if the addressed PHY is not present, there is no change in the functionality of this field.<br>Access restrictions apply to this field. It clears automatically and writing 0 to it has no effect.<br>0b - Disabled<br>1b - Enabled |

## 72.17.46 MAC MDIO Data (MAC_MDIO_Data)

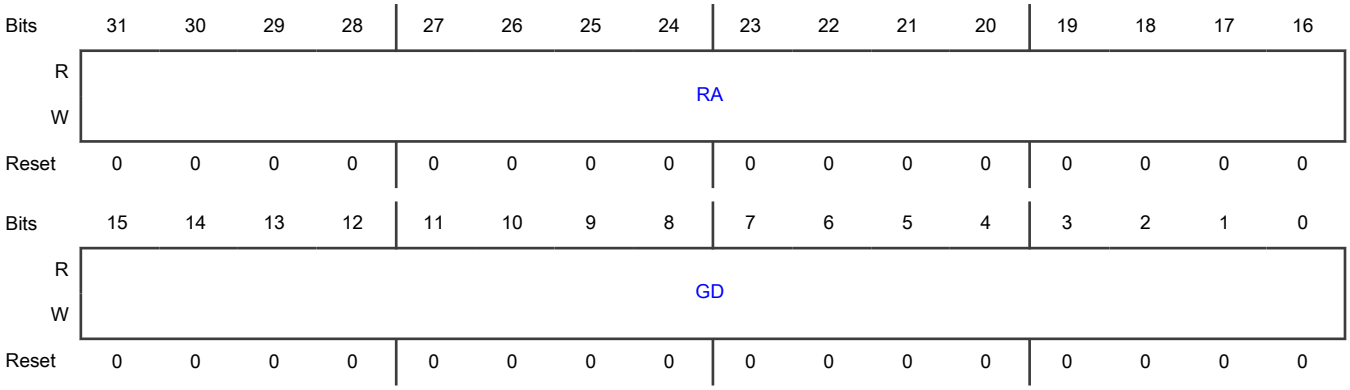
### Offset

| Register | Offset |
|---|---|
| MAC_MDIO_Data | 204h |

### Function

Is used to store the write data to be written to the PHY register located at the address specified in MAC MDIO Address (MAC_MDIO_Address). This register also stores the read data from the PHY register located at the address that MAC MDIO Address (MAC_MDIO_Address) specifies.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>RA | Register Address<br>Contains the PHY register address intended for the MDIO frame.<br>This field is valid only when MAC_MDIO_Address[C45E] = 1. |
| 15-0<br>GD | GMII Data<br>Contains the 16-bit data value read from PHY or RevMII after a management read operation or the 16-bit data value to be written to PHY or RevMII before a management write operation. |

## 72.17.47 MAC CSR Software Control (MAC_CSR_SW_Ctrl)

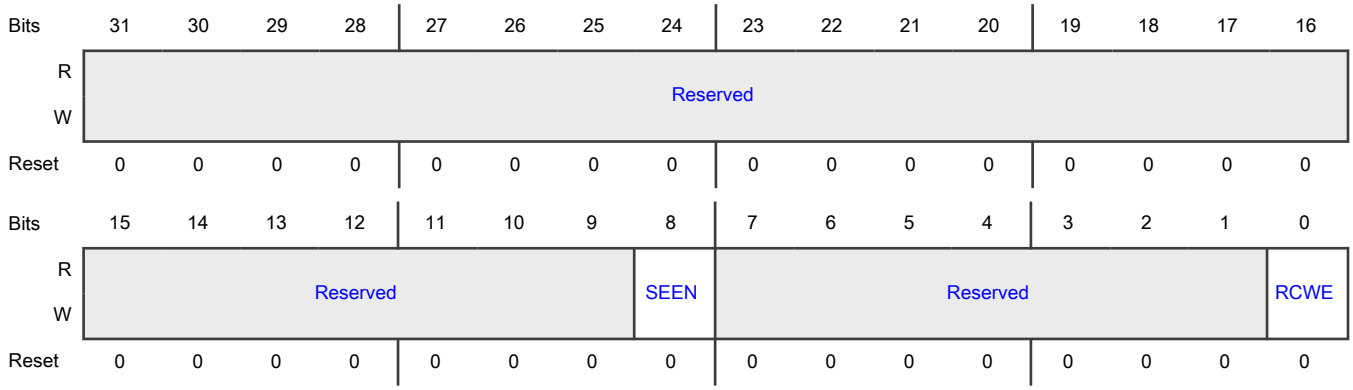
### Offset

| Register | Offset |
|---|---|
| MAC_CSR_SW_Ctrl | 230h |

### Function

Contains software programmable controls for changing the CSR access response and clearing the status bits on read.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>SEEN | Slave Error Response Enable<br>Indicates the status of the slave error response.<br>• If this field is 1, MAC responds with a slave error for accesses to reserved registers in the CSR space.<br>• If this field is 0, MAC sends an "okay" response to registers accessed from the CSR space.<br>0b - Disabled<br>1b - Enabled |
| 7-1<br>— | Reserved |
| 0<br>RCWE | Enable Register Write 1 To Clear (W1C)<br>Enables and disables the W1C feature for some registers.<br>When this field is 1, the access mode of some of the other fields changes from R2C to W1C.<br>When this field is 0, the access mode of those fields remains R2C.<br>0b - Disabled<br>1b - Enabled |

## 72.17.48 MAC FPE Control STS (MAC_FPE_CTRL_STS)

### Offset

| Register | Offset |
|---|---|
| MAC_FPE_CTRL_STS | 234h |

### Function

Controls the operation of frame preemption.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | TRSP | TVER | RRSP | RVER |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | S1_SET_0 | SRSP | SVER | EFPE |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved |
| 19<br>TRSP | Transmitted Respond Frame<br>Indicates whether the respond frame is transmitted.<br>This field becomes 1 when a respond frame is transmitted (triggered by writing 1 to the SRSP field). An interrupt can be generated for this event if you write 1 to MAC_Interrupt_Enable[FPEIE].<br>Access restrictions apply to this field. TRSP becomes a W1C field when MAC_CSR_SW_Ctrl[RCWE] = 1, and becomes an R2C field when MAC_CSR_SW_Ctrl[RCWE] = 0. Thus, TRSP becomes 0 sometimes when you read it and sometimes when you write 1 to it. It automatically becomes 1 on an internal event.<br>0b - Not transmitted<br>1b - Transmitted |
| 18<br>TVER | Transmitted Verify Frame<br>Indicates whether the verify frame is transmitted.<br>This field becomes 1 when a verify frame is transmitted (triggered by writing 1 to the SVER field). An interrupt can be generated for this event if you write 1 to MAC_Interrupt_Enable[FPEIE].<br>Access restrictions apply to this field. The field clears on read or write of 1 when MAC_CSR_SW_Ctrl[RCWE] = 1. It automatically becomes 1 on an internal event.<br>0b - Not transmitted<br>1b - Transmitted |
| 17<br>RRSP | Received Respond Frame<br>Indicates whether the respond frame is received. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | This field becomes 1 when a respond frame is received. An interrupt can be generated for this event if you write 1 to MAC_Interrupt_Enable[FPEIE].<br>Access restrictions apply to this field. The field clears on read or write of 1 when MAC_CSR_SW_Ctrl[RCWE] = 1. It automatically becomes 1 on an internal event.<br>0b - Not received<br>1b - Received |
| 16<br>RVER | Received Verify Frame<br>Indicates whether the verify frame is received.<br>This field becomes 1 when a verify frame is received. An interrupt can be generated on the transmission of a packet if you write 1 to MAC_Interrupt_Enable[FPEIE].<br>Access restrictions apply to this field. The field clears on read or write of 1 when MAC_CSR_SW_Ctrl[RCWE] = 1. It automatically becomes 1 on an internal event.<br>0b - Not received<br>1b - Received |
| 15-4<br>— | Reserved |
| 3<br>S1_SET_0 | S1 SET 0<br>Is reserved for NXP internal use and must always be 0 unless instructed by NXP. Writing 1 to this field may cause unexpected behavior. |
| 2<br>SRSP | Send Respond mPacket<br>Indicates the status of the send respond mPacket.<br>If this field is 1, it indicates the hardware to send a respond mPacket. The hardware resets the field after sending the respond mPacket.<br>Access restrictions apply to this field. It clears automatically and writing 0 to it has no effect.<br>0b - Disabled<br>1b - Enabled |
| 1<br>SVER | Send Verify mPacket<br>Enables and disables the sending of a verify mPacket.<br>Write 1 to this field to command EMAC to send a verify mPacket. EMAC writes 0 to this field after sending the verify mPacket.<br>Access restrictions apply to this field. It clears automatically and writing 0 to it has no effect.<br>0b - Disabled<br>1b - Enabled |
| 0 | Enable Transmit Frame Preemption |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| EFPE | Enables or disables the frame preemption transmit functionality.<br>0b - Disabled<br>1b - Enabled |

## 72.17.49 MAC Presentation Time (MAC_Presn_Time_ns)

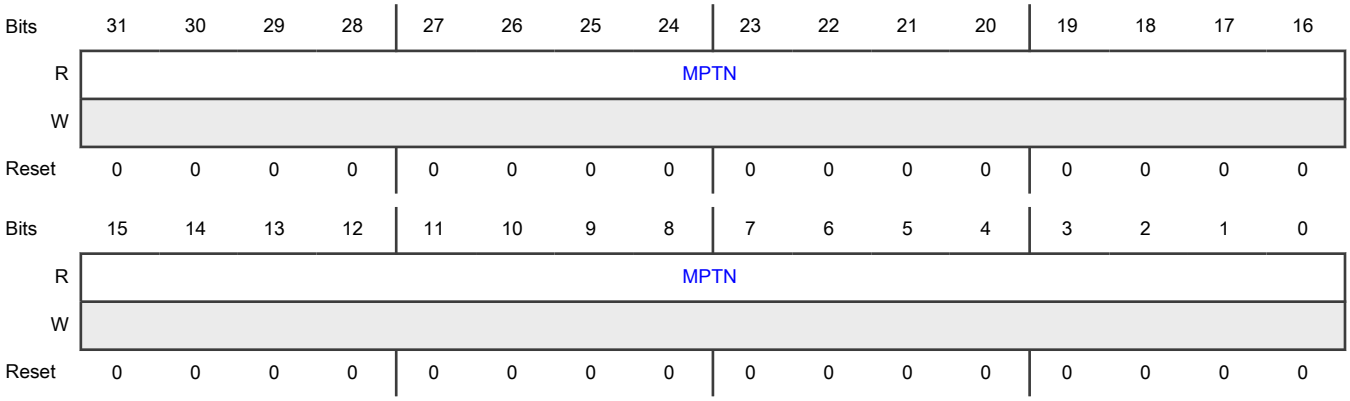
### Offset

| Register | Offset |
|---|---|
| MAC_Presn_Time_ns | 240h |

### Function

Contains the 32-bit binary rollover equivalent time of the PTP system time (in nanoseconds) and exists when DWC_EQOS_FLEXI_PPS_OUT_EN is configured.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>MPTN | MAC 1722 Presentation Time (In Nanoseconds)<br>Indicates the value of the 32-bit binary rollover equivalent time of the PTP System Time (in ns). |

## 72.17.50 MAC Presentation Time Update (MAC_Presn_Time_Updt)

### Offset

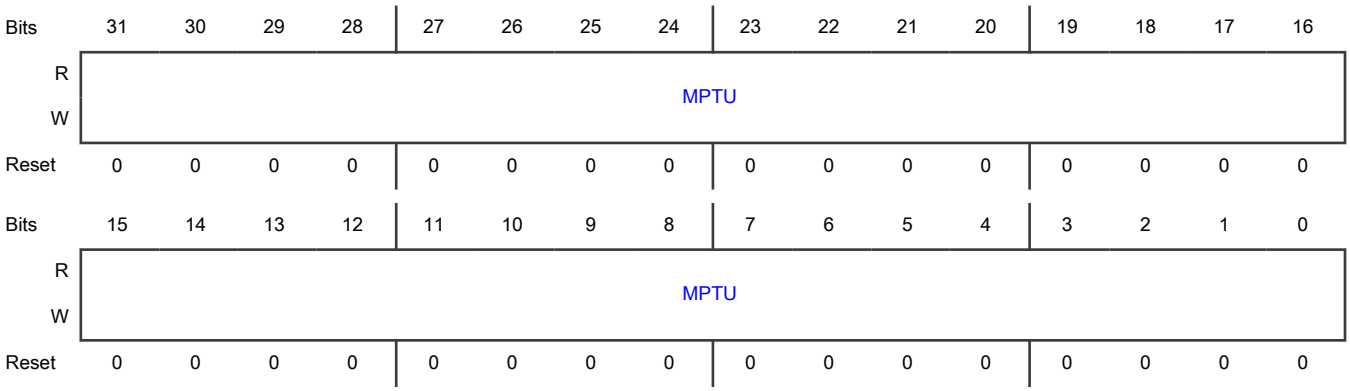
| Register | Offset |
|---|---|
| MAC_Presn_Time_Updt | 244h |

### Function

Holds the 32-bit value of MAC 1722 presentation time (in ns), which must be added to the current presentation time counter value. Initialization happens when MAC_Timestamp_Control[TSINIT] = 1, and an update happens when MAC_Timestamp_Control[TSUPDT] = 1.

Exists when DWC_EQOS_FLEXI_PPS_OUT_EN is configured.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>MPTU | MAC 1722 Presentation Time Update<br>Holds the initial value or the update value for the presentation time. When used for an update, this field holds the 32-bit value (in ns), which must be added to the current presentation time counter value. Initialization happens when MAC_Timestamp_Control[TSINIT] = 1, and an update happens when MAC_Timestamp_Control[TSUPDT] = 1.<br>When MAC_System_Time_Nanoseconds_Update[ADDSUB] = 1, this value is directly used for subtraction. |

## 72.17.51 MAC Address 0 High (MAC_Address0_High)

### Offset

| Register | Offset |
|---|---|
| MAC_Address0_High | 300h |

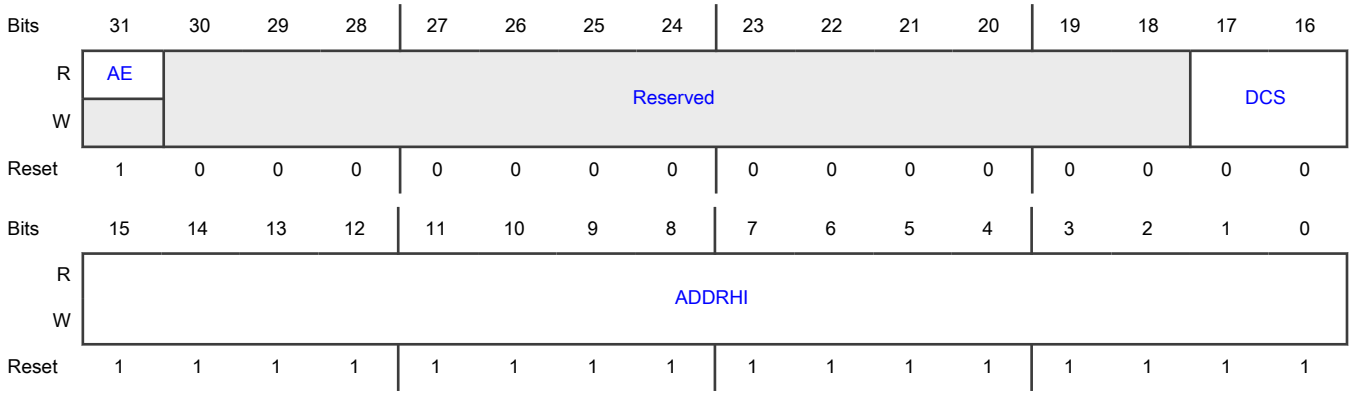
### Function

Holds the upper 16 bits of the first 6-byte MAC address of the station.

The first DA byte that is received on the (G)MII interface corresponds to the LS byte (bits [7:0]) of MAC Address 0 Low (MAC_Address0_Low). For example, if 112233445566h is received (11h in lane 0 of the first column) on (G)MII as the destination address, MAC Address 0 Low (MAC_Address0_Low)[47:0] is compared to 665544332211h.

If MAC address registers are configured to be double-synchronized with the (G)MII clock domains, the synchronization is triggered only when bits [31:24] (in Little-Endian mode) or bits [7:0] (in Big-Endian mode) of MAC Address 0 Low (MAC_Address0_Low) are written to. For proper synchronization updates, the consecutive writes to MAC Address 0 Low (MAC_Address0_Low) must be performed after at least four clock cycles in the destination clock domain.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>AE | Address Enable<br>Enables MAC address<br>0b - Disabled and invalid (the field's value must always be 1)<br>1b - Enabled |
| 30-18<br>— | Reserved |
| 17-16<br>DCS | DMA Channel Select<br>If MAC_Ext_Configuration[PDC] = 0, this field contains the binary representation of the DMA channel number to which a receive packet whose DA matches MAC Address 0 Low (MAC_Address0_Low) content is routed.<br>If MAC_Ext_Configuration[PDC] = 1, this field contains the one-hot representation of one or more DMA channel numbers to which a receive packet whose DA matches MAC Address 0 Low (MAC_Address0_Low) content is routed. |
| 15-0<br>ADDRHI | MAC Address 0 [47:32]<br>Contains the upper 16 bits [47:32] of the first 6-byte MAC address.<br>MAC uses this field for filtering the received packets and inserting the MAC address in the transmit flow control (pause) packets. |

## 72.17.52 MAC Address 0 Low (MAC_Address0_Low)

### Offset

| Register | Offset |
|---|---|
| MAC_Address0_Low | 304h |

### Function

Holds the lower 32 bits of the 6-byte first MAC address of the station.

### Diagram

### Fields

| Field | Function |
|---|---|
| 31-0<br>ADDRLO | MAC Address 0 [31:0]<br>Contains the lower 32 bits of the first 6-byte MAC address. MAC uses this field to filter the received packets and insert the MAC address in the transmit flow control (pause) packets. |

## 72.17.53 MAC Address 1 High (MAC_Address1_High)

### Offset

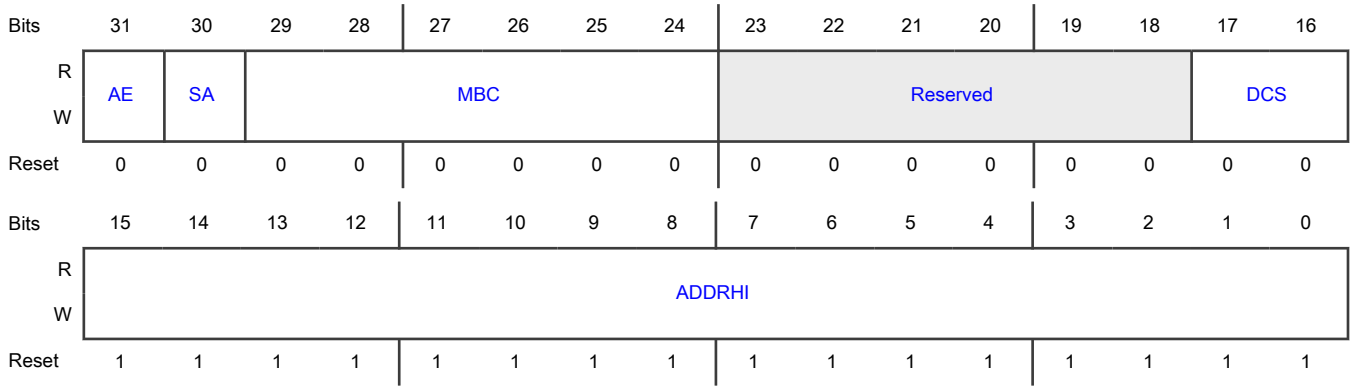
| Register | Offset |
|---|---|
| MAC_Address1_High | 308h |

### Function

Holds the upper 16 bits of the second 6-byte MAC address of the station.

If the MAC address registers are configured to be double-synchronized with the (G)MII clock domains, the synchronization is triggered only when bits [31:24] (in Little-Endian mode) or bits [7:0] (in Big-Endian mode) of MAC Address 1 Low (MAC_Address1_Low) are written to. For proper synchronization updates, the consecutive writes to MAC Address 1 Low (MAC_Address1_Low) must be performed after at least four clock cycles in the destination clock domain.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>AE | Address Enable<br>Indicates whether the MAC address is enabled or ignored.<br>If this field is 1, the address filter module uses the second MAC address for perfect filtering. If this field is 0, the address filter module ignores the address used for filtering.<br>0b - Ignored<br>1b - Enabled |
| 30<br>SA | Source Address<br>Indicates whether the MAC address is compared to the source or destination address.<br>If this field is 1, MAC Address 1[47:0] is compared to the SA fields of the received packet. If this field is 0, MAC Address 1[47:0] is compared to the DA fields of the received packet.<br>0b - Destination address<br>1b - Source address |
| 29-24<br>MBC | Mask Byte Control<br>Contains mask control bits for comparing each of the MAC address bytes. If this field is 1, MAC does not compare the corresponding byte of the received DA or SA with the contents of the MAC address 1 registers. Each bit controls the masking of the bytes as follows:<br>• Bit 29: MAC Address 1 High[15:8]<br>• Bit 28: MAC Address 1 High[7:0]<br>• Bit 27: MAC Address 1 Low[31:24]<br>• - ..<br>• Bit 24: MAC Address 1 Low[7:0]<br>You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. |
| 23-18 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 17-16<br>DCS | DMA Channel Select<br>If MAC_Ext_Configuration[PDC] = 0, this field contains the binary representation of the DMA channel number to which a receive packet whose DA matches MAC Address 1 Low (MAC_Address1_Low) content is routed.<br>If MAC_Ext_Configuration[PDC] = 1, this field contains the one-hot representation of one or more DMA Channel numbers to which a receive packet whose DA matches MAC Address 1 Low (MAC_Address1_Low) content is routed. |
| 15-0<br>ADDRHI | MAC Address 1 [47:32]<br>Contains the upper 16 bits [47:32] of the second 6-byte MAC address. |

## 72.17.54 MAC Address 1 Low (MAC_Address1_Low)

### Offset

| Register | Offset |
|---|---|
| MAC_Address1_Low | 30Ch |

### Function

Holds the lower 32 bits of the second 6-byte MAC address of the station.

### Diagram

### Fields

| Field | Function |
|---|---|
| 31-0 | MAC Address 1 [31:0]<br>Contains the lower 32 bits of the second 6-byte MAC address. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| ADDRLO | The contents of this field are undefined until you load them after the initialization process. |

## 72.17.55 MAC Address 2 High (MAC_Address2_High)

### Offset

| Register | Offset |
|---|---|
| MAC_Address2_High | 310h |

### Function

Holds the upper 16 bits of the second 6-byte MAC address of the station.

If the MAC address registers are configured to be double-synchronized with the (G)MII clock domains, the synchronization is triggered only when bits [31:24] (in Little-Endian mode) or bits [7:0] (in Big-Endian mode) of MAC Address 1 Low (MAC_Address1_Low) are written to. For proper synchronization updates, the consecutive writes to MAC Address 1 Low (MAC_Address1_Low) must be performed after at least four clock cycles in the destination clock domain.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | AE | SA | MBC | | | | | | Reserved | | | | | | DCS | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31<br>AE | Address Enable<br>Indicates whether the MAC address is enabled or ignored.<br>If this field is 1, the address filter module uses the second MAC address for perfect filtering. If this field is 0, the address filter module ignores the address for filtering.<br>0b - Ignored<br>1b - Enabled |
| 30<br>SA | Source Address<br>Indicates whether the MAC address is compared to the source or destination address. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | If this field is 1, MAC Address 1[47:0] is compared to the SA fields of the received packet. If this field is 0, MAC Address 1[47:0] is compared to the DA fields of the received packet.<br>0b - Destination address<br>1b - Source address |
| 29-24<br>MBC | Mask Byte Control<br>Contains mask control bits for comparing each of the MAC address bytes. If this field is 1, MAC does not compare the corresponding byte of the received DA or SA with the contents of MAC address 1 registers. Each bit controls the masking of the bytes as follows:<br>• Bit 29: MAC Address 1 High[15:8]<br>• Bit 28: MAC Address 1 High[7:0]<br>• Bit 27: MAC Address 1 Low[31:24]<br>• - ..<br>• Bit 24: MAC Address 1 Low[7:0]<br>You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. |
| 23-18<br>— | Reserved |
| 17-16<br>DCS | DMA Channel Select<br>If MAC_Ext_Configuration[PDC] = 0, this field contains the binary representation of the DMA channel number to which a receive packet whose DA matches MAC Address 2 Low (MAC_Address2_Low) content is routed.<br>If MAC_Ext_Configuration[PDC] = 1, this field contains the one-hot representation of one or more DMA Channel numbers to which a receive packet whose DA matches MAC Address 2 Low (MAC_Address2_Low) content is routed. |
| 15-0<br>ADDRHI | MAC Address 1 [47:32]<br>Contains the upper 16 bits [47:32] of the second 6-byte MAC address. |

## 72.17.56 MAC Address 2 Low (MAC_Address2_Low)

### Offset

| Register | Offset |
|---|---|
| MAC_Address2_Low | 314h |

### Function

Holds the lower 32 bits of the second 6-byte MAC address of the station.

### Diagram

### Fields

| Field | Function |
|---|---|
| 31-0<br>ADDRLO | MAC Address 1 [31:0]<br>Contains the lower 32 bits of the second 6-byte MAC address. The content of this field remains undefined until you load it after the initialization process. |

## 72.17.57 MMC Control (MMC_Control)

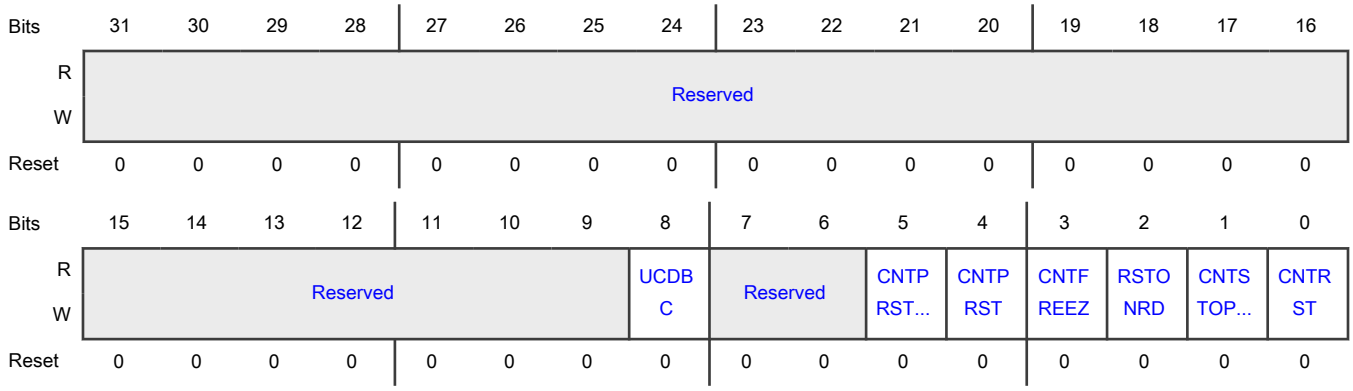
### Offset

| Register | Offset |
|---|---|
| MMC_Control | 700h |

### Function

Establishes the operating mode for MMC.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>UCDBC | Update MMC Counters For Dropped Broadcast Packets<br>Indicates the status of MMC counters for dropped broadcast packets.<br>• If this field is 1, MAC updates all the related MMC counters for broadcast packets that are dropped because MAC_Packet_Filter[DBF] = 1.<br>• If this field is 0, the MMC counters are not updated for dropped broadcast packets.<br>**NOTE**<br>The CNTRST field has a higher priority than the CNTPRST field. Therefore, if you write 1 to both these fields in the same write cycle, all the counters are cleared and the CNTPRST field becomes 0.<br>0b - Disabled<br>1b - Enabled |
| 7-6<br>— | Reserved |
| 5<br>CNTPRSTLVL | Full-Half Preset<br>Indicates whether full-half preset is enabled or disabled.<br>• If this field is 0 and the CNTPRST field is 1, all the MMC counters are preset to the almost-half value. All octet counters are preset to 7FFF_F800h (half 2 Kbytes) and all packet counters are preset to 7FFF_FFF0h (half 16).<br>• If this field is 1 and the CNTPRST field is 1 too, all MMC counters are preset to the almost-full value. All octet counters are preset to FFFF_F800h (full 2 Kbytes) and all packet counters are preset to FFFF_FFF0h (full 16).<br>For 16-bit counters, the almost-half preset values are 7800h and 7FF0h for the respective octet and packet counters. Similarly, the almost-full preset values for the 16-bit counters are 0xF800 and 0xFFF0.<br>0b - Disabled<br>1b - Enabled |
| 4<br>CNTPRST | Counters Preset<br>Indicates whether MMC counter preset is enabled or disabled.<br>If this field is 1, all the counters are initialized or preset to almost full or almost half according to the settings of the CNTPRSTLVL field.<br>This field clears automatically after 1 clock cycle, and along with the CNTPRSTLVL field, is useful for debugging and testing the assertion of interrupts because of the MMC counter becoming half-full or full.<br>Access restrictions apply to this field, which clears automatically.<br>0b - Disabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Enabled |
| 3<br>CNTFREEZ | MMC Counter Freeze<br>Indicates the status of MMC counter freeze.<br>If this field is 1, the field freezes all the MMC counters to their current values.<br>Until this field becomes 0, no MMC counter is updated because of any transmitted or received packets. If an MMC counter is read when RSTONRD = 1, that counter is also cleared in this mode.<br>0b - Disabled<br>1b - Enabled |
| 2<br>RSTONRD | Reset On Read<br>Indicates whether the reset on read for MMC counters is enabled or disabled.<br>If this field is 1, MMC counters are reset to 0 after read (they clear automatically after reset). The counters are cleared when the least significant byte lane (bits [7:0]) is read.<br>0b - Disabled<br>1b - Enabled |
| 1<br>CNTSTOPRO | Counter Stop Rollover<br>Indicates the status of counter stop rollover.<br>If this field is 1, the counter does not roll over to 0 after reaching the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 0<br>CNTRST | Counters Reset<br>Indicates whether MMC counters are reset.<br>If this field is 1, all the counters are reset. This field clears automatically after 1 clock cycle.<br>Access restrictions apply to this field that clears automatically.<br>0b - Counters are not reset<br>1b - All counters are reset |

## 72.17.58 MMC Receive Interrupt (MMC_Rx_Interrupt)

### Offset

| Register | Offset |
|---|---|
| MMC_Rx_Interrupt | 704h |

### Function

Maintains the interrupts generated from all receive statistics counters when the following conditions occur:

- Receive statistic counters reach half of their maximum values (8000_0000h for a 32-bit counter and 8000h for a 16-bit counter).
- Receive statistic counters cross their maximum values (FFFF_FFFFh for a 32-bit counter and FFFFh for a 16-bit counter).

If MMC_Control[CNTSTOPRO] = 1, interrupts are set but the counter remains at all-ones. This is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (bits [7:0]) of the respective counter must be read to clear the interrupt bit.

**NOTE**

R_SS_RC means that this register bit is set internally, and is cleared when the counter register is read.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | Reserved | Reserved | RXCTRLP... | RXRCVER... | RXWDOGP... | RXVLANG... | RXFOVPIS | RXPAUSP... | RXORANG... | RXLENER... | RXUCGPIS | RX1024T... |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RX512T1... | RX256T5... | RX128T2... | RX65T12... | RX64OCT... | RXOSIZE... | RXUSIZE... | RXJABER... | RXRUNTP... | RXALGNE... | RXCRCER... | RXMCGPIS | RXBCGPIS | RXGOCTIS | RXGBOCT... | RXGBPKT... |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27<br>— | Reserved |
| 26<br>— | Reserved |
| 25<br>RXCTRLPIS | MMC Receive Control Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive control packet counter interrupt is detected.<br>This field becomes 1 when the rxctrlpackets_g counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 24<br>RXRCVERRPIS | MMC Receive Error Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive error packet counter interrupt is detected.<br>This field becomes 1 when the rxrcverror counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 23<br>RXWDOGPIS | MMC Receive Watchdog Error Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive watchdog error packet counter interrupt is detected.<br>This field becomes 1 when the rxwatchdog error counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 22<br>RXVLANGBPIS | MMC Receive VLAN Good Bad Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive VLAN good bad packet counter interrupt is detected.<br>This field becomes 1 when the rxvlanpackets_gb counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 21<br>RXFOVPIS | MMC Receive FIFO Overflow Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive FIFO overflow packet counter interrupt is detected.<br>This field becomes 1 when the rxfifooverflow counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 20<br>RXPAUSPIS | MMC Receive Pause Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive pause packet counter interrupt is detected.<br>This field becomes 1 when the rxpausepackets counter reaches half of its maximum value or the maximum value. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 19<br>RXORANGEPIS | MMC Receive Out-Of-Range Error Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive out-of-range error packet counter interrupt is detected.<br>This field becomes 1 when the rxoutofrangetype counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 18<br>RXLENERPIS | MMC Receive Length Error Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive length error packet counter interrupt is detected.<br>This field becomes 1 when the rxlengtherror counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 17<br>RXUCGPIS | MMC Receive Unicast Good Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive unicast good packet counter interrupt is detected.<br>This field becomes 1 when the rxunicastpackets_g counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 16<br>RX1024TMAXOCTGBPIS | MMC Receive 1024 To Maximum Octet Good Bad Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive 1024 to maximum octet good bad packet counter interrupt is detected.<br>This field becomes 1 when the rx1024tomaxoctets_gb counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Not detected<br>1b - Detected |
| 15<br>RX512T1023OCTGBPIS | MMC Receive 512 To 1023 Octet Good Bad Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive 512 to 1023 octet good bad packet counter interrupt is detected.<br>This field becomes 1 when the rx512to1023octets_gb counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 14<br>RX256T511OCTGBPIS | MMC Receive 256 To 511 Octet Good Bad Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive 256 to 511 octet good bad packet counter interrupt is detected.<br>This field becomes 1 when the rx256to511octets_gb counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 13<br>RX128T255OCTGBPIS | MMC Receive 128 To 255 Octet Good Bad Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive 128 to 255 octet good bad packet counter interrupt is detected.<br>This field becomes 1 when the rx128to255octets_gb counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 12<br>RX65T127OCTGBPIS | MMC Receive 65 To 127 Octet Good Bad Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive 65 to 127 octet good bad packet counter interrupt is detected.<br>The field becomes 1 when the rx65to127octets_gb counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Not detected<br>1b - Detected |
| 11<br>RX64OCTGBPIS | MMC Receive 64 Octet Good Bad Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive 64-octet good bad packet counter interrupt is detected.<br>The field becomes 1 when the rx64octets_gb counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 10<br>RXOSIZEGPIS | MMC Receive Oversize Good Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive oversize good packet counter interrupt is detected.<br>This field becomes 1 when the rxoversize_g counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 9<br>RXUSIZEGPIS | MMC Receive Undersize Good Packet Counter Interrupt Status<br>Indicates whether the status of MMC receive undersize good packet counter interrupt is detected.<br>This field becomes 1 when the rxundersize_g counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 8<br>RXJABERPIS | MMC Receive Jabber Error Packet Counter Interrupt Status<br>Indicates whether the status of MMC receive jabber error packet counter interrupt is detected.<br>This field becomes 1 when the rxjabbererror counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 7<br>RXRUNTPIS | MMC Receive Runt Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive runt packet counter interrupt is detected.<br>This field becomes 1 when the rxrunterror counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 6<br>RXALGNERPIS | MMC Receive Alignment Error Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive alignment error packet counter interrupt is detected.<br>This field becomes 1 when the rxalignmenterror counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 5<br>RXCRCERPIS | MMC Receive CRC Error Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive CRC error packet counter interrupt is detected.<br>This field becomes 1 when the rxcrcerror counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 4<br>RXMCGPIS | MMC Receive Multicast Good Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive multicast good packet counter interrupt is detected.<br>This field becomes 1 when the rxmulticastpackets_g counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 3<br>RXBCGPIS | MMC Receive Broadcast Good Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive broadcast good packet counter interrupt is detected.<br>This field becomes 1 when the rxbroadcastpackets_g counter reaches half of its maximum value or the maximum value. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 2<br>RXGOCTIS | MMC Receive Good Octet Counter Interrupt Status<br>Indicates whether the status of the MMC receive good octet counter interrupt is detected.<br>This field becomes 1 when the rxoctetcount_g counter reaches half of its maximum value or its maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 1<br>RXGBOCTIS | MMC Receive Good Bad Octet Counter Interrupt Status<br>Indicates whether the status of the MMC receive good bad octet counter interrupt is detected.<br>This field becomes 1 when the rxoctetcount_gb counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 0<br>RXGBPKTIS | MMC Receive Good Bad Packet Counter Interrupt Status<br>Indicates whether the status of the MMC receive good bad packet counter interrupt is detected.<br>This field becomes 1 when the rxpacketcount_gb counter reaches half of its maximum value or its maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |

## 72.17.59 MMC Transmit Interrupt (MMC_Tx_Interrupt)

**Offset**

| Register | Offset |
|---|---|
| MMC_Tx_Interrupt | 708h |

### Function

Maintains the interrupts generated from all the transmit statistics counters.

This register maintains those interrupts that are generated when transmit statistic counters:

- Reach half of their maximum values (8000_0000h for a 32-bit counter and 0x8000 for a 16-bit counter)
- Cross their maximum values (FFFF_FFFFh for a 32-bit counter and FFFFh for a 16-bit counter)

If MMC_Control[CNTSTOPRO] = 1, the interrupts are set but the counter remains at all-ones.

This is a 32 bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read.

The least significant byte lane (bits [7:0]) of the respective counter must be read to clear the interrupt bit.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | Reserved | Reserved | TXOSIZE... | TXVLANG... | TXPAUSP... | TXEXDEF... | TXGPKTIS | TXGOCTIS | TXCARER... | TXEXCOL... | TXLATCO... | TXDEFPIS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TXMCOLG... | TXSCOLG... | TXUFLOW... | TXBCGBP... | TXMCGBP... | TXUCGBP... | TX1024T... | TX512T1... | TX256T5... | TX128T2... | TX65T12... | TX64OCT... | TXMCGPIS | TXBCGPIS | TXGBPKT... | TXGBOCT... |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27<br>— | Reserved |
| 26<br>— | Reserved |
| 25<br>TXOSIZEGPIS | MMC Transmit Oversize Good Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit oversize good packet counter interrupt is detected.<br>This field becomes 1 when the txoversize_g counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Detected |
| 24<br>TXVLANGPIS | MMC Transmit VLAN Good Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit VLAN good packet counter interrupt is detected.<br>This field becomes 1 when the txvlanpackets_g counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 23<br>TXPAUSPIS | MMC Transmit Pause Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit pause packet counter interrupt is detected.<br>This field becomes 1 when the txpausepacketserror counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 22<br>TXEXDEFPIS | MMC Transmit Excessive Deferral Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit excessive deferral packet counter interrupt is detected.<br>This field becomes 1 when the txexcessdef counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 21<br>TXGPKTIS | MMC Transmit Good Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit good packet counter interrupt is detected.<br>This field becomes 1 when the txpacketcount_g counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 20 | MMC Transmit Good Octet Counter Interrupt Status |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| TXGOCTIS | Indicates whether the status of the MMC transmit good octet counter interrupt is detected.<br>This field becomes 1 when the txoctetcount_g counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 19<br>TXCARERPIS | MMC Transmit Carrier Error Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit carrier error packet counter interrupt is detected.<br>This field becomes 1 when the txcarriererror counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 18<br>TXEXCOLPIS | MMC Transmit Excessive Collision Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit excessive collision packet counter interrupt is detected.<br>This field becomes 1 when the txexesscol counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 17<br>TXLATCOLPIS | MMC Transmit Late Collision Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit late collision packet counter interrupt is detected.<br>This field becomes 1 when the txlatecol counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 16<br>TXDEFPIS | MMC Transmit Deferred Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit deferred packet counter interrupt is detected.<br>This field becomes 1 when the txdeferred counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Not detected<br>1b - Detected |
| 15<br>TXMCOLGPIS | MMC Transmit Multiple Collision Good Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit multiple collision good packet counter interrupt is detected.<br>This field becomes 1 when the txmulticol_g counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 14<br>TXSCOLGPIS | MMC Transmit Single Collision Good Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit single collision good packet counter interrupt is detected.<br>This field becomes 1 when the txsinglecol_g counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 13<br>TXUFLOWERPIS | MMC Transmit Underflow Error Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit underflow error packet counter interrupt is detected.<br>This field becomes 1 when the txunderflowerror counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 12<br>TXBCGBPIS | MMC Transmit Broadcast Good Bad Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit broadcast good bad packet counter interrupt is detected.<br>This field becomes 1 when the txbroadcastpackets_gb counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 11<br>TXMCGBPIS | MMC Transmit Multicast Good Bad Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit multicast good bad packet counter interrupt is detected.<br>This field becomes 1 when the txmulticastpackets_gb counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 10<br>TXUCGBPIS | MMC Transmit Unicast Good Bad Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit unicast good bad packet counter interrupt is detected.<br>This field becomes 1 when the txunicastpackets_gb counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 9<br>TX1024TMAXOCTGBPIS | MMC Transmit 1024 To Maximum Octet Good Bad Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit 1024 to maximum octet good bad packet counter interrupt is detected.<br>This field becomes 1 when the tx1024tomaxoctets_gb counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 8<br>TX512T1023OCTGBPIS | MMC Transmit 512 To 1023 Octet Good Bad Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit 512 to 1023 octet good bad packet counter interrupt is detected.<br>This field becomes 1 when the tx512to1023octets_gb counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 7 | MMC Transmit 256 To 511 Octet Good Bad Packet Counter Interrupt Status |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| TX256T511OCTGBPIS | Indicates whether the status of the MMC transmit 256 to 511 octet good bad packet counter interrupt is detected.<br>This field becomes 1 when the tx256to511octets_gb counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 6<br>TX128T255OCTGBPIS | MMC Transmit 128 To 255 Octet Good Bad Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit 128 to 255 octet good bad packet counter interrupt is detected.<br>This field becomes 1 when the tx128to255octets_gb counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 5<br>TX65T127OCTGBPIS | MMC Transmit 65 To 127 Octet Good Bad Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit 65 to 127 octet good bad packet counter interrupt is detected.<br>This field becomes 1 when the tx65to127octets_gb counter reaches half of its maximum value, and also when it reaches the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 4<br>TX64OCTGBPIS | MMC Transmit 64-Octet Good Bad Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit 64-octet good bad packet counter interrupt is detected.<br>This field becomes 1 when the tx64octets_gb counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 3<br>TXMCGPIS | MMC Transmit Multicast Good Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit multicast good packet counter interrupt is detected. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | This field becomes 1 when the txmulticastpackets_g counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 2<br>TXBCGPIS | MMC Transmit Broadcast Good Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit broadcast good packet counter interrupt is detected.<br>This field becomes 1 when the txbroadcastpackets_g counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 1<br>TXGBPKTIS | MMC Transmit Good Bad Packet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit good bad packet counter interrupt is detected.<br>This field becomes 1 when the txpacketcount_gb counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 0<br>TXGBOCTIS | MMC Transmit Good Bad Octet Counter Interrupt Status<br>Indicates whether the status of the MMC transmit good bad octet counter interrupt is detected.<br>This field becomes 1 when the txoctetcount_gb counter reaches half of its maximum value or the maximum value.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |

## 72.17.60 MMC Receive Interrupt Mask (MMC_Rx_Interrupt_Mask)

**Offset**

| Register | Offset |
|---|---|
| MMC_Rx_Interrupt_Mask | 70Ch |

### Function

Maintains the masks for interrupts generated from all the receive statistic counters. These interrupts are generated when the receive statistic counters reach half of their maximum values or the maximum values.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | Reserved | Reserved | RXCTRLP... | RXRCVER... | RXWDOGP... | RXVLANG... | RXFOVPIM | RXPAUSP... | RXORANG... | RXLENER... | RXUCGPIM | RX1024T... |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RX512T1... | RX256T5... | RX128T2... | RX65T12... | RX64OCT... | RXOSIZE... | RXUSIZE... | RXJABER... | RXRUNTP... | RXALGNE... | RXCRCER... | RXMCGPIM | RXBCGPIM | RXGOCTIM | RXGBOCT... | RXGBPKT... |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27<br>— | Reserved |
| 26<br>— | Reserved |
| 25<br>RXCTRLPIM | MMC Receive Control Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive control packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rxctrlpackets_g counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 24<br>RXRCVERRPIM | MMC Receive Error Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive error packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rxrcverror counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 23 | MMC Receive Watchdog Error Packet Counter Interrupt Mask |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| RXWDOGPIM | Indicates the status of the MMC receive watchdog error packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rxwatchdog counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 22<br>RXVLANGBPIM | MMC Receive VLAN Good Bad Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive VLAN good bad packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rxvlanpackets_gb counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 21<br>RXFOVPIM | MMC Receive FIFO Overflow Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive FIFO overflow packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rxfifooverflow counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 20<br>RXPAUSPIM | MMC Receive Pause Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive pause packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rxpausepackets counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 19<br>RXORANGEPIM | MMC Receive Out-Of-Range Error Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive out-of-range error packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rxoutofrangetype counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 18<br>RXLENERPIM | MMC Receive Length Error Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive length error packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rxlengtherror counter reaches half of its maximum value or the maximum value. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |
| 17<br>RXUCGPIM | MMC Receive Unicast Good Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive unicast good packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rxunicastpackets_g counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 16<br>RX1024TMAXOCTGBPIM | MMC Receive 1024 To Maximum Octet Good Bad Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive 1024 to maximum octet good bad packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rx1024tomaxoctets_gb counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 15<br>RX512T1023OCTGBPIM | MMC Receive 512 To 1023 Octet Good Bad Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive 512 to 1023 octet good bad packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rx512to1023octets_gb counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 14<br>RX256T511OCTGBPIM | MMC Receive 256 To 511 Octet Good Bad Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive 256 to 511 octet good bad packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rx256to511octets_gb counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 13<br>RX128T255OCTGBPIM | MMC Receive 128 To 255 Octet Good Bad Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive 128 to 255 octet good bad packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rx128to255octets_gb counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 12 | MMC Receive 65 To 127 Octet Good Bad Packet Counter Interrupt Mask |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| RX65T127OCTGBPIM | Indicates the status of the MMC receive 65 to 127 octet good bad packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rx65to127octets_gb counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 11<br>RX64OCTGBPIM | MMC Receive 64-Octet Good Bad Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive 64-octet good bad packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rx64octets_gb counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 10<br>RXOSIZEGPIM | MMC Receive Oversize Good Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive oversize good packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rxoversize_g counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 9<br>RXUSIZEGPIM | MMC Receive Undersize Good Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive undersize good packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rxundersize_g counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 8<br>RXJABERPIM | MMC Receive Jabber Error Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive jabber error packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rxjabbererror counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 7<br>RXRUNTPIM | MMC Receive Runt Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive runt packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rxrunterror counter reaches half of its maximum value or the maximum value. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |
| 6<br>RXALGNERPIM | MMC Receive Alignment Error Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive alignment error packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rxalignmenterror counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 5<br>RXCRCERPIM | MMC Receive CRC Error Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive CRC error packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rxcrcerror counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 4<br>RXMCGPIM | MMC Receive Multicast Good Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive multicast good packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rxmulticastpackets_g counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 3<br>RXBCGPIM | MMC Receive Broadcast Good Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive broadcast good packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rxbroadcastpackets_g counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 2<br>RXGOCTIM | MMC Receive Good Octet Counter Interrupt Mask<br>Indicates the status of the MMC receive good octet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rxoctetcount_g counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 1 | MMC Receive Good Bad Octet Counter Interrupt Mask |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| RXGBOCTIM | Indicates the status of the MMC receive good bad octet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rxoctetcount_gb counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 0<br>RXGBPKTIM | MMC Receive Good Bad Packet Counter Interrupt Mask<br>Indicates the status of the MMC receive good bad packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the rxpacketcount_gb counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |

## 72.17.61 MMC Transmit Interrupt Mask (MMC_Tx_Interrupt_Mask)

### Offset

| Register | Offset |
|---|---|
| MMC_Tx_Interrupt_Mask | 710h |

### Function

Maintains the masks for interrupts that are generated when the transmit statistic counters reach half of their maximum values or their maximum values.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | Reserved | Reserved | TXOSIZE... | TXVLANG... | TXPAUSP... | TXEXDEF... | TXGPKTIM | TXGOCTIM | TXCARER... | TXEXCOL... | TXLATCO... | TXDEFPIM |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TXMCOLG... | TXSCOLG... | TXUFLOW... | TXBCGBP... | TXMCGBP... | TXUCGBP... | TX1024T... | TX512T1... | TX256T5... | TX128T2... | TX65T12... | TX64OCT... | TXMCGPIM | TXBCGPIM | TXGBPKT... | TXGBOCT... |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27<br>— | Reserved |
| 26<br>— | Reserved |
| 25<br>TXOSIZEGPIM | MMC Transmit Oversize Good Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit oversize good packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the txoversize_g counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 24<br>TXVLANGPIM | MMC Transmit VLAN Good Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit VLAN good packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the txvlanpackets_g counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 23<br>TXPAUSPIM | MMC Transmit Pause Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit pause packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the txpausepackets counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 22<br>TXEXDEFPIM | MMC Transmit Excessive Deferral Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit excessive deferral packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the txexcessdef counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 21<br>TXGPKTIM | MMC Transmit Good Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit good packet counter interrupt mask. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Writing 1 to this field masks the interrupt when the txpacketcount_g counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 20<br>TXGOCTIM | MMC Transmit Good Octet Counter Interrupt Mask<br>Indicates the status of the MMC transmit good octet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the txoctetcount_g counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 19<br>TXCARERPIM | MMC Transmit Carrier Error Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit carrier error packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the txcarriererror counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 18<br>TXEXCOLPIM | MMC Transmit Excessive Collision Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit excessive collision packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the txexcesscol counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 17<br>TXLATCOLPIM | MMC Transmit Late Collision Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit late collision packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the txlatecol counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 16<br>TXDEFPIM | MMC Transmit Deferred Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit deferred packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the txdeferred counter reaches half of the maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 15<br>TXMCOLGPIM | MMC Transmit Multiple Collision Good Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit multiple collision good packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the txmulticol_g counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 14<br>TXSCOLGPIM | MMC Transmit Single Collision Good Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit single collision good packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the txsinglecol_g counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 13<br>TXUFLOWERPIM | MMC Transmit Underflow Error Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit underflow error packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the txunderflowerror counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 12<br>TXBCGBPIM | MMC Transmit Broadcast Good Bad Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit broadcast good bad packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the txbroadcastpackets_gb counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 11<br>TXMCGBPIM | MMC Transmit Multicast Good Bad Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit multicast good bad packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the txmulticastpackets_gb counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 10<br>TXUCGBPIM | MMC Transmit Unicast Good Bad Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit unicast good bad packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the txunicastpackets_gb counter reaches half of its maximum value or the maximum value. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |
| 9<br>TX1024TMAXOCTGBPIM | MMC Transmit 1024 To Maximum Octet Good Bad Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit 1024 to maximum octet good bad packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the tx1024tomaxoctets_gb counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 8<br>TX512T1023OCTGBPIM | MMC Transmit 512 To 1023 Octet Good Bad Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit 512 to 1023 octet good bad packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the tx512to1023octets_gb counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 7<br>TX256T511OCTGBPIM | MMC Transmit 256 To 511 Octet Good Bad Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit 256 to 511 octet good bad packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the tx256to511octets_gb counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 6<br>TX128T255OCTGBPIM | MMC Transmit 128 To 255 Octet Good Bad Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit 128 to 255 octet good bad packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the tx128to255octets_gb counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 5<br>TX65T127OCTGBPIM | MMC Transmit 65 To 127 Octet Good Bad Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit 65 to 127 octet good bad packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the tx65to127octets_gb counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 4 | MMC Transmit 64-Octet Good Bad Packet Counter Interrupt Mask |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| TX64OCTGBPIM | Indicates the status of the MMC transmit 64-octet good bad packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the tx64octets_gb counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 3<br>TXMCGPIM | MMC Transmit Multicast Good Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit multicast good packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the txmulticastpackets_g counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 2<br>TXBCGPIM | MMC Transmit Broadcast Good Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit broadcast good packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the txbroadcastpackets_g counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 1<br>TXGBPKTIM | MMC Transmit Good Bad Packet Counter Interrupt Mask<br>Indicates the status of the MMC transmit good bad packet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the txpacketcount_gb counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |
| 0<br>TXGBOCTIM | MMC Transmit Good Bad Octet Counter Interrupt Mask<br>Indicates the status of the MMC transmit good bad octet counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the txoctetcount_gb counter reaches half of its maximum value or the maximum value.<br>0b - Disabled<br>1b - Enabled |

## 72.17.62 Transmit Octet Count Good Bad (Tx_Octet_Count_Good_Bad)

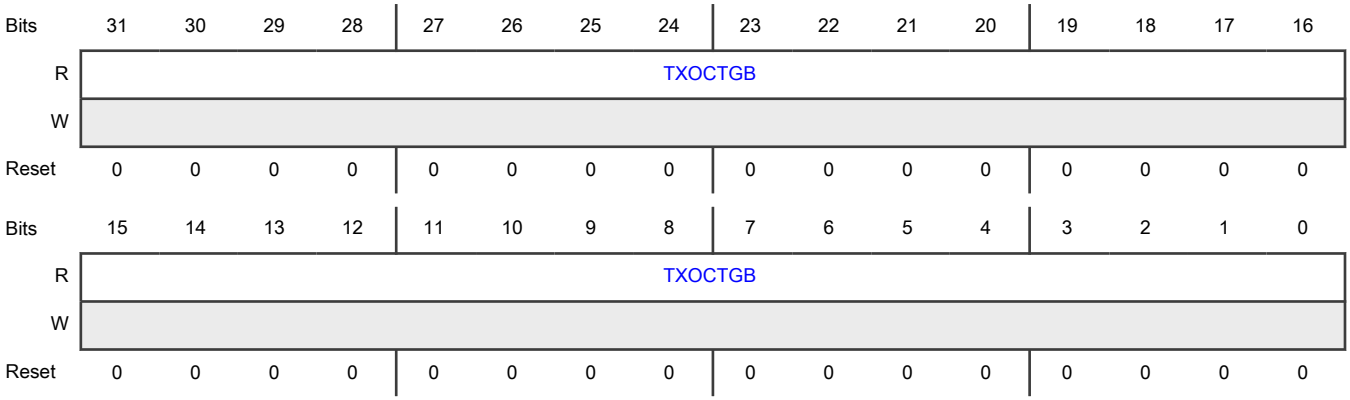
### Offset

| Register | Offset |
|---|---|
| Tx_Octet_Count_Good_Bad | 714h |

### Function

Provides the number of bytes that the module transmitted, exclusive of preamble and retried bytes, in good and bad packets.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>TXOCTGB | Transmit Octet Count Good Bad<br>Indicates the number of bytes transmitted, exclusive of preamble and retried bytes, in good and bad packets. |

## 72.17.63 Transmit Packet Count Good Bad (Tx_Packet_Count_Good_Bad)

### Offset

| Register | Offset |
|---|---|
| Tx_Packet_Count_Good_Bad | 718h |

### Function

Provides the number of good and bad packets that the module transmitted, exclusive of retried packets.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>TXPKTGB | Transmit Packet Count Good Bad<br>Indicates the number of good and bad packets transmitted, exclusive of retried packets. |

## 72.17.64 Transmit Broadcast Packets Good (Tx_Broadcast_Packets_Good)

### Offset

| Register | Offset |
|---|---|
| Tx_Broadcast_Packets_Good | 71Ch |

### Function

Provides the number of good broadcast packets that the module transmitted.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>TXBCASTG | Transmit Broadcast Packets Good<br>Indicates the number of good broadcast packets transmitted. |

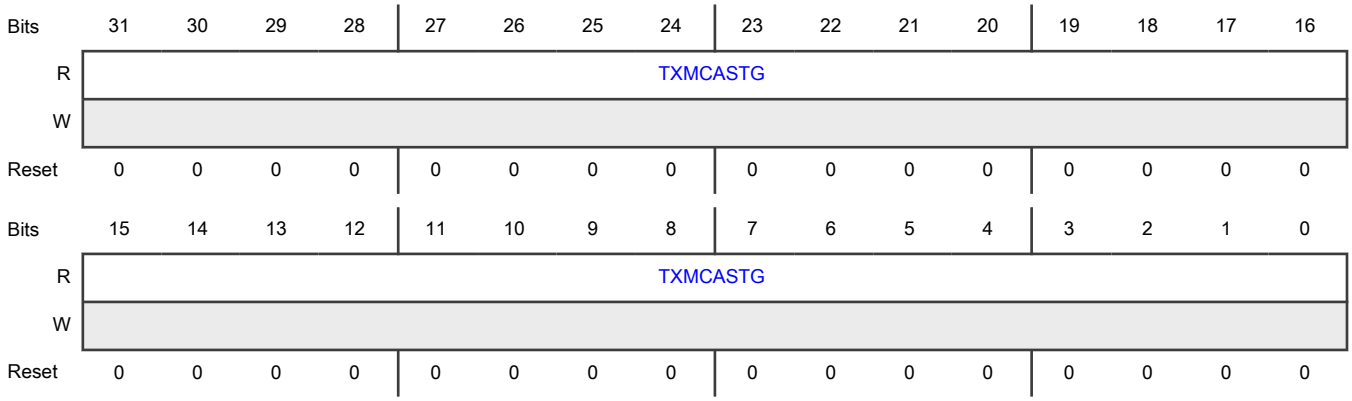
## 72.17.65 Transmit Multicast Packets Good (Tx_Multicast_Packets_Good)

### Offset

| Register | Offset |
|---|---|
| Tx_Multicast_Packets_Good | 720h |

### Function
Provides the number of good multicast packets that the module transmitted.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>TXMCASTG | Transmit Multicast Packets Good<br>Indicates the number of good multicast packets transmitted. |

## 72.17.66 Transmit 64-Octet Packets Good Bad (Tx_64Octets_Packets_Good_Bad)

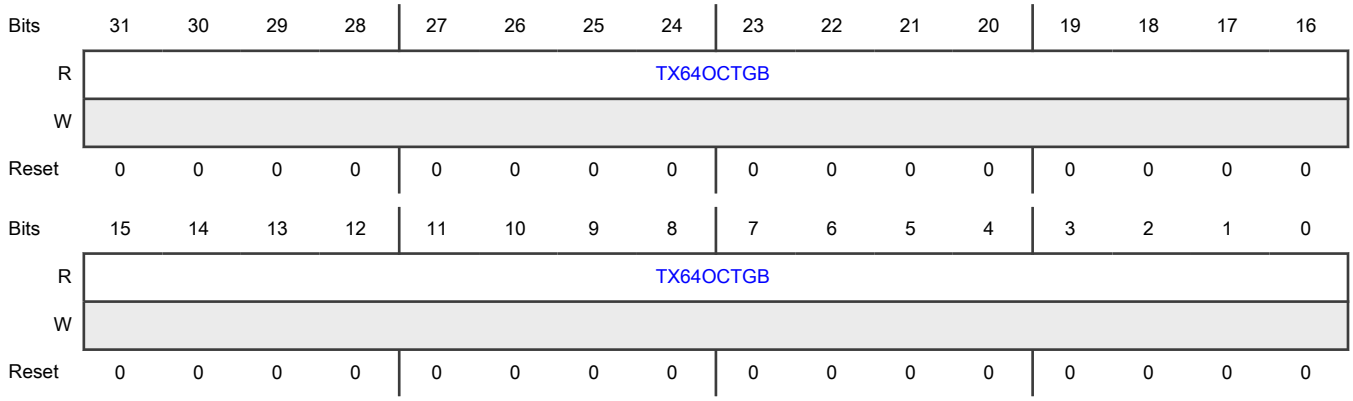
### Offset

| Register | Offset |
|---|---|
| Tx_64Octets_Packets_Good_Bad | 724h |

### Function

Provides the number of 64-byte good and bad packets that the module transmitted, exclusive of preamble and retried packets.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>TX64OCTGB | Transmit 64-Octet Packets Good Bad<br>Indicates the number of transmitted 64-byte good and bad packets, exclusive of preamble and retried packets. |

## 72.17.67 Transmit 65 To 127 Octet Packets Good Bad (Tx_65To127Octets_Packets_Good_Bad)

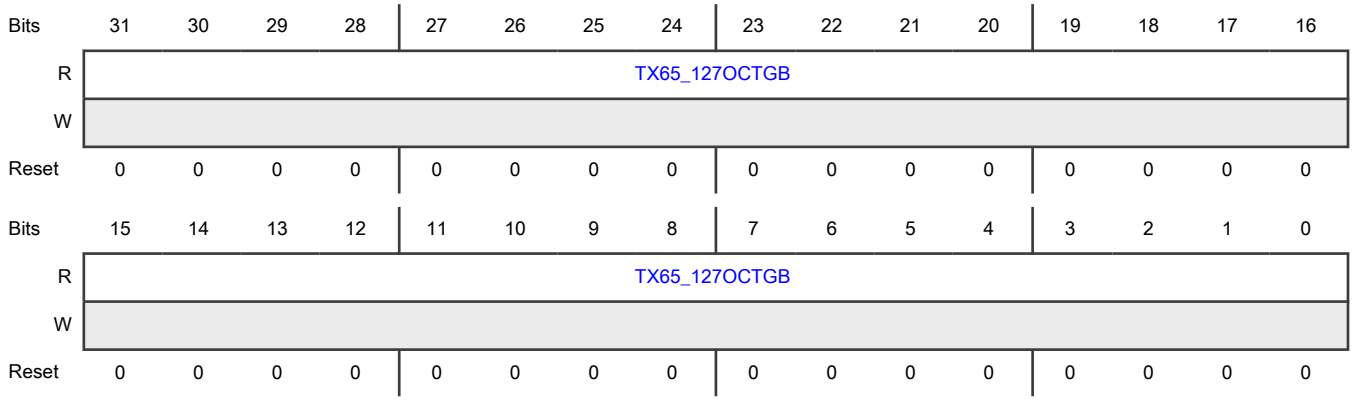
### Offset

| Register | Offset |
|---|---|
| Tx_65To127Octets_Packets_Good_Bad | 728h |

### Function

Provides the number of good and bad packets, having length between 65 and 127 (inclusive) bytes, that the module transmitted, exclusive of preamble and retried packets.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>TX65_127OCTGB | Transmit 65 To 127 Octet Packets Good Bad<br>Indicates the number of good and bad packets transmitted, having length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried packets. |

## 72.17.68 Transmit 128 To 255 Octet Packets Good Bad (Tx_128To255Octets_Packets_Good_Bad)

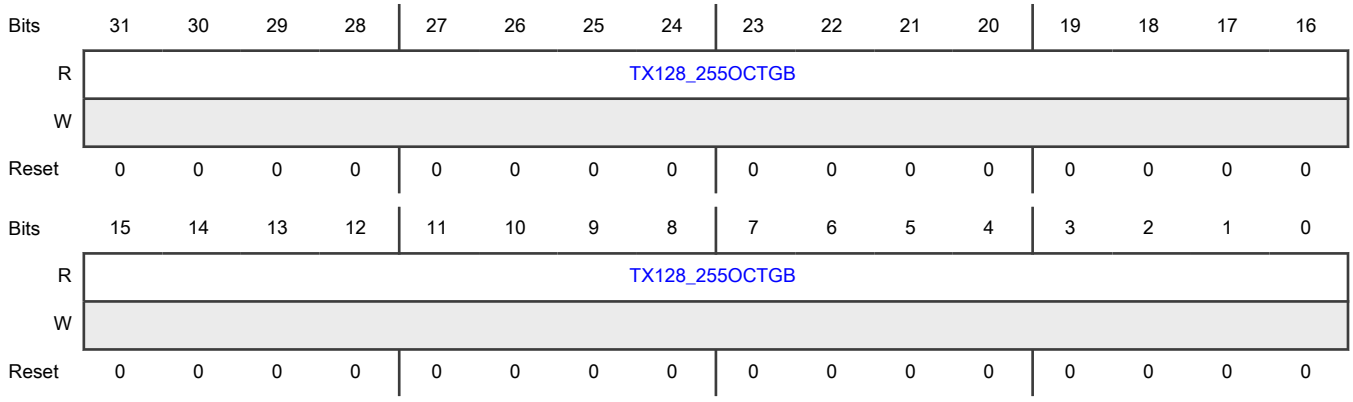
### Offset

| Register | Offset |
|---|---|
| Tx_128To255Octets_Packets_Good_Bad | 72Ch |

### Function

Provides the number of good and bad packets that the module transmitted, having length between 128 to 255 (inclusive) bytes, exclusive of preamble and retried packets.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-0<br>TX128_255OCTGB | Transmit 128 To 255 Octet Packets Good Bad<br>Indicates the number of good and bad packets transmitted, having length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried packets. |

## 72.17.69 Transmit 256 To 511 Octet Packets Good Bad (Tx_256To511Octets_Packets_Good_Bad)

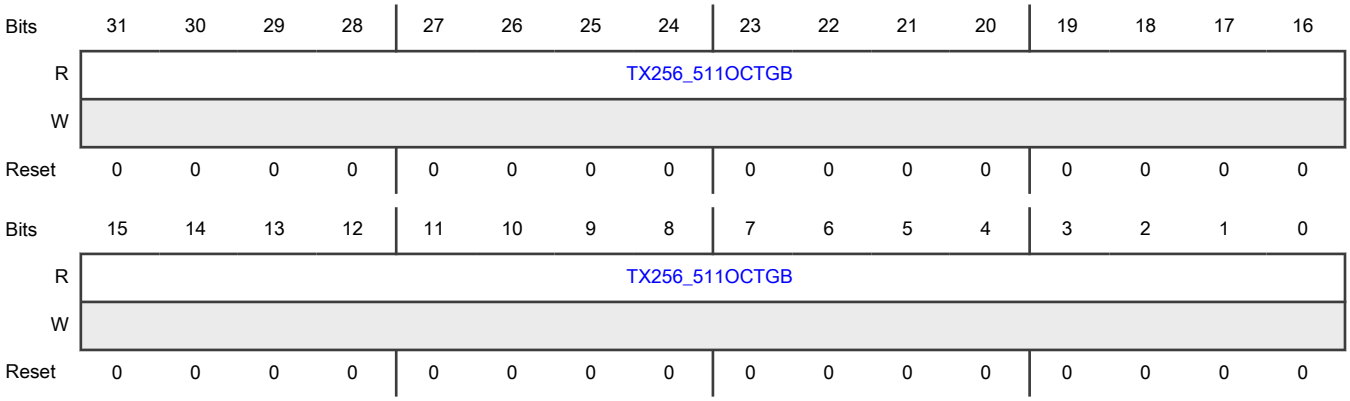
### Offset

| Register | Offset |
|---|---|
| Tx_256To511Octets_Packets_Good_Bad | 730h |

### Function

Provides the number of good and bad packets that the module transmitted, having length between 256 to 511 (inclusive) bytes, exclusive of preamble and retried packets.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>TX256_511OCTGB | Transmit 256 To 511 Octet Packets Good Bad<br>Indicates the number of good and bad packets transmitted, having length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried packets. |

## 72.17.70 Transmit 512 To 1023 Octet Packets Good Bad (Tx_512To1023Octets_Packets_Good_Bad)

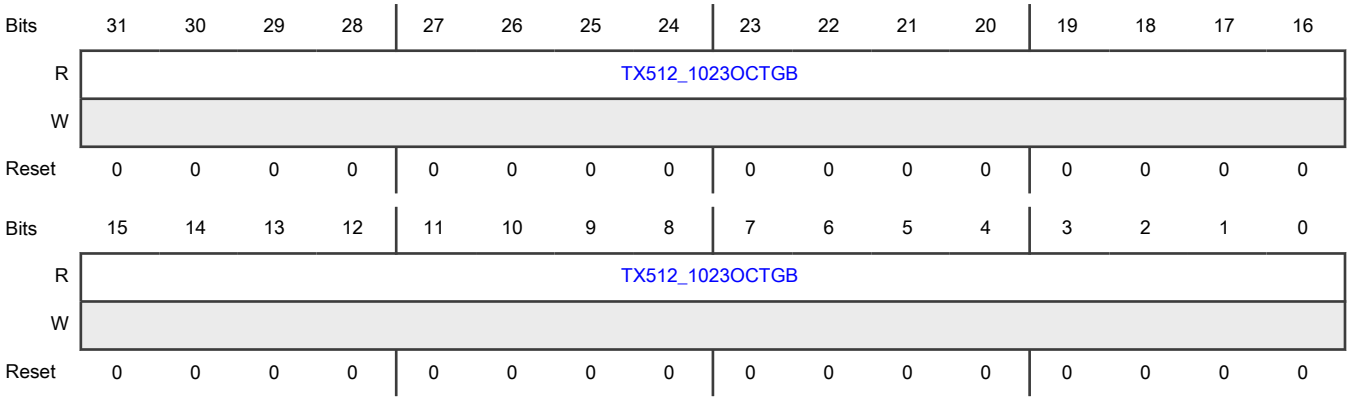
### Offset

| Register | Offset |
|---|---|
| Tx_512To1023Octets_Packets_Good_Bad | 734h |

### Function

Provides the number of good and bad packets that the module transmitted, having length between 512 to 1023 (inclusive) bytes, exclusive of preamble and retried packets.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>TX512_1023OCTGB | Transmit 512 To 1023 Octet Packets Good Bad<br>Indicates the number of good and bad packets transmitted, having length between 512 and 1023 (inclusive) bytes, exclusive of preamble and retried packets. |

## 72.17.71 Transmit 1024 To Max Octet Packets Good Bad (Tx_1024ToMaxOctets_Packets_Good_Bad)

### Offset

| Register | Offset |
|---|---|
| Tx_1024ToMaxOctets_Packets_Good_Bad | 738h |

### Function

Provides the number of good and bad packets that the module transmitted, having a length between 1024 bytes (inclusive) and the maximum size, exclusive of preamble and retried packets.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TX1024_MAXOCTGB | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TX1024_MAXOCTGB | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>TX1024_MAXOCTGB | Transmit 1024 To Max Octet Packets Good Bad<br>Indicates the number of good and bad packets transmitted, having a length between 1024 bytes (inclusive) and the maximum size, exclusive of preamble and retried packets. |

## 72.17.72 Transmit Unicast Packets Good Bad (Tx_Unicast_Packets_Good_Bad)

### Offset

| Register | Offset |
|---|---|
| Tx_Unicast_Packets_Good_Bad | 73Ch |

### Function

Provides the number of good and bad unicast packets that the module transmitted.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>TXUCASTGB | Transmit Unicast Packets Good Bad<br>Indicates the number of good and bad unicast packets transmitted. |

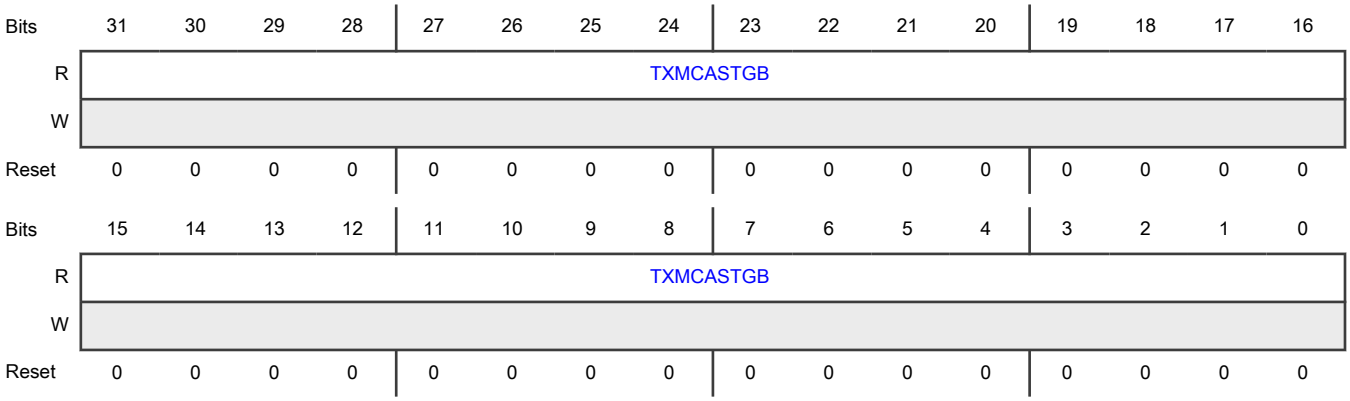
## 72.17.73 Transmit Multicast Packets Good Bad (Tx_Multicast_Packets_Good_Bad)

### Offset

| Register | Offset |
|---|---|
| Tx_Multicast_Packets_Good_Bad | 740h |

### Function
Provides the number of good and bad multicast packets that the module transmitted.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>TXMCASTGB | Transmit Multicast Packets Good Bad<br>Indicates the number of good and bad multicast packets transmitted. |

## 72.17.74 Transmit Broadcast Packets Good Bad (Tx_Broadcast_Packets_Good_Bad)

### Offset

| Register | Offset |
|---|---|
| Tx_Broadcast_Packets_Good_Bad | 744h |

### Function

Provides the number of good and bad broadcast packets that the module transmitted.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>TXBCASTGB | Transmit Broadcast Packets Good Bad<br>Indicates the number of good and bad broadcast packets transmitted. |

## 72.17.75 Transmit Underflow Error Packets (Tx_Underflow_Error_Packets)

### Offset

| Register | Offset |
|---|---|
| Tx_Underflow_Error_Packets | 748h |

### Function

Provides the number of packets that the module aborted because of a packet underflow error.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>TXUNDRFLW | Transmit Underflow Error Packets<br>Indicates the number of packets aborted because of a packet underflow error. |

## 72.17.76 Transmit Single Collision Good Packets (Tx_Single_Collision_Good_Packets)

### Offset

| Register | Offset |
|---|---|
| Tx_Single_Collision_Good_Packets | 74Ch |

### Function
Provides the number of packets that the module successfully transmitted after a single collision in Half-Duplex mode.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>TXSNGLCOLG | Transmit Single Collision Good Packets<br>Indicates the number of successfully transmitted packets after a single collision in Half-Duplex mode. |

## 72.17.77 Transmit Multiple Collision Good Packets (Tx_Multiple_Collision_Good_Packets)

### Offset

| Register | Offset |
|---|---|
| Tx_Multiple_Collision_Good_Packets | 750h |

### Function

Provides the number of packets that the module successfully transmitted after multiple collisions in Half-Duplex mode.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>TXMULTCOLG | Transmit Multiple Collision Good Packets<br>Indicates the number of successfully transmitted packets after multiple collisions in Half-Duplex mode. |

## 72.17.78 Transmit Deferred Packets (Tx_Deferred_Packets)

### Offset

| Register | Offset |
|---|---|
| Tx_Deferred_Packets | 754h |

### Function

Provides the number of packets that the module successfully transmitted after a deferral in Half-Duplex mode.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>TXDEFRD | Transmit Deferred Packets<br>Indicates the number of successfully transmitted packets after a deferral in Half-Duplex mode. |

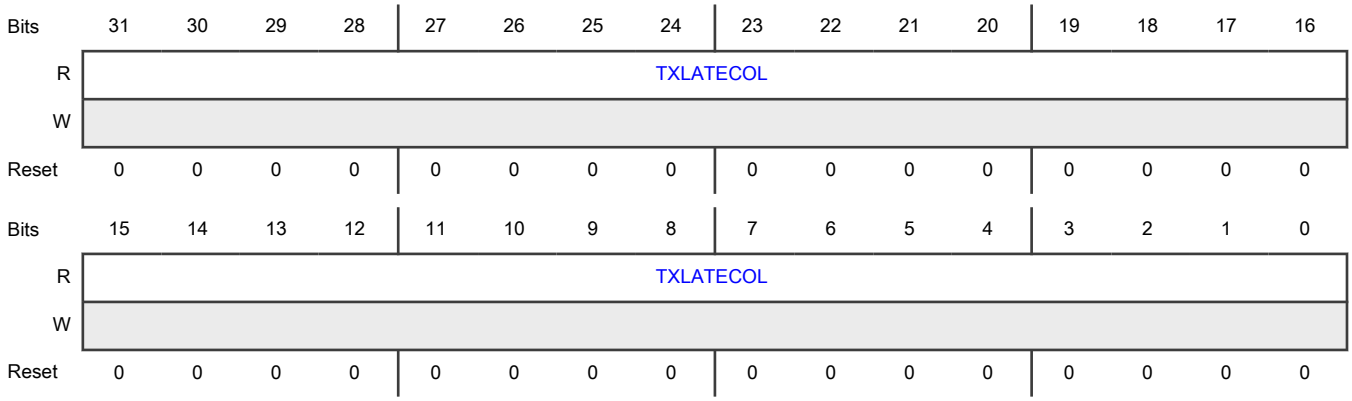
## 72.17.79 Transmit Late Collision Packets (Tx_Late_Collision_Packets)

### Offset

| Register | Offset |
|---|---|
| Tx_Late_Collision_Packets | 758h |

### Function
Provides the number of packets that the module aborted because of a late collision error.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>TXLATECOL | Transmit Late Collision Packets<br>Indicates the number of packets aborted because of a late collision error. |

## 72.17.80 Transmit Excessive Collision Packets (Tx_Excessive_Collision_Packets)

### Offset

| Register | Offset |
|---|---|
| Tx_Excessive_Collision_Packets | 75Ch |

### Function

Provides the number of packets that the module aborted because of excessive (16) collision errors.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>TXEXSCOL | Transmit Excessive Collision Packets<br>Indicates the number of packets aborted because of excessive (16) collision errors. |

## 72.17.81 Transmit Carrier Error Packets (Tx_Carrier_Error_Packets)

### Offset

| Register | Offset |
|---|---|
| Tx_Carrier_Error_Packets | 760h |

### Function

Provides the number of packets that the module aborted because of a carrier sense error (such as no carrier or loss of carrier).

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>TXCARR | Transmit Carrier Error Packets<br>Indicates the number of packets aborted because of a carrier sense error (such as no carrier or loss of carrier). |

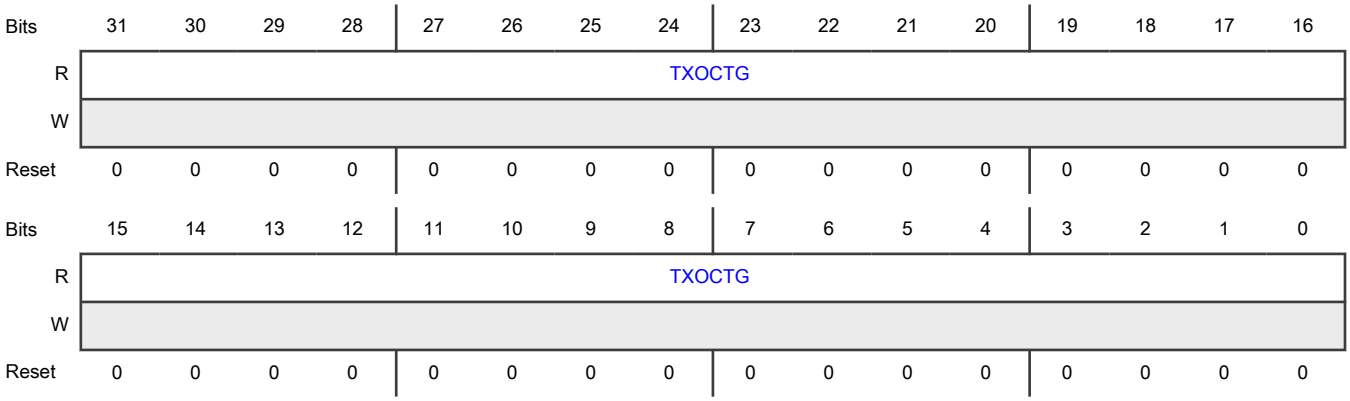
## 72.17.82 Transmit Octet Count Good (Tx_Octet_Count_Good)

### Offset

| Register | Offset |
|---|---|
| Tx_Octet_Count_Good | 764h |

### Function
Provides the number of bytes that the module transmitted, exclusive of preamble, only in good packets.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>TXOCTG | Transmit Octet Count Good<br>Indicates the number of bytes transmitted, exclusive of preamble, only in good packets. |

## 72.17.83 Transmit Packet Count Good (Tx_Packet_Count_Good)

### Offset

| Register | Offset |
|---|---|
| Tx_Packet_Count_Good | 768h |

### Function

Provides the number of good packets that the module transmitted.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>TXPKTG | Transmit Packet Count Good<br>Indicates the number of good packets transmitted. |

## 72.17.84 Transmit Excessive Deferral Error (Tx_Excessive_Deferral_Error)

### Offset

| Register | Offset |
|---|---|
| Tx_Excessive_Deferral_Error | 76Ch |

### Function

Provides the number of packets that the module aborted because of an excessive deferral error (deferred for more than two max-sized packet times).

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>TXEXSDEF | Transmit Excessive Deferral Error<br>Indicates the number of packets aborted because of excessive deferral error (deferred for more than two max-sized packet times). |

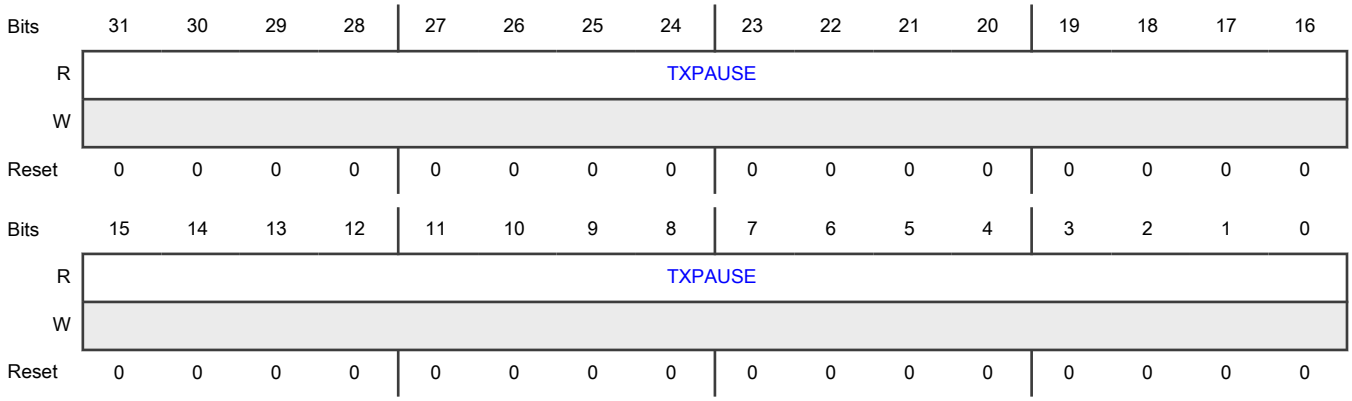
## 72.17.85 Transmit Pause Packets (Tx_Pause_Packets)

### Offset

| Register | Offset |
|---|---|
| Tx_Pause_Packets | 770h |

### Function
Provides the number of good pause packets that the module transmitted.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>TXPAUSE | Transmit Pause Packets<br>Indicates the number of good pause packets transmitted. |

## 72.17.86 Transmit VLAN Packets Good (Tx_VLAN_Packets_Good)

### Offset

| Register | Offset |
|---|---|
| Tx_VLAN_Packets_Good | 774h |

### Function

Provides the number of good VLAN packets that the module transmitted.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>TXVLANG | Transmit VLAN Packets Good<br>Provides the number of good VLAN packets transmitted. |

## 72.17.87 Transmit O Size Packets Good (Tx_OSize_Packets_Good)

### Offset

| Register | Offset |
|---|---|
| Tx_OSize_Packets_Good | 778h |

### Function

Provides the number of packets that the module transmitted without errors, and having a length greater than the maximum size, which is 1,518 or 1,522 bytes for VLAN-tagged packets. This size is 2000 bytes if MAC_Configuration[S2KP] = 1.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>TXOSIZG | Transmit O Size Packets Good<br>Indicates the number of packets transmitted without errors, and having a length greater than the maximum size, which is 1,518 or 1,522 bytes for VLAN-tagged packets. This size is 2000 bytes if MAC_Configuration[S2KP] = 1. |

## 72.17.88 Receive Packets Count Good Bad (Rx_Packets_Count_Good_Bad)

### Offset

| Register | Offset |
|---|---|
| Rx_Packets_Count_Good_Bad | 780h |

### Function
Provides the number of good and bad packets that the module received.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>RXPKTGB | Receive Packets Count Good Bad<br>Indicates the number of good and bad packets received. |

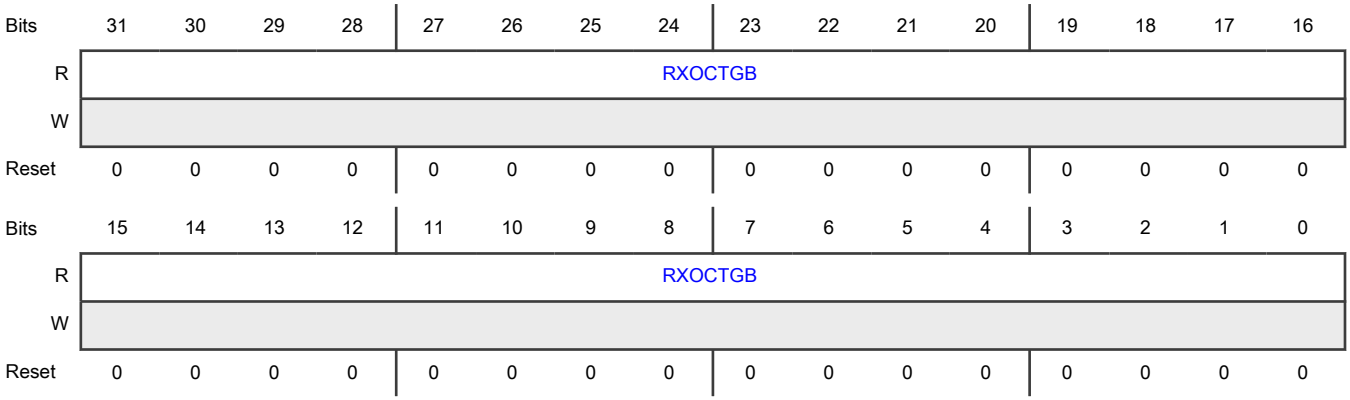
## 72.17.89 Receive Octet Count Good Bad (Rx_Octet_Count_Good_Bad)

### Offset

| Register | Offset |
|---|---|
| Rx_Octet_Count_Good_Bad | 784h |

### Function
Provides the number of bytes that DWC_ther_qos received, exclusive of preamble, in good and bad packets.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>RXOCTGB | Receive Octet Count Good Bad<br>Indicates the number of bytes received, exclusive of preamble, in good and bad packets. |

## 72.17.90 Receive Octet Count Good (Rx_Octet_Count_Good)

### Offset

| Register | Offset |
|---|---|
| Rx_Octet_Count_Good | 788h |

### Function
Provides the number of bytes that the module received, exclusive of preamble, only in good packets.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>RXOCTG | Receive Octet Count Good<br>Indicates the number of bytes received, exclusive of preamble, only in good packets. |

## 72.17.91 Receive Broadcast Packets Good (Rx_Broadcast_Packets_Good)

### Offset

| Register | Offset |
|---|---|
| Rx_Broadcast_Packets_Good | 78Ch |

### Function

Provides the number of good broadcast packets that the module received.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>RXBCASTG | Receive Broadcast Packets Good<br>Indicates the number of good broadcast packets received. |

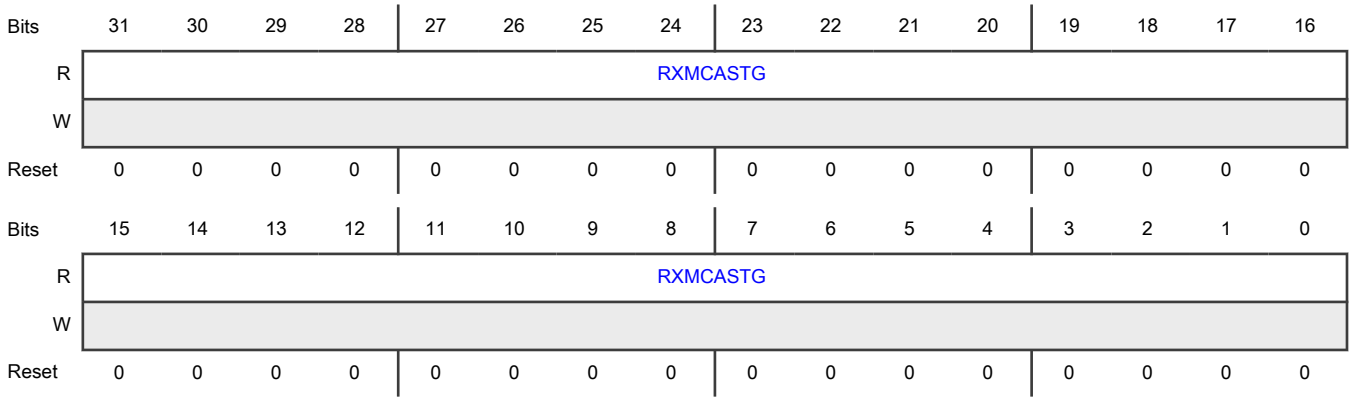
## 72.17.92 Receive Multicast Packets Good (Rx_Multicast_Packets_Good)

### Offset

| Register | Offset |
|---|---|
| Rx_Multicast_Packets_Good | 790h |

### Function
Provides the number of good multicast packets that the module received.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>RXMCASTG | Receive Multicast Packets Good<br>Indicates the number of good multicast packets received. |

## 72.17.93 Receive CRC Error Packets (Rx_CRC_Error_Packets)

### Offset

| Register | Offset |
|---|---|
| Rx_CRC_Error_Packets | 794h |

### Function

Provides the number of packets that the module received with a CRC error.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>RXCRCERR | Receive CRC Error Packets<br>Indicates the number of packets received with a CRC error. |

## 72.17.94 Receive Alignment Error Packets (Rx_Alignment_Error_Packets)

### Offset

| Register | Offset |
|---|---|
| Rx_Alignment_Error_Packets | 798h |

### Function

Provides the number of packets that the module received with an alignment (dribble) error. It is valid only in 10/100 mode.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>RXALGNERR | Receive Alignment Error Packets<br>Indicates the number of packets received with alignment (dribble) error. It is valid only in 10/100 mode. |

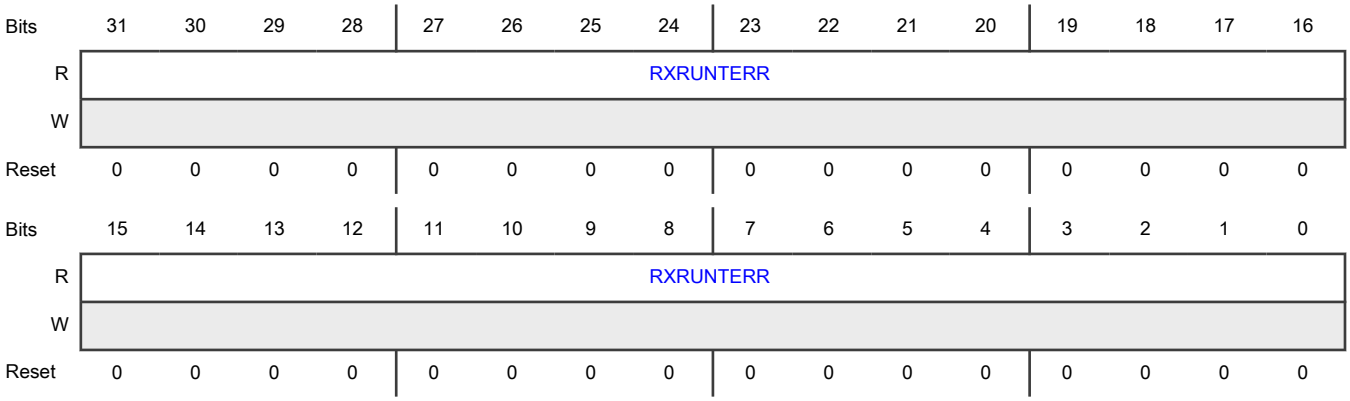
## 72.17.95 Receive Runt Error Packets (Rx_Runt_Error_Packets)

### Offset

| Register | Offset |
|---|---|
| Rx_Runt_Error_Packets | 79Ch |

### Function
Provides the number of runt packets, which have a length less than 64 bytes and a CRC error, that the module received.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>RXRUNTERR | Receive Runt Error Packets<br>Indicates the number of runt packets received. |

## 72.17.96 Receive Jabber Error Packets (Rx_Jabber_Error_Packets)

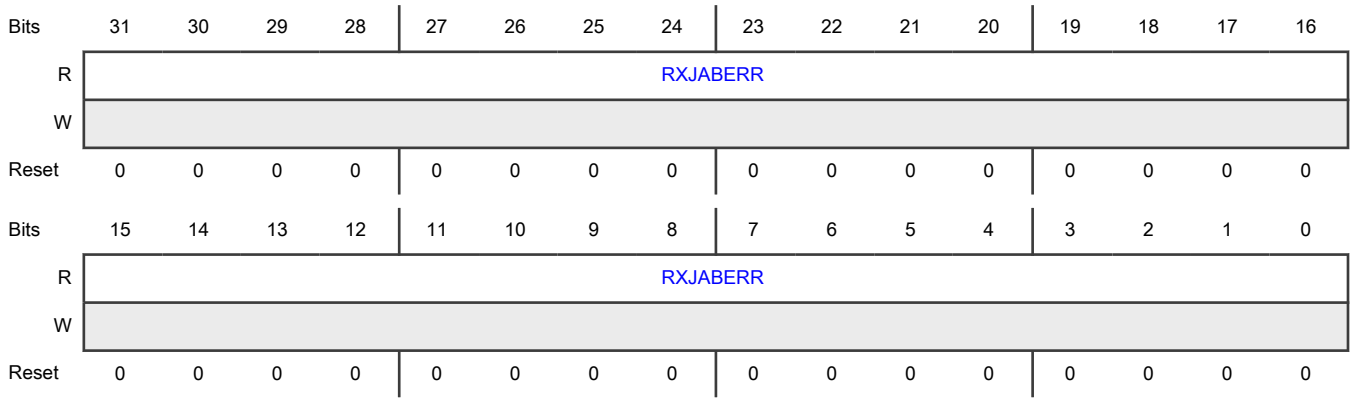
### Offset

| Register | Offset |
|---|---|
| Rx_Jabber_Error_Packets | 7A0h |

### Function

Provides the number of giant packets that the module received, having a length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN-tagged packets), and with a CRC error. If Jumbo Packet mode is enabled, packets of length greater than 9,018 bytes (9,022 bytes for VLAN-tagged packets) are considered as giant packets.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>RXJABERR | Receive Jabber Error Packets<br>Indicates the number of giant packets received, having a length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN-tagged packets), and with a CRC error. If Jumbo Packet mode is enabled, packets of length greater than 9,018 bytes (9,022 bytes for VLAN-tagged packets) are considered as giant packets. |

## 72.17.97 Receive Undersize Packets Good (Rx_Undersize_Packets_Good)

### Offset

| Register | Offset |
|---|---|
| Rx_Undersize_Packets_Good | 7A4h |

### Function

Provides the number of packets that the module received, having a length less than 64 bytes, but without any errors.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>RXUNDERSZG | Receive Undersize Packets Good<br>Indicates the number of packets received, having a length less than 64 bytes, but without any errors. |

## 72.17.98 Receive Oversize Packets Good (Rx_Oversize_Packets_Good)

### Offset

| Register | Offset |
|---|---|
| Rx_Oversize_Packets_Good | 7A8h |

### Function

Provides the number of packets that the module received without any errors, having a length greater than the maximum size (1,518 bytes or 1,522 bytes for VLAN-tagged packets; 2000 bytes if MAC_Configuration[S2KP] = 1).

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>RXOVERSZG | Receive Oversize Packets Good<br>Indicates the number of packets received without errors, with a length greater than the maximum size (1,518 bytes or 1,522 bytes for VLAN-tagged packets; 2000 bytes if MAC_Configuration[S2KP] = 1). |

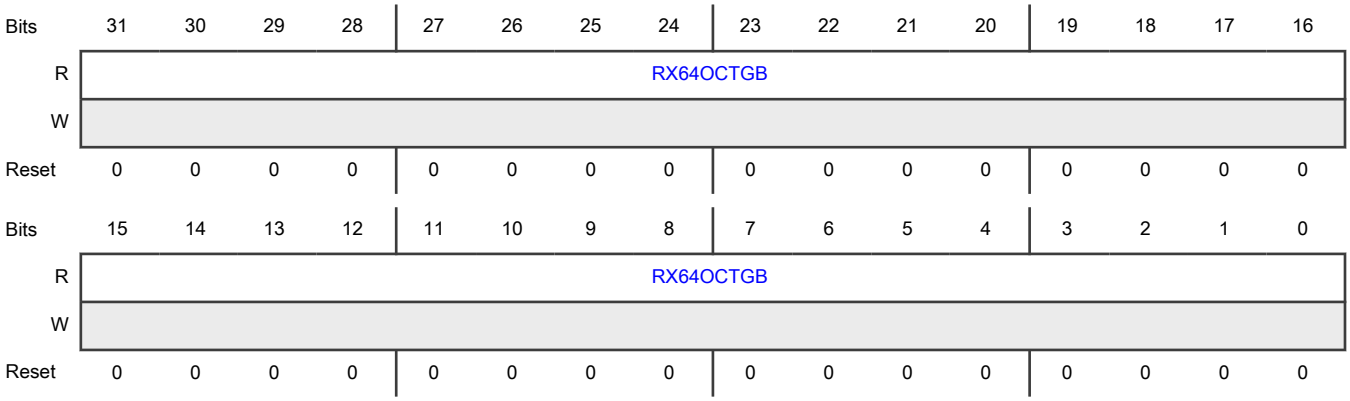
## 72.17.99 Receive 64 Octets Packets Good Bad (Rx_64Octets_Packets_Good_Bad)

### Offset

| Register | Offset |
|---|---|
| Rx_64Octets_Packets_Good_Bad | 7ACh |

### Function
Provides the number of 64-byte good and bad packets that the module received, exclusive of the preamble.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>RX64OCTGB | Receive 64 Octets Packets Good Bad<br>Provides the number of 64-byte good and bad packets that the module received, exclusive of the preamble. |

## 72.17.100 Receive 65-127 Octets Packets Good Bad (Rx_65To127Octets_Packets_Good_Bad)

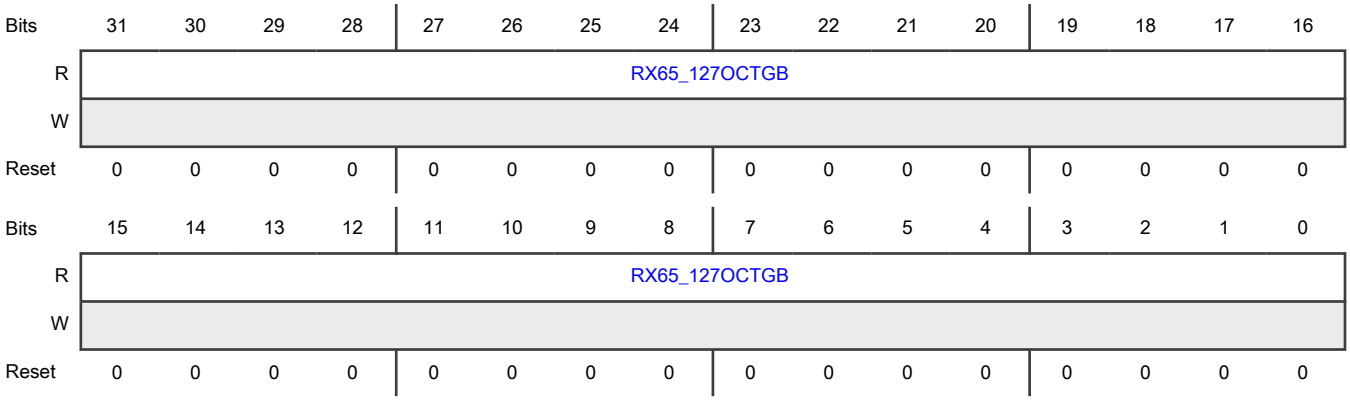
### Offset

| Register | Offset |
|---|---|
| Rx_65To127Octets_Packets_Good_Bad | 7B0h |

### Function

Provides the number of good and bad packets that the module received, ranging between 65 and 127 (inclusive) bytes, exclusive of the preamble.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>RX65_127OCTGB | Receive 65-127 Octets Packets Good Bad<br>Indicates the number of good and bad packets received, ranging between 65 and 127 (inclusive) bytes, exclusive of the preamble. |

## 72.17.101 Receive 128-255 Octets Packets Good Bad (Rx_128To255Octets_Packets_Good_Bad)

### Offset

| Register | Offset |
|---|---|
| Rx_128To255Octets_Packets_Good_Bad | 7B4h |

### Function

Provides the number of good and bad packets that the module received, with packet length between 128 and 255 (inclusive) bytes, exclusive of the preamble.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RX128_255OCTGB | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RX128_255OCTGB | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>RX128_255OCTGB | Receive 128-255 Octets Packets Good Bad<br>Indicates the number of good and bad packets received, with packet length between 128 and 255 (inclusive) bytes, exclusive of the preamble. |

## 72.17.102 Receive 256-511 Octets Packets Good Bad (Rx_256To511Octets_Packets_Good_Bad)

### Offset

| Register | Offset |
|---|---|
| Rx_256To511Octets_Packets_Good_Bad | 7B8h |

### Function

Provides the number of good and bad packets that the module received, having a packet length between 256 and 511 (inclusive) bytes, exclusive of the preamble.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RX256_511OCTGB | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RX256_511OCTGB | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>RX256_511OCTGB | Receive 256-511 Octets Packets Good Bad<br>Indicates the number of good and bad packets received, having a packet length between 256 and 511 (inclusive) bytes, exclusive of the preamble. |

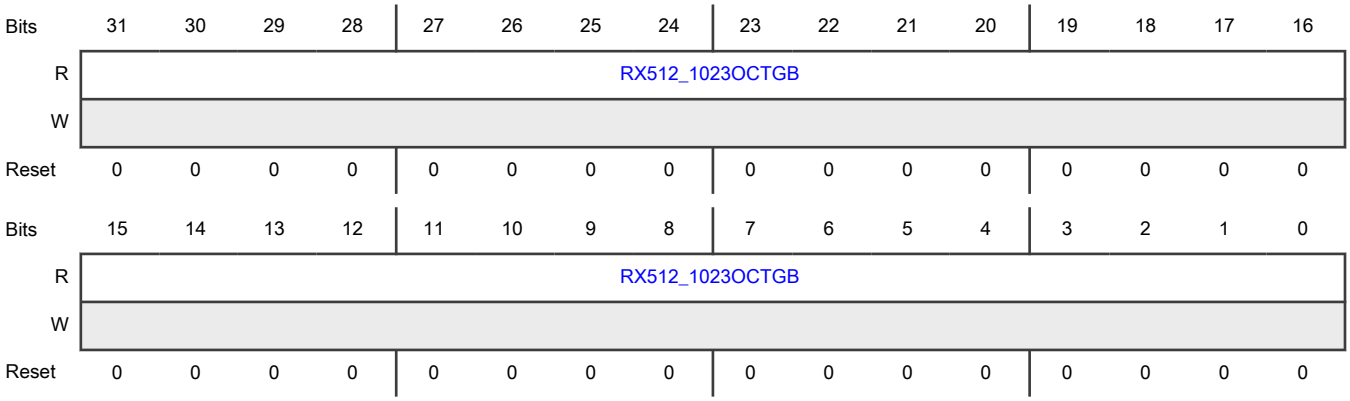
## 72.17.103 Receive 512-1023 Octets Packets Good Bad (Rx_512To1023Octets_Packets_Good_Bad)

### Offset

| Register | Offset |
|---|---|
| Rx_512To1023Octets_Packets_Good_Bad | 7BCh |

### Function
Provides the number of good and bad packets that the module received, having a packet length between 512 and 1023 (inclusive) bytes, exclusive of the preamble.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>RX512_1023OCTGB | Receive 512-1023 Octets Packets Good Bad<br>Indicates the number of good and bad packets received, having a packet length between 512 and 1023 (inclusive) bytes, exclusive of the preamble. |

## 72.17.104 Receive 1024 To Max Octets Good Bad (Rx_1024ToMaxOctets_Packets_Good_Bad)

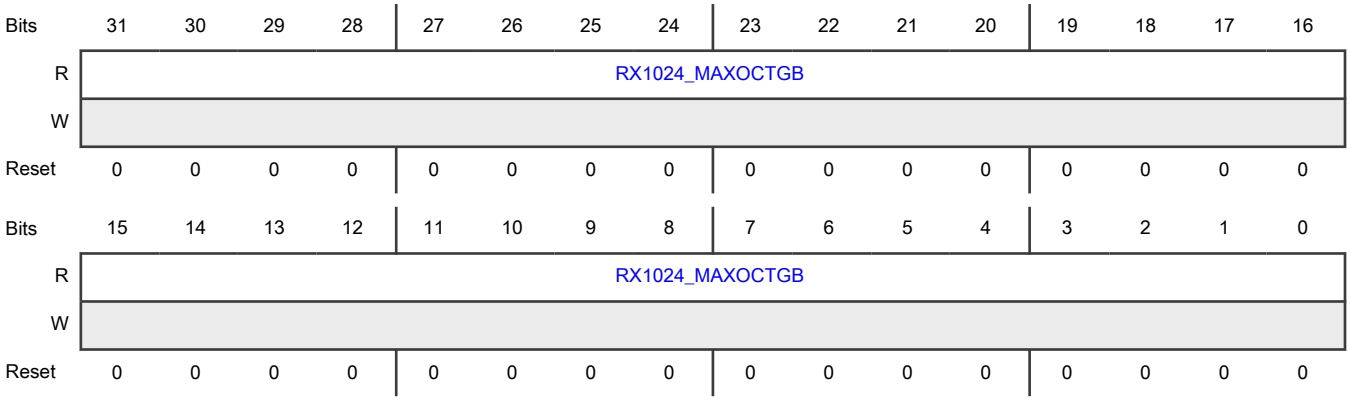
### Offset

| Register | Offset |
|---|---|
| Rx_1024ToMaxOctets_Packets_Good_Bad | 7C0h |

### Function

Provides the number of good and bad packets that the module received, having an inclusive packet length between 1024 bytes and the maximum byte size, exclusive of the preamble.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>RX1024_MAXOCTGB | Receive 1024-Max Octets Good Bad<br>Indicates the number of good and bad packets received with length between 1024 and the maximum size (inclusive) bytes, exclusive of the preamble. |

## 72.17.105 Receive Unicast Packets Good (Rx_Unicast_Packets_Good)

### Offset

| Register | Offset |
|---|---|
| Rx_Unicast_Packets_Good | 7C4h |

### Function

Provides the number of good unicast packets that the module received.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>RXUCASTG | Receive Unicast Packets Good<br>Indicates the number of good unicast packets received. |

## 72.17.106 Receive Length Error Packets (Rx_Length_Error_Packets)

### Offset

| Register | Offset |
|---|---|
| Rx_Length_Error_Packets | 7C8h |

### Function

Provides the number of packets that the module received with a length error (the Length/Type (LT) field of the RDES3 normal descriptor not equal to the packet size) for all the packets with a valid length field. For more information on the LT field, see the "RDES3 normal descriptor (write-back format)" section.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>RXLENERR | Receive Length Error Packets<br>Indicates the number of packets received with a length error (the Length/Type (LT) field of the RDES3 normal descriptor not equal to the packet size) for all the packets with a valid length field. For more information on the LT field, see the "RDES3 normal descriptor (write-back format)" section. |

## 72.17.107 Receive Out of Range Type Packet (Rx_Out_Of_Range_Type_Packets)

### Offset

| Register | Offset |
|---|---|
| Rx_Out_Of_Range_Type_Packets | 7CCh |

### Function

Provides the number of packets that the module received, with the length field not equal to the valid packet size (greater than 1,500 but less than 1,536).

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>RXOUTOFRNG | Receive Out of Range Type Packet<br>Indicates the number of packets received with length field not equal to the valid packet size (greater than 1,500 but less than 1,536). |

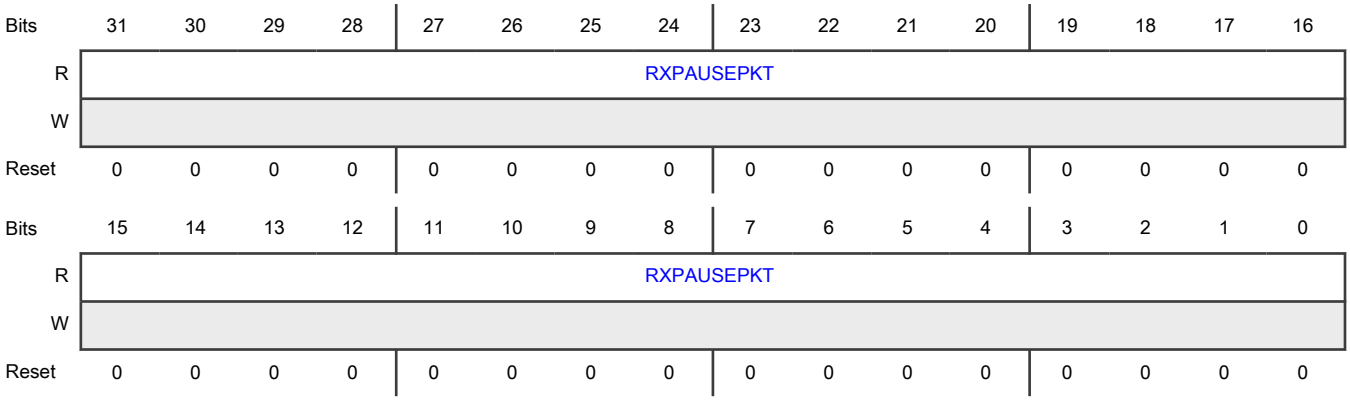
## 72.17.108 Receive Pause Packets (Rx_Pause_Packets)

### Offset

| Register | Offset |
|---|---|
| Rx_Pause_Packets | 7D0h |

### Function
Provides the number of good and valid pause packets that the module received.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>RXPAUSEPKT | Receive Pause Packets<br>Indicates the number of good and valid pause packets received. |

## 72.17.109 Receive FIFO Overflow Packets (Rx_FIFO_Overflow_Packets)

### Offset

| Register | Offset |
|---|---|
| Rx_FIFO_Overflow_Packets | 7D4h |

### Function
Provides the number of missed received packets because of FIFO overflow in the module.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>RXFIFOOVFL | Receive FIFO Overflow Packets<br>Indicates the number of missed received packets because of FIFO overflow. |

## 72.17.110 Receive VLAN Packets Good Bad (Rx_VLAN_Packets_Good_Bad)

### Offset

| Register | Offset |
|---|---|
| Rx_VLAN_Packets_Good_Bad | 7D8h |

### Function

Provides the number of good and bad VLAN packets that the module received.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>RXVLANPKTGB | Receive VLAN Packets Good Bad<br>Indicates the number of good and bad VLAN packets received. |

## 72.17.111 Receive Watchdog Error Packets (Rx_Watchdog_Error_Packets)

### Offset

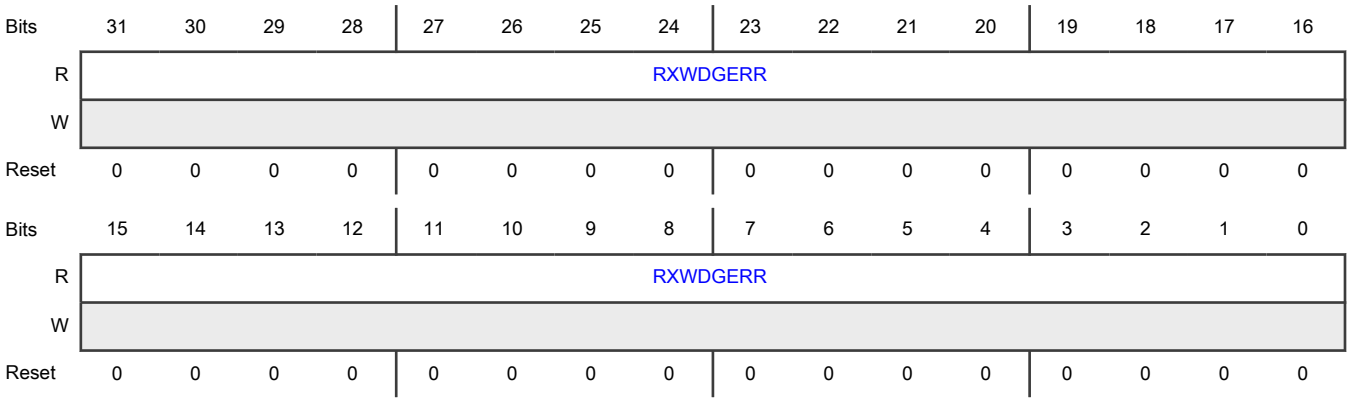
| Register | Offset |
|---|---|
| Rx_Watchdog_Error_Packets | 7DCh |

### Function

Provides the number of packets that the module received with a watchdog timeout error. Packets with a data load larger than the following are received:

- 2,048 bytes when MAC_Configuration[JE] = 0 and MAC_Configuration[WD] = 0
- 10,240 bytes when MAC_Configuration[JE] = 1 and MAC_Configuration[WD] = 0
- 16,384 bytes when MAC_Configuration[WD] = 1 or the value programmed in MAC Watchdog Timeout (MAC_Watchdog_Timeout)

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>RXWDGERR | Receive Watchdog Error Packets<br>Indicates the number of packets received with a watchdog timeout error. Packets with a data load larger than the following are received: |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • 2,048 bytes when MAC_Configuration[JE] = 0 and MAC_Configuration[WD] = 0<br>• 10,240 bytes when MAC_Configuration[JE] = 1 and MAC_Configuration[WD] = 0<br>• 16,384 bytes when MAC_Configuration[WD] = 1 or the value programmed in MAC Watchdog Timeout (MAC_Watchdog_Timeout) |

## 72.17.112 Receive Error Packets (Rx_Receive_Error_Packets)

### Offset

| Register | Offset |
|---|---|
| Rx_Receive_Error_Packets | 7E0h |

### Function
Provides the number of packets that the module received with a receive or packet extension error on GMII or MII.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>RXRCVERR | Receive Error Packets<br>Indicates the number of packets received with a receive or packet extension error on GMII or MII. |

## 72.17.113 Receive Control Packets Good (Rx_Control_Packets_Good)

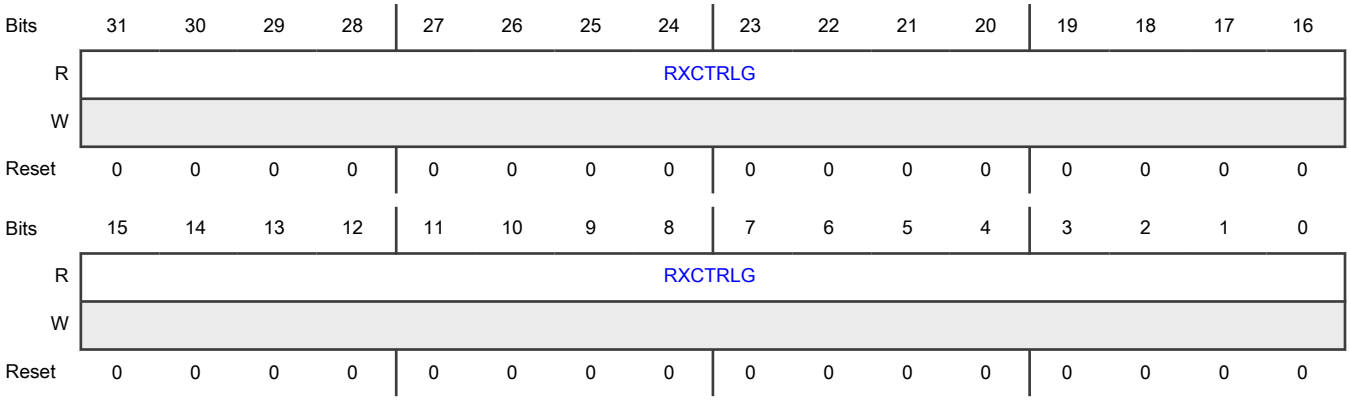
### Offset

| Register | Offset |
|---|---|
| Rx_Control_Packets_Good | 7E4h |

### Function

Provides the number of good control packets that the module received.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>RXCTRLG | Receive Control Packets Good<br>Indicates the number of good control packets received. |

## 72.17.114 MMC Transmit FPE Fragment Counter Interrupt Status (MMC_FPE_Tx_Interrupt)

### Offset

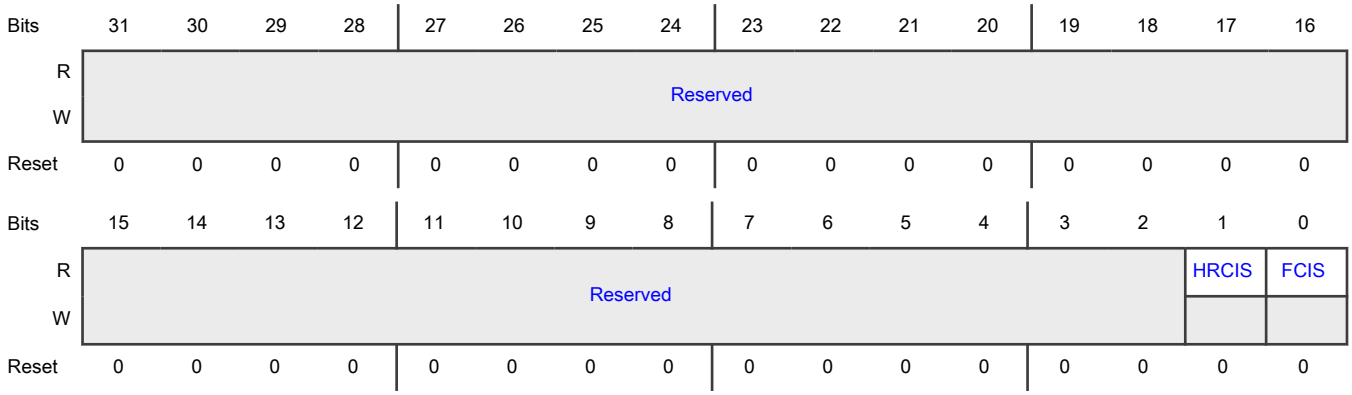
| Register | Offset |
|---|---|
| MMC_FPE_Tx_Interrupt | 8A0h |

### Function

Maintains the interrupts generated from all the FPE-related transmit statistic counters when they reach half of their maximum values (8000_0000h for a 32-bit counter and 8000h for a 16-bit counter), and when they cross their maximum values (FFFF_FFFFh for a 32-bit counter and FFFFh for a 16-bit counter). When MMC_Control[CNTSTOPRO] = 1, the interrupts are set but the counter remains at all-ones.

An interrupt bit of this register becomes 0 when the respective MMC counter that caused the interrupt is read. The least significant byte lane (bits [7:0]) of the respective counter must be read to clear the interrupt bit.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>HRCIS | MMC Transmit Hold Request Counter Interrupt Status<br>Indicates whether the MMC Transmit hold request counter interrupt status is detected.<br>This field:<br>• Becomes 1 when the Tx_Hold_Req_Cntr counter reaches half of the maximum value or the maximum value.<br>• Exists when any one of the receive or transmit MMC counters is enabled during FPE with AV_EST-enabled configuration.<br>Access restrictions apply to this field that clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 0<br>FCIS | MMC Transmit FPE Fragment Counter Interrupt Status<br>Indicates whether the MMC Transmit FPE fragment counter interrupt status is detected.<br>This field:<br>• Becomes 1 when the Tx_FPE_Fragment_Cntr counter reaches half of the maximum value or the maximum value.<br>• Exists when any one of the receive or transmit MMC counters is enabled during FPE-enabled configuration.<br>Access restrictions apply to this field that clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |

## 72.17.115 MMC FPE Transmit Interrupt Mask (MMC_FPE_Tx_Interrupt_Mask)

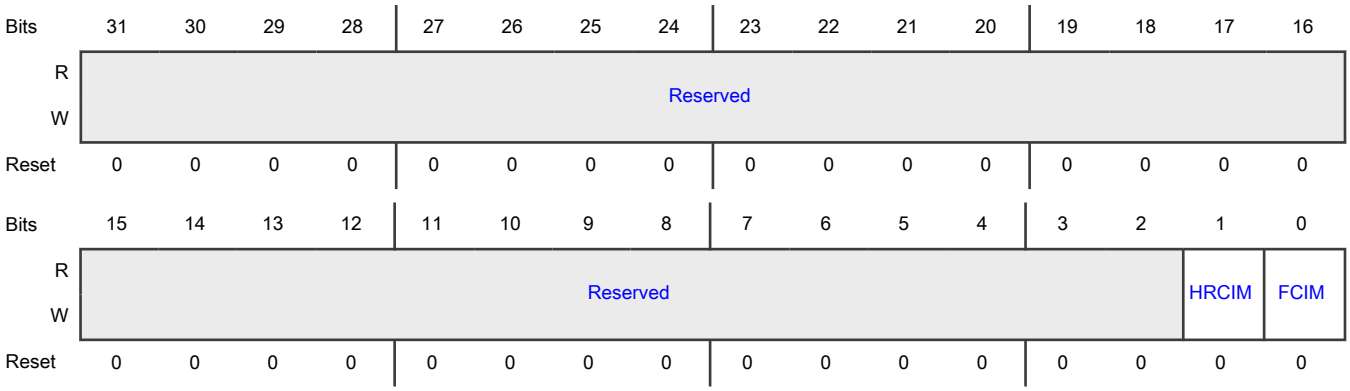
### Offset

| Register | Offset |
|---|---|
| MMC_FPE_Tx_Interrupt_Mask | 8A4h |

### Function

Maintains the masks for interrupts generated from all FPE-related transmit statistics counters. MMC Receive Interrupt Mask (MMC_Rx_Interrupt_Mask) maintains the masks for the interrupts generated when FPE-related receive statistic counters reach half of their maximum value or the maximum value.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>HRCIM | MMC Transmit Hold Request Counter Interrupt Mask<br>Indicates the status of the MMC transmit hold request counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the Tx_Hold_Req_Cntr counter reaches half of its maximum value or the maximum value.<br>The field exists when any one of the receive, transmit, or MMC counters is enabled during FPE with AV_EST-enabled configuration.<br>0b - Disabled<br>1b - Enabled |
| 0<br>FCIM | MMC Transmit Fragment Counter Interrupt Mask<br>Indicates the status of the MMC transmit fragment counter interrupt mask. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Writing 1 to this field masks the interrupt when the Tx_FPE_Fragment_Cntr counter reaches half of its maximum value or the maximum value. The field exists when any one of the receive, transmit, or MMC counters is enabled during FPE-enabled configuration.<br>0b - Disabled<br>1b - Enabled |

## 72.17.116 Transmit FPE Fragment Counter (MMC_Tx_FPE_Fragment_Cntr)

### Offset

| Register | Offset |
|---|---|
| MMC_Tx_FPE_Fragment_Cntr | 8A8h |

### Function

Provides the number of additional mPackets transmitted because of preemption.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>TXFFC | Transmit FPE Fragment Counter<br>Indicates the number of additional mPackets transmitted because of preemption.<br>Exists when any one of the receive, transmit, or MMC counters is enabled during FPE-enabled configuration. |

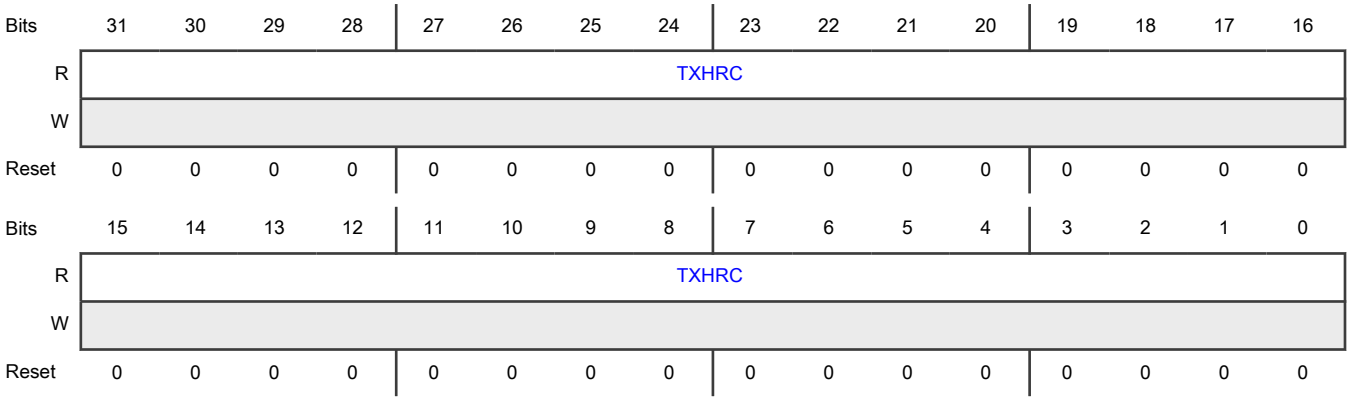
## 72.17.117 Transmit Hold Request Counter (MMC_Tx_Hold_Req_Cntr)

### Offset

| Register | Offset |
|---|---|
| MMC_Tx_Hold_Req_Cntr | 8ACh |

### Function
Provides the count of times that a hold request is given to MAC.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>TXHRC | Transmit Hold Request Counter<br>Indicates the count of times that a hold request is given to MAC.<br>Exists when any one of the receive, transmit, or MMC counters is enabled during FPE with AV_EST-enabled configuration. |

## 72.17.118 MMC Receive Packet Assembly Error Counter Interrupt Status (MMC_FPE_Rx_Interrupt)

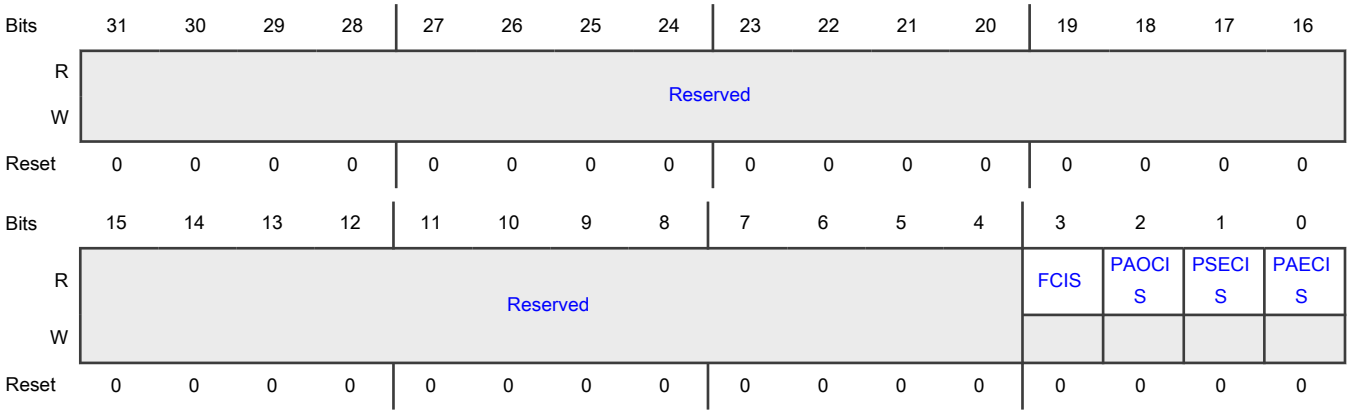
### Offset

| Register | Offset |
|---|---|
| MMC_FPE_Rx_Interrupt | 8C0h |

### Function
Maintains the interrupts generated from all the FPE-related receive statistic counters, when these transmit statistic counters reach half of their maximum values (8000_0000h for a 32-bit counter and 8000h for a 16-bit counter), and when they cross their maximum values (FFFF_FFFFh for a 32-bit counter and FFFFh for 16-bit counter). When MMC_Control[CNTSTOPRO] = 1, the interrupts are set but the counter remains at all-ones. An interrupt bit of this register becomes 0 when the respective

MMC counter that caused the interrupt is read. The least significant byte lane (bits [7:0]) of the respective counter must be read to clear the interrupt bit.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3<br>FCIS | MMC Receive FPE Fragment Counter Interrupt Status<br>Indicates whether the MMC receive FPE fragment counter interrupt status is detected.<br>This field becomes 1 when the Rx_FPE_Fragment_Cntr counter reaches half of its maximum value or the maximum value.<br>Exists when any one of the receive, transmit, or MMC counters is enabled during FPE-enabled configuration.<br>Access restrictions apply to this field that clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 2<br>PAOCIS | MMC Receive Packet Assembly OK Counter Interrupt Status<br>Indicates whether the MMC receive packet assembly OK counter interrupt status is detected.<br>This field becomes 1 when the Rx_Packet_Assemble_Ok_Cntr counter reaches half of its maximum value or the maximum value.<br>Exists when any one of the receive, transmit, or MMC counters is enabled during FPE-enabled configuration.<br>Access restrictions apply to this field that clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 1<br>PSECIS | MMC Receive, Transmit, Packet SMD Error Counter Interrupt Status<br>Indicates whether the status of the MMC receive packet SMD error counter interrupt is detected.<br>This field becomes 1 when the Rx_Packet_SMD_Err_Cntr counter reaches half of its maximum value or the maximum value.<br>Exists when any one of the receive, transmit, or MMC counters is enabled during FPE-enabled configuration.<br>Access restrictions apply to this field that clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 0<br>PAECIS | MMC Receive, transmit, Packet Assembly Error Counter Interrupt Status<br>Indicates whether the status of the MMC receive packet assembly error counter interrupt is detected.<br>This field becomes 1 when the Rx_Packet_Assemble_Err_Cntr counter reaches half of its maximum value or the maximum value.<br>Exists when any one of the receive, transmit, or MMC counters is enabled during the FPE-enabled configuration.<br>Access restrictions apply to this field that clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |

## 72.17.119 MMC FPE Receive Interrupt Mask (MMC_FPE_Rx_Interrupt_Mask)

### Offset

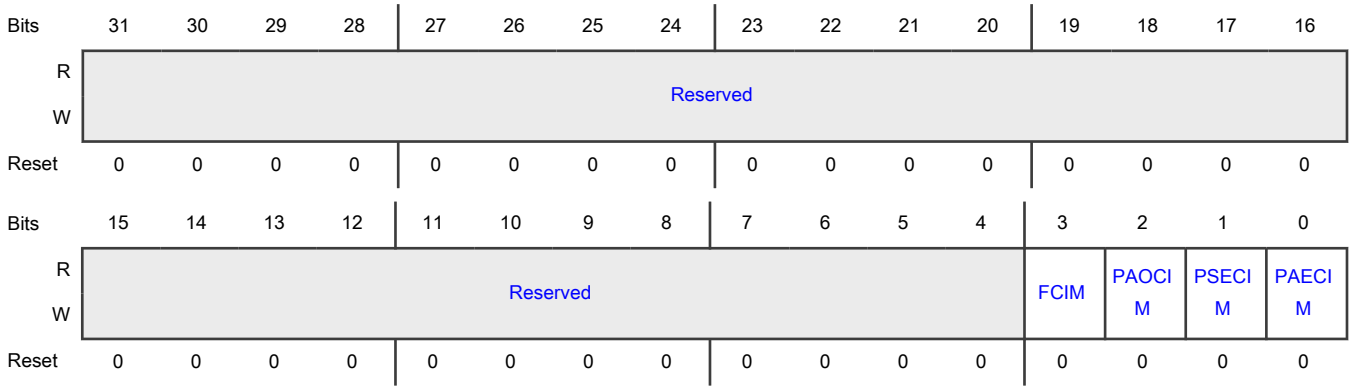
| Register | Offset |
|---|---|
| MMC_FPE_Rx_Interrupt_Mask | 8C4h |

### Function

Maintains the masks for interrupts generated from all FPE-related receive statistic counters.

This register maintains the masks for the interrupts generated when FPE-related receive statistic counters reach half of their maximum value or the maximum value.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved |
| 3<br>FCIM | MMC Receive FPE Fragment Counter Interrupt Mask<br>Indicates the status of the MMC receive FPE fragment counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the Tx_FPE_Fragment_Cntr counter reaches half of its maximum value or the maximum value.<br>Exists when any one of the receive, transmit, or MMC counters is enabled during FPE-enabled configuration.<br>0b - Disabled<br>1b - Enabled |
| 2<br>PAOCIM | MMC Receive Packet Assembly OK Counter Interrupt Mask<br>Indicates the status of the MMC receive packet assembly OK counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the Rx_Packet_Assemble_Ok_Cntr counter reaches half of its maximum value or the maximum value.<br>Exists when any one of the receive, transmit, or MMC counters is enabled during FPE-enabled configuration.<br>0b - Disabled<br>1b - Enabled |
| 1<br>PSECIM | MMC Receive Packet SMD Error Counter Interrupt Mask<br>Indicates the status of the MMC receive packet SMD error counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the Rx_Packet_SMD_Err_Cntr counter reaches half of its maximum value or the maximum value.<br>Exists when any one of the receive, transmit, or MMC counters is enabled during FPE-enabled configuration. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |
| 0<br>PAECIM | MMC receive Packet Assembly Error Counter Interrupt Mask<br>Indicates the status of the MMC receive packet assembly error counter interrupt mask.<br>Writing 1 to this field masks the interrupt when the Rx_Packet_Assemble_Err_Cntr counter reaches half of its maximum value or the maximum value.<br>Exists when any one of the receive, transmit, or MMC counters is enabled during FPE-enabled configuration.<br>0b - Disabled<br>1b - Enabled |

## 72.17.120 MMC Receive Packet Assembly Error Counter (MMC_Rx_Packet_Assembly_Err_Cntr)

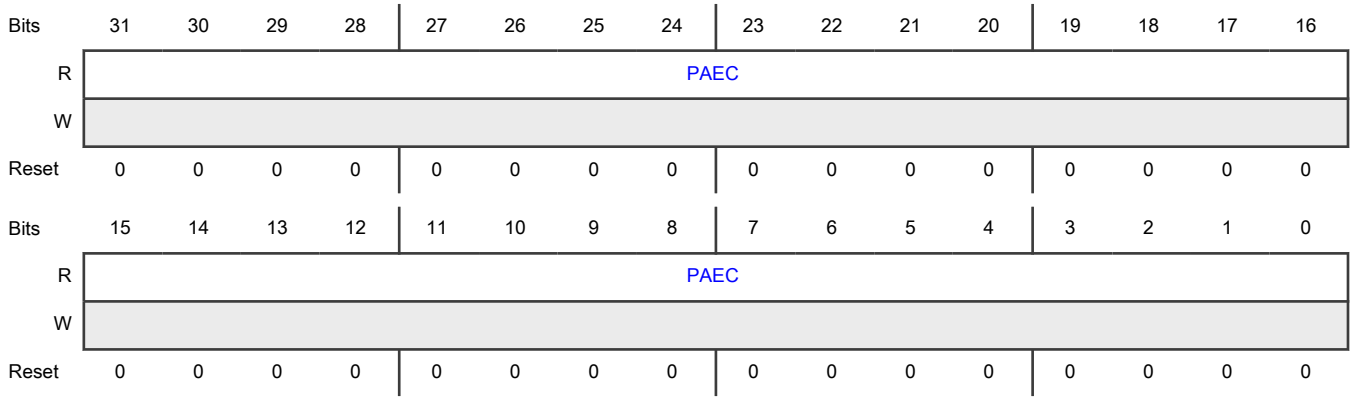
### Offset

| Register | Offset |
|---|---|
| MMC_Rx_Packet_Assembly_Err_Cntr | 8C8h |

### Function

Provides the number of MAC frames having reassembly errors on the receiver arising out of mismatch in the fragment count value.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-0<br>PAEC | Packet Assembly Error Counter<br>Indicates the number of MAC frames having reassembly errors on the receiver arising out of mismatch in the fragment count value.<br>Exists when any one of the receive, transmit, or MMC counters is enabled during FPE-enabled configuration. |

## 72.17.121 MMC Receive Packet SMD Error Counter (MMC_Rx_Packet_SMD_Err_Cntr)

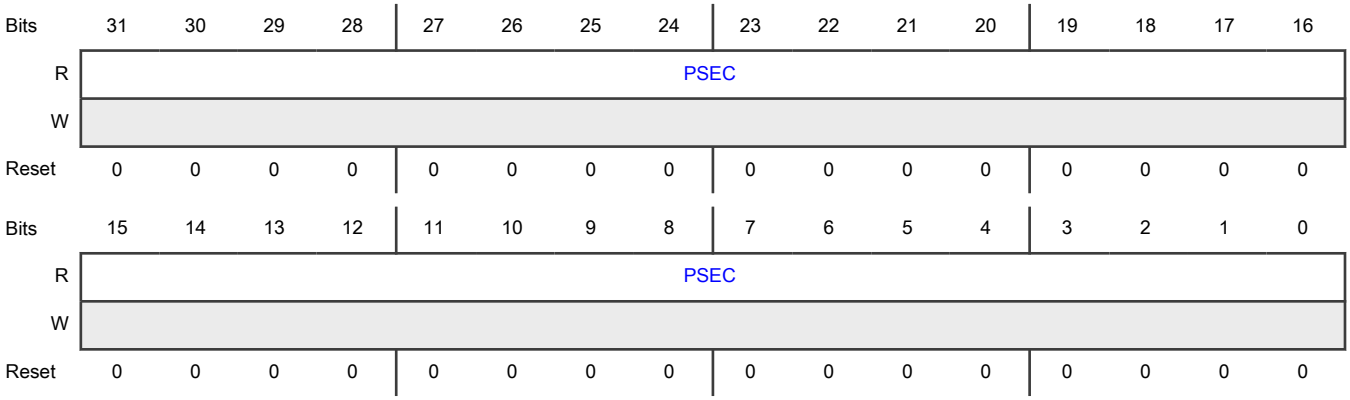
### Offset

| Register | Offset |
|---|---|
| MMC_Rx_Packet_SMD_Err_Cntr | 8CCh |

### Function

Provides the number of received MAC frames rejected because of an unknown SMD value and MAC frame fragments rejected because of arriving with SMD-C when there was no preceding preempted frame.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>PSEC | Packet SMD Error Counter<br>Indicates the number of MAC frames rejected because of an unknown SMD value and MAC frame fragments rejected because of arriving with SMD-C when there was no preceding preempted frame.<br>Exists when at least one of the receive, transmit, or MMC counters is enabled during FPE-enabled configuration. |

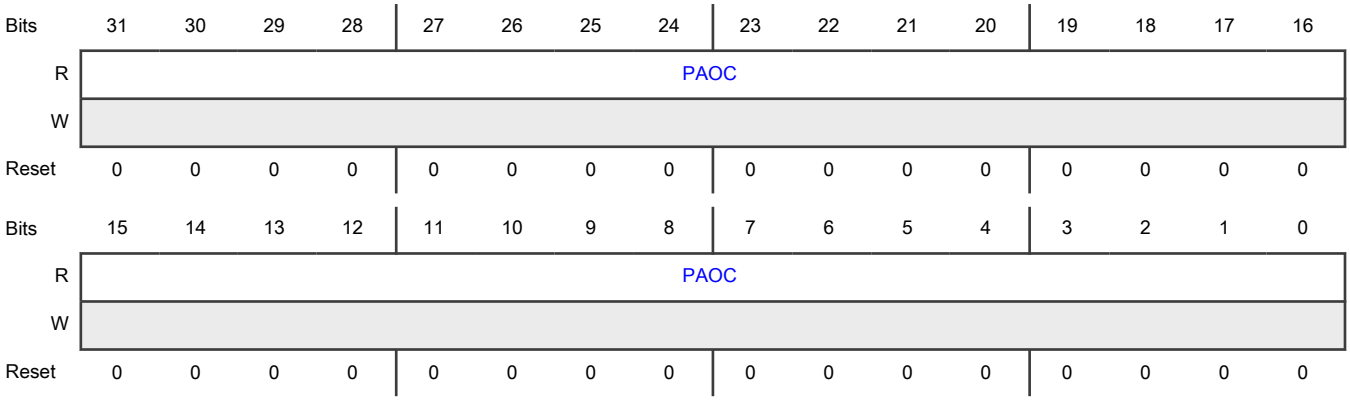
## 72.17.122 MMC Receive Packet Assembly OK Counter (MMC_Rx_Packet_Assembly_OK_Cntr)

### Offset

| Register | Offset |
|---|---|
| MMC_Rx_Packet_Assembly_OK_Cntr | 8D0h |

### Function
Provides the number of MAC frames that were successfully reassembled and delivered to MAC.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>PAOC | Packet Assembly OK Counter<br>Indicates the number of MAC frames that were successfully reassembled and delivered to MAC.<br>Exists when at least one of the receive, transmit, or MMC counters is enabled during FPE-enabled configuration. |

## 72.17.123 MMC Receive FPE Fragment Counter (MMC_Rx_FPE_Fragment_Cntr)

### Offset

| Register | Offset |
|---|---|
| MMC_Rx_FPE_Fragment_Cntr | 8D4h |

### Function
Provides the number of additional mPackets received because of preemption.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>FFC | FPE Fragment Counter<br>Indicates the number of additional mPackets received because of preemption.<br>Exists when at least one of the receive, transmit, or MMC counters is enabled during FPE-enabled configuration. |

## 72.17.124 MAC Layer 3 Layer 4 Control 0 (MAC_L3_L4_Control0)

### Offset

| Register | Offset |
|---|---|
| MAC_L3_L4_Control0 | 900h |

### Function
Controls the filter 0 operations of the layer 3 and layer 4 protocols.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | DMCHEN0 | Reserved | | | DMCHN0 | Reserved | | L4DPIM0 | L4DPM0 | L4SPIM0 | L4SPM0 | Reserved | L4PEN0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | L3HDBM0 | | | | L3HSBM0 | | | | | | L3DAIM0 | L3DAM0 | L3SAIM0 | L3SAM0 | Reserved | L3PEN0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |
| 28<br>DMCHEN0 | DMA Channel Select Enable<br>Indicates the status of the DMA channel number.<br>• If this field is 1, it enables the selection of the DMA channel number for the packet that the L3_L4 filter passes. The DMCHN fields indicate the DMA channel number.<br>• If this field is 0, the filter does not decide the DMA channel number.<br>0b - Disabled<br>1b - Enabled |
| 27-25<br>— | Reserved |
| 24<br>DMCHN0 | DMA Channel Number<br>Indicates the DMA channel number.<br>If the value of the DMCHEN fields is 1, this field selects the DMA channel number to which the packet that this filter passes is routed. The width of this field depends on the number of the DMA channels present in your configuration. |
| 23-22<br>— | Reserved |
| 21<br>L4DPIM0 | Layer 4 Destination Port Inverse Match Enable<br>Indicates the status of layer 4 destination port inverse matching.<br>• If this field is 1, MAC_Layer4_Address0[L4DP0] is enabled for inverse matching.<br>• If this field is 0, MAC_Layer4_Address0[L4DP0] is enabled for perfect matching.<br>This field is valid and applicable only when the L4DPM0 field is 1.<br>0b - Disabled<br>1b - Enabled |
| 20<br>L4DPM0 | Layer 4 Destination Port Match Enable<br>Indicates the status of layer 4 destination port matching.<br>• If this field is 1, MAC_Layer4_Address0[L4DP0] is enabled for matching.<br>• If this field is 0, MAC ignores MAC_Layer4_Address0[L4DP0] for matching.<br>0b - Disabled<br>1b - Enabled |
| 19 | Layer 4 Source Port Inverse Match Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| L4SPIM0 | Indicates the status of layer 4 source port inverse matching.<br>• If this field is 1, MAC_Layer4_Address0[L4SP0] is enabled for inverse matching.<br>• If this field is 0, MAC_Layer4_Address0[L4SP0] is enabled for perfect matching.<br>This field is valid and applicable only when MAC_Layer4_Address0[L4SP0] = 1.<br>0b - Disabled<br>1b - Enabled |
| 18<br>L4SPM0 | Layer 4 Source Port Match Enable<br>Indicates the status of layer 4 source port matching.<br>• If this field is 1, MAC_Layer4_Address0[L4SP0] is enabled for matching.<br>• If this field is 0, MAC ignores MAC_Layer4_Address0[L4SP0] for matching.<br>0b - Disabled<br>1b - Enabled |
| 17<br>— | Reserved |
| 16<br>L4PEN0 | Layer 4 Protocol Enable<br>Indicates the status of layer 4 protocol.<br>• If this field is 1, MAC_Layer4_Address0[L4SP0] and MAC_Layer4_Address0[L4DP0] are used for matching UDP packets.<br>• If this field is 0, MAC_Layer4_Address0[L4SP0] and MAC_Layer4_Address0[L4DP0] are used for matching TCP packets.<br>Layer 4 matching is performed only when the L4SPM0 field or the L4DPM0 field is 1.<br>0b - Disabled<br>1b - Enabled |
| 15-11<br>L3HDBM0 | Layer 3 IP DA Higher Bits Match<br>IPv4 packets:<br>This field contains the number of higher bits of IP DA that are matched in the IPv4 packets. The following list describes the values of this field:<br>• 0: No bits are masked.<br>• 1: LSb[0] is masked.<br>• 2: Two LSbs [1:0] are masked.<br>• - ..<br>• 31: All bits except MSb are masked.<br>IPv6 packets: |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Bits [12:11] of this field correspond to bits [6:5] of the L3HSBM0 field, which indicates the number of lower bits of IP SA or DA that are masked in the IPv6 packets. The following list describes the concatenated values of the L3HDBM0[1:0] and L3HSBM0 fields:<br>• 0: No bits are masked.<br>• 1: LSb[0] is masked.<br>• 2: Two LSbs [1:0] are masked.<br>• ..<br>• 127: All bits except MSb are masked.<br>This field is valid and applicable only when the L3DAM0 field or the L3SAM0 field is 1. |
| 10-6<br>L3HSBM0 | Layer 3 IP SA Higher Bits Match<br>IPv4 packets:<br>This field contains the number of lower bits of IP SA that are masked for matching in the IPv4 packets. The following list describes the values of this field:<br>• 0: No bits are masked.<br>• 1: LSb[0] is masked.<br>• 2: Two LSbs [1:0] are masked.<br>• ..<br>• 31: All bits except MSb are masked.<br>IPv6 packets:<br>This field contains bits [4:0] of L3HSBM0. These bits indicate the number of higher bits of IP SA or DA matched in the IPv6 packets. The field is valid and applicable only when the L3DAM0 field or the L3SAM0 field is 1. |
| 5<br>L3DAIM0 | Layer 3 IP DA Inverse Match Enable<br>Indicates the status of layer 3 IP DA inverse matching.<br>• If this field is 1, layer 3 IP DA is enabled for inverse matching.<br>• If this field is 0, layer 3 IP DA is enabled for perfect matching.<br>This field is valid and applicable only if L3DAM0 = 1.<br>0b - Disabled<br>1b - Enabled |
| 4<br>L3DAM0 | Layer 3 IP DA Match Enable<br>Indicates the status of layer 3 IP DA matching.<br>• If this field is 1, layer 3 IP DA is enabled for matching.<br>• If this field is 0, MAC ignores layer 3 IP DA for matching. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>If L3PEN0 = 1, you must write 1 to either this field or the L3SAM0 field because either the IPv6 DA or SA can be checked for filtering.<br>0b - Disabled<br>1b - Enabled |
| 3<br>L3SAIM0 | Layer 3 IP SA Inverse Match Enable<br>Indicates the status of layer 3 IP SA inverse matching.<br>• If this field is 1, layer 3 IP SA is enabled for inverse matching.<br>• If this field is 0, layer 3 IP SA is enabled for perfect matching.<br>This field is valid and applicable only if the L3SAM0 field is 1.<br>0b - Disabled<br>1b - Enabled |
| 2<br>L3SAM0 | Layer 3 IP SA Match Enable<br>Indicates the status of layer 3 IP SA matching.<br>• If this field is 1, layer 3 IP SA matching is enabled.<br>• If this field is 0, MAC ignores layer 3 IP SA matching.<br>**NOTE**<br>If L3PEN0 = 1, you must write 1 to either this field or the L3DAM0 field because either IPv6 SA or DA can be checked for filtering.<br>0b - Disabled<br>1b - Enabled |
| 1<br>— | Reserved |
| 0<br>L3PEN0 | Layer 3 Protocol Enable<br>Indicates the status of layer 3 protocol.<br>• If this field is 1, the layer 3 IP SA or DA matching is enabled for IPv6 packets.<br>• If this field is 0, the layer 3 IP SA or DA matching is enabled for IPv4 packets.<br>The layer 3 matching is performed only when either the L3SAM0 field or the L3DAM0 field is 1.<br>0b - Disabled<br>1b - Enabled |

## 72.17.125 MAC Layer 4 Address 0 (MAC_Layer4_Address0)

### Offset

| Register | Offset |
|---|---|
| MAC_Layer4_Address0 | 904h |

### Function

Provides the layer 4 source and destination port numbers.

MAC Layer 4 Address 1 (MAC_Layer4_Address1), MAC L3 L4 Control 1 (MAC_L3_L4_Control1), MAC Layer 3 Address 0 Reg 1 (MAC_Layer3_Addr0_Reg1), MAC Layer 3 Address 1 Reg 1 (MAC_Layer3_Addr1_Reg1), MAC Layer 3 Address 2 Reg 1 (MAC_Layer3_Addr2_Reg1), and MAC Layer 3 Address 3 Reg 1 (MAC_Layer3_Addr3_Reg1) are reserved registers (RO with a default value) if MAC_Packet_Filter[IPFE] is 0 when configuring the core.

You can configure the layer 3 and layer 4 address registers to be double-synchronized by selecting the synchronize layer 3 and layer 4 address registers to the receive clock domain option while configuring the core. When you select this option, the synchronization is triggered only when bits [31:24] (in Little-Endian mode) or bits [7:0] (in Big-Endian mode) of the layer 3 and layer 4 address registers are written to. For proper synchronization updates, you must perform consecutive writes to the same layer 3 and layer 4 address registers after a delay of at least four destination clock cycles.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | L4DP0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | L4SP0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-16<br>L4DP0 | Layer 4 Destination Port Number<br>Indicates layer 4 destination port number.<br>If MAC_L3_L4_Control0[L4PEN0] = 0 and MAC_L3_L4_Control0[L4DPM0] = 1, this field contains the value to be matched with the TCP destination port number field in the IPv4 or IPv6 packets.<br>If MAC_L3_L4_Control0[L4PEN0] and MAC_L3_L4_Control0[L4DPM0] are 1, this field contains the value to be matched with the UDP destination port number field in the IPv4 or IPv6 packets. |
| 15-0<br>L4SP0 | Layer 4 Source Port Number<br>Indicates layer 4 source port number. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | If MAC_L3_L4_Control0[L4PEN0] = 0 and MAC_L3_L4_Control0[L4SPM0] = 1, this field contains the value to be matched with the TCP source port number field in the IPv4 or IPv6 packets.<br>If MAC_L3_L4_Control0[L4PEN0] and MAC_L3_L4_Control0[L4SPM0] are 1, this field contains the value to be matched with the UDP source port number field in the IPv4 or IPv6 packets. |

## 72.17.126 MAC Layer 3 Address 0 Reg 0 (MAC_Layer3_Addr0_Reg0)

### Offset

| Register | Offset |
|---|---|
| MAC_Layer3_Addr0_Reg0 | 910h |

### Function

Contains the 32-bit IP source address field for IPv4 packets. For IPv6 packets, the field contains bits [31:0] of the 128-bit IP source address or destination address field.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A00 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A00 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>L3A00 | Layer 3 Address 0<br>Indicates layer 3 address 0.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3SAM0] are 1, this field contains the value to be matched with bits [31:0] of the IP source address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3DAM0] are 1, this field contains the value to be matched with bits [31:0] of the IP destination address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] = 0, and MAC_L3_L4_Control0[L3SAM0] = 1, this field contains the value to be matched with the IP source address field in the IPv4 packets. |

## 72.17.127 MAC Layer 3 Address 1 Reg 0 (MAC_Layer3_Addr1_Reg0)

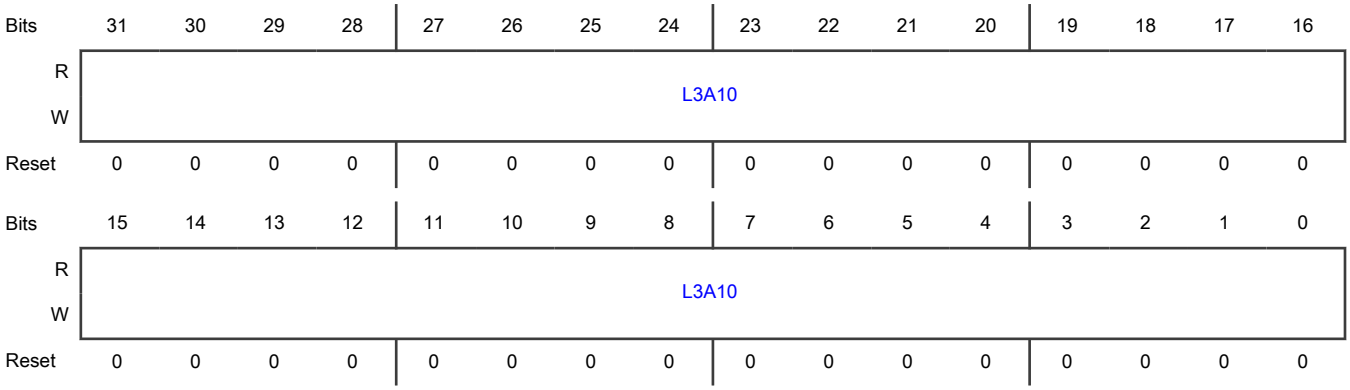
### Offset

| Register | Offset |
|---|---|
| MAC_Layer3_Addr1_Reg0 | 914h |

### Function

Contains the 32-bit IP destination address field for IPv4 packets. For IPv6 packets, the field contains bits [63:32] of the 128-bit IP source address or destination address field.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>L3A10 | Layer 3 Address 1<br>Indicates layer 3 address 1.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3SAM0] are 1, this field contains the value to be matched with bits [63:32] of the IP source address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3DAM0] are 1, this field contains the value to be matched with bits [63:32] of the IP destination address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] = 0, and MAC_L3_L4_Control0[L3SAM0] = 1, this field contains the value to be matched with the IP source address field in the IPv4 packets. |

## 72.17.128 MAC Layer 3 Address 2 Reg 0 (MAC_Layer3_Addr2_Reg0)

### Offset

| Register | Offset |
|---|---|
| MAC_Layer3_Addr2_Reg0 | 918h |

### Function

Contains the 32-bit IP destination address field for IPv4 packets. For IPv6 packets, the field contains bits [95:64] of the 128-bit IP source address or destination address field.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A20 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A20 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>L3A20 | Layer 3 Address 2<br>Indicates layer 3 address 2.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3SAM0] are 1, this field contains the value to be matched with bits [95:64] of the IP source address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3DAM0] are 1, this field contains the value to be matched with bits [95:64] of the IP destination address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] = 0, this field is not used. |

## 72.17.129 MAC Layer 3 Address 3 Reg 0 (MAC_Layer3_Addr3_Reg0)

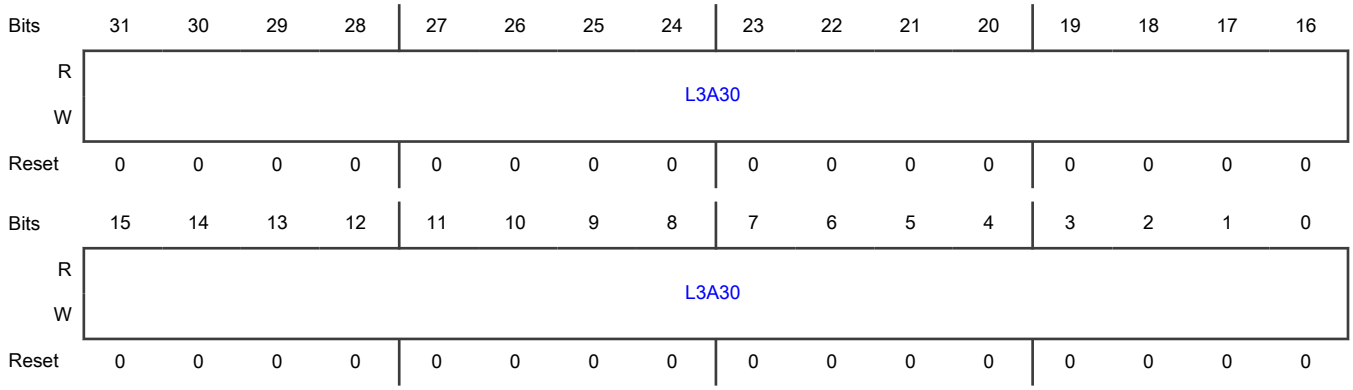
### Offset

| Register | Offset |
|---|---|
| MAC_Layer3_Addr3_Reg0 | 91Ch |

### Function

Contains the 32-bit IP destination address field for IPv4 packets. For IPv6 packets, the field contains bits [127:96] of the 128-bit IP source address or destination address field.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>L3A30 | Layer 3 Address 3<br>Indicates layer 3 address 3.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3SAM0] are 1, this field contains the value to be matched with bits [127:96] of the IP source address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3DAM0] are 1, this field contains the value to be matched with bits [127:96] of the IP destination address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] = 0, this field is not used. |

## 72.17.130 MAC L3 L4 Control 1 (MAC_L3_L4_Control1)

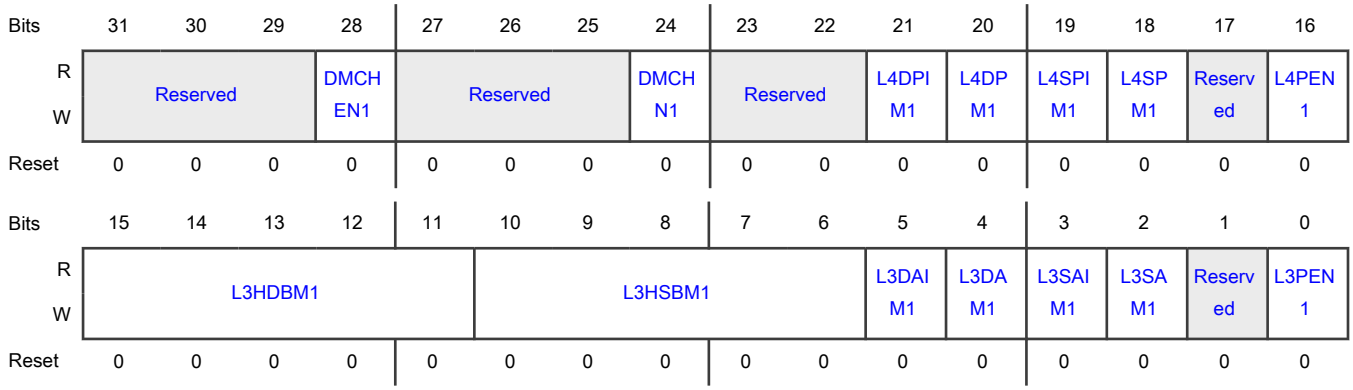
### Offset

| Register | Offset |
|---|---|
| MAC_L3_L4_Control1 | 930h |

### Function

Controls the filter 0 operations of the layer 3 and layer 4 protocols.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |
| 28<br>DMCHEN1 | DMA Channel Select Enable 1<br>Indicates the status of the DMA channel number.<br>• If this field is 1, it enables the selection of the DMA channel number for the packet that the L3_L4 filter passes. The DMCHN fields indicate the DMA channel number.<br>• If this field is 0, the filter does not decide the DMA channel number.<br>0b - Disabled<br>1b - Enabled |
| 27-25<br>— | Reserved |
| 24<br>DMCHN1 | DMA Channel Number 1<br>Indicates the DMA channel number.<br>If the value of the DMCHEN fields is 1, this field selects the DMA channel number to which the packet that this filter passes is routed. The width of this field depends on the number of the DMA channels present in your configuration. |
| 23-22<br>— | Reserved |
| 21<br>L4DPIM1 | Layer 4 Destination Port Inverse Match Enable 1<br>Indicates the status of layer 4 destination port inverse matching.<br>• If this field is 1, MAC_Layer4_Address0[L4DP0] is enabled for inverse matching.<br>• If this field is 0, MAC_Layer4_Address0[L4DP0] is enabled for perfect matching.<br>This field is valid and applicable only when the L4DPM0 field is 1. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |
| 20<br>L4DPM1 | Layer 4 Destination Port Match Enable 1<br>Indicates the status of layer 4 destination port matching.<br>• If this field is 1, MAC_Layer4_Address0[L4DP0] is enabled for matching.<br>• If this field is 0, MAC ignores MAC_Layer4_Address0[L4DP0] for matching.<br>0b - Disabled<br>1b - Enabled |
| 19<br>L4SPIM1 | Layer 4 Source Port Inverse Match Enable 1<br>Indicates the status of layer 4 source port inverse matching.<br>• If this field is 1, MAC_Layer4_Address0[L4SP0] is enabled for inverse matching.<br>• If this field is 0, MAC_Layer4_Address0[L4SP0] is enabled for perfect matching.<br>This field is valid and applicable only when MAC_Layer4_Address0[L4SP0] = 1.<br>0b - Disabled<br>1b - Enabled |
| 18<br>L4SPM1 | Layer 4 Source Port Match Enable 1<br>Indicates the status of layer 4 source port matching.<br>• If this field is 1, MAC_Layer4_Address0[L4SP0] is enabled for matching.<br>• If this field is 0, MAC ignores MAC_Layer4_Address0[L4SP0] for matching.<br>0b - Disabled<br>1b - Enabled |
| 17<br>— | Reserved |
| 16<br>L4PEN1 | Layer 4 Protocol Enable 1<br>Indicates the status of layer 4 protocol.<br>• If this field is 1, MAC_Layer4_Address0[L4SP0] and MAC_Layer4_Address0[L4DP0] of UDP packets are used for matching.<br>• If this field is 0, MAC_Layer4_Address0[L4SP0] and MAC_Layer4_Address0[L4DP0] of TCP packets are used for matching.<br>Layer 4 matching is performed only when the L4SPM0 field or the L4DPM0 field is 1.<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 15-11<br>L3HDBM1 | Layer 3 IP DA Higher Bits Match 1<br>IPv4 Packets:<br>This field contains the number of higher bits of IP DA that are matched in the IPv4 packets. The following list describes the values of this field:<br>• 0: No bits are masked.<br>• 1: LSb[0] is masked.<br>• 2: Two LSbs [1:0] are masked.<br>• ..<br>• 31: All bits except MSb are masked.<br>IPv6 Packets:<br>Bits [12:11] of this field correspond to bits [6:5] of MAC_L3_L4_Control0[L3HSBM0], which indicates the number of lower bits of IP SA or DA that are masked in the IPv6 packets. The following list describes the concatenated values of MAC_L3_L4_Control0[L3HDBM0][1:0] and MAC_L3_L4_Control0[L3HSBM0]:<br>• 0: No bits are masked.<br>• 1: LSb[0] is masked.<br>• 2: Two LSbs [1:0] are masked<br>• ..<br>• 127: All bits except MSb are masked.<br>This field is valid and applicable only when MAC_L3_L4_Control0[L3DAM0] or MAC_L3_L4_Control0[L3SAM0] is 1. |
| 10-6<br>L3HSBM1 | Layer 3 IP SA Higher Bits Match 1<br>IPv4 packets:<br>This field contains the number of lower bits of IP SA that are masked for matching in the IPv4 packets. The following list describes the values of this field:<br>• 0: No bits are masked.<br>• 1: LSb[0] is masked.<br>• 2: Two LSbs [1:0] are masked.<br>• ..<br>• 31: All bits except MSb are masked.<br>IPv6 packets:<br>This field contains bits [4:0] of MAC_L3_L4_Control0[L3HSBM0]. These bits indicate the number of higher bits of IP SA or DA matched in the IPv6 packets. The field is valid and applicable only when MAC_L3_L4_Control0[L3DAM0] or MAC_L3_L4_Control0[L3SAM0] is 1. |
| 5<br>L3DAIM1 | Layer 3 IP DA Inverse Match Enable 1<br>Indicates the status of layer 3 IP DA inverse matching. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • If this field is 1, layer 3 IP DA is enabled for inverse matching.<br>• If this field is 0, layer 3 IP DA is enabled for perfect matching.<br>This field is valid and applicable only if MAC_L3_L4_Control0[L3DAM0] = 1.<br>0b - Disabled<br>1b - Enabled |
| 4<br>L3DAM1 | Layer 3 IP DA Match Enable 1<br>Indicates the status of layer 3 IP DA matching.<br>• If this field is 1, layer 3 IP DA is enabled for matching.<br>• If this field is 0, MAC ignores layer 3 IP DA for matching.<br>**NOTE**<br>If MAC_L3_L4_Control0[L3PEN0] = 1, you must write 1 to either this field or to MAC_L3_L4_Control0[L3SAM0] because either the IPv6 DA or SA can be checked for filtering.<br>0b - Disabled<br>1b - Enabled |
| 3<br>L3SAIM1 | Layer 3 IP SA Inverse Match Enable 1<br>Indicates the status of layer 3 IP SA inverse matching.<br>• If this field is 1, layer 3 IP SA is enabled for inverse matching.<br>• If this field is 0, layer 3 IP SA is enabled for perfect matching.<br>This field is valid and applicable only if MAC_L3_L4_Control0[L3SAM0] is 1.<br>0b - Disabled<br>1b - Enabled |
| 2<br>L3SAM1 | Layer 3 IP SA Match Enable 1<br>Indicates the status of layer 3 IP SA matching.<br>• If this field is 1, layer 3 IP SA matching is enabled.<br>• If this field is 0, MAC ignores layer 3 IP SA matching.<br>**NOTE**<br>If MAC_L3_L4_Control0[L3PEN0] = 1, you must write 1 to either this field or to MAC_L3_L4_Control0[L3DAM0] because either IPv6 SA or DA can be checked for filtering.<br>0b - Disabled<br>1b - Enabled |
| 1<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 0<br>L3PEN1 | Layer 3 Protocol Enable 1<br>Indicates the status of layer 3 protocol.<br>• If this field is 1, the layer 3 IP SA or DA matching is enabled for IPv6 packets.<br>• If this field is 0, the layer 3 IP SA or DA matching is enabled for IPv4 packets.<br>The layer 3 matching is performed only when either MAC_L3_L4_Control0[L3SAM0] or MAC_L3_L4_Control0[L3DAM0] is 1.<br>0b - Disabled<br>1b - Enabled |

## 72.17.131 MAC Layer 4 Address 1 (MAC_Layer4_Address1)

### Offset

| Register | Offset |
|---|---|
| MAC_Layer4_Address1 | 934h |

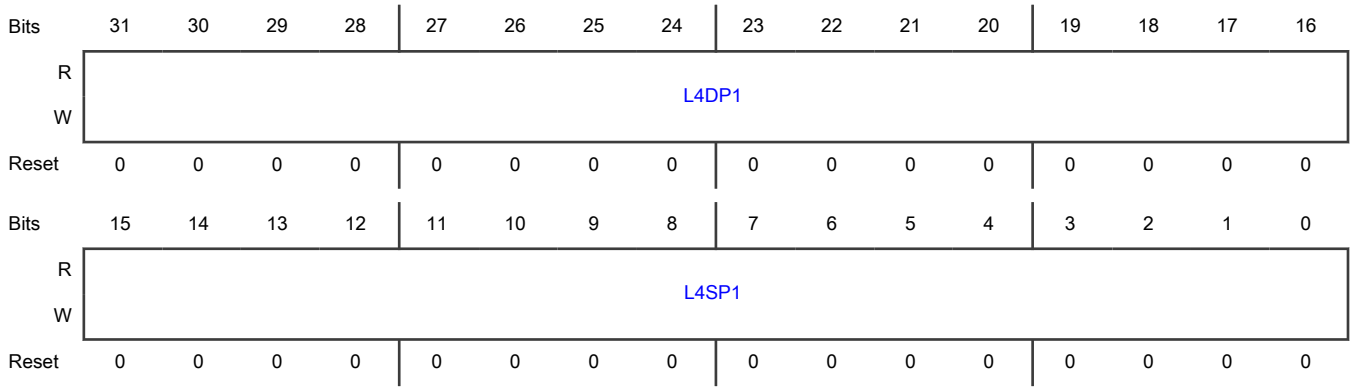
### Function

Provides the layer 4 source and destination port numbers.

MAC Layer 4 Address 1 (MAC_Layer4_Address1), MAC L3 L4 Control 1 (MAC_L3_L4_Control1), MAC Layer 3 Address 0 Reg 1 (MAC_Layer3_Addr0_Reg1), MAC Layer 3 Address 1 Reg 1 (MAC_Layer3_Addr1_Reg1), MAC Layer 3 Address 2 Reg 1 (MAC_Layer3_Addr2_Reg1), and MAC Layer 3 Address 3 Reg 1 (MAC_Layer3_Addr3_Reg1) are reserved registers (RO with a default value) if the enable layer 3 and layer 4 packet filter option is not selected when configuring the core.

You can configure the layer 3 and layer 4 address registers to be double-synchronized by selecting the synchronize layer 3 and layer 4 address registers to the receive clock domain option while configuring the core. When you select this option, the synchronization is triggered only when bits [31:24] (in Little-Endian mode) or bits [7:0] (in Big-Endian mode) of the layer 3 and layer 4 address registers are written to. For proper synchronization updates, you must perform consecutive writes to the same layer 3 and layer 4 address registers after a delay of at least four destination clock cycles.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-16<br>L4DP1 | Layer 4 Destination Port Number 1<br>Indicates layer 4 destination port number.<br>If MAC_L3_L4_Control0[L4PEN0] = 0 and MAC_L3_L4_Control0[L4DPM0] = 1, this field contains the value to be matched with the TCP destination port number field in the IPv4 or IPv6 packets.<br>If MAC_L3_L4_Control0[L4PEN0] and MAC_L3_L4_Control0[L4DPM0] are 1, this field contains the value to be matched with the UDP destination port number field in the IPv4 or IPv6 packets. |
| 15-0<br>L4SP1 | Layer 4 Source Port Number 1<br>Indicates layer 4 source port number.<br>If MAC_L3_L4_Control0[L4PEN0] = 0 and MAC_L3_L4_Control0[L4SPM0] = 1, this field contains the value to be matched with the TCP source port number field in the IPv4 or IPv6 packets.<br>If MAC_L3_L4_Control0[L4PEN0] and MAC_L3_L4_Control0[L4SPM0] are 1, this field contains the value to be matched with the UDP source port number field in the IPv4 or IPv6 packets. |

## 72.17.132 MAC Layer 3 Address 0 Reg 1 (MAC_Layer3_Addr0_Reg1)

### Offset

| Register | Offset |
|---|---|
| MAC_Layer3_Addr0_Reg1 | 940h |

### Function

Contains the 32-bit IP source address field for IPv4 packets. For IPv6 packets, the field contains bits [31:0] of the 128-bit IP source address or destination address field.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A01 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A01 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>L3A01 | Layer 3 Address 0<br>Indicates layer 3 address 0.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3SAM0] are 1, this field contains the value to be matched with bits [31:0] of the IP source address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3DAM0] are 1, this field contains the value to be matched with bits [31:0] of the IP destination address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] = 0, and MAC_L3_L4_Control0[L3SAM0] = 1, this field contains the value to be matched with the IP source address field in the IPv4 packets. |

## 72.17.133 MAC Layer 3 Address 1 Reg 1 (MAC_Layer3_Addr1_Reg1)

### Offset

| Register | Offset |
|---|---|
| MAC_Layer3_Addr1_Reg1 | 944h |

### Function

Contains the 32-bit IP destination address field for IPv4 packets. For IPv6 packets, the field contains bits[63:32] of the 128-bit IP source address or destination address field.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A11 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A11 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>L3A11 | Layer 3 Address 1<br>Indicates layer 3 address 1. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3SAM0] are 1, this field contains the value to be matched with bits [63:32] of the IP source address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3DAM0] are 1, this field contains the value to be matched with bits [63:32] of the IP destination address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] = 0, and MAC_L3_L4_Control0[L3SAM0] = 1, this field contains the value to be matched with the IP source address field in the IPv4 packets. |

## 72.17.134 MAC Layer 3 Address 2 Reg 1 (MAC_Layer3_Addr2_Reg1)

### Offset

| Register | Offset |
|---|---|
| MAC_Layer3_Addr2_Reg1 | 948h |

### Function

Contains the 32-bit IP destination address field for IPv4 packets. For IPv6 packets, the field contains bits [95:64] of the 128-bit IP source address or destination address field.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A21 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A21 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>L3A21 | Layer 3 Address 2<br>Indicates layer 3 address 2.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3SAM0] are 1, this field contains the value to be matched with bits [95:64] of the IP source address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3DAM0] are 1, this field contains the value to be matched with bits [95:64] of the IP destination address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] = 0, this field is not used. |

## 72.17.135 MAC Layer 3 Address 3 Reg 1 (MAC_Layer3_Addr3_Reg1)

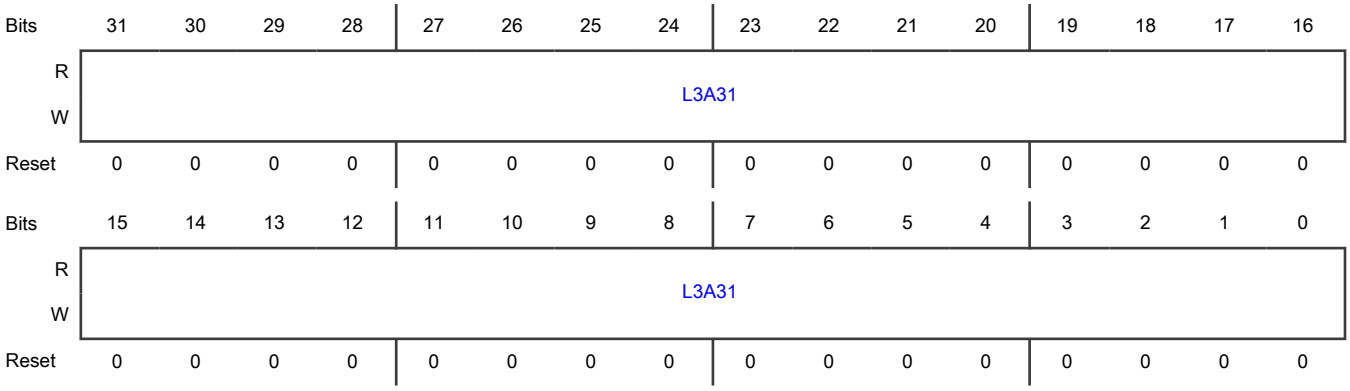
### Offset

| Register | Offset |
|---|---|
| MAC_Layer3_Addr3_Reg1 | 94Ch |

### Function

Contains the 32-bit IP destination address field for IPv4 packets. For IPv6 packets, the field contains bits [127:96] of the 128-bit IP source address or destination address field.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>L3A31 | Layer 3 Address 3<br>Indicates layer 3 address 3.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3SAM0] are 1, this field contains the value to be matched with bits [127:96] of the IP source address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3DAM0] are 1, this field contains the value to be matched with bits [127:96] of the IP destination address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] = 0, this field is not used. |

## 72.17.136 MAC L3 L4 Control 2 (MAC_L3_L4_Control2)

### Offset

| Register | Offset |
|---|---|
| MAC_L3_L4_Control2 | 960h |

### Function

Controls the filter 0 operations of the layer 3 and layer 4 protocols.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | DMCHEN2 | Reserved | | | DMCHN2 | Reserved | | L4DPIM2 | L4DPM2 | L4SPIM2 | L4SPM2 | Reserved | L4PEN2 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3HDBM2 | | | | L3HSBM2 | | | | | | L3DAIM2 | L3DAM2 | L3SAIM2 | L3SAM2 | Reserved | L3PEN2 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |
| 28<br>DMCHEN2 | DMA Channel Select Enable 2<br>Indicates the status of the DMA channel number.<br>• If this field is 1, it enables the selection of the DMA channel number for the packet that the L3_L4 filter passes. The DMCHN fields indicate the DMA channel number.<br>• If this field is 0, the filter does not decide the DMA channel number.<br>0b - Disabled<br>1b - Enabled |
| 27-25<br>— | Reserved |
| 24<br>DMCHN2 | DMA Channel Number 2<br>Indicates the DMA channel number.<br>If the value of the DMCHEN fields is 1, this field selects the DMA channel number to which the packet that this filter passes is routed. The width of this field depends on the number of the DMA channels present in your configuration. |
| 23-22<br>— | Reserved |
| 21<br>L4DPIM2 | Layer 4 Destination Port Inverse Match Enable 2<br>Indicates the status of layer 4 destination port inverse matching.<br>• If this field is 1, MAC_Layer4_Address0[L4DP0] is enabled for inverse matching.<br>• If this field is 0, MAC_Layer4_Address0[L4DP0] is enabled for perfect matching.<br>This field is valid and applicable only when the L4DPM0 field is 1. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |
| 20<br>L4DPM2 | Layer 4 Destination Port Match Enable 2<br>Indicates the status of layer 4 destination port matching.<br>• If this field is 1, MAC_Layer4_Address0[L4DP0] is enabled for matching.<br>• If this field is 0, MAC ignores MAC_Layer4_Address0[L4DP0] for matching.<br>0b - Disabled<br>1b - Enabled |
| 19<br>L4SPIM2 | Layer 4 Source Port Inverse Match Enable 2<br>Indicates the status of layer 4 source port inverse matching.<br>• If this field is 1, MAC_Layer4_Address0[L4SP0] is enabled for inverse matching.<br>• If this field is 0, MAC_Layer4_Address0[L4SP0] is enabled for perfect matching.<br>This field is valid and applicable only when MAC_Layer4_Address0[L4SP0] = 1.<br>0b - Disabled<br>1b - Enabled |
| 18<br>L4SPM2 | Layer 4 Source Port Match Enable 2<br>Indicates the status of layer 4 source port matching.<br>• If this field is 1, MAC_Layer4_Address0[L4SP0] is enabled for matching.<br>• If this field is 0, MAC ignores MAC_Layer4_Address0[L4SP0] for matching.<br>0b - Disabled<br>1b - Enabled |
| 17<br>— | Reserved |
| 16<br>L4PEN2 | Layer 4 Protocol Enable 2<br>Indicates the status of layer 4 protocol.<br>• If this field is 1, MAC_Layer4_Address0[L4SP0] and MAC_Layer4_Address0[L4DP0] of UDP packets are used for matching.<br>• If this field is 0, MAC_Layer4_Address0[L4SP0] and MAC_Layer4_Address0[L4DP0] of TCP packets are used for matching.<br>Layer 4 matching is performed only when the L4SPM0 field or the L4DPM0 field is 1.<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 15-11<br>L3HDBM2 | Layer 3 IP DA Higher Bits Match 2<br>IPv4 Packets:<br>This field contains the number of higher bits of IP DA that are matched in the IPv4 packets. The following list describes the values of this field:<br>• 0: No bits are masked.<br>• 1: LSb[0] is masked.<br>• 2: Two LSbs [1:0] are masked.<br>• ..<br>• 31: All bits except MSb are masked.<br>IPv6 Packets:<br>Bits [12:11] of this field correspond to bits [6:5] of MAC_L3_L4_Control0[L3HSBM0], which indicates the number of lower bits of IP SA or DA that are masked in the IPv6 packets. The following list describes the concatenated values of MAC_L3_L4_Control0[L3HDBM0][1:0] and MAC_L3_L4_Control0[L3HSBM0]:<br>• 0: No bits are masked.<br>• 1: LSb[0] is masked.<br>• 2: Two LSbs [1:0] are masked<br>• ..<br>• 127: All bits except MSb are masked.<br>This field is valid and applicable only when MAC_L3_L4_Control0[L3DAM0] or MAC_L3_L4_Control0[L3SAM0] is 1. |
| 10-6<br>L3HSBM2 | Layer 3 IP SA Higher Bits Match 2<br>IPv4 packets:<br>This field contains the number of lower bits of IP SA that are masked for matching in the IPv4 packets. The following list describes the values of this field:<br>• 0: No bits are masked.<br>• 1: LSb[0] is masked.<br>• 2: Two LSbs [1:0] are masked.<br>• ..<br>• 31: All bits except MSb are masked.<br>IPv6 packets:<br>This field contains bits [4:0] of MAC_L3_L4_Control0[L3HSBM0]. These bits indicate the number of higher bits of IP SA or DA matched in the IPv6 packets. The field is valid and applicable only when MAC_L3_L4_Control0[L3DAM0] or MAC_L3_L4_Control0[L3SAM0] is 1. |
| 5<br>L3DAIM2 | Layer 3 IP DA Inverse Match Enable 2<br>Indicates the status of layer 3 IP DA inverse matching. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • If this field is 1, layer 3 IP DA is enabled for inverse matching.<br>• If this field is 0, layer 3 IP DA is enabled for perfect matching.<br>This field is valid and applicable only if MAC_L3_L4_Control0[L3DAM0] = 1.<br>0b - Disabled<br>1b - Enabled |
| 4<br>L3DAM2 | Layer 3 IP DA Match Enable 2<br>Indicates the status of layer 3 IP DA matching.<br>• If this field is 1, layer 3 IP DA is enabled for matching.<br>• If this field is 0, MAC ignores layer 3 IP DA for matching.<br>**NOTE**<br>If MAC_L3_L4_Control0[L3PEN0] = 1, you must write 1 to either this field or to MAC_L3_L4_Control0[L3SAM0] because either the IPv6 DA or SA can be checked for filtering.<br>0b - Disabled<br>1b - Enabled |
| 3<br>L3SAIM2 | Layer 3 IP SA Inverse Match Enable 2<br>Indicates the status of layer 3 IP SA inverse matching.<br>• If this field is 1, layer 3 IP SA is enabled for inverse matching.<br>• If this field is 0, layer 3 IP SA is enabled for perfect matching.<br>This field is valid and applicable only if MAC_L3_L4_Control0[L3SAM0] is 1.<br>0b - Disabled<br>1b - Enabled |
| 2<br>L3SAM2 | Layer 3 IP SA Match Enable 2<br>Indicates the status of layer 3 IP SA matching.<br>• If this field is 1, layer 3 IP SA matching is enabled.<br>• If this field is 0, MAC ignores layer 3 IP SA matching.<br>**NOTE**<br>If MAC_L3_L4_Control0[L3PEN0] = 1, you must write 1 to either this field or to MAC_L3_L4_Control0[L3DAM0] because either IPv6 SA or DA can be checked for filtering.<br>0b - Disabled<br>1b - Enabled |
| 1<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 0<br>L3PEN2 | Layer 3 Protocol Enable 2<br>Indicates the status of layer 3 protocol.<br>• If this field is 1, the layer 3 IP SA or DA matching is enabled for IPv6 packets.<br>• If this field is 0, the layer 3 IP SA or DA matching is enabled for IPv4 packets.<br>The layer 3 matching is performed only when either MAC_L3_L4_Control0[L3SAM0] or MAC_L3_L4_Control0[L3DAM0] is 1.<br>0b - Disabled<br>1b - Enabled |

## 72.17.137 MAC Layer 4 Address 2 (MAC_Layer4_Address2)

### Offset

| Register | Offset |
|---|---|
| MAC_Layer4_Address2 | 964h |

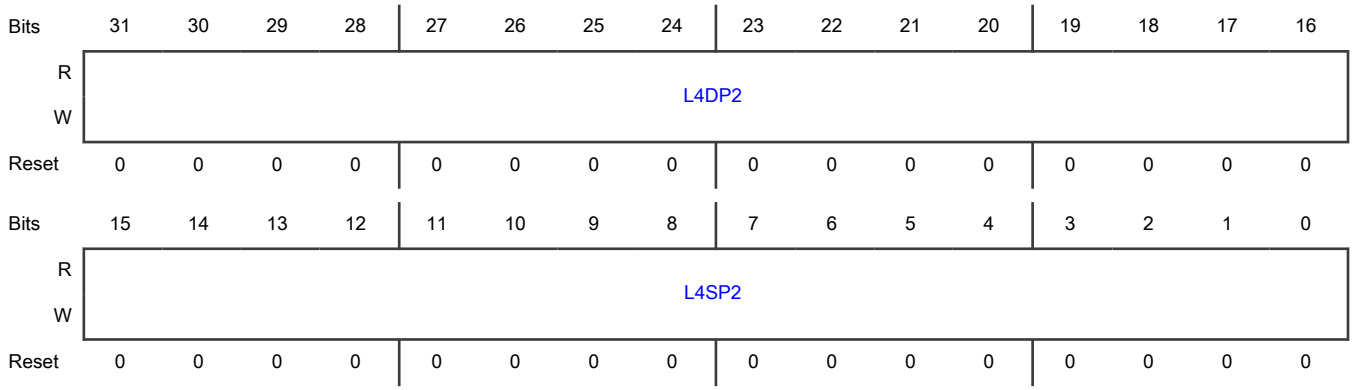
### Function

Provides the layer 4 source and destination port numbers.

MAC Layer 4 Address 1 (MAC_Layer4_Address1), MAC L3 L4 Control 1 (MAC_L3_L4_Control1), MAC Layer 3 Address 0 Reg 1 (MAC_Layer3_Addr0_Reg1), MAC Layer 3 Address 1 Reg 1 (MAC_Layer3_Addr1_Reg1), MAC Layer 3 Address 2 Reg 1 (MAC_Layer3_Addr2_Reg1), and MAC Layer 3 Address 3 Reg 1 (MAC_Layer3_Addr3_Reg1) are reserved registers (RO with a default value) if the enable layer 3 and layer 4 packet filter option is not selected when configuring the core.

You can configure the layer 3 and layer 4 address registers to be double-synchronized by selecting the synchronize layer 3 and layer 4 address registers to the receive clock domain option while configuring the core. When you select this option, the synchronization is triggered only when bits [31:24] (in Little-Endian mode) or bits [7:0] (in Big-Endian mode) of the layer 3 and layer 4 address registers are written to. For proper synchronization updates, you must perform consecutive writes to the same layer 3 and layer 4 address registers after a delay of at least four destination clock cycles.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-16<br>L4DP2 | Layer 4 Destination Port Number 2<br>Indicates layer 4 destination port number.<br>If MAC_L3_L4_Control0[L4PEN0] = 0 and MAC_L3_L4_Control0[L4DPM0] = 1, this field contains the value to be matched with the TCP destination port number field in the IPv4 or IPv6 packets.<br>If MAC_L3_L4_Control0[L4PEN0] and MAC_L3_L4_Control0[L4DPM0] are 1, this field contains the value to be matched with the UDP destination port number field in the IPv4 or IPv6 packets. |
| 15-0<br>L4SP2 | Layer 4 Source Port Number 2<br>Indicates layer 4 source port number.<br>If MAC_L3_L4_Control0[L4PEN0] = 0 and MAC_L3_L4_Control0[L4SPM0] = 1, this field contains the value to be matched with the TCP source port number field in the IPv4 or IPv6 packets.<br>If MAC_L3_L4_Control0[L4PEN0] and MAC_L3_L4_Control0[L4SPM0] are 1, this field contains the value to be matched with the UDP source port number field in the IPv4 or IPv6 packets. |

## 72.17.138 MAC Layer 3 Address 0 Reg 2 (MAC_Layer3_Addr0_Reg2)

### Offset

| Register | Offset |
|---|---|
| MAC_Layer3_Addr0_Reg2 | 970h |

### Function

Contains the 32-bit IP source address field for IPv4 packets. For IPv6 packets, the field contains bits [31:0] of the 128-bit IP source address or destination address field.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A02 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A02 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>L3A02 | Layer 3 Address 0<br>Indicates layer 3 address 0.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3SAM0] are 1, this field contains the value to be matched with bits [31:0] of the IP source address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3DAM0] are 1, this field contains the value to be matched with bits [31:0] of the IP destination address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] = 0, and MAC_L3_L4_Control0[L3SAM0] = 1, this field contains the value to be matched with the IP source address field in the IPv4 packets. |

## 72.17.139 MAC Layer 3 Address 1 Reg 2 (MAC_Layer3_Addr1_Reg2)

### Offset

| Register | Offset |
|---|---|
| MAC_Layer3_Addr1_Reg2 | 974h |

### Function

Contains the 32-bit IP destination address field for IPv4 packets. For IPv6 packets, the field contains bits[63:32] of the 128-bit IP source address or destination address field.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A12 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A12 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>L3A12 | Layer 3 Address 1<br>Indicates layer 3 address 1. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3SAM0] are 1, this field contains the value to be matched with bits [63:32] of the IP source address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3DAM0] are 1, this field contains the value to be matched with bits [63:32] of the IP destination address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] = 0, and MAC_L3_L4_Control0[L3SAM0] = 1, this field contains the value to be matched with the IP source address field in the IPv4 packets. |

## 72.17.140 MAC Layer 3 Address 2 Reg 2 (MAC_Layer3_Addr2_Reg2)

### Offset

| Register | Offset |
|---|---|
| MAC_Layer3_Addr2_Reg2 | 978h |

### Function

Contains the 32-bit IP destination address field for IPv4 packets. For IPv6 packets, the field contains bits [95:64] of the 128-bit IP source address or destination address field.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A22 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A22 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>L3A22 | Layer 3 Address 2<br>Indicates layer 3 address 2.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3SAM0] are 1, this field contains the value to be matched with bits [95:64] of the IP source address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3DAM0] are 1, this field contains the value to be matched with bits [95:64] of the IP destination address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] = 0, this field is not used. |

## 72.17.141 MAC Layer 3 Address 3 Reg 2 (MAC_Layer3_Addr3_Reg2)

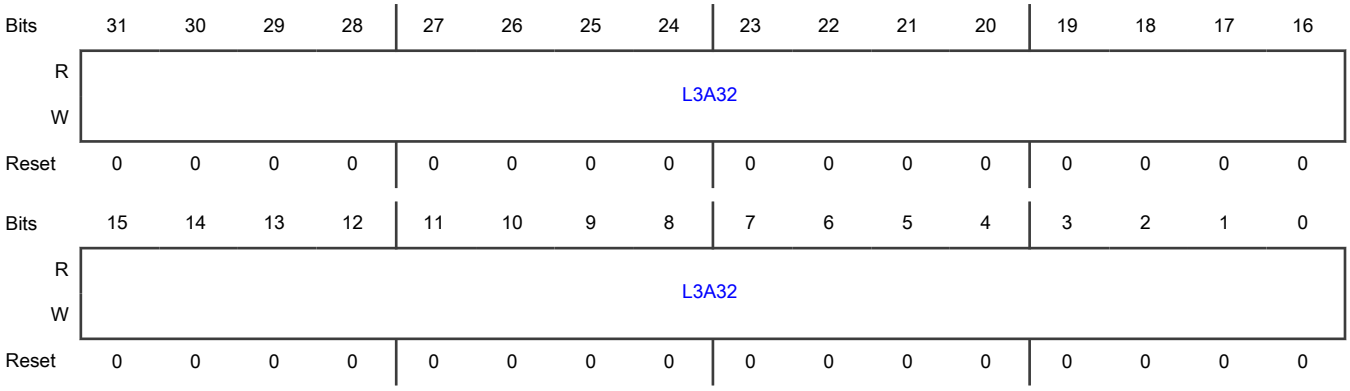
### Offset

| Register | Offset |
|---|---|
| MAC_Layer3_Addr3_Reg2 | 97Ch |

### Function

Contains the 32-bit IP destination address field for IPv4 packets. For IPv6 packets, the field contains bits [127:96] of the 128-bit IP source address or destination address field.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>L3A32 | Layer 3 Address 3<br>Indicates layer 3 address 3.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3SAM0] are 1, this field contains the value to be matched with bits [127:96] of the IP source address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3DAM0] are 1, this field contains the value to be matched with bits [127:96] of the IP destination address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] = 0, this field is not used. |

## 72.17.142 MAC L3 L4 Control 3 (MAC_L3_L4_Control3)

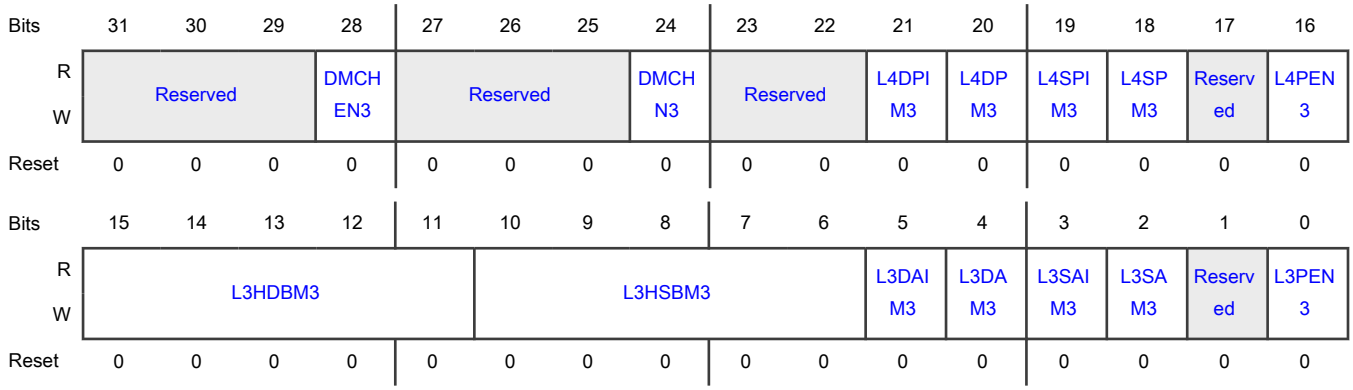
### Offset

| Register | Offset |
|---|---|
| MAC_L3_L4_Control3 | 990h |

### Function

Controls the filter 0 operations of the layer 3 and layer 4 protocols.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |
| 28<br>DMCHEN3 | DMA Channel Select Enable 3<br>Indicates the status of the DMA channel number.<br>• If this field is 1, it enables the selection of the DMA channel number for the packet that the L3_L4 filter passes. The DMCHN fields indicate the DMA channel number.<br>• If this field is 0, the filter does not decide the DMA channel number.<br>0b - Disabled<br>1b - Enabled |
| 27-25<br>— | Reserved |
| 24<br>DMCHN3 | DMA Channel Number 2<br>Indicates the DMA channel number.<br>If the value of the DMCHEN fields is 1, this field selects the DMA channel number to which the packet that this filter passes is routed. The width of this field depends on the number of the DMA channels present in your configuration. |
| 23-22<br>— | Reserved |
| 21<br>L4DPIM3 | Layer 4 Destination Port Inverse Match Enable 3<br>Indicates the status of layer 4 destination port inverse matching.<br>• If this field is 1, MAC_Layer4_Address0[L4DP0] is enabled for inverse matching.<br>• If this field is 0, MAC_Layer4_Address0[L4DP0] is enabled for perfect matching.<br>This field is valid and applicable only when the L4DPM0 field is 1. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |
| 20<br>L4DPM3 | Layer 4 Destination Port Match Enable 3<br>Indicates the status of layer 4 destination port matching.<br>• If this field is 1, MAC_Layer4_Address0[L4DP0] is enabled for matching.<br>• If this field is 0, MAC ignores MAC_Layer4_Address0[L4DP0] for matching.<br>0b - Disabled<br>1b - Enabled |
| 19<br>L4SPIM3 | Layer 4 Source Port Inverse Match Enable 3<br>Indicates the status of layer 4 source port inverse matching.<br>• If this field is 1, MAC_Layer4_Address0[L4SP0] is enabled for inverse matching.<br>• If this field is 0, MAC_Layer4_Address0[L4SP0] is enabled for perfect matching.<br>This field is valid and applicable only when MAC_Layer4_Address0[L4SP0] = 1.<br>0b - Disabled<br>1b - Enabled |
| 18<br>L4SPM3 | Layer 4 Source Port Match Enable 3<br>Indicates the status of layer 4 source port matching.<br>• If this field is 1, MAC_Layer4_Address0[L4SP0] is enabled for matching.<br>• If this field is 0, MAC ignores MAC_Layer4_Address0[L4SP0] for matching.<br>0b - Disabled<br>1b - Enabled |
| 17<br>— | Reserved |
| 16<br>L4PEN3 | Layer 4 Protocol Enable 3<br>Indicates the status of layer 4 protocol.<br>• If this field is 1, MAC_Layer4_Address0[L4SP0] and MAC_Layer4_Address0[L4DP0] of UDP packets are used for matching.<br>• If this field is 0, MAC_Layer4_Address0[L4SP0] and MAC_Layer4_Address0[L4DP0] of TCP packets are used for matching.<br>Layer 4 matching is performed only when the L4SPM0 field or the L4DPM0 field is 1.<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 15-11<br>L3HDBM3 | Layer 3 IP DA Higher Bits Match 3<br>IPv4 Packets:<br>This field contains the number of higher bits of IP DA that are matched in the IPv4 packets. The following list describes the values of this field:<br>• 0: No bits are masked.<br>• 1: LSb[0] is masked.<br>• 2: Two LSbs [1:0] are masked.<br>• ..<br>• 31: All bits except MSb are masked.<br>IPv6 Packets:<br>Bits [12:11] of this field correspond to bits [6:5] of MAC_L3_L4_Control0[L3HSBM0], which indicates the number of lower bits of IP SA or DA that are masked in the IPv6 packets. The following list describes the concatenated values of MAC_L3_L4_Control0[L3HDBM0][1:0] and MAC_L3_L4_Control0[L3HSBM0]:<br>• 0: No bits are masked.<br>• 1: LSb[0] is masked.<br>• 2: Two LSbs [1:0] are masked<br>• ..<br>• 127: All bits except MSb are masked.<br>This field is valid and applicable only when MAC_L3_L4_Control0[L3DAM0] or MAC_L3_L4_Control0[L3SAM0] is 1.<br>. |
| 10-6<br>L3HSBM3 | Layer 3 IP SA Higher Bits Match 3<br>IPv4 packets:<br>This field contains the number of lower bits of IP SA that are masked for matching in the IPv4 packets. The following list describes the values of this field:<br>• 0: No bits are masked.<br>• 1: LSb[0] is masked.<br>• 2: Two LSbs [1:0] are masked.<br>• ..<br>• 31: All bits except MSb are masked.<br>IPv6 packets:<br>This field contains bits [4:0] of MAC_L3_L4_Control0[L3HSBM0]. These bits indicate the number of higher bits of IP SA or DA matched in the IPv6 packets. The field is valid and applicable only when MAC_L3_L4_Control0[L3DAM0] or MAC_L3_L4_Control0[L3SAM0] is 1. |
| 5 | Layer 3 IP DA Inverse Match Enable 3 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| L3DAIM3 | Indicates the status of layer 3 IP DA inverse matching.<br>• If this field is 1, layer 3 IP DA is enabled for inverse matching.<br>• If this field is 0, layer 3 IP DA is enabled for perfect matching.<br>This field is valid and applicable only if MAC_L3_L4_Control0[L3DAM0] = 1.<br>0b - Disabled<br>1b - Enabled |
| 4<br>L3DAM3 | Layer 3 IP DA Match Enable 3<br>Indicates the status of layer 3 IP DA matching.<br>• If this field is 1, layer 3 IP DA is enabled for matching.<br>• If this field is 0, MAC ignores layer 3 IP DA for matching.<br>**NOTE**<br>If MAC_L3_L4_Control0[L3PEN0] = 1, you must write 1 to either this field or to MAC_L3_L4_Control0[L3SAM0] because either the IPv6 DA or SA can be checked for filtering.<br>0b - Disabled<br>1b - Enabled |
| 3<br>L3SAIM3 | Layer 3 IP SA Inverse Match Enable 3<br>Indicates the status of layer 3 IP SA inverse matching.<br>• If this field is 1, layer 3 IP SA is enabled for inverse matching.<br>• If this field is 0, layer 3 IP SA is enabled for perfect matching.<br>This field is valid and applicable only if MAC_L3_L4_Control0[L3SAM0] is 1.<br>0b - Disabled<br>1b - Enabled |
| 2<br>L3SAM3 | Layer 3 IP SA Match Enable 3<br>Indicates the status of layer 3 IP SA matching.<br>• If this field is 1, layer 3 IP SA matching is enabled.<br>• If this field is 0, MAC ignores layer 3 IP SA matching.<br>**NOTE**<br>If MAC_L3_L4_Control0[L3PEN0] = 1, you must write 1 to either this field or to MAC_L3_L4_Control0[L3DAM0] because either IPv6 SA or DA can be checked for filtering.<br>0b - Disabled<br>1b - Enabled |
| 1 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 0<br>L3PEN3 | Layer 3 Protocol Enable 3<br>Indicates the status of layer 3 protocol.<br>• If this field is 1, the layer 3 IP SA or DA matching is enabled for IPv6 packets.<br>• If this field is 0, the layer 3 IP SA or DA matching is enabled for IPv4 packets.<br>The layer 3 matching is performed only when either MAC_L3_L4_Control0[L3SAM0] or MAC_L3_L4_Control0[L3DAM0] is 1.<br>0b - Disabled<br>1b - Enabled |

## 72.17.143 MAC Layer 4 Address 3 (MAC_Layer4_Address3)

### Offset

| Register | Offset |
|---|---|
| MAC_Layer4_Address3 | 994h |

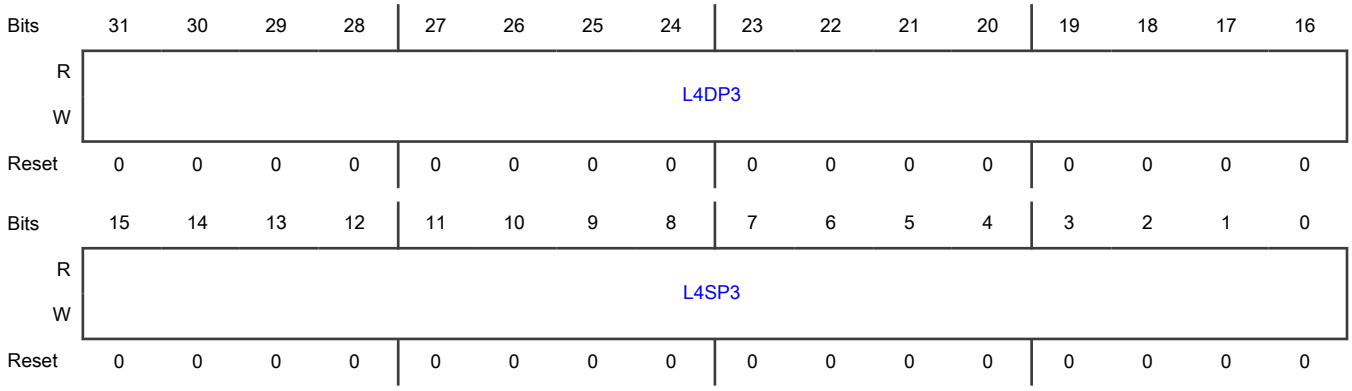
### Function

Provides the layer 4 source and destination port numbers.

MAC Layer 4 Address 1 (MAC_Layer4_Address1), MAC L3 L4 Control 1 (MAC_L3_L4_Control1), MAC Layer 3 Address 0 Reg 1 (MAC_Layer3_Addr0_Reg1), MAC Layer 3 Address 1 Reg 1 (MAC_Layer3_Addr1_Reg1), MAC Layer 3 Address 2 Reg 1 (MAC_Layer3_Addr2_Reg1), and MAC Layer 3 Address 3 Reg 1 (MAC_Layer3_Addr3_Reg1) are reserved registers (RO with a default value) if the enable layer 3 and layer 4 packet filter option is not selected when configuring the core.

You can configure the layer 3 and layer 4 address registers to be double-synchronized by selecting the synchronize layer 3 and layer 4 address registers to the receive clock domain option while configuring the core. When you select this option, the synchronization is triggered only when bits [31:24] (in Little-Endian mode) or bits [7:0] (in Big-Endian mode) of the layer 3 and layer 4 address registers are written to. For proper synchronization updates, you must perform consecutive writes to the same layer 3 and layer 4 address registers after a delay of at least four destination clock cycles.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-16<br>L4DP3 | Layer 4 Destination Port Number 3<br>Indicates layer 4 destination port number.<br>If MAC_L3_L4_Control0[L4PEN0] = 0 and MAC_L3_L4_Control0[L4DPM0] = 1, this field contains the value to be matched with the TCP destination port number field in the IPv4 or IPv6 packets.<br>If MAC_L3_L4_Control0[L4PEN0] and MAC_L3_L4_Control0[L4DPM0] are 1, this field contains the value to be matched with the UDP destination port number field in the IPv4 or IPv6 packets. |
| 15-0<br>L4SP3 | Layer 4 Source Port Number 3<br>Indicates layer 4 source port number.<br>If MAC_L3_L4_Control0[L4PEN0] = 0 and MAC_L3_L4_Control0[L4SPM0] = 1, this field contains the value to be matched with the TCP source port number field in the IPv4 or IPv6 packets.<br>If MAC_L3_L4_Control0[L4PEN0] and MAC_L3_L4_Control0[L4SPM0] are 1, this field contains the value to be matched with the UDP source port number field in the IPv4 or IPv6 packets. |

## 72.17.144 MAC Layer 3 Address 0 Reg 3 (MAC_Layer3_Addr0_Reg3)

### Offset

| Register | Offset |
|---|---|
| MAC_Layer3_Addr0_Reg3 | 9A0h |

### Function

Contains the 32-bit IP source address field for IPv4 packets. For IPv6 packets, the field contains bits [31:0] of the 128-bit IP source address or destination address field.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A03 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A03 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>L3A03 | Layer 3 Address 0<br>Indicates layer 3 address 0.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3SAM0] are 1, this field contains the value to be matched with bits [31:0] of the IP source address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3DAM0] are 1, this field contains the value to be matched with bits [31:0] of the IP destination address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] = 0, and MAC_L3_L4_Control0[L3SAM0] = 1, this field contains the value to be matched with the IP source address field in the IPv4 packets. |

## 72.17.145 MAC Layer 3 Address 1 Reg 3 (MAC_Layer3_Addr1_Reg3)

### Offset

| Register | Offset |
|---|---|
| MAC_Layer3_Addr1_Reg3 | 9A4h |

### Function

Contains the 32-bit IP destination address field for IPv4 packets. For IPv6 packets, the field contains bits [63:32] of the 128-bit IP source address or destination address field.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A13 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A13 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>L3A13 | Layer 3 Address 1<br>Indicates layer 3 address 1. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | • If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3SAM0] are 1, this field contains the value to be matched with bits [63:32] of the IP source address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3DAM0] are 1, this field contains the value to be matched with bits [63:32] of the IP destination address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] = 0, and MAC_L3_L4_Control0[L3SAM0] = 1, this field contains the value to be matched with the IP source address field in the IPv4 packets. |

## 72.17.146 MAC Layer 3 Address 2 Reg 3 (MAC_Layer3_Addr2_Reg3)

### Offset

| Register | Offset |
|---|---|
| MAC_Layer3_Addr2_Reg3 | 9A8h |

### Function

Contains the 32-bit IP destination address field for IPv4 packets. For IPv6 packets, the field contains bits [95:64] of the 128-bit IP source address or destination address field.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A23 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | L3A23 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>L3A23 | Layer 3 Address 2<br>Indicates layer 3 address 2.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3SAM0] are 1, this field contains the value to be matched with bits [95:64] of the IP source address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3DAM0] are 1, this field contains the value to be matched with bits [95:64] of the IP destination address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] = 0, this field is not used. |

## 72.17.147 MAC Layer 3 Address 3 Reg 3 (MAC_Layer3_Addr3_Reg3)

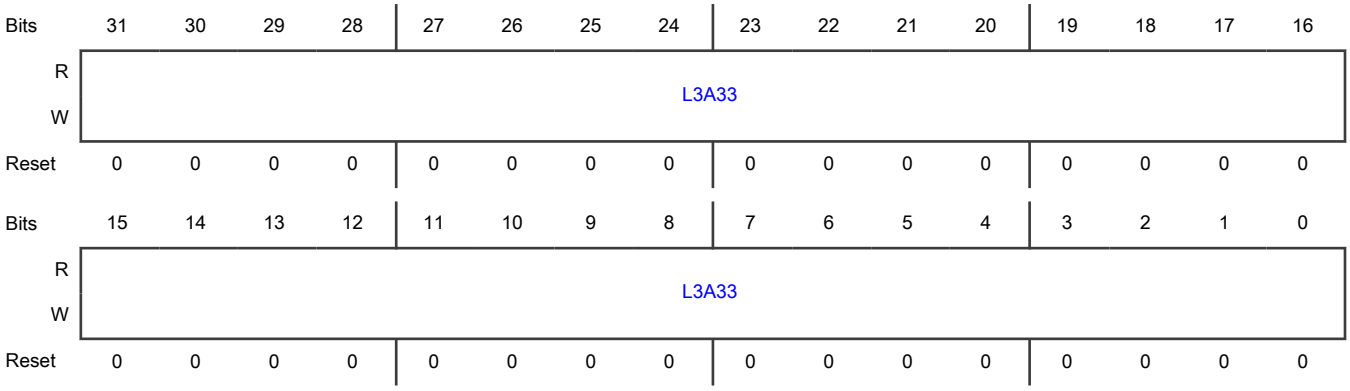
### Offset

| Register | Offset |
|---|---|
| MAC_Layer3_Addr3_Reg3 | 9ACh |

### Function

Contains the 32-bit IP destination address field for IPv4 packets. For IPv6 packets, the field contains bits [127:96] of the 128-bit IP source address or destination address field.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>L3A33 | Layer 3 Address 3<br>Indicates layer 3 address 3.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3SAM0] are 1, this field contains the value to be matched with bits [127:96] of the IP source address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] and MAC_L3_L4_Control0[L3DAM0] are 1, this field contains the value to be matched with bits [127:96] of the IP destination address field in the IPv6 packets.<br>• If MAC_L3_L4_Control0[L3PEN0] = 0, this field is not used. |

## 72.17.148 MAC Timestamp Control (MAC_Timestamp_Control)

### Offset

| Register | Offset |
|---|---|
| MAC_Timestamp_Control | B00h |

### Function

Controls the operation of the system time generator and processing of PTP packets for timestamping in the receiver.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | AV8021A... | Reserved | | | TXTSSTSM | Reserved | | | ESTI | Reserved | TSENMAC... | SNAPTYPSEL | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSMSTRE... | TSEVNTE... | TSIPV4E... | TSIPV6E... | TSIPENA | TSVER2E... | TSCTRLS... | TSENALL | Reserved | PTGE | TSADDREG | Reserved | TSUPDT | TSINIT | TSCFUPDT | TSENA |
| Reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |
| 28<br>AV8021ASMEN | AV 802.1AS Mode Enable<br>Indicates the status of AV 802.1AS mode.<br>If this field is 1, MAC processes only untagged PTP over Ethernet packets for providing PTP status and capturing timestamp snapshots, which means using the IEEE 802.1AS mode of operation.<br>If MAC_HW_Feature1[PTOEN] = 1, for the purpose of PTP offload, the transport-specific field in the PTP header is generated and checked based on the value of the AV8021ASMEN field.<br>0b - Disabled<br>1b - Enabled |
| 27-25<br>— | Reserved |
| 24<br>TXTSSTSM | Transmit Timestamp Status Mode<br>Indicates the status of transmit timestamp status mode.<br>• If this field is 1, MAC overwrites the previous transmit timestamp status even if the software does not read it. MAC indicates this by writing 1 to MAC_Tx_Timestamp_Status_Nanoseconds[TXTSSMIS].<br>• If this field is 0, MAC ignores the timestamp status of the current packet if the software does not read the timestamp status of the previous packet. MAC indicates this by writing 1 to MAC_Tx_Timestamp_Status_Nanoseconds[TXTSSMIS].<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 23-21<br>— | Reserved |
| 20<br>ESTI | External System Time Input<br>Indicates the status of external system time input.<br>If this field is 1, MAC uses the external 64-bit reference system time input for these functions:<br>• To consider the timestamp provided as status<br>• To insert the timestamp in transmit PTP packets if the one-step timestamp or the timestamp offload feature is enabled<br>If this field is 1, MAC uses the internal reference system time.<br>0b - Disabled<br>1b - Enabled |
| 19<br>— | Reserved |
| 18<br>TSENMACADDR | Enable MAC Address For PTP Packet Filtering<br>Indicates whether the MAC address for PTP packet filtering is enabled.<br>If this field is 1, the DA MAC address, which matches any MAC Address register, is used to filter the PTP packets when PTP is directly sent over Ethernet.<br>If this field is 1, received PTP packets with DA containing a special multicast or unicast address that matches the one programmed in the MAC address registers are considered for processing when PTP is directly sent over Ethernet.<br>• For a normal time stamping operation, MAC address registers 0 to 31 are considered for unicast destination address matching.<br>• For PTP offload, only MAC address register 0 is considered for unicast destination address matching.<br>0b - Disabled<br>1b - Enabled |
| 17-16<br>SNAPTYPSEL | Select PTP Packets For Taking Snapshots<br>Determines the set of PTP packet types for which a snapshot needs to be taken.<br>This field, along with the TSEVNTENA and TSMSTRENA fields, determines the set of PTP packet types for which a snapshot needs to be taken. See Receive path functions for encoding. |
| 15<br>TSMSTRENA | Enable Snapshot For Messages Relevant To Master<br>Indicates whether the snapshot for messages relevant to master is enabled.<br>If this field is 1, the snapshot is taken only for the messages that are relevant to the master node. Otherwise, the snapshot is taken for the messages relevant to the slave node. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |
| 14<br>TSEVNTENA | Enable Timestamp Snapshot For Event Messages<br>Enables or disables timestamp snapshot for event messages.<br>If this field is 1, the timestamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When the field becomes 0, the snapshot is taken for all the messages except Announce, Management, and Signaling. For more information on timestamp snapshot dependency on register bits, see Receive path functions.<br>0b - Disabled<br>1b - Enabled |
| 13<br>TSIPV4ENA | Enable Processing Of PTP Packets Sent Over IPv4-UDP<br>Indicates whether the processing of PTP packets sent over IPv4-UDP is enabled.<br>If this field is 1, the MAC receiver processes the PTP packets encapsulated in the IPv4-UDP packets. When this field becomes 0, MAC ignores the PTP transported over the IPv4-UDP packets. The default value of this field is 1.<br>0b - Disabled<br>1b - Enabled |
| 12<br>TSIPV6ENA | Enable Processing Of PTP Packets Sent Over IPv6-UDP<br>Indicates whether the processing of PTP packets sent over IPv6-UDP is enabled.<br>If this field is 1, the MAC receiver processes the PTP packets encapsulated in the IPv6-UDP packets. If the value is 0, MAC ignores the PTP transported over IPv6-UDP packets.<br>0b - Disabled<br>1b - Enabled |
| 11<br>TSIPENA | Enable Processing Of PTP Over Ethernet Packets<br>Indicates the status of PTP processing over Ethernet packets.<br>If this field is 1, the MAC receiver processes the PTP packets encapsulated directly in the Ethernet packets. When the field becomes 0, MAC ignores the PTP over Ethernet packets.<br>0b - Disabled<br>1b - Enabled |
| 10<br>TSVER2ENA | Enable PTP Packet Processing For Version 2 Format<br>Indicates the status of PTP packet processing for version 2 format.<br>If this field is 1, the IEEE 1588 version 2 format is used to process the PTP packets. When this field becomes 0, the IEEE 1588 version 1 format is used to process the PTP packets. See PTP processing and control for more information on the IEEE 1588 formats. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |
| 9<br>TSCTRLSSR | Timestamp Digital Or Binary Rollover Control<br>Indicates the status of timestamp digital or binary rollover control.<br>• If this field is 1, the Timestamp Low register rolls over after the 3B9A_C9FFh value (that is, 1 nanosecond accuracy) and increments the timestamp (high) seconds.<br>• If this field is 0, the rollover value of the Subsecond register is 7FFF_FFFFh. The sub-second increment must be programmed correctly depending on the PTP reference clock frequency and the value of this field.<br>0b - Disabled<br>1b - Enabled |
| 8<br>TSENALL | Enable Timestamp For All Packets<br>Indicates the status of timestamp snapshot for all packets.<br>If this field is 1, the timestamp snapshot is enabled for all packets that MAC receives.<br>0b - Disabled<br>1b - Enabled |
| 7<br>— | Reserved |
| 6<br>PTGE | Presentation Time Generation Enable<br>Indicates the status of presentation time generation.<br>If this field is 1, the presentation time generation is enabled.<br>0b - Disabled<br>1b - Enabled |
| 5<br>TSADDREG | Update Addend Register<br>Indicates whether MAC Timestamp Addend (MAC_Timestamp_Addend) is updated.<br>If this field is 1, the contents of MAC Timestamp Addend (MAC_Timestamp_Addend) are updated in the PTP block for fine correction. The field becomes 0 after the update completes. The value of the field must also be 0 before you write 1 to it.<br>Access restrictions apply to this field that clears automatically. Writing 0 has no effect.<br>0b - Not updated<br>1b - Updated |
| 4<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 3<br>TSUPDT | Update Timestamp<br>Indicates whether the timestamp is updated.<br>If this field is 1, the system time is updated (added or subtracted) with the value specified in MAC System Time Seconds Update (MAC_System_Time_Seconds_Update) and MAC System Time Nanoseconds Update (MAC_System_Time_Nanoseconds_Update).<br>The value of this field must be 0 before you update it. It resets after the update is complete in hardware. MAC_System_Time_Higher_Word_Seconds[TSHWR], if enabled during core configuration, is not updated.<br>When media clock generation and recovery is configured (DWC_EQOS_FLEXI_PPS_OUT_EN) and enabled, MAC Presentation Time Update (MAC_Presn_Time_Updt) must also be updated before you write 1 to this field.<br>Access restrictions apply to this field that clears automatically. Writing 0 has no effect.<br>0b - Not updated<br>1b - Updated |
| 2<br>TSINIT | Initialize Timestamp<br>Indicates whether the timestamp is initialized.<br>If this field is 1, the system time is initialized (overwritten) with the value specified in MAC System Time Seconds Update (MAC_System_Time_Seconds_Update) and MAC System Time Nanoseconds Update (MAC_System_Time_Nanoseconds_Update).<br>The value of this field must be 0 before you update it, and the field is reset after the initialization is complete. MAC_System_Time_Higher_Word_Seconds[TSHWR], if enabled during core configuration, can only be initialized.<br>When media clock generation and recovery is configured (DWC_EQOS_FLEXI_PPS_OUT_EN) and enabled, MAC Presentation Time Update (MAC_Presn_Time_Updt) must also be updated before you write 1 to this field.<br>Access restrictions apply to this field that clears automatically. Writing 0 has no effect.<br>0b - Not initialized<br>1b - Initialized |
| 1<br>TSCFUPDT | Fine Or Coarse Timestamp Update<br>Indicates the method used to update system timestamp.<br>• If this field is 1, the fine method is used to update system timestamp.<br>• If this field is 0, the coarse method is used to update the system timestamp.<br>0b - Coarse method<br>1b - Fine method |
| 0<br>TSENA | Timestamp Enable<br>Enables or disables timestamp for transmit and receive packets. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • If this field is 1, the timestamp is added for transmit and receive packets.<br>• If this field is 0, timestamp is not added for transmit and receive packets and the timestamp generator is also suspended. You need to initialize the timestamp (system time) after enabling this mode.<br>On the receive side, MAC processes the 1588 packets only if this field is 1.<br>0b - Disabled<br>1b - Enabled |

## 72.17.149 MAC Sub Second Increment (MAC_Sub_Second_Increment)

### Offset

| Register | Offset |
|---|---|
| MAC_Sub_Second_Increment | B04h |

### Function

Specifies the value to be added to the internal system time register at every cycle of the CLK_PTP_REF_I clock.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23-16<br>SSINC | Sub-Second Increment Value |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Contains the sub-second increment value, which accumulates every clock cycle (of clk_ptp_i) with the contents of the Subsecond register. For example, if the PTP clock is 50 MHz (period is 20 ns), you must program 20 (14h) when MAC System Time In Nanoseconds (MAC_System_Time_Nanoseconds) has an accuracy of 1 ns (MAC_Timestamp_Control[TSCTRLSSR] = 1). When TSCTRLSSR = 0, the Nanoseconds register has a resolution of ~0.465 ns. In this case, you must program a value of 43 (2Bh), which is derived by 20 ns/0.465. |
| 15-8<br>SNSINC | Sub-Nanosecond Increment Value<br>Contains the sub-nanosecond increment value, represented in nanoseconds multiplied by 2^8. This value is accumulated in the sub-nanoseconds field of the Subsecond register. For example, if MAC_Timestamp_Control[TSCTRLSSR] = 1, and if the required increment is 5.3 ns, then this field must be 4Ch and the SSINC field must be 05h. |
| 7-0<br>— | Reserved |

## 72.17.150 MAC System Time In Seconds (MAC_System_Time_Seconds)

### Offset

| Register | Offset |
|---|---|
| MAC_System_Time_Seconds | B08h |

### Function

Indicates, along with MAC System Time In Nanoseconds (MAC_System_Time_Nanoseconds), the current value of the system time that MAC maintains. Although it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies (from CLK_PTP_REF_I to the CSR clock).

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>TSS | Timestamp Second<br>Indicates the current value, in seconds, of the system time that MAC maintains. |

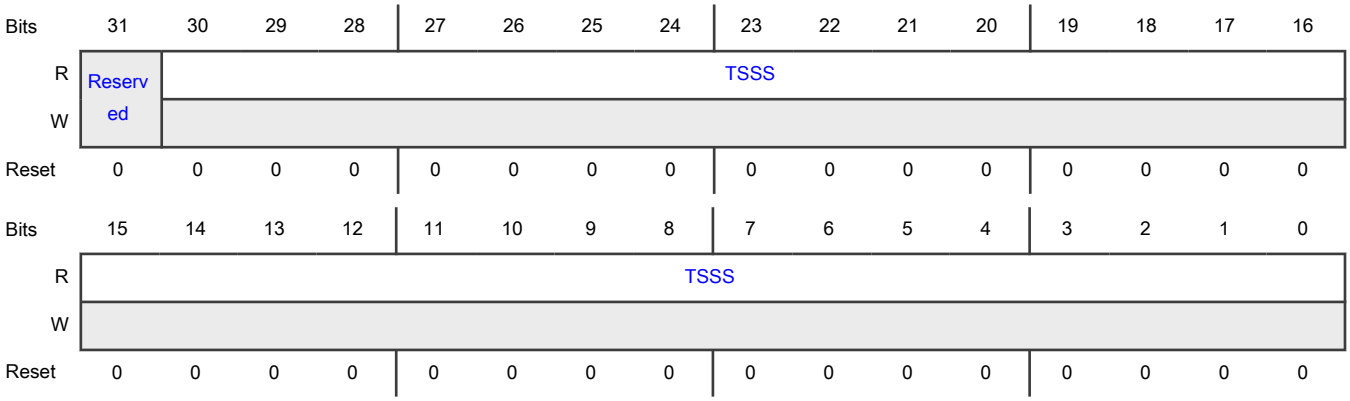
## 72.17.151 MAC System Time In Nanoseconds (MAC_System_Time_Nanoseconds)

### Offset

| Register | Offset |
|---|---|
| MAC_System_Time_Nanoseconds | B0Ch |

### Function
Indicates, along with MAC System Time In Seconds (MAC_System_Time_Seconds), the current value of the system time that MAC maintains.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-0<br>TSSS | Timestamp Sub Seconds<br>Indicates the sub-second representation of time, with an accuracy of 0.46 ns. When MAC_Timestamp_Control[TSCTRLSSR] = 1, each bit represents 1 ns. The maximum value is 3B9A_C9FFh after which it rolls over to 0. |

## 72.17.152 MAC System Time Seconds Update (MAC_System_Time_Seconds_Update)

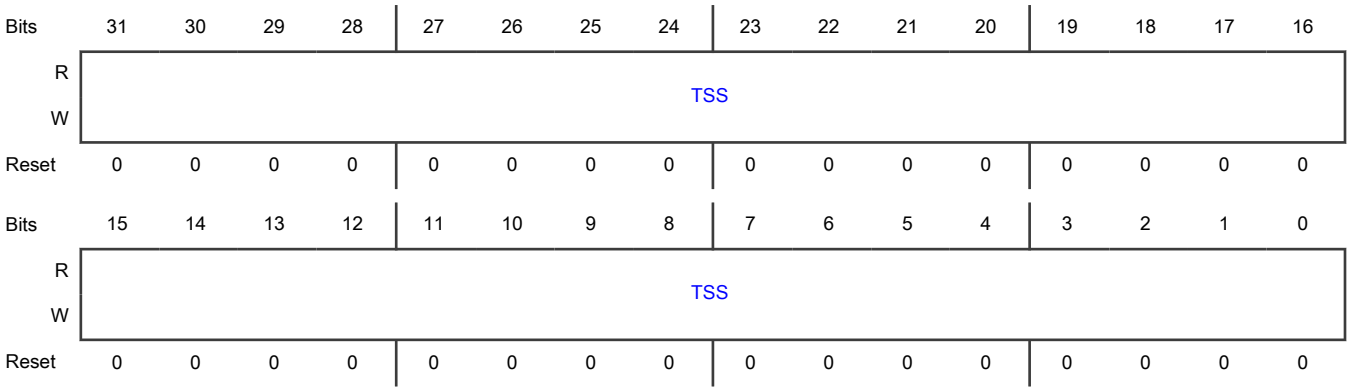
### Offset

| Register | Offset |
|---|---|
| MAC_System_Time_Seconds_Update | B10h |

### Function

Initializes or updates, along with MAC System Time Nanoseconds Update (MAC_System_Time_Nanoseconds_Update), the system time that MAC maintains. You must write both to this register and the MAC System Time Nanoseconds Update register before writing 1 to MAC_Timestamp_Control[TSINIT] and MAC_Timestamp_Control[TSUPDT].

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>TSS | Timestamp Seconds<br>Indicates the timestamp seconds value.<br>The value in this field is the seconds part of the update.<br>• If MAC_System_Time_Nanoseconds_Update[ADDSUB] = 0, this field must be programmed with the seconds part of the update value.<br>• If MAC_System_Time_Nanoseconds_Update[ADDSUB] = 1, this field must be programmed with the complement of the seconds part of the update value.<br>For example, if 2.000000001 seconds need to be subtracted from the system time, MAC_System_Time_Seconds[TSS] must be FFFF_FFFEh (that is, 2^32 - 2). |

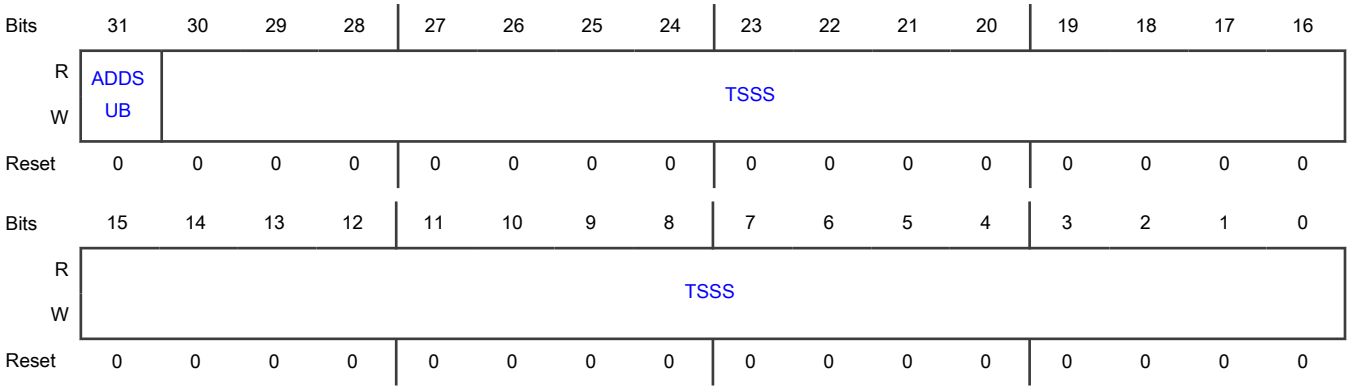
## 72.17.153 MAC System Time Nanoseconds Update (MAC_System_Time_Nanoseconds_Update)

### Offset

| Register | Offset |
|---|---|
| MAC_System_Time_Nanoseconds_Update | B14h |

### Function
Indicates system time nanoseconds update.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>ADDSUB | Add Or Subtract Time<br>Indicates whether the time value is added or subtracted from the contents of the update register.<br>• If this field is 1, the time value is subtracted from the contents of the update register.<br>• If this field is 0, the time value is added to the contents of the update register.<br>0b - Add time<br>1b - Subtract time |
| 30-0<br>TSSS | Timestamp Subseconds<br>Indicates the sub-second part of the update.<br>• If ADDSUB = 0, this field must be programmed with the sub-second part of the update value, with an accuracy based on MAC_Timestamp_Control[TSCTRLSSR].<br>• If ADDSUB = 0, this field must be programmed with the complement of the sub-second part of the update value as described below.<br>• If MAC_Timestamp_Control[TSCTRLSSR] = 1, the programmed value must be 10^9 - <sub-second value>, and if MAC_Timestamp_Control[TSCTRLSSR] = 0, the programmed value must be 2^31 - <sub-second_value>. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • If MAC_Timestamp_Control[TSCTRLSSR] = 0, each bit of it represents an accuracy of 0.46 ns, and if MAC_Timestamp_Control[TSCTRLSSR] = 1, each bit represents 1 ns and the programmed value must not exceed 3B9A_C9FFh.<br>For example, if 2.000000001 seconds need to be subtracted from the system time, this field must be 7FFF_FFFFh (that is, 2^31 - 1) if MAC_Timestamp_Control[TSCTRLSSR] = 0. This field must be 3B9A_C9FFh (that is, 10^9 - 1) if MAC_Timestamp_Control[TSCTRLSSR] = 1. |

## 72.17.154 MAC Timestamp Addend (MAC_Timestamp_Addend)

### Offset

| Register | Offset |
|---|---|
| MAC_Timestamp_Addend | B18h |

### Function

Is used only when the system time is configured for Fine Update mode (MAC_Timestamp_Control[TSINIT] = 1). The contents of this register are added to a 32-bit accumulator in every clock cycle (of CLK_PTP_REF_I), and the system time is updated whenever the accumulator overflows.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSAR | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSAR | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>TSAR | Timestamp Addend Register<br>Indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization. |

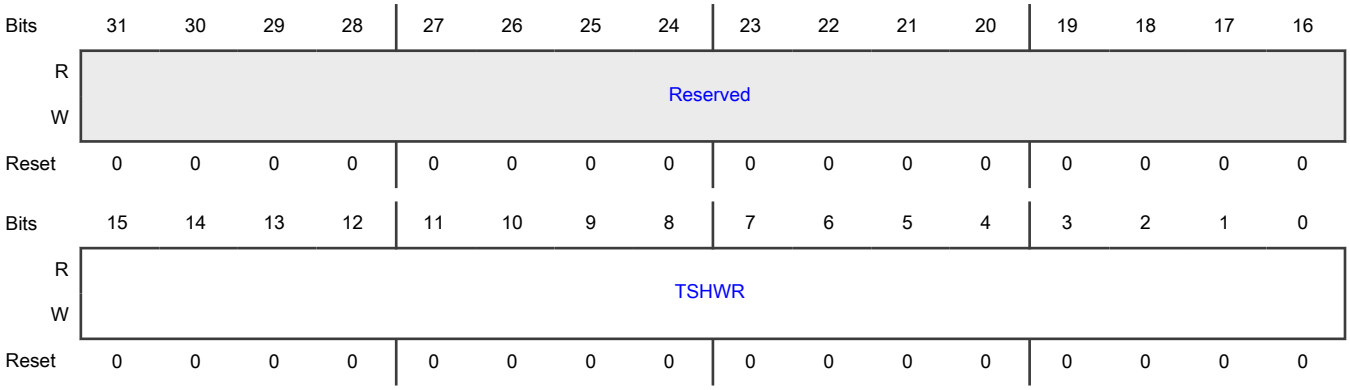
## 72.17.155 MAC System Time Higher Word In Seconds (MAC_System_Time_Higher_Word_Seconds)

### Offset

| Register | Offset |
|---|---|
| MAC_System_Time_Higher_Word_Seconds | B1Ch |

### Function
Indicates system time - higher word in seconds.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>TSHWR | Timestamp Higher Word Register<br>Contains the most-significant 16 bits of the timestamp seconds value. This register is optional, and you can add it if you select the IEEE 1588 higher-word register option. You write to this register directly to initialize the value, which increments when there is an overflow from the 32 bits of MAC System Time In Seconds (MAC_System_Time_Seconds).<br>Access restrictions apply to this field, which updates based on an event occurrence. |

## 72.17.156 MAC Timestamp Status (MAC_Timestamp_Status)

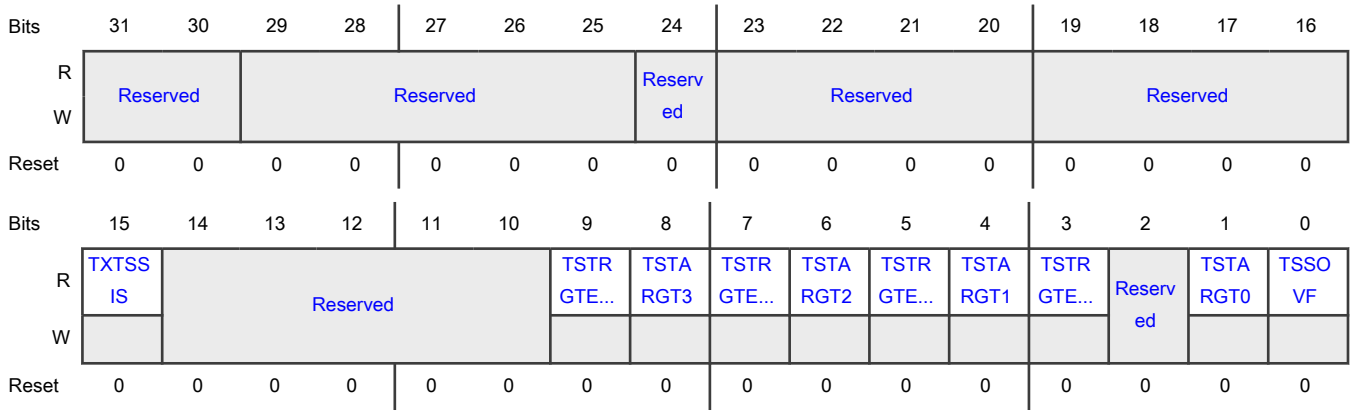
### Offset

| Register | Offset |
|---|---|
| MAC_Timestamp_Status | B20h |

### Function

Indicates timestamp status. All the bits except [27:25] are cleared when the application reads this register.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29-25<br>— | Reserved |
| 24<br>— | Reserved |
| 23-20<br>— | Reserved |
| 19-16<br>— | Reserved |
| 15<br>TXTSSIS | Transmit Timestamp Status Interrupt Status<br>Indicates whether the transmit timestamp interrupt status is detected.<br>In non-EQOS_CORE configurations, if the drop transmit status is enabled in MTL, this field becomes 1 when the captured transmit timestamp is updated in MAC Transmit Timestamp Status In Seconds (MAC_Tx_Timestamp_Status_Seconds) and MAC Transmit Timestamp Status In Nanoseconds (MAC_Tx_Timestamp_Status_Nanoseconds).<br>If MAC_HW_Feature1[PTOEN] = 1, TXTSSIS becomes 1 when the captured transmit timestamp is updated in MAC Transmit Timestamp Status In Seconds (MAC_Tx_Timestamp_Status_Seconds) and MAC Transmit Timestamp Status In Nanoseconds (MAC_Tx_Timestamp_Status_Nanoseconds) for PTO-generated delay request and Pdelay request packets. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | This field becomes 0 if you read or write to MAC Transmit Timestamp Status In Seconds (MAC_Tx_Timestamp_Status_Seconds) when MAC_CSR_SW_Ctrl[RCWE] = 1.<br>0b - Not detected<br>1b - Detected |
| 14-10<br>— | Reserved |
| 9<br>TSTRGTERR3 | Timestamp Target Time Error<br>Indicates whether the timestamp target time error status is detected.<br>This field becomes 1 if the latest target time programmed in MAC PPS3 Target Time In Seconds (MAC_PPS3_Target_Time_Seconds) and MAC PPS3 Target Time In Nanoseconds (MAC_PPS3_Target_Time_Nanoseconds) elapses. The field becomes 0 when the application reads it.<br>Access restrictions apply to this field that clears when read (or when this field is written to 1 if MAC_CSR_SW_Ctrl[RCWE] = 1). Also, the field automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 8<br>TSTARGT3 | Timestamp Target Time Reached For Target Time PPS3<br>Indicates whether the timestamp target time reached for target time PPS3 status is detected.<br>If this field is 1, it indicates that the value of the system time is greater than or equal to the value specified in MAC PPS3 Target Time In Seconds (MAC_PPS3_Target_Time_Seconds) and MAC PPS3 Target Time In Nanoseconds (MAC_PPS3_Target_Time_Nanoseconds).<br>Access restrictions apply to this field that clears when read (or when this field is written to 1 if MAC_CSR_SW_Ctrl[RCWE] = 1). Also, the field automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 7<br>TSTRGTERR2 | Timestamp Target Time Error<br>Indicates whether the timestamp target time error status is detected.<br>This field becomes 1 when the latest target time programmed in MAC PPS2 Target Time In Seconds (MAC_PPS2_Target_Time_Seconds) and MAC PPS2 Target Time In Nanoseconds (MAC_PPS2_Target_Time_Nanoseconds) elapses. The field becomes 0 when the application reads it.<br>Access restrictions apply to this field that clears when read (or when this field is written to 1 if MAC_CSR_SW_Ctrl[RCWE] = 1). Also, the field automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 6<br>TSTARGT2 | Timestamp Target Time Reached For Target Time PPS2<br>Indicates whether the timestamp target time reached for the target time PPS2 status is detected. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | If this field is 1, it indicates that the value of system time is greater than or equal to the value specified in MAC PPS2 Target Time In Seconds (MAC_PPS2_Target_Time_Seconds) and MAC PPS2 Target Time In Nanoseconds (MAC_PPS2_Target_Time_Nanoseconds).<br>Access restrictions apply to this field that clears when read (or when this field is written to 1 if MAC_CSR_SW_Ctrl[RCWE] = 1). Also, the field automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 5<br>TSTRGTERR1 | Timestamp Target Time Error<br>Indicates whether the timestamp target time error status is detected.<br>This field becomes 1 when the latest target time programmed in MAC PPS1 Target Time In Seconds (MAC_PPS1_Target_Time_Seconds) and MAC PPS1 Target Time In Nanoseconds (MAC_PPS1_Target_Time_Nanoseconds) elapses. The field becomes 0 when the application reads it.<br>Access restrictions apply to this field that clears when read (or when this field is written to 1 if MAC_CSR_SW_Ctrl[RCWE] = 1). Also, the field automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 4<br>TSTARGT1 | Timestamp Target Time Reached For Target Time PPS1<br>Indicates whether the timestamp target time reached for target time PPS1 status is detected.<br>If this field is 1, it indicates that the value of system time is greater than or equal to the value specified in MAC PPS1 Target Time In Seconds (MAC_PPS1_Target_Time_Seconds) and MAC PPS1 Target Time In Nanoseconds (MAC_PPS1_Target_Time_Nanoseconds)<br>Access restrictions apply to this field that clears when read (or when this field is written to 1 if MAC_CSR_SW_Ctrl[RCWE] = 1). Also, the field automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 3<br>TSTRGTERR0 | Timestamp Target Time Error<br>Indicates whether the timestamp target time error status is detected.<br>This field becomes 1 when the latest target time programmed in MAC PPS0 Target Time In Seconds (MAC_PPS0_Target_Time_Seconds) and MAC PPS0 Target Time In Nanoseconds (MAC_PPS0_Target_Time_Nanoseconds) elapses. The field becomes 0 when the application reads it.<br>Access restrictions apply to this field that clears when read (or when this field is written to 1 if MAC_CSR_SW_Ctrl[RCWE] = 1). Also, the field automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 2<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 1<br>TSTARGT0 | Timestamp Target Time Reached<br>Indicates whether the timestamp target time reached status is detected.<br>If this field is 1, it indicates that the value of the system time is greater than or equal to the value specified in MAC PPS0 Target Time In Seconds (MAC_PPS0_Target_Time_Seconds) and MAC PPS0 Target Time In Nanoseconds (MAC_PPS0_Target_Time_Nanoseconds)<br>Access restrictions apply to this field that clears when read (or when this field is written to 1 if MAC_CSR_SW_Ctrl[RCWE] = 1). Also, the field automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 0<br>TSSOVF | Timestamp Seconds Overflow<br>Indicates whether the timestamp seconds overflow status is detected.<br>If this field is 1, it indicates that the seconds value of the timestamp (when supporting version 2 format) has overflowed beyond 32'hFFFF_FFFF.<br>Access restrictions apply to this field that clears when read (or when this field is written to 1 if MAC_CSR_SW_Ctrl[RCWE] = 1). Also, the field automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |

## 72.17.157 MAC Transmit Timestamp Status In Nanoseconds (MAC_Tx_Timestamp_Status_Nanoseconds)

### Offset

| Register | Offset |
|---|---|
| MAC_Tx_Timestamp_Status_Nanoseconds | B30h |

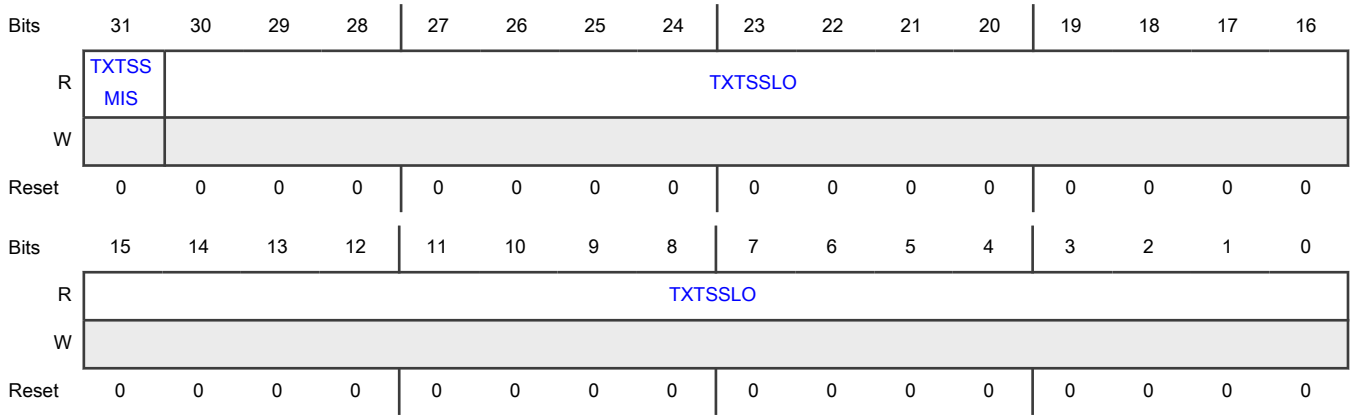
### Function

Contains the nanosecond part of the timestamp captured for tansmit packets when the transmit status is disabled.

This register, along with MAC Transmit Timestamp Status In Seconds (MAC_Tx_Timestamp_Status_Seconds), gives the 64-bit timestamp captured for the PTP packet that MAC successfully transmits. The application considers reading this value after the last byte of this register is read. In Little-Endian mode, this means when bits [31:24] are read, and in Big-Endian mode, this means when bits [7:0] are read.

If the application does not read these registers and the timestamp of another packet is captured, then either the current timestamp is lost (overwritten) or the new timestamp is lost (dropped), depending on the settings specified in MAC_Timestamp_Control[TXTSSTSM]. The status bit TXTSC bit [15] in the MAC_Timestamp_Status register is set whenever the MAC transmitter captures the timestamp.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>TXTSSMIS | Transmit Timestamp Status Missed<br>Indicates whether the transmit timestamp missed status is detected.<br>If this field is 1, it indicates one of these conditions:<br>• The timestamp of the current packet is ignored if MAC_Timestamp_Control[TXTSSTSM] = 0.<br>• The timestamp of the previous packet is overwritten with the timestamp of the current packet if MAC_Timestamp_Control[TXTSSTSM] = 1.<br>Access restrictions apply to this field that cleas on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 30-0<br>TXTSSLO | Transmit Timestamp Status Low<br>Contains 31 bits of the Nanoseconds field of the transmit packet's captured timestamp. |

## 72.17.158 MAC Transmit Timestamp Status In Seconds (MAC_Tx_Timestamp_Status_Seconds)

### Offset

| Register | Offset |
|---|---|
| MAC_Tx_Timestamp_Status_Seconds | B34h |

### Function

Contains the higher 32 bits of the timestamp (in seconds) captured when a PTP packet is transmitted.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>TXTSSHI | Transmit Timestamp Status High<br>Contains the lower 32 bits of the Seconds field of the transmitted packet's captured timestamp. |

## 72.17.159 MAC Timestamp Ingress Asymmetry Correction (MAC_Timestamp_Ingress_Asym_Corr)

### Offset

| Register | Offset |
|---|---|
| MAC_Timestamp_Ingress_Asym_Corr | B50h |

### Function

Contains the ingress asymmetry correction value to be used when updating the correction field in the PDelay_Resp PTP messages.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>OSTIAC | One-Step Timestamp Ingress Asymmetry Correction<br>Contains the ingress path asymmetry value to be added to the correctionField of the Pdelay_Resp PTP packet.<br>The programmed value must be expressed in units of nanoseconds and multiplied by 2^16. For example, 2.5 ns is represented as 00028000h. This value can also be negative, which is represented in the two's-complement form with bit 31 representing the sign bit. |

## 72.17.160 MAC Timestamp Egress Asymmetry Correction (MAC_Timestamp_Egress_Asym_Corr)

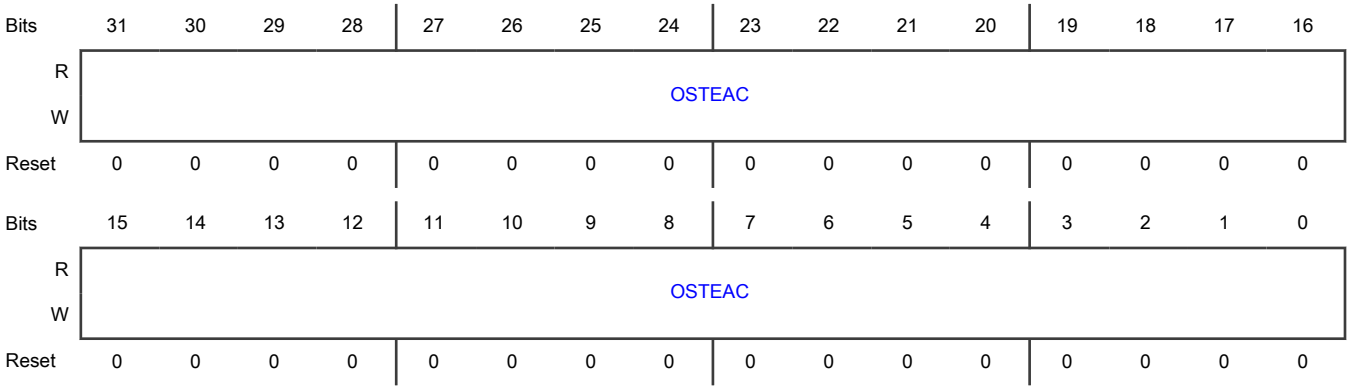
### Offset

| Register | Offset |
|---|---|
| MAC_Timestamp_Egress_Asym_Corr | B54h |

### Function

Contains the egress asymmetry correction value to be used when updating the correction field in the PDelay_Req PTP messages.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>OSTEAC | One-Step Timestamp Egress Asymmetry Correction<br>Contains the egress path asymmetry value to be subtracted from the correctionField of the Pdelay_Resp PTP packet. The programmed value must be the negated value in units of nanoseconds multiplied by 2^16.<br>For example, if the required correction is +2.5 ns, the programmed value must be FFFD_8000h, which is the two's-complement of 0002_8000h (2.5 * 216). Similarly, if the required correction is -3.3 ns, the programmed value is 0003_4CCCh (3.3 * 216). |

## 72.17.161 MAC Timestamp Ingress Correction In Nanoseconds (MAC_Timestamp_Ingress_Corr_Nanosecond)

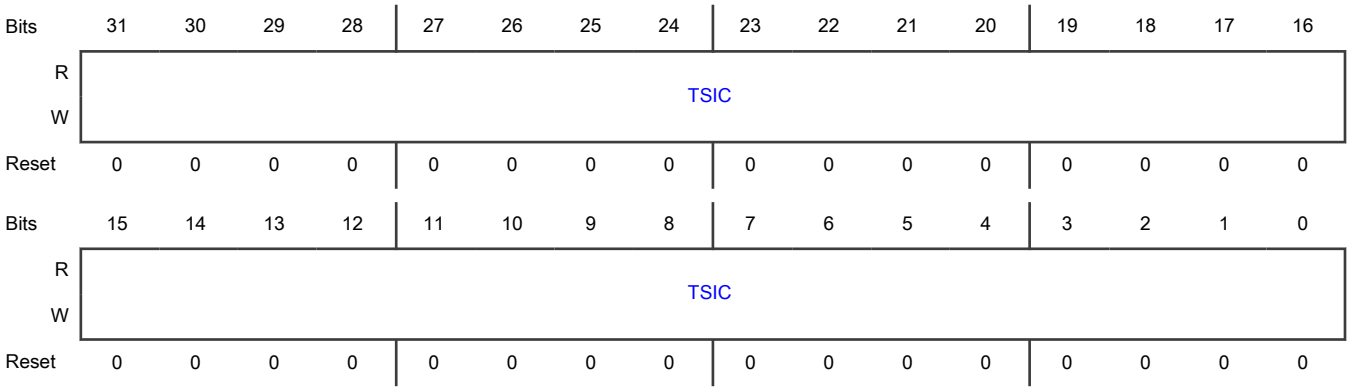
### Offset

| Register | Offset |
|---|---|
| MAC_Timestamp_Ingress_Corr_Nanosecond | B58h |

### Function

Contains the correction value, in nanoseconds, to be used with the captured timestamp value in the ingress path.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>TSIC | Timestamp Ingress Correction<br>Contains the ingress path correction value as defined by the ingress correction expression. |

## 72.17.162 MAC Timestamp Egress Correction In Nanoseconds (MAC_Timestamp_Egress_Corr_Nanosecond)

### Offset

| Register | Offset |
|---|---|
| MAC_Timestamp_Egress_Corr_Nanosecond | B5Ch |

### Function

Contains the correction value, in nanoseconds, to be used with the captured timestamp value in the egress path.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSEC | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSEC | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>TSEC | Timestamp Egress Correction<br>Contains the nanoseconds part of the egress path correction value as defined by the egress correction expression. |

## 72.17.163 MAC Timestamp Ingress Correction In Subnanoseconds (MAC_Timestamp_Ingress_Corr_Subnanosec)

### Offset

| Register | Offset |
|---|---|
| MAC_Timestamp_Ingress_Corr_Subnanosec | B60h |

### Function

Contains the sub-nanosecond part of the correction value to be used with the captured timestamp value for ingress direction.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSICSNS | | | | | | | | Reserved | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>TSICSNS | Timestamp Ingress Correction In Sub-Nanoseconds<br>Contains the sub-nanoseconds part of the ingress path correction value as defined by the Ingress Correction expression. |
| 7-0<br>— | Reserved |

## 72.17.164 MAC Timestamp Engress Correction In Subnanoseconds (MAC_Timestamp_Egress_Corr_Subnanosec)

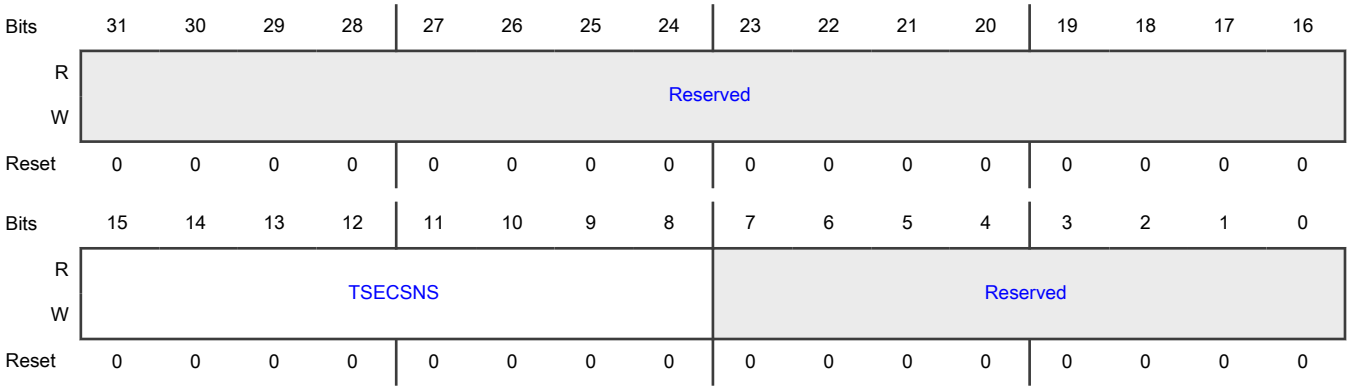
### Offset

| Register | Offset |
|---|---|
| MAC_Timestamp_Egress_Corr_Subnanosec | B64h |

### Function

Contains the sub-nanosecond part of the correction value to be used with the captured timestamp value for the egress direction.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 15-8<br>TSECSNS | Timestamp Egress Correction In Sub-Nanoseconds<br>Contains the sub-nanoseconds part of the egress path correction value, as defined by the Egress Correction expression. |
| 7-0<br>— | Reserved |

## 72.17.165 MAC Timestamp Ingress Latency (MAC_Timestamp_Ingress_Latency)

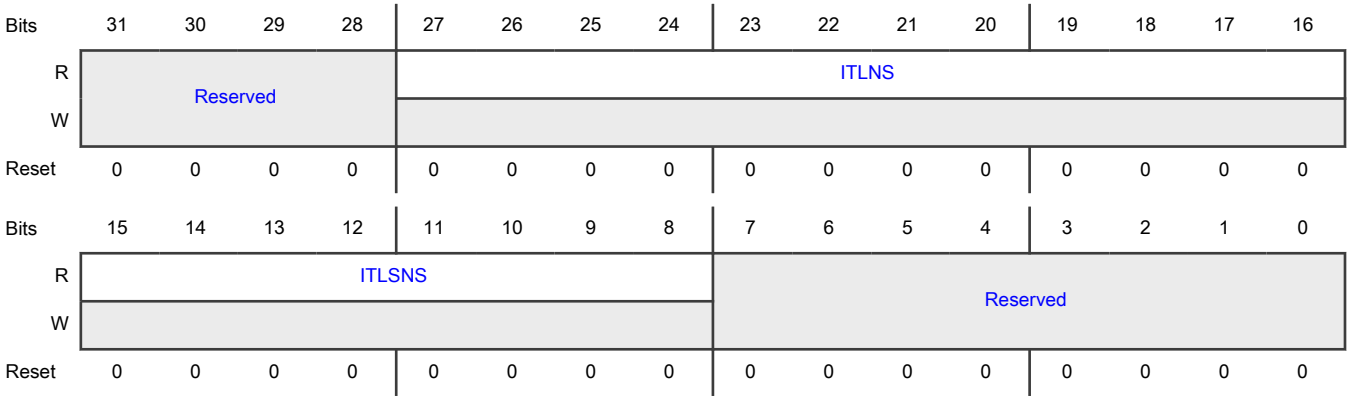
### Offset

| Register | Offset |
|---|---|
| MAC_Timestamp_Ingress_Latency | B68h |

### Function

Holds the ingress MAC latency.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-16 | Ingress Timestamp Latency In Sub-Nanoseconds |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| ITLNS | Holds the average latency, in sub-nanoseconds, between the MAC input ports (PHY_RXD_I) and the actual point (GMII/MII), where the ingress timestamp is taken. The ingress correction value is computed as described in section 7.1.2.4.1 of QoS Databook. |
| 15-8<br>ITLSNS | Ingress Timestamp Latency In Nanoseconds<br>Holds the average latency, in nanoseconds, between the MAC input ports (PHY_RXD_I) and the actual point (GMII/MII), where the ingress timestamp is taken. The ingress correction value is computed as described in section 7.1.2.4.1 of QoS Databook. |
| 7-0<br>— | Reserved |

## 72.17.166 MAC Timestamp Egress Latency (MAC_Timestamp_Egress_Latency)

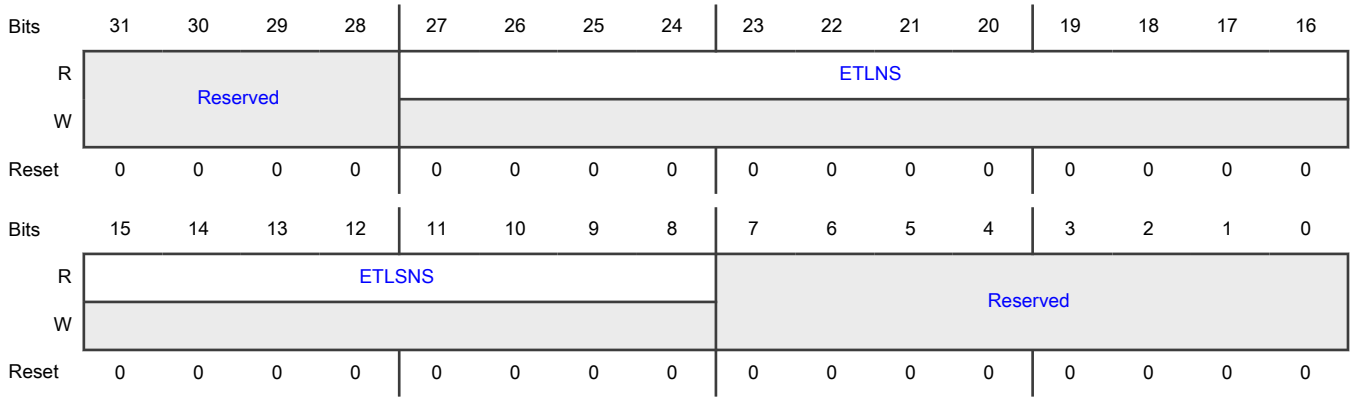
### Offset

| Register | Offset |
|---|---|
| MAC_Timestamp_Egress_Latency | B6Ch |

### Function

Holds the egress MAC latency.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 27-16<br>ETLNS | Egress Timestamp Latency In Nanoseconds<br>Holds the average latency, in nanoseconds, between the actual point (GMII/MII) where the egress timestamp is taken and the MAC output ports (PHY_TXD_O). The ingress correction value is computed as described in section 7.1.2.4.2 of QoS Databook. |
| 15-8<br>ETLSNS | Egress Timestamp Latency In Sub-Nanoseconds<br>Holds the average latency, in sub-nanoseconds, between the actual point (GMII/MII) where the egress timestamp is taken and the MAC output ports (PHY_TXD_O). The ingress correction value is computed as described in section 7.1.2.4.2 of QoS Databook. |
| 7-0<br>— | Reserved |

## 72.17.167 MAC PPS Control (MAC_PPS_Control)

### Offset

| Register | Offset |
|---|---|
| MAC_PPS_Control | B70h |

### Function

Indicates PPS control.

In this register:

- Bits [30:24] are valid only when four flexible PPS outputs are selected.
- Bits [22:16] are valid only when three or more flexible PPS outputs are selected.
- Bits [14:8] are valid only when two or more flexible PPS outputs are selected.
- Bits [6:4] are valid only when the flexible PPS feature is selected.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MCGREN3 | TRGTMODSEL3 | | Reserved | PPSCMD3 | | | | MCGREN2 | TRGTMODSEL2 | | Reserved | PPSCMD2 | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MCGREN1 | TRGTMODSEL1 | | Reserved | PPSCMD1 | | | | MCGREN0 | TRGTMODSEL0 | | PPSEN0 | PPSCTRL_PPSCMD | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>MCGREN3 | MCGR Mode Enable For PPS3 Output<br>Enables the third PPS instance to operate in PPS or MCGR mode. If this field is 1, it operates in MCGR mode, and if the value is 0, it operates in PPS mode. |
| 30-29<br>TRGTMODSEL3 | Target Time Register Mode For PPS3 Output<br>Indicates MAC PPS3 Target Time In Seconds (MAC_PPS3_Target_Time_Seconds) and MAC PPS3 Target Time In Nanoseconds (MAC_PPS3_Target_Time_Nanoseconds) mode for the PPS3 output signal.<br>00b - Target Time registers are programmed only for generating the interrupt event. The flexible PPS function must not be enabled in this mode, otherwise spurious transitions may be observed on the corresponding PTP_PPS_O output port.<br>01b - Reserved<br>10b - Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS0 output signal generation.<br>11b - Target Time registers are programmed only for starting or stopping the PPS0 output signal generation. No interrupt is asserted. |
| 28<br>— | Reserved |
| 27-24<br>PPSCMD3 | Flexible PPS3 Output Control<br>Controls the flexible PPS3 output (PTP_PPS_O[3]) signal. The functioning of this field is similar to that of PPSCMD0[2:0] (presentation time control bits).<br>If MCGREN3 = 1, PPSCMD3 that includes bits [27:24] are considered as presentation time control bits for media clock generation and recovery for comparator instance 3. If MCGREN3 is not 1, only three bits [26:24] are considered as PPSCMD3 and the fourth bit becomes 0.<br>Access restrictions apply to this field that clears automatically. Writing 0 to it has no effect. |
| 23<br>MCGREN2 | MCGR Mode Enable For PPS2 Output<br>Enables or disables the second PPS instance to operate in PPS or MCGR mode. If this field is 1, it operates in MCGR mode, and if the value is 0, it operates in PPS mode.<br>0b - Disabled<br>1b - Enabled |
| 22-21<br>TRGTMODSEL2 | Target Time Register Mode For PPS2 Output<br>Indicates MAC PPS2 Target Time In Seconds (MAC_PPS2_Target_Time_Seconds) and MAC PPS2 Target Time In Nanoseconds (MAC_PPS2_Target_Time_Nanoseconds) mode for the PPS2 output signal.<br>00b - Target Time registers are programmed only for generating the interrupt event. The flexible PPS function must not be enabled in this mode, otherwise spurious transitions may be observed on the corresponding PTP_PPS_O output port.<br>01b - Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 10b - Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS0 output signal generation.<br>11b - Target Time registers are programmed only for starting or stopping the PPS0 output signal generation. No interrupt is asserted. |
| 20<br>— | Reserved |
| 19-16<br>PPSCMD2 | Flexible PPS2 Output Control<br>Controls the flexible PPS2 output (PTP_PPS_O[2]) signal. The functioning of this field is similar to that of the PPSCMD0 (presentation time control bits).<br>If MCGREN2 = 1, the bits included in PPSCMD2 [19:16] are considered as presentation time control bits for media clock generation and recovery for comparator instance 2. If MCGREN2 is not 1, only three bits [18:16] are used as PPSCMD2 and the fourth bit becomes 0.<br>Access restrictions apply to this field that clears automatically. Writing 0 to it has no effect. |
| 15<br>MCGREN1 | MCGR Mode Enable For PPS1 Output<br>Enables or disables the first PPS instance to operate in PPS or MCGR mode. If this field is 1, it operates in MCGR mode, and if the value is 0, it operates in PPS mode.<br>0b - Disabled<br>1b - Enabled |
| 14-13<br>TRGTMODSEL1 | Target Time Register Mode For PPS1 Output<br>Indicates MAC PPS1 Target Time In Seconds (MAC_PPS1_Target_Time_Seconds) and MAC PPS1 Target Time In Nanoseconds (MAC_PPS1_Target_Time_Nanoseconds) mode for the PPS1 output signal.<br>00b - Target Time registers are programmed only for generating the interrupt event. The flexible PPS function must not be 1 in this mode, otherwise spurious transitions may be observed on the corresponding PTP_PPS_O output port.<br>01b - Reserved<br>10b - Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS0 output signal generation.<br>11b - Target Time registers are programmed only for starting or stopping the PPS0 output signal generation. No interrupt is asserted. |
| 12<br>— | Reserved |
| 11-8<br>PPSCMD1 | Flexible PPS1 Output Control<br>Controls the flexible PPS1 output (PTP_PPS_O[1]) signal. The functioning of this field is similar to that of the PPSCMD0 field. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | If MCGREN1 = 1, the bits included in PPSCMD1 [11:8] are considered as presentation time control bits for media clock generation and recovery for comparator instance 1. The functioning of PPSCMD1 is similar to that of the PPSCMD0 presentation time control bits. If MCGREN1 is not 1, only three bits [10:8] are used as PPSCMD1 and the fourth bit becomes 0.<br>Access restrictions apply to this field that clears automatically. Writing 0 to it has no effect. |
| 7<br>MCGREN0 | MCGR Mode Enable For PPS0 Output<br>Indicates whether the 0th PPS instance is enabled to operate in PPS or MCGR mode. If this field is 1, it operates in MCGR mode, and if the value is 0, it operates in PPS mode.<br>0b - PPS mode<br>1b - MCGR mode |
| 6-5<br>TRGTMODSEL0 | Target Time Register Mode For PPS0 Output<br>Indicates MAC PPS0 Target Time In Seconds (MAC_PPS0_Target_Time_Seconds) and MAC PPS0 Target Time In Nanoseconds (MAC_PPS0_Target_Time_Nanoseconds) mode for the PPS0 output signal.<br>00b - Target Time registers are programmed only for generating the interrupt event. The flexible PPS function must not be 1 in this mode, otherwise spurious transitions may be observed on the corresponding PTP_PPS_O output port.<br>01b - Reserved<br>10b - Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS0 output signal generation.<br>11b - Target Time registers are programmed only for starting or stopping the PPS0 output signal generation. No interrupt is asserted. |
| 4<br>PPSEN0 | Flexible PPS Output Mode Enable 0<br>Indicates the status of Flexible PPS Output mode.<br>• If this field is 1, the PPSCTRL_PPSCMD field functions as PPSCMD.<br>• If this field is 0, the PPSCTRL_PPSCMD field functions as PPSCTRL (fixed PPS mode).<br>0b - Disabled<br>1b - Enabled |
| 3-0<br>PPSCTRL_PPSCMD | PPS Output Frequency Control<br>Controls the frequency of the PPS0 output (PTP_PPS_O[0] signal). The default value of this field is 0, and the PPS output is 1 pulse (of width clk_ptp_i) every second. For the other values of PPSCTRL, the PPS output becomes a generated clock having these frequencies:<br>• 0001: The binary rollover is 2 Hz, and the digital rollover is 1 Hz.<br>• 0010: The binary rollover is 4 Hz, and the digital rollover is 2 Hz.<br>• 0011: The binary rollover is 8 Hz, and the digital rollover is 4 Hz.<br>• 0100: The binary rollover is 16 Hz, and the digital rollover is 8 Hz. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • ..<br>• 1111: The binary rollover is 32.768 kHz and the digital rollover is 16.384 kHz.<br>**NOTE**<br>In Binary Rollover mode, the PPS output (PTP_PPS_O) has a duty cycle of 50 percent with these frequencies. In Digital Rollover mode, the PPS output frequency is an average number. The actual clock has a different frequency that is synchronized every second.<br>For example,<br>• If PPSCTRL = 0001, PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms.<br>• If PPSCTRL = 0010, PPS (2 Hz) is a sequence of one clock of 50 percent duty cycle and a 537 ms period. The second clock has a period of 463 ms (268 ms low and 195 ms high).<br>• If PPSCTRL = 0011, PPS (4 Hz) is a sequence of three clocks of 50 percent duty cycle and a 268 ms period, and the fourth clock of 195 ms period (134 ms low and 61 ms high).<br>This behavior is a result of the non-linear toggling of bits in Digital Rollover mode in MAC System Time In Nanoseconds (MAC_System_Time_Nanoseconds) or flexible PPS output (PTP_PPS_O[0]) control.<br>Programming these bits with a non-zero value instructs MAC to initiate an event. When the command is transferred or synchronized with the PTP clock domain, these bits automatically become 0. You must ensure that these bits are programmed only when they are "all-zero." The following list describes the values of PPSCMD0.<br>• 0000: No command<br>• 0001: Start a single pulse. This command generates a single pulse rising at the start point defined in MAC PPS0 Target Time In Seconds (MAC_PPS0_Target_Time_Seconds) and MAC PPS0 Target Time In Nanoseconds (MAC_PPS0_Target_Time_Nanoseconds), for a duration defined in MAC PPS0 Width (MAC_PPS0_Width).<br>• 0010: Start pulse train. This command generates the train of pulses rising at the start point defined in the Target Time registers and for a duration defined in MAC PPS0 Width (MAC_PPS0_Width), repeated at an interval defined in the PPS Interval registers. By default, the PPS pulse train runs freely unless the "stop pulse train at time" or "stop pulse train immediately" command stops it.<br>• 0011: Cancel start. This command cancels the start single pulse and start pulse train commands if the system time has not crossed the programmed start time.<br>• 0100: Stop pulse train at time. This command stops the train of pulses that the start pulse train command (PPSCMD = 0010) initiates after the time programmed in the Target Time registers elapses.<br>• 0101: Stop pulse train immediately. This command immediately stops the train of pulses that the start pulse train command (PPSCMD = 0010) initiates.<br>• 110: Cancel stop pulse train. This command cancels the stop pulse train at the time command if the programmed stop time has not elapsed. The PPS pulse train runs freely if the command executes successfully.<br>• 0111-1111: Reserved or presentation time control. If MAC_PPS_Control[MCGREN0] = 0, then these are treated as presentation time control bits.<br>This list describes the values of PPSCMD0: |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • 0000: The MCGR operation is not carried out. If set to this value in the mid of clock recovery or generation, all the processing inputs are flushed.<br>• 0001: Captures the presentation time at the rising edge of MCG_PST_TRIG_I[0] in MAC PPS0 Target Time In Seconds (MAC_PPS0_Target_Time_Seconds)<br>• 0010: Captures the presentation time at the falling edge of MCG_PST_TRIG_I[0] in MAC PPS0 Target Time In Seconds (MAC_PPS0_Target_Time_Seconds)<br>• 0011: Captures the presentation time at both the edges of MCG_PST_TRIG_I[0] in MAC PPS0 Target Time In Seconds (MAC_PPS0_Target_Time_Seconds)<br>• 0100-1000: Reserved<br>• 1001: Toggle output on compare<br>• 1010: Pulse output low on compare for one PTP-clock cycle<br>• 1011: Pulse output high on compare for one PTP-clock cycle<br>• 1100-1111: Reserved |

## 72.17.168 MAC PPS0 Target Time In Seconds (MAC_PPS0_Target_Time_Seconds)

### Offset

| Register | Offset |
|---|---|
| MAC_PPS0_Target_Time_Seconds | B80h |

### Function

Is used, along with the PPS Target Time Nanoseconds register, to schedule an interrupt event (MAC_Timestamp_Status[TSTARGT0]) when the system time exceeds the value programmed in these registers.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSTRH0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSTRH0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>TSTRH0 | PPS Target Time In Seconds Register<br>Stores the target time in seconds. When the timestamp value matches or exceeds both the Target Timestamp registers, MAC starts or stops the PPS signal output and generates an interrupt (if enabled), based on Target Time mode selected for the corresponding PPS output in MAC PPS Control (MAC_PPS_Control).<br>If DWC_EQOS_FLEXI_PPS_OUT_EN is enabled in the configuration and MAC_Timestamp_Control[PTGE] = 1, with the presentation time control set in Recovery mode, these fields indicate that the application programs the TPT. In Generation mode, it indicates that CPT is generated at the sampled trigger. |

## 72.17.169 MAC PPS0 Target Time In Nanoseconds (MAC_PPS0_Target_Time_Nanoseconds)

### Offset

| Register | Offset |
|---|---|
| MAC_PPS0_Target_Time_Nanoseconds | B84h |

### Function
Indicates the target time in nanoseconds for PPS0.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TRGTBUS... | | | | TTSL0 | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TTSL0 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>TRGTBUSY0 | PPS Target Time Busy Status 0<br>Indicates whether the PPS target time busy status 0 is detected. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • MAC writes 1 to this field when MAC_PPS_Control[PPSCTRL_PPSCMD] is programmed to 010 or 011. This instructs MAC to synchronize the Target Time registers with the PTP clock domain.<br>• MAC writes 0 to this field after synchronizing the Target Time registers with the PTP clock domain. The application must not update the Target Time registers when this field is read as 1. Otherwise, the synchronization of the previously programmed time is corrupted.<br>0b - Not detected<br>1b - Detected |
| 30-0<br>TTSL0 | Target Time Low For PPS0<br>Stores the time in (signed) nanoseconds.<br>If the value of the timestamp matches the value in both the Target Timestamp registers, MAC starts or stops the PPS signal output and generates an interrupt, if enabled, based on MAC_PPS_Control[TRGTMODSEL0].<br>• If MAC_Timestamp_Control[TSCTRLSSR] = 0, this value must be calculated as (time in ns / 0.465). The actual start or stop time of the PPS signal output might have an error margin up to one unit of the sub-second increment value.<br>• If MAC_Timestamp_Control[TSCTRLSSR] = 1, this value must not exceed 3B9A_C9FFh. The actual start or stop time of the PPS signal output might have an error margin up to one unit of the sub-second increment value.<br>Access restriction apply to this field that clears automatically. Writing 0 to it has no effect. |

## 72.17.170 MAC PPS0 Interval (MAC_PPS0_Interval)

### Offset

| Register | Offset |
|---|---|
| MAC_PPS0_Interval | B88h |

### Function

Contains the number of units of the sub-second increment value between the rising edges of the PPS0 signal output (PTP_PPS_O[0]).

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSINT0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSINT0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>PPSINT0 | PPS Output Signal Interval 0<br>Stores the interval between the rising edges of the PPS0 signal output. The interval is stored in terms of the number of units of the sub-second increment value.<br>You must program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20 ns), and the desired interval between the rising edges of the PPS0 signal output is 100 ns (that is, five units of the sub-second increment value), you must program value 4 (5-1) in this register. |

## 72.17.171 MAC PPS0 Width (MAC_PPS0_Width)

### Offset

| Register | Offset |
|---|---|
| MAC_PPS0_Width | B8Ch |

### Function

Contains the number of units of the sub-second increment value between the rising and corresponding falling edges of the PPS0 signal output (PTP_PPS_O[0]).

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSWIDTH0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSWIDTH0 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>PPSWIDTH0 | PPS Output Signal Width 0<br>Stores the width between the rising and corresponding falling edges of the PPS0 signal output. The width is stored in terms of the number of units of the sub-second increment value.<br>You must program one value less than the required interval. For example, if the PTP reference clock has a frequency of 50 MHz (period of 20 ns), and a width between the rising and corresponding falling edges of the PPS0 signal output is 80 ns (that is, four units of the sub-second increment value), you should program value 3 (4-1) in this register.<br>**NOTE**<br>The value programmed in this register must be lesser than the value programmed in MAC PPS0 Interval (MAC_PPS0_Interval). |

## 72.17.172 MAC PPS1 Target Time In Seconds (MAC_PPS1_Target_Time_Seconds)

### Offset

| Register | Offset |
|---|---|
| MAC_PPS1_Target_Time_Seconds | B90h |

### Function
Is used to schedule an interrupt event (MAC_Timestamp_Status[TSTARGT0]), along with the PPS Target Time Nanoseconds register, when the system time exceeds the value programmed in these registers.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSTRH1 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSTRH1 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0 | PPS Target Time In Seconds 1 |

*Table continues on the next page...*

| Field | Function |
|---|---|
| TSTRH1 | Stores the target time in seconds.<br>When the timestamp value matches or exceeds both the Target Timestamp registers, MAC starts or stops the PPS signal output and generates an interrupt, if enabled, based on Target Time mode selected for the corresponding PPS output in MAC PPS Control (MAC_PPS_Control).<br>If DWC_EQOS_FLEXI_PPS_OUT_EN is enabled in the configuration and MAC_Timestamp_Control[PTGE] = 1, with the presentation time control set in Recovery mode, these fields indicate that the application programs the TPT. In Generation mode, it indicates that CPT is generated at the sampled trigger. |

## 72.17.173 MAC PPS1 Target Time In Nanoseconds (MAC_PPS1_Target_Time_Nanoseconds)

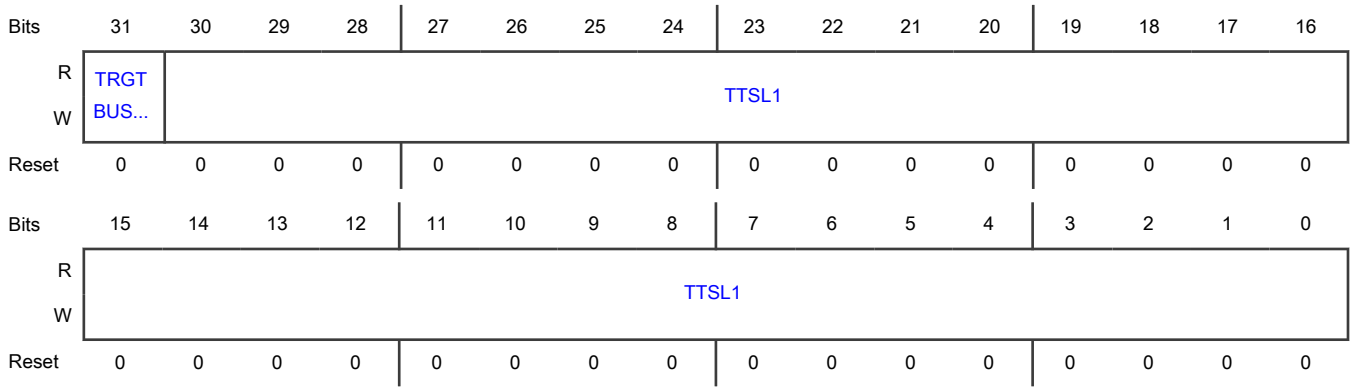
### Offset

| Register | Offset |
|---|---|
| MAC_PPS1_Target_Time_Nanoseconds | B94h |

### Function

Indicates the target time in nanoseconds for PPS1.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>TRGTBUSY1 | PPS Target Time Busy Status 1<br>Indicates whether the PPS target time busy status 1 is detected.<br>• MAC writes 1 to this field when MAC_PPS_Control[PPSCTRL_PPSCMD] is programmed to 010 or 011. Programming the PPSCMD0 field as 010 or 011 instructs MAC to synchronize the Target Time registers with the PTP clock domain. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • MAC writes 0 to this field after synchronizing the Target Time registers with the PTP clock domain. The application must not update the Target Time registers when this field is read as 1. Otherwise, the synchronization of the previously programmed time is corrupted.<br>0b - Not detected<br>1b - Detected |
| 30-0<br>TTSL1 | Target Time Low For PPS1<br>Stores the time in (signed) nanoseconds.<br>If the value of the timestamp matches the value in both the Target Timestamp registers, MAC starts or stops the PPS signal output and generates an interrupt, if enabled, based on MAC_PPS_Control[TRGTMODSEL0].<br>• If MAC_Timestamp_Control[TSCTRLSSR] = 0, this value must be defined as (time in ns / 0.465). The actual start or stop time of the PPS signal output might have an error margin up to one unit of the sub-second increment value.<br>• If MAC_Timestamp_Control[TSCTRLSSR] = 1, this value must not exceed 3B9A_C9FFh. The actual start or stop time of the PPS signal output might have an error margin up to one unit of the sub-second increment value.<br>Access restriction apply to this field that clears automatically. Writing 0 to it has no effect. |

## 72.17.174 MAC PPS1 Interval (MAC_PPS1_Interval)

### Offset

| Register | Offset |
|---|---|
| MAC_PPS1_Interval | B98h |

### Function

Contains the number of units of the sub-second increment value between the rising edges of the PPS0 signal output (PTP_PPS_O[0]).

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSINT1 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSINT1 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>PPSINT1 | PPS Output Signal Interval 1<br>Stores the interval between the rising edges of the PPS0 signal output. The interval is stored in terms of the number of units of the sub-second increment value.<br>You must program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20 ns), and the desired interval between the rising edges of the PPS0 signal output is 100 ns (that is, five units of the sub-second increment value), you must program value 4 (5-1) in this register. |

## 72.17.175 MAC PPS1 Width (MAC_PPS1_Width)

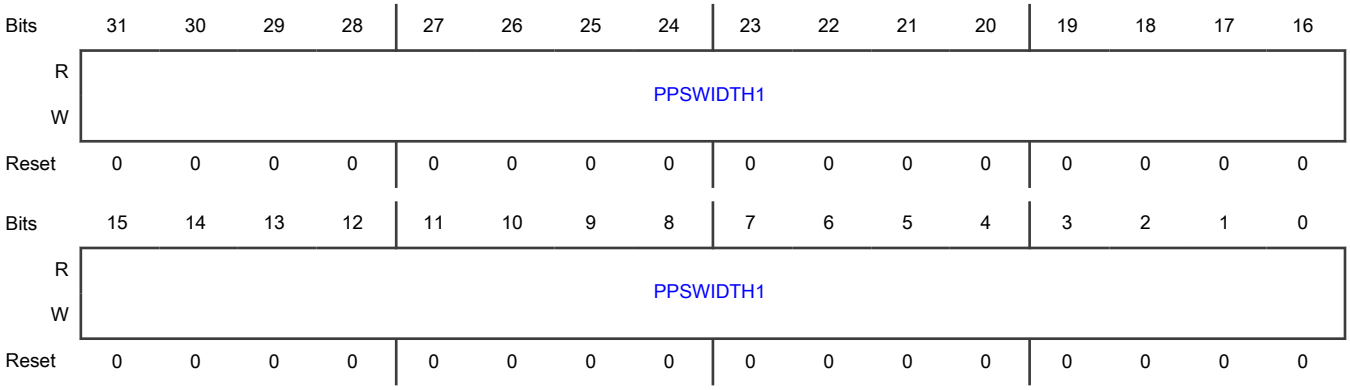
### Offset

| Register | Offset |
|---|---|
| MAC_PPS1_Width | B9Ch |

### Function

Contains the number of units of the sub-second increment value between the rising and corresponding falling edges of the PPS0 signal output (PTP_PPS_O[0]).

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>PPSWIDTH1 | PPS Output Signal Width 1<br>Stores the width between the rising and corresponding falling edges of the PPS0 signal output. The width is stored in terms of the number of units of the sub-second increment value.<br>You must program one value less than the required interval. For example, if the PTP reference clock has a frequency of 50 MHz (period of 20 ns), and a width between the rising and corresponding falling edges of the PPS0 signal output is 80 ns (that is, four units of the sub-second increment value), you should program value 3 (4-1) in this register. |

*Table continues on the next page...*

| Field | Function |
|---|---|
| | **NOTE**<br>The value programmed in this register must be lesser than the value programmed in MAC PPS0 Interval (MAC_PPS0_Interval). |

## 72.17.176 MAC PPS2 Target Time In Seconds (MAC_PPS2_Target_Time_Seconds)

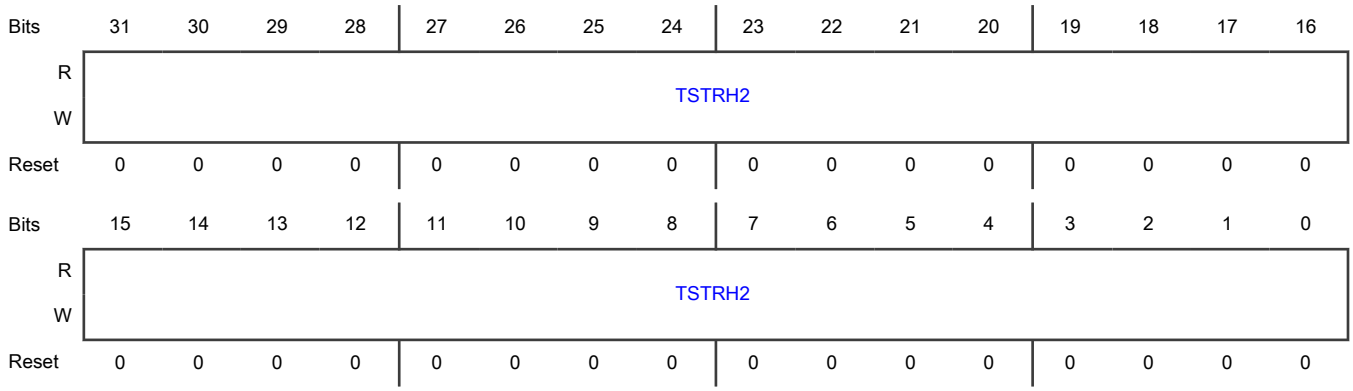
### Offset

| Register | Offset |
|---|---|
| MAC_PPS2_Target_Time_Seconds | BA0h |

### Function

Is used to schedule an interrupt event (MAC_Timestamp_Status[TSTARGT0]), along with the PPS Target Time Nanoseconds register, when the system time exceeds the value programmed in these registers.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>TSTRH2 | PPS Target Time In Seconds 2<br>Stores the target time in seconds.<br>When the timestamp value matches or exceeds both the Target Timestamp registers, MAC starts or stops the PPS signal output and generates an interrupt, if enabled, based on Target Time mode selected for the corresponding PPS output in MAC PPS Control (MAC_PPS_Control).<br>If DWC_EQOS_FLEXI_PPS_OUT_EN is enabled in the configuration and MAC_Timestamp_Control[PTGE] = 1, with the presentation time control set in Recovery mode, these fields indicate that the application programs the TPT. In Generation mode, it indicates that CPT is generated at the sampled trigger. |

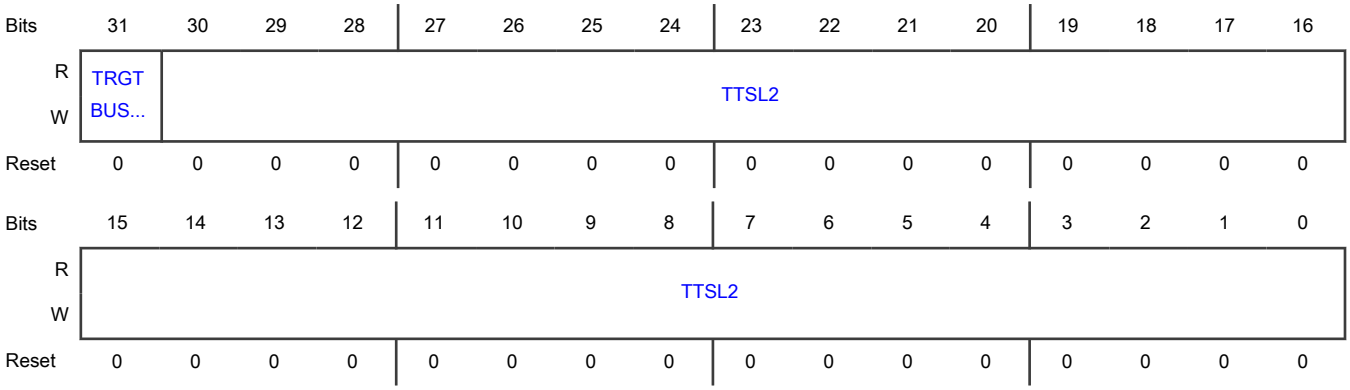
## 72.17.177 MAC PPS2 Target Time In Nanoseconds (MAC_PPS2_Target_Time_Nanoseconds)

### Offset

| Register | Offset |
|---|---|
| MAC_PPS2_Target_Time_Nanoseconds | BA4h |

### Function
Indicates the target time in nanoseconds for PPS2.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>TRGTBUSY2 | PPS Target Time Busy Status 2<br>Indicates whether the PPS target time busy status 2 is detected.<br>• MAC writes 1 to this field when MAC_PPS_Control[PPSCTRL_PPSCMD] is programmed to 010 or 011. This instructs MAC to synchronize the Target Time registers with the PTP clock domain.<br>• MAC writes 0 to this field after synchronizing the Target Time registers with the PTP clock domain. The application must not update the Target Time registers when this field is read as 1. Otherwise, the synchronization of the previously programmed time is corrupted.<br>0b - Not detected<br>1b - Detected |
| 30-0<br>TTSL2 | Target Time Low For PPS2<br>Stores the time in (signed) nanoseconds.<br>If the value of the timestamp matches the value in both the Target Timestamp registers, MAC starts or stops the PPS signal output and generates an interrupt, if enabled, based on MAC_PPS_Control[TRGTMODSEL0]. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • If MAC_Timestamp_Control[TSCTRLSSR] = 0, this value must be defined as (time in ns / 0.465). The actual start or stop time of the PPS signal output might have an error margin up to one unit of the sub-second increment value.<br>• If MAC_Timestamp_Control[TSCTRLSSR] = 1, this value must not exceed 3B9A_C9FFh. The actual start or stop time of the PPS signal output might have an error margin up to one unit of the sub-second increment value.<br>Access restriction apply to this field that clears automatically. Writing 0 to it has no effect. |

## 72.17.178 MAC PPS2 Interval (MAC_PPS2_Interval)

### Offset

| Register | Offset |
|---|---|
| MAC_PPS2_Interval | BA8h |

### Function

Contains the number of units of the sub-second increment value between the rising edges of the PPS0 signal output (PTP_PPS_O[0]).

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSINT2 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSINT2 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>PPSINT2 | PPS Output Signal Interval 2<br>Stores the interval between the rising edges of the PPS0 signal output. The interval is stored in terms of the number of units of the sub-second increment value.<br>You must program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20 ns), and the desired interval between the rising edges of the PPS0 signal output is 100 ns (that is, five units of the sub-second increment value), you must program value 4 (5-1) in this register. |

## 72.17.179 MAC PPS2 Width (MAC_PPS2_Width)

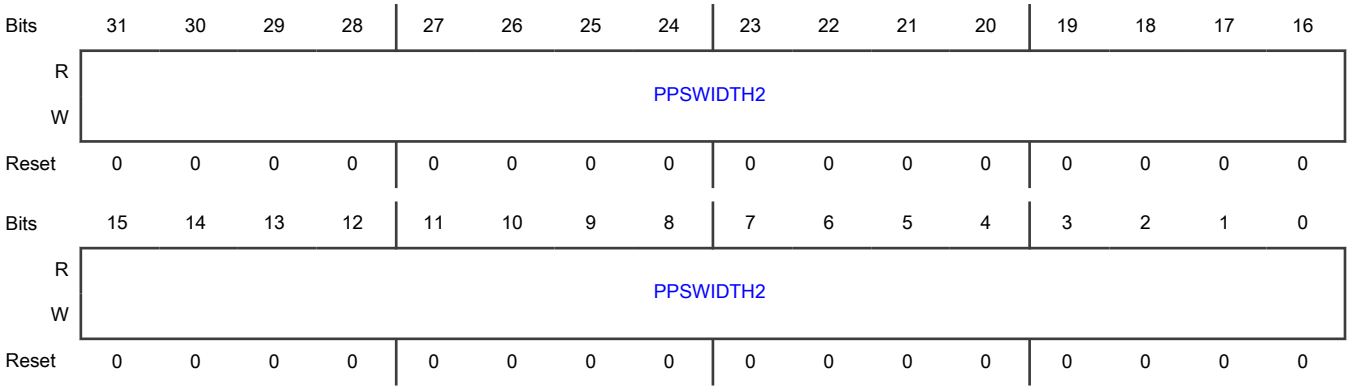
### Offset

| Register | Offset |
|---|---|
| MAC_PPS2_Width | BACh |

### Function

Contains the number of units of the sub-second increment value between the rising and corresponding falling edges of the PPS0 signal output (PTP_PPS_O[0]).

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>PPSWIDTH2 | PPS Output Signal Width 2<br>Stores the width between the rising and corresponding falling edges of the PPS0 signal output. The width is stored in terms of the number of units of the sub-second increment value.<br>You must program one value less than the required interval. For example, if the PTP reference clock has a frequency of 50 MHz (period of 20 ns), and a width between the rising and corresponding falling edges of the PPS0 signal output is 80 ns (that is, four units of the sub-second increment value), you should program value 3 (4-1) in this register.<br>**NOTE**<br>The value programmed in this register must be lesser than the value programmed in MAC PPS0 Interval (MAC_PPS0_Interval). |

## 72.17.180 MAC PPS3 Target Time In Seconds (MAC_PPS3_Target_Time_Seconds)

### Offset

| Register | Offset |
|---|---|
| MAC_PPS3_Target_Time_Seconds | BB0h |

### Function

Is used to schedule an interrupt event (MAC_Timestamp_Status[TSTARGT0]), along with the PPS Target Time Nanoseconds register, when the system time exceeds the value programmed in these registers.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSTRH3 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TSTRH3 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>TSTRH3 | PPS Target Time In Seconds 3<br>Stores the target time in seconds.<br>When the timestamp value matches or exceeds both the Target Timestamp registers, MAC starts or stops the PPS signal output and generates an interrupt, if enabled, based on Target Time mode selected for the corresponding PPS output in MAC PPS Control (MAC_PPS_Control).<br>If DWC_EQOS_FLEXI_PPS_OUT_EN is enabled in the configuration and MAC_Timestamp_Control[PTGE] = 1, with the presentation time control set in Recovery mode, these fields indicate that the application programs the TPT. In Generation mode, it indicates that CPT is generated at the sampled trigger. |

## 72.17.181 MAC PPS3 Target Time In Nanoseconds (MAC_PPS3_Target_Time_Nanoseconds)

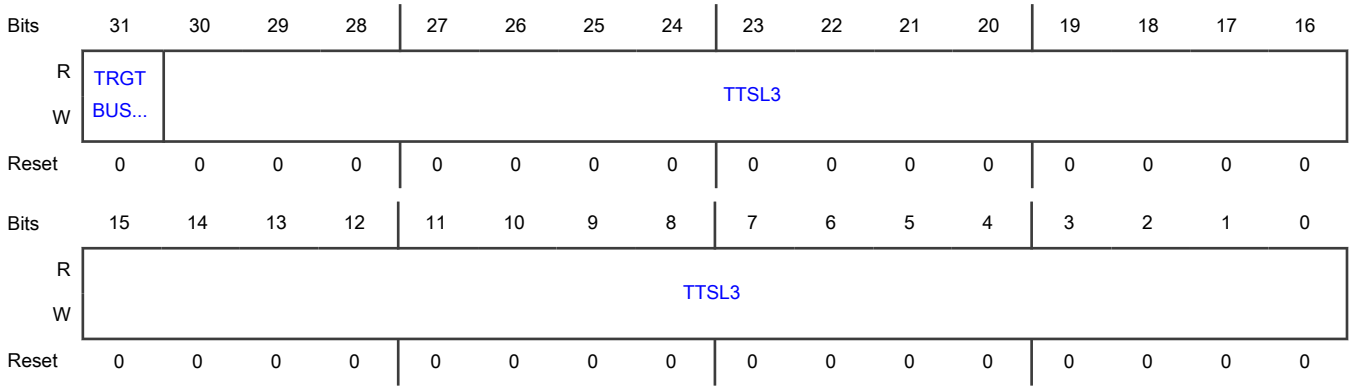
### Offset

| Register | Offset |
|---|---|
| MAC_PPS3_Target_Time_Nanoseconds | BB4h |

### Function

Indicates the target time in nanoseconds for PPS3.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>TRGTBUSY3 | PPS Target Time Register Busy 3<br>Indicates whether the PPS target time register busy status is detected.<br>MAC writes 1 to this field when MAC_PPS_Control[PPSCTRL_PPSCMD] is programmed to 010 or 011. This instructs MAC to synchronize the Target Time registers with the PTP clock domain.<br>MAC writes 0 to this field after synchronizing the Target Time registers with the PTP clock domain. The application must not update the Target Time registers when this field is read as 1. Otherwise, the synchronization of the previously programmed time is corrupted.<br>0b - Not detected<br>1b - Detected |
| 30-0<br>TTSL3 | Target Time Low For PPS3<br>Stores the time in (signed) nanoseconds.<br>If the value of the timestamp matches the value in both the Target Timestamp registers, MAC starts or stops the PPS signal output and generates an interrupt, if enabled, based on MAC_PPS_Control[TRGTMODSEL0].<br>• If MAC_Timestamp_Control[TSCTRLSSR] = 0, this value must be defined as (time in ns / 0.465). The actual start or stop time of the PPS signal output might have an error margin up to one unit of the sub-second increment value.<br>• If MAC_Timestamp_Control[TSCTRLSSR] = 1, this value must not exceed 3B9A_C9FFh. The actual start or stop time of the PPS signal output might have an error margin up to one unit of the sub-second increment value.<br>Access restriction apply to this field that clears automatically. Writing 0 to it has no effect. |

## 72.17.182 MAC PPS3 Interval (MAC_PPS3_Interval)

### Offset

| Register | Offset |
|---|---|
| MAC_PPS3_Interval | BB8h |

### Function

Contains the number of units of the sub-second increment value between the rising edges of the PPS0 signal output (PTP_PPS_O[0]).

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSINT3 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | PPSINT3 | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>PPSINT3 | PPS Output Signal Interval<br>Stores the interval between the rising edges of the PPS0 signal output. The interval is stored in terms of the number of units of the sub-second increment value.<br>You must program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20 ns), and the desired interval between the rising edges of the PPS0 signal output is 100 ns (that is, five units of the sub-second increment value), you must program value 4 (5-1) in this register. |

## 72.17.183 MAC PPS3 Width (MAC_PPS3_Width)

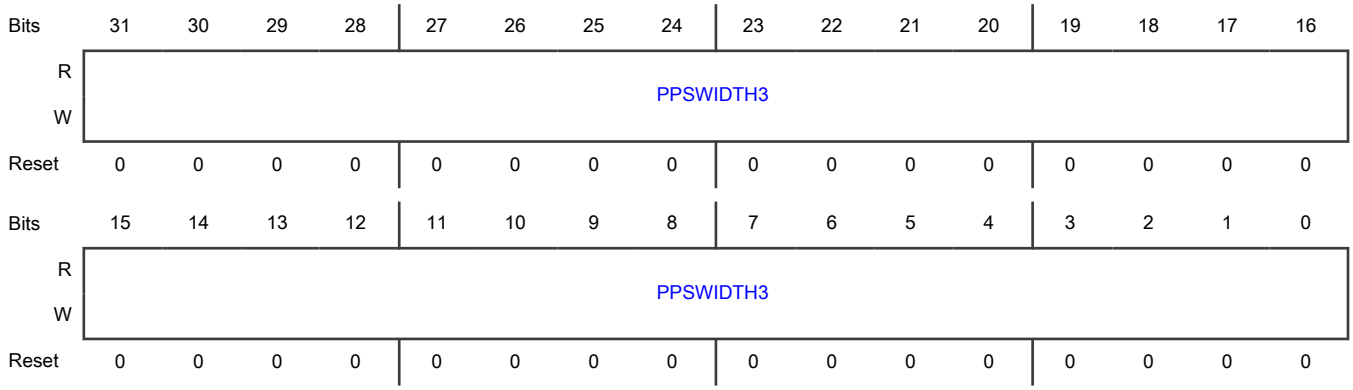
### Offset

| Register | Offset |
|---|---|
| MAC_PPS3_Width | BBCh |

### Function

Contains the number of units of the sub-second increment value between the rising and corresponding falling edges of the PPS0 signal output (PTP_PPS_O[0]).

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>PPSWIDTH3 | PPS Output Signal Width 3<br>Stores the width between the rising and corresponding falling edges of the PPS0 signal output. The width is stored in terms of the number of units of the sub-second increment value.<br>You must program one value less than the required interval. For example, if the PTP reference clock has a frequency of 50 MHz (period of 20 ns), and a width between the rising and corresponding falling edges of the PPS0 signal output is 80 ns (that is, four units of the sub-second increment value), you should program value 3 (4-1) in this register.<br>**NOTE**<br>The value programmed in this register must be lesser than the value programmed in MAC PPS0 Interval (MAC_PPS0_Interval). |

## 72.17.184 MTL Operation Mode (MTL_Operation_Mode)

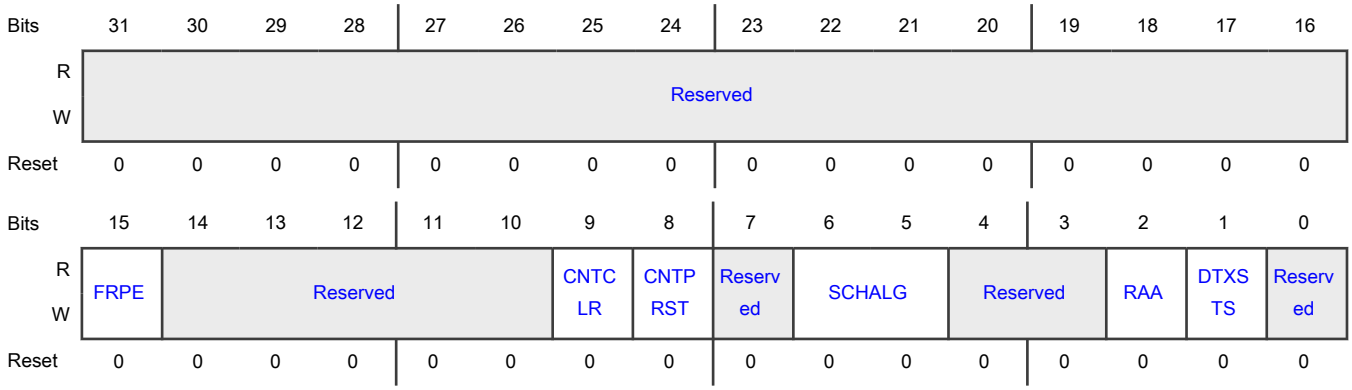
### Offset

| Register | Offset |
|---|---|
| MTL_Operation_Mode | C00h |

### Function

Establishes the transmit and receive operating modes and commands.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>FRPE | Flexible Receive Parser Enable<br>Indicates the status of the flexible receive parser.<br>• If this field is 1, the programmable receive parser functionality is enabled.<br>• If the receive parser is disabled and is in the middle of parsing, the receive parser functionality is disabled only after completing the current packet parsing.<br>When the receive parser is enabled from the disabled state, the receive parser is activated for the next immediate packet.<br>0b - Disabled<br>1b - Enabled |
| 14-10<br>— | Reserved |
| 9<br>CNTCLR | Counters Reset<br>Indicates whether the counters are reset.<br>• If this field is 1, all the counters are reset. The field clears automatically after one clock cycle.<br>• If you write 1 to this field along with the CNTPRST field, the CNT_PRESET field takes precedence.<br>Access restrictions apply to this field that clears automatically. Writing 0 to it has no effect.<br>0b - Not reset<br>1b - Reset |
| 8<br>CNTPRST | Counters Preset<br>Indicates the status of preset counters.<br>If this field is 1, |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • MTL Tx Queue 0 Underflow (MTL_TxQ0_Underflow)[bits 0-7] is initialized or preset to 12'h7F0.<br>• Missed packet and overflow packet counters in MTL Rx Queue Missed Packet Overflow Count (MTL_RxQ0_Missed_Packet_Overflow_Cnt)[bits 0-7] are initialized or preset to 12'h7F0.<br>Access restrictions apply to this field that clears automatically. Writing 0 to it has no effect.<br>0b - Disabled<br>1b - Enabled |
| 7<br>— | Reserved |
| 6-5<br>SCHALG | Transmit Scheduling Algorithm<br>Indicates the algorithm for transmit scheduling.<br>00b - WRR algorithm<br>01b - WFQ algorithm when DCB feature selected; otherwise, reserved<br>10b - DWRR algorithm when DCB feature selected; otherwise, reserved<br>11b - Strict priority algorithm |
| 4-3<br>— | Reserved |
| 2<br>RAA | Receive Arbitration Algorithm<br>Is used to select the arbitration algorithm for the receive side. Queue 0 has the lowest priority and the last queue has the highest priority.<br>0b - Strict priority (SP)<br>1b - Weighted strict priority (WSP) |
| 1<br>DTXSTS | Drop Transmit Status<br>Indicates whether the drop transmit status is enabled.<br>• If this field is 1, the transmit packet status received from MAC is dropped in MTL.<br>• If this field is 0, the transmit packet status received from MAC is forwarded to the application.<br>0b - Disabled<br>1b - Enabled |
| 0<br>— | Reserved |

## 72.17.185 MTL Debug Control (MTL_DBG_CTL)

### Offset

| Register | Offset |
|---|---|
| MTL_DBG_CTL | C08h |

### Function

Controls, along with MTL Debug Status (MTL_DBG_STS), the operation mode of FIFO debug access.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | | | | | | EIEC | | EIEE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | STSIE | PKTIE | FIFOSEL | | FIFOWREN | FIFORDEN | RSTSEL | RSTALL | Reserved | PKTSTATE | | Reserved | BYTEEN | | DBGMOD | FDBGEN |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18-17<br>EIEC | ECC Inject Error Control<br>Indicates the status of the ECC inject error control feature for transmit, receive, and TSO memories.<br>If EIEE = 1, following are the errors inserted based on the value encoded in this field.<br>00b - 1-bit error<br>01b - 2-bit errors<br>10b - 3-bit errors<br>11b - 1-bit error in the address field |
| 16<br>EIEE | ECC Inject Error Enable<br>Indicates the status of the ECC error injection feature for transmit, receive, and TSO memories.<br>• If the value if this field is 1, it enables the ECC error injection feature.<br>• If the value if this field is 0, it disables the ECC error injection feature.<br>0b - Disabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Enabled |
| 15<br>STSIE | Transmit Status Available Interrupt Status Enable<br>Indicates whether the transmit packet available interrupt status is enabled.<br>If this field is 1, an interrupt is generated when transmit status is available in Slave mode.<br>0b - Disabled<br>1b - Enabled |
| 14<br>PKTIE | Receive Packet Available Interrupt Status Enable<br>Indicates whether the receive packet available interrupt status is enabled.<br>If this field is 1, an interrupt is generated when EOP of the received packet is written to the receive FIFO.<br>0b - Disabled<br>1b - Enabled |
| 13-12<br>FIFOSEL | FIFO Selected for Access<br>Indicates the FIFO selected for debug access.<br>00b - Transmit FIFO<br>01b - Transmit status FIFO (only read access when SLVMOD is set)<br>10b - TSO FIFO (cannot be accessed when SLVMOD is set)<br>11b - Receive FIFO |
| 11<br>FIFOWREN | FIFO Write Enable<br>Indicates the status of FIFO write.<br>If this field is 1, it enables the write operation on the selected FIFO when FIFO debug access is enabled.<br>This field must not be written to 1 when FIFO debug access is disabled, that is, when FDBGEN = 0.<br>Access restrictions apply to this field that clears automatically. Also, writing 0 to the field clears it and writing 1 sets it.<br>0b - Disabled<br>1b - Enabled |
| 10<br>FIFORDEN | FIFO Read Enable<br>Indicates the status of FIFO read.<br>If this field is 1, it enables the read operation on the selected FIFO when FIFO debug access is enabled.<br>This field must not be written to 1 when FIFO debug access is disabled, that is, when FDBGEN = 0.<br>Access restrictions apply to this field that clears automatically. Also, writing 0 to the field clears it and writing 1 sets it.<br>0b - Disabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Enabled |
| 9<br>RSTSEL | Reset Pointers Of Selected FIFO<br>Indicates whether the resetting of pointers for the selected FIFO is enabled.<br>If this field is 1, the pointers of the currently selected FIFO are reset when FIFO debug access is enabled.<br>This field must not be written to 1 when FIFO debug access is disabled, that is, when FDBGEN = 0.<br>Access restrictions apply to this field that clears automatically. Also, writing 0 to the field clears it and writing 1 sets it.<br>0b - Disabled<br>1b - Enabled |
| 8<br>RSTALL | Reset All Pointers<br>Indicates whether the resetting of all the pointers is enabled.<br>If this field is 1, the pointers of all the FIFOs are reset when the FIFO debug access is enabled.<br>This field must not be written to 1 when FIFO debug access is disabled, that is, when FDBGEN = 0.<br>Access restrictions apply to this field that clears automatically. Also, writing 0 to the field clears it and writing 1 sets it.<br>0b - Disabled<br>1b - Enabled |
| 7<br>— | Reserved |
| 6-5<br>PKTSTATE | Encoded Packet State<br>Is used to write the control information to the transmit FIFO or receive FIFO.<br>These are the values related to the transmit FIFO:<br>• 00: Packet data<br>• 01: Control word<br>• 10: SOP data<br>• 11: EOP data<br>These are the values related to the receive FIFO:<br>• 00: Packet data<br>• 01: Normal status<br>• 10: Last status<br>• 11: EOP<br>00b - Packet data |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 01b - Control word/normal status<br>10b - SOP data/last status<br>11b - EOP data/EOP |
| 4<br>— | Reserved |
| 3-2<br>BYTEEN | Byte Enables<br>Indicates the number of data bytes valid in the data register during a write operation. This is valid only when MTL_DBG_STS[PKTSTATE] is 2'b10 (EOP), and the transmit FIFO or receive FIFO is selected.<br>00b - Byte 0 valid<br>01b - Byte 0 and byte 1 valid<br>10b - Byte 0, byte 1, and byte 2 valid<br>11b - All four bytes valid |
| 1<br>DBGMOD | Debug Mode Access to FIFO<br>Indicates the status of Debug mode access to FIFO.<br>If this field is 1, it indicates that the current access to FIFO is read, write, and debug. In this mode, the following access types are allowed:<br>• Read and write access to transmit FIFO, TSO FIFO, and receive FIFO<br>• Read access to transmit status FIFO<br>If this field is 0, it indicates that the current access to FIFO is slave access bypassing DMA. In this mode, the following access types are allowed:<br>• Write access to transmit FIFO<br>• Read access to receive FIFO and transmit status FIFO<br>0b - Disabled<br>1b - Enabled |
| 0<br>FDBGEN | FIFO Debug Access Enable<br>Indicates the status of FIFO debug access.<br>• If this field is 1, it indicates that the FIFO debug mode access is enabled.<br>• If the value of the field is 0, it indicates that FIFO can be accessed only through a master interface.<br>0b - Disabled<br>1b - Enabled |

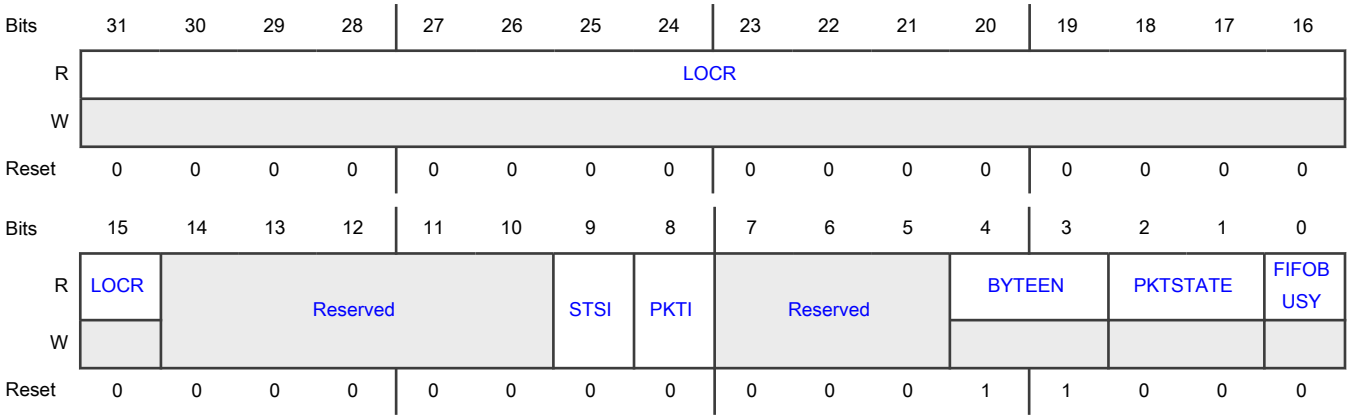
## 72.17.186 MTL Debug Status (MTL_DBG_STS)

### Offset

| Register | Offset |
|---|---|
| MTL_DBG_STS | C0Ch |

### Function
Contains the status of FIFO debug access.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-15<br>LOCR | Remaining Locations In FIFO<br>Indicates the remaining locations in FIFO. For Slave Access mode, the field indicates the space available in the selected FIFO, and for Debug Access mode, this field contains the write or read pointer value of the selected FIFO during the write or read operation, respectively.<br>Reset: In single transmit queue configurations, (DWC_EQOS_TXFIFO_SIZE/(DWC_EQOS_DATAWIDTH/8)), otherwise 0000h. |
| 14-10<br>— | Reserved |
| 9<br>STSI | Transmit Status Available Interrupt Status<br>Indicates whether the transmit status available interrupt status is detected.<br>If this field is 1, it indicates that Slave mode transmit packet is transmitted, and the status is available in transmit status FIFO. The field resets when you write 1 to it.<br>0b - Not detected<br>1b - Detected |
| 8 | Receive Packet Available Interrupt Status |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| PKTI | Indicates whether the receive packet available interrupt status is detected.<br>If this field is 1, it indicates that the MAC layer has written the EOP of the received packet to the receive FIFO. The field resets when you write 1 to it.<br>0b - Not detected<br>1b - Detected |
| 7-5<br>— | Reserved |
| 4-3<br>BYTEEN | Byte Enables<br>Indicates the number of data bytes valid in the data register during a read operation. This is valid only when MTL_DBG_STS[PKTSTATE] is 2'b10 (EOP), and the transmit FIFO or receive FIFO is selected.<br>00b - Byte 0 valid<br>01b - Byte 0 and byte 1 valid<br>10b - Byte 0, byte 1, and byte 2 valid<br>11b - All four bytes valid |
| 2-1<br>PKTSTATE | Encoded Packet State<br>Is used to get the control or status information of the selected FIFO.<br>These are the values related to the transmit FIFO.<br>• 00: Packet data<br>• 01: Control word<br>• 10: SOP data<br>• 11: EOP data<br>These are the values related to the receive FIFO.<br>• 00: Packet data<br>• 01: Normal status<br>• 10: Last status<br>• 11: EOP<br>This field is applicable only for transmit and receive FIFOs during a read operation.<br>00b - Packet data<br>01b - Control word/normal status<br>10b - SOP data/last status<br>11b - EOP data/EOP |
| 0<br>FIFOBUSY | FIFO Busy<br>Indicates whether the FIFO busy status is detected. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | If this field is 1, it indicates that a FIFO operation is in progress in MAC and the content of the following is invalid:<br>• All other fields of this register<br>• All the fields of MTL FIFO Debug Data (MTL_FIFO_Debug_Data)<br>0b - Not detected<br>1b - Detected |

## 72.17.187 MTL FIFO Debug Data (MTL_FIFO_Debug_Data)

### Offset

| Register | Offset |
|---|---|
| MTL_FIFO_Debug_Data | C10h |

### Function
Contains the data to be written to or read from the FIFOs.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FDBGDATA | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | FDBGDATA | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-0<br>FDBGDATA | FIFO Debug Data<br>Contains the data to be written to the transmit FIFO, receive FIFO, or TSO FIFO during a debug or slave access write operations. The field contains the data read from the transmit FIFO, receive FIFO, TSO FIFO, or transmit status FIFO during debug or slave access read operations. |

## 72.17.188 MTL Interrupt Status (MTL_Interrupt_Status)

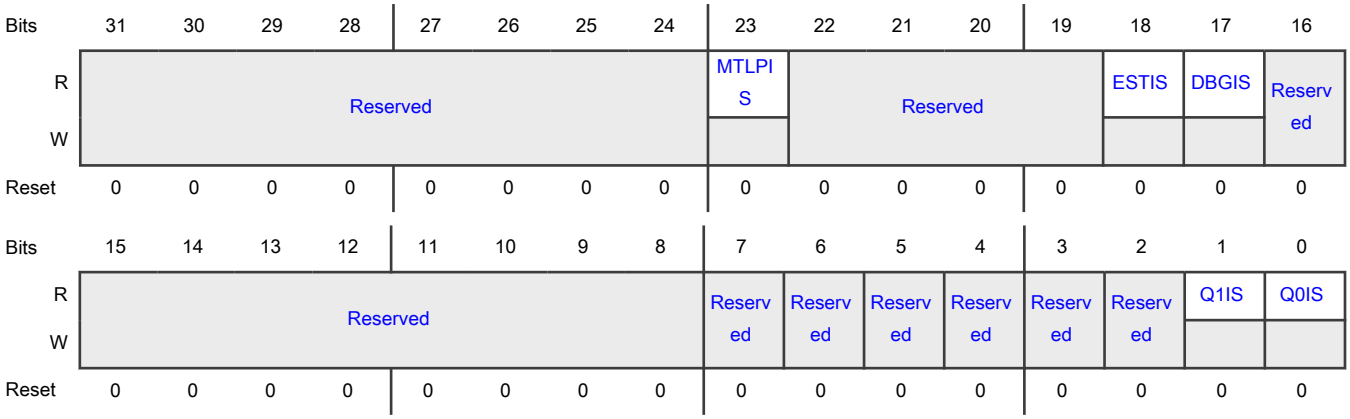
### Offset

| Register | Offset |
|---|---|
| MTL_Interrupt_Status | C20h |

### Function

Determines the interrupt status of MAC and MTL queues. The software driver (application) reads this register during an interrupt service routine or polling to perform this function.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23<br>MTLPIS | MTL Receive Parser Interrupt Status<br>Indicates whether the MTL receive parser interrupt status is detected.<br>This field indicates an interrupt from the receive parser block. To reset this field, the application must read MTL Rx Parser Interrupt Control Status (MTL_RXP_Interrupt_Control_Status) to get the exact cause of the interrupt and clear its source.<br>0b - Not detected<br>1b - Detected |
| 22-19<br>— | Reserved |
| 18<br>ESTIS | EST (TAS- 802.1Qbv) Interrupt Status<br>Indicates whether the EST (TAS- 802.1Qbv) interrupt status is detected. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | This field indicates an interrupt event during the operation of 802.1Qbv. To reset the field, the application must clear the error or event that caused the interrupt.<br>0b - Not detected<br>1b - Detected |
| 17<br>DBGIS | Debug Interrupt Status<br>Indicates whether the debug interrupt status is detected.<br>This field indicates an interrupt event during the slave access. To reset the field, the application must read MTL Debug Status (MTL_DBG_STS) to get the exact cause of the interrupt and clear its source.<br>0b - Not detected<br>1b - Detected |
| 16<br>— | Reserved |
| 15-8<br>— | Reserved |
| 7<br>— | Reserved |
| 6<br>— | Reserved |
| 5<br>— | Reserved |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>— | Reserved |
| 1<br>Q1IS | Queue 1 Interrupt Status<br>Indicates whether the queue 1 interrupt status is detected.<br>To reset this field, the application must read MTL Queue 1 Interrupt Control Status (MTL_Q1_Interrupt_Control_Status) to get the exact cause of the interrupt and clear its source.<br>0b - Not detected |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Detected |
| 0<br>Q0IS | Queue 0 Interrupt Status<br>Indicates whether the queue 0 interrupt status is detected.<br>To reset this field, the application must read MTL Queue 0 Interrupt Control Status (MTL_Q0_Interrupt_Control_Status) to get the exact cause of the interrupt and clear its source.<br>0b - Not detected<br>1b - Detected |

## 72.17.189 MTL Receive Queue DMA Map 0 (MTL_RxQ_DMA_Map0)

### Offset

| Register | Offset |
|---|---|
| MTL_RxQ_DMA_Map0 | C30h |

### Function
Reserves in EQOS-CORE and EQOS-MTL configurations.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | Reserved | Reserved | | | Reserved | Reserved | | | Reserved | Reserved | | | Reserved |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | Q1DDMACH | Reserved | | | Q1MDMACH | Reserved | | | Q0DDMACH | Reserved | | | Q0MDMACH |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved. |
| 28<br>— | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 27-25<br>— | Reserved. |
| 24<br>— | Reserved. |
| 23-21<br>— | Reserved. |
| 20<br>— | Reserved. |
| 19-17<br>— | Reserved. |
| 16<br>— | Reserved. |
| 15-13<br>— | Reserved. |
| 12<br>Q1DDMACH | Queue 1 Enabled for DA-based DMA Channel Selection<br>Enables or disables queue 1 for DA-based DMA channel selection.<br>If this field is 1, it indicates that the packets received in queue 1 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address.<br>When this field becomes 0, it indicates that the packets received in queue 1 are routed to the DMA Channel programmed in the MTL_RxQ_DMA_Map0[Q1MDMACH] (bits [10:8]).<br>0b - Disabled<br>1b - Enabled |
| 11-9<br>— | Reserved. |
| 8<br>Q1MDMACH | Queue 1 Mapped to DMA Channel<br>Controls the routing of the received packet in Queue 1 to the DMA channel:<br>000b - DMA Channel 0<br>001b - DMA Channel 1<br>010b - DMA Channel 2<br>011b - DMA Channel 3 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 100b - DMA Channel 4<br>101b - DMA Channel 5<br>110b - DMA Channel 6<br>111b - DMA Channel 7<br>This field is valid when the MTL_RxQ_DMA_Map0[Q1DDMACH] resets.<br>The field's width depends on the number of RX DMA channels and in some configurations all the values may not be valid . For example, if two RX DMA channels are selected, then only 000b and 001b are valid, the other bits are reserved. |
| 7-5<br>— | Reserved. |
| 4<br>Q0DDMACH | Queue 0 Enabled for DA-based DMA Channel Selection<br>Enables or disables queue 0 for DA-based DMA channel selection.<br>If this field is 1, it indicates that the packets received in queue 0 are routed to a particular DMA channel as decided in the MAC Receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address.<br>When this field becomes 0, it indicate that the packets received in queue 0 are routed to the DMA Channel programmed in the MTL_RxQ_DMA_Map0[Q0MDMACH].<br>0b - Disable<br>1b - Enable |
| 3-1<br>— | Reserved. |
| 0<br>Q0MDMACH | Queue 0 Mapped to DMA Channel<br>Controls the routing of the packet received in queue 0 to the DMA channel:<br>000b - DMA channel 0<br>001b - DMA channel 1<br>010b - DMA channel 2<br>011b - DMA channel 3<br>100b - DMA channel 4<br>101b - DMA channel 5<br>110b - DMA channel 6<br>111b - DMA channel 7<br>This field is valid when the MTL_RxQ_DMA_Map0[Q0DDMACH] resets. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | The field's width depends on the number of RX DMA channels in some configurations all the values may not be valid. For example, if the number of RX DMA channels selected is 2, only 000 and 001 bits are valid, the other bits are reserved. |

## 72.17.190 MTL TBS Control (MTL_TBS_CTRL)

### Offset

| Register | Offset |
|---|---|
| MTL_TBS_CTRL | C40h |

### Function

Controls the operation of time based scheduling.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LEOS | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | LEOS | | | | | | | | Reserved | LEGOS | | | Reserved | | LEOV | ESTM |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-8<br>LEOS | Launch Expiry Offset<br>Computes the launch expiry time.<br>To compute the launch expiry time the value in units of 256 nanoseconds is added to the launch time. The field value is valid only if MTL_TBS_CTRL[LEOV] is 1.<br>Max value: 999,999,999 nanosecond should be smaller than CTR-1 value when ESTM mode = 1, because this value is a modulo CTR value. |
| 7<br>— | Reserved. |
| 6-4 | Launch Expiry GSN Offset |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| LEGOS | Computes the Launch expiry time.<br>To compute the launch expiry time the number of GSN slots is to be added to the launch GSN. The field value is valid only if MTL_TBS_CTRL[LEOV] is 1. |
| 3-2<br>— | Reserved. |
| 1<br>LEOV | Launch Expiry Offset Valid<br>Indicates if MTL_TBS_CTRL[LEOS] is valid or invalid.<br>When this field is 1, it indicates that MTL_TBS_CTRL[LEOS] is valid.<br>When not 1, it indicates that the launch expiry Offset is not valid and the MTL must not check for launch expiry time.<br>0b - Invalid<br>1b - Valid |
| 0<br>ESTM | EST offset Mode<br>Enables or disables EST Offset mode.<br>If this field is 1, the launch time value used in Time Based Scheduling is interpreted as an EST offset value and is added to the BTR of the current list.<br>When this field becomes 0, the launch time value is used as an absolute value that must be compared with the system time [39:8].<br>0b - Disabled<br>1b - Enabled |

## 72.17.191 MTL EST Control (MTL_EST_Control)

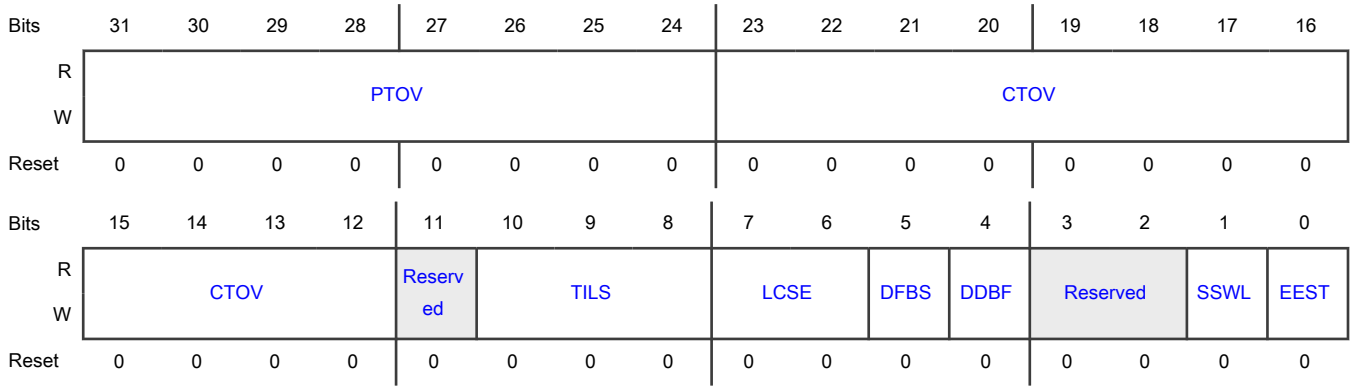
### Offset

| Register | Offset |
|---|---|
| MTL_EST_Control | C50h |

### Function

Controls the operation of enhancements to scheduled transmission (IEEE802.1Qbv).

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>PTOV | PTP Time Offset Value<br>Indicates the PTP time offset value.<br>You must multiply 6 to the value of PTP clock period in nanoseconds. At the beginning of the installation of a new GCL this value avoid transmission overruns. |
| 23-12<br>CTOV | Current Time Offset Value<br>Provides a 12 bit time offset value in nano second which is added to the current time to compensate for all the implementation pipeline delays such as the CDC sync delay, buffering delays, data path delays and so on. This offset ensures that the impact of gate controls is visible on the line exactly at the pre-determined schedule (or as close to the schedule as possible). |
| 11<br>— | Reserved. |
| 10-8<br>TILS | Time Interval Left Shift Amount<br>Provides the left shift amount for the programmed time interval values used in the gate control lists.<br>000b - No left shift needed (equal to x1ns)<br>001b - Left shift TI by 1 bit (equal to x2ns)<br>010b - Left shift TI by 2 bits (equal to x4ns)<br>-<br>-<br>100b - Left shift TI by 7 bits (equal to x128ns)<br>On the basis of configuration you must consider one or more bits of this field as reserved or read-only. |
| 7-6<br>LCSE | Loop Count to Report Scheduling Error<br>Provides programmable number of GCL list iterations before reporting an HLBS error which is defined in MTL_EST_Status. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 00b - 4 iterations<br>01b - 8 iterations<br>10b - 16 iterations<br>11b - 32 iterations |
| 5<br>DFBS | Drop Frames Causing Scheduling Error<br>Provides the status of the frames causing scheduling error.<br>If this field is 1 then the frames report to cause an HOL blocking because the MTL_EST_Status[HLBS] is not scheduled after 4,8,16,and 32 (on the basis of MTL_EST_Control[LCSE]) GCL iterations are dropped.<br>0b - Do not drop frames<br>1b - Drop frames |
| 4<br>DDBF | Do not Drop Frames during Frame Size Error<br>Provides status of frames during Frame Size Error.<br>If this field is 1, it indicates that the frames are not dropped during head-of-line blocking because of the frame size error (MTL_EST_Status[HLBF]).<br>0b - Drop frames during frame size error<br>1b - Do not Drop frames during frame size error |
| 3-2<br>— | Reserved. |
| 1<br>SSWL | Switch to Software Owned List<br>If this field is 1, it indicates that the software has programmed that list that it currently owns (SWOL) and the hardware should switch to the new list based on the new BTR. Hardware clears this bit when the switch to the SWOL happens to indicate the completion of the switch or when an BTR error (BTRE in Status register) is set. When BTRE is set this bit is cleared but SWOL is not updated as the switch was not successful.<br>Access restriction apply to this field that clears on read and automatically becomes 1 on an internal event occurrence.<br>Enables or disables the switch to the software owned list.<br>0b - Disabled<br>1b - Enabled |
| 0<br>EEST | Enable EST<br>Enables or disables EST.<br>If the field becomes 0, it indicate that the gate control list processing is halted and all gates are assumed to be in open state. The field must be 1 for the hardware to start processing the gate control lists. During the toggle from 0 to 1, when the MTL_EST_Control[SSWL] is 1, the gate control list processing starts.<br>During the configuration when DWC_EQOS_ASP_ECC is selected, the hardware resets this field and disables the EST function if any uncorrectable error is detected in the EST memory . |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |

## 72.17.192 MTL EST Status (MTL_EST_Status)

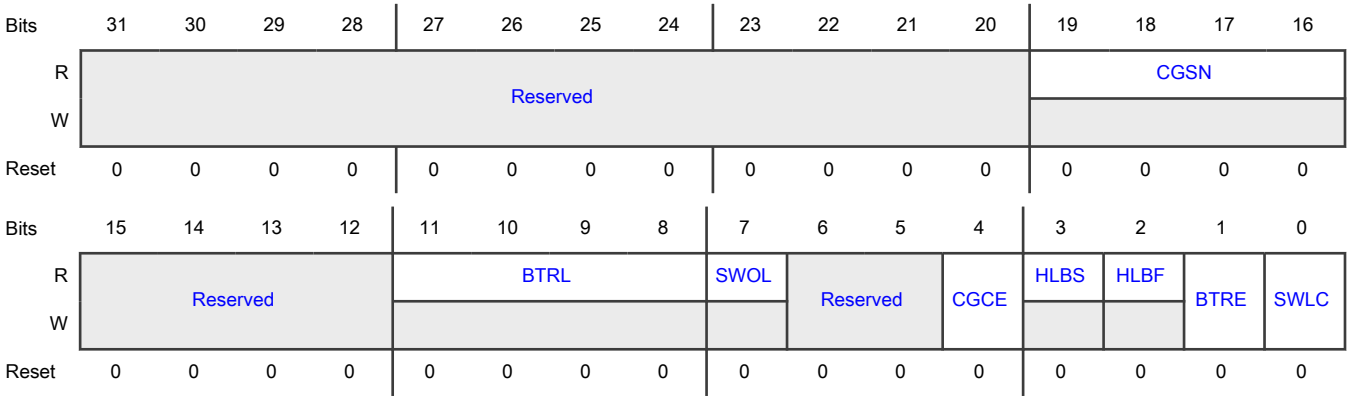
### Offset

| Register | Offset |
|---|---|
| MTL_EST_Status | C58h |

### Function

Provides status of enhancements to scheduled transmission (IEEE802.1Qbv).

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved. |
| 19-16<br>CGSN | Current GCL Slot Number<br>Indicates the slot number of the GCL list. Slot number is a modulo 16 count of the GCL List loops executed so far.<br>Even if a new GCL list is installed, the count is incremental. |
| 15-12<br>— | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 11-8<br>BTRL | BTR Error Loop Count<br>Provides the minimum count (N) for which the equation Current Time =< New BTR + (N * New Cycle Time) becomes true. N = "1111" indicates the iterations exceeded the value of 8 and the hardware was not able to update New BTR to be equal to or greater than Current Time. Software intervention is needed to update the New BTR. Value cleared when BTRE field of this register is cleared. |
| 7<br>SWOL | S/W owned list<br>When '0' indicates Gate control list number "0" is owned by software and when "1" indicates the Gate Control list "1" is owned by the software. Any reads/writes by the software (using indirect access via GCL_Control) is directed to the list indicated by this value by default. The inverse of this value is treated as HWOL.<br>R/W operations performed by hardware are directed to the list pointed by HWOL by default.<br>0b - Gate control list number "0" is owned by software<br>1b - Gate control list number "1" is owned by software |
| 6-5<br>— | Reserved. |
| 4<br>CGCE | Constant Gate Control Error<br>This error occurs when the list length (LLR) is 1 and the Cycle Time (CTR) is less than or equal to the programmed Time Interval (TI) value after the optional Left Shifting. The above programming implies Gates are either always Closed or always Open based on the Gate Control values; the same effect can be achieved by other simpler (non TSN) programming mechanisms. Since the implementation does not support such a programming an error is reported.<br>Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.<br>0b - Constant Gate Control Error not detected<br>1b - Constant Gate Control Error detected |
| 3<br>HLBS | Head-Of-Line Blocking due to Scheduling<br>Set when the frame is not able to win arbitration and get scheduled even after 4 iterations of the GCL. Indicates to software a potential programming error. The one hot encoded values of the Queue Numbers that are not able to make progress are indicated in the MTL_EST_Sch_Error register. Bit cleared when MTL_EST_Sch_Error register is all zeros.<br>0b - Head-Of-Line Blocking due to Scheduling not detected<br>1b - Head-Of-Line Blocking due to Scheduling detected |
| 2<br>HLBF | Head-Of-Line Blocking due to Frame Size<br>Set when HOL Blocking is noticed on one or more Queues as a result of none of the Time Intervals of gate open in the GCL being greater than or equal to the duration needed for frame size (or frame fragment size when preemption is enabled) transmission. The one hot encoded Queue numbers that are experiencing HLBF are indicated in the MTL_EST_Frm_Size_Error register. Additionally, the first Queue number that experienced HLBF along with the frame size is captured in MTL_EST_Frm_Size_Capture register. Bit cleared when MTL_EST_Frame_Size_ Error register is all zeros. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Head-Of-Line Blocking due to Frame Size not detected<br>1b - Head-Of-Line Blocking due to Frame Size detected |
| 1<br>BTRE | BTR Error<br>Indicates whether the BTR error is detected.<br>When this field is 1, it indicates a programming error in the BTR of SWOL where the programmed value is less than the current time. If the BTRL = "1111", SWOL is not updated and software must reprogram the BTR to a value greater than current time and then set SSWL to reinitiate the switch to SWOL. Else if the value of BTRL < "1111", SWOL is updated and this field indicates the number of iterations (of + CycleTime) taken by hardware to update the BTR to a value greater than Current Time.<br>Access restrictions apply to this field. It sets automatically to 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - BTR Error not detected<br>1b - BTR Error detected |
| 0<br>SWLC | Switch to Software Owned List Complete<br>Indicates whether the switch to software owned list complete is detected.<br>When the field is 1, it indicates that the hardware has successfully switched to the SWOL, and updated the MTL_EST_Status[SWOL] to that effect.<br>This field is 0 when MTL_EST_Control[SSWL] transitions from 0 to 1, or on a software write.<br>Access restrictions apply to this field. It clears automatically and writing 0 to it has no effect.<br>0b - Not detected<br>1b - Detected |

## 72.17.193 MTL EST Scheduling Error (MTL_EST_Sch_Error)

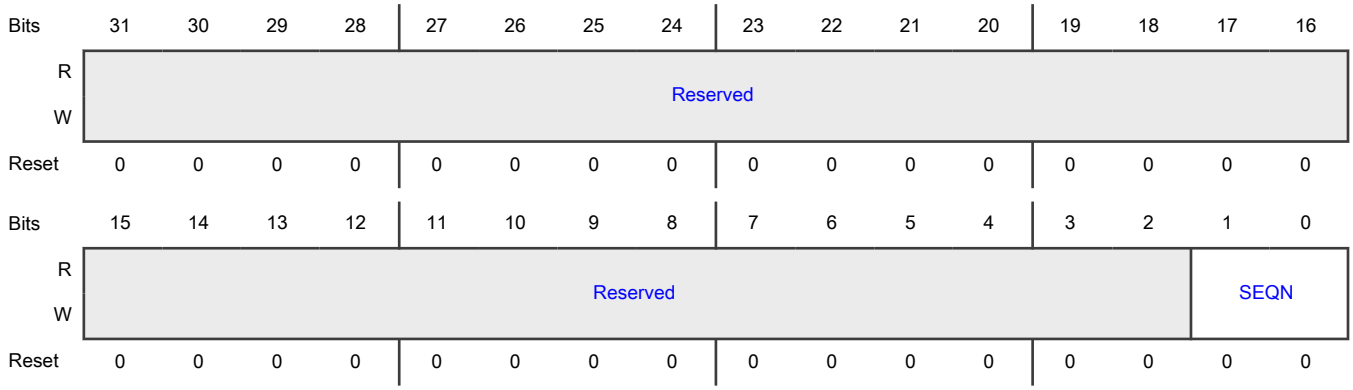
### Offset

| Register | Offset |
|---|---|
| MTL_EST_Sch_Error | C60h |

### Function

Provides the one hot encoded queue numbers that have the scheduling related error (timeout).

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved. |
| 1-0<br>SEQN | Schedule Error Queue Number<br>Provides the one hot encoded queue numbers that have experienced error or timeout described in MTL_EST_Status[HLBS].<br>Access restriction apply to this field. It automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 to it has no effect. |

## 72.17.194 MTL EST Frame Size Error (MTL_EST_Frm_Size_Error)

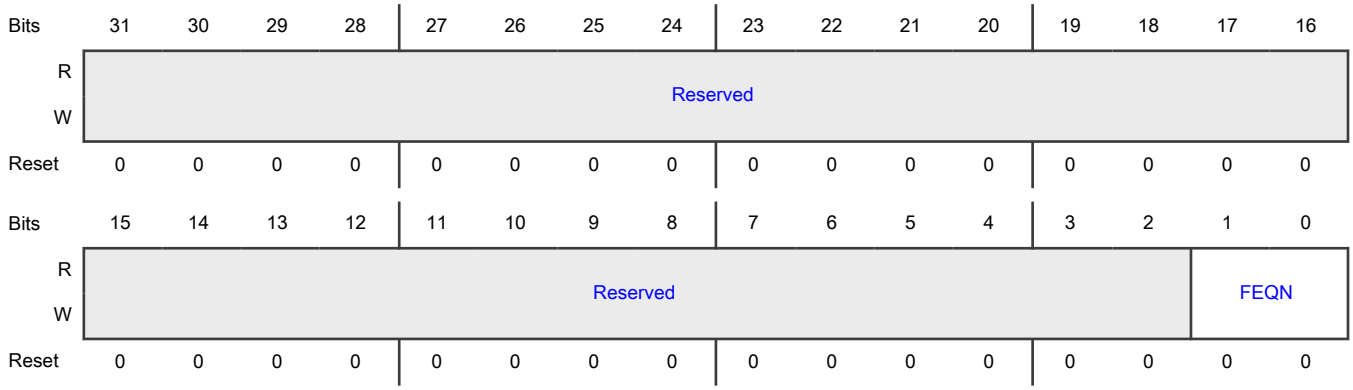
### Offset

| Register | Offset |
|---|---|
| MTL_EST_Frm_Size_Error | C64h |

### Function

Provides the one hot encoded queue numbers that have the frame Size related error.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved. |
| 1-0<br>FEQN | Frame Size Error Queue Number<br>Provides the one hot encoded queue Nnumbers that have experienced error described in MTL_EST_Status[HLBF].<br>Access restriction apply to this field. It automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 to it has no effect. |

## 72.17.195 MTL EST Frame Size Capture (MTL_EST_Frm_Size_Capture)

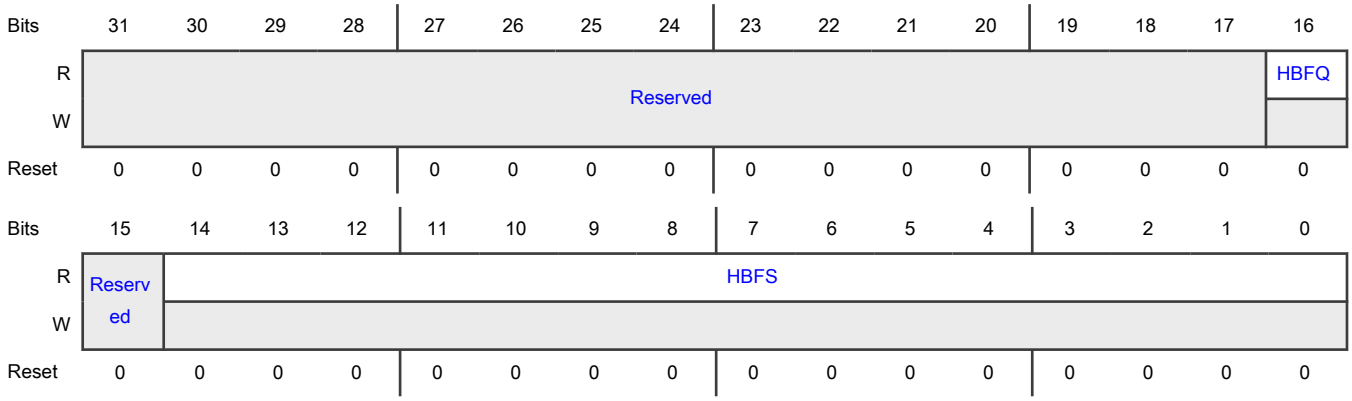
### Offset

| Register | Offset |
|---|---|
| MTL_EST_Frm_Size_Capture | C68h |

### Function

Captures the frame Size and queue number of the first occurrence of the frame Size related error. When you write 0, it captures the data of immediate next occurrence of a similar error.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-17<br>— | Reserved. |
| 16<br>HBFQ | Queue Number of HLBF<br>Captures the binary value of the first queue (number) which experiences the HLBF error (see MTL_EST_Status[HLBF]). Any subsequent queue errors similar to HLBF errors does not alter the value after it is written. After this field is 0, it captures the queue number of the next occurring HLBF error.<br>Field's width is based on the number of transit queues configured; remaining fields are read-only. This field clears when MTL_EST_Frm_Size_Error is all zeros. |
| 15<br>— | Reserved. |
| 14-0<br>HBFS | Frame Size of HLBF<br>Captures the frame Size of the dropped frame related to queue number indicated in MTL_EST_Frm_Size_Capture[HBFQ].<br>If this field is zero, then this register must be considered invalid.<br>This field clears when MTL_EST_Frm_Size_Error is all zeros. |

## 72.17.196 MTL EST Interrupt Enable (MTL_EST_Intr_Enable)

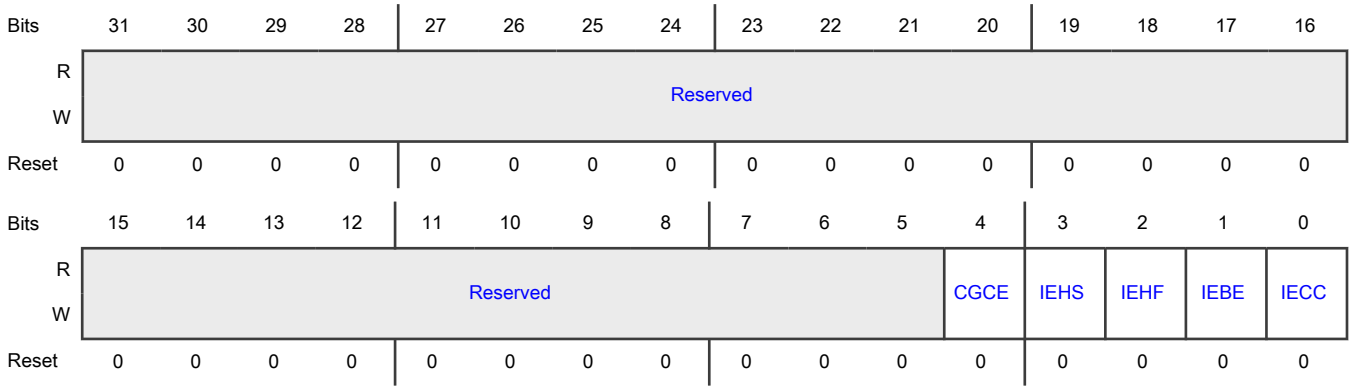
### Offset

| Register | Offset |
|---|---|
| MTL_EST_Intr_Enable | C70h |

### Function

Implements the interrupt enable fields for the various events that generate an interrupt. Bit positions have a 1 to 1 correlation with the status bit positions in MTL_ETS_Status register.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-5<br>— | Reserved. |
| 4<br>CGCE | Interrupt Enable for CGCE<br>Indicates whether an interrupt for MTL_EST_Intr_Enable[CGCE] is enabled.<br>When this field is 1, it generates an interrupt when the constant gate control error occurs and is indicated in the status. When this field becomes 0, this event does not generate an interrupt<br>0b - Disabled<br>1b - Enabled |
| 3<br>IEHS | Interrupt Enable for HLBS<br>Indicates whether an interrupt for MTL_EST_Status[HLBS] is enabled.<br>When this field is 1, it generates an interrupt when the head-of-line blocking due to scheduling issue occurs and is indicated in the status. When this field becomes 0, this event does not generate an interrupt.<br>0b - Disabled<br>1b - Enabled |
| 2<br>IEHF | Interrupt Enable for HLBF<br>Indicates whether an interrupt for MTL_EST_Status[HLBF] is enabled.<br>When this field is 1, it generates an interrupt when the head-of-line blocking due to frame size error occurs and is indicated in the status. When this field becomes 0, this event does not generate an interrupt.<br>0b - Disabled<br>1b - Enabled |
| 1<br>IEBE | Interrupt Enable for BTR Error<br>Indicates whether an interrupt for BTR error is enabled. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When this field is 1, it generates an interrupt when the BTR error occurs and is indicated in the status. When this field becomes 0, this event does not generate an interrupt.<br>0b - Disabled<br>1b - Enabled |
| 0<br>IECC | Interrupt Enable for Switch List<br>Indicates whether the interrupt for switch list is enabled.<br>When this field is 1, it generates an interrupt when the configuration change is successful and the hardware switches to the new list. When this field becomes 0, this event does not generate an interrupt.<br>0b - Disabled<br>1b - Enabled |

## 72.17.197 MTL EST GCL Control (MTL_EST_GCL_Control)

### Offset

| Register | Offset |
|---|---|
| MTL_EST_GCL_Control | C80h |

### Function

Provides the control information for reading and writing to the gate control lists.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | ESTEIEC | | ESTEIEE | ERR0 | Reserved | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ADDR | | | | | | | | Reserved | | DBGB | DBGM | Reserved | GCRR | R1W0 | SRWO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-24 | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 23-22<br>ESTEIEC | ECC Inject Error Control for EST Memory<br>Provides the status of inject error control for EST memory.<br>When MTL_EST_GCL_Control[ESTEIEE] = 1, it indicates that on the basis of the value encoded in this field following errors are inserted.<br>This field is valid only if you select DWC_EQOS_ASP_ECC feature during the configuration, otherwise it is reserved.<br>00b - Insert 1 bit error<br>01b - Insert 2 bit errors<br>10b - Insert 3 bit errors<br>11b - Insert 1 bit error in address field |
| 21<br>ESTEIEE | EST ECC Inject Error Enable<br>Indicates whether the EST ECC inject error is enabled.<br>If this field along with MTL_EST_Control[EEST] = 1, it enables the ECC error injection feature.<br>When this field becomes 0, it disables the ECC error injection feature.<br>0b - Disabled<br>1b - Enabled |
| 20<br>ERR0 | If this field is 1, it indicates that when the software writes to GCL the last write operation was aborted and when MTL_EST_Control[SSWL] is 1, GCL registers are prohibited.<br>Access restriction apply to this field and automatically becomes 1 on an internal event occurrence. Writing 1 to this field clears it and writing 0 to this field has no effect.<br>0b - ERR0 is disabled<br>1b - ERR1 is enabled |
| 19-16<br>— | Reserved. |
| 15-8<br>ADDR | Gate Control List Address: (GCLA when GCRR is "0").<br>Provides the address (row number) of the gate control list at which the R/W operation has to be performed. By default the gate control list to which MTL_EST_Status[SWOL] points is selected for read and write. If MTL_EST_GCL_Control[DBGM] is 1, it indicates that MTL_EST_GCL_Control[DBGB] gives Debug mode access to read and write. The field's width depends on DWC_EQOS_EST_DEP and the unused bits must be treated as read only.<br>Gate Control list Related Registers Address: (GCRA when GCRR is "1").<br>By default the GCL related register set pointed by MTL_EST_Status[SWOL] is selected for read or write. If MTL_EST_GCL_Control[DBGM], it indicates that MTL_EST_GCL_Control[DBGB] gives Debug mode access to read write. Lower 3 bits are only used in this mode, higher order bits are treated as don't cares. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 000b - BTR Low (31:0)<br>001b - BTR High (63:31)<br>010b - CTR Low (31:0)<br>011b - CTR High (39:32)<br>100b - TER (31:0)<br>101b - LLR (n:0) (where n is (log{DWC_EQOS_EST_DEP} / log2))<br>Others - Reserved |
| 7-6<br>— | Reserved. |
| 5<br>DBGB | Debug Mode Bank Select<br>Indicates whether the read and write in Debug mode is directed to bank 0 or bank 1.<br>When this field is 0, it indicates that read or write in Debug mode must be directed to bank 0 (GCL0 and corresponding Time related registers). When this field is 1, it indicates that read or write in Debug mode should be directed to bank 1 (GCL1 and corresponding Time related registers). When MTL_EST_GCL_Control[DBGM] is 1 you can use this value and overrides with the MTL_EST_Status[SWOL] value.<br>0b - Directed to bank 0<br>1b - Directed to bank 1 |
| 4<br>DBGM | Debug Mode<br>Enables or disables Debug mode.<br>When this field is 1, it indicates that the read and write is in Debug mode where the MTL_EST_GCL_Control[DBGB] value explicitly provides the memory bank (for GCL and time related registers). When this field is 0, it indicates that MTL_EST_Status[SWOL] determines which bank to use.<br>0b - Disabled<br>1b - Enabled |
| 3<br>— | Reserved. |
| 2<br>GCRR | Gate Control Related Registers<br>Enables or disables gate control related registers.<br>When this field is 1, it indicates that the read or write access is for the GCL related registers (BTR, CTR, TER, LLR) for which GCRA provides the address. When this field is 0, it indicates that read and write must be directed to GCL from the address which GCLA provides.<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 1<br>R1W0 | Read '1', Write '0'<br>Indicates whether this field performs read operation or write operation.<br>When this field is 1 it indicates that it performs a read operation.<br>When this field is 0 it indicates that it performs a write operation.<br>0b - Write operation<br>1b - Read operation |
| 0<br>SRWO | Start Read/Write Operation<br>Indicates whether the read/write operation is enabled.<br>When this field is 1, it indicates that a read/write operation has started and is in progress.<br>When hardware resets this field and MTL_EST_GCL_Control[ERR0] = 1 , it indicates that the read/write operation has completed or an error has occurred.<br>Reads: When this field becomes 0, it indicates that MTL_EST_GCL_Data reads data.<br>Writes: Before writing 1 to this field, you must program MTL_EST_GCL_Data with write data.<br>Access restrictions apply to this field. It clears automatically. Writing 1 sets this field and writing 0 to it has no effect.<br>0b - Disabled<br>1b - Enabled |

## 72.17.198 MTL EST GCL Data (MTL_EST_GCL_Data)

### Offset

| Register | Offset |
|---|---|
| MTL_EST_GCL_Data | C84h |

### Function

Holds the read data or write data in case of reads and writes respectively.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>GCD | Gate Control Data<br>Provides the data corresponding to the address selected in the MTL_EST_GCL_Control. This field is used for both read and write operations. |

## 72.17.199 MTL FPE Control Status (MTL_FPE_CTRL_STS)

### Offset

| Register | Offset |
|---|---|
| MTL_FPE_CTRL_STS | C90h |

### Function

Controls the operation of, and provides status for frame preemption (IEEE802.1Qbu/802.3br).

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | PEC | | Reserved | | | | | | AFSZ | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Fields**

| Field | Function |
|---|---|
| 31-29<br>— | Reserved. |
| 28<br>HRS | Hold/Release Status<br>Indicates whether a set-and-release-MAC operation which was last executed and the pMAC is in release state or in hold state.<br>0b - Release state<br>1b - Hold state |
| 27-16<br>— | Reserved. |
| 15-10<br>— | Reserved. |
| 9-8<br>PEC | Preemption Classification<br>When this field is 1, it indicates that you must classify the corresponding queue as preemptable.<br>When this field is 0, it indicates that you must classify the queue as express.<br>When set indicates the corresponding Queue must be classified as preemptable, when '0' Queue is classified as express. When both EST (Qbv) and Preemption are enabled, Queue-0 is always assumed to be preemptable. When you enables EST (Qbv), the preemptable queues are always assumed to be in open state in gate control list. |
| 7-2<br>— | Reserved. |
| 1-0<br>AFSZ | Additional Fragment Size<br>Indicates that, in units of 64 bytes, the minimum number of bytes over 64 bytes required in non-final fragments of preempted frames. The minimum non-final fragment size is (AFSZ +1) * 64 bytes. |

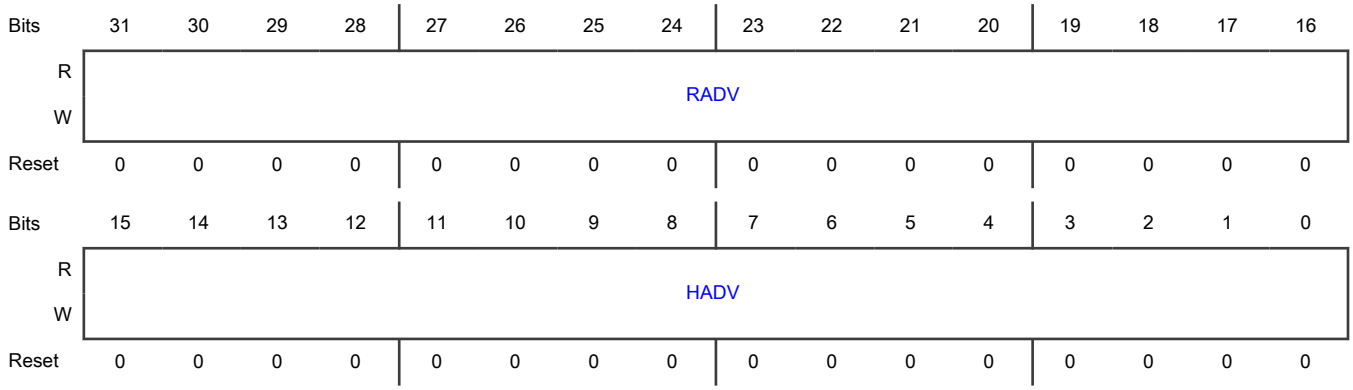
## 72.17.200 MTL FPE Advance (MTL_FPE_Advance)

**Offset**

| Register | Offset |
|---|---|
| MTL_FPE_Advance | C94h |

**Function**
Holds the hold and release advance time.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>RADV | Release Advance<br>Provides the maximum time in nanoseconds that can elapse between issuing a release to the MAC and the MAC being ready to resume transmission of preemptable frames, in the absence of there being any express frames available for transmission. |
| 15-0<br>HADV | Hold Advance<br>Provides the maximum time in nanoseconds that can elapse between issuing a hold to the MAC and the MAC ceasing to transmit any preemptable frame that is in the process of transmission or any preemptable frames that are queued for transmission. |

## 72.17.201 MTL Rx Parser Control Status (MTL_RXP_Control_Status)

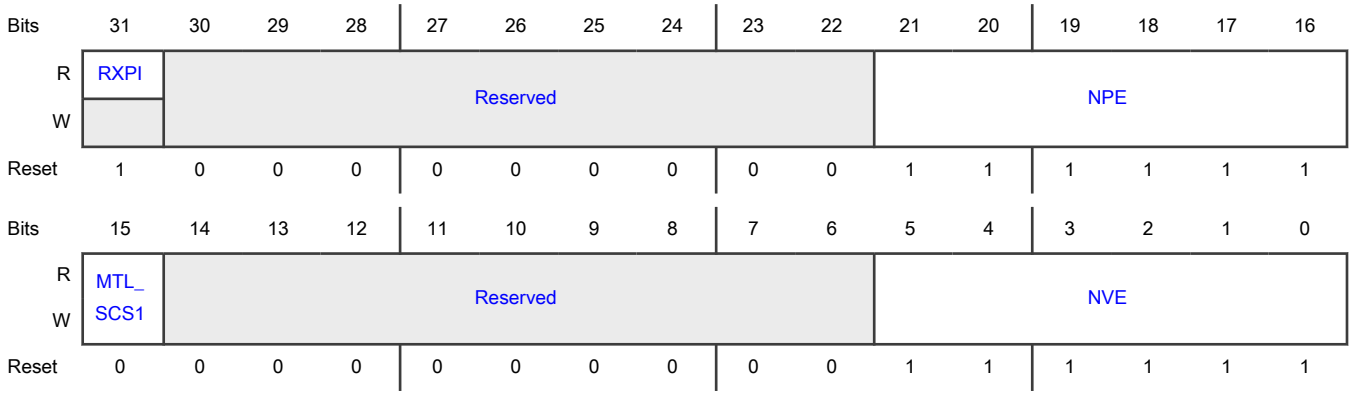
### Offset

| Register | Offset |
|---|---|
| MTL_RXP_Control_Status | CA0h |

### Function

Establishes the operating mode of receive parser and provides some status.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>RXPI | RX Parser in Idle State<br>Indicates whether the receive parser is in idle state.<br>When this field is 1, it indicates that the receive parser is in idle state and waiting for the processing of a new packet. When parser disables, you can use this field as a handshake with software and also the software can update the receive parser instruction table when you write 1 to this field.<br>0b - Not in Idle state<br>1b - Idle state |
| 30-22<br>— | Reserved. |
| 21-16<br>NPE | Number of parsable entries in the Instruction table<br>Indicates the number of parsable entries in the instruction memory. This is used in receive parser logic to detect programming error.<br>MTL_RXP_Interrupt_Control_Status[NPEOVIS] = 1, if the number of parsed entries for a packet is more than this entry. |
| 15<br>MTL_SCS1 | MTL_SCS1<br>Is reserved for NXP internal use.<br>The value of this field must always be 0 unless NXP instructs you otherwise. Writing 1 to any of the bits of this field might cause unexpected behavior in th IP. |
| 14-6<br>— | Reserved. |
| 5-0<br>NVE | Number Of Valid Entry Address Or Index In The Instruction Table<br>Indicates the number of valid entry address or index in the instruction memory. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | The maximum valid entry address is = NVE + 1 and the addresses or indices range from 0 to 32. Therefore, if the value of this field is 32, the number of valid entry address is 33. This value is used in receive parser logic to detect if any programming errors exist. When parsing, the module writes 1 to NVEOVIS if the number of the memory address is found to be more than the defined maximum valid entry address number.<br>**NOTE**<br>The minimum value of this field must be two. |

## 72.17.202 MTL Rx Parser Interrupt Control Status (MTL_RXP_Interrupt_Control_Status)

### Offset

| Register | Offset |
|---|---|
| MTL_RXP_Interrupt_Control_Status | CA4h |

### Function

Provides enable control for the interrupts and also provides interrupt status.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | PDRFIE | FOOVIE | NPEOVIE | NVEOVIE |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | PDRFIS | FOOVIS | NPEOVIS | NVEOVIS |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved. |
| 19<br>PDRFIE | Packet Drop due to RF Interrupt Enable<br>Indicates whether the packet drop due to RF interrupt is enabled.<br>If this field is 1, it enables the PDRFIS interrupt. When this field becomes 0, it disables the PDRFIS interrupt. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |
| 18<br>FOOVIE | Frame Offset Overflow Interrupt Enable<br>Indicates whether the frame offset overflow interrupt is enabled.<br>If this field is 1, it enables the FOOVIS interrupt.<br>When this field becomes 0, it disables the FOOVIS interrupt.<br>0b - Disabled<br>1b - Enabled |
| 17<br>NPEOVIE | Number of Parsable Entries Overflow Interrupt Enable<br>Enables or disables the number of parsable entries overflow interrupt.<br>If this field is 1, it enables the NPEOVIS interrupt.<br>When this field becomes 0, it disables the NPEOVIS interrupt.<br>0b - Disable<br>1b - Enable |
| 16<br>NVEOVIE | Number of Valid Entries Overflow Interrupt Enable<br>Enables or disables the number of valid entries overflow interrupt.<br>If this field is 1, it enables the NVEOVIS interrupt.<br>When this field becomes 0, it disables the NVEOVIS interrupt.<br>0b - Disable<br>1b - Enable |
| 15-4<br>— | Reserved. |
| 3<br>PDRFIS | Packet Dropped due to RF Interrupt Status<br>Indicates whether the packet dropped due to RF interrupt status is detected.<br>If the Rx Parser result says to drop the packet by setting RF=1 in the instruction memory, then this bit is set to 1.<br>This field clears when the application writes 1 to this field.<br>Access apply to this field. It automatically sets to 1 on an internal event occurrence. Writing 1 clears this field and writing 0 to it has no effect.<br>0b - Not detected<br>1b - Detected |
| 2 | Frame Offset Overflow Interrupt Status |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| FOOVIS | Indicates whether the frame offset overflow interrupt status is detected.<br>If this field is 1, it indicates that when parsing, the instruction table entry's frame offset is more than the EOF offset.<br>This field clears when the application writes 1 to this field.<br>Access apply to this field. It automatically sets to 1 on an internal event occurrence. Writing 1 clears this field and writing 0 to it has no effect.<br>0b - Not detected<br>1b - Detected |
| 1<br>NPEOVIS | Number of Parsable Entries Overflow Interrupt Status<br>Indicates whether the number of parsable entries overflow interrupt status is detected.<br>If this field is 1, it indicates that when parsing a packet the number of parsed entries are more than NPE[] (MTL_RXP_Control_Status[NPE]).<br>This field clears when the application writes 1 to this field.<br>Access apply to this field. It automatically sets to 1 on an internal event occurrence. Writing 1 clears this field and writing 0 to it has no effect.<br>0b - Not detected<br>1b - Detected |
| 0<br>NVEOVIS | Number of Valid Entry Address/Index Overflow Interrupt Status<br>Indicates whether the number of valid entry address or index overflow interrupt status is detected.<br>If this field is 1, it indicates that when parsing, the instruction address is more than MTL_RXP_Control_Status[NVE].<br>For example, when MTL_RXP_Control_Status[NVE] = 31, the maximum valid entry address/index is NVE+1 that is 32 (addresses/indices = 0 to 32, or 33 entries). MTL_RXP_Interrupt_Control_Status[NVEOVIS] = 1 when currently processed entry indicates that the next address is 33 or more, that is 34th or later entries.<br>This field clears when the application writes 1 to this field.<br>Access apply to this field. It automatically sets to 1 on an internal event occurrence. Writing 1 clears this field and writing 0 to it has no effect.<br>0b - Not detected<br>1b - Detected |

## 72.17.203 MTL Rx Parser Drop Count (MTL_RXP_Drop_Cnt)

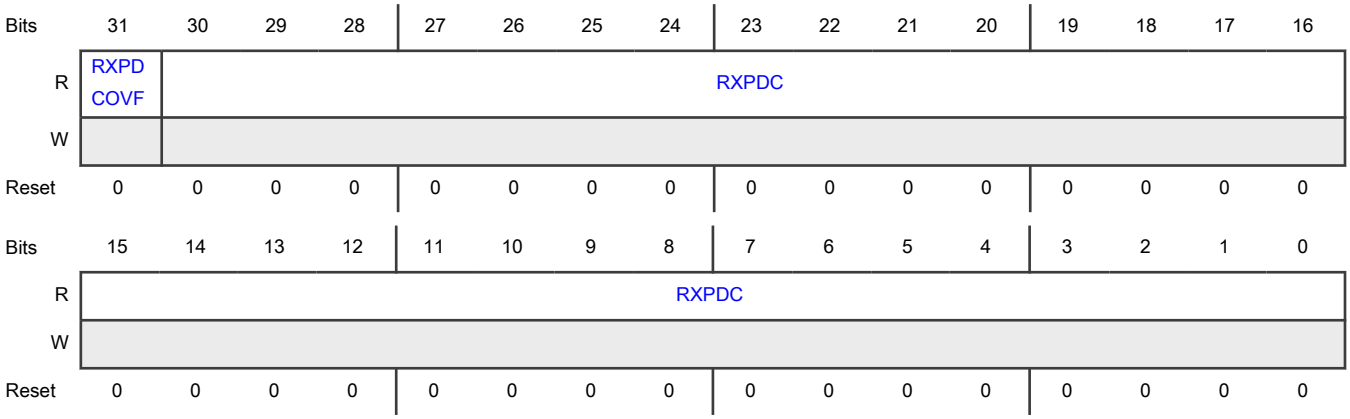
### Offset

| Register | Offset |
|---|---|
| MTL_RXP_Drop_Cnt | CA8h |

### Function

Provides the drop count of receive parser initiated drops.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>RXPDCOVF | Rx Parser Drop Counter Overflow Bit<br>Indicates whether the receive parser drop count overflow has occurred.<br>When this field is 1, it indicates that the MTL_RXP_Drop_Cnt[RXPDC] counter field has crossed the maximum limit.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not occurred<br>1b - Occurred |
| 30-0<br>RXPDC | Rx Parser Drop Count<br>Provides the receive parser drop count.<br>This 31-bit counter is implemented whenever a receive parser drops a packet due to RF = 1. The counter clears when you read the register. |

## 72.17.204 MTL Rx Parser Error Count (MTL_RXP_Error_Cnt)

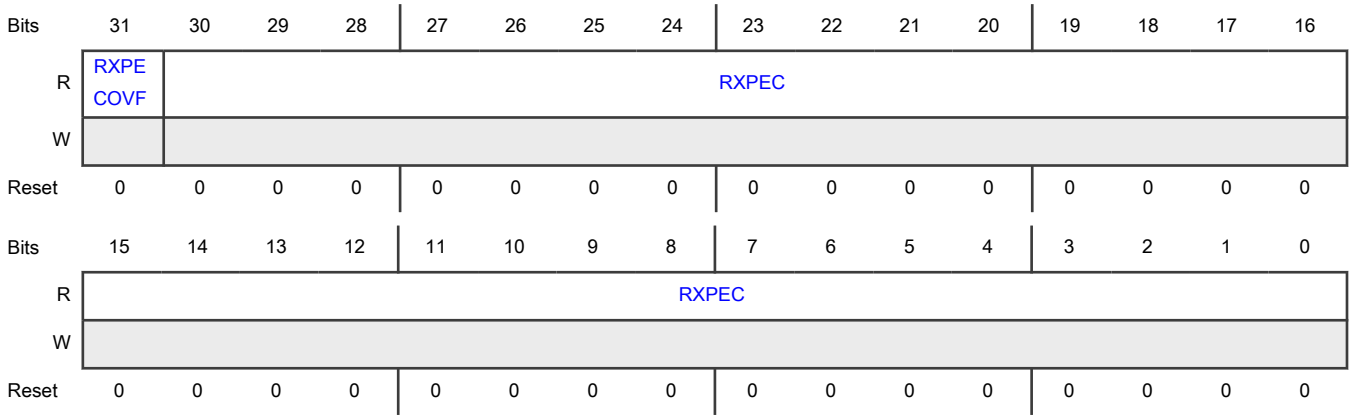
### Offset

| Register | Offset |
|---|---|
| MTL_RXP_Error_Cnt | CACh |

### Function

Provides the receive parser related error occurrence count.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>RXPECOVF | Rx Parser Error Counter Overflow Bit<br>Indicates whether the receive parser error counter overflow has occurred.<br>When this field is 1, it indicates that MTL_RXP_Error_Cnt[RXPEC] counter field has crossed the maximum limit.<br>Access restrictions apply to this field, which clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not occurred<br>1b - Occurred |
| 30-0<br>RXPEC | Rx Parser Error Count<br>Provides the receive parser error count.<br>This 31-bit counter is implemented whenever a receive parser encounters following error scenarios:<br>Entry address >= NVE[]<br>Number parsed entries >= NPE[]<br>Entry address > EOF data entry address<br>The counter clears when you read the register. |

## 72.17.205 MTL Rx Parser Indirect Access Control Status (MTL_RXP_Indirect_Acc_Control_Status)

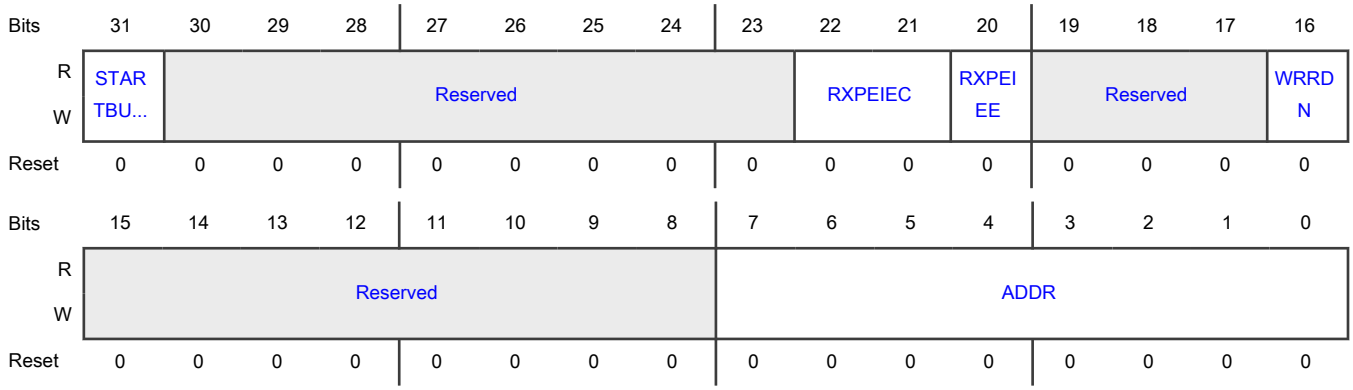
### Offset

| Register | Offset |
|---|---|
| MTL_RXP_Indirect_Acc_Control_Status | CB0h |

### Function

Provides the indirect access control and status for receive parser memory.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>STARTBUSY | FRP Instruction Table Access Busy<br>When this bit is set to 1 by the software then it indicates to start the Read/Write operation from/to the Rx Parser Memory.<br>Software should read this bit as 0 before issuing read or write request to access the Parser Memory Instructions.<br>This bit when set to 1 indicates that hardware is busy until its gets cleared by hardware and software should not issue any read or write operation.<br>0b - hardware not busy<br>1b - hardware is busy (Read/Write operation from/to the Rx Parser Memory) |
| 30-23<br>— | Reserved. |
| 22-21<br>RXPEIEC | ECC Inject Error Control for Rx Parser Memory<br>Controls the ECC inject error for receive parser memory.<br>When MTL_RXP_Indirect_Acc_Control_Status[RXPEIEE] = 1, it indicates that on the basis of the value encoded in this field following errors are inserted:<br>00b - Insert 1 bit error<br>01b - Insert 2 bit errors<br>10b - Insert 3 bit errors<br>11b - Insert 1 bit error in address field |
| 20<br>RXPEIEE | ECC Inject Error Enable for Rx Parser Memory<br>Indicates whether the ECC inject error for receive parser memory is enabled.<br>If this field is 1, it enables the ECC error injection feature.<br>When this field becomes 0, it disables the ECC error injection feature. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |
| 19-17<br>— | Reserved. |
| 16<br>WRRDN | Read Write Control<br>Controls the read and write operation to the receive parser memory.<br>If this field is 1, it indicates the write operation to the receive parser memory.<br>When this field is 0, it indicates the read operation to the receive parser memory.<br>0b - Read operation to the receive parser memory<br>1b - Write operation to the receive parser memory |
| 15-8<br>— | Reserved. |
| 7-0<br>ADDR | FRP Instruction Table Offset Address<br>Indicates the ADDR of the 32-bit entry in receive parser instruction table. Each entry has 128-bit (4x32-bit words).<br>The X depends on the configurable DWC_EQOS_FRP_ENTRIES.<br>If DWC_EQOS_FRP_ENTRIES = 256 , then X = 9<br>If DWC_EQOS_FRP_ENTRIES = 128 , then X = 8<br>If DWC_EQOS_FRP_ENTRIES = 64, then X = 7 |

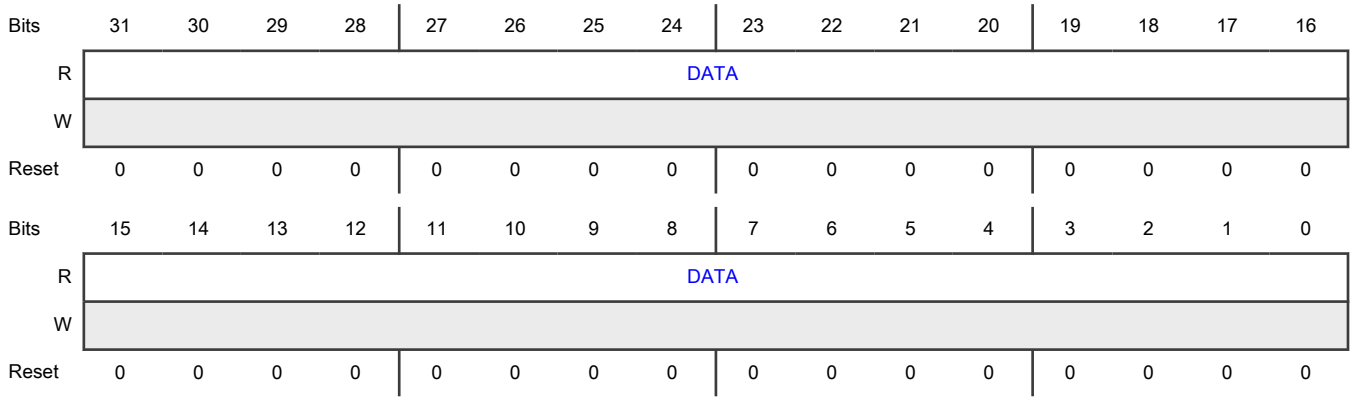
## 72.17.206 MTL Rx Parser Indirect Access Data (MTL_RXP_Indirect_Acc_Data)

**Offset**

| Register | Offset |
|---|---|
| MTL_RXP_Indirect_Acc_Data | CB4h |

**Function**
Provides the data associated to indirect access to receive parser memory.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>DATA | FRP Instruction Table Write/Read Data<br>You must write this register before issuing any write command.<br>After read command, when MTL_RXP_Indirect_Acc_Control_Status[STARTBUSY] = 0, the hardware provides the read data from the receive parser memory for read operation. |

## 72.17.207 MTL ECC Control (MTL_ECC_Control)

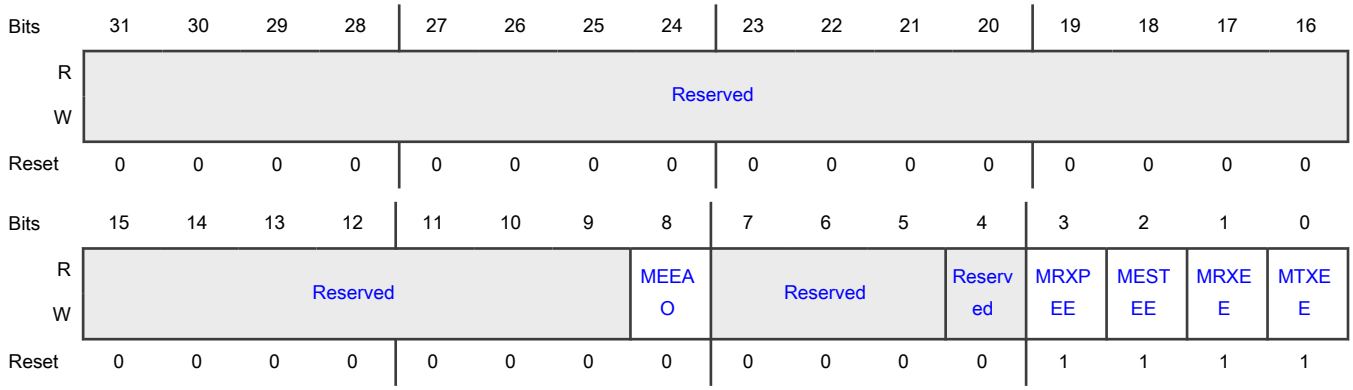
### Offset

| Register | Offset |
|---|---|
| MTL_ECC_Control | CC0h |

### Function

Establishes the operating mode of ECC related to MTL memories.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved. |
| 8<br>MEEAO | MTL ECC Error Address Status Over-ride<br>Enables or disables the MTL ECC error address status over-ride.<br>When this field is 1, it indicates that MTL_ECC_Err_Addr_Status[EUEAS] and MTL_ECC_Err_Addr_Status[ECEAS] hold the last valid address where the error is detected. When this field becomes 0, it indicates that MTL_ECC_Err_Addr_Status[EUEAS] and MTL_ECC_Err_Addr_Status[ECEAS] hold the first address where the error is detected.<br>0b - Disable<br>1b - Enable |
| 7-5<br>— | Reserved. |
| 4<br>— | Reserved. |
| 3<br>MRXPEE | MTL Rx Parser ECC Enable<br>Indicates whether the MTL receive parser ECC is enabled.<br>If this field is 1, it enables the ECC feature for receive parser memory. When this field is 0, it disables the ECC feature for receive parser memory.<br>0b - Disabled<br>1b - Enabled |
| 2<br>MESTEE | MTL EST ECC Enable<br>Indicates whether the MTL EST ECC is enabled.<br>If this field is 1, it enables the ECC feature for EST memory. When this field is 0, it disables the ECC feature for EST memory.<br>0b - Disabled<br>1b - Enabled |
| 1<br>MRXEE | MTL Rx FIFO ECC Enable<br>Indicates whether the MTL Rx FIFO ECC is enabled.<br>If this field is 1, it enables the ECC feature for MTL receive FIFO memory. When this field is 0, it disables the ECC feature for MTL receive FIFO memory.<br>0b - Disabled<br>1b - Enabled |
| 0 | MTL Tx FIFO ECC Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| MTXEE | Indicates whether the MTL Tx FIFO ECC is enabled.<br>If this field is 1, it enables the ECC feature for MTL Tx FIFO memory. When field is 0, it disables the ECC feature for MTL Tx FIFO memory.<br>0b - Disabled<br>1b - Enabled |

## 72.17.208 MTL Safety Interrupt Status (MTL_Safety_Interrupt_Status)

### Offset

| Register | Offset |
|---|---|
| MTL_Safety_Interrupt_Status | CC4h |

### Function

Provides safety interrupt status.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | | | | | | | | | | | MEUIS | MECIS |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved<br>Indicates whether the MAC safety uncorrectable interrupt status is detected.<br>0b - Not detected<br>1b - Detected |
| 30-2 | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 1<br>MEUIS | MTL ECC Uncorrectable Error Interrupt Status<br>Indicates whether the MTL ECC uncorrectable error interrupt status is detected.<br>Indicates that an uncorrectable error interrupt event in the MTL ECC safety feature. The application must read MTL_ECC_Interrupt_Status to get the exact cause of the interrupt.<br>0b - Not detected<br>1b - Detected |
| 0<br>MECIS | MTL ECC Correctable Error Interrupt Status<br>Indicates whether the MTL ECC correctable error interrupt status is detected.<br>Indicates that a correctable error interrupt event in the MTL ECC safety feature. The application must read MTL_ECC_Interrupt_Status to get the exact cause of the interrupt .<br>0b - MTL ECC Correctable error Interrupt Status not detected<br>1b - MTL ECC Correctable error Interrupt Status detected |

## 72.17.209 MTL ECC Interrupt Enable (MTL_ECC_Interrupt_Enable)

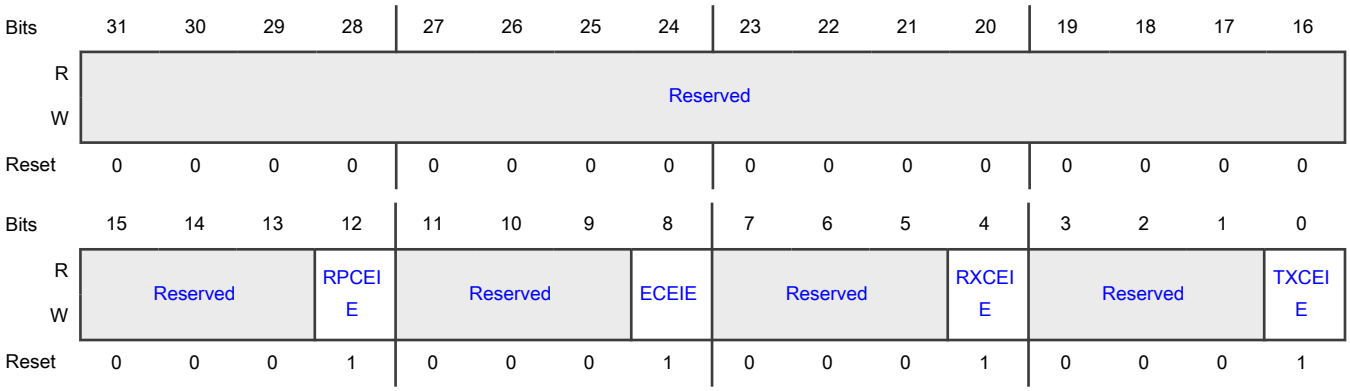
### Offset

| Register | Offset |
|---|---|
| MTL_ECC_Interrupt_Enable | CC8h |

### Function

Provides enable bits for the ECC interrupts.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-13<br>— | Reserved. |
| 12<br>RPCEIE | Rx Parser Memory Correctable Error Interrupt Enable<br>Indicates whether the Rx parser memory correctable error interrupt is enabled.<br>When this field is 1, it generates an interrupt when an uncorrectable error is detected at the Rx parser memory interface. It is indicated in MTL_ECC_Interrupt_Status[RPCES].<br>When this field becomes 0, it indicates that this event does not generates an interrupt.<br>0b - Disabled<br>1b - Enabled |
| 11-9<br>— | Reserved. |
| 8<br>ECEIE | EST Memory Correctable Error Interrupt Enable<br>Indicates whether the EST memory correctable error interrupt is enabled.<br>When this field is 1, it generates an interrupt when a correctable error is detected at the MTL EST memory interface. It is indicated in MTL_ECC_Interrupt_Status[ECES].<br>When this field becomes 0, it indicates that this event does not generates an interrupt.<br>0b - Disabled<br>1b - Enabled |
| 7-5<br>— | Reserved. |
| 4<br>RXCEIE | Rx Memory Correctable Error Interrupt Enable<br>Indicates whether the Rx memory correctable error interrupt is enabled.<br>When this field is 1, it generates an interrupt when a correctable error is detected at the MTL Rx memory interface. It is indicated in MTL_ECC_Interrupt_Status[RXCES].<br>When this field becomes 0, it indicates that this event does not generates an interrupt.<br>0b - Disabled<br>1b - Enabled |
| 3-1<br>— | Reserved. |
| 0<br>TXCEIE | Tx Memory Correctable Error Interrupt Enable<br>Indicates whether the Tx memory correctable error interrupt is enabled. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When this field is 1, it generates an interrupt when a correctable error is detected at the MTL Tx memory interface. It is indicated in MTL_ECC_Interrupt_Status[TXCES].<br>When this field becomes 0, it indicates that this event does not generates an interrupt.<br>0b - Disabled<br>1b - Enabled |

## 72.17.210 MTL ECC Interrupt Status (MTL_ECC_Interrupt_Status)

### Offset

| Register | Offset |
|---|---|
| MTL_ECC_Interrupt_Status | CCCh |

### Function

Provides MTL ECC Interrupt Status.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | RPUES | RPAMS | RPCES | Reserved | EUES | EAMS | ECES | Reserved | RXUES | RXAMS | RXCES | Reserved | TXUES | TXAMS | TXCES |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-15<br>— | Reserved. |
| 14<br>RPUES | Rx Parser Memory Uncorrectable Error Status<br>Indicates whether the Rx parser memory uncorrectable error status is detected.<br>When this field is 1, it indicates that an uncorrectable error is detected at Rx parser memory interface. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Not detected<br>1b - Detected |
| 13<br>RPAMS | MTL Rx Parser Memory Address Mismatch Status<br>Indicates whether the MTL Rx parser memory address mismatch status is detected.<br>When this field is 1, it indicates that an address mismatch is found for Rx parser memory's address bus.<br>0b - Not detected<br>1b - Detected |
| 12<br>RPCES | MTL Rx Parser Memory Correctable Error Status<br>Indicates whether the MTL Rx parser memory correctable error status is detected.<br>When this field is 1, it indicates that a correctable error is detected at RX parser memory interface.<br>0b - Not detected<br>1b - Detected |
| 11<br>— | Reserved. |
| 10<br>EUES | MTL EST Memory Uncorrectable Error Status<br>Indicates whether the MTL EST memory uncorrectable error status is detected.<br>When this field is 1, it indicates that an uncorrectable error is detected at MTL EST memory interface.<br>0b - Not detected<br>1b - Detected |
| 9<br>EAMS | MTL EST Memory Address Mismatch Status<br>Indicates whether the MTL EST memory address mismatch status is detected.<br>When this field is 1, it indicates that an address mismatch is found for MTL EST memory's address bus.<br>0b - Not detected<br>1b - Detected |
| 8<br>ECES | MTL EST Memory Correctable Error Status<br>Indicates whether the MTL EST memory correctable error status is detected.<br>When this field is 1, it indicates that a correctable error is detected at the MTL EST memory.<br>0b - Not detected<br>1b - Detected |
| 7<br>— | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 6<br>RXUES | MTL Rx Memory Uncorrectable Error Status<br>Indicates whether the MTL Rx memory uncorrectable error status is detected.<br>When this field is 1, it indicates that an uncorrectable error is detected at the MTL Rx memory interface.<br>0b - Not detected<br>1b - Detected |
| 5<br>RXAMS | MTL Rx Memory Address Mismatch Status<br>Indicates whether the MTL Rx memory address mismatch status is detected.<br>When this field is 1, it indicates that an address mismatch is found for MTL Rx memory's address bus.<br>0b - Not detected<br>1b - Detected |
| 4<br>RXCES | MTL Rx memory Correctable Error Status<br>Indicates whether the MTL Rx memory correctable error status is detected.<br>When this field is 1, it indicates that a correctable error is detected at the MTL Rx memory.<br>0b - Not detected<br>1b - Detected |
| 3<br>— | Reserved. |
| 2<br>TXUES | MTL Tx Memory Uncorrectable Error Status<br>Indicates whether the MTL Tx memory uncorrectable error status is detected.<br>When this field is 1, it indicates that an uncorrectable error is detected at the MTL TX memory interface.<br>0b - Not detected<br>1b - Detected |
| 1<br>TXAMS | MTL Tx Memory Address Mismatch Status<br>Indicates whether the MTL Tx memory address mismatch status is detected.<br>When this field is 1, it indicates that an address mismatch is found for MTL Tx memory's address bus.<br>0b - Not detected<br>1b - Detected |
| 0<br>TXCES | MTL Tx Memory Correctable Error Status<br>Indicates whether the MTL Tx memory correctable error status is detected.<br>When this field is 1, it indicates that a correctable error is detected at the MTL Tx memory.<br>0b - Not detected |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Detected |

## 72.17.211 MTL ECC Error Status (MTL_ECC_Err_Sts_Rctl)

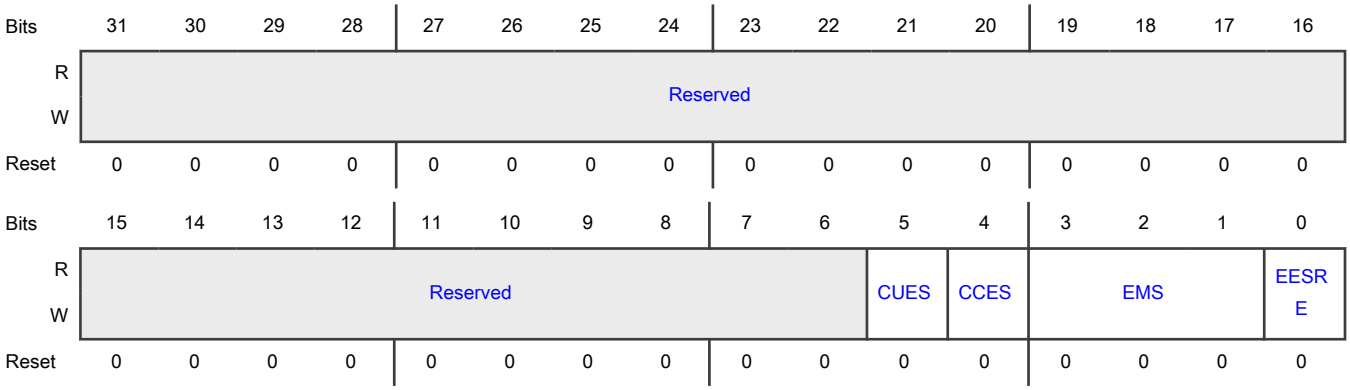
### Offset

| Register | Offset |
|---|---|
| MTL_ECC_Err_Sts_Rctl | CD0h |

### Function

Establishes the control for ECC error status capture.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-6<br>— | Reserved. |
| 5<br>CUES | Clear Uncorrectable Error Status<br>Indicates whether the clear uncorrectable error status is detected.<br>When this field and MTL_ECC_Err_Sts_Rctl[EESRE] is 1, it indicates that based on the MTL_ECC_Err_Sts_Rctl[EMS], the respective memory's uncorrectable error address and uncorrectable error count values are cleared upon reading.<br>When all the error status values are cleared hardware resets this field.<br>0b - Not detected<br>1b - Detected |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 4<br>CCES | Clear Correctable Error Status<br>Indicates whether the clear correctable error status is detected.<br>When this field and MTL_ECC_Err_Sts_Rctl[EESRE] is 1, it indicates that based on the MTL_ECC_Err_Sts_Rctl[EMS],, the respective memory's correctable error address and correctable error count values are cleared upon reading.<br>When all the error status values are cleared, hardware resets this field.<br>0b - Not detected<br>1b - Detected |
| 3-1<br>EMS | MTL ECC Memory Selection<br>Provides the memory selection encoding.<br>When MTL_ECC_Err_Sts_Rctl[EESRE] is 1, this field indicates which memory's error status value to be read.<br>The memory selection encoding is as described below.<br>000b - MTL Tx memory<br>001b - MTL Rx memory<br>010b - MTL EST memory<br>011b - MTL Rx Parser memory<br>100b - DMA TSO memory |
| 0<br>EESRE | MTL ECC Error Status Read Enable<br>Indicates whether the MTL ECC error status read is enabled.<br>When this field is 1, it indicates that based on MTL_ECC_Err_Sts_Rctl[EMS], the respective memory's error status values are captured as described below:<br>• The correctable and uncorrectable error count values are captured into MTL_ECC_Err_Cnt_Status.<br>• The address location of correctable and uncorrectable errors are captured into MTL_ECC_Err_Addr_Status.<br>When all the status values are captured into MTL_ECC_Err_Cnt_Status and MTL_ECC_Err_Addr_Status, hardware resets this field.<br>0b - Disabled<br>1b - Enabled |

## 72.17.212 MTL ECC Error Address Status (MTL_ECC_Err_Addr_Status)

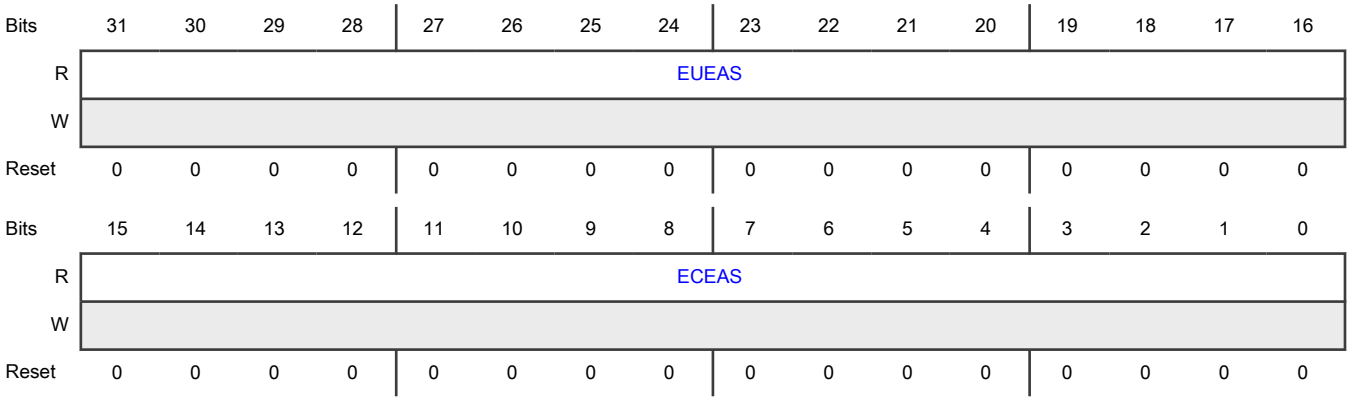
### Offset

| Register | Offset |
|---|---|
| MTL_ECC_Err_Addr_Status | CD4h |

### Function

Provides the memory addresses for the correctable and uncorrectable errors.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>EUEAS | MTL ECC Uncorrectable Error Address Status<br>Based on the MTL_ECC_Err_Sts_Rctl[EMS], this field holds the respective memory's address locations for which an uncorrectable error or address mismatch is detected.<br>When MTL_ECC_Control[MEEAO] = 1, this field holds the last valid address of memory for which either an uncorrectable error or an address mismatch is detected.<br>When MTL_ECC_Control[MEEAO] = 0, this field holds the first address of the memory for which either an uncorrectable error or address mismatch is detected. |
| 15-0<br>ECEAS | MTL ECC Correctable Error Address Status<br>Based on the MTL_ECC_Err_Sts_Rctl[EMS], this field holds the respective memory's address locations for which a correctable error is detected.<br>When MTL_ECC_Control[MEEAO] = 1, this field holds the last valid address of memory for which the correctable error or address mismatch is detected.<br>When MTL_ECC_Control[MEEAO] = 0, this field holds the first address of the memory for which correctable error is detected. |

## 72.17.213 MTL ECC Error Control Status (MTL_ECC_Err_Cntr_Status)

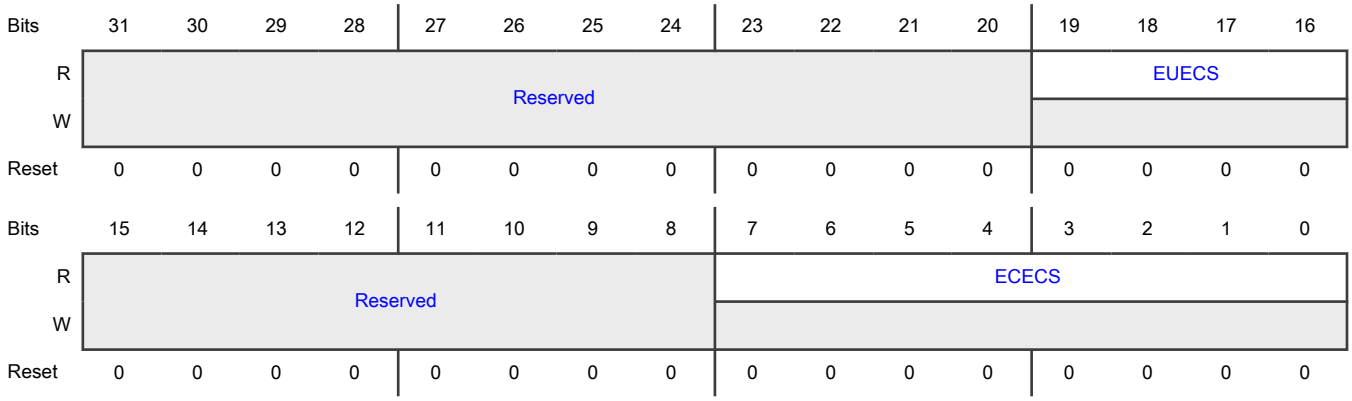
### Offset

| Register | Offset |
|---|---|
| MTL_ECC_Err_Cntr_Status | CD8h |

### Function

Provides ECC Error count for correctable and uncorrectable errors.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved. |
| 19-16<br>EUECS | MTL ECC Uncorrectable Error Counter Status<br>Based on the EMS field of MTL_ECC_Err_Cntr_Rctl register, this field holds the respective memory's uncorrectable error count value. |
| 15-8<br>— | Reserved. |
| 7-0<br>ECECS | MTL ECC Correctable Error Counter Status<br>Based on the EMS field of MTL_ECC_Err_Cntr_Rctl register, this field holds the respective memory's correctable error count value. |

## 72.17.214 MTL DPP Control (MTL_DPP_Control)

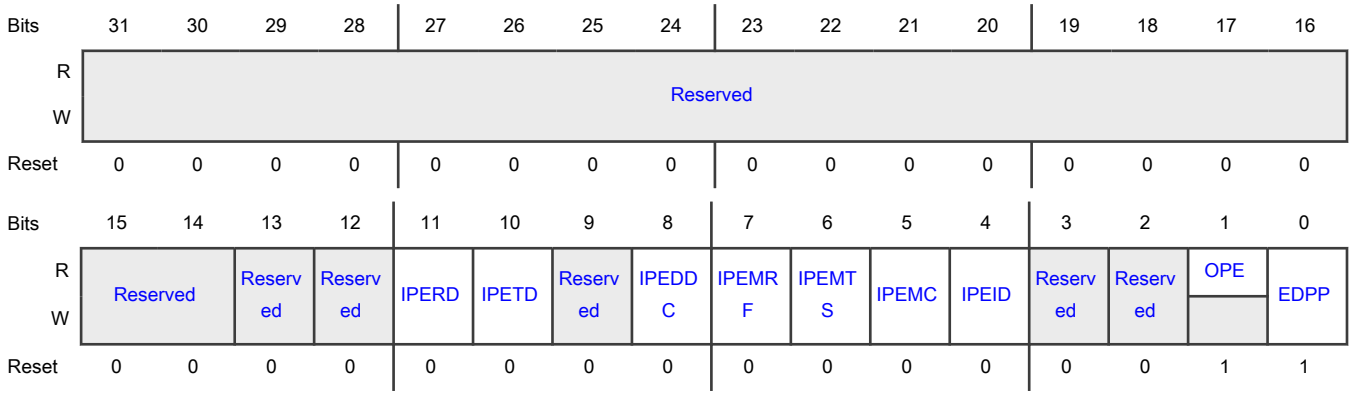
### Offset

| Register | Offset |
|---|---|
| MTL_DPP_Control | CE0h |

### Function

Establishes the operating mode of data parity protection and error injection.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-14<br>— | Reserved. |
| 13<br>— | Reserved. |
| 12<br>— | Reserved. |
| 11<br>IPERD | Insert Parity error in Rx write-back Descriptor parity generator<br>Indicates whether the insert parity error in receive write-back descriptor parity generator is enabled.<br>When this field is 1, it flips the parity bit of first valid data to which the DMA Rx write-back descriptor parity generator(or at PG8 as shown in Receive data path parity protection diagram) generates.<br>0b - Disabled<br>1b - Enabled |
| 10<br>IPETD | Insert Parity error in Tx write-back Descriptor parity generator<br>Indicates whether the insert parity error in transit write-back descriptor parity generator is enabled.<br>When this field is 1, it flips the parity bit of first valid data to which the DMA Tx write-back descriptor parity generator(or at PG4 as shown in Transmit data path parity protection diagram) generates.<br>Hardware clears this field, when the respective parity bit flips.<br>0b - Disabled<br>1b - Enabled |
| 9<br>— | Reserved. |
| 8 | Insert Parity Error in DMA DTX Control Word Parity Generator |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| IPEDDC | Indicates whether the insert parity error in DMA DTX control word parity generator is enabled.<br>When this field is 1, it flips the parity bit of first valid data to which the DMA DTX Control word parity generator (or at PG2 as shown in Transmit data path parity protection diagram) generates.<br>Hardware clears this field, when the respective parity bit flips.<br>0b - Disabled<br>1b - Enabled |
| 7<br>IPEMRF | Insert Parity Error in MTL Rx FIFO Read Control Parity Generator<br>Inserts parity Error in MTL receive FIFO read control parity generator.<br>When this field is 1, it flips the parity bit of first valid data to which the MTL Rx FIFO read control parity generator (or at PG7 as shown in Receive data path parity protection diagram) generates.<br>Hardware clears this field, when the respective parity bit flips.<br>0b - Disabled<br>1b - Enabled |
| 6<br>IPEMTS | Insert Parity Error in MTL Tx Status Parity Generator<br>Indicates whether the insert parity error in MTL transit status parity generator is enabled.<br>When this field is 1, it flips the parity bit of first valid data to which the MTL Tx Status parity generator (or at PG6 as shown in Transmit data path parity protection diagram) generates.<br>Hardware clears this field, when the respective parity bit flips.<br>0b - Disabled<br>1b - Enabled |
| 5<br>IPEMC | Insert Parity Error in MTL Checksum Parity Generator<br>Indicates whether the insert parity error in MTL checksum parity generator is enabled.<br>When this field is 1, it flips the parity bit of first valid data to which the MTL checksum parity generator (or at PG5 as shown in Transmit data path parity protection diagram) generates.<br>Hardware clears this field, when the respective parity bit flips.<br>0b - Disabled<br>1b - Enabled |
| 4<br>IPEID | Insert Parity Error in Interface Data Parity Generator<br>Indicates whether the insert parity error in interface data parity generator is enabled.<br>When this field is 1, it flips the parity bit of first valid input data to which the interface data parity generator (or at PG1 as shown in Transmit data path parity protection diagram) generates.<br>Following are the input data bus on which parity bits are generated on the basis of configuration selected.<br>In AHB Config, hrdata_i<br>In AXI config, rdata_m_i |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | In DMA Config, mdc_rdata_i<br>In MTL Config, ati_data_i<br>Hardware clears this field, when the respective parity bit flips.<br>0b - Disabled<br>1b - Enabled |
| 3<br>— | Reserved. |
| 2<br>— | Reserved. |
| 1<br>OPE | Odd Parity Enable<br>Indicates whether an odd parity protection is enabled.<br>When this field is 1, it enables an odd parity protection on all the external interfaces and when this field is 0, it enables an even parity protection on all the external interfaces.<br>0b - Disabled<br>1b - Enabled |
| 0<br>EDPP | Enable Data path Parity Protection<br>Enables or disables the data path parity protection.<br>When this field is 1, it indicates that by generating and checking the parity on EQOS datapath it enables the parity protection for EQOS datapath . When this field is 0, it disables the parity protection for EQOS datapath.<br>0b - Disable<br>1b - Enable |

## 72.17.215 MTL Tx Queue 0 Operation Mode (MTL_TxQ0_Operation_Mode)

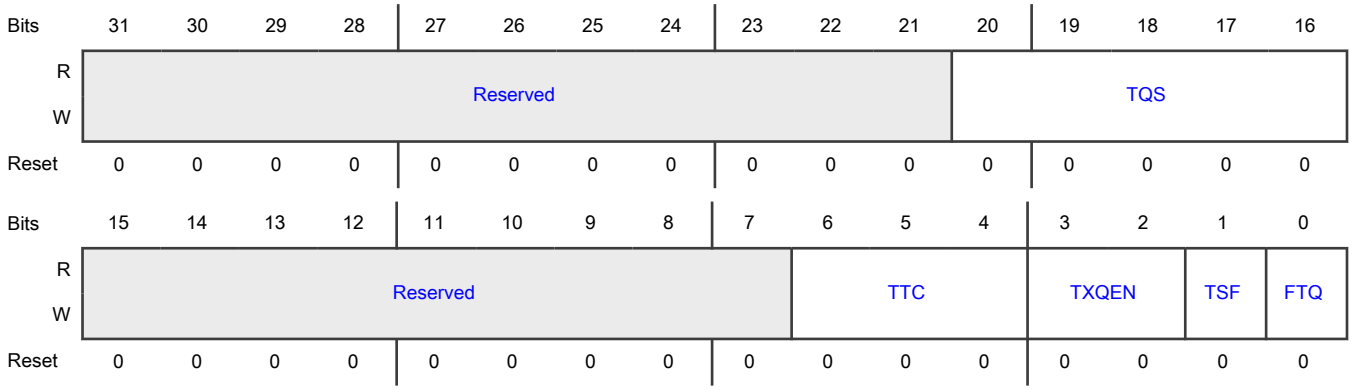
### Offset

| Register | Offset |
|---|---|
| MTL_TxQ0_Operation_Mode | D00h |

### Function

Establishes the transmit queue operating modes and commands.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-21<br>— | Reserved. |
| 20-16<br>TQS | Transmit Queue Size<br>Indicates the size of the allocated transmit queues in blocks of 256 bytes. This field is read-write only if the number of transit queues are more than one, the reset value is 0x0 and indicates 256 bytes size. This means that value of 0x0 = 256 bytes, 0x1 = 512 bytes and so on. You must program TQS [5:0] = 6'b001111 to allocate queue size of 4096 (4K) bytes . In general, the size of the Queue = (TQS+1)*256 bytes.<br>When the number of transit queue is one, the field is read-only and the reset value reflects the configured TX FIFO size in blocks of 256 bytes.<br>The field's width depends on the transit memory size selected in your configuration. For example, if the memory size is 2048, the field's width is 3 bits:<br>LOG2(2048/256) = LOG2(8) = 3 bits |
| 15-7<br>— | Reserved. |
| 6-4<br>TTC | Transmit Threshold Control<br>Controls the threshold level of the MTL transit queue. The transmission starts when the packet size within the MTL transit queue is larger than the threshold, also transmits full packets with length less than the threshold. These fields are used only when MTL_TxQ0_Operation_Mode[TSF] resets.<br>000b - 32<br>001b - 64<br>010b - 96<br>011b - 128<br>100b - 192<br>101b - 256 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 110b - 384<br>111b - 512 |
| 3-2<br>TXQEN | Transmit Queue Enable<br>Enables or disables the transmit queue 0.<br>2'b00 - Not enabled<br>2'b01 - Reserved<br>2'b10 - Enabled<br>2'b11 - Reserved<br>This field is read only in single queue configurations and read write in multiple queue configurations.<br>**NOTE**<br>In multiple transit queue configuration, all the queues disables by default and programming this field enables the transit queue.<br>00b - Not enabled<br>01b - Enable in AV mode (Reserved in non-AV)<br>10b - Enabled<br>11b - Reserved |
| 1<br>TSF | Transmit Store and Forward<br>Indicates whether the transmit store and forward is enabled.<br>When this field is 1, it indicates that the transmission starts when a full packet resides in the MTL transit queue and the TTC values specified in Bits[6:4] of this register are ignored. This field must be changed only when you stop the transmission.<br>0b - Disabled<br>1b - Enabled |
| 0<br>FTQ | Flush Transmit Queue<br>Indicates whether the flush transmit queue is enabled.<br>When this field is 1, it indicates that the transit queue controller logic resets to its default values. Therefore, all the data in the transit queue is lost or flushed. After the flushing operation completes this field resets internally. You must not write to the MTL_TxQ1_Operation_Mode until this field resets. The data to which the MAC transmitter has already accepted is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission.<br>**NOTE**<br>The flush operation completes only when the transit queue is empty and the application accepts the pending transit status of all the transmitted packets. To complete the flush operation, the PHY Tx clock (CLK_TX_I) must be active. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Access restriction apply to this field. It clears automatically. Writing 1 sets this field and writing 0 has no effect.<br>0b - Disabled<br>1b - Enabled |

## 72.17.216 MTL Tx Queue 0 Underflow (MTL_TxQ0_Underflow)

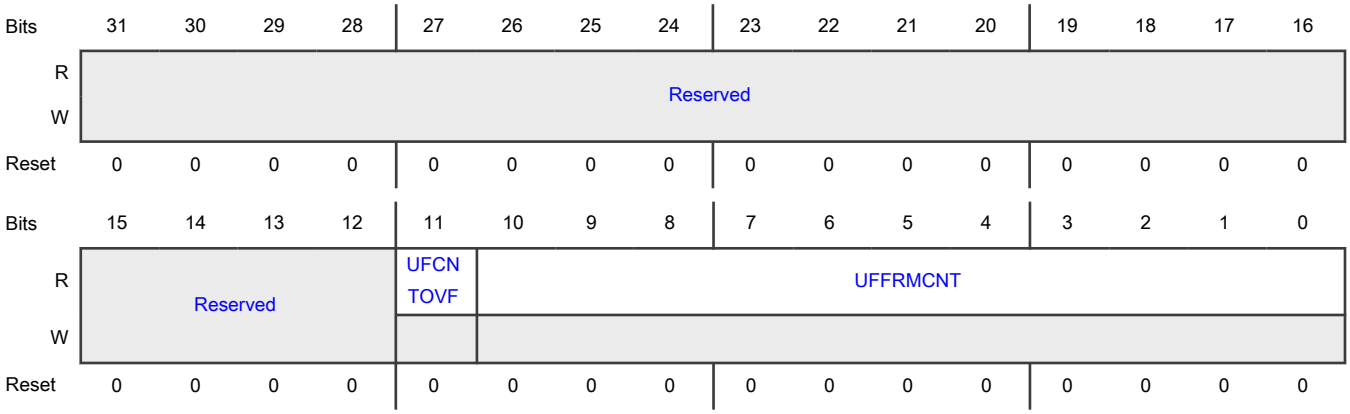
### Offset

| Register | Offset |
|---|---|
| MTL_TxQ0_Underflow | D04h |

### Function

Contains the counter for packets aborted because of transmit queue underflow and packets missed because of receive queue packet flush.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved. |
| 11<br>UFCNTOVF | Overflow Bit for Underflow Packet Counter<br>Indicates whether the overflow is detected for underflow packet counter. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | This field is 1, when the transit queue underflow packet counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter resets to all-zeros and this field indicates that the rollover occurred.<br>Access restriction apply to this field. It clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 10-0<br>UFFRMCNT | Underflow Packet Counter<br>Indicates the number of packets aborted by the controller because of transit queue underflow. This counter increments each time the MAC aborts outgoing packet because of underflow. The counter clears when this register is read with mci_be_i[0] at 1'b1.<br>Access restriction apply to this field. It clears on read and automatically becomes 1 on an internal event occurrence. |

## 72.17.217 MTL Tx Queue 0 Debug (MTL_TxQ0_Debug)

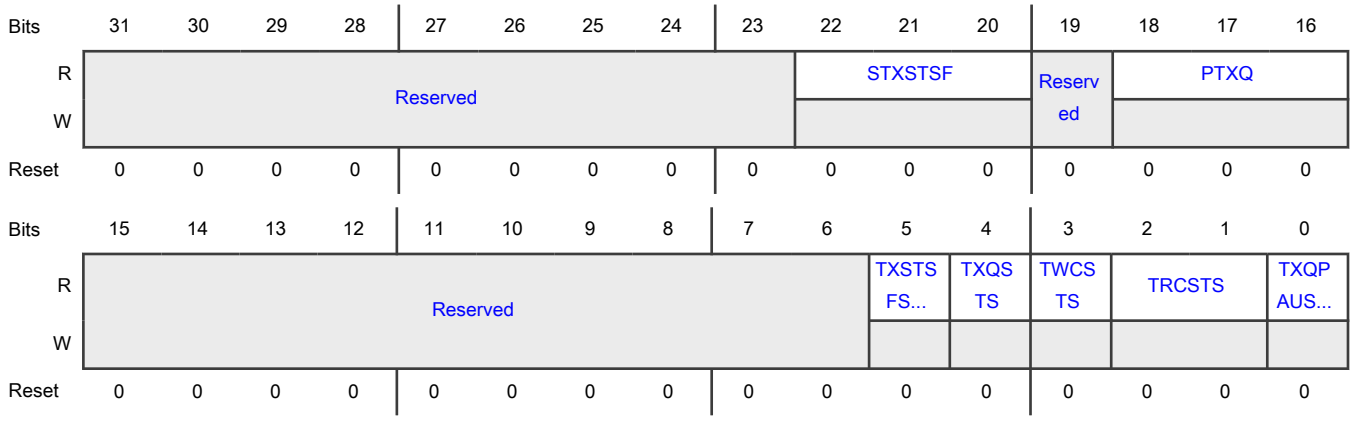
### Offset

| Register | Offset |
|---|---|
| MTL_TxQ0_Debug | D08h |

### Function

Provides the debug status of various blocks related to the transmit queue.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-23<br>— | Reserved. |
| 22-20<br>STXSTSF | Number of Status Words in Tx Status FIFO of Queue<br>Indicates the current number of status in the transit status FIFO of this queue.<br>This field does not reflect the number of status words in transit status FIFO, when MTL_Operation_Mode[DTXSTS] = 1. |
| 19<br>— | Reserved. |
| 18-16<br>PTXQ | Number of Packets in the Transmit Queue<br>Indicates the current number of packets in the transit queue.<br>This field does not reflect the number of packets in the transmit queue, when MTL_Operation_Mode[DTXSTS] = 1. |
| 15-6<br>— | Reserved. |
| 5<br>TXSTSFSTS | MTL Tx Status FIFO Full Status<br>Indicates whether the MTL transit status and FIFO full status is detected.<br>When this field is 1, it indicates that the MTL transit status FIFO is full. Therefore, MTL cannot accept any more packets for transmission.<br>0b - Not detected<br>1b - Detected |
| 4<br>TXQSTS | MTL Tx Queue Not Empty Status<br>Indicates whether the MTL transit queue not empty status is detected.<br>When this field is high, it indicates that the MTL transit queue is not empty and some data is left for transmission.<br>0b - Not detected<br>1b - Detected |
| 3<br>TWCSTS | MTL Tx Queue Write Controller Status<br>Indicates whether the MTL transit queue write controller status is detected.<br>When high, this field indicates that the MTL transit queue write controller is active, and transfers the data to the transit queue.<br>0b - Not detected<br>1b - Detected |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 2-1<br>TRCSTS | MTL Tx Queue Read Controller Status<br>Indicates the state of the transit queue read controller.<br>00b - Idle state<br>01b - Read state (transferring data to the MAC transmitter)<br>10b - Waiting for pending transit status from the MAC transmitter<br>11b - Flushing the transit queue because of the packet abort request from the MAC |
| 0<br>TXQPAUSED | Transmit Queue in Pause<br>Indicates whether the transmit queue in pause is detected.<br>When this field is 1 and the receive flow control is enabled, it indicates that the transit queue is in the pause condition (in the full-duplex only mode) because of the following scenario:<br>• Receives PFC packet with the priorities assigned to the transit queue when PFC is enabled.<br>• Receives 802.3x pause packet when PFC is disabled.<br>0b - Not detected<br>1b - Detected |

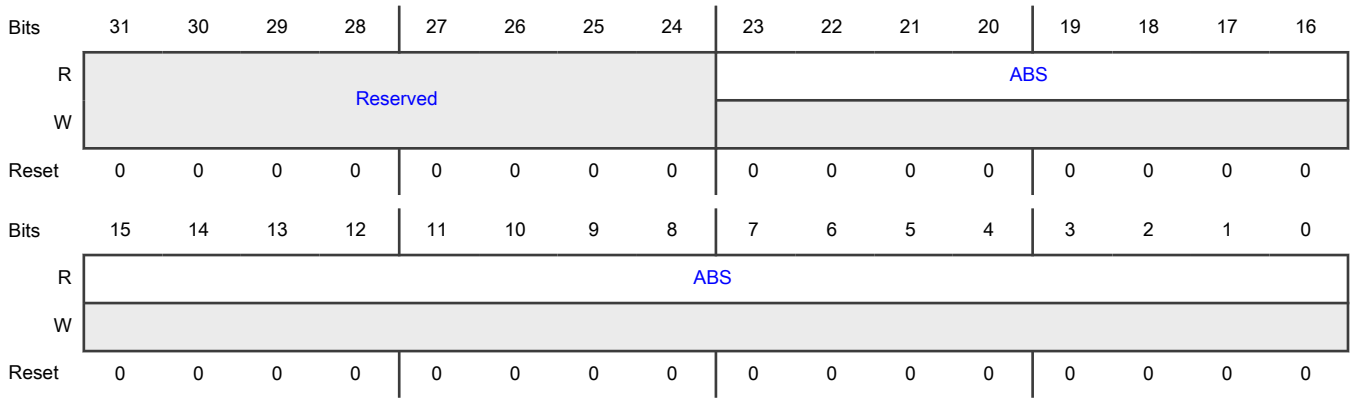
## 72.17.218 MTL Tx Queue 0 ETS Status (MTL_TxQ0_ETS_Status)

### Offset

| Register | Offset |
|---|---|
| MTL_TxQ0_ETS_Status | D14h |

### Function
Provides the average traffic transmitted in queue 0.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-0<br>ABS | Average Bits per Slot<br>Contains the average transmitted bits per slot.<br>Computes over every 10 million bit times slot (4 ms in 2500 Mbit/s; 10 ms in 1000 Mbit/s; 100 ms in 100 Mbit/s), when the DCB operation enables for queue 0. The maximum value is 0x989680. |

## 72.17.219 MTL Tx Queue Quantum Weight (MTL_TxQ0_Quantum_Weight)

### Offset

| Register | Offset |
|---|---|
| MTL_TxQ0_Quantum_Weight | D18h |

### Function
Contains the quantum value for Deficit Weighted Round Robin (DWRR), weights for the Weighted Round Robin (WRR), and Weighted Fair Queuing (WFQ) for queue 0.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | ISCQW | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | ISCQW | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-21<br>— | Reserved. |
| 20-0 | Quantum or Weights |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| ISCQW | Contains the quantum value in bytes which is added to credit during every queue scanning cycle, when DCB operation enables with DWRR algorithm for queue 0 traffic. The maximum value is 0x1312D0 bytes.<br>Contains the weight for this queue, when DCB operation enables with WFQ algorithm for queue 0 traffic. The maximum value is 0x3FFF where weight of 0 indicates 100% bandwidth. Write 0 to bits[20:14]. The higher the programmed weights, the lesser is the bandwidth that is allocated for the particular transmit queue. This is because the weights compute the packet finish time (weights*packet_size). The lesser the finish time, the higher is the probability of the packet getting scheduled first and consuming more bandwidth.<br>Contains the weight for this queue, when DCB operation or generic queuing operation enables with WRR algorithm for queue 0 traffic. The maximum value is 0x64.<br>Write 0 to bits [20:7]. |

## 72.17.220 MTL Queue 0 Interrupt Control Status (MTL_Q0_Interrupt_Control_Status)

### Offset

| Register | Offset |
|---|---|
| MTL_Q0_Interrupt_Control_Status | D2Ch |

### Function

Contains the interrupt enable and status bits for the queue 0 interrupts.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | RXOIE | Reserved | | | | | | | RXOVFIS |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | ABPSIE | TXUIE | Reserved | | | | | | ABPSIS | TXUNFIS |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 24<br>RXOIE | Receive Queue Overflow Interrupt Enable<br>Enables or disables the receive queue overflow interrupt.<br>When this field is 1, it enables the receive queue overflow interrupt.<br>When this field becomes 0, it disables the receive queue overflow interrupt.<br>0b - Disabled<br>1b - Enabled |
| 23-17<br>— | Reserved. |
| 16<br>RXOVFIS | Receive Queue Overflow Interrupt Status<br>Indicates whether the status of receive queue overflow interrupt is detected.<br>This bit indicates that the receive queue had an overflow when you receive the packet. If you transfer a partial packet to the application, the overflow status sets in RDES3[21]. This field is 0 when the application writes 1 to it.<br>Access restriction apply to this field. It becomes 0 automatically on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 15-10<br>— | Reserved. |
| 9<br>ABPSIE | Average Bits Per Slot Interrupt Enable<br>Enables or disables the average bits per slot interrupt.<br>When this field is 1, it indicates that when you update the average bits per slot status, the MAC asserts the sbd_intr_o or mci_intr_o interrupt.<br>When this field is 0, it indicates that the interrupt do not assert for such an event.<br>0b - Disabled<br>1b - Enabled |
| 8<br>TXUIE | Transmit Queue Underflow Interrupt Enable<br>Enables or disables transmit queue underflow interrupt.<br>When this field is 1, it enables the transmit queue underflow interrupt.<br>When this field becomes 0, it disables the transmit queue underflow interrupt.<br>0b - Disabled<br>1b - Enabled |
| 7-2 | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 1<br>ABPSIS | Average Bits Per Slot Interrupt Status<br>Indicates whether the status of average bits per slot interrupt is detected.<br>When this field is 1, it indicates that the MAC has updated the ABS value. This field is 0 when the application writes 1 it.<br>Access restriction apply to this field. It becomes 0 automatically on an internal event occurrence. Writing 1 sets this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 0<br>TXUNFIS | Transmit Queue Underflow Interrupt Status<br>Indicates whether the status of transmit queue underflow interrupt is detected.<br>This field indicates that, when you transmit the packet,the transmit queue had an underflow. Suspend the transmission and write 1 to an underflow error TDES3[2].<br>This field is 0, when the application writes 1 to it.<br>Access restriction apply to this field. It becomes 0 automatically on an internal event occurrence. Writing 1 sets this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |

## 72.17.221 MTL Rx Queue 0 Operation Mode (MTL_RxQ0_Operation_Mode)

### Offset

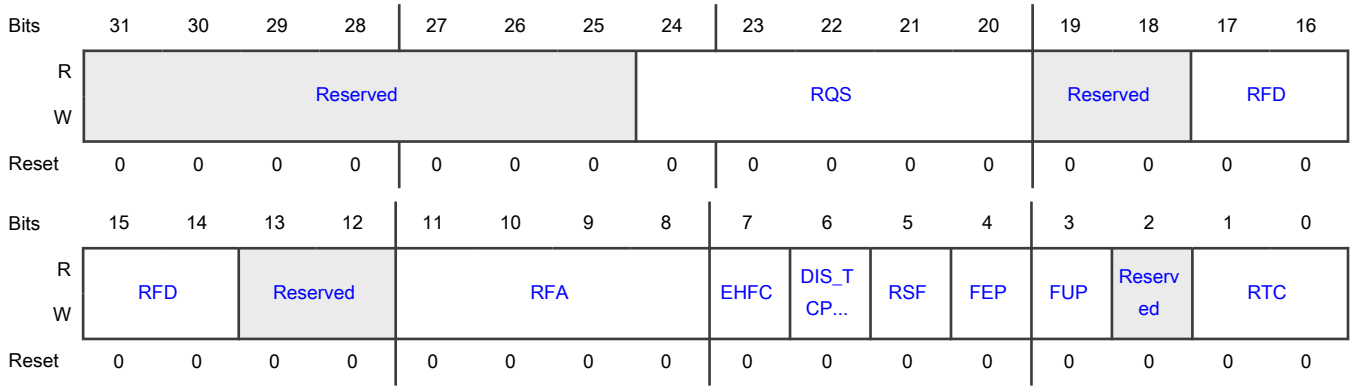
| Register | Offset |
|---|---|
| MTL_RxQ0_Operation_Mode | D30h |

### Function

Establishes the receive queue operating modes and command.

The RFA and RFD fields are not backward compatible with the RFA and RFD fields of 4.00a release

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved. |
| 24-20<br>RQS | Receive Queue Size<br>Indicates the size of the allocated receive queues in blocks of 256 bytes. The MTL_RxQ0_Operation_Mode[RQS] is read-write only if the number of receive queues are more than one, the reset value is 0x0 and indicates size of 256 bytes. This means that value of 0x0 = 256 bytes, 0x1 = 512 bytes and so on. You must program RQS [5:0] = 6'b001111 to allocate queue size of 4096 (4K) bytes. In general, the size of the Queue = (RQS+1)*256 bytes.<br>When the number of receive queues is one, the field is read-only and the configured receive FIFO size in blocks of 256 bytes is reflected in the reset value.<br>The field width depends on the receive memory size selected in your configuration. For example, if the memory size is 2048, the field width is 3 bits:<br>LOG2(2048/256) = LOG2(8) = 3 bits |
| 19-18<br>— | Reserved. |
| 17-14<br>RFD | Threshold for Deactivating Flow Control (in half-duplex and full-duplex modes)<br>Controls the threshold (fill-level of Rx queue) at which the flow control is de-asserts after activation.<br>0 - Full minus 1 KB, that is, FULL 1 KB<br>1 - Full minus 1.5 KB, that is, FULL 1.5 KB<br>2 - Full minus 2 KB, that is, FULL 2 KB<br>3 - Full minus 2.5 KB, that is, FULL 2.5 KB<br>...<br>14 - Full minus 8 KB, that is, FULL 8 KB<br>15 - Full minus 8.5 KB, that is, FULL 8.5 KB |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | De-assertion is effective only after flow control is asserted.<br>**NOTE**<br>You must program the value in such a way that the threshold is a positive number.<br>When MTL_RxQ0_Operation_Mode[EHFC] = 1, these values are applicable only when the receive queue size which the MTL_RxQ0_Operation_Mode[RQS] determines is equal to or greater than 4 KB.<br>For a given queue size, the values ranges between 0 and the encoding for FULL minus (QSIZE - 0.5 KB) and all other values are illegal. Here the term FULL and QSIZE refers to the queue size which the MTL_RxQ0_Operation_Mode[RQS] determines.<br>The field width depends on RX FIFO size selected during the configuration. Remaining bits are reserved and read only. |
| 13-12<br>— | Reserved. |
| 11-8<br>RFA | Threshold for Activating Flow Control (in half-duplex and full-duplex)<br>Controls the threshold (fill-level of receive queue) at which the flow control is activated.<br>See MTL_RxQ0_Operation_Mode[RFD] for more information on encoding for this field. |
| 7<br>EHFC | Enable Hardware Flow Control<br>Enables or disables hardware flow control.<br>If this field is 1, it enables the flow control signal operation, on the basis of the fill-level of receive queue.<br>When this field becomes 0, it disables the flow control operation.<br>0b - Disable<br>1b - Enable |
| 6<br>DIS_TCP_EF | Disable Dropping of TCP/IP Checksum Error Packets<br>Enables or disable dropping of TCP or IP checksum error packets.<br>When this field is 1, it indicates that the MAC does not drop the packets which only have the errors which the receive checksum offload engine detects. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC.<br>When this field becomes 0, it indicates that all error packets are dropped if MTL_RxQ0_Operation_Mode[FEP] resets.<br>0b - Enable<br>1b - Disable |
| 5<br>RSF | Receive Queue Store and Forward<br>Indicates whether the receive queue store and forward is enabled.<br>When this field is 1, it indicates that the module reads a packet from the receive queue only after the complete packet has been written to it, ignores the MTL_RxQ0_Operation_Mode[RTC]. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When this field becomes 0, it indicates that the receive queue operates in the Threshold (cut-through) mode, subject to the threshold which the MTL_RxQ0_Operation_Mode[RTC] specifies.<br>0b - Disabled<br>1b - Enabled |
| 4<br>FEP | Forward Error Packets<br>Indicates whether the forward error packets are enabled.<br>When this field becomes 0, it indicates that the receive queue drop packets with an error status (CRC error, GMII_ER, watchdog timeout, or overflow). However, the packet does not drop if the start byte (write) pointer of a packet is already transferred to the read controller side (in Threshold mode).<br>When this field is 1, it indicates that all packets except the runt error packets are forwarded to the application or DMA. The packet is dropped irrespective of the setting of this field, if MTL_RxQ0_Operation_Mode[RSF] is 1 and the receive queue overflows when a partial packet is written. However, if MTL_RxQ0_Operation_Mode[RSF] becomes 0 and the receive queue overflows when a partial packet is written, a partial packet might be forwarded to the application or DMA.<br>0b - Disabled<br>1b - Enabled |
| 3<br>FUP | Forward Undersized Good Packets<br>Indicates whether the forward undersized good packets are enabled.<br>When this field is 1, it indicates that the receive queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC.<br>When this field becomes 0, it indicates that the receive queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of receive threshold, for example, MTL_RxQ0_Operation_Mode[RTC] = 01.<br>0b - Disabled<br>1b - Enabled |
| 2<br>— | Reserved. |
| 1-0<br>RTC | Receive Queue Threshold Control<br>Controls the threshold level of the MTL receive queue (in bytes).<br>The received packet is transferred to the application or DMA when the packet size within the MTL receive queue is larger than the threshold. In addition, automatically transfer full packets with length less than the threshold.<br>This field is valid only when MTL_RxQ0_Operation_Mode[RSF] = 0.<br>This field is ignored when MTL_RxQ0_Operation_Mode[RSF] = 1.<br>00b - 64 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 01b - 32<br>10b - 96<br>11b - 128 |

## 72.17.222 MTL Rx Queue Missed Packet Overflow Count (MTL_RxQ0_Missed_Packet_Overflow_Cnt)

### Offset

| Register | Offset |
|---|---|
| MTL_RxQ0_Missed_Packet_Overflow_Cnt | D34h |

### Function

Contains the counter for packets missed because of receive queue packet flush and packets discarded because of receive queue overflow.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | MISCNTO... | MISPKTCNT | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | | | | OVFCNTO... | OVFPKTCNT | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27<br>MISCNTOVF | Missed Packet Counter Overflow Bit<br>Indicates whether the missed packet counter overflow is detected.<br>When this field is 1, it indicates that the receive queue missed packet counter has crossed the maximum limit. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Access restriction apply to this field. It clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 26-16<br>MISPKTCNT | Missed Packet Counter<br>Indicates the number of packets this module has missed because the application asserts ari_pkt_flush_i[] for this queue. This counter resets when this register reads with mci_be_i[0] at 1b1.<br>This counter increments by 1 when the DMA discards the packet because of buffer unavailability.<br>Access restriction apply to this field. It clears on read and automatically becomes 1 on an internal event occurrence. |
| 15-12<br>— | Reserved. |
| 11<br>OVFCNTOVF | Overflow Counter Overflow Bit<br>Indicates whether the counter overflow is detected.<br>When this field is 1, it indicates that the receive queue overflow packet counter field has crossed the maximum limit.<br>Access restriction apply to this field. It clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 10-0<br>OVFPKTCNT | Overflow Packet Counter<br>Indicates the number of packets which this module discards because of receive queue overflow. This counter increments each time the module discards an incoming packet because of overflow. This counter resets when this register reads with mci_be_i[0] at 1'b1.<br>Access restriction apply to this field. It clears on read and automatically becomes 1 on an internal event occurrence. |

## 72.17.223 MTL Rx Queue 0 Debug (MTL_RxQ0_Debug)

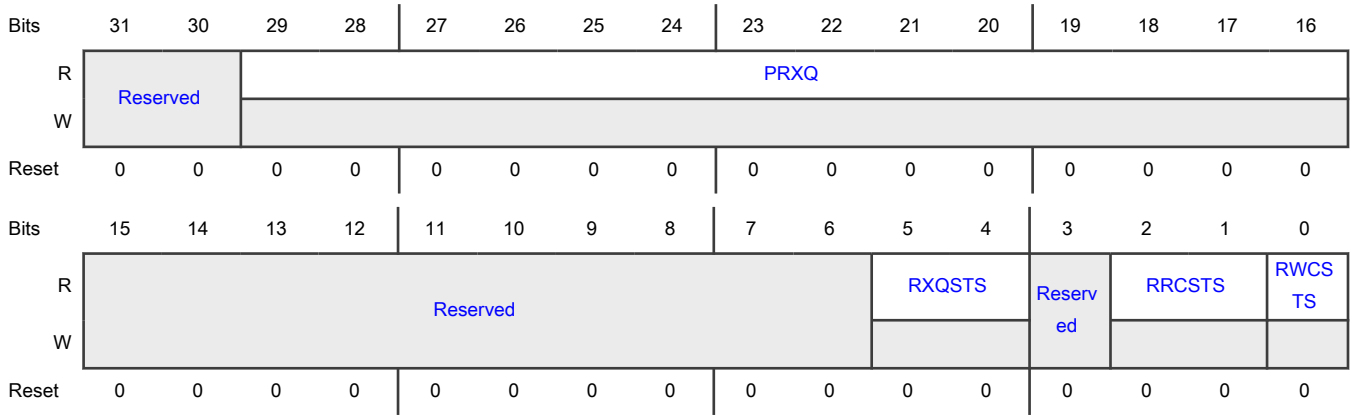
### Offset

| Register | Offset |
|---|---|
| MTL_RxQ0_Debug | D38h |

### Function

Provides the debug status of various blocks related to the receive queue.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30<br>— | Reserved. |
| 29-16<br>PRXQ | Number of Packets in Receive Queue<br>Indicates the current number of packets in the receive queue. The theoretical maximum value for this field is 256KB/16B = 16K packets, that is, Max_Queue_Size/Min_Packet_Size. |
| 15-6<br>— | Reserved. |
| 5-4<br>RXQSTS | MTL Rx Queue Fill-Level Status<br>Provides the status of the receive queue fill-level.<br>00b - Rx Queue empty<br>01b - Rx Queue fill-level below flow-control deactivate threshold<br>10b - Rx Queue fill-level above flow-control activate threshold<br>11b - Rx Queue full |
| 3<br>— | Reserved. |
| 2-1<br>RRCSTS | MTL Rx Queue Read Controller State<br>Provides the state of the receive queue read controller.<br>00b - Idle state<br>01b - Reading packet data<br>10b - Reading packet status (or timestamp)<br>11b - Flushing the packet data and status |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 0<br>RWCSTS | MTL Rx Queue Write Controller Active Status<br>Indicates whether the MTL receive queue write controller active status is detected.<br>When high, this bit indicates that the MTL receive queue write controller is active, and it is transferring a received packet to the receive queue.<br>0b - Not detected<br>1b - Detected |

## 72.17.224 MTL Rx Queue 0 Control 0 (MTL_RxQ0_Control)

### Offset

| Register | Offset |
|---|---|
| MTL_RxQ0_Control | D3Ch |

### Function

Controls the receive arbitration and passing of received packets to the application.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | RXQ_FRM... | RXQ_WEGT | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved. |
| 3<br>RXQ_FRM_ARBIT | Receive Queue Packet Arbitration<br>Indicates whether the receive queue packet arbitration is enabled. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When this field is 1, it indicates that the module drives the packet data to the ARI interface such that the entire packet data of currently-selected queue transmits before switching to other queue.<br>When this field becomes 0, it indicates that the module drives the packet data to the ARI interface such that the following amount of data of currently-selected queue transmits before switching to other queue:<br>• PBL amount of data (indicated by ari_qN_pbl_i[]), or<br>• Complete data of a packet<br>The status and the timestamp are not a part of the PBL data. Therefore, the module drives the complete status (including timestamp status) during first PBL request for the packet (in store-and-forward mode) or the last PBL request for the packet (in Threshold mode).<br>0b - Disabled<br>1b - Enabled |
| 2-0<br>RXQ_WEGT | Receive Queue Weight<br>Indicates the weight assigned to receive queue 0.<br>You must write a value to this field that is 1 less than the required queue weight. Therefore, if this field is 0, it indicates that the queue weight is 1, if this field is 1, it indicates that the queue weight is 2, and so on. You must use this weight to calculate the number of continuous PBL or packet requests, depending on the value of RXQ_FRM_ARBIT, allocated to the queue in an arbitration cycle.<br>The field's value changes when the current service round completes or when there is a change from RAA=SP to RAA=WSP algorithm. This approach is required for a smooth transition. You must configure MTL Rx Queue 0 Control 0 (MTL_RxQ0_Control) before MTL Operation Mode (MTL_Operation_Mode) if you want to change the field's value before either of these two processes completes. |

## 72.17.225 MTL Tx Queue 1 Operation Mode (MTL_TxQ1_Operation_Mode)

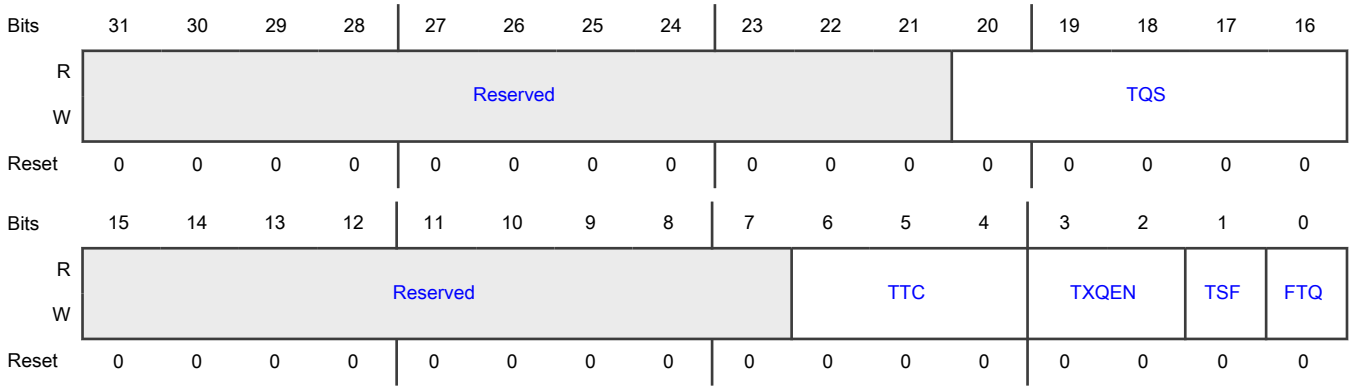
### Offset

| Register | Offset |
|---|---|
| MTL_TxQ1_Operation_Mode | D40h |

### Function

Establishes the transmit queue operating modes and commands.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-21<br>— | Reserved. |
| 20-16<br>TQS | Transmit Queue Size<br>Indicates the size of the allocated transmit queues in blocks of 256 bytes. This field is read-write only if the number of transit queues is more than one, the reset value is 0x0 and indicates size of 256 bytes. This means that value of 0x0 = 256 bytes, 0x1 = 512 bytes and so on. You must program TQS [5:0] = 6'b001111 to allocate queue size of 4096 (4K) bytes. In general, the size of the queue = (TQS+1)*256 bytes.<br>When the number of transit queues is one, the field is read-only and the reset value reflects the configured TX FIFO size in blocks of 256 bytes.<br>The field width depends on the transit memory size selected in your configuration. For example, if the memory size is 2048, the field width is 3 bits:<br>LOG2(2048/256) = LOG2(8) = 3 bits |
| 15-7<br>— | Reserved. |
| 6-4<br>TTC | Transmit Threshold Control<br>Controls the threshold level of the MTL Tx Queue. The transmission starts when the packet size within the MTL Tx Queue is larger than the threshold. In addition, it also transmit full packets with length less than the threshold. These fields are used only when the MTL_TxQ1_Operation_Mode[TSF] resets.<br>000b - 32<br>001b - 64<br>010b - 96<br>011b - 128<br>100b - 192<br>101b - 256 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 110b - 384<br>111b - 512 |
| 3-2<br>TXQEN | Transmit Queue Enable<br>Enables or disables the transmit queue 0.<br>2'b00 - Not enabled<br>2'b01 - Enable in AV mode<br>2'b10 - Enabled<br>2'b11 - Reserved<br>**NOTE**<br>All the queues disables by default, in multiple transit queues configuration. You must program this field to enable the transit queue.<br>00b - Not enabled<br>01b - Enable in AV mode (Reserved in non-AV)<br>10b - Enabled<br>11b - Reserved |
| 1<br>TSF | Transmit Store and Forward<br>Indicates whether the transmit store and forward is enabled.<br>When this field is 1, it indicates that the transmission starts when full packet resides in the MTL transit queue and the TTC values specified in MTL_TxQ1_Operation_Mode[TTC] are ignored. This field must change only when the transmission stops.<br>0b - Transmit Store and Forward is disabled<br>1b - Transmit Store and Forward is enabled |
| 0<br>FTQ | Flush Transmit Queue<br>Indicates whether the flush transmit queue is enabled.<br>When this field is 1, it indicates that the transmit queue controller logic resets to its default values. Therefore, all the data in the transit queue is lost or flushed. This field resets internally when the flushing operation completes. Until this field becomes 0, you must not write to MTL_TxQ1_Operation_Mode. The data which the MAC transmitter has already accepted is not flushed. It is scheduled for transmission and results in an underflow and runt packet transmission.<br>**NOTE**<br>The flush operation completes only when the transit queue is empty and the application has accepted the pending transit status of all transmitted packets. To complete this flush operation, the PHY Tx clock (CLK_TX_I) must be active.<br>Access restriction apply to this field. Writing 1 sets this field and it clears automatically. Writing 0 has no effect. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disabled<br>1b - Enabled |

## 72.17.226 MTL Tx Queue 1 Underflow (MTL_TxQ1_Underflow)

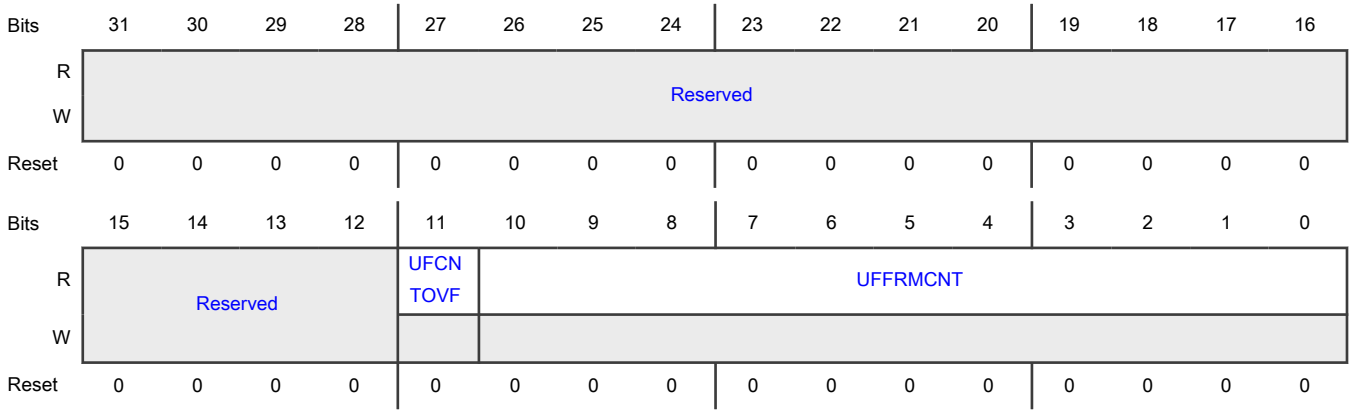
### Offset

| Register | Offset |
|---|---|
| MTL_TxQ1_Underflow | D44h |

### Function

Contains the counter for packets aborted because of transmit queue underflow and packets missed because of receive queue packet flush.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-12<br>— | Reserved. |
| 11<br>UFCNTOVF | Overflow Bit for Underflow Packet Counter<br>Indicates whether the overflow is detected for underflow packet counter.<br>When this field is 1, it indicates that every time the transit queue underflow packet counter field overflows, that is, it has crossed the maximum count. In such case, the overflow packet counter resets to all-zeros and this field indicates that the rollover have occurred. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Access restriction apply to this field. It clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 10-0<br>UFFRMCNT | Underflow Packet Counter<br>Indicates the number of packets which the controller abort because of transit queue underflow. This counter increments each time the MAC aborts outgoing packet because of underflow. The counter clears when this register is read with mci_be_i[0] at 1'b1.<br>Access restriction apply to this field. It clears on read and automatically becomes 1 on an internal event occurrence. |

## 72.17.227 MTL Tx Queue 1 Debug (MTL_TxQ1_Debug)

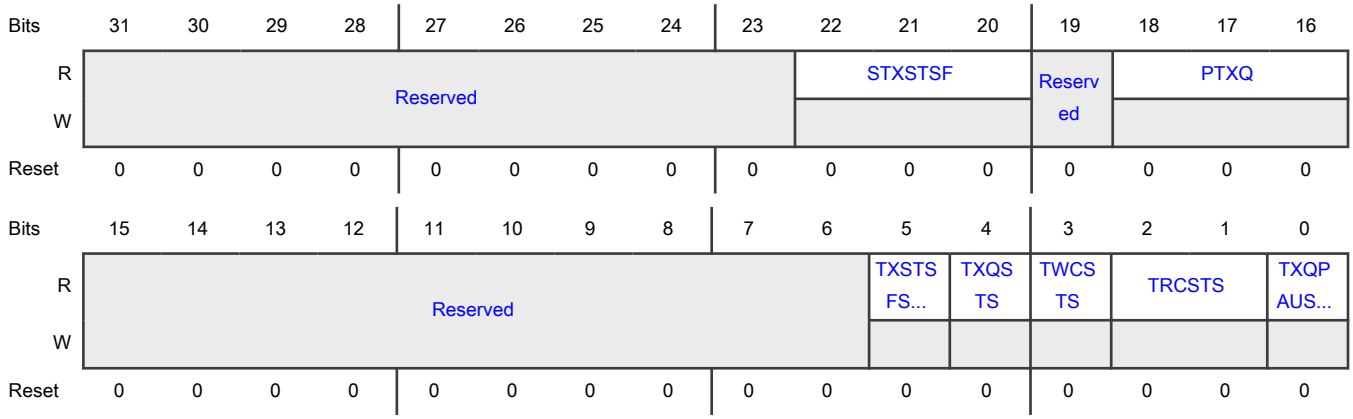
### Offset

| Register | Offset |
|---|---|
| MTL_TxQ1_Debug | D48h |

### Function

Provides the debug status of various blocks related to the transmit queue.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-23 | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 22-20<br>STXSTSF | Number of Status Words in Tx Status FIFO of Queue<br>Indicates the current number of status in the transit status FIFO of this queue.<br>When MTL_Operation_Mode[DTXSTS] = 1, this field does not reflect the number of status words in transit status FIFO. |
| 19<br>— | Reserved. |
| 18-16<br>PTXQ | Number of Packets in the Transmit Queue<br>Indicates the current number of packets in the transit queue.<br>When MTL_Operation_Mode[DTXSTS] = 1, this field does not reflect the number of packets in the transmit queue. |
| 15-6<br>— | Reserved. |
| 5<br>TXSTSFSTS | MTL Tx Status FIFO Full Status<br>Indicates whether the MTL transit status FIFO full status is detected.<br>When this field is 1, it indicates that the MTL transit status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission.<br>0b - Not detected<br>1b - Detected |
| 4<br>TXQSTS | MTL Tx Queue Not Empty Status<br>Indicates whether the MTL transit queue not empty status is detected.<br>When this field is 1, it indicates that the MTL transit queue is not empty and some data is left for transmission.<br>0b - Not detected<br>1b - Detected |
| 3<br>TWCSTS | MTL Tx Queue Write Controller Status<br>Indicates whether the MTL transit queue write controller status is detected.<br>When this field is 1, it indicates that the MTL transit queue write controller is active, and it is transferring the data to the transit queue.<br>0b - Not detected<br>1b - Detected |
| 2-1<br>TRCSTS | MTL Tx Queue Read Controller Status<br>Indicates the state of the transit queue read controller. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 00b - Idle state<br>01b - Read state (transferring data to the MAC transmitter)<br>10b - Waiting for pending transit status from the MAC transmitter<br>11b - Flushing the transit queue because of the packet abort request from the MAC |
| 0<br>TXQPAUSED | Transmit Queue in Pause<br>Indicates whether the transmit queue in pause status is detected.<br>When this field is 1 and the receive flow control is enabled, it indicates that the transit queue is in the pause condition (in the full-duplex only mode) because of these scenarios:<br>• Reception of the PFC packet for the priorities assigned to the transit queue when PFC is enabled.<br>• Reception of 802.3x pause packet when PFC is disabled.<br>0b - Not detected<br>1b - Detected |

## 72.17.228 MTL Tx Queue 1 ETS Control (MTL_TxQ1_ETS_Control)

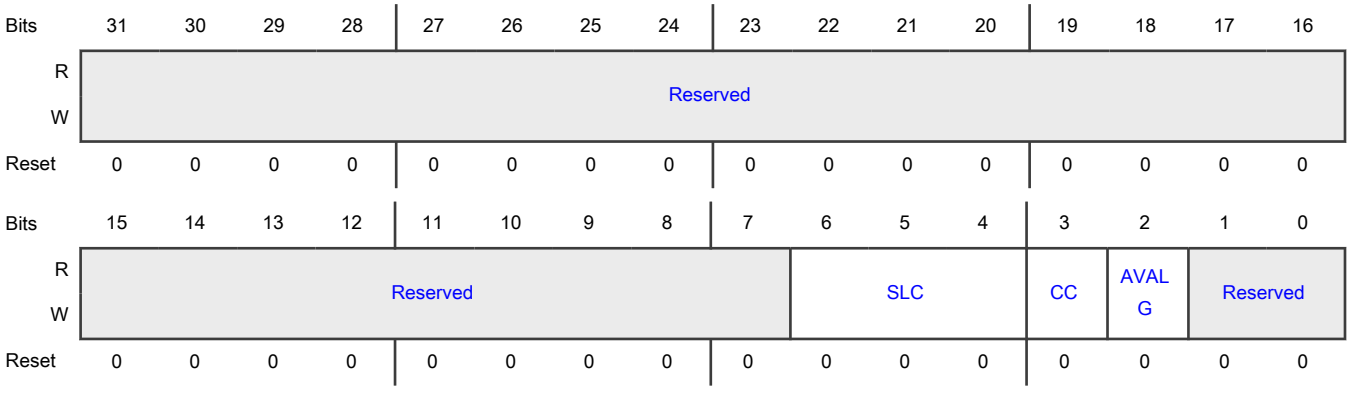
### Offset

| Register | Offset |
|---|---|
| MTL_TxQ1_ETS_Control | D50h |

### Function

Controls the enhanced transmission selection operation.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-7<br>— | Reserved. |
| 6-4<br>SLC | Slot Count<br>If the credit-based shaper algorithm is enabled, then you can program the number of slots (of duration programmed in DMA_CH(#i)_Slot_Interval register) over which the average transmitted bits per slot, provided in the MTL_TxQ(#i)_ETS_Status register, is computed for queue. The encoding is as follows:<br>000b - 1 slot<br>001b - 2 slots<br>010b - 4 slots<br>011b - 8 slots<br>100b - 16 slots<br>101b - Reserved |
| 3<br>CC | Credit Control<br>Indicates whether the credit control is enabled.<br>When this field is 1, it indicates that the accumulated credit parameter in the credit-based shaper algorithm logic is not reset to zero, when there is positive credit and no packet to transmit in channel 1. The credit accumulates even when no packet is waiting in channel 1 and another channel is transmitting.<br>When this field becomes 0, it indicates that the accumulated credit parameter in the credit-based shaper algorithm logic sets to zero, when there is positive credit and no packet to transmit in channel 1. No credit accumulates when no packet is waiting in channel 1 and other channel is transmitting.<br>0b - Disabled<br>1b - Enabled |
| 2<br>AVALG | AV Algorithm<br>Indicates whether the CBS algorithm is enabled.<br>If you program queue 1 for AV, this field configures the scheduling algorithm for this queue.<br>When this field is 1, it indicates that the credit based shaper algorithm (CBS) is selected for queue 1 traffic.<br>If this field becomes 0, strict priority is selected.<br>0b - Disabled<br>1b - Enabled |
| 1-0<br>— | Reserved. |

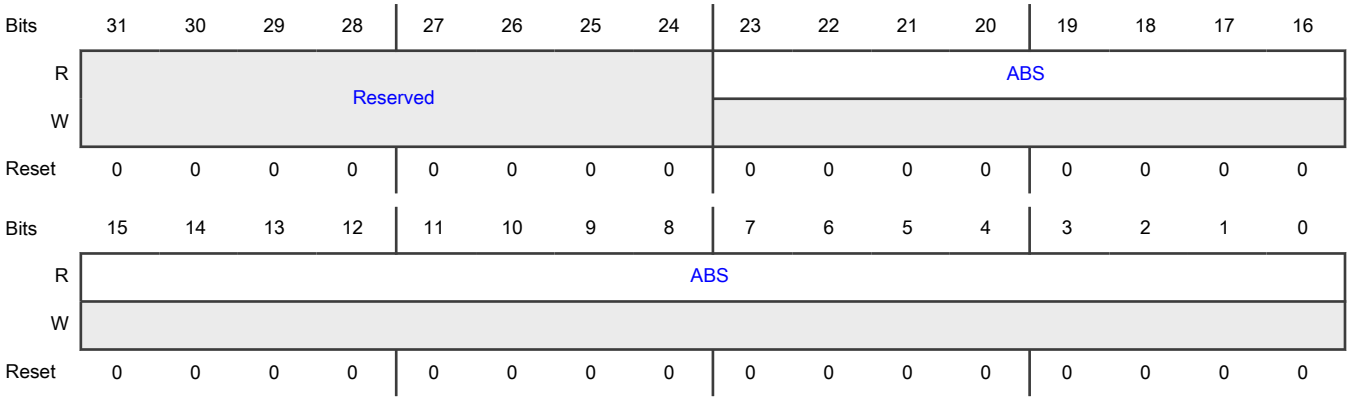
## 72.17.229 MTL Tx Queue 1 ETS Status (MTL_TxQ1_ETS_Status)

### Offset

| Register | Offset |
|---|---|
| MTL_TxQ1_ETS_Status | D54h |

### Function
Provides the average traffic transmitted in queue 1.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved. |
| 23-0<br>ABS | Average Bits per Slot<br>Contains the average transmitted bits per slot.<br>Computes this field over number of slots programmed in the MTL_TxQ(#i)_ETS_CONTROL register's SLC field of, if you enables AV operation. This field's maximum value is 0x6_4000 in 100 Mbit/s, 0x3E_8000 in 1000 Mbit/s and 9C_4000 in 2500 Mbit/s mode respectively.<br>Computes this field over every 10 million bit times slot (4 ms in 2500 Mbit/s; 10 ms in 1000 Mbit/s; 100 ms in 100 Mbit/s) when you enables DCB operation for queue. The maximum value is 0x989680. |

## 72.17.230 MTL Tx Queue 1 Quantum Weight (MTL_TxQ1_Quantum_Weight)

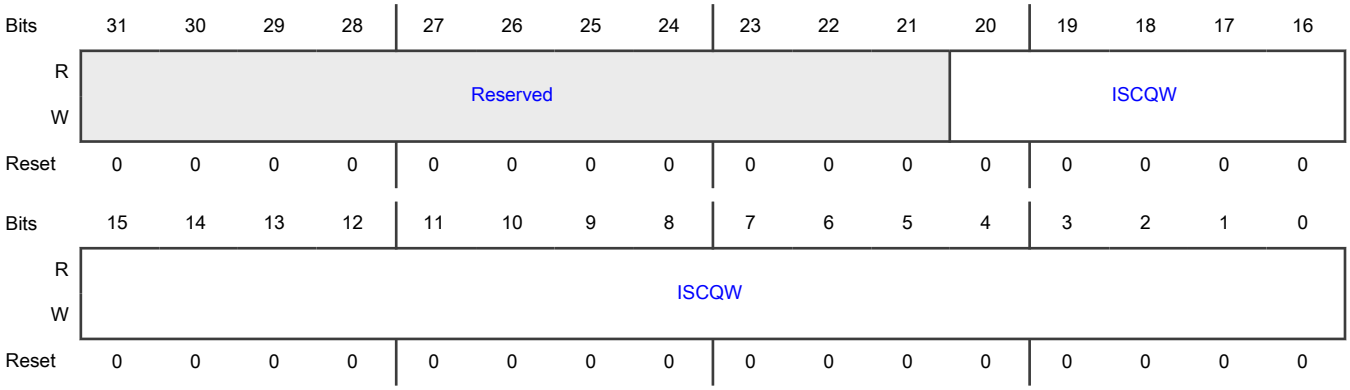
### Offset

| Register | Offset |
|---|---|
| MTL_TxQ1_Quantum_Weight | D58h |

### Function

Provides the average traffic transmitted in queue 1.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-21<br>— | Reserved. |
| 20-0<br>ISCQW | idleSlopeCredit, Quantum or Weights<br>idleSlopeCredit<br>Contains the idleSlopeCredit value required for the credit-based shaper algorithm for queue 1, when you enables the AV feature. This is the rate of change of credit in bits per cycle (40 ns for 100 Mbit/s; 8 ns for 1000 Mbit/s; 3.2 ns for 2500 Mbit/s) when the credit is increasing. You must program this field with computed credit in bits per cycle scaled by 1,024. The maximum value is portTransmitRate, that is, 0x2000 in 1000/2500 Mbit/s mode and 0x1000 in 100 Mbit/s mode. Bits[20:14] must be written to zero.<br>Quantum<br>Contains the quantum value in bytes to be added to credit during every queue scanning cycle, when you enables the DCB operation with DWRR algorithm for queue 1 traffic. The maximum value is 0x1312D0 bytes.<br>Weights<br>Contains the weight for this queue, when you enables the DCB operation with WFQ algorithm for queue 1 traffic. The maximum value is 0x3FFF where weight of 0 indicates 100% bandwidth. Bits[20:14] must be written to zero.<br>Contains the weight for this queue, when you enables the DCB operation or generic queuing operation with WRR algorithm for queue 1 traffic. The maximum value is 0x64.<br>Bits [20:7] must be written to zero. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>In multiple Queue configuration you must program this field in respective per queue register to some non-zero value when multiple queues are enabled or single queue other than Q0 is enabled. You must not program this field when only Q0 is enabled. In general, you must program a non-zero value on both receive and transmit when WRR algorithm is selected. In receive, the register is MTL_Operation_Mode register.<br>**NOTE**<br>For WFQ algorithm, higher the programmed weights lesser the bandwidth allocated for that transmit queue. The finish time is not a function of particular packet alone but it is according to the formula: (previous_finish_time of particular Transmit Queue + (weights*packet_size))<br>**NOTE**<br>The programmed weights do not correspond to the number of packets but the fraction of bandwidth or time allocated for particular queue w.r.t. total BW or time. |

## 72.17.231 MTL Tx Queue 1 Sendslope Credit (MTL_TxQ1_SendSlopeCredit)

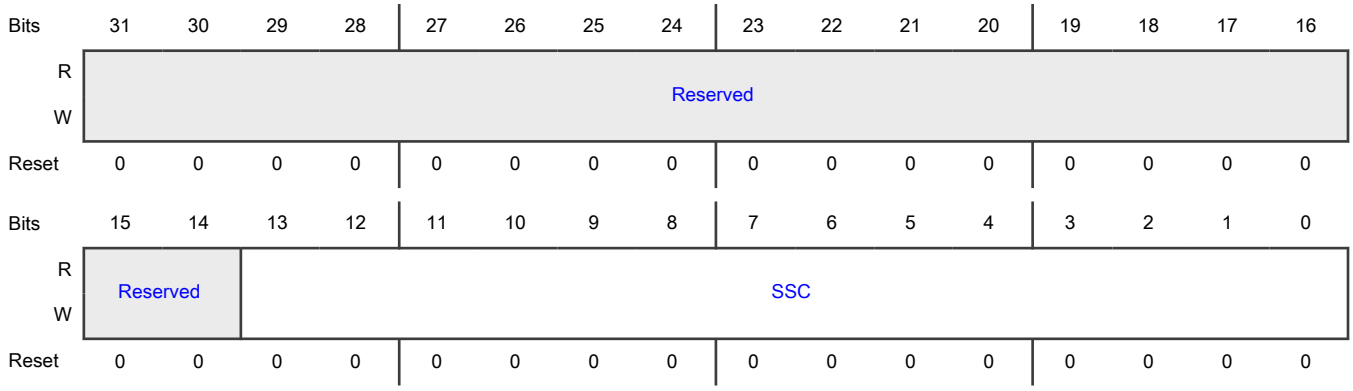
### Offset

| Register | Offset |
|---|---|
| MTL_TxQ1_SendSlopeCredit | D5Ch |

### Function

Contains the sendSlope credit value required for the credit-based shaper algorithm for the queue.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-14<br>— | Reserved. |
| 13-0<br>SSC | sendSlopeCredit Value<br>Contains the sendSlopeCredit value required for credit-based shaper algorithm for Queue 1, when you enables the AV operation. This is the rate of change of credit in bits per cycle (40 ns, 8 ns and 3.2 ns for 100 Mbit/s, 1000 Mbit/s and 2500 Mbit/s respectively) when the credit is decreasing. You must program this field with computed credit in bits per cycle scaled by 1,024. The maximum value is portTransmitRate, that is, 0x2000 in 1000/2500 Mbit/s mode and 0x1000 in 100 Mbit/s mode. You must program this field with absolute sendSlopeCredit value. The credit-based shaper logic subtracts it from the accumulated credit when channel 1 is selected for transmission. |

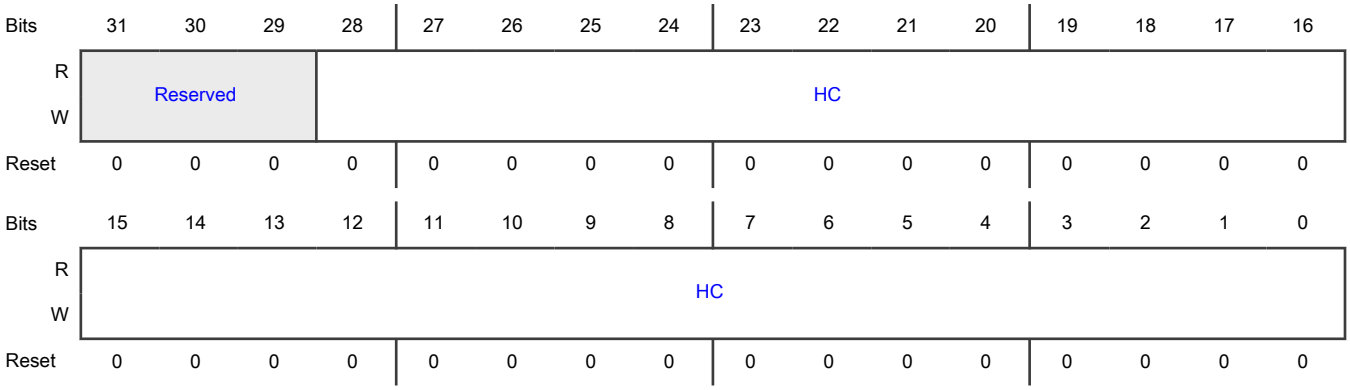
## 72.17.232 MTL Tx Queue 1 HiCredit (MTL_TxQ1_HiCredit)

### Offset

| Register | Offset |
|---|---|
| MTL_TxQ1_HiCredit | D60h |

### Function
Contains the hiCredit value required for the credit-based shaper algorithm for the queue.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 28-0<br>HC | hiCredit Value<br>Contains the hiCredit value required for the credit-based shaper algorithm, when you enables the AV feature. This is the maximum value that you can accumulate in the credit parameter. This is specified in bits scaled by 1,024.<br>The maximum value is maxInterferenceSize, that is, best-effort maximum packet size (16,384 bytes or 131,072 bits). The specified value is 131,072 * 1,024 = 134,217,728 or 0x0800_0000. |

## 72.17.233 MTL Tx Queue 1 LoCredit (MTL_TxQ1_LoCredit)

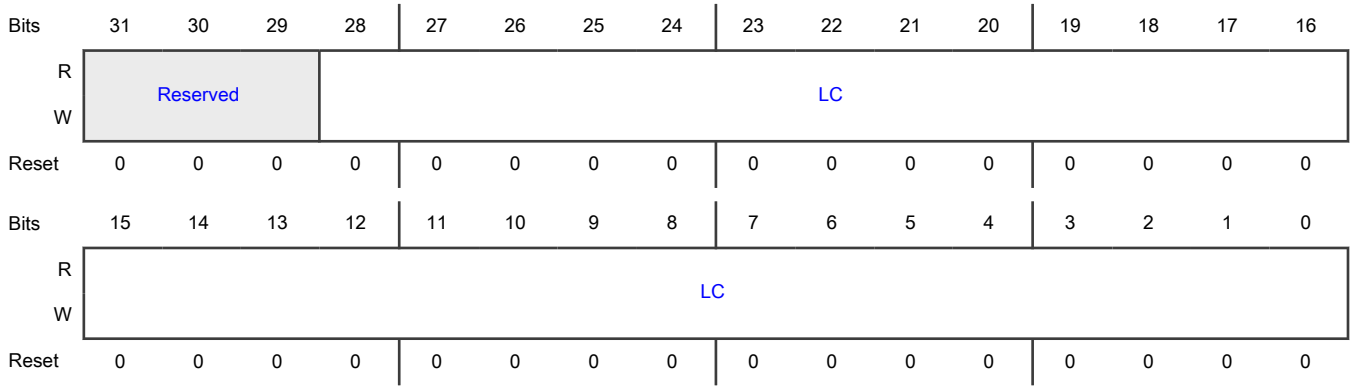
### Offset

| Register | Offset |
|---|---|
| MTL_TxQ1_LoCredit | D64h |

### Function

Contains the loCredit value required for the credit-based shaper algorithm for the queue.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved. |
| 28-0<br>LC | loCredit Value<br>Contains the loCredit value required for the credit-based shaper algorithm, when you enables the AV operation. This is the minimum value that you can accumulate in the credit parameter. This is specified in bits scaled by 1,024. The maximum value you can program corresponds to twice the maxFrameSize this queue transmits. If the maxFrameSize is 8192 bytes, then (8192*2) * 8 * 1024 = 134,217,728 or 0x0800_0000. The programmed value is 2's complement of the value, that is, 0x1800_0000 because it is a negative value. |

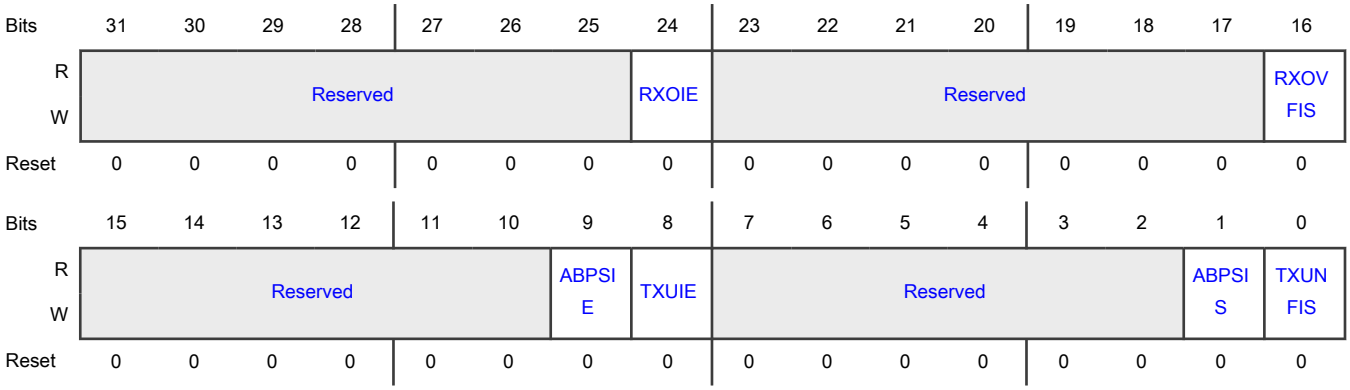
## 72.17.234 MTL Queue 1 Interrupt Control Status (MTL_Q1_Interrupt_Control_Status)

### Offset

| Register | Offset |
|---|---|
| MTL_Q1_Interrupt_Control_Status | D6Ch |

### Function
Contains the interrupt enable and status bits for the queue 1 interrupts.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved. |
| 24<br>RXOIE | Receive Queue Overflow Interrupt Enable<br>Enables or disables the receive queue overflow interrupt.<br>When this field is 1, it enables the receive queue overflow interrupt.<br>When this field becomes 0, it disables the receive queue overflow interrupt.<br>0b - Disable<br>1b - Enable |
| 23-17<br>— | Reserved. |
| 16<br>RXOVFIS | Receive Queue Overflow Interrupt Status<br>Indicates whether the status of receive queue overflow interrupt is detected.<br>Indicates that the receive queue had an overflow when it receives the packet. If you transfers a partial packet to the application, the overflow status sets in RDES3[21]. This field is 0 when the application writes 1 to it. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Access restriction apply to this field. It automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 15-10<br>— | Reserved. |
| 9<br>ABPSIE | Average Bits Per Slot Interrupt Enable<br>Indicates whether an average bits per slot interrupt is enabled.<br>When this field is 1, it indicates that the MAC asserts the sbd_intr_o or mci_intr_o interrupt when you update the average bits per slot status.<br>When this field is 0, it indicates that the interrupt is not asserted for such an event.<br>0b - Disabled<br>1b - Enabled |
| 8<br>TXUIE | Transmit Queue Underflow Interrupt Enable<br>Enables or disables the transmit queue underflow interrupt.<br>When this field is 1, it enables the transmit queue underflow interrupt.<br>When this field becomes 0, it disables the transmit queue underflow interrupt.<br>0b - Disabled<br>1b - Enabled |
| 7-2<br>— | Reserved. |
| 1<br>ABPSIS | Average Bits Per Slot Interrupt Status<br>Indicates whether the average bits per slot interrupt status is detected.<br>When this field is 1, it indicates that the MAC has updated the ABS value. This field is 0 when the application writes 1 to it.<br>Access restriction apply to this field. It automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 0<br>TXUNFIS | Transmit Queue Underflow Interrupt Status<br>Indicates whether the transmit queue underflow interrupt status is detected.<br>Indicates that the transmit queue had an underflow when you transmits the packet. Suspends the transmission and sets an underflow error TDES3[2]. This field is 0 when the application writes 1 to this field. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Access restriction apply to this field. It automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |

## 72.17.235 MTL Rx Queue 1 Operation Mode (MTL_RxQ1_Operation_Mode)

### Offset

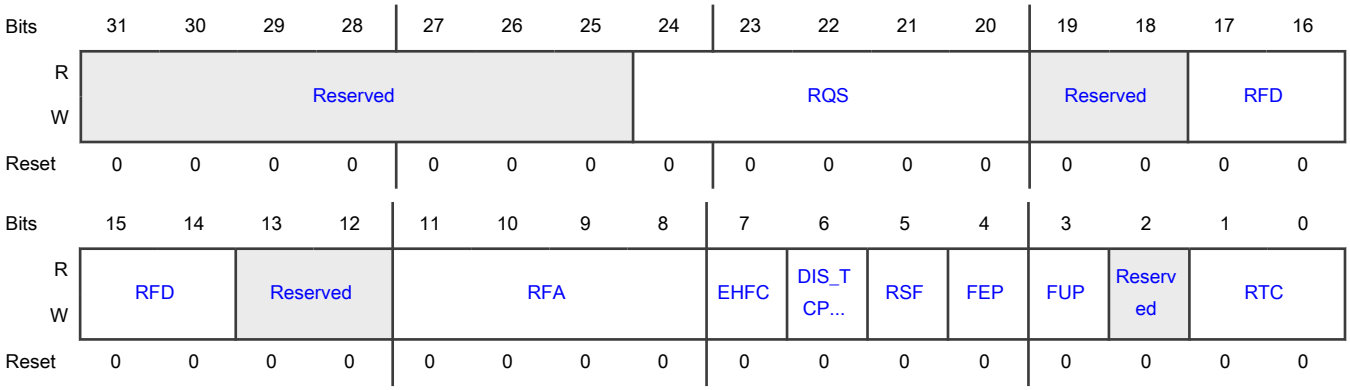
| Register | Offset |
|---|---|
| MTL_RxQ1_Operation_Mode | D70h |

### Function

Establishes the receive queue operating modes and command.

The MTL_RxQ1_Operation_Mode[RFA] and MTL_RxQ1_Operation_Mode[RFD] are not backward compatible with the RFA and RFD fields of 4.00a release.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved. |
| 24-20<br>RQS | Receive Queue Size<br>Indicates the size of the allocated receive queues in blocks of 256 bytes.<br>MTL_RxQ1_Operation_Mode[RQS] is read-write only if the number of receive queues is more than one, the |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | reset value is 0x0 and indicates 256 bytes size. This means that value of 0x0 = 256 bytes, 0x1 = 512 bytes and so on. You must program RQS [5:0] = 6'b001111 to allocate queue size of 4096 (4K) bytes. In general, the size of the Queue = (RQS+1)*256 bytes.<br>When the number of receive queues is one, the field is read-only and the reset value reflects the configured RX FIFO size in blocks of 256 bytes.<br>The field width depends on the receive memory size selected in your configuration. For example, if the memory size is 2048, the field width is 3 bits:<br>LOG2(2048/256) = LOG2(8) = 3 bits |
| 19-18<br>— | Reserved. |
| 17-14<br>RFD | Threshold for Deactivating Flow Control (in half-duplex and full-duplex modes)<br>Controls the threshold (fill-level of Rx queue) at which the flow control de-asserts after activation<br>0 - Full minus 1 KB, that is, FULL 1 KB<br>1 - Full minus 1.5 KB, that is, FULL 1.5 KB<br>2 - Full minus 2 KB, that is, FULL 2 KB<br>3 -Full minus 2.5 KB, that is, FULL 2.5 KB<br>...<br>14 - Full minus 8 KB, that is, FULL 8 KB<br>15 - Full minus 8.5 KB, that is, FULL 8.5 KB<br>The de-assertion is effective only when flow control asserts.<br>**NOTE**<br>Program the value in such a way that the threshold is a positive number.<br>When MTL_RxQ1_Operation_Mode[EHFC] = 1, these values are applicable only when the receive queue size which MTL_RxQ1_Operation_Mode[RQS] determines, is equal to or greater than 4 KB.<br>For a given queue size, the values ranges between 0 and the encoding for FULL minus (QSIZE - 0.5 KB) and all other values are illegal. Here the term FULL and QSIZE refers to the queue size which MTL_RxQ1_Operation_Mode[RQS] determines.<br>The field width depends on RX FIFO size selected during the configuration. Remaining bits are reserved and read only. |
| 13-12<br>— | Reserved. |
| 11-8<br>RFA | Threshold for Activating Flow Control (in half-duplex and full-duplex<br>Controls the threshold (fill-level of Rx queue) at which the flow control activates.<br>See MTL_RxQ1_Operation_Mode[RFD], for more information on encoding for this field. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 7<br>EHFC | Enable Hardware Flow Control<br>Enables or disables hardware flow control.<br>If this field is 1, it enables the flow control signal operation, based on the receive queue's fill-level.<br>When this field becomes 0, it disables the flow control operation.<br>0b - Disable<br>1b - Enable |
| 6<br>DIS_TCP_EF | Disable Dropping of TCP or IP Checksum Error Packets<br>Indicates whether the dropping of TCP or IP checksum error packets is enabled.<br>When this field is 1, it indicates that the MAC does not drop the packets which only have the errors that the receive checksum offload engine detects. Such packets have errors only in the encapsulated payload. MAC receives an Ethernet packet in which there are no errors (including FCS error).<br>When this field becomes 0, it indicates that all the error packets drop if MTL_RxQ1_Operation_Mode[FEP] resets.<br>0b - Enabled<br>1b - Disabled |
| 5<br>RSF | Receive Queue Store and Forward<br>Indicates whether the receive queue store and forward is enabled.<br>When this field is 1, it indicates that the module reads a packet from the receive queue only after the complete packet is written to it and ignores MTL_RxQ1_Operation_Mode[RTC].<br>When this field becomes 0, it indicates that the receive queue operates in the Threshold (cut-through) mode, subject to the threshold which MTL_RxQ1_Operation_Mode[RTC] specifies.<br>0b - Disabled<br>1b - Enabled |
| 4<br>FEP | Forward Error Packets<br>Indicates whether the forward error packets are enabled.<br>If this field becomes 0, it indicates that the receive queue drops the packets with an error status (CRC error, GMII_ER, watchdog timeout, or overflow). However, the packet does not drops, if the start byte (write) pointer of a packet is already transferred to the read controller side in Threshold mode.<br>When this field is 1, it indicates that all the packets except the runt error packets forward to the application or DMA. If MTL_RxQ1_Operation_Mode[RSF] is 1 and the receive queue overflows when a partial packet is written, the packet drops irrespective of the setting of this field. However, if MTL_RxQ1_Operation_Mode[RSF] resets and the receive queue overflows when a partial packet is written, you may forward a partial packet to the application or DMA.<br>0b - Disabled<br>1b - Enabled |
| 3 | Forward Undersized Good Packets |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| FUP | Indicates whether the forward undersized good packets are enabled.<br>When this field is 1, it indicates that the receive queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC.<br>When this field becomes 0, it indicates that the receive queue drops all the packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx Threshold, for example, RTC = 01.<br>0b - Disabled<br>1b - Enabled |
| 2<br>— | Reserved. |
| 1-0<br>RTC | Receive Queue Threshold Control<br>Controls the threshold level of the MTL Rx queue (in bytes):<br>The received packet is transferred to the application or DMA when the packet size within the MTL receive queue is larger than the threshold. Also, the full packets with length less than the threshold are automatically transferred.<br>This field is valid only when MTL_RxQ1_Operation_Mode[RSF] = 0.<br>This field is ignored when MTL_RxQ1_Operation_Mode[RSF] = 1.<br>00b - 64<br>01b - 32<br>10b - 96<br>11b - 128 |

## 72.17.236 MTL Rx Queue 1 Missed Packet Overflow Counter (MTL_RxQ1_Missed_Packet_Overflow_Cnt)

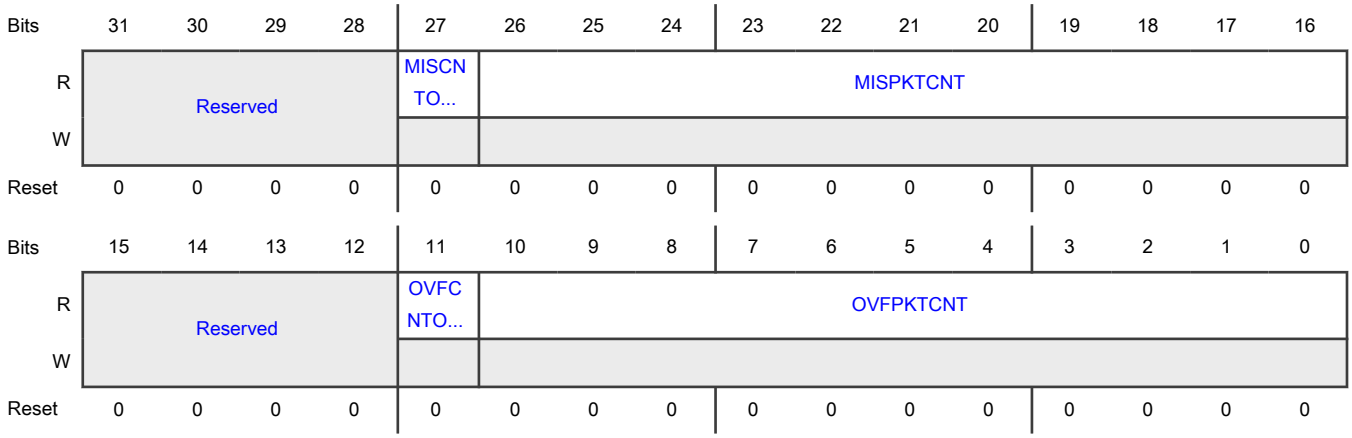
### Offset

| Register | Offset |
|---|---|
| MTL_RxQ1_Missed_Packet_Overflow_Cnt | D74h |

### Function

Contains the counter for packets missed because of receive queue packet flush and packets discarded because of receive queue overflow.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27<br>MISCNTOVF | Missed Packet Counter Overflow Bit<br>Indicates whether the missed packet counter overflow is detected.<br>When this field is 1, it indicates that the receive queue missed packet counter has crossed the maximum limit.<br>Access restriction apply to this field. It clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 26-16<br>MISPKTCNT | Missed Packet Counter<br>Indicates the number of packets the module has missed because the application asserted ari_pkt_flush_i[] for this queue. This counter resets when this register is read with mci_be_i[0] at 1b1.<br>This counter increases by 1 when the DMA discards the packet because of buffer unavailability.<br>Access restriction apply to this field. It clears on read and automatically becomes 1 on an internal event occurrence. |
| 15-12<br>— | Reserved. |
| 11<br>OVFCNTOVF | Overflow Counter Overflow Bit<br>Indicates whether the counter overflow status is detected.<br>When this field is 1, it indicates that the receive queue overflow packet counter field has crossed the maximum limit. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Access restriction apply to this field. It clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not detected<br>1b - Detected |
| 10-0<br>OVFPKTCNT | Overflow Packet Counter<br>Indicates the number of packets which the module discards because of receive queue overflow. This counter increments each time the module discards an incoming packet because of overflow. It resets when this register is read with mci_be_i[0] at 1'b1.<br>Access restriction apply to this field. It clears on read and automatically becomes 1 on an internal event occurrence. |

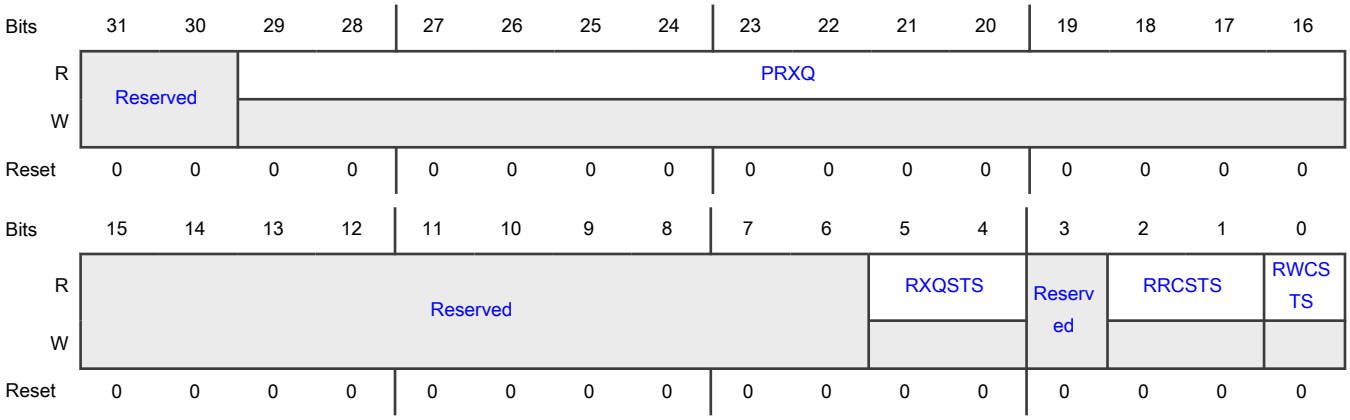
## 72.17.237 MTL Rx Queue 1 Debug (MTL_RxQ1_Debug)

### Offset

| Register | Offset |
|---|---|
| MTL_RxQ1_Debug | D78h |

### Function
Provides the debug status of various blocks related to the receive queue.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-30 | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 29-16<br>PRXQ | Number of Packets in Receive Queue<br>Indicates the current number of packets in the receive queue. The theoretical maximum field value is 256KB/16B = 16K packets, that is, Max_Queue_Size/Min_Packet_Size. |
| 15-6<br>— | Reserved. |
| 5-4<br>RXQSTS | MTL Rx Queue Fill-Level Status<br>Provides the status of the receive queue fill-level.<br>00b - Rx Queue empty<br>01b - Rx Queue fill-level below flow-control deactivate threshold<br>10b - Rx Queue fill-level above flow-control activate threshold<br>11b - Rx Queue full |
| 3<br>— | Reserved. |
| 2-1<br>RRCSTS | MTL Rx Queue Read Controller State<br>Provides the state of the receive queue read controller.<br>00b - Idle state<br>01b - Reading packet data<br>10b - Reading packet status (or timestamp)<br>11b - Flushing the packet data and status |
| 0<br>RWCSTS | MTL Rx Queue Write Controller Active Status<br>Indicates whether the MTL receive queue write controller active status is detected.<br>When this field is 1, it indicates that the MTL receive queue write controller is active, and it is transferring a received packet to the receive queue.<br>0b - Not detected<br>1b - Detected |

## 72.17.238 MTL Rx Queue 1 Control (MTL_RxQ1_Control)

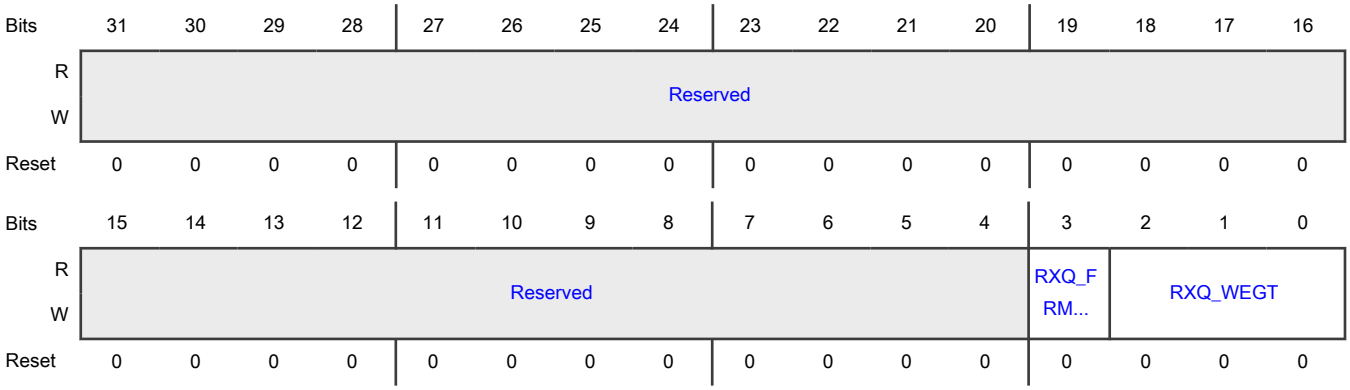
**Offset**

| Register | Offset |
|---|---|
| MTL_RxQ1_Control | D7Ch |

### Function

Controls the receive arbitration and passing of received packets to the application.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-4<br>— | Reserved. |
| 3<br>RXQ_FRM_ARBIT | Receive Queue Packet Arbitration<br>Indicates whether the receive queue packet arbitration is enabled.<br>When this field is 1, it indicates that the module drives the packet data to the ARI interface such that the entire packet data of currently-selected queue transmits before switching to other queue.<br>When this field becomes 0, it indicates that the module drives the packet data to the ARI interface such that the following amount of data of currently-selected queue transmits before switching to other queue:<br>• PBL amount of data (indicated by ari_qN_pbl_i[]), or<br>• Complete data of a packet<br>The status and the timestamp are not a part of the PBL data. Therefore, the module drives the complete status (including timestamp status) during the packet's first PBL request (in store-and-forward mode) or the last PBL request (in Threshold mode).<br>0b - Disabled<br>1b - Enabled |
| 2-0<br>RXQ_WEGT | Receive Queue Weight<br>Indicates the weight assigned to the receive queue 0. Program this field with one value less than the required weight, that is reset value of 0 indicates weight of 1, value of 1 indicates weight of 2, and so on. The weight is used as the number of continuous PBL or packets requests (depending on the RXQ_FRM_ARBIT) allocated to the queue in one arbitration cycle.<br>Note: The change in value of RXQ_WEGT takes effect only after the completion of current service round or when there is change from RAA=SP to RAA=WSP algorithm. This approach is taken so that there is smooth transition. For the RXQ_WEGT value to take effect at the start, the MTL_RxQ(#i)_Control registers must be programmed before the MTL_Operation_Mode register. |

## 72.17.239 DMA Mode (DMA_Mode)

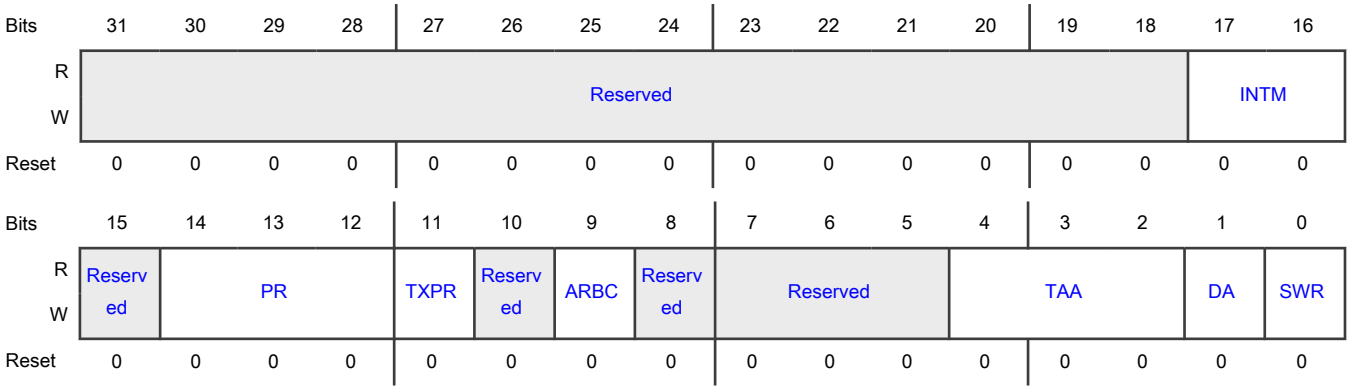
### Offset

| Register | Offset |
|---|---|
| DMA_Mode | 1000h |

### Function

Establishes the bus operating modes for the DMA.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-18<br>— | Reserved. |
| 17-16<br>INTM | Interrupt Mode<br>Defines the interrupt mode of module.<br>The behavior of the following outputs changes depending on these settings:<br>sbd_perch_tx_intr_o[] (Transmit per channel interrupt)<br>sbd_perch_rx_intr_o[] (Receive per channel interrupt)<br>sbd_intr_o (Common interrupt)<br>It also changes the behavior of the RI/TI bits in DMA_CH0_Status.<br>00 - sbd_perch_* are pulse signals for each TX/RX packet transfer completion events (irrespective of whether corresponding interrupts are enabled) for which IOC bits are enabled in descriptor. sbd_intr_o is also asserted when corresponding interrupts are enabled and cleared only when software clears the corresponding RI/TI status bits.<br>01 - sbd_perch_* are level signals asserted on TX/RX packet transfer completion event when corresponding interrupts are enabled and de-asserted when the software clears the corresponding RI/TI status bits. The sbd_intr_o is not asserted for these TX/RX packet transfer completion events. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 10 - sbd_perch_* are level signals asserted on TX/RX packet transfer completion event when corresponding interrupts are enabled and de-asserted when the software clears the corresponding RI/TI status bits. However, the signal is asserted again if the same event occurred again before it was cleared. The sbd_intr_o is not asserted for these TX/RX packet transfer completion events.<br>11 - Reserved<br>See table "Transfer Complete Interrupt Behavior" for more information.<br>00b - See above description<br>01b - See above description<br>10b - See above description<br>11b - Reserved |
| 15<br>— | Reserved. |
| 14-12<br>PR | Priority Ratio<br>Controls the priority ratio in weighted round-robin arbitration between the Rx DMA and Tx DMA. These fields are valid only when DMA_Mode[DA] resets. The priority ratio is Rx:Tx or Tx:Rx depending on whether DMA_Mode[TXPR] = 1 or it resets.<br>000b - The priority ratio is 1:1<br>001b - The priority ratio is 2:1<br>010b - The priority ratio is 3:1<br>011b - The priority ratio is 4:1<br>100b - The priority ratio is 5:1<br>101b - The priority ratio is 6:1<br>110b - The priority ratio is 7:1<br>111b - The priority ratio is 8:1 |
| 11<br>TXPR | Transmit Priority<br>Indicates whether the transmit priority is enabled.<br>When this field is 1, it indicates that the Tx DMA has higher priority than the Rx DMA during arbitration for the system-side bus.<br>0b - Disabled<br>1b - Enabled |
| 10<br>— | Reserved. |
| 9<br>ARBC | Is reserved for NXP internal use. This field must always be 0 unless instructed by NXP. Writing 1 to this field may cause unexpected behavior. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - NXP reserved field disabled<br>1b - NXP reserved field enabled up on NXP request |
| 8<br>— | Reserved. |
| 7-5<br>— | Reserved. |
| 4-2<br>TAA | Transmit Arbitration Algorithm<br>Selects the arbitration algorithm for the transmit side when you select multiple transit DMAs.<br>000b - Fixed priority (Channel 0 has the lowest priority and the last channel has the highest priority)<br>001b - Weighted Strict Priority (WSP)<br>010b - Weighted Round-Robin (WRR)<br>011b - Reserved (for 3'b011 to 3'b111) |
| 1<br>DA | DMA Tx or Rx Arbitration Scheme<br>Specifies the arbitration scheme between the transmit and receive paths of all channels.<br>0 - Weighted round-robin with Rx:Tx or Tx:Rx<br>The priority between the paths is according to the priority specified in bits[14:12] and the priority weight is specified in DMA_Mode[TXPR].<br>1 - Fixed Priority<br>The transit path has priority over the receive path when DMA_Mode[TXPR] = 1. Otherwise, the receive path has priority over the transit path.<br>0b - Weighted Round-Robin with Rx:Tx or Tx:Rx<br>1b - Fixed Priority |
| 0<br>SWR | Software Reset<br>Indicates whether the software reset is enabled.<br>When this field is 1, it indicates that the MAC and the DMA controller resets the logic and all internal registers of the DMA, MTL, and MAC. This field automatically clears after the reset operation completes in all module clock domains. A value of zero should be read in this field before reprogramming any module register.<br>When this field is 1, it indicates that this field must read at least 4 CSR clock cycles.<br>**NOTE**<br>The reset operation completes only when all resets in all active clock domains de-assert. Therefore, to complete software reset, it is essential that all PHY input clocks (applicable for the selected PHY interface) are present. The time to complete the software reset operation depends on the frequency of the slowest active clock. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Access restriction apply to this field. It clears automatically. Writing 1 sets this field and writing 0 has no effect.<br>0b - Disabled<br>1b - Enabled |

## 72.17.240 DMA System Bus Mode (DMA_SysBus_Mode)

### Offset

| Register | Offset |
|---|---|
| DMA_SysBus_Mode | 1004h |

### Function

Controls the behavior of the AHB or AXI master. It mainly controls burst splitting and the number of outstanding requests.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | Reserved | Reserved | | | | Reserved | | Reserved | | | | Reserved | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | RB | MB | Reserved | AAL | Reserved | Reserved | Reserved | | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | FB |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved. |
| 30<br>— | Reserved. |
| 29-26<br>— | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 25-24<br>— | Reserved. |
| 23-18<br>— | Reserved. |
| 17-16<br>— | Reserved. |
| 15<br>RB | Rebuild INCRx Burst<br>Indicates whether the rebuild INCRx burst is enabled.<br>When this field is 1 and the AHB master gets SPLIT, RETRY, or Early Burst Termination (EBT) response, it indicates that the AHB master interface rebuilds the pending beats of any initiated burst transfer with INCRx and SINGLE transfers. By default, the AHB master interface rebuilds pending beats of an EBT with an unspecified (INCR) burst.<br>0b - Disabled<br>1b - Enabled |
| 14<br>MB | Mixed Burst<br>Indicates whether the mixed burst is enabled.<br>When this field is 1 and DMA_SysBus_Mode[FB] is 0, it indicates that the AHB master performs an undefined bursts transfers (INCR) for burst length of 16 or more. The AHB master performs fixed burst transfers (INCRx and SINGLE), for burst length of 16 or less.<br>0b - Disabled<br>1b - Enabled |
| 13<br>— | Reserved. |
| 12<br>AAL | Address-Aligned Beats<br>Indicates whether the address-aligned beats are enabled.<br>When this field is 1, it indicates that the EQOS-AXI or EQOS-AHB master performs address-aligned burst transfers on read and write channels.<br>0b - Disabled<br>1b - Enabled |
| 11<br>— | Reserved. |
| 10<br>— | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 9-8<br>— | Reserved. |
| 7<br>— | Reserved. |
| 6<br>— | Reserved. |
| 5<br>— | Reserved. |
| 4<br>— | Reserved. |
| 3<br>— | Reserved. |
| 2<br>— | Reserved. |
| 1<br>— | Reserved. |
| 0<br>FB | Fixed Burst Length<br>Indicates whether the fixed burst length is enabled.<br>When this field is 1, it indicates that the AHB master initiates burst transfers of specified length (INCRx or SINGLE).<br>When this field is 0, it indicates that the AHB master initiates transfers of unspecified length (INCR) or SINGLE transfers.<br>0b - Disabled<br>1b - Enabled |

## 72.17.241 DMA Interrupt Status (DMA_Interrupt_Status)

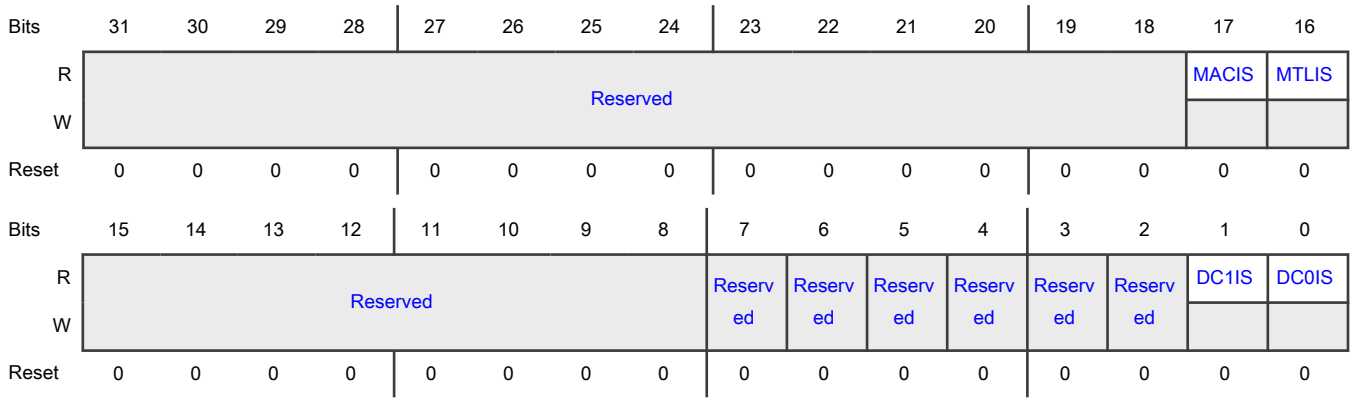
**Offset**

| Register | Offset |
|---|---|
| DMA_Interrupt_Status | 1008h |

### Function

The application reads this Interrupt Status register during interrupt service routine or polling to determine the interrupt status of DMA channels, MTL queues, and the MAC.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-18<br>— | Reserved. |
| 17<br>MACIS | MAC Interrupt Status<br>Indicates whether the MAC interrupt status is detected.<br>You must read the corresponding register in the MAC to get the exact cause of the interrupt and clear its source, to reset this field to 1'b0.<br>0b - Not detected<br>1b - Detected |
| 16<br>MTLIS | MTL Interrupt Status<br>Indicates whether the MTL interrupt status is detected.<br>You must read the corresponding register in the MTL to get the exact cause of the interrupt and clear its source, to reset this field to 1'b0.<br>0b - Not detected<br>1b - Detected |
| 15-8<br>— | Reserved. |
| 7<br>— | Reserved. |
| 6 | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 5<br>— | Reserved. |
| 4<br>— | Reserved. |
| 3<br>— | Reserved. |
| 2<br>— | Reserved. |
| 1<br>DC1IS | DMA Channel 1 Interrupt Status<br>Indicates whether the DMA channel 1 interrupt status is detected.<br>You must read the corresponding register in DMA channel 1 to get the exact cause of the interrupt and clear its source, to reset this field to 1'b0.<br>0b - Not detected<br>1b - Detected |
| 0<br>DC0IS | DMA Channel 0 Interrupt Status<br>Indicates whether the DMA channel 0 interrupt status is detected.<br>You must read the corresponding register in DMA channel 0 to get the exact cause of the interrupt and clear its source, to reset this field to 1'b0.<br>0b - Not detected<br>1b - Detected |

## 72.17.242 DMA Debug Status 0 (DMA_Debug_Status0)

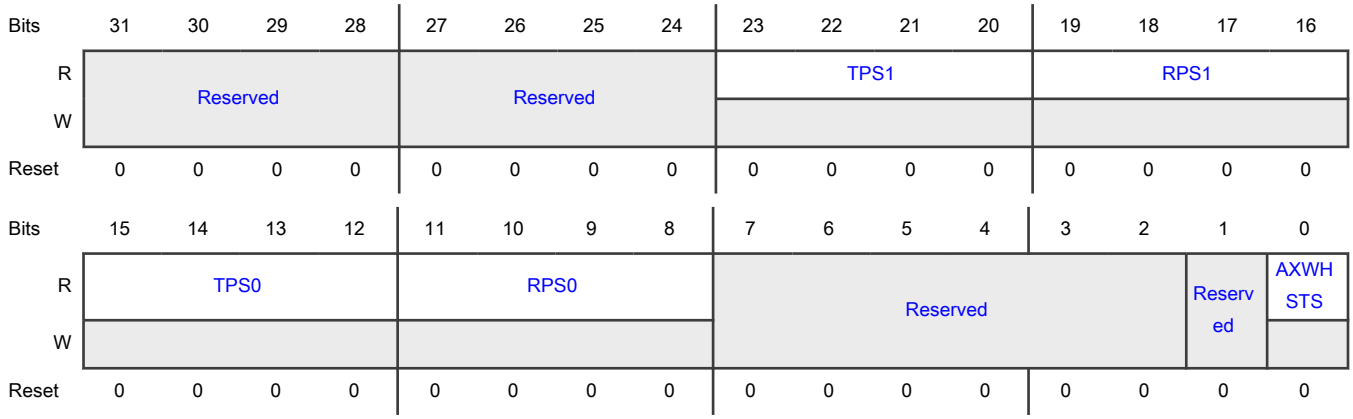
### Offset

| Register | Offset |
|---|---|
| DMA_Debug_Status0 | 100Ch |

### Function

Provides the receive and transmit process status for DMA Channel 0-Channel 2 for debugging purpose.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved. |
| 27-24<br>— | Reserved. |
| 23-20<br>TPS1 | DMA Channel 1 Transmit Process State<br>Indicates the transmit DMA FSM state for channel 1.<br>The MSB of this field always returns 0. This field does not generate an interrupt.<br>0000b - Stopped (Reset or stop transmit command issued)<br>0001b - Running (Fetching transmit transfer descriptor)<br>0010b - Running (Waiting for status)<br>0011b - Running (Reading data from system memory buffer and queuing it to the transmit buffer (Tx FIFO))<br>0100b - Timestamp write state<br>0101b - Reserved for future use<br>0110b - Suspended (Transmit descriptor unavailable or transmit buffer underflow)<br>0111b - Running (Closing transmit descriptor) |
| 19-16<br>RPS1 | DMA Channel 1 Receive Process State<br>Indicates the receive DMA FSM state for channel 1.<br>The MSB of this field always returns 0. This field does not generate an interrupt.<br>0000b - Stopped (Reset or Stop receive command issued)<br>0001b - Running (Fetching receive transfer descriptor)<br>0010b - Reserved for future use |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0011b - Running (Waiting for receive packet)<br>0100b - Suspended (Receive descriptor unavailable)<br>0101b - Running (Closing the receive descriptor)<br>0110b - Timestamp write state<br>0111b - Running (Transferring the received packet data from the receive buffer to the system memory) |
| 15-12<br>TPS0 | DMA Channel 0 Transmit Process State<br>Indicates the transmit DMA FSM state for channel 0.<br>The MSB of this field always returns 0. This field does not generate an interrupt.<br>0000b - Stopped (Reset or stop transmit command issued)<br>0001b - Running (Fetching transmit transfer descriptor)<br>0010b - Running (Waiting for status)<br>0011b - Running (Reading data from system memory buffer and queuing it to the transmit buffer (Tx FIFO))<br>0100b - Timestamp write state<br>0101b - Reserved for future use<br>0110b - Suspended (Transmit descriptor unavailable or transmit buffer underflow)<br>0111b - Running (Closing transmit descriptor) |
| 11-8<br>RPS0 | DMA Channel 0 Receive Process State<br>Indicates the receive DMA FSM state for channel 0.<br>The MSB of this field always returns 0. This field does not generate an interrupt.<br>0000b - Stopped (Reset or stop receive command issued)<br>0001b - Running (Fetching receive transfer descriptor)<br>0010b - Reserved for future use<br>0011b - Running (Waiting for receive packet)<br>0100b - Suspended (Receive descriptor unavailable)<br>0101b - Running (Closing the receive descriptor)<br>0110b - Timestamp write state<br>0111b - Running (Transferring the received packet data from the receive buffer to the system memory) |
| 7-2<br>— | Reserved. |
| 1 | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 0<br>AXWHSTS | AHB Master Status<br>Indicates whether the AXI master write channel or AHB master status is detected.<br>When this field is 1, it indicates that the AHB master FSMs are in the non-idle state.<br>0b - Not detected<br>1b - detected |

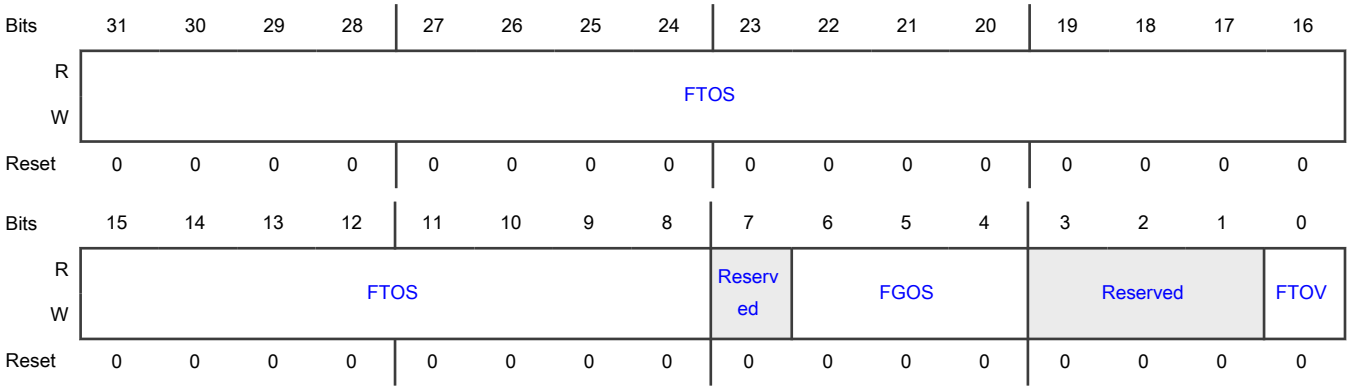
## 72.17.243 DMA TBS Control (DMA_TBS_CTRL)

### Offset

| Register | Offset |
|---|---|
| DMA_TBS_CTRL | 1050h |

### Function
Controls the TBS attributes.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-8<br>FTOS | Fetch Time Offset<br>Deduct the value in units of 256 nanoseconds, from the launch time to compute the fetch Time.<br>Max value: 999,999,999 nanosecond, must be smaller than CTR-1 value when ESTM mode = 1, because this value is a modulo CTR value. |
| 7 | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 6-4<br>FGOS | Fetch GSN Offset<br>Deduct the number GSN slots from the launch GSN to compute the fetch GSN. The value is valid only when DMA_TBS_CTRL[FTOV] = 1. |
| 3-1<br>— | Reserved. |
| 0<br>FTOV | Fetch Time Offset Valid<br>Indicates whether the fetch time offset is valid.<br>When this field is 1, it indicates that DMA_TBS_CTRL[FTOS] is valid. When this field is not 1, it indicates that the fetch offset is not valid and the DMA engine can fetch the frames from host memory without any time restrictions.<br>0b - Invalid<br>1b - Valid |

## 72.17.244 DMA Safety Interrupt Status (DMA_Safety_Interrupt_Status)

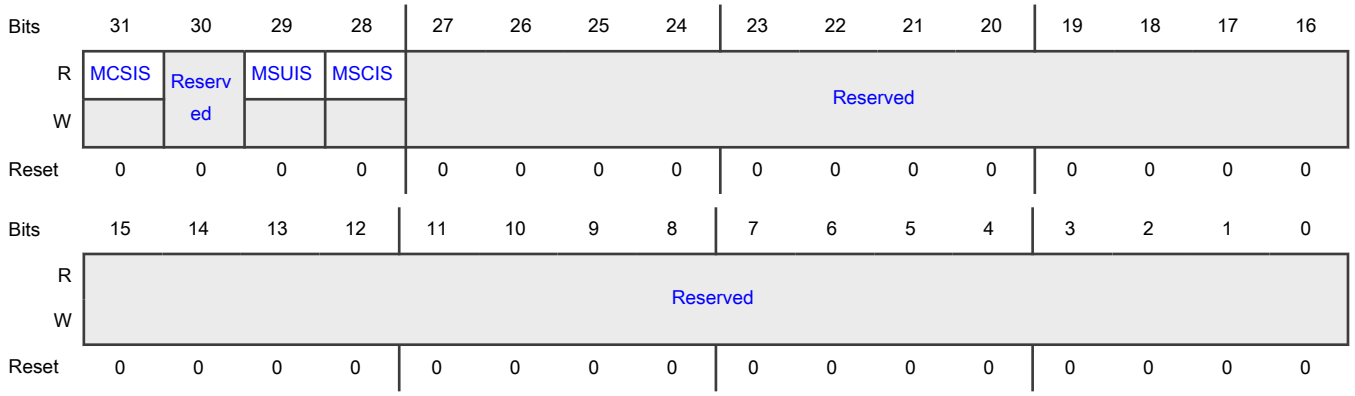
### Offset

| Register | Offset |
|---|---|
| DMA_Safety_Interrupt_Status | 1080h |

### Function

Indicates summary (whether error occurred in DMA/MTL/MAC and correctable/uncorrectable) of the automotive safety related error interrupts.

### Diagram



**Fields**

| Field | Function |
|---|---|
| 31<br>MCSIS | MAC Safety Uncorrectable Interrupt Status<br>Indicates whether the MAC safety uncorrectable interrupt status is detected.<br>Indicates that an uncorrectable safety related interrupt is 1 in the MAC module. When this field is 1, it indicates that you must read the MAC_DPP_FSM_Interrupt_Status, to get the cause of the safety interrupt in MAC.<br>0b - Not detected<br>1b - Detected |
| 30<br>— | Reserved. |
| 29<br>MSUIS | MTL Safety Uncorrectable Error Interrupt Status<br>Indicates whether the MTL safety uncorrectable error interrupt status is detected.<br>Indicates an uncorrectable error interrupt event in MTL. To get exact cause of the interrupt you must read the MTL_Safety_Interrupt_Status.<br>0b - Not detected<br>1b - Detected |
| 28<br>MSCIS | MTL Safety Correctable Error Interrupt Status<br>Indicates whether the MTL safety correctable error interrupt status is detected.<br>Indicates a correctable error interrupt event in MTL. To get the exact cause of the interrupt you must read the MTL_Safety_Interrupt_Status.<br>0b - Not detected<br>1b - Detected |
| 27-0<br>— | Reserved. |

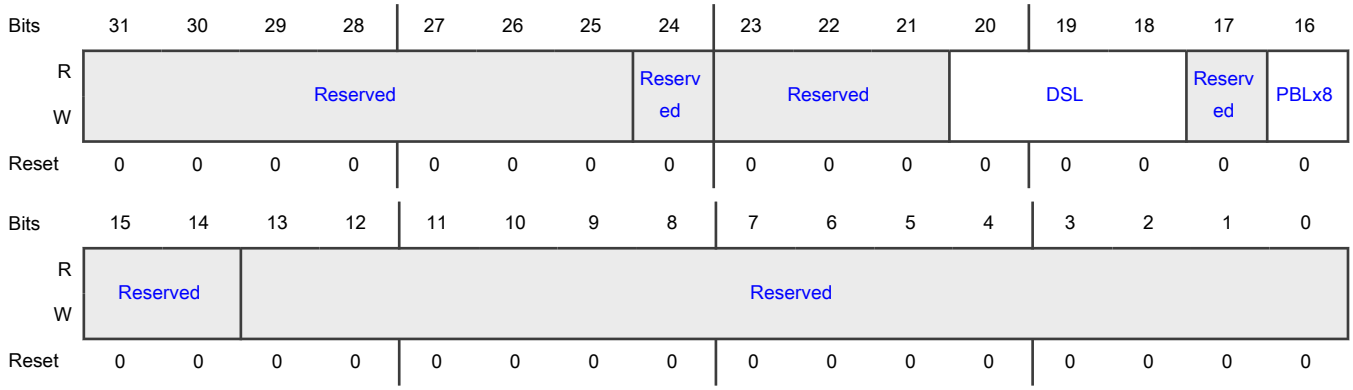
## 72.17.245 DMA Channel 0 Control (DMA_CH0_Control)

**Offset**

| Register | Offset |
|---|---|
| DMA_CH0_Control | 1100h |

**Function**
Specifies the MSS value for segmentation, length to skip between two descriptors, and also the features such as header splitting and 8xPBL mode.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved. |
| 24<br>— | Reserved. |
| 23-21<br>— | Reserved. |
| 20-18<br>DSL | Descriptor Skip Length<br>Specifies the Word, Dword, or Lword number (depending on the 32-bit, 64-bit, or 128-bit bus) to skip between two unchained descriptors. The address skipping starts from the end of the current descriptor to the start of the next descriptor.<br>The DMA assumes the descriptor table as contiguous, when the DSL value is equal to zero. |
| 17<br>— | Reserved. |
| 16<br>PBLx8 | 8xPBL mode<br>Indicates whether 8xPBL mode is enabled.<br>When this field is 1, it indicates that the PBL value programmed in Bits[21:16] in DMA_CH(#i)_Tx_Control and Bits[21:16] in DMA_CH(#i)_Rx_Control is multiplied by eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value.<br>0b - Disabled<br>1b - Enabled |
| 15-14<br>— | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 13-0<br>— | Reserved. |

## 72.17.246 DMA Channel Tx Control (DMA_CH0_Tx_Control)

### Offset

| Register | Offset |
|---|---|
| DMA_CH0_Tx_Control | 1104h |

### Function

Controls the transmit features such as PBL, TCP segmentation, and transmit channel weights.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | EDSE | Reserved | | | | Reserved | ETIC | TxPBL | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | Reserved | | Reserved | Reserved | | | | | | | OSF | TCW | | | ST |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved. |
| 28<br>EDSE | Enhanced Descriptor Enable<br>Indicates whether an enhanced descriptor is enabled.<br>When this field is 1, it indicates that the corresponding channel uses 32 bytes enhanced descriptors for both normal and context descriptors.<br>When this field becomes 0, it indicates that the corresponding channel uses the 16 bytes descriptors.<br>0b - Disabled<br>1b - Enabled |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 27-24<br>— | Reserved. |
| 23<br>— | Reserved. |
| 22<br>ETIC | Early Transmit Interrupt Control<br>Indicates whether an early transmit interrupt is enabled.<br>When this field is 1, it indicates that an early transmit interrupt (ETI) status = 1 after the data transfer from buffers of a transmit descriptor completes in which IOC bit (TDES2[31]) = 1.<br>When this field becomes 1, it indicates that ETI = 1 only after a complete packet transfers to the MTL TX FIFO memory.<br>0b - Disabled<br>1b - Enabled |
| 21-16<br>TxPBL | Transmit Programmable Burst Length<br>Indicates the maximum number of beats to transfer in one DMA block data transfer. DMA always attempts max burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of these values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior.<br>Perform these steps to transfer more than 32 beats:<br>1. Set DMA_CH0_Control[PBLx8].<br>2. Set the TxPBL.<br>**NOTE**<br>The maximum value of TxPBL must be less than or equal to half the transit queue size (TQS field of MTL_TxQ(#i)_Operation_Mode register) in terms of beats. This is required so that the transit queue has space to store at least another TxPBL worth of data while the MTL Tx queue controller transfers data to MAC. For example, in 64-bit data width configurations the total locations in transit queue of size 512 bytes is 64. You must program TxPBL and 8xPBL to less than or equal to 32. |
| 15<br>— | Reserved. |
| 14-13<br>— | Reserved. |
| 12<br>— | Reserved. |
| 11-5<br>— | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 4<br>OSF | Operate on Second Packet<br>Indicates whether the operation on second packet is enabled.<br>When this field is 1, it instructs DMA to process the second packet of the transmit data even before you receive the status for the first packet.<br>0b - Disabled<br>1b - Enabled |
| 3-1<br>TCW | Transmit Channel Weight<br>Indicates the weight assigned to the corresponding transmit channel. When reset completes, this field is 0 for all the channels by default, which results in equal weights to all channels. |
| 0<br>ST | Start or Stop Transmission Command<br>Indicates whether to start or stop the transmission command.<br>When this field is 1, it indicates that the transmission is in the running state. DMA checks the transmit list at the current position for a packet to be transmitted.<br>DMA tries to acquire descriptor from either of these positions:<br>• The current position in the list. This is the base address of the transmit list to which DMA_CH0_TxDesc_List_Address sets.<br>• The position at which the transmission was previously stopped.<br>The transmission enters the suspended state and DMA_CH0_Status[TBU] = 1, if DMA does not own the current descriptor. The start transmission command is effective only when the transmission stops. If the command is issued before setting DMA_CH0_TxDesc_List_Address, you cannot predict the DMA behavior.<br>When this field becomes 0, it indicates that the transmission process is placed in the stopped state when the transmission of the current packet completes . The next descriptor position in the transmit list is saved, and it becomes the current position when you restart the transmission. To change the list address, you must program DMA_CH0_TxDesc_List_Address with a new value when this field becomes 0.You can consider the new value when this field is 1 again. The stop transmission command is effective only when the transmission of the current packet completes or the transmission is in the suspended state.<br>0b - Stop<br>1b - Start |

## 72.17.247 DMA Channel Rx Control (DMA_CH0_Rx_Control)

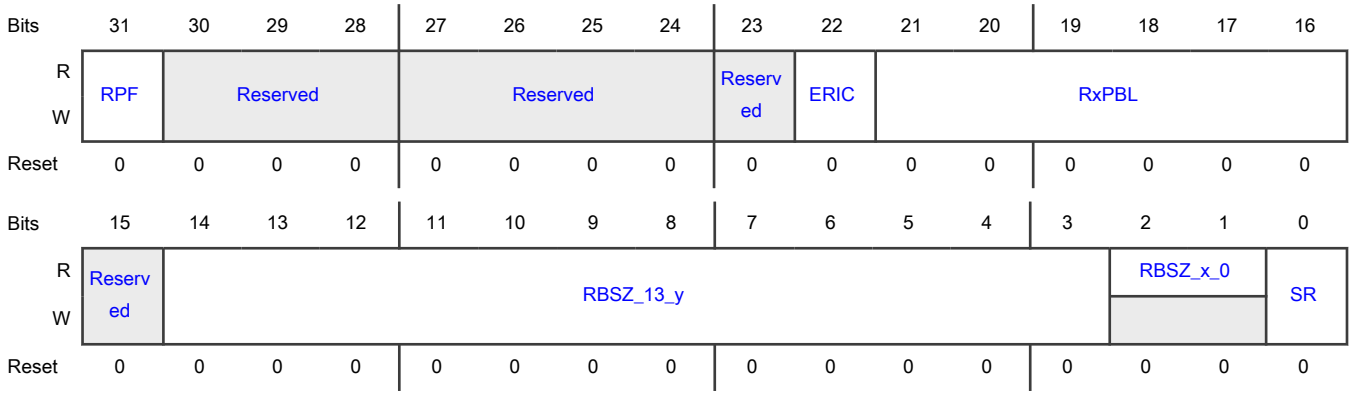
### Offset

| Register | Offset |
|---|---|
| DMA_CH0_Rx_Control | 1108h |

### Function

Controls the receive features such as PBL, buffer size, and extended status.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>RPF | Rx Packet Flush<br>Indicates whether the receive packet flush is enabled.<br>When this field is 1, it indicates that the module automatically flushes the packet from the receive queues destined to this DMA receive channel, when it stops. When this field remains 1 and the software driver re-starts DMA, the packets residing in the receive queues that were received when this RxDMA stops and flushes out. The packets that MAC receives after the RxDMA re-starts, route to the RxDMA. The flushing takes place on the read side of the receive queue.<br>When this field is 0, it indicates that the module do not flush the packet in the receive queue destined to this RxDMA channel when it is in STOP state. This may cause head-of-line blocking in the corresponding receive queue.<br>**NOTE**<br>The packet flow from a receive DMA channel to the application by writing 1 to this field stops, only when there is one-to-one mapping of receive queue to receive DMA channels. In Dynamic mapping mode, writing 1 to this field in any DMA_CH(#i)_Rx_Control register might flush packets from an unintended receive queues which are destined to the stopped receive DMA channel.<br>0b - Disabled<br>1b - Enabled |
| 30-28<br>— | Reserved. |
| 27-24<br>— | Reserved. |
| 23<br>— | Reserved. |
| 22 | Early Receive Interrupt Control |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| ERIC | Indicates whether an early receive interrupt is enabled.<br>When this field is 1, it indicates that the early receive interrupt (ERI) status sets after every burst transfer of data from the Rx DMA to the buffer completes.<br>When this field becomes 0, it indicates that ERI sets only after a complete buffer is filled by the RxDMA.<br>0b - Disabled<br>1b - Enabled |
| 21-16<br>RxPBL | Receive Programmable Burst Length<br>Indicates the maximum number of beats to be transferred in one DMA block data transfer. DMA always attempts max burst as specified in PBL every time it starts a burst transfer on the application bus. You can program PBL with any of these values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior.<br>To transfer more than 32 beats, perform the following steps:<br>1. Set DMA_CH0_Control[PBLx8].<br>2. Set the RxPBL.<br>**NOTE**<br>The maximum value of RxPBL must be less than or equal to half the receive queue size (RQS field of MTL_RxQ(#i)_Operation_Mode register) in terms of beats. This is required so that the receive queue has space to store at least another Rx PBL worth of data while the receive DMA transfers a block of data. For example, in 64-bit data width configurations the total locations in receive queue of size 512 bytes is 64. You must program RxPBL and 8xPBL to less than or equal to 32. |
| 15<br>— | Reserved. |
| 14-3<br>RBSZ_13_y | Receive Buffer size High<br>RBSZ[13:0] splits into two fields, higher field is RBSZ_13_y and lower field is RBSZ_x_0. RBSZ[13:0] field indicates the receive buffers size specified in bytes. The maximum buffer size is limited to 16K bytes. The buffer size is applicable to payload buffers when split headers are enabled.<br>**NOTE**<br>The buffer size must be a multiple of 4, 8, or 16 depending on the data bus widths (32-bit, 64-bit, or 128-bit respectively). This is required even if the buffer address pointer value is not aligned with data bus width. Hence the lower RBSZ_x_0 fields are read-only and the value is considered as all-zero. Thus RBSZ_13_y field indicates the buffer size in terms of locations (with the width same as bus-width). |
| 2-1<br>RBSZ_x_0 | Receive Buffer size Low<br>RBSZ[13:0] splits into two fields RBSZ_13_y and RBSZ_x_0. The RBSZ_x_0 is the lower field whose width is based on data bus width of the configuration.<br>The field width is of 2, 3, or 4 bits for 32-bit, 64-bit, or 128-bit data bus width respectively. This field is read-only (RO). |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 0<br>SR | Start or Stop Receive<br>Indicates whether to start or stop receive.<br>When this field is 1, it indicates that the DMA tries to acquire the descriptor from the receive list and processes the incoming packets.<br>DMA tries to acquire descriptor from either of these positions:<br>• The current position in the list. This is the address which DMA_CH0_RxDesc_List_Address sets.<br>• The position at which the receive process was previously stopped.<br>If DMA does not own the current descriptor, the reception is suspended and DMA_CH0_Status[RBU] = 1. The start receive command is effective only when the reception stops. If the command is issued before setting DMA_CH0_RxDesc_List_Address, the DMA behavior is unpredictable.<br>When this field becomes 0, it indicates that the receive DMA operation stops after the transfer of the current packet. The next descriptor position in the receive list is saved, and it becomes the current position after the receive process restarts. The stop receive command is effective only when the receive process is in the running (waiting for receive packet) or suspended state.<br>0b - Stop<br>1b - Start |

## 72.17.248 DMA Channel 0 Tx Descriptor List Address (DMA_CH0_TxDesc_List_Address)

### Offset

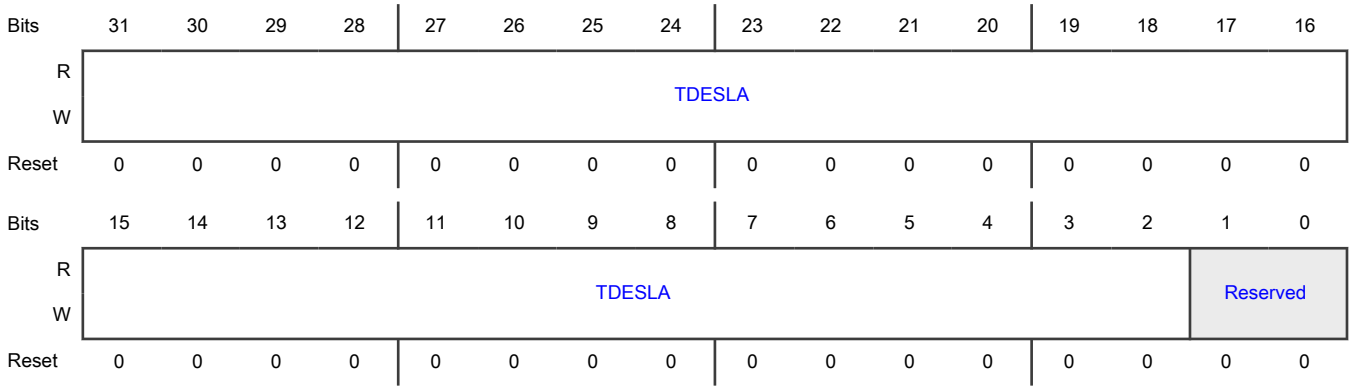
| Register | Offset |
|---|---|
| DMA_CH0_TxDesc_List_Address | 1114h |

### Function

Specifies DMA to the start of transmit descriptor list. The descriptor lists reside in the physical memory space of the application and must be Word, Dword, or Lword-aligned (for 32-bit, 64-bit, or 128-bit data bus). DMA internally converts it to bus width aligned address by making the corresponding LSB low.

You can write to this register only when the transit DMA stops, that is, DMA_CH0_Tx_Control[ST] = 0. When stopped, write this register with a new descriptor list address. When DMA_CH0_Tx_Control[ST] = 1, DMA uses the newly-programmed descriptor base address. If this register does not change when DMA_CH0_Tx_Control[ST] = 0, DMA uses the descriptor address where it was stopped earlier.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>TDESLA | Start of Transmit List<br>Contains the base address of the first descriptor in the transmit descriptor list. DMA ignores the LSB bits (1:0, 2:0, or 3:0) for 32-bit, 64-bit, or 128-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).<br>The field width depends on the configuration:<br>31:2 for 32-bit configuration<br>31:3 for 64-bit configuration<br>31:4 for 128-bit configuration |
| 1-0<br>— | Reserved. |

## 72.17.249 DMA Channel 0 Rx Descriptor List Address (DMA_CH0_RxDesc_List_Address)

### Offset

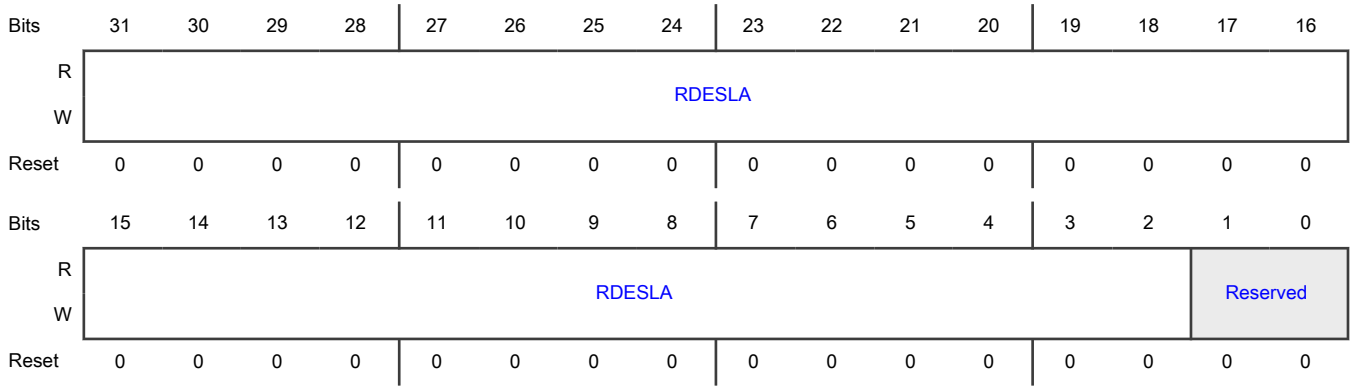
| Register | Offset |
|---|---|
| DMA_CH0_RxDesc_List_Address | 111Ch |

### Function

Specifies DMA to the start of receive descriptor list. The descriptor lists reside in the physical memory space of the application and must be Word, Dword, or Lword-aligned (for 32-bit, 64-bit, or 128-bit data bus). DMA internally converts it to bus width aligned address by making the corresponding LS bits low. You can write to this register only when reception stops. When stopped, write this register before the receive start command is given. You can write to this register only when Rx DMA stops, that is, DMA_CH0_Rx_Control[SR] = 0. When stopped, write this register with a new descriptor list address.

When DMA_CH0_Rx_Control[SR] = 1, it indicates that DMA takes the newly programmed descriptor base address.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>RDESLA | Start of Receive List<br>Contains the base address of the first descriptor in the receive descriptor list. DMA ignores the LSB bits (1:0, 2:0, or 3:0) for 32-bit, 64-bit, or 128-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).<br>The field width depends on the configuration:<br>31:2 for 32-bit configuration<br>31:3 for 64-bit configuration<br>31:4 for 128-bit configuration |
| 1-0<br>— | Reserved. |

## 72.17.250 DMA Channel 0 Tx Descriptor Tail Pointer (DMA_CH0_TxDesc_Tail_Pointer)

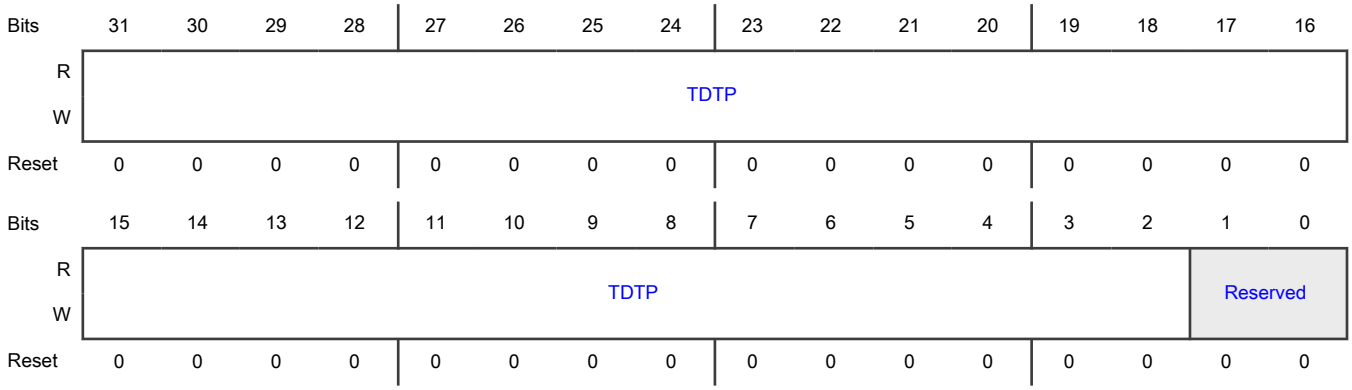
### Offset

| Register | Offset |
|---|---|
| DMA_CH0_TxDesc_Tail_Pointer | 1120h |

### Function

Points to an offset from the base and indicates the location of the last valid descriptor.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>TDTP | Transmit Descriptor Tail Pointer<br>Contains the tail pointer for the transit descriptor ring. You must write the tail pointer to add more descriptors to the transit channel. The hardware tries to transmit all packets to the descriptors referenced between the head and the tail pointer registers.<br>The field width depends on the configuration:<br>31:2 for 32-bit configuration<br>31:3 for 64-bit configuration<br>31:4 for 128-bit configuration |
| 1-0<br>— | Reserved. |

## 72.17.251 DMA Channel 0 Rx Descriptor List Pointer (DMA_CH0_RxDesc_Tail_Pointer)

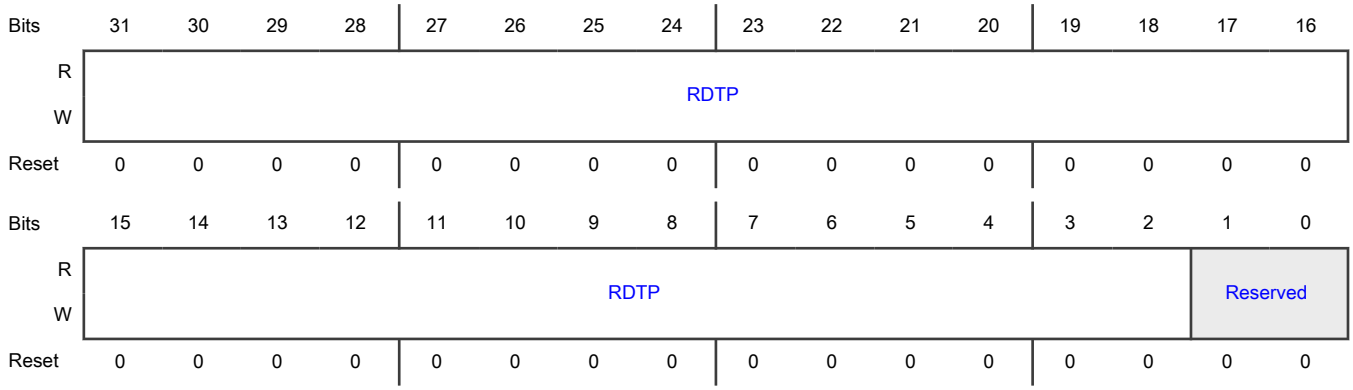
### Offset

| Register | Offset |
|---|---|
| DMA_CH0_RxDesc_Tail_Pointer | 1128h |

### Function

Specifies an offset from the base and indicates the location of the last valid descriptor.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>RDTP | Receive Descriptor Tail Pointer<br>Contains the tail pointer for the receive descriptor ring. You must write the tail pointer to add more descriptors to the receive channel. The hardware tries to write all received packets to the descriptors referenced between the head and the tail pointer registers.<br>The field width depends on the configuration:<br>31:2 for 32-bit configuration<br>31:3 for 64-bit configuration<br>31:4 for 128-bit configuration |
| 1-0<br>— | Reserved. |

## 72.17.252 DMA Channel 0 Tx Descriptor Ring Length (DMA_CH0_TxDesc_Ring_Length)

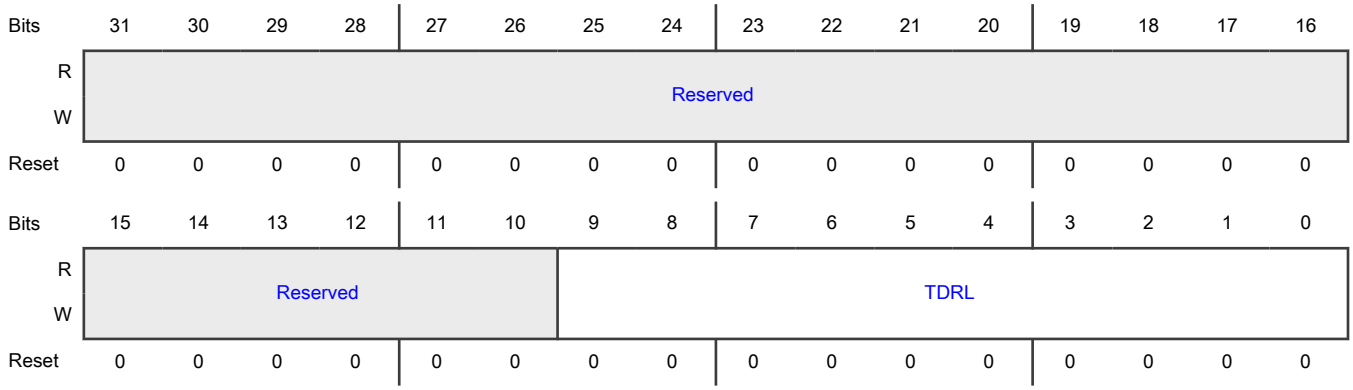
### Offset

| Register | Offset |
|---|---|
| DMA_CH0_TxDesc_Ring_Length | 112Ch |

### Function

Contains the length of the transmit descriptor ring.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved. |
| 9-0<br>TDRL | Transmit Descriptor Ring Length<br>Sets the maximum number of transmit descriptors in the circular descriptor ring. The maximum number of descriptors are limited to 1K descriptors. NXP recommends a minimum ring descriptor length of 4. For example, you can program any value up to 0x3FF in this field. This field is 10 bits wide, if you program 0x3FF, you can have 1024 descriptors. Program this field to a value of 0x9, if you want to have 10 descriptors. |

## 72.17.253 DMA Channel 0 Rx Descriptor Ring Length (DMA_CH0_RxDesc_Ring_Length)

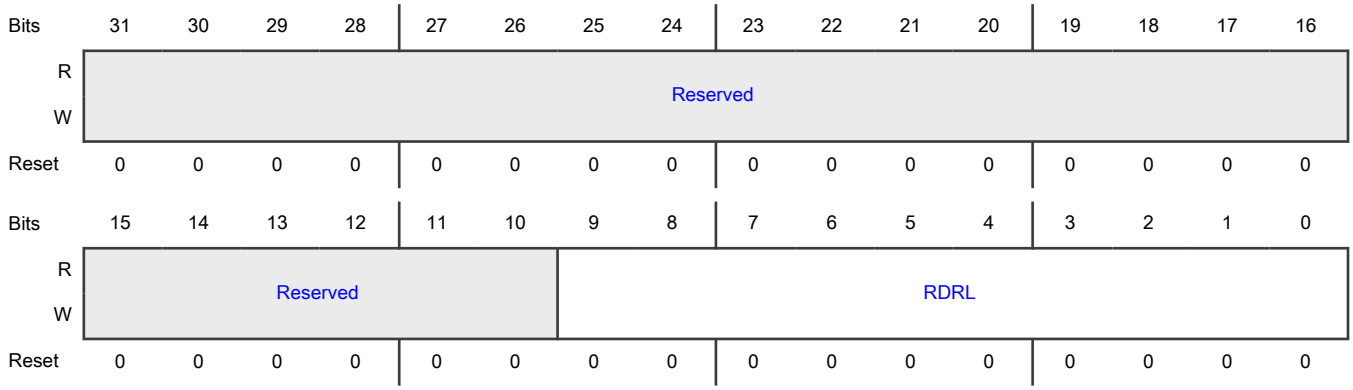
### Offset

| Register | Offset |
|---|---|
| DMA_CH0_RxDesc_Ring_Length | 1130h |

### Function

Contains the length of the receive descriptor circular ring.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved. |
| 9-0<br>RDRL | Receive Descriptor Ring Length<br>Sets the maximum number of receive descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors. For example, you can program any value up to 0x3FF in this field. This field is 10 bits wide, if you program 0x3FF, you can have 1024 descriptors. Program this field to a value of 0x9, if you want to have 10 descriptors. |

## 72.17.254 DMA Channel 0 Interrupt Enable (DMA_CH0_Interrupt_Enable)

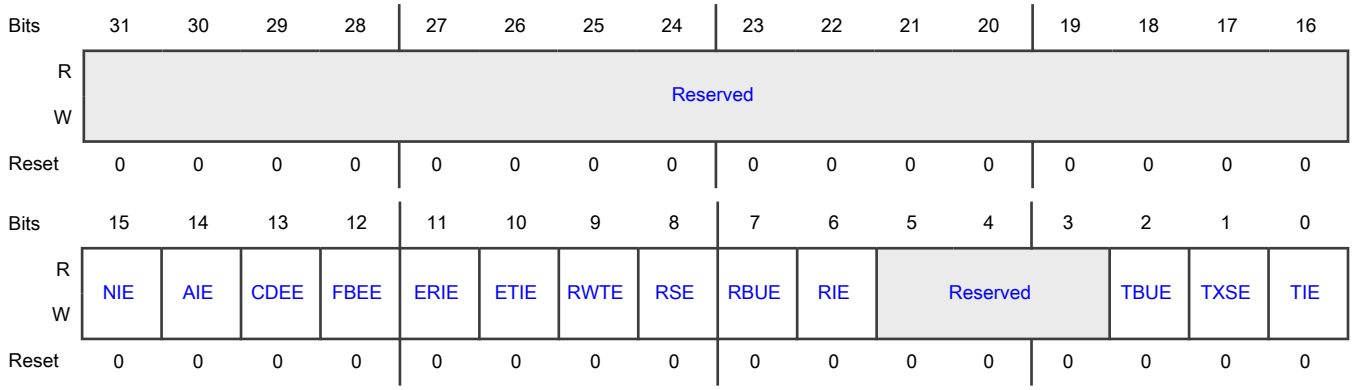
### Offset

| Register | Offset |
|---|---|
| DMA_CH0_Interrupt_Enable | 1134h |

### Function

Enables the interrupts which the status register report.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved. |
| 15<br>NIE | Normal Interrupt Summary Enable<br>Enables or disables the normal interrupt summary.<br>When this field is 1, it enables the normal interrupt summary. This field also enables the following interrupts in DMA_CH0_Status:<br>-Bit 0 - Transmit interrupt<br>Bit 2 - Transmit buffer unavailable<br>Bit 6 - Receive interrupt<br>Bit 11 - Early receive interrupt<br>When this field becomes 0, it disables the normal interrupt summary.<br>0b - Disable<br>1b - Enable |
| 14<br>AIE | Abnormal Interrupt Summary Enable<br>Enables or disables the abnormal interrupt summary.<br>When this field is 1, it enables the abnormal interrupt summary. This field also enables the following interrupts in DMA_CH0_Status:<br>Bit 1 - Transmit process stopped<br>Bit 7 - Rx buffer unavailable<br>Bit 8 - Receive process stopped<br>Bit 9 - Receive watchdog timeout<br>Bit 10 - Early transmit interrupt<br>Bit 12 - Fatal bus error<br>Bit 13 - Context descriptor error |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When this field becomes 0, it disables the abnormal interrupt summary.<br>0b - Disable<br>1b - Enable |
| 13<br>CDEE | Context Descriptor Error Enable<br>Enables or disables the context descriptor error.<br>When this field is 1 along with DMA_CH0_Interrupt_Enable[AIE], it enables the descriptor error interrupt. When this field becomes 0, it disables the descriptor error interrupt.<br>0b - Disable<br>1b - Enable |
| 12<br>FBEE | Fatal Bus Error Enable<br>Enables or disables the fatal bus error.<br>When this field is 1 along with DMA_CH0_Interrupt_Enable[AIE], it enables the fatal bus error interrupt. When this field becomes 0, it disables the fatal bus error interrupt.<br>0b - Disable<br>1b - Enable |
| 11<br>ERIE | Early Receive Interrupt Enable<br>Enables or disables the early receive interrupt.<br>When this field is 1 along with DMA_CH0_Interrupt_Enable[NIE], it enables the early receive interrupt. When this field becomes 0, it disables the early receive interrupt.<br>0b - Disable<br>1b - Enable |
| 10<br>ETIE | Early Transmit Interrupt Enable<br>Enables or disables the early transmit interrupt.<br>When this field is 1 along with DMA_CH0_Interrupt_Enable[AIE], it enables the early transmit interrupt. When this field becomes 0, it disables the early transmit interrupt.<br>0b - Disable<br>1b - Enable |
| 9<br>RWTE | Receive Watchdog Timeout Enable<br>Enables or disables the receive watchdog timeout.<br>When this field is 1 along with DMA_CH0_Interrupt_Enable[AIE], it enables the receive watchdog timeout interrupt. When this field becomes 0, it disables the receive watchdog timeout interrupt.<br>0b - Disable<br>1b - Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 8<br>RSE | Receive Stopped Enable<br>Enables or disables the receive stopped interrupt.<br>When this field is 1 along with DMA_CH0_Interrupt_Enable[AIE], it enables the receive stopped interrupt. When this field becomes 0, it disables the receive stopped interrupt.<br>0b - Disable<br>1b - Enable |
| 7<br>RBUE | Receive Buffer Unavailable Enable<br>Enables or disables the receive buffer unavailable interrupt.<br>When this field is 1 along with DMA_CH0_Interrupt_Enable[AIE], it enables the receive buffer unavailable interrupt. When this field becomes 0, it disables the receive buffer unavailable interrupt.<br>0b - Disable<br>1b - Enable |
| 6<br>RIE | Receive Interrupt Enable<br>Enables or disables the receive interrupt.<br>When this field is 1 along with DMA_CH0_Interrupt_Enable[NIE], it enables the receive interrupt. When this field becomes 0, it disables the receive interrupt.<br>0b - Disable<br>1b - Enable |
| 5-3<br>— | Reserved. |
| 2<br>TBUE | Transmit Buffer Unavailable Enable<br>Enables or disables the transmit buffer unavailable interrupt.<br>When this field is 1 along with DMA_CH0_Interrupt_Enable[NIE], it enables the transmit buffer unavailable interrupt. When this field becomes 0, it disables the transmit buffer unavailable interrupt.<br>0b - Disable<br>1b - Enable |
| 1<br>TXSE | Transmit Stopped Enable<br>Enables or disables the transmit stopped interrupt.<br>When this field is 1 along with DMA_CH0_Interrupt_Enable[AIE], it enables the transmission stopped interrupt. When this field becomes 0, it disables the transmission stopped interrupt.<br>0b - Disable<br>1b - Enable |
| 0 | Transmit Interrupt Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| TIE | Enables or disables the transmit interrupt.<br>When this field is 1 along with DMA_CH0_Interrupt_Enable[NIE], it enables the transmit interrupt. When this field becomes 0, it disables the transmit interrupt.<br>0b - Disable<br>1b - Enable |

## 72.17.255 DMA Channel 0 Rx Interrupt Watchdog Timer (DMA_CH0_Rx_Interrupt_Watchdog_Timer)

### Offset

| Register | Offset |
|---|---|
| DMA_CH0_Rx_Interrupt_Watchdog_Timer | 1138h |

### Function

Indicates the watchdog timeout for receive interrupt (RI) from DMA. When you write this register with a non-zero value, it enables the watchdog timer for the RI bit of the DMA_CHi_Status register.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | RWTU | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | RWT | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-18<br>— | Reserved. |
| 17-16<br>RWTU | Receive Interrupt Watchdog Timer Count Units<br>Indicates the system clock cycles number corresponding to one unit in DMA_CH0_Rx_Interrupt_Watchdog_Timer[RWT]. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 2'b00 - 256<br>2'b01 - 512<br>2'b10 - 1024<br>2'b11 - 2048<br>For example, when DMA_CH0_Rx_Interrupt_Watchdog_Timer[RWT] = 2 and DMA_CH0_Rx_Interrupt_Watchdog_Timer[RWTU] = 1, the watchdog timer sets for 2*512=1024 system clock cycles. |
| 15-8<br>— | Reserved. |
| 7-0<br>RWT | Receive Interrupt Watchdog Timer Count<br>Indicates the number of system clock cycles, multiplied by factor which DMA_CH0_Rx_Interrupt_Watchdog_Timer[RWTU] indicates, for which you set the watchdog timer.<br>The watchdog timer triggers with the programmed value after the receive DMA completes the packet transfer for which the RI bit is not set in the DMA_CH(#i)_Status register, because of the setting of interrupt enable bit in the corresponding descriptor RDES3[30].<br>When the watchdog timer runs out, it indicates that the RI bit is 1 and the timer stops. The watchdog timer resets when the RI bit is 1 because of automatic setting of RI per the interrupt enable bit RDES3[30] of any received packet. |

## 72.17.256 DMA Channel 0 Slot Function Control Status (DMA_CH0_Slot_Function_Control_Status)

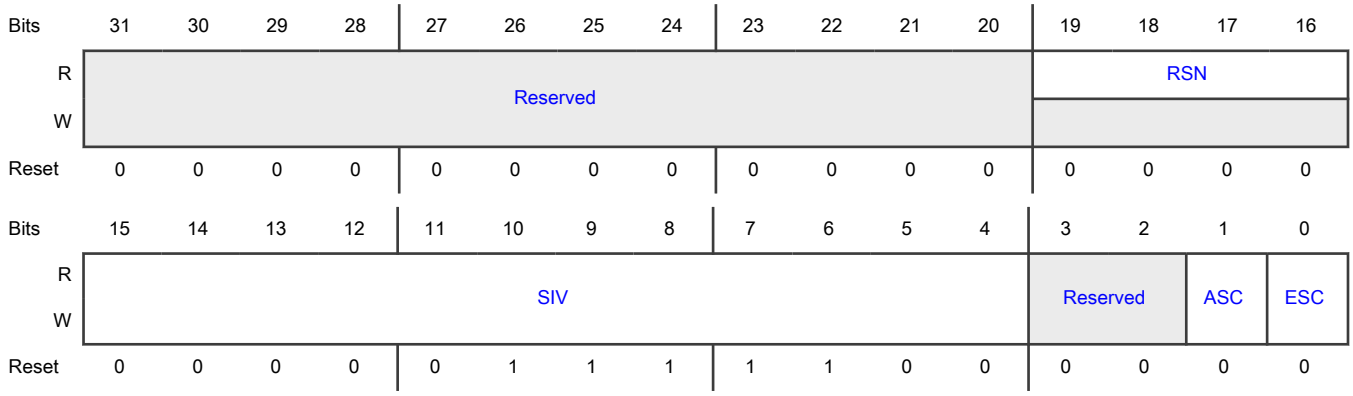
### Offset

| Register | Offset |
|---|---|
| DMA_CH0_Slot_Function_Control_Status | 113Ch |

### Function

Contains the control bits for slot function and the status for transmit path.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved. |
| 19-16<br>RSN | Reference Slot Number<br>Provides the current value of the reference slot number in DMA. It is used for slot comparison. |
| 15-4<br>SIV | Slot Interval Value<br>Controls the period of the slot interval in which the TxDMA fetches the scheduled packets. A value of 0 specifies the slot interval of 1 us while the maximum value 4095 specifies the slot interval of 4096 us. The default or reset value is 0x07C which corresponds to slot interval of 125 us |
| 3-2<br>— | Reserved. |
| 1<br>ASC | Advance Slot Check<br>Indicates whether the advance slot check is enabled.<br>When this field is 1, it enables DMA to fetch the data from the buffer when the slot number (SLOTNUM) programmed in the transit descriptor is either:<br>Equal to the reference slot number given in DMA_CH0_Slot_Function_Control_Status[RSN], or<br>Ahead of the reference slot number by up to two slots.<br>This field is applicable only when DMA_CH0_Slot_Function_Control_Status[ESC] = 1.<br>0b - Disabled<br>1b - Enabled |
| 0<br>ESC | Enable Slot Comparison<br>Indicates whether the slot comparison is enabled. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When this field is 1, it enables the checking of the slot numbers programmed in the transit descriptor with the current reference given in DMA_CH0_Slot_Function_Control_Status[RSN]. DMA fetches the data from the corresponding buffer only when the slot number is either:<br>• Equal to the reference slot number, or<br>• Ahead of the reference slot number by one slot.<br>When this field becomes 0, it disables the checking of the slot numbers. DMA fetches the data immediately after you process the descriptor.<br>**NOTE**<br>You must not enable the UFO (UDP Fragmentation over IPv4)/TSO/USO along with TBS/AVB slot number check. The UFO/TSO/USO involves multiple packets/segments/ fragments transmission for single packet received from application and the slot number check are applicable for fetching only first segment/fragment. As a result it might be difficult for you to specify slot number for subsequent packets.<br>0b - Disabled<br>1b - Enabled |

## 72.17.257 DMA Channel 0 Current Application Transmit Descriptor (DMA_CH0_Current_App_TxDesc)

### Offset

| Register | Offset |
|---|---|
| DMA_CH0_Current_App_TxDesc | 1144h |

### Function
Specifies the current transmit descriptor which DMA reads.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>CURTDESAPTR | Application Transmit Descriptor Address Pointer<br>Indicates that DMA updates this pointer during the transit operation. This pointer clears when reset. |

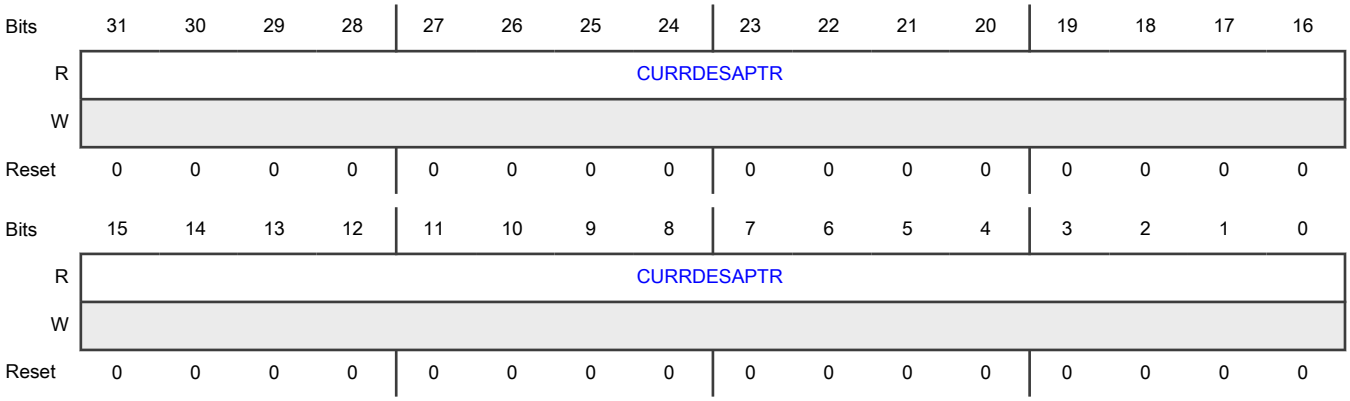
## 72.17.258 DMA Channel 0 Current Application Receive Descriptor (DMA_CH0_Current_App_RxDesc)

### Offset

| Register | Offset |
|---|---|
| DMA_CH0_Current_App_RxDesc | 114Ch |

### Function
Specifies the current receive descriptor which DMA read.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>CURRDESAPTR | Application Receive Descriptor Address Pointer<br>Indicates that the DMA updates this pointer during the receive operation. This pointer clears when reset. |

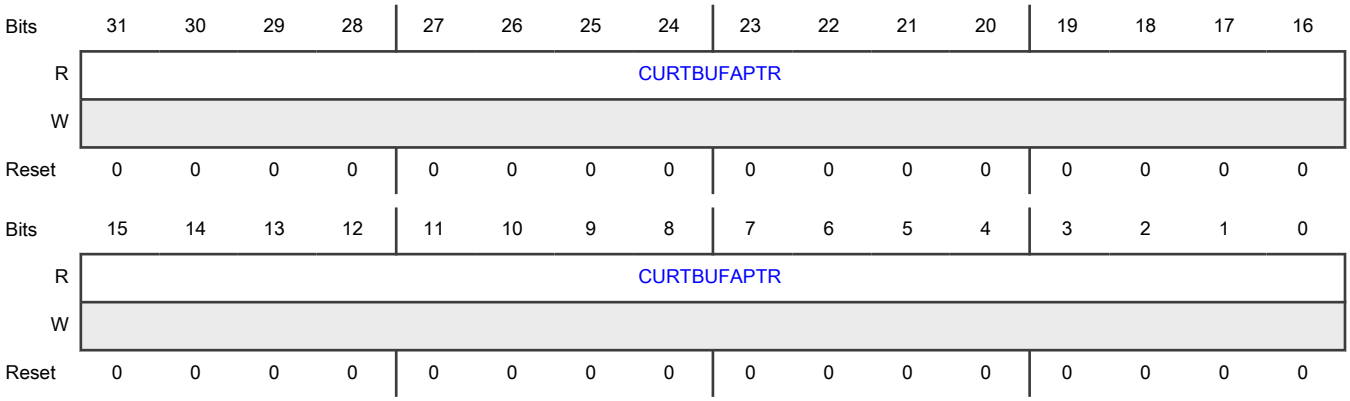
## 72.17.259 DMA Channel 0 Current Application Transmit Descriptor (DMA_CH0_Current_App_TxBuffer)

### Offset

| Register | Offset |
|---|---|
| DMA_CH0_Current_App_TxBuffer | 1154h |

### Function
Specifies the current transit buffer address which DMA reads.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>CURTBUFAPTR | Application Transmit Buffer Address Pointer<br>Indicates that DMA updates this pointer during the transit operation. This pointer clears when reset. |

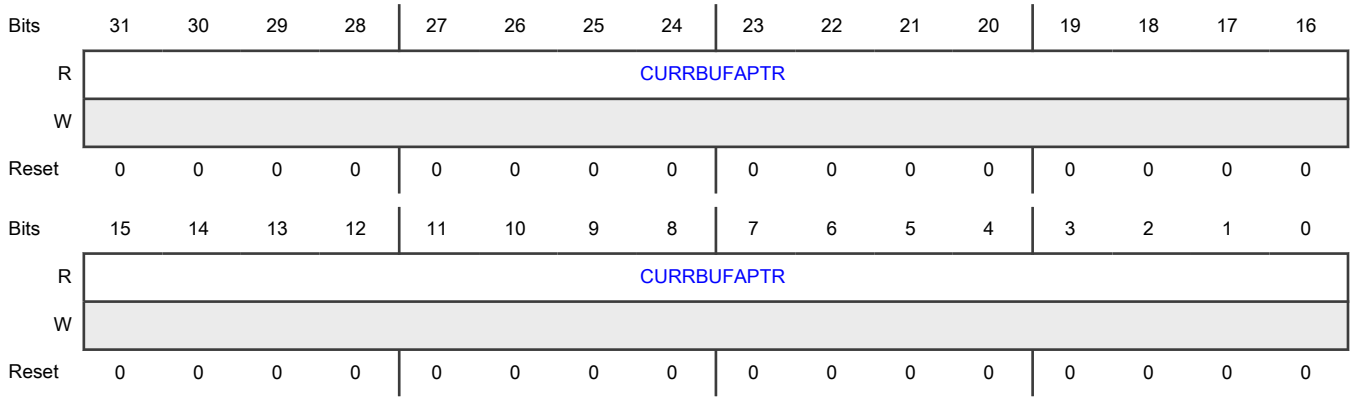
## 72.17.260 DMA Channel 0 Current Application Receive Buffer (DMA_CH0_Current_App_RxBuffer)

### Offset

| Register | Offset |
|---|---|
| DMA_CH0_Current_App_RxBuffer | 115Ch |

### Function
Specifies the current receive buffer address which DMA read.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>CURRBUFAPTR | Application Receive Buffer Address Pointer<br>Indicates that DMA updates this pointer during the receive operation. This pointer clears when reset. |

## 72.17.261 DMA Channel 0 Status (DMA_CH0_Status)

### Offset

| Register | Offset |
|---|---|
| DMA_CH0_Status | 1160h |

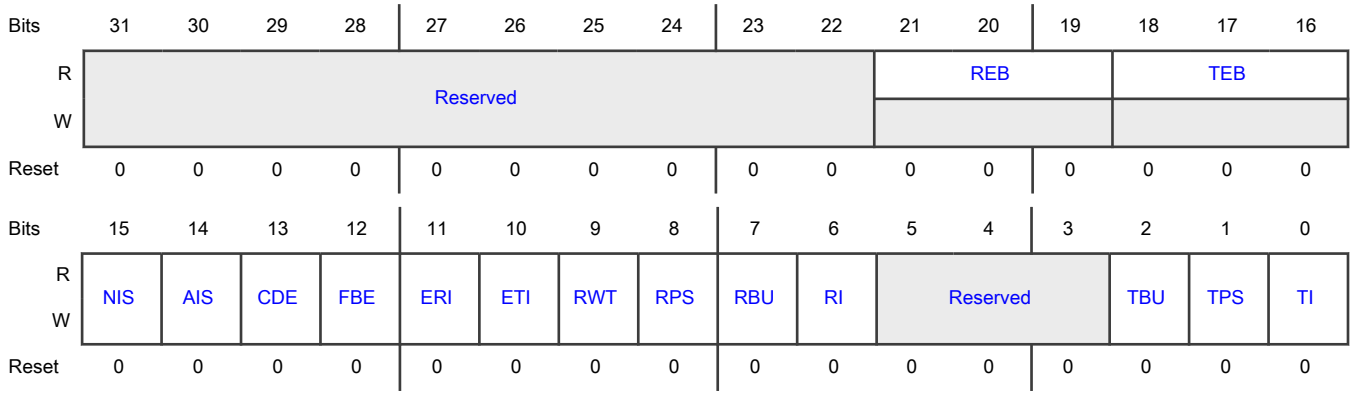
### Function

Indicates that the software driver (application) reads the status register during an interrupt service routine or polling to determine the DMA status.

**NOTE**

The number of DMA_CH(#i)_Status register in the configuration is the higher of number of receive DMA channels and transit DMA channels.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-22<br>— | Reserved. |
| 21-19<br>REB | Rx DMA Error Bits<br>Indicates the type of error that causes a bus error. For example, error response on the AHB or AXI interface.<br>Bit 21<br>1'b1 - Error during data transfer by Rx DMA<br>1'b0 - No Error during data transfer by Rx DMA<br>Bit 20<br>1'b1 - Error during descriptor access<br>1'b0 - Error during data buffer access<br>Bit 19<br>1'b1 - Error during read transfer<br>1'b0 - Error during write transfer<br>This field is valid only when DMA_CH0_Status[FBE] = 1. This field does not generate an interrupt. |
| 18-16<br>TEB | Tx DMA Error Bits<br>Indicates the type of error that causes a bus error. For example, error response on the AHB or AXI interface.<br>Bit 18<br>1'b1 - Error during data transfer by Tx DMA<br>1'b0 - No Error during data transfer by Tx DMA<br>Bit 17<br>1'b1 - Error during descriptor access<br>1'b0 - Error during data buffer access |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Bit 16<br>1'b1 - Error during read transfer<br>1'b0 - Error during write transfer<br>This field is valid only when DMA_CH0_Status[FBE] = 1. This field does not generate an interrupt. |
| 15<br>NIS | Normal Interrupt Summary<br>Indicates whether the normal interrupt summary status is detected.<br>This field is the logical OR of the following bits when you enables the corresponding interrupt bits in DMA_CH0_Interrupt_Enable:<br>Bit 0 - Transmit interrupt<br>Bit 2 - Transmit buffer unavailable<br>Bit 6 - Receive interrupt<br>Bit 11 - Early receive interrupt<br>Only unmasked bits (interrupts for which interrupt enable sets in DMA_CH0_Interrupt_Enable) affect this field.<br>This is a sticky bit. You must clear this field by writing 1 to it each time a corresponding field which sets this field clears.<br>Access restriction apply to this field. It automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 14<br>AIS | Abnormal Interrupt Summary<br>Indicates whether an abnormal interrupt summary status is detected.<br>This field is the logical OR of the following bits when you enables the corresponding interrupt bits in DMA_CH0_Interrupt_Enable:<br>Bit 1 - Transmit process stopped<br>Bit 7 - Receive buffer unavailable<br>Bit 8 - Receive process stopped<br>Bit 10 - Early transmit interrupt<br>Bit 12 - Fatal bus error<br>Bit 13 - Context descriptor error<br>Only unmasked bits affect this field.<br>This is a sticky bit. You must clear this field by writing 1 to it each time a corresponding field, which sets this field, clears.<br>Access restriction apply to this field. It automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Not detected<br>1b - Detected |
| 13<br>CDE | Context Descriptor Error<br>Indicates whether the context descriptor error status is detected.<br>This field indicates that the DMA Tx/Rx engine receives a descriptor error, which indicates an invalid context in the middle of packet flow ( intermediate descriptor) or all one's descriptor in transit case and on receive side it indicates DMA has read a descriptor with either of the buffer address as ones which is considered to be invalid.<br>Access restriction apply to this field. It automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 12<br>FBE | Fatal Bus Error<br>Indicates whether the fatal bus error status is detected.<br>This field indicates that a bus error occurred (as described in the EB field). When this field is 1, it indicates that the corresponding DMA channel engine disables all bus accesses.<br>Access restriction apply to this field. It automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 11<br>ERI | Early Receive Interrupt<br>Indicates whether an early receive interrupt status is detected.<br>When this field is 1, it indicates that the RxDMA completes the packet data transfer to the memory.<br>In configs supporting ERIC, this field is 1 only after the receive DMA fills a complete receive buffer with packet data, when DMA_CH0_Rx_Control[ERIC] = 0. This field is 1 after every burst transfer of data from the receive DMA to the buffer, when DMA_CH0_Rx_Control[ERIC] = 1.<br>Writing 1 to RI field automatically clears this field.<br>Access restriction apply to this field. It automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 10<br>ETI | Early Transmit Interrupt<br>Indicates whether an early transmit interrupt status is detected.<br>When this field is 1, it indicates that the TxDMA completes the packet data transfer to the MTL TXFIFO memory. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | In configs supporting ERIC: this field is 1 only after the transit DMA transfers a complete packet to MTL, when DMA_CH0_Tx_Control[ETIC] = 0. This field is 1 after packet data transfers from buffer completes (partial) in the transmit descriptor in which IOC = 1, when DMA_CH0_Tx_Control[ETIC] = 1.<br>Access restriction apply to this field. It automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 9<br>RWT | Receive Watchdog Timeout<br>Indicates whether the receive watchdog timeout status is detected.<br>Asserts when you receives a packet with length greater than 2,048 bytes (10,240 bytes when Jumbo Packet mode is enabled).<br>0b - Not detected<br>1b - Detected |
| 8<br>RPS | Receive Process Stopped<br>Indicates whether the receive process stopped status is detected.<br>Asserts when the receive process enters the stopped state.<br>Access restriction apply to this field. It automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 7<br>RBU | Receive Buffer Unavailable<br>Indicates whether the receive buffer unavailable status is detected.<br>This field indicates that the application owns the next descriptor in the receive list, and DMA cannot acquire it. The receive process is suspended. To resume processing receive descriptors, the application must change the ownership of the descriptor and issue a receive poll demand command. If this command is not issued, the receive process resumes when you receive the next recognized incoming packet. In ring mode, the application must advance the receive descriptor tail pointer register of a channel. This field is 1 only when DMA owns the previous receive descriptor.<br>Access restriction apply to this field. It automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 6<br>RI | Receive Interrupt<br>Indicates that the receive interrupt status is detected. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | This field indicates that the packet reception is complete. When packet reception completes, bit 31 of RDES3 resets in the last descriptor, and you can update the specific packet status information in the descriptor.<br>The reception remains in the running state.<br>Access restriction apply to this field. It automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 5-3<br>— | Reserved. |
| 2<br>TBU | Transmit Buffer Unavailable<br>Indicates whether the transmit buffer unavailable status is detected.<br>This field indicates that the application owns the next descriptor in the transmit list, and the DMA cannot acquire it. Transmission is suspended. DMA_Debug_Status0[TPS0] explains the transmit process state transitions.<br>To resume processing the transmit descriptors, the application must perform these steps:<br>1. Change the ownership of the descriptor by writing 1 to bit 31 of TDES3.<br>2. Issue a transmit poll demand command.<br>For Ring mode, the application must advance the transmit descriptor tail pointer register of a channel.<br>Access restriction apply to this field. It automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 1<br>TPS | Transmit Process Stopped<br>Indicates whether the transmit process stopped status is detected.<br>This field is 1 when the transmission stops.<br>Access restriction apply to this field. It automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 0<br>TI | Transmit Interrupt<br>Indicates whether the transmit interrupt status is detected.<br>This field indicates that the packet transmission is complete. When transmission completes, Bit 31 of TDES3 resets in the last descriptor, and you can update the specific packet status information in the descriptor. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Access restriction apply to this field. It automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |

## 72.17.262 DMA Channel 0 Miss Frame Counter (DMA_CH0_Miss_Frame_Cnt)

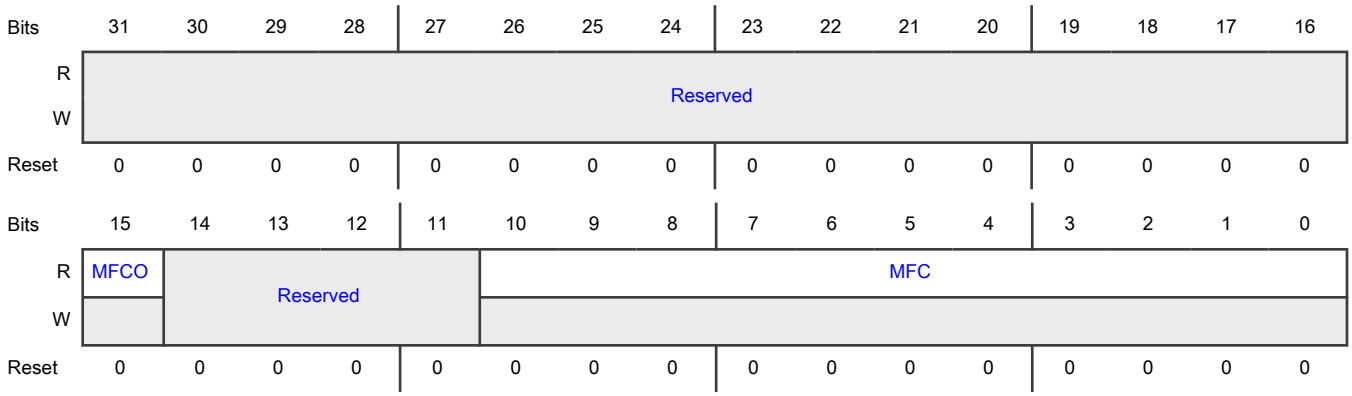
### Offset

| Register | Offset |
|---|---|
| DMA_CH0_Miss_Frame_Cnt | 1164h |

### Function

Indicates the number of packet counter that DMA drops either due to bus error or due to programming RPF field in DMA_CH${i}_Rx_Control register.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved. |
| 15<br>MFCO | Overflow status of the MFC Counter<br>Indicates whether the miss frame counter overflow has occurred.<br>When this field is 1, it indicates that the MFC counter does not increment further. This field is 0 when this register is read. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Access restriction apply to this field. It clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not occurred<br>1b - Occurred |
| 14-11<br>— | Reserved. |
| 10-0<br>MFC | Dropped Packet Counters<br>Indicates the number of packet counters that DMA drops either because of bus error or because of programing RPF field in DMA_CH${i}_Rx_Control register. This field is 0 when this register is read.<br>Access restriction apply to this field. It clears on read and automatically becomes 1 on an internal event occurrence. |

## 72.17.263 DMA Channel 0 Rx Parser Accept Count (DMA_CH0_RXP_Accept_Cnt)

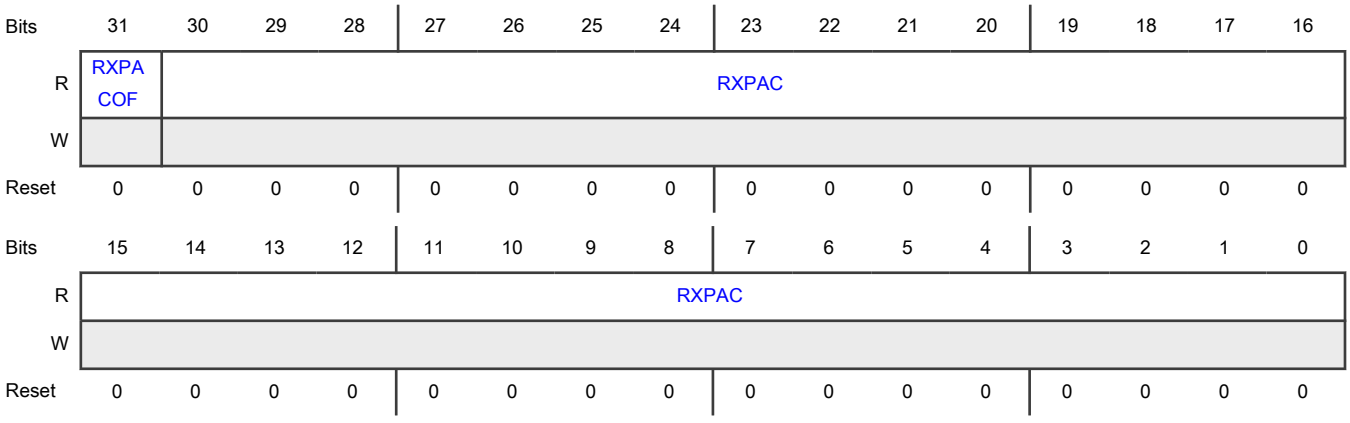
### Offset

| Register | Offset |
|---|---|
| DMA_CH0_RXP_Accept_Cnt | 1168h |

### Function

Provides the count of the number of frames which the receive parser accept.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31<br>RXPACOF | Rx Parser Accept Counter Overflow Bit<br>Indicates whether the receive parser accept counter overflow has occurred.<br>When this field is 1, it indicates that the RXPAC counter field has crossed the maximum limit.<br>Access restriction apply to this field. It clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not occurred<br>1b - Occurred |
| 30-0<br>RXPAC | Rx Parser Accept Counter<br>Implements whenever a receive parser accept a packet because AF = 1. The counter clears when the register is read. |

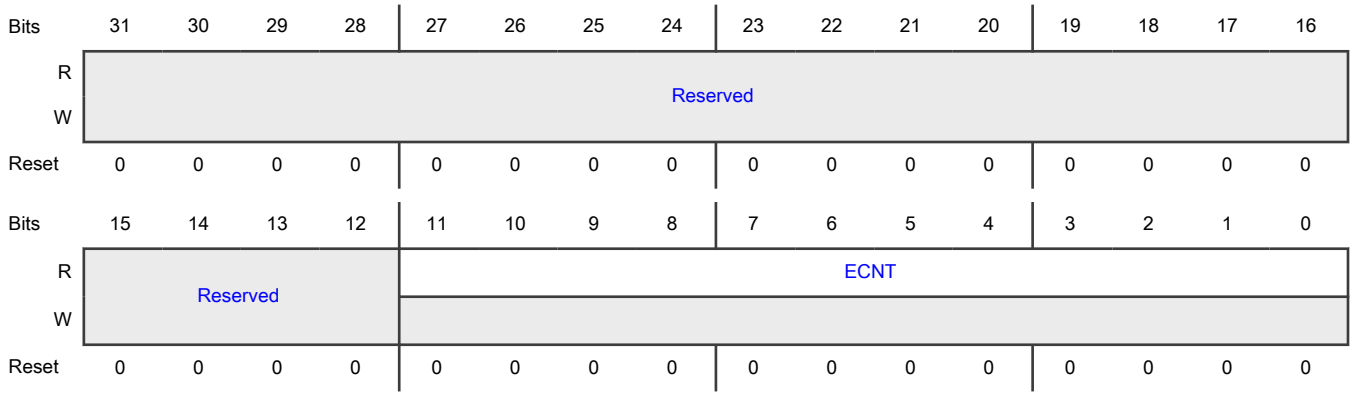
## 72.17.264 DMA Channel 0 Rx ERI Count (DMA_CH0_RX_ERI_Cnt)

### Offset

| Register | Offset |
|---|---|
| DMA_CH0_RX_ERI_Cnt | 116Ch |

### Function
Provides the count of the number of times ERI is asserted.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-12 | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 11-0<br>ECNT | ERI Counter<br>Indicates that when ERIC bit of DMA_CH(#i)_RX_Control register is 1, this counter increments for burst transfer which the receive DMA completes from the start of packet transfer. This field becomes 0 at the start of new packet. |

## 72.17.265 DMA Channel 1 Control (DMA_CH1_Control)

### Offset

| Register | Offset |
|---|---|
| DMA_CH1_Control | 1180h |

### Function

Specifies the MSS value for segmentation, length to skip between two descriptors, and also the features such as header splitting and 8xPBL mode.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | Reserved | Reserved | | | | DSL | | | Reserved | PBLx8 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | Reserved | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-25<br>— | Reserved. |
| 24<br>— | Reserved. |
| 23-21 | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 20-18<br>DSL | Descriptor Skip Length<br>Specifies the Word, Dword, or Lword number (depending on the 32-bit, 64-bit, or 128-bit bus) to skip between two unchained descriptors. The address skipping starts from the end of the current descriptor to the start of the next descriptor.<br>When this field is 0, it indicates that DMA takes the descriptor table as contiguous. |
| 17<br>— | Reserved. |
| 16<br>PBLx8 | 8xPBL mode<br>Indicates whether the 8xPBL mode is enabled.<br>When this field is 1, it indicates that you must program the PBL value in Bits[21:16] in DMA_CH(#i)_Tx_Control and Bits[21:16] in DMA_CH(#i)_Rx_Control and multiply it eight times. Therefore, DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value.<br>0b - Disabled<br>1b - Enabled |
| 15-14<br>— | Reserved. |
| 13-0<br>— | Reserved. |

## 72.17.266 DMA Channel 1 Tx Control (DMA_CH1_Tx_Control)

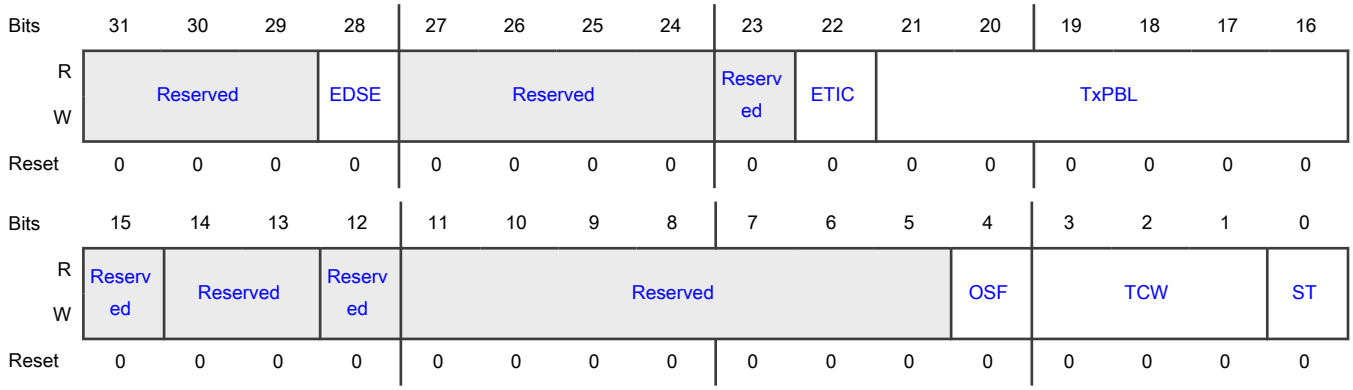
### Offset

| Register | Offset |
|---|---|
| DMA_CH1_Tx_Control | 1184h |

### Function

Controls the transmit features such as PBL, TCP segmentation, and transmit channel weights.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved. |
| 28<br>EDSE | Enhanced Descriptor Enable<br>Indicates whether an enhanced descriptor is enabled.<br>When this field is 1, it indicates that the corresponding channel uses enhanced descriptors that are 32 Bytes for both normal and context descriptors.<br>When this field becomes 0, it indicates that the corresponding channel uses the descriptors that are 16 Bytes.<br>0b - Disabled<br>1b - Enabled |
| 27-24<br>— | Reserved. |
| 23<br>— | Reserved. |
| 22<br>ETIC | Early Transmit Interrupt Control<br>Indicates whether an early transmit interrupt is enabled.<br>When this field is 1, it indicates that an early transmit interrupt (ETI) status sets after the data transfer completes from buffers of a transmit descriptor in which IOC bit (TDES2[31]) sets.<br>When this field becomes 0, it indicates that the ETI sets only after a complete packet transfers to the MTL TX FIFO memory.<br>0b - Disabled<br>1b - Enabled |
| 21-16 | Transmit Programmable Burst Length |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| TxPBL | Indicates the maximum number of beats to transfer in one DMA block data. DMA always attempts max burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of these values: 1, 2, 4, 8, 16, or 32. Any other value results in an undefined behavior.<br>To transfer more than 32 beats, perform the following steps:<br>1. Write 1 to DMA_CH0_Control[PBLx8].<br>2. Write 1 to this field.<br>**NOTE**<br>The maximum value of this field must be less than or equal to half the transit queue size (TQS field of MTL_TxQ(#i)_Operation_Mode register) in terms of beats. This is required so that the transit queue has space to store at least another Tx PBL worth of data while the MTL Tx Queue controller transfers data to MAC. For example, in 64-bit data width configurations the total locations in transit queue of size 512 bytes is 64. You must program TxPBL and 8xPBL to less than or equal to 32. |
| 15<br>— | Reserved. |
| 14-13<br>— | Reserved. |
| 12<br>— | Reserved. |
| 11-5<br>— | Reserved. |
| 4<br>OSF | Operate on Second Packet<br>Indicates whether an operation on second packet is enabled.<br>When this field is 1, it instructs DMA to process the second packet of the transmit data even before the first packet status is obtained.<br>0b - Disabled<br>1b - Enabled |
| 3-1<br>TCW | Transmit Channel Weight<br>Indicates the weight assigned to the corresponding transmit channel. When reset completes, this field becomes 0 for all channels by default, resulting in equal weights to all channels. |
| 0<br>ST | Start or Stop Transmission Command<br>Indicates whether to start or stop transmission command.<br>When this field is 1, it indicates that the transmission is placed in the running state. DMA checks the transmit list at the current position to transmit a packet. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | DMA tries to acquire descriptor from either of the following positions:<br>• The current position in the list. This is the base address of the transmit list which DMA_CH0_TxDesc_List_Address sets.<br>• The position at which the transmission was previously stopped<br>If DMA does not own the current descriptor, the transmission enters the suspended state and DMA_CH0_Status[TBU] = 1. The start transmission command is effective only when the transmission stops. If you issue the command before writing 1 to DMA_CH0_TxDesc_List_Address, then DMA behavior is unpredictable.<br>When this field becomes 0, it indicates that the transmission process is placed in the stopped state after the transmission of the current packet completes. The next descriptor position in the transmit list is saved, and it becomes the current position when the transmission restarts. To change the list address, you must program DMA_CH0_TxDesc_List_Address with a new value when this field becomes 0. The new value is considered when this field is 1 again. The stop transmission command is effective only when the transmission of the current packet completes or the transmission is in the suspended state.<br>0b - Stop<br>1b - Start |

## 72.17.267 DMA Channel 1 Rx Control (DMA_CH1_Rx_Control)

### Offset

| Register | Offset |
|---|---|
| DMA_CH1_Rx_Control | 1188h |

### Function

Controls the receive features such as PBL, buffer size, and extended status.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | RPF | Reserved | | | Reserved | | | | Reserved | ERIC | RxPBL | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | RBSZ_13_y | | | | | | | | | | | | RBSZ_x_0 | | SR |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>RPF | Rx Packet Flush<br>Indicates whether the receive packet flush is enabled.<br>When this field is 1, it indicates that the module automatically flushes the packet from the receive queues destined to this DMA receive channel, when it stops. When this field remains 1 and the software driver re-starts DMA, the packets residing in the receive queues that were received when this RxDMA was stopped, flushes out. Route the packets to the RxDMA that the MAC receives after the RxDMA re-starts. The flushing takes place on the read side of the receive queue.<br>When this field is 0, it indicates that the module do not flush the packet in the receive queue destined to this RxDMA channel when it is in stop state. This might in turn cause head-of-line blocking in the corresponding RxQueue.<br>**NOTE**<br>The stopping of packet flow from a Rx DMA Channel to the application by writing 1 to this field works only when there is one-to-one mapping of receive queue to the receive DMA channels. In Dynamic mapping mode, writing 1 to this field in any DMA_CH(#i)_Rx_Control register might flush packets from unintended receive queues which are destined to the stopped Rx DMA Channel.<br>0b - Disabled<br>1b - Enabled |
| 30-28<br>— | Reserved. |
| 27-24<br>— | Reserved. |
| 23<br>— | Reserved. |
| 22<br>ERIC | Early Receive Interrupt Control<br>Indicates whether an early receive interrupt control status is enabled.<br>When this field is 1, it indicates that an early receive interrupt (ERI) status sets after every burst transfer of data from the receive DMA to the buffer completes.<br>When this field becomes 0, it indicates that ERI sets only after the RxDMA fills the complete buffer is filled.<br>0b - Disabled<br>1b - Enabled |
| 21-16<br>RxPBL | Receive Programmable Burst Length<br>Indicates the maximum number of beats to transfer in one DMA block data. DMA always attempts max burst as specified in PBL, each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in an undefined behavior.<br>To transfer more than 32 beats, perform the following steps: |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1. Write 1 to DMA_CH0_Control[PBLx8].<br>2. Write 1 to RxPBL.<br>**NOTE**<br>The maximum value of RxPBL must be less than or equal to half the receive queue size (RQS field of MTL_RxQ(#i)_Operation_Mode register) in terms of beats. This is required so that the receive queue has space to store at least another Rx PBL worth of data when the receive DMA transfers a block of data. For example, in 64-bit data width configurations the total locations in receive queue of size 512 bytes is 64, so you must program RxPBL and 8xPBL to less than or equal to 32. |
| 15<br>— | Reserved. |
| 14-3<br>RBSZ_13_y | Receive Buffer size High<br>Indicates that RBSZ[13:0] splits into two fields higher RBSZ_13_y and lower RBSZ_x_0. RBSZ[13:0] field indicates the size of the receive buffers specified in bytes. The maximum buffer size is limited to 16K bytes. The buffer size is applicable to payload buffers when you enable split headers.<br>**NOTE**<br>The buffer size must be a multiple of 4, 8, or 16 depending on the data bus widths (32-bit, 64-bit, or 128-bit respectively). This is required even if the value of buffer address pointer is not aligned to data bus width. Hence the lower RBSZ_x_0 bits are read-only and the value is considered as all-zero. Thus RBSZ_13_y indicates the buffer size in terms of locations (with the width same as bus-width). |
| 2-1<br>RBSZ_x_0 | Receive Buffer size Low<br>Indicates that RBSZ[13:0] splits into two fields RBSZ_13_y and RBSZ_x_0. RBSZ_x_0 is the lower field whose width is based on data bus width of the configuration.<br>This field width is of 2, 3, or 4 bits for 32-bit, 64-bit, or 128-bit data bus width respectively. This field is read-only (RO). |
| 0<br>SR | Start or Stop Receive<br>Indicates whether to start or stop receive.<br>When this field is 1, it indicates that DMA tries to acquire the descriptor from the receive list and processes the incoming packets.<br>DMA tries to acquire descriptor from either of the following positions:<br>• The current position in the list. This is the address that DMA_CH0_RxDesc_List_Address sets.<br>• The position at which the receive process was previously stopped<br>If DMA does not own the current descriptor, the reception suspends and DMA_CH0_Status[RBU] = 1. The start receive command is effective only when the reception stops. If the command is issued before writing 1 to DMA_CH0_RxDesc_List_Address, DMA behavior is unpredictable. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When this field becomes 0, it indicates that the receive DMA operation stops after the transfer of the current packet. The next descriptor position in the receive list is saved, and it becomes the current position after the receive process restarts. The stop receive command is effective only when the receive process is in the running (waiting for Rx packet) or suspended state.<br>0b - Stop<br>1b - Start |

## 72.17.268 DMA Channel 1 Tx Descriptor List Address (DMA_CH1_TxDesc_List_Address)

### Offset

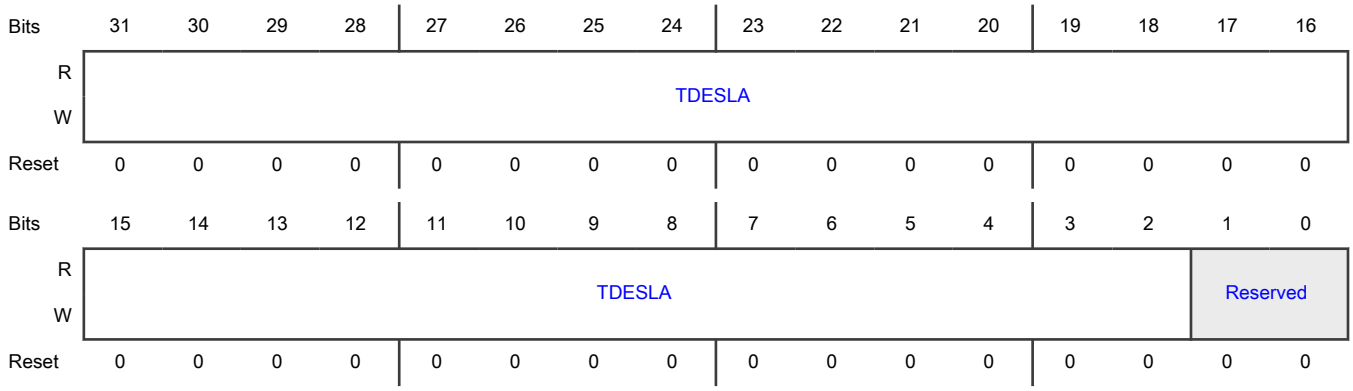
| Register | Offset |
|---|---|
| DMA_CH1_TxDesc_List_Address | 1194h |

### Function

Specifies DMA to the start of transmit descriptor list. The descriptor lists reside in the application's physical memory space and must be Word, Dword, or Lword-aligned (for 32-bit, 64-bit, or 128-bit data bus). The DMA converts the corresponding LSB bits to low to internally convert the descriptor lists to bus width aligned address.

You can write to this register only when the transmit DMA stops, that is, DMA_CH0_Tx_Control[ST] = 0. When stopped, you can write this register with a new descriptor list address. When DMA_CH0_Tx_Control[ST] = 1, the DMA takes the newly-programmed descriptor base address. If this register do not change when DMA_CH0_Tx_Control[ST] = 0, DMA takes the descriptor address where it was stopped earlier.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2 | Start of Transmit List |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| TDESLA | Contains the first descriptor base address in the transmit descriptor list. DMA ignores the LSB bits (1:0, 2:0, or 3:0) for 32-bit, 64-bit, or 128-bit bus width and internally assumes these bits as all-zero. Therefore, these LSB bits are read-only (RO).<br>The field width depends on the configuration:<br>31:2 for 32-bit configuration<br>31:3 for 64-bit configuration<br>31:4 for 128-bit configuration |
| 1-0<br>— | Reserved. |

## 72.17.269 DMA Channel 1 Rx Descriptor List Address (DMA_CH1_RxDesc_List_Address)

### Offset

| Register | Offset |
|---|---|
| DMA_CH1_RxDesc_List_Address | 119Ch |

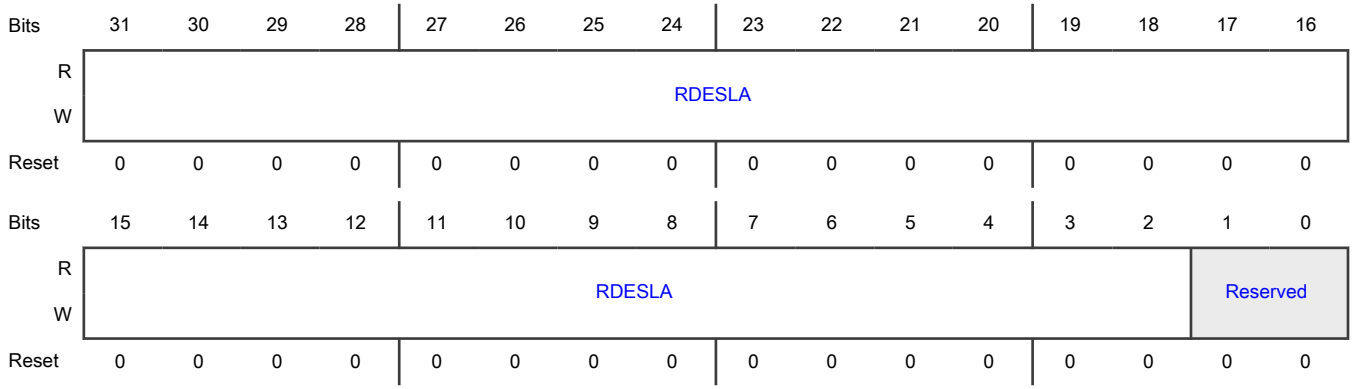
### Function

Specifies DMA to the start of receive descriptor list. The descriptor lists resides in the application's physical memory space and must be Word, Dword, or Lword-aligned (for 32-bit, 64-bit, or 128-bit data bus). DMA converts the corresponding LS bits low to internally convert the descriptor lists to bus width aligned address. You can write to this register only when reception stops. When stopped, you must write this register before the receive start command is given.

You can write to this register when receive DMA has stopped, that is, DMA_CH0_Rx_Control[SR] = 0. When stopped, you can this register with a new descriptor list address.

When DMA_CH0_Rx_Control[SR] = 1, DMA takes the newly programmed descriptor base address.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-2<br>RDESLA | Start of Receive List<br>Contains the first descriptor base address in the receive descriptor list. DMA ignores the LSB bits (1:0, 2:0, or 3:0) for 32-bit, 64-bit, or 128-bit bus width and internally assume these bits as all-zero. Therefore, these LSB bits are read-only (RO).<br>The field width depends on the configuration:<br>31:2 for 32-bit configuration<br>31:3 for 64-bit configuration<br>31:4 for 128-bit configuration |
| 1-0<br>— | Reserved. |

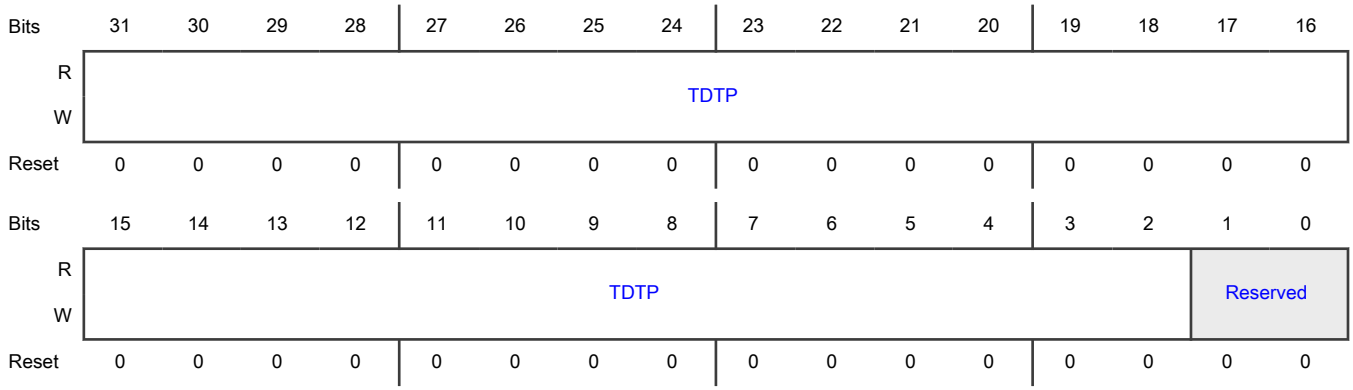
## 72.17.270 DMA Channel 1 Tx Descriptor Tail Pointer (DMA_CH1_TxDesc_Tail_Pointer)

### Offset

| Register | Offset |
|---|---|
| DMA_CH1_TxDesc_Tail_Pointer | 11A0h |

### Function
Specifies to an offset from the base and indicates the location of the last valid descriptor.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2 | Transmit Descriptor Tail Pointer |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| TDTP | Contains the tail pointer for the transit descriptor ring. The software writes the tail pointer to add more descriptors to the transit channel. The hardware tries to transmit all the packets which the descriptors referenced between the head and the tail pointer registers.<br>The field width depends on the configuration:<br>31:2 for 32-bit configuration<br>31:3 for 64-bit configuration<br>31:4 for 128-bit configuration |
| 1-0<br>— | Reserved. |

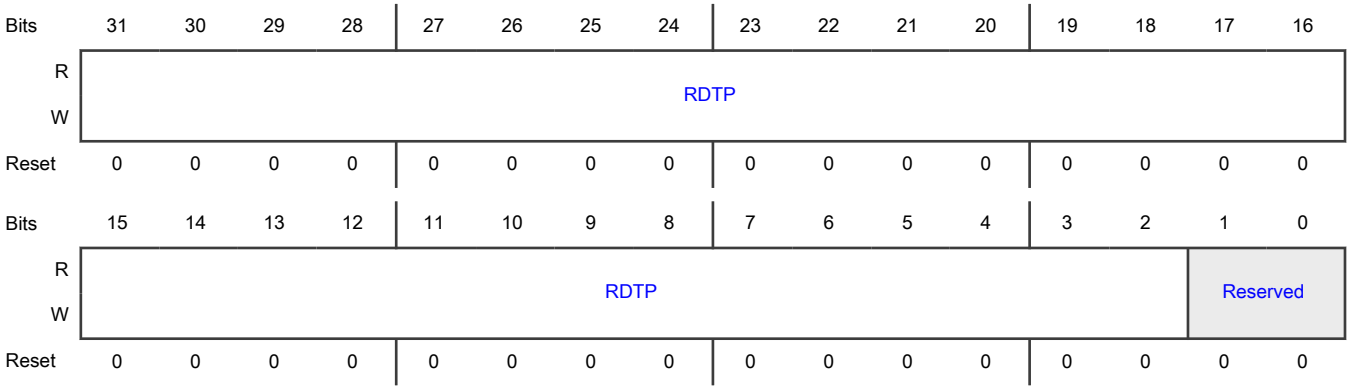
## 72.17.271 DMA Channel 1 Rx Descriptor Tail Pointer (DMA_CH1_RxDesc_Tail_Pointer)

### Offset

| Register | Offset |
|---|---|
| DMA_CH1_RxDesc_Tail_Pointer | 11A8h |

### Function
Specifies to an offset from the base and indicates the location of the last valid descriptor.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>RDTP | Receive Descriptor Tail Pointer |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Contains the tail pointer for the receive descriptor ring. The software writes the tail pointer to add more descriptors to the receive channel. The hardware tries to write all received packets to the descriptors referenced between the head and the tail pointer registers.<br>The field width depends on the configuration:<br>31:2 for 32-bit configuration<br>31:3 for 64-bit configuration<br>31:4 for 128-bit configuration |
| 1-0<br>— | Reserved. |

## 72.17.272 DMA Channel 1 Tx Descriptor Ring Length (DMA_CH1_TxDesc_Ring_Length)

### Offset

| Register | Offset |
|---|---|
| DMA_CH1_TxDesc_Ring_Length | 11ACh |

### Function
Contains the length of the transmit descriptor ring.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | TDRL | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 9-0<br>TDRL | Transmit Descriptor Ring Length<br>Sets the maximum number of transmit descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1 K descriptors. NXP recommends a minimum ring descriptor length of 4. For example, you can program any value up to 0x3FF in this field. This field is 10 bits wide, if you program 0x3FF, you can have 1024 descriptors. Program it to a value of 0x9 if you want to have 10 descriptors. |

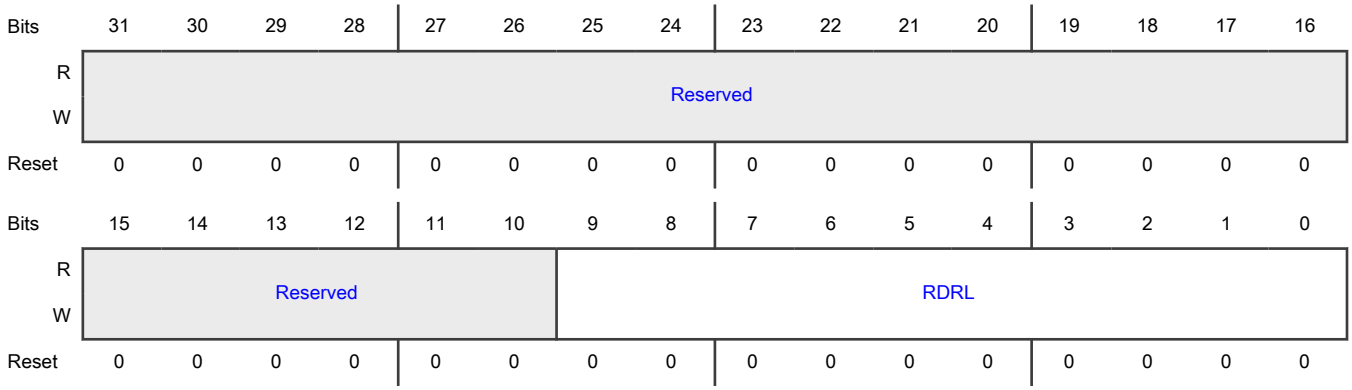
## 72.17.273 DMA Channel 1 Rx Descriptor Ring Length (DMA_CH1_RxDesc_Ring_Length)

### Offset

| Register | Offset |
|---|---|
| DMA_CH1_RxDesc_Ring_Length | 11B0h |

### Function
Contains the length of the receive descriptor circular ring.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-10<br>— | Reserved. |
| 9-0<br>RDRL | Receive Descriptor Ring Length<br>Sets the maximum number of receive descriptors in the circular descriptor ring. The maximum number of descriptors are limited to 1 K descriptors. For example, you can program any value up to 0x3FF in this field. This field is 10 bits wide, if you program 0x3FF, you can have 1024 descriptors. Program it to a value of 0x9 if you want to have 10 descriptors. |

## 72.17.274 DMA Channel 1 Interrupt Enable (DMA_CH1_Interrupt_Enable)

### Offset

| Register | Offset |
|---|---|
| DMA_CH1_Interrupt_Enable | 11B4h |

### Function
Enables the interrupts which the status register reports.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | NIE | AIE | CDEE | FBEE | ERIE | ETIE | RWTE | RSE | RBUE | RIE | Reserved | | | TBUE | TXSE | TIE |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved. |
| 15<br>NIE | Normal Interrupt Summary Enable<br>Enables or disables the normal interrupt summary.<br>When this field is 1, it enables the normal interrupt summary. This field also enables the following interrupts in DMA_CH0_Status:<br>Bit 0 - Transmit interrupt<br>Bit 2 - Transmit buffer unavailable<br>Bit 6 - Receive interrupt<br>Bit 11 - Early receive interrupt<br>When this field becomes 0, it disables the normal interrupt summary.<br>0b - Disable<br>1b - Enable |
| 14 | Abnormal Interrupt Summary Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| AIE | Enables or disables the abnormal interrupt summary.<br>When this field is 1, it enables the abnormal interrupt summary. This field also enables the following interrupts in DMA_CH0_Status:<br>Bit 1 - Transmit process stopped<br>Bit 7 - Rx buffer unavailable<br>Bit 8 - Receive process stopped<br>Bit 9 - Receive watchdog timeout<br>Bit 10 - Early transmit interrupt<br>Bit 12 - Fatal bus error<br>Bit 13 - Context descriptor error<br>When this field becomes 0, it disables the abnormal interrupt summary.<br>0b - Disable<br>1b - Enable |
| 13<br>CDEE | Context Descriptor Error Enable<br>Enables or disables the context descriptor error.<br>When this field is 1 along with DMA_CH0_Interrupt_Enable[AIE], it enables the descriptor error interrupt. When this field becomes 0, it disables the descriptor error interrupt.<br>0b - Disable<br>1b - Enable |
| 12<br>FBEE | Fatal Bus Error Enable<br>Enables or disables fatal bus error.<br>When this field is 1 along with DMA_CH0_Interrupt_Enable[AIE], it enables the fatal bus error interrupt. When this field becomes 0, it disables the fatal bus error interrupt.<br>0b - Disable<br>1b - Enable |
| 11<br>ERIE | Early Receive Interrupt Enable<br>Enables or disables the early receive interrupt.<br>When this field is 1 along with DMA_CH0_Interrupt_Enable[NIE], it enables the early receive interrupt. When this field becomes 0, it disables the early receive interrupt.<br>0b - Disable<br>1b - Enable |
| 10<br>ETIE | Early Transmit Interrupt Enable<br>Enables or disables the early transmit interrupt. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When this field is 1 along with DMA_CH0_Interrupt_Enable[AIE], it enables the early transmit interrupt. When this field becomes 0, it disables the early transmit interrupt.<br>0b - Disable<br>1b - Enable |
| 9<br>RWTE | Receive Watchdog Timeout Enable<br>Enables or disables the receive watchdog timeout interrupt.<br>When this field is 1 along with DMA_CH0_Interrupt_Enable[AIE], it enables the receive watchdog timeout interrupt. When this field becomes 0, it disables the receive watchdog timeout interrupt.<br>0b - Disable<br>1b - Enable |
| 8<br>RSE | Receive Stopped Enable<br>Enables or disables the receive stopped interrupt.<br>When this field is 1 along with DMA_CH0_Interrupt_Enable[AIE], It enables the receive stopped interrupt. When this field becomes 0, it disables the receive stopped interrupt.<br>0b - Disable<br>1b - Enable |
| 7<br>RBUE | Receive Buffer Unavailable Enable<br>Enables or disables the receive buffer unavailable interrupt.<br>When this field is 1 along with DMA_CH0_Interrupt_Enable[AIE], it enables the receive buffer unavailable interrupt. When this field becomes 0, it disables the receive buffer unavailable interrupt.<br>0b - Disable<br>1b - Enable |
| 6<br>RIE | Receive Interrupt Enable<br>Enables or disables the receive interrupt.<br>When this field is 1 along with DMA_CH0_Interrupt_Enable[NIE], it enables the receive interrupt. When this field becomes 0, it disables the receive interrupt.<br>0b - Disable<br>1b - Enable |
| 5-3<br>— | Reserved. |
| 2<br>TBUE | Transmit Buffer Unavailable Enable<br>Enables or disables the transmit buffer unavailable interrupt. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When this field is 1 along with DMA_CH0_Interrupt_Enable[NIE], it enables the transmit buffer unavailable interrupt. When this field becomes 0, it disables the transmit buffer unavailable interrupt.<br>0b - Disable<br>1b - Enable |
| 1<br>TXSE | Transmit Stopped Enable<br>Enables or disables the transmit stopped interrupt.<br>When this field is 1 along with DMA_CH0_Interrupt_Enable[AIE], it enables the transmission stopped interrupt. When this field becomes 0, it disables the transmission Stopped interrupt.<br>0b - Disable<br>1b - Enable |
| 0<br>TIE | Transmit Interrupt Enable<br>Enables or disables the transmit interrupt.<br>When this field is 1 along with DMA_CH0_Interrupt_Enable[NIE], it enables the transmit interrupt. When this field becomes 0, it disables the transmit interrupt.<br>0b - Disable<br>1b - Enable |

## 72.17.275 DMA Channel 1 Rx Interrupt Watchdog Timer (DMA_CH1_Rx_Interrupt_Watchdog_Timer)

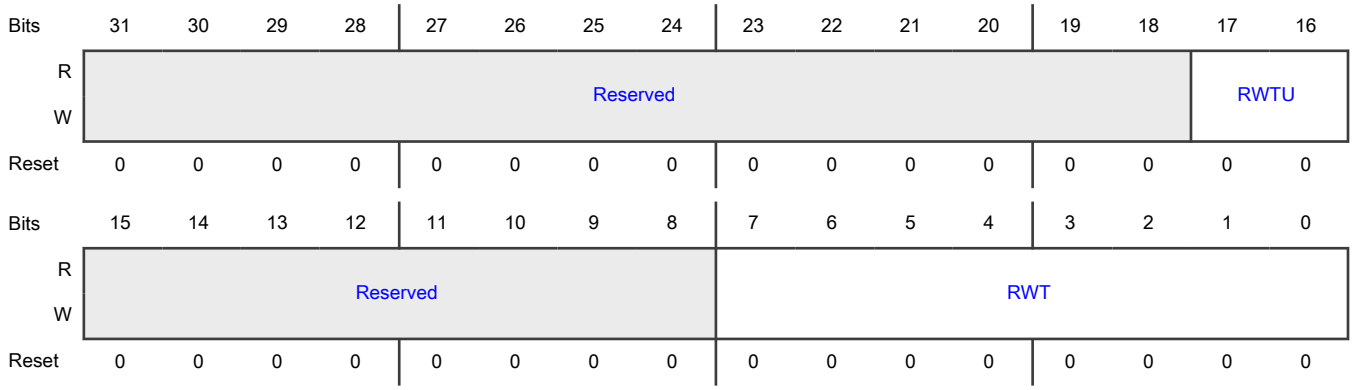
### Offset

| Register | Offset |
|---|---|
| DMA_CH1_Rx_Interrupt_Watchdog_Timer | 11B8h |

### Function

Indicates the watchdog timeout for receive interrupt (RI) from DMA. When you write this register with a non-zero value, it enables the watchdog timer for the RI bit of the DMA_CHi_Status register.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-18<br>— | Reserved. |
| 17-16<br>RWTU | Receive Interrupt Watchdog Timer Count Units<br>Indicates the number of system clock cycles corresponding to one unit in DMA_CH1_Rx_Interrupt_Watchdog_Timer[RWT].<br>2'b00 - 256<br>2'b01 - 512<br>2'b10 - 1024<br>2'b11 - 2048<br>For example, the watchdog timer sets for 2*512=1024 system clock cycles, when DMA_CH1_Rx_Interrupt_Watchdog_Timer[RWT] = 2 and DMA_CH1_Rx_Interrupt_Watchdog_Timer[RWTU] = 1. |
| 15-8<br>— | Reserved. |
| 7-0<br>RWT | Receive Interrupt Watchdog Timer Count<br>Indicates the number of system clock cycles, multiplied by factor indicated in DMA_CH1_Rx_Interrupt_Watchdog_Timer[RWTU], for which the watchdog timer sets.<br>The watchdog timer triggers with the programmed value after the receive DMA completes the packet transfer for which the RI bit is not 1 in DMA_CH(#i)_Status register, because of the setting of interrupt enable bit in the corresponding descriptor RDES3[30].<br>RI bit is 1 and the timer stops when the watchdog timer runs out. The watchdog timer becomes 0 when RI bit is 1 because RI sets automatically per the interrupt enable bit RDES3[30] of any received packet. |

## 72.17.276 DMA Channel 1 Slot Function Control Status (DMA_CH1_Slot_Function_Control_Status)

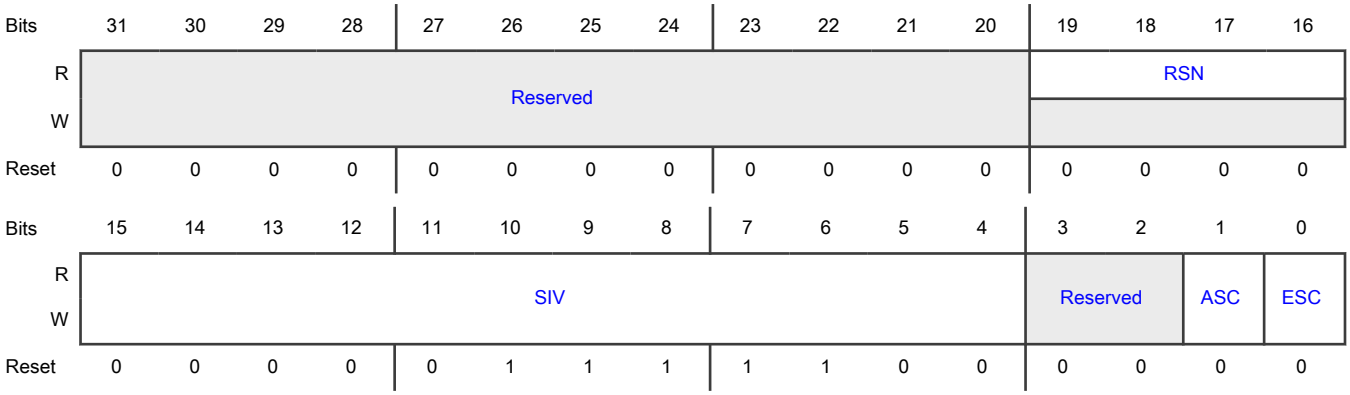
### Offset

| Register | Offset |
|---|---|
| DMA_CH1_Slot_Function_Control_Status | 11BCh |

### Function

Contains the control field for slot function and the status for transmit path.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-20<br>— | Reserved. |
| 19-16<br>RSN | Reference Slot Number<br>Provides the current value of the reference slot number in DMA. It is used for slot comparison. |
| 15-4<br>SIV | Slot Interval Value<br>Controls the period of the slot interval in which the TxDMA fetches the scheduled packets. A value of 0 specifies the slot interval of 1 us while the maximum value 4095 specifies the slot interval of 4096 us. The default or reset value is 0x07C which corresponds to slot interval of 125 us. |
| 3-2<br>— | Reserved. |
| 1<br>ASC | Advance Slot Check<br>Indicates whether an advance slot check is enabled.<br>When this field is 1, it enables DMA to fetch the data from the buffer when the slot number (SLOTNUM) programmed in the transmit descriptor is: |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • Equal to the reference slot number given in DMA_CH1_Slot_Function_Control_Status[RSN], or<br>• Ahead of the reference slot number by up to two slots.<br>This field is applicable only when DMA_CH1_Slot_Function_Control_Status[ESC] = 1.<br>0b - Disabled<br>1b - Enabled |
| 0<br>ESC | Enable Slot Comparison<br>Enables slot comparison.<br>When this field is 1, it enables the checking of the slot numbers programmed in the transmit descriptor with the current reference given in DMA_CH1_Slot_Function_Control_Status[RSN]. DMA fetches the data from the corresponding buffer only when the slot number is:<br>• Equal to the reference slot number, or<br>• Ahead of the reference slot number by one slot.<br>When this field becomes 0, it disables the checking of the slot numbers. DMA fetches the data immediately after you process the descriptor.<br>**NOTE**<br>You must not enable the UFO (UDP Fragmentation over IPv4)/TSO/USO along with TBS/AVB slot number check. The UFO/TSO/USO involves multiple packets or segments or fragments transmission for single packet received from application and the slot number check are applicable for fetching only first segment/fragment. As a result it might be difficult for you to specify slot number for subsequent packets.<br>0b - Disable<br>1b - Enable |

## 72.17.277 DMA Channel 1 Current Application Transmit Descriptor (DMA_CH1_Current_App_TxDesc)

**Offset**

| Register | Offset |
|---|---|
| DMA_CH1_Current_App_TxDesc | 11C4h |

**Function**
Specifies the current transmit descriptor which DMA reads.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>CURTDESAPTR | Application Transmit Descriptor Address Pointer<br>Indicates that DMA updates this pointer during the transmit operation. This pointer is 0 on reset. |

## 72.17.278 DMA Channel 1 Current Application Receive Descriptor (DMA_CH1_Current_App_RxDesc)

### Offset

| Register | Offset |
|---|---|
| DMA_CH1_Current_App_RxDesc | 11CCh |

### Function

Specifies the current receive descriptor which DMA reads.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>CURRDESAPTR | Application Receive Descriptor Address Pointer<br>Indicates that DMA updates this pointer during the receive operation. This pointer is 0 on reset. |

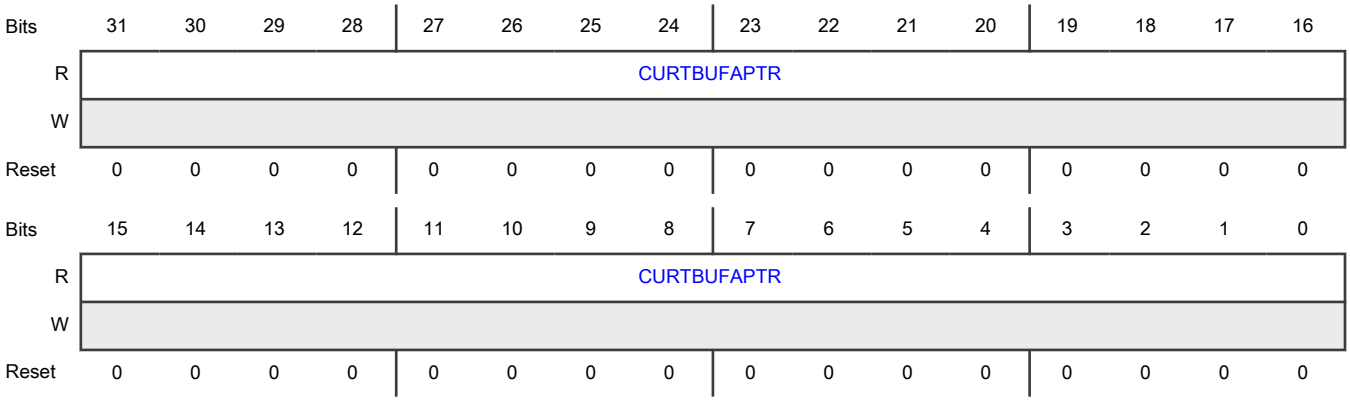
## 72.17.279 DMA Channel 1 Current Application Transmit Buffer (DMA_CH1_Current_App_TxBuffer)

### Offset

| Register | Offset |
|---|---|
| DMA_CH1_Current_App_TxBuffer | 11D4h |

### Function
Specifies the current transmit buffer address which DMA reads.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>CURTBUFAPTR | Application Transmit Buffer Address Pointer<br>Indicates that DMA updates this pointer during the transmit operation. This pointer is 0 on reset. |

## 72.17.280 DMA Channel 1 Current Application Receive Buffer (DMA_CH1_Current_App_RxBuffer)

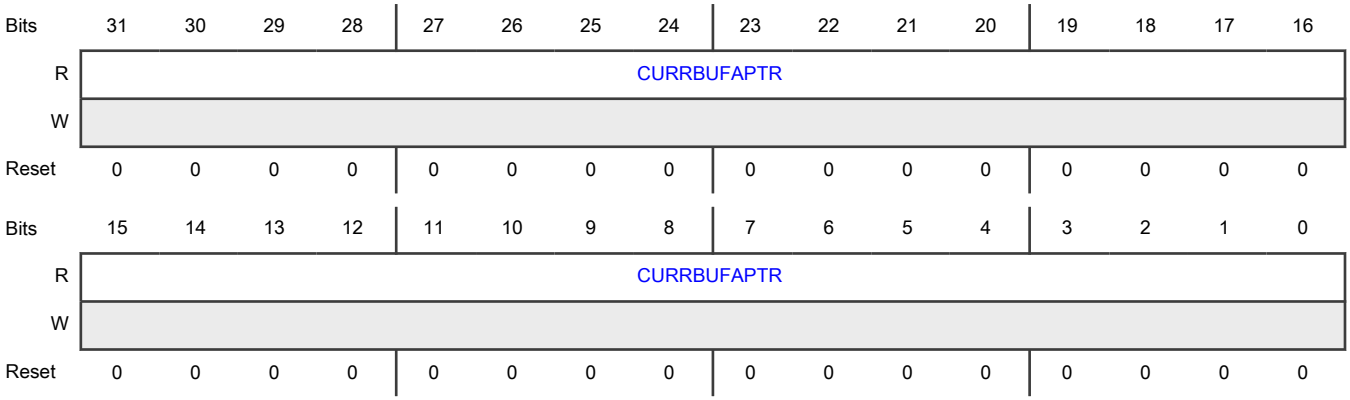
### Offset

| Register | Offset |
|---|---|
| DMA_CH1_Current_App_RxBuffer | 11DCh |

### Function

Indicates that DMA_CH0_Current_App_RxBuffer points to the current receive buffer address which the DMA reads.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>CURRBUFAPTR | Application Receive Buffer Address Pointer<br>Indicates that DMA update this pointer during the receive operation. This pointer is 0 on reset. |

## 72.17.281 DMA Channel 1 Status (DMA_CH1_Status)

### Offset

| Register | Offset |
|---|---|
| DMA_CH1_Status | 11E0h |

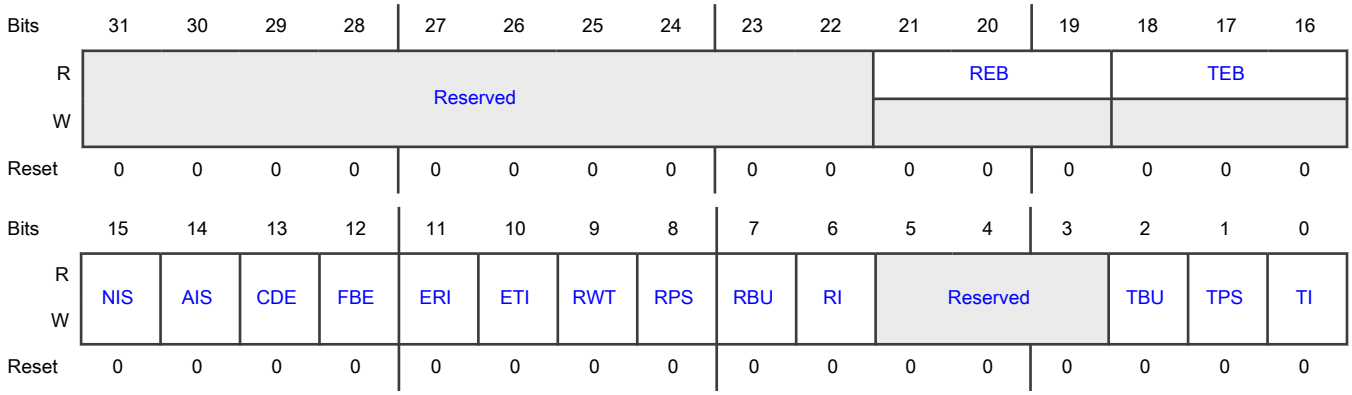
### Function

Indicates that to determine the status of DMA, the application reads the status register during an interrupt service routine or polling.

**NOTE**

The number of DMA_CH(#i)_Status register in the configuration is the higher of number of receive DMA Channels and transmit DMA Channels.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-22<br>— | Reserved. |
| 21-19<br>REB | Rx DMA Error Bits<br>Indicates the type of error that causes a bus error. For example, an error response on the AHB or AXI interface.<br>Bit 21<br>1'b1 - Error during data transfer by Rx DMA<br>1'b0 - No Error during data transfer by Rx DMA<br>Bit 20<br>1'b1 - Error during descriptor access<br>1'b0 - Error during data buffer access<br>Bit 19<br>1'b1 - Error during read transfer<br>1'b0 - Error during write transfer<br>This field is valid only when DMA_CH1_Status[FBE] = 1. It does not generate an interrupt. |
| 18-16<br>TEB | Tx DMA Error Bits<br>Indicates the type of error that causes a bus error. For example, error response on the AHB or AXI interface.<br>Bit 18<br>1'b1 - Error during data transfer by Tx DMA<br>1'b0 - No Error during data transfer by Tx DMA<br>Bit 17<br>1'b1 - Error during descriptor access |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1'b0 - Error during data buffer access<br>Bit 16<br>1'b1 - Error during read transfer<br>1'b0 - Error during write transfer<br>This field is valid only when DMA_CH1_Status[FBE] = 1. It does not generate an interrupt. |
| 15<br>NIS | Normal Interrupt Summary<br>Indicates whether the normal interrupt summary status is detected.<br>The field value is the logical OR of the following bits when the corresponding interrupt bits enable in DMA_CH0_Interrupt_Enable:<br>Bit 0 - Transmit interrupt<br>Bit 2 - Transmit buffer unavailable<br>Bit 6 - Receive interrupt<br>Bit 11 - Early receive interrupt<br>Only unmasked bits (interrupts for which interrupt enable sets in DMA_CH0_Interrupt_Enable) affect this field.<br>This is a sticky field. You must clear this field (by writing 1 to this field) each time a corresponding bit which sets it, clears.<br>Access restriction apply to this field. Automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 14<br>AIS | Abnormal Interrupt Summary<br>Indicates whether an abnormal interrupt summary status is detected.<br>The field value is the logical OR of the following bits when the corresponding interrupt bits enable in DMA_CH0_Interrupt_Enable:<br>Bit 1 - Transmit process stopped<br>Bit 7 - Receive buffer unavailable<br>Bit 8 - Receive process stopped<br>Bit 10 - Early transmit interrupt<br>Bit 12 - Fatal bus error<br>Bit 13 - Context descriptor error<br>Only unmasked bits affect this field.<br>This is a sticky bit. You must clear this field (by writing 1 to this bit) each time a corresponding bit, which sets it, clears. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Access restriction apply to this field. Automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 13<br>CDE | Context Descriptor Error<br>Indicates whether the context descriptor error status is detected.<br>This field indicates that the DMA Tx/Rx engine receive a descriptor error, which indicates an invalid context in the middle of packet flow ( intermediate descriptor) or all one's descriptor in transmit case and on receive side. It indicates that DMA has read a descriptor with either of the buffer address as ones which you can consider invalid.<br>Access restriction apply to this field. Automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 12<br>FBE | Fatal Bus Error<br>Indicates whether the fatal bus error status is detected.<br>This field indicates that a bus error has occurred (as described in the EB field). When this field is 1, it indicates that the corresponding DMA channel engine disables all bus accesses.<br>Access restriction apply to this field. Automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 11<br>ERI | Early Receive Interrupt<br>Indicates whether an early receive interrupt status is detected.<br>When this field is 1, it indicates that the RxDMA has completed the packet data transfer to the memory.<br>In configs supporting ERIC, When DMA_CH1_Rx_Control[ERIC] = 0, this field is 1 only after the Rx DMA fills a complete receive buffer with packet data. When DMA_CH1_Rx_Control[ERIC] = 1, this field is 1 after every burst data transfer from the receive DMA to the buffer.<br>If DMA_CH1_Status[RI] = 1, this field clears automatically.<br>Access restriction apply to this field. Automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 10<br>ETI | Early Transmit Interrupt<br>Indicates whether an early transmit interrupt status is detected. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When this field is 1, it indicates that the TxDMA has completed the packet data transfer to the MTL TXFIFO memory.<br>In configs supporting ERIC: When DMA_CH1_Tx_Control[ETIC] = 0, this field is 1 only after the Tx DMA transfers a complete packet to MTL. When DMA_CH1_Tx_Control[ETIC] = 1, this field is 1 after the packet data transfer completes (partial) from buffers in the Transmit descriptor in which IOC=1.<br>Access restriction apply to this field. Automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 9<br>RWT | Receive Watchdog Timeout<br>Indicates whether a receive watchdog timeout status is detected.<br>This field asserts when it receives a packet with length greater than 2,048 bytes (10,240 bytes when Jumbo Packet mode is enabled).<br>0b - Not detected<br>1b - Detected |
| 8<br>RPS | Receive Process Stopped<br>Indicates whether a receive process stopped status is detected.<br>This field asserts when the receive process enters the stopped state.<br>Access restriction apply to this field. Automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 7<br>RBU | Receive Buffer Unavailable<br>Indicates whether a receive buffer unavailable status is detected.<br>This field indicates that the application owns the next descriptor in the receive list, and the DMA cannot acquire it. The receive process is suspended. To resume processing receive descriptors, the application must change the ownership of the descriptor and issue a receive poll demand command. If this command is not issued, the receive process resumes when you receives the next recognized incoming packet. In Ring mode, the application must advance the receive descriptor tail pointer register of a channel. This field is 1 only when the DMA owns the previous receive descriptor.<br>Access restriction apply to this field. Automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 6<br>RI | Receive Interrupt<br>Indicates whether the receive interrupt status is detected. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | This field indicates that the packet reception is complete. Bit 31 of RDES3 resets in the last descriptor, and updates the specific packet status information in the descriptor, when the packet reception is complete.<br>The reception remains in the running state.<br>Access restriction apply to this field. Automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 5-3<br>— | Reserved. |
| 2<br>TBU | Transmit Buffer Unavailable<br>Indicates whether the transmit buffer unavailable status is detected.<br>This field indicates that the application owns the next descriptor in the transmit list, and the DMA cannot acquire it. Transmission is suspended. DMA_Debug_Status0[TPS0] explains the transmit process state transitions.<br>The application must perform these actions, to resume the processing of transmit descriptors:<br>1. Write 1 to bit 31 of TDES3 to change the ownership of the descriptor.<br>2. Issue a transmit poll demand command.<br>For Ring mode, the application must advance the transmit descriptor tail pointer register of a channel.<br>Access restriction apply to this field. Automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 1<br>TPS | Transmit Process Stopped<br>Indicates that the transmit process stopped status is detected.<br>This field is 1 when the transmission stops.<br>Access restriction apply to this field. Automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect.<br>0b - Not detected<br>1b - Detected |
| 0<br>TI | Transmit Interrupt<br>Indicates whether the transmit interrupt status is detected.<br>This field indicates that the packet transmission is complete. When transmission completes, bit 31 of TDES3 resets in the last descriptor, and the specific packet status information is updated in the descriptor.<br>Access restriction apply to this field. Automatically becomes 1 on an internal event occurrence. Writing 1 clears this field and writing 0 has no effect. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Not detected<br>1b - Detected |

## 72.17.282 DMA Channel 1 Miss Frame Counter (DMA_CH1_Miss_Frame_Cnt)

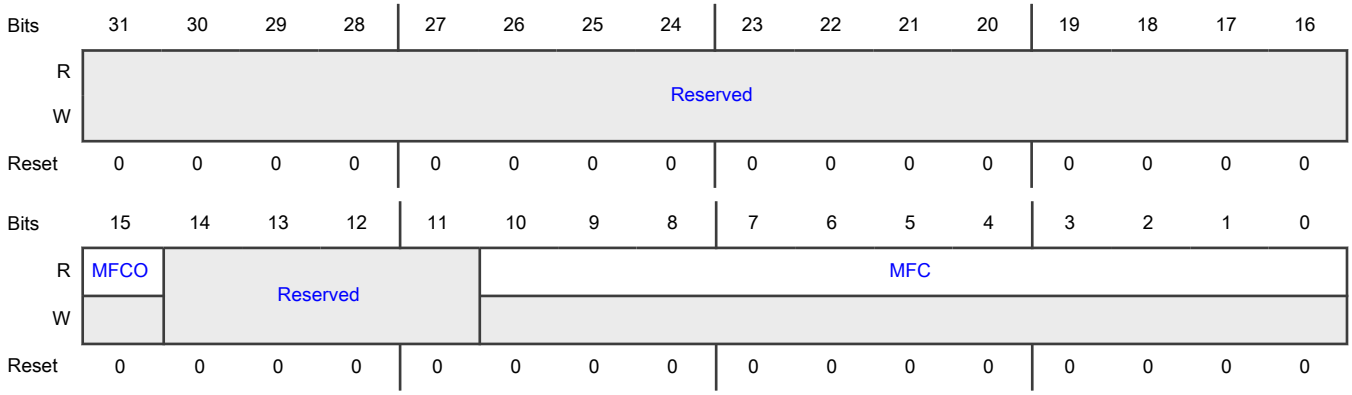
### Offset

| Register | Offset |
|---|---|
| DMA_CH1_Miss_Frame_Cnt | 11E4h |

### Function

Provides the number of packet counter that the DMA drops either because of bus error or programming RPF field in DMA_CH${i}_Rx_Control register.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved. |
| 15<br>MFCO | Overflow status of the MFC Counter<br>Indicates whether the miss frame counter overflow has occurred.<br>When this field is 1, it indicates that the MFC counter does not increments further.<br>This field becomes 0 when this register is read.<br>Access restriction apply to this field. It clears on read and automatically becomes 1 on an internal event occurrence. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Not occurred<br>1b - Occurred |
| 14-11<br>— | Reserved. |
| 10-0<br>MFC | Dropped Packet Counters<br>Indicates the number of packet counters that DMA drops either because of bus error or programing RPF field in DMA_CH${i}_Rx_Control register. This counter is 0 when this register is read.<br>Access restriction apply to this field. It clears on read and automatically becomes 1 on an internal event occurrence. |

## 72.17.283 DMA Channel 1 Rx Parser Accept Count (DMA_CH1_RXP_Accept_Cnt)

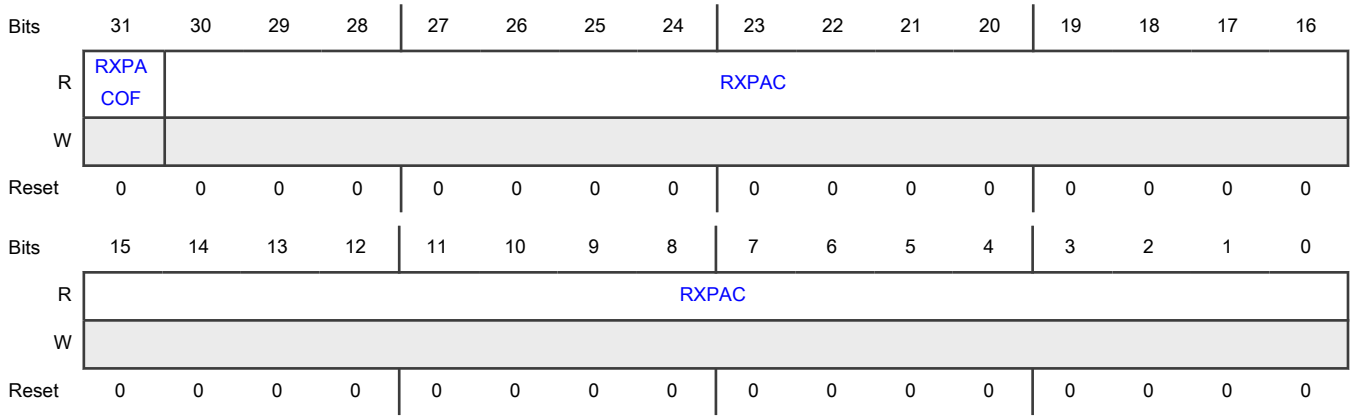
### Offset

| Register | Offset |
|---|---|
| DMA_CH1_RXP_Accept_Cnt | 11E8h |

### Function

Provides the count of the number of frames which the receive parser accepts.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31 | Rx Parser Accept Counter Overflow Bit |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| RXPACOF | Indicates whether the receive parser accept counter overflow have occurred.<br>When this field is 1, it indicates that the DMA_CH1_RXP_Accept_Cnt[RXPAC] has crossed the maximum limit.<br>Access restriction apply to this field. It clears on read and automatically becomes 1 on an internal event occurrence.<br>0b - Not occurred<br>1b - Occurred |
| 30-0<br>RXPAC | Rx Parser Accept Counter<br>Implements whenever a receive parser accept a packet because AF = 1. The counter clears when the register is read. |

## 72.17.284 DMA Channel 1 Rx ERI Count (DMA_CH1_RX_ERI_Cnt)

### Offset

| Register | Offset |
|---|---|
| DMA_CH1_RX_ERI_Cnt | 11ECh |

### Function

Provides the count of the number of times ERI asserts.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | Reserved | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

### Fields

| Field | Function |
|---|---|
| 31-12 | Reserved. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 11-0<br>ECNT | ERI Counter<br>ERI Counter<br>When ERIC bit of DMA_CH(#i)_RX_Control register is 1, it indicates that this counter increments for burst transfer which the Rx DMA completes from the start of packet transfer. This counter resets at the start of new packet. |

## 72.18 Glossary

**AFM** Address filtering module

**ARI** Application receive interface

**ATI** Application transmit interface

**AVB** Audio video bridging

**AXI** Advanced extensible interface

**BTR** Base time register

**CPT** Current presentation time

**CRC** Cyclic redundancy check

**DA** Destination address

**DCB** Data center bridging

**DUT** Device under test

**EOP** End of packet

**FCS** A 32-bit cyclic redundancy check used to detect an in-transit corruption of data. Also marks the end of an Ethernet frame.

**FIFO** First in first out

**GCL** Gate control list

**GMII** Gigabit media independent interface

**MAC** Media access controller

**MCI** MAC control interface

**MDC** Management data clock

**MDIO** Management data input/output

**MII** Media independent interface

**MRI** MAC receive interface

**MTI** MAC transmit interface

**MTL** MAC transmit layer

**PBL** Programmable burst length

| | |
|---|---|
| **PHY** | The physical interface transceiver that implements the OSI physical layer for an ethernet network. The IEEE 802.3 standard defines it. |
| **RGMII** | Reduced gigabit media independent interface |
| **RMII** | Reduced media independent interface |
| **RMON** | Remote network monitoring |
| **SA** | Source address |
| **SFD** | Start frame delimiter |
| **SGMII** | Serial gigabit media independent interface |
| **SMA** | Station management agent |
| **TCP** | Transmission control protocol |
| **UDP** | User datagram protocol |
| **VLAN** | Virtual local area network |

# Chapter 73
# Low Power Universal Asynchronous Receiver/ Transmitter (LPUART)

## 73.1 Chip-specific LPUART information

### 73.1.1 LPUART instances and configuration

Table 542. LPUART instances

| Instance | MCXE31B | MCXE315/MCXE316 | MCXE317 |
|---|---|---|---|
| LPUART_0 | Yes | Yes | Yes |
| LPUART_1 | Yes | Yes | Yes |
| LPUART_2 | Yes | Yes | Yes |
| LPUART_3 | Yes | Yes | Yes |
| LPUART_4 | Yes | No | Yes |
| LPUART_5 | Yes | No | Yes |
| LPUART_6 | Yes | No | Yes |
| LPUART_7 | Yes | No | Yes |
| LPUART_8 | Yes | No | No |
| LPUART_9 | Yes | No | No |
| LPUART_10 | Yes | No | No |
| LPUART_11 | Yes | No | No |
| LPUART_12 | Yes | No | No |
| LPUART_13 | Yes | No | No |
| LPUART_14 | Yes | No | No |
| LPUART_15 | Yes | No | No |

Table 543. LPUART Configuration

| Feature | Configuration |
|---|---|
| | MCXE317 |
| TX FIFO size | 4 Words |
| RX FIFO size | 4 Words |
| Functionality supported | Standard LPUART functionality with MODEM/IrDA |
| | LIN master and slave operation |

**NOTE**

LPUART instances are not available during Standby.

LPUART[0:2]_trg_input is coming from TRGMUX and you should take care of the pulse width of trigger. It should follow the requirement mentioned in section "Peripheral Triggers". These triggers are not present in MCXE31B.

## 73.2 Overview

LPUART provides asynchronous, serial communication capabilities with external devices. It supports the non-return-to-zero (NRZ) encoding format and infrared data association (IrDA)-compatible, low-speed serial infrared (SIR) protocol. LPUART can continue operating when the processor is in Low-Power mode, if an appropriate peripheral clock is available.

### 73.2.1 Block diagram

Figure 423 shows the transmitter portion of LPUART.

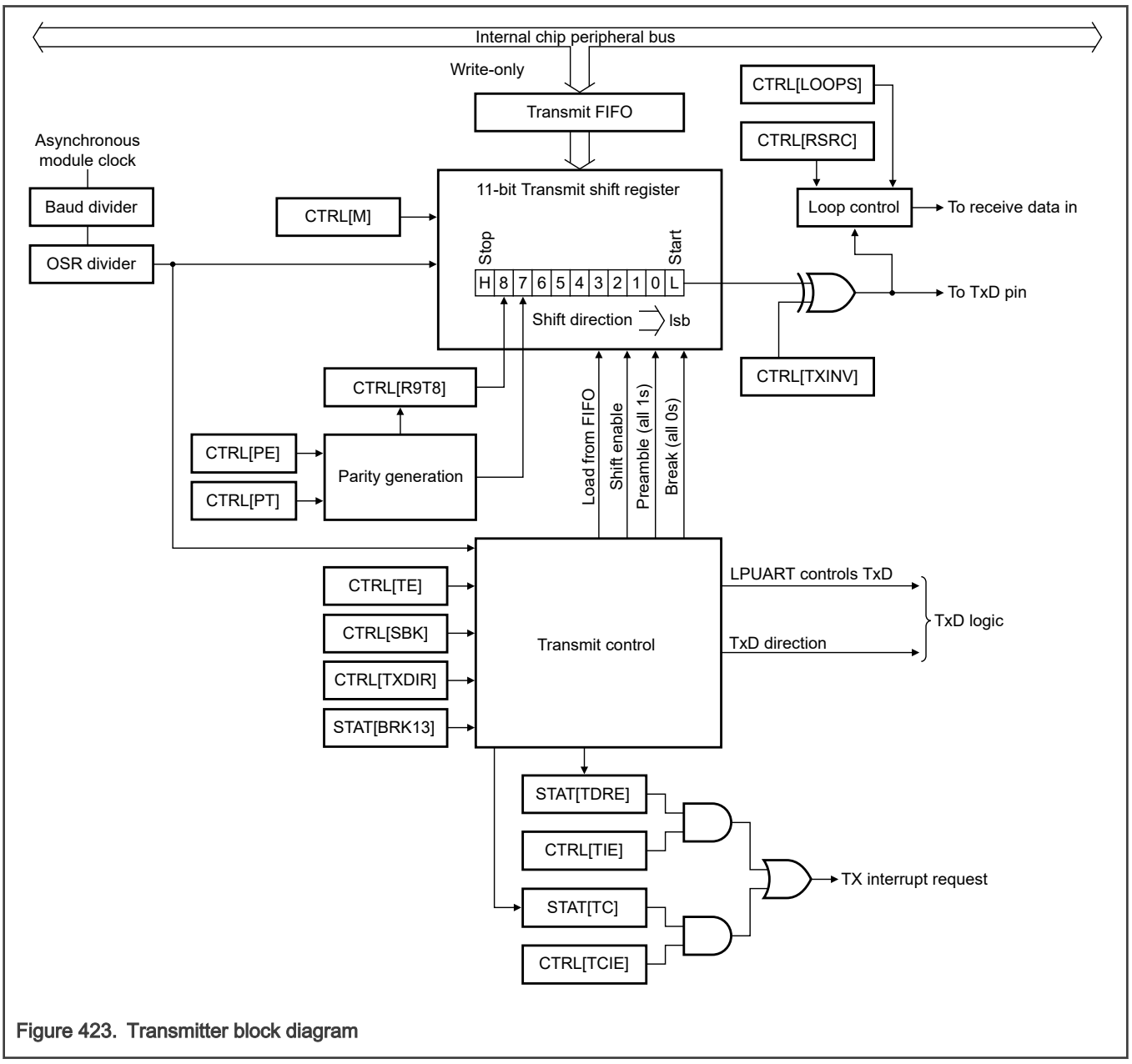



**Figure 423. Transmitter block diagram**

Figure 424 shows the receiver portion of LPUART.

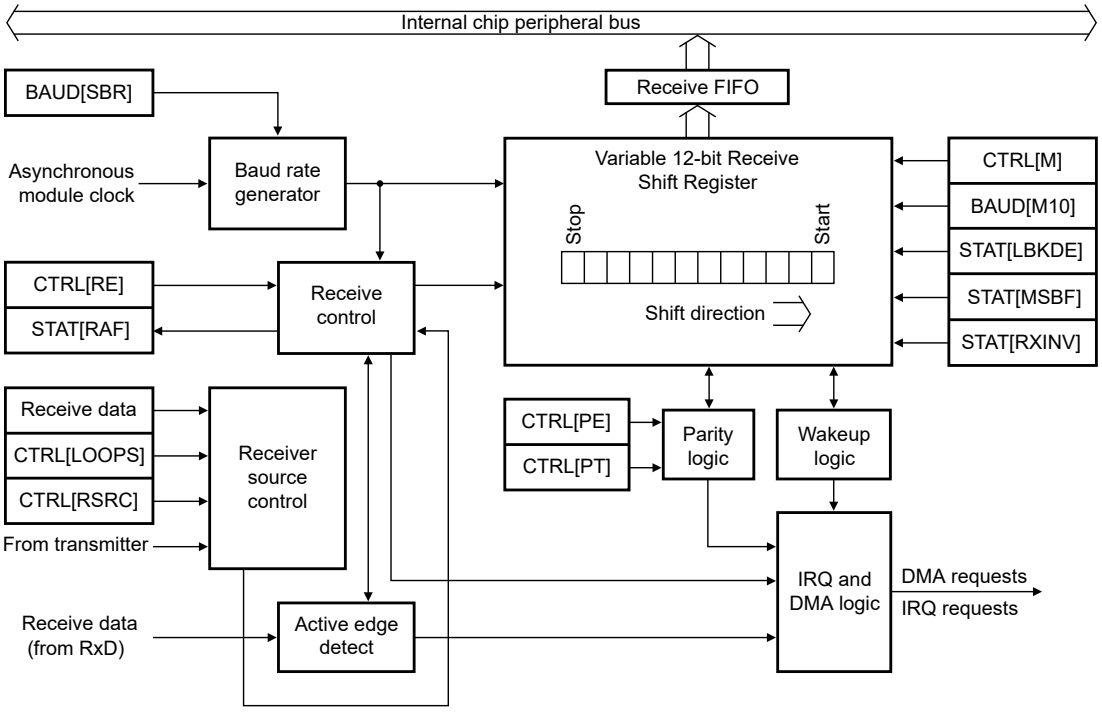



**Figure 424. Receiver block diagram**

## 73.2.2 Features

- Full-duplex, standard NRZ format
- Programmable baud rates (13-bit modulo divider) with a configurable oversampling ratio (OSR) from 4× to 32×
- Asynchronous operation of transmit and receive baud rates with respect to the bus clock:
  - Baud rate can be configured independently of the bus clock frequency.
  - Operation in Low-Power modes is supported.
- Interrupt, DMA, or polled operations:
  - Transmit data empty and transmission complete
  - Receive data full
  - Receive overrun, parity error, framing error, and noise error
  - Idle receiver detect
  - Active edge on receive pin
  - Break detect supporting LIN
  - Receive data match
- Hardware parity generation and checking
- Programmable 7-bit, 8-bit, 9-bit, or 10-bit character length
- Programmable 1-bit or 2-bit stop bits
- Support for three receiver wake-up methods:

  - Idle line wake-up
  - Address mark wake-up
  - Receive data match
- Automatic address matching to reduce ISR overhead:
  - Address mark matching
  - Idle line address matching
  - Address match start, address match end
- Optional 13-bit and 11-bit break character generation
- Configurable idle length detection supporting 1, 2, 4, 8, 16, 32, 64, or 128 idle characters
- Selectable transmitter output and receiver input polarity
- Hardware flow control support for request to send (RTS) and clear to send (CTS) signals
- Selectable IrDA 1.4 return-to-zero-inverted (RZI) format with a programmable pulse width
- Independent FIFO structure for transmit and receive functions:
  - Separate configurable watermarks for receive and transmit requests
  - Option for receiver to assert request after a configurable number of idle characters, if receive FIFO is not empty

## 73.3 Functional description

LPUART supports full-duplex, asynchronous, NRZ serial communication and comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. The following sections describe all LPUART blocks.

### 73.3.1 Baud rate generation

A 13-bit modulus counter in the baud rate generator derives the baud rate for both the receiver and transmitter. The value, ranging from 1 to 8191, written to BAUD[SBR] determines the baud clock divisor for the asynchronous LPUART baud clock. The baud rate clock drives the receiver, while a bit clock, generated from the baud rate clock divided by the OSR, drives the transmitter. Depending on the OSR, the receiver has an acquisition rate of 4 to 32 samples per bit time. LPUART requires BAUD[SBR] and BAUD[OSR] to accurately match the baud rate requirement. There is a relationship between the required baud rate and the input clock frequency, SBR, and OSR. See the below figure for details.

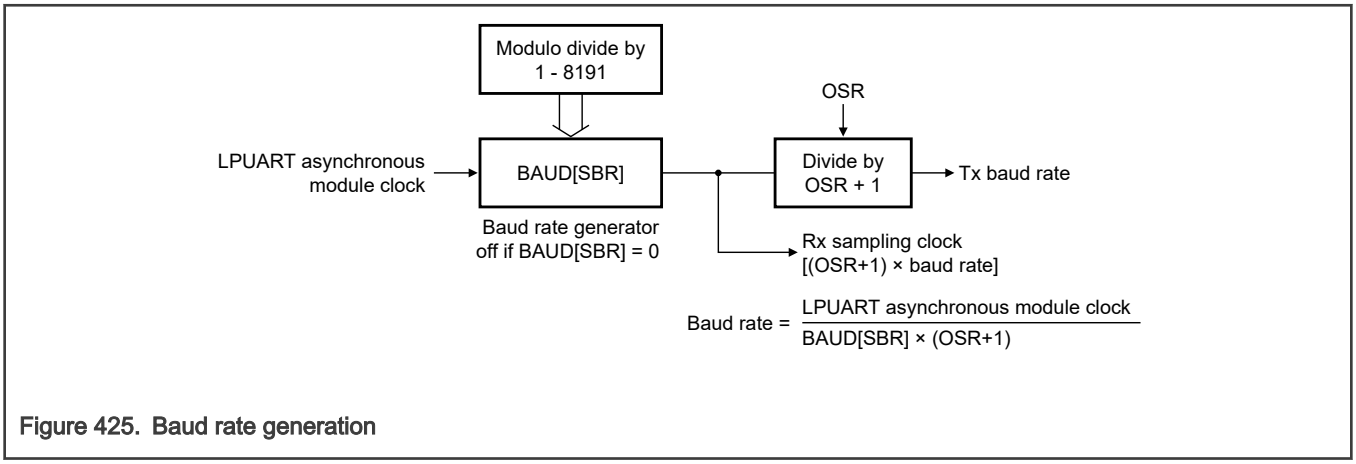



**Figure 425. Baud rate generation**

Baud rate generation is subject to these sources of error:

- Integer division of the asynchronous LPUART baud clock may not give the exact target frequency.
- Synchronization with the asynchronous LPUART baud clock can lead to a phase shift.

Baud rate generation is a free-running counter that continues whenever the transmitter or receiver is enabled. The transmitter bit clock continues whenever the transmitter is enabled; each transmitted character aligns to the next edge of the transmit bit clock.

In general, configuring OSR for a higher ratio and/or sampling on both edges of the clock slightly improves LPUART's tolerance to baud rate mismatch between the received data and LPUART configured baud rate. However, the three data samples in each bit (see Data sampling technique) are also closer together, which may impact noise sensitivity.

### 73.3.2 Baud rate tolerance

A transmitting device may operate at a baud rate below or above that of the receiver.

Accumulated bit time misalignment can cause one of the three stop bit data samples to fall outside the actual stop bit. A noise error will occur if the three samples are not all the same logical values. A framing error will occur if the receiver clock is misaligned in such a way that the majority of the three stop bit samples are a logic zero.

As the receiver samples an incoming frame, it may resynchronize the oversampling clock on any valid falling edge within the frame. Resynchronization within frames will correct a misalignment between transmitter bit times and receiver bit times.

In general, increasing the number of samples per bit will increase the baud rate tolerance and decreasing the number of samples per bit will reduce the baud rate tolerance. Note that since LPUART implements triple voting on consecutive receive data samples, increasing the number of samples per bit will move those samples closer together which would reduce the width of noise that can be filtered by the triple voting logic.

### 73.3.3 Calculating baud rate tolerance

Using the following definitions:

- SAM is the number of sample points per bit (valid range from 8 to 32; equal to (OSR + 1) × (BOTHEDGE + 1)).
- BIT is the number of bits in a character including start, data and stop bits (valid range from 9 to 13).

The ideal baud rate tolerance can be calculated as follows:

- Slow data rate tolerance = ( (SAM ÷ 2) - 1) ÷ ((SAM × BIT) - (SAM ÷ 2) + 2)
- Fast data rate tolerance = ( (SAM ÷ 2) - 2) ÷ (SAM × BIT)

As an example, if configured for 8-bit data, 1 stop bit (BIT = 10) and with OSR=0x7 and BOTHEDGE = 1 (SAM = 16):

- Slow data rate tolerance = (8 - 1) ÷ (160 - 8 + 2) = 7 ÷ 154 = 4.54%
- Fast data rate tolerance = (8 - 2) ÷ 160 = 6 ÷ 160 = 3.75%

If configured for 9-bit data with 1 stop bit (BIT = 11) with same oversampling configuration, then:

- Slow data rate tolerance = (8 - 1) ÷ (176 - 8 + 2) = 7 ÷ 170 = 4.12%
- Fast data rate tolerance = (8 - 2) ÷ 176 = 6 ÷ 176 = 3.41%

**NOTE**

Additional factors can contribute to a lower baud rate tolerance than the ideal. These include clock uncertainty or jitter on the LPUART functional clock source, differences in rise and fall times on the transmitter output and synchronization of the external receive pin to the local LPUART functional clock.

### 73.3.4 Transmitter functional description

This section describes the functioning of the LPUART transmitter, as shown in the transmitter portion of Block diagram, as well as specialized functions for sending break and idle characters.

The transmitter output (TXD) idle state defaults to logic high; the transmitter output is inverted when you write 1 to CTRL[TXINV], which becomes 0 following reset. You can enable the transmitter by writing 1 to CTRL[TE]. This queues a preamble character that is one full character frame of the Idle state. The transmitter then remains idle until data is available in the transmit FIFO and programs store data in the transmit FIFO by writing to Data (DATA).

The central element of the LPUART transmitter is the transmit shift register that is 9-bit to 13-bit long depending on the settings of CTRL[M], CTRL[M7], BAUD[M10], and BAUD[SBNS]. Going forward in this discussion, assume that CTRL[M], CTRL[M7], BAUD[M10], and BAUD[SBNS] are 0, selecting the normal 8-bit Data mode, in which the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new character, the value waiting in transmit FIFO is transferred to the transmit shift register, synchronized with the baud rate clock, and STAT[TDRE] becomes 1 to indicate that another character may be written to the transmit FIFO at Data (DATA).

If no new character is waiting in the transmit FIFO after a stop bit is shifted out of the TXD pin, the transmitter sets the transmit complete flag and enters an idle mode, with TXD high, waiting for more characters to transmit.

Writing 0 to CTRL[TE] does not immediately disable the transmitter. The current transmit activity in progress must first be completed (that could include a data character, idle character, or break character), although the transmitter does not start transmitting another character.

#### 73.3.4.1 Break character length

CTRL[SBK] sends break characters, originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0, 9-bit to 12-bit times, including the start and stop bits. You can enable a longer break of 13-bit times by writing 1 to STAT[BRK13]. Normally, a program waits for STAT[TDRE] to become 1 to indicate that the last character of a message has moved to the transmit shifter. Next, the program writes 1 and then writes 0 to CTRL[SBK]. This action queues a break character to be sent as soon as the shifter is available. If CTRL[SBK] remains 1 when the queued break moves into the shifter, synchronized with the baud rate clock, an additional break character is queued. When LPUART is the receiving module, it receives a break character as 0s in all data bits and a framing error (STAT[FE] = 1) is detected.

You can also transmit a break character by writing to Data (DATA) with DATA[FRETSC] = 1 and the data bits clear. This supports transmitting the break character as part of the normal data stream and also allows DMA to transmit a break character.

When idle line wake-up is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program waits for STAT[TDRE] to become 1 to indicate that the last character of a message has moved to the transmit shifter. Next, write 0 and then write 1 to CTRL[TE]. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while CTRL[TE] becomes 0, the LPUART transmitter does not release control of the TXD pin.

You can also write to Data (DATA) to transmit an idle character, with DATA[FRETSC] and DATA[R9T9] = 1 and the values of all the other fields = 0. This supports transmitting the idle character as part of the normal data stream and also allows DMA to transmit an idle character.

As shown in the following table, STAT[BRK13], CTRL[M], CTRL[M7], BAUD[M10], and BAUD[SBNS] affect the length of the break character.

**Table 544. Break character length**

| STAT[BRK13] | CTRL[M] | BAUD[M10] | CTRL[M7] | BAUD[SBNS] | Break character length (in bit times) |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 10 |
| 0 | 0 | 0 | 0 | 1 | 11 |
| 0 | 0 | 0 | 1 | 0 | 9 |
| 0 | 0 | 0 | 1 | 1 | 10 |
| 0 | 1 | 0 | — | 0 | 11 |
| 0 | 1 | 0 | — | 1 | 12 |
| 0 | — | 1 | — | 0 | 12 |
| 0 | — | 1 | — | 1 | 13 |
| 1 | 0 | 0 | 0 | 0 | 13 |

*Table continues on the next page...*

Table 544. Break character length (continued)

| STAT[BRK13] | CTRL[M] | BAUD[M10] | CTRL[M7] | BAUD[SBNS] | Break character length (in bit times) |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 13 |
| 1 | 0 | 0 | 1 | 0 | 12 |
| 1 | 0 | 0 | 1 | 1 | 12 |
| 1 | 1 | 0 | — | 0 | 14 |
| 1 | 1 | 0 | — | 1 | 14 |
| 1 | — | 1 | — | 0 | 15 |
| 1 | — | 1 | — | 1 | 15 |

#### 73.3.4.2 Hardware flow control

The transmitter supports hardware flow control by gating the transmission with the value of CTS_B. If the CTS operation is enabled, the character is transmitted when CTS_B is asserted. If CTS_B is deasserted in the middle of a transmission with characters remaining in the transmitter FIFO, the character in the transmit shift register is complete. Any characters in the FIFO wait for CTS_B to assert again, and TXD remains in the mark state (idle state) until CTS_B is reasserted. The CTS_B pin must assert for longer than one bit period to guarantee that a new transmission is started when the transmitter is idle with CTS.

If the CTS operation is disabled, the transmitter ignores the state of CTS_B.

The transmitter's CTS_B signal can be enabled even if the same LPUART receiver's RTS_B signal is disabled.

#### 73.3.4.3 Transceiver driver enable

The transmitter can use RTS_B as an enable signal for the driver of an external transceiver. See Transceiver driver enable using RTS_B for details. If the RTS operation is enabled, when a character is placed into an empty transmit shift register, RTS_B asserts 1-bit time before the start bit is transmitted. RTS_B remains asserted for the whole time that the transmit shift register has any characters. RTS_B deasserts 1-bit time after all characters in the transmit FIFO and shift register are completely sent, including the last stop bit. In other words, when RTS_B is used as a transceiver enable, RTS_B asserts 1-bit time before the transmitter starts transmitting and negates 1-bit time after the transmitter goes idle.

Transmitting a break character also asserts RTS_B, with the same assertion and deassertion timing as having a character in the transmit shift register.

The transmitter's RTS_B signal asserts only when the transmitter is enabled. However, the transmitter's RTS_B signal is unaffected by its CTS_B signal. RTS_B remains asserted until the transfer is complete, even if the transmitter is disabled mid-way through a data transfer.

#### 73.3.4.4 Transceiver driver enable using RTS_B

RS-485 is a multiple drop communication protocol in which the LPUART transceiver's driver is three-stated unless LPUART is driving. The transmitter can use the RTS_B signal to enable the driver of a transceiver. The polarity of RTS_B can be matched to the polarity of the transceiver's driver enable signal.

The following figure shows the receiver enable signal asserted. This connection can also connect RTS_B to both DE and RE_B. The transceiver's receiver is disabled when driving. A pullup can pull RXD to a nonfloating value during this time. You can refine this option further by operating LPUART in Single-Wire mode, freeing the RXD pin for other uses.

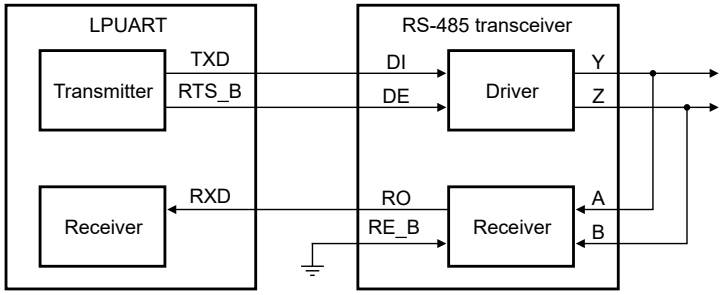



Figure 426. Transceiver driver enable using RTS_B

## 73.3.5 Receiver functional description

This section discusses the functioning of the LPUART receiver, as shown in the receiver portion of Block diagram. The section also discusses:

- The data sampling technique used to reconstruct receiver data.
- Different variations of the receiver wake-up function.

You can invert the receiver input by writing 1 to STAT[RXINV] and enable the receiver by writing 1 to CTRL[RE]. Character frames consist of a start bit of logic 0, along with N (7, 8, 9, 10) bits (MSB or LSB first), and one or two stop bits of logic 1. For information about 7-bit, 9-bit, or 10-bit Data mode, see Data modes. Going forward in this discussion, assume that LPUART is configured for a normal 8-bit Data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full (STAT[RDRF] = 0), the data character is transferred to the receive FIFO, resulting in STAT[RDRF] becoming 1. However, if STAT[RDRF] is already 1, indicating that the receive data buffer is already full, STAT[OR] becomes 1 and the new data is lost.

Because the LPUART receiver is separate from the receive FIFO, the receive shift register can receive the next word when the receive FIFO is full, and it is only at the end of the character that the next data is written into the receive FIFO, potentially triggering the overrun flag if the FIFO is full.

When a program detects that the receive data register is full (STAT[RDRF] = 1), it gets the data from the FIFO by reading Data (DATA). See Interrupts for details about flag clearing.

### 73.3.5.1 Data sampling technique

The LPUART receiver supports a configurable oversampling rate of between 4× and 32× of the baud rate clock for sampling. The receiver starts by considering logic level samples at the oversampling rate times the baud rate to search for a falling edge on the RXD serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The oversampling baud rate clock divides the bit time into 4 to 32 segments from 1 to OSR (where OSR is the configured oversampling ratio). When a falling edge is located, three more samples are taken at (OSR ÷ 2), (OSR ÷ 2) + 1, and (OSR ÷ 2) + 2 to ensure that this is a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes they are synchronized to a received character. If another falling edge is detected before the receiver is considered synchronized, the receiver restarts sampling from the first segment.

The receiver then samples each bit time, including the start and stop bits, at (OSR ÷ 2), (OSR ÷ 2) + 1, and (OSR ÷ 2) + 2, to determine the logic level for that bit. The logic level is interpreted to be that of the majority of the samples taken during the bit time. If any sample in any bit time, including the start and stop bits, in a character frame fails to agree with the logic level for that bit, noise flag (STAT[NF]) becomes 1 when the received character is transferred to the receive FIFO.

When the LPUART receiver is configured to sample on both edges of the baud rate clock (that is, when BAUD[BOTHEDGE] = 1), the number of segments in each received bit is effectively doubled (from 1 to OSR× 2). The start and data bits are then sampled at OSR, OSR + 1, and OSR + 2. You must enable sampling on both edges of the clock for oversampling rates of 4× to 7×. This sampling is optional for higher oversampling rates.

The synchronization feature of LPUART synchronizes the internal oversampling counter with a detected falling edge on the receive signal, and to adjust the data sampling window. The falling edge detection needs three consecutive 1s prior to the "1->0" (one to zero) transition. After the initial falling edge detection for the start bit, the circuit continuously monitors the next falling edge, and resets the counter after another falling edge is detected. This synchronization to the start bit is termed as resynchronization.

When BAUD[RESYNCDIS] is 0, you perform this falling edge detection and resynchronization not only for the start bit but also for the rest of the character reception after the start bit.

When BAUD[RESYNCDIS] is 1, you perform the falling edge detection and resynchronization only for the start bit. The use case for disabling the resynchronization is protocols that require this (for example, LIN 2.1 prohibits resynchronization within a byte).

The following table and figure explain LPUART resynchronization.

**Table 545. LPUART resynchronization settings**

| Resynchronization | BAUD[RESYNCDIS] = 0 | BAUD[RESYNCDIS] = 1 |
|---|---|---|
| For the starting bit falling edge | Yes | Yes |
| For all falling edges after the start bit | Yes | No |

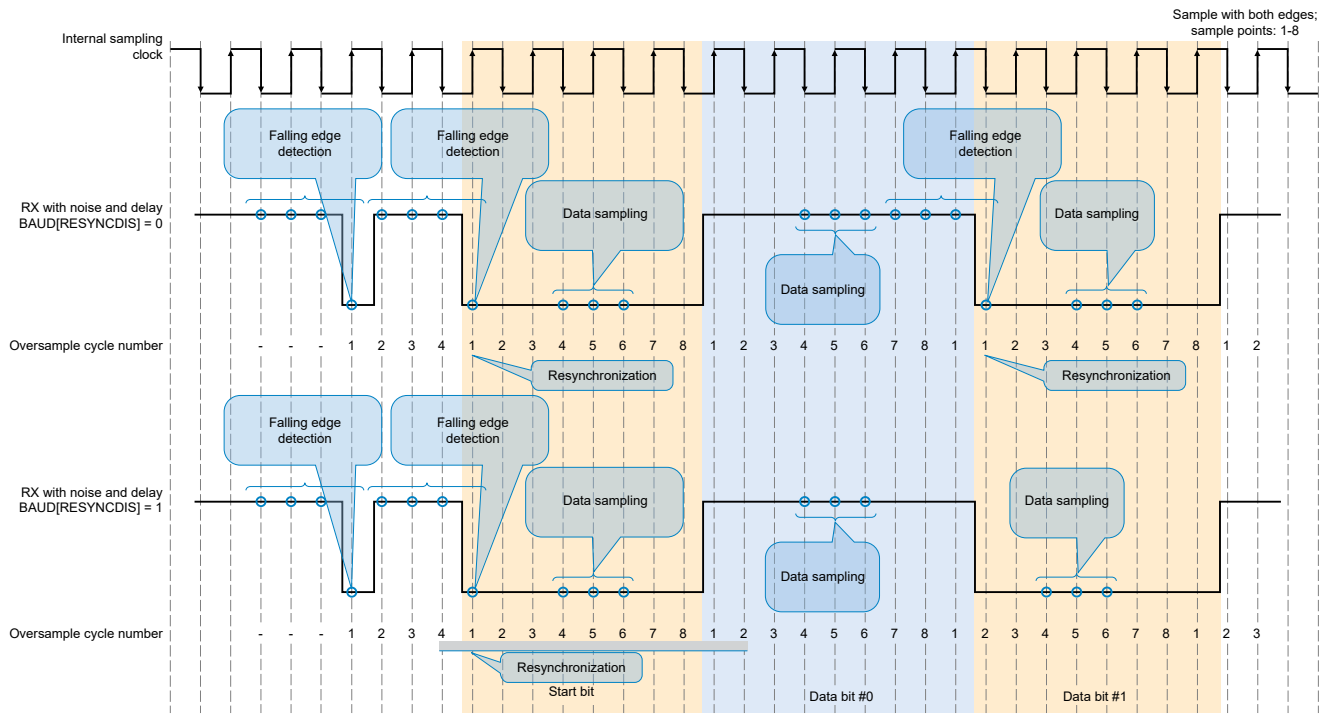



**Figure 427. LPUART resynchronization diagram**

### 73.3.5.2 Receiver wake-up operation

Receiver wake-up and receiver address matching are hardware mechanisms that allow an LPUART receiver to ignore the characters in a message intended for a different receiver.

During receiver wake-up, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write 1 to CTRL[RWU].

When CTRL[RWU] and STAT[RWUID] are 1, the status fields associated with the receiver, with the exception of STAT[IDLE], are inhibited from becoming 1, thus eliminating the software overhead for handling the unimportant message characters. At the end of a message, all receivers automatically force CTRL[RWU] to become 0. This results in all receivers waking up in time to look at the first character(s) of the next message.

During receiver address matching, the address matching is performed in hardware and the LPUART receiver ignores all characters that do not meet the address match requirements.

Table 546. Receiver wake-up options

| CTRL[RWU] | BAUD[MAEN1] \| BAUD[MAEN2] | BAUD[MATCFG] | CTRL[WAKE]: STAT[RWUID] | Receiver wake-up |
|---|---|---|---|---|
| 0 | 0 | X | X | Normal operation |
| 1 | 0 | 00 | 00 | Receiver wake-up on idle line; STAT[IDLE] = 0 |
| 1 | 0 | 00 | 01 | Receiver wake-up on idle line; STAT[IDLE] = 1 |
| 1 | 0 | 00 | 10 | Receiver wake-up on address mark |
| 1 | 1 | 11 | 10 | Receiver wake-up on data match |
| 0 | 1 | 00 | X0 | Address mark address match; STAT[IDLE] = 0 for discarded characters |
| 0 | 1 | 00 | X1 | Address mark address match; STAT[IDLE] = 1 for discarded characters |
| 0 | 1 | 01 | X0 | Idle line address match |
| 0 | 1 | 10 | X0 | Match on and match off; STAT[IDLE] = 0 for discarded characters |
| 0 | 1 | 10 | X1 | Match on and match off; STAT[IDLE] = 1 for discarded characters |

#### 73.3.5.2.1 Idle line wake-up

When CTRL[WAKE] is 0, you can configure the receiver for an idle line wake-up. In this mode, CTRL[RWU] becomes 0 automatically when the receiver detects a full character time of the idle-line level.

CTRL[M], CTRL[M7], and BAUD[M10] select 7-bit to 10-bit Data mode and BAUD[SBNS] selects a 1-bit or 2-bit stop bit number that determines how many bit times of idle are needed to constitute a full character time, 9 to 13 bit times because of the start and stop bits.

When CTRL[RWU] is 1 and STAT[RWUID] is 0, the idle condition that wakes up the receiver does not lead to STAT[IDLE] becoming 1. The receiver wakes up and waits for the first data character of the next message that leads to STAT[RDRF] becoming 1 and generates an interrupt if enabled. When STAT[RWUID] is 1, any idle condition leads to STAT[IDLE] becoming 1 and generates an interrupt if enabled, regardless of whether CTRL[RWU] is 0 or 1.

These are the ways to detect an idle line:

- When CTRL[ILT] is 0, the idle bit counter starts after the start bit so that the stop bit and any logic 1s at the end of a character count to calculate the full character time of idle.
- When CTRL[ILT] is 1, the idle bit counter does not start until after the stop bit time so that the data in the last character of the previous message does not impact the idle detection.

#### 73.3.5.2.2 Address mark wake-up

When CTRL[WAKE] is 1, you can configure the receiver for an address mark wake-up. In this mode, CTRL[RWU] becomes 0 automatically when the receiver detects a logic 1 in the most significant bit of the received character. When parity is enabled, the second most significant bit is used for address mark wake-up.

Address mark wake-up allows messages to contain idle characters, but requires one bit to be reserved for use in address frames. The logic 1 in the most significant bit (or second most significant bit when parity is enabled) of an address frame writes 0 to CTRL[RWU] and writes 1 to STAT[RDRF]. In this case, the character with the address mark bit is received even if the receiver is sleeping during most of this character time.

#### 73.3.5.2.3 Data match wake-up

When CTRL[RWU] and CTRL[WAKE] are 1, and BAUD[MATCFG] equals 11, the receiver is configured for a data match wake-up. In this mode, CTRL[RWU] becomes 0 automatically when the receiver detects a character that matches MATCH[MA1] when BAUD[MAEN1] is 1, or that matches MATCH[MA2] when BAUD[MAEN2] is 1.

#### 73.3.5.2.4 Address match operation

You can enable the address match operation when either BAUD[MAEN1] or BAUD[MAEN2] is 1 and BAUD[MATCFG] is 0. In this function, a character that the RXD pin receives with a logic 1 in the most significant bit (or the second most significant bit when parity is enabled) is considered an address and is compared to the associated MATCH[MA1] or MATCH[MA2]. The character is only transferred to the receive buffer, and STAT[RDRF] becomes 1 if the comparison matches. All subsequent characters received with a logic 0 in the most significant bit (or the second most significant bit when parity is enabled) are considered to be data associated with the address and are transferred to the receive FIFO. If no marked address match occurs, no transfer is made to the receive FIFO, and all the characters that follow, with logic 0 in the most significant bit (or second most significant bit when parity is enabled), are also discarded. If both BAUD[MAEN1] and BAUD[MAEN2] are 0, the receiver operates normally, and all the received data is transferred to the receive FIFO.

The address match operation functions in the same way for both MATCH[MA1] and MATCH[MA2]:

- If either BAUD[MAEN1] or BAUD[MAEN2] is 1, a marked address is compared only to the associated Match Address (MATCH) and data is transferred to the receive FIFO only on a match.
- If both BAUD[MAEN1] and BAUD[MAEN2] are 1, a marked address is compared to both MATCH[MA1] and MATCH[MA2] and data is transferred only on a match with either of these fields.

#### 73.3.5.2.5 Idle match operation

You can enable the idle match operation when either BAUD[MAEN1] or BAUD[MAEN2] is 1 and BAUD[MATCFG] is 1. In this function, the first character that the RXD pin receives after an idle line condition is considered an address and is compared to the associated MATCH[MA1] or MATCH[MA2]. The character is transferred only to the receive buffer, and STAT[RDRF] becomes 1, if the comparison matches. All subsequent characters are considered to be data associated with the address and are transferred to the receive FIFO until the next idle line condition is detected. If no address match occurs, no transfer is made to the receive FIFO, and all the frames that follow, until the next idle condition, are also discarded. If both BAUD[MAEN1] and BAUD[MAEN2] are 0, the receiver operates normally, and all the received data is transferred to the receive FIFO.

An idle match operation functions in the same way for both MATCH[MA1] and MATCH[MA2]:

- If either BAUD[MAEN1] or BAUD[MAEN2] is 1, the first character after an idle line is compared only to the associated Data (DATA) and data is transferred to the receive FIFO only on a match.

- If both BAUD[MAEN1] and BAUD[MAEN2] are 1, the first character after an idle line is compared to both MATCH[MA1] and MATCH[MA2] and data is transferred only on a match with either of these fields.

#### 73.3.5.2.6 Match on, match off operation

The match on, match off operation is enabled when both BAUD[MAEN1] and BAUD[MAEN2] are 1 and BAUD[MATCFG] = 10. In this function, a character that the RXD pin receives matches MATCH[MA1] and is transferred to the receive buffer, and STAT[RDRF] becomes 1. All subsequent characters are considered to be data and are also transferred to the receive FIFO, until a character that matches MATCH[MA2] is received. The character that matches MATCH[MA2], along with all subsequent characters, is discarded; and this continues until another character that matches MATCH[MA1] is received. If both BAUD[MAEN1] and BAUD[MAEN2] are 0, the receiver operates normally, and all the received data is transferred to the receive FIFO.

**NOTE**

The match on, match off operation requires both BAUD[MAEN1] and BAUD[MAEN2] to be 1.

### 73.3.5.3 Hardware flow control

To support hardware flow control, you can program the receiver to automatically assert and deassert RTS_B:

- RTS_B remains asserted until the transfer is complete, even if the transmitter is disabled midway through a data transfer. See Transceiver driver enable using RTS_B for more information.
- If the receiver RTS functionality is enabled, the receiver automatically deasserts RTS_B if STAT[RDRF] is 1 or a start bit is detected that causes STAT[RDRF] to become 1.
- The receiver asserts RTS_B when STAT[RDRF] is 0 and has not detected a start bit that causes STAT[RDRF] to become 1. There is no impact if STAT[RDRF] is 1 already.
- Even if RTS_B is deasserted, the receiver continues to receive characters until the receive FIFO is overrun.
- If the receiver RTS functionality is disabled, the receiver's RTS_B remains deasserted.
- When RTS is driven by receiver, it will be negated when the receiver is disabled irrespective of the FIFO level.

## 73.3.6 Additional LPUART functions

### 73.3.6.1 Data modes

You can configure the LPUART transmitter and receiver to operate in 7-bit Data mode by writing 1 to CTRL[M7], 9-bit Data mode by writing 1 to CTRL[M], or 10-bit Data mode by writing 1 to BAUD[M10]. In 9-bit Data mode, there exists a ninth data bit and in 10-bit mode, there exists a tenth data bit.

When performing 8-bit writes to the transmit FIFO, the ninth and tenth bits are pushed into the FIFO from CTRL[T8] and CTRL[T9]. For coherent 8-bit writes, you must write to CTRL[T8] and CTRL[T9] before writing to Data (DATA)[7:0]. However, if the values in CTRL[T8] or CTRL[T9] do not need to change, it is not necessary to update CTRL[T8] and CTRL[T9] before every 8-bit write to Data (DATA).

When performing 16-bit or 32-bit writes to the transmit FIFO, all 10 bits are pushed into the transmit FIFO from the write data.

When performing 8-bit reads of the receive FIFO, the ninth and tenth bits are held in CTRL[R8] and CTRL[R9] but you must read them before reading Data (DATA). A 16-bit or 32-bit read of the receive FIFO returns all 10 bits in Data (DATA).

The 9-bit Data mode is typically used with parity to allow eight bits of data plus the parity in the ninth bit, or it is used with the address mark wake-up so that the ninth data bit can serve as the wake-up bit. The 10-bit Data mode is typically used with parity and address mark wake-up so that the ninth data bit can serve as the wake-up bit and the tenth bit can serve as the parity bit. In custom protocols, the ninth and/or tenth bits can also serve as software-controlled markers.

#### 73.3.6.2 Idle length

An idle character is one where the start bit, all data bits, and stop bits are in the mark position (idle state, generally logic 1). You can configure CTRL[ILT] to start detecting an idle character from the previous start bit (any data bits and stop bits count for idle character detection) or from the previous stop bit.

You can also use CTRL[IDLECFG] to configure the number of idle characters that must be received before an idle line condition is detected. This field configures the number of idle characters that must be received before STAT[IDLE] becomes 1, STAT[RAF] becomes 0, and DATA[IDLINE] becomes 1 with the next received character.

CTRL[IDLECFG] also affects the idle line wake-up and idle match operations. When either the address match or match on/off operation is enabled, writing 1 to STAT[RWUID] causes any discarded characters to be treated as idle characters.

#### 73.3.6.3 Loop mode

Enable Loop mode by setting CTRL[LOOPS] = 1 and CTRL[RSRC] = 0. You, sometimes, use Loop mode to check software, independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is internally connected to the receiver input and LPUART does not use the RXD pin.

#### 73.3.6.4 Single-Wire mode

Enable Single-Wire mode by setting CTRL[LOOPS] = 1 and CTRL[RSRC] = 1. Single-Wire mode implements a half-duplex serial connection. The receiver is internally connected to the transmitter output and TXD pin (the RXD pin is not used).

In Single-Wire mode, CTRL[TXDIR] controls the direction of serial data on the TXD pin. When CTRL[TXDIR] becomes 0, the TXD pin is an input to the receiver and the transmitter is temporarily disconnected from the TXD pin so that an external device can send serial data to the receiver. When CTRL[TXDIR] = 1, the TXD pin is an output that the transmitter drives. The internal loop back connection is disabled, and as a result, the receiver is unable to receive characters that the transmitter sends out.

### 73.3.7 Peripheral triggers

The connection of the LPUART peripheral triggers with other peripherals is chip-specific.

#### 73.3.7.1 Output triggers

LPUART generates the following output triggers that can be connected to other peripherals on the chip:

- The transmit word trigger asserts at the end of each transmitted word and negates after 1-bit period.
- The transmit data trigger is identical to the TXD pin output, but without support for input trigger modulation.
- The receive word trigger asserts at the end of each received word that is written to the receive FIFO, for one oversampling clock period.
- The receive idle trigger asserts when STAT[IDLE] becomes 1, for one oversampling clock period.

#### 73.3.7.2 Input trigger

LPUART supports a peripheral input trigger that you can configure in one of the following ways:

- By enabling the CTS function: You can connect the input trigger instead of the CTS_B pin input. The input trigger must assert for longer than 1-bit clock period when the transmitter is idle, with data to send, to guarantee a new transmission.
- By making the input trigger modulate the transmit data output (trigger is logically ANDed with the TXD output): The input trigger is expected to be a free-running clock (carrier signal) that generates from a timer or PWM source with a frequency that is greater than the bit-clock frequency. The carrier signal must not toggle faster than the maximum supported bit time.
- By connecting the input trigger instead of the RXD pin input: The input trigger is expected to be generated from a receive data source, such as an analog comparator or external pin.

### 73.3.8 Infrared (IR) interface

LPUART provides the capability of transmitting narrow pulses to an IR LED and receiving narrow pulses, transforming them to serial bits, which are then sent to LPUART. The IrDA physical layer specification defines a half-duplex IR communication link for exchanging data. The full standard includes data rates up to 16 Mbit/s. The LPUART IrDA support is limited to SIR mode that supports data rates only between 2.4 kbit/s and 115.2 kbit/s.

LPUART has an infrared transmit encoder and a receive decoder. The infrared decoder converts the received character from the IrDA format to the NRZ format, which the receiver uses. It also has an OSR oversampling baud rate clock counter that filters noise and indicates when a 1 is received. LPUART transmits serial bits of data, which the infrared submodule encodes, to transmit a narrow pulse for every zero bit. No pulse is transmitted for every single bit. When receiving data, an IR photo diode (external to LPUART) detects the IR pulses. The IR receive decoder transforms them to CMOS levels. The infrared receive decoder then stretches the narrow pulses to get back to a serial bit stream that LPUART receives. You can invert the polarity of transmitted pulses and expected receive pulses so that a direct connection can be made to external IrDA transceiver modules that use active-high pulses.

The IR submodule receives its clock sources from LPUART. The submodule selects one of these clocks to generate either 1 ÷ OSR, 2 ÷ OSR, 3 ÷ OSR, or 4 ÷ OSR narrow pulses during transmission.

#### 73.3.8.1 Infrared transmit encoder

The infrared transmit encoder converts serial bits of data from the transmit shift register to the TXD signal. A narrow pulse is transmitted for a 0 bit and no pulse is transmitted for a 1 bit. The narrow pulse is sent at the start of the bit with a duration of 1 ÷ OSR, 2 ÷ OSR, 3 ÷ OSR, or 4 ÷ OSR of a bit time. A narrow low pulse is transmitted for a 0 bit when CTRL[TXINV] is 0, while a narrow high pulse is transmitted for a 0 bit when CTRL[TXINV] is 1.

#### 73.3.8.2 Infrared receive decoder

The infrared receive block converts data from the RXD signal to the receive shift register. A narrow pulse is expected for each 0 received and no pulse is expected for each 1 received. A narrow low pulse is expected for a 0 bit when STAT[RXINV] is 0, while a narrow high pulse is expected for a 0 bit when STAT[RXINV] is 1. This receive decoder meets the edge jitter requirement as defined by the IrDA serial infrared physical layer specification.

#### 73.3.8.3 Start-bit detection

When STAT[RXINV] is 0, the first falling edge of the received character corresponds to the start bit. The infrared decoder resets its counter. At this time, the receiver also begins its start bit detection process. After the start bit is detected, the receiver synchronizes its bit times to this start bit time. For the rest of the character reception, the infrared decoder's counter and the receiver's bit time counter count independently of each other.

#### 73.3.8.4 Noise filtering

The decoder ignores any rising edges detected during the first half of the infrared decoder counter, and can leave any pulses less than one oversampling baud clock as undetected. This is regardless of whether the pulse is seen in the first or second half of the count.

#### 73.3.8.5 Low-bit detection

During the second half of the decoder count, a rising edge is decoded as 0, which is sent to the receiver. The decoder counter is also reset.

#### 73.3.8.6 High-bit detection

At OSR oversampling baud rate clocks after the previous rising edge, if a rising edge is not seen, the decoder sends a 1 to the receiver.

If the next bit is 0, which arrives late, a low bit is detected according to Low-bit detection. The value sent to the receiver is changed from 1 to 0. Then, if a noise pulse occurs outside the receiver's bit time sampling period, the delay of a 0 is not recorded as noise.

## 73.3.9 Modes of operation

### 73.3.9.1 Low-Power modes

LPUART remains functional during low-power modes, provided CTRL[DOZEEN] is 0 and the LPUART functional clock is enabled. LPUART can generate an interrupt or DMA request to cause a wake-up from low-power modes.

You can configure LPUART to be disabled in low-power modes, when CTRL[DOZEEN] is 1. In this case, the transmitter and receiver finish transmitting and receiving the current word.

If LPUART is disabled in low-power modes, it can generate a wake-up via STAT[RXEDGIF] if the receiver detects an active edge.

**NOTE**

See the chip-specific information for specific low-power modes available on your chip.

### 73.3.9.2 Debug mode

LPUART remains functional in Debug mode.

## 73.3.10 Clocking

**Table 547. Types of clocks**

| Clock | Description |
|---|---|
| Functional | Is asynchronous to the bus clock and can remain enabled in Low-Power modes to support transmit and/or receive functions, including low-power wake-up. |
| Bus | Is only used for bus accesses to the control and configuration registers. The bus clock frequency must be sufficient to support the data bandwidth requirements of the LPUART transmit and receive registers, including the FIFOs. |

## 73.3.11 Reset

**Table 548. Types of resets**

| Reset | Description |
|---|---|
| Chip | Enables the logic and registers for the LPUART transmitter and receiver to reset to their default states. |
| Software | Resets the LPUART logic and registers to their default states, except for Global (GLOBAL).<br>GLOBAL[RST] controls the LPUART software reset. |
| FIFO | Implements write-only control fields that reset the transmit FIFO (FIFO[TXFLUSH]) and receive FIFO (FIFO[RXFLUSH]). After a FIFO is reset, that FIFO becomes empty. |

## 73.3.12 Interrupts

The LPUART transmitter has two status fields that can optionally generate hardware interrupt requests. If STAT[TDRE] is 1, it indicates that there is room in the transmit FIFO to write another transmit character to Data (DATA). If CTRL[TIE] is 1, a hardware interrupt is requested when STAT[TDRE] is 1.

STAT[TC] indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with TXD at the inactive level. This field is often used in systems with modems to determine when it is safe to turn off the modem. If CTRL[TCIE] is 1, a hardware interrupt is requested when STAT[TC] is 1. Instead of hardware interrupts, software polling may be used to monitor STAT[TDRE] and STAT[TC] if the corresponding CTRL[TIE] or CTRL[TCIE] field is 0.

When a program detects that STAT[RDRF] is 1, it gets the data from this field by reading Data (DATA). The field becomes 0 by reading Data (DATA).

STAT[IDLE] includes logic that prevents it from becoming 1 repeatedly when the RXD line remains idle for an extended period of time. STAT[IDLE] becomes 0 when you write 1 to it, and cannot become 1 again until the receiver has received at least one new character and has 1 as the value of STAT[RDRF].

If the associated error is detected in the received character that caused STAT[RDRF] to become 1, STAT[NF], STAT[FE], and STAT[PF] become 1 at the same time STAT[RDRF] becomes 1. These flags do not become 1 in overrun cases.

If STAT[RDRF] is already 1 when a new character is ready to be transferred from the receive shifter to the receive FIFO, STAT[OR] becomes 1, instead of the data along with any associated STAT[NF], STAT[FE], or STAT[PF] condition getting lost.

If the received character matches the contents of MATCH[MA1] and/or MATCH[MA2], then STAT[MA1F] and/or STAT[MA2F] become 1 at the same time that STAT[RDRF] becomes 1.

At any time, an active edge on the RXD serial data input pin causes STAT[RXEDGIF] to become 1. STAT[RXEDGIF] becomes 0 when you write 1 to it. This function depends on the receiver being enabled (the value of CTRL[RE] being 1).

## 73.4 External signals

**Table 549. External signals**

| Signal | Description | I/O |
|---|---|---|
| TXD | Transmit data: This pin is normally an output, but is an input (tristated) in Single-Wire mode whenever the transmitter is disabled or the transmit direction is configured for receive data. | I/O |
| RXD | Receive data | I |
| CTS_B | Clear-to-send | I |
| RTS_B | Request-to-send | O |

## 73.5 Initialization

This module does not require initialization.

## 73.6 LPUART register descriptions

LPUART includes registers to control baud rate, select options, report status, and store transmit and receive data. Access to an address outside the valid memory map generates a bus error.

**NOTE**

Writing to a read-only (RO) register or reading a write-only (WO) register can cause bus errors. LPUART does not verify whether programmed values in the registers are correct; you must write valid values to them.

### 73.6.1 LPUART memory map

LPUART_0 base address: 4032_8000h

LPUART_1 base address: 4032_C000h

LPUART_2 base address: 4033_0000h

LPUART_3 base address: 4033_4000h

LPUART_4 base address: 4033_8000h

LPUART_5 base address: 4033_C000h

LPUART_6 base address: 4034_0000h

LPUART_7 base address: 4034_4000h

LPUART_8 base address: 4048_C000h

LPUART_9 base address: 4049_0000h

LPUART_10 base address: 4049_4000h

LPUART_11 base address: 4049_8000h

LPUART_12 base address: 4049_C000h

LPUART_13 base address: 404A_0000h

LPUART_14 base address: 404A_4000h

LPUART_15 base address: 404A_8000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Version ID (VERID) | 32 | R | 0403_0003h |
| 4h | Parameter (PARAM) | 32 | R | 0000_0202h |
| 8h | Global (GLOBAL) | 32 | RW | 0000_0000h |
| Ch | Pin Configuration (PINCFG) | 32 | RW | 0000_0000h |
| 10h | Baud Rate (BAUD) | 32 | RW | 0F00_0004h |
| 14h | Status (STAT) | 32 | RW | 00C0_0000h |
| 18h | Control (CTRL) | 32 | RW | 0000_0000h |
| 1Ch | Data (DATA) | 32 | RW | 0000_1000h |
| 20h | Match Address (MATCH) | 32 | RW | 0000_0000h |
| 24h | MODEM IrDA (MODIR) | 32 | RW | 0000_0000h |
| 28h | FIFO (FIFO) | 32 | RW | 00C0_0011h |
| 2Ch | Watermark (WATER) | 32 | RW | 0000_0000h |
| 30h | Data Read-Only (DATARO) | 32 | R | 0000_1000h |

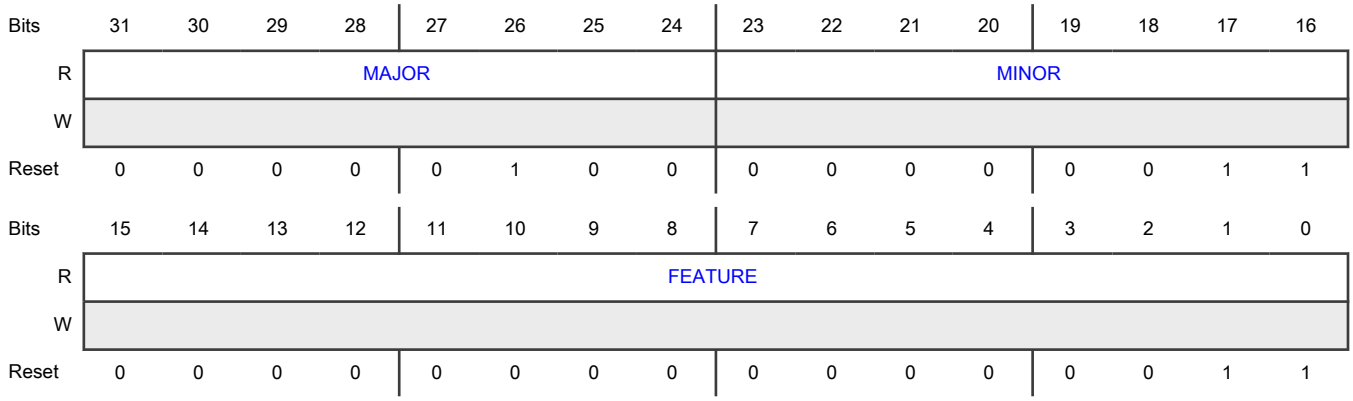
### 73.6.2 Version ID (VERID)

**Offset**

| Register | Offset |
|---|---|
| VERID | 0h |

**Function**
Indicates the version integrated for this instance on the chip and also specifies the inclusion and exclusion of several optional features.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-24<br>MAJOR | Major Version Number<br>Indicates the major version number for the module specification. |
| 23-16<br>MINOR | Minor Version Number<br>Indicates the minor version number for the module specification. |
| 15-0<br>FEATURE | Feature Identification Number<br>Indicates the feature set number.<br>0000_0000_0000_0001b - Standard feature set<br>0000_0000_0000_0011b - Standard feature set with MODEM and IrDA support |

## 73.6.3 Parameter (PARAM)

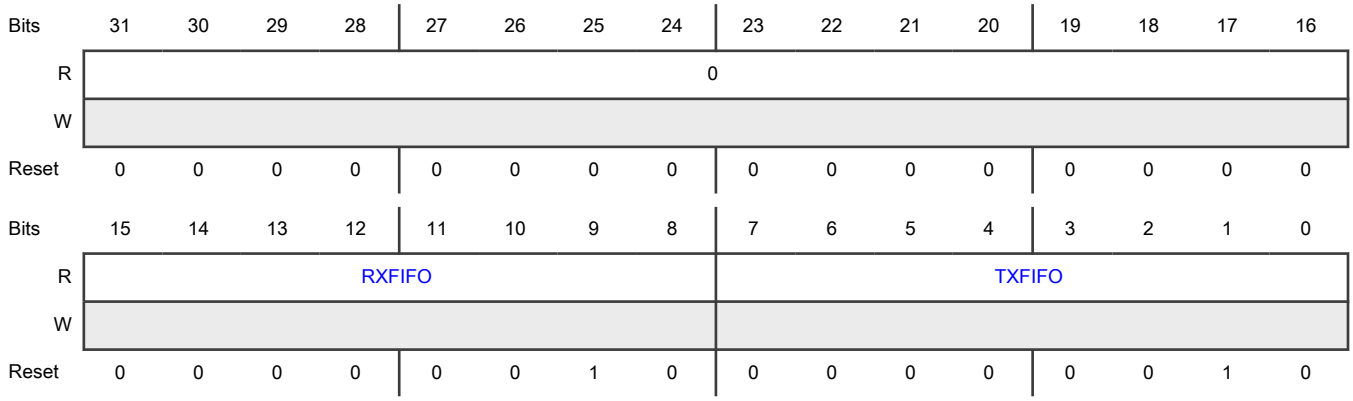
### Offset

| Register | Offset |
|---|---|
| PARAM | 4h |

### Function

Indicates the parameter configuration for this instance on the chip.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-8<br>RXFIFO | Receive FIFO Size<br>Indicates the number of characters in the receive FIFO, which is 2^RXFIFO. |
| 7-0<br>TXFIFO | Transmit FIFO Size<br>Indicates the number of characters in the transmit FIFO, which is 2^TXFIFO. |

## 73.6.4 Global (GLOBAL)

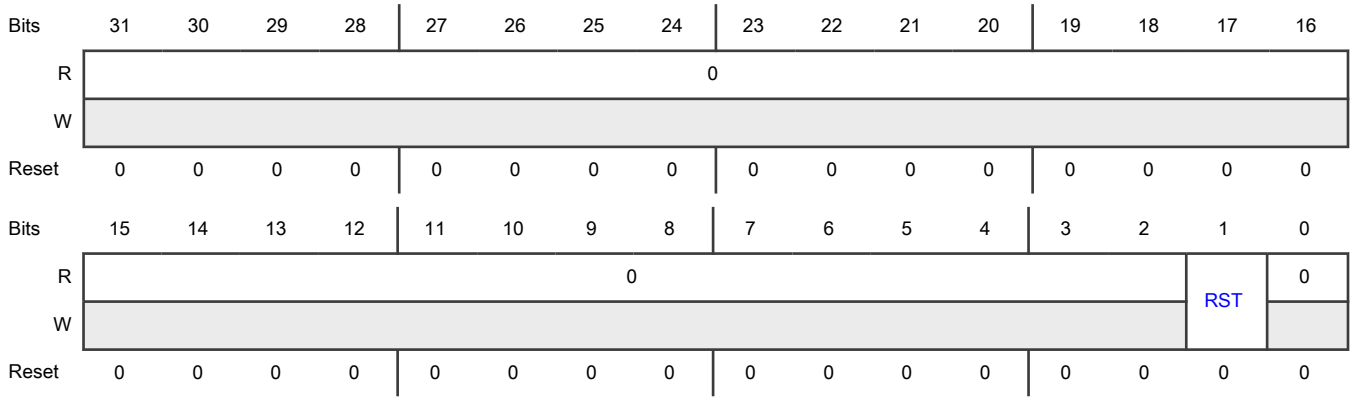
### Offset

| Register | Offset |
|---|---|
| GLOBAL | 8h |

### Function

Performs global functions.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>RST | Software Reset<br>Specifies whether the module is reset.<br>This field resets all internal logic and registers, except Global (GLOBAL). The reset takes effect immediately and remains asserted until you negate it. There is no minimum delay required before clearing the software reset.<br>0b - Not reset<br>1b - Reset |
| 0<br>— | Reserved |

## 73.6.5 Pin Configuration (PINCFG)

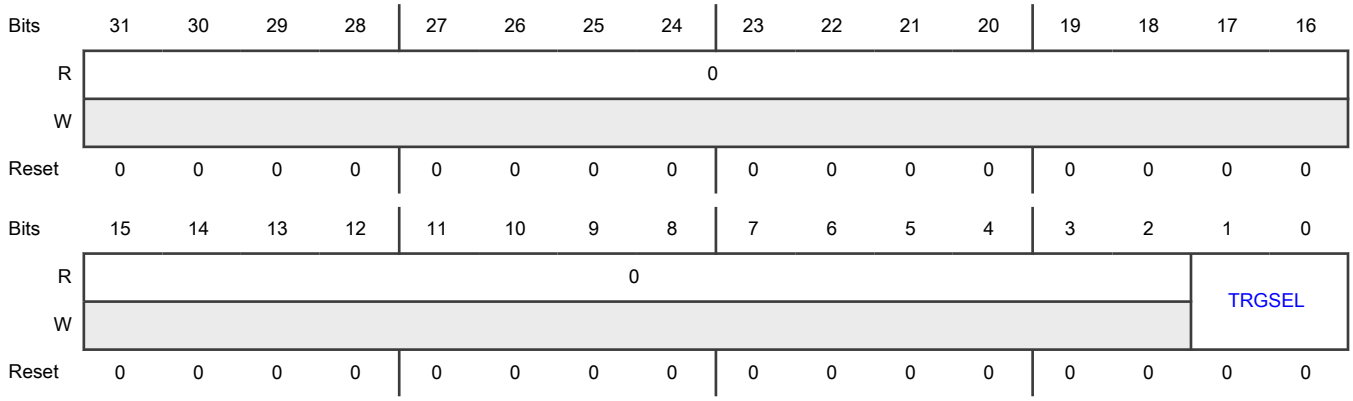
### Offset

| Register | Offset |
|---|---|
| PINCFG | Ch |

### Function

Enables the selection of input pins.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1-0<br>TRGSEL | Trigger Select<br>Configures the input trigger usage.<br>You must change the value of this field only when both the transmitter and receiver are disabled.<br>00b - Input trigger disabled<br>01b - Input trigger used instead of the RXD pin input<br>10b - Input trigger used instead of the CTS_B pin input<br>11b - Input trigger used to modulate the TXD pin output, which (after TXINV configuration) is internally ANDed with the input trigger |

## 73.6.6 Baud Rate (BAUD)

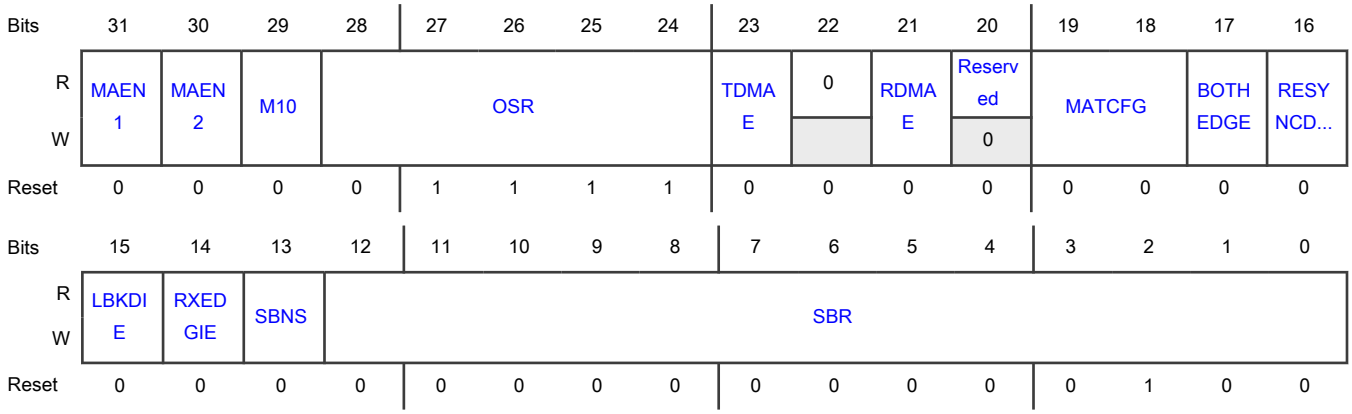
### Offset

| Register | Offset |
|---|---|
| BAUD | 10h |

### Function

Configures the baud rate.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>MAEN1 | Match Address Mode Enable 1<br>Enables automatic address matching or data matching mode for MATCH[MA1]. If this field is 0, normal operation takes place.<br>0b - Disable<br>1b - Enable |
| 30<br>MAEN2 | Match Address Mode Enable 2<br>Enables automatic address matching or data matching mode for MATCH[MA2]. If this field is 0, normal operation takes place.<br>0b - Disable<br>1b - Enable |
| 29<br>M10 | 10-Bit Mode Select<br>Causes the tenth bit to be a part of the serial transmission.<br>You must change the value of this field only when both the transmitter and receiver are disabled.<br>0b - Receiver and transmitter use 7-bit to 9-bit data characters<br>1b - Receiver and transmitter use 10-bit data characters |
| 28-24<br>OSR | Oversampling Ratio<br>Configures the OSR of the receiver.<br>You must change the value of this field only when both the transmitter and receiver are disabled.<br>**NOTE**<br>BAUD[OSR] results in an OSR of BAUD[OSR] + 1, for example, BAUD[OSR] = 0_0101b results in a final division by 6.<br>0_0000b - Results in an OSR of 16 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0_0001b - Reserved<br>0_0010b - Reserved<br>0_0011b - Results in an OSR of 4 (requires BAUD[BOTHEDGE] to be 1)<br>0_0100b - Results in an OSR of 5 (requires BAUD[BOTHEDGE] to be 1)<br>0_0101b - Results in an OSR of 6 (requires BAUD[BOTHEDGE] to be 1)<br>0_0110b - Results in an OSR of 7 (requires BAUD[BOTHEDGE] to be 1)<br>0_0111b - Results in an OSR of 8<br>0_1000b - Results in an OSR of 9<br>0_1001b - Results in an OSR of 10<br>0_1010b - Results in an OSR of 11<br>0_1011b - Results in an OSR of 12<br>0_1100b - Results in an OSR of 13<br>0_1101b - Results in an OSR of 14<br>0_1110b - Results in an OSR of 15<br>0_1111b - Results in an OSR of 16<br>1_0000b - Results in an OSR of 17<br>1_0001b - Results in an OSR of 18<br>1_0010b - Results in an OSR of 19<br>1_0011b - Results in an OSR of 20<br>1_0100b - Results in an OSR of 21<br>1_0101b - Results in an OSR of 22<br>1_0110b - Results in an OSR of 23<br>1_0111b - Results in an OSR of 24<br>1_1000b - Results in an OSR of 25<br>1_1001b - Results in an OSR of 26<br>1_1010b - Results in an OSR of 27<br>1_1011b - Results in an OSR of 28<br>1_1100b - Results in an OSR of 29<br>1_1101b - Results in an OSR of 30<br>1_1110b - Results in an OSR of 31<br>1_1111b - Results in an OSR of 32 |
| 23<br>TDMAE | Transmitter DMA Enable<br>Enables STAT[TDRE] to generate a DMA request. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disable<br>1b - Enable |
| 22<br>— | Reserved |
| 21<br>RDMAE | Receiver Full DMA Enable<br>Enables STAT[RDRF] to generate a DMA request.<br>0b - Disable<br>1b - Enable |
| 20<br>— | Reserved |
| 19-18<br>MATCFG | Match Configuration<br>Configures the match addressing mode used.<br>You must change the value of this field only when both the transmitter and receiver are disabled.<br>00b - Address match wake-up<br>01b - Idle match wake-up<br>10b - Match on and match off<br>11b - Enables RWU on data match and match on or off for the transmitter CTS input |
| 17<br>BOTHEDGE | Both Edge Sampling<br>Enables sampling of the received data on both edges of the baud rate clock, effectively doubling the number of times the receiver samples the input data for a given OSR.<br>This field must be 1 for OSRs between x4 and x7 and is optional for higher OSRs. You must change the value of this field only when the receiver is disabled.<br>If this field is 0, the receiver samples input data using the rising edge of the baud rate clock. If this field is 1, the receiver samples input data using the rising and falling edges of the baud rate clock.<br>0b - Rising edge<br>1b - Both rising and falling edges |
| 16<br>RESYNCDIS | Resynchronization Disable<br>Disables resynchronization of the received data word when a data one followed by data zero transition is detected.<br>You must change the value of this field only when the receiver is disabled.<br>0b - Enable<br>1b - Disable |
| 15 | LIN Break Detect Interrupt Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| LBKDIE | Enables STAT[LBKDIF] to generate hardware interrupt requests.<br>If this field is 0, hardware interrupts from STAT[LBKDIF] (uses polling) are disabled. If this field is 1, hardware interrupts are requested when STAT[LBKDIF] is 1.<br>0b - Disable<br>1b - Enable |
| 14<br>RXEDGIE | RX Input Active Edge Interrupt Enable<br>Enables STAT[RXEDGIF] to generate interrupt requests. If this field is 0, hardware interrupts from STAT[RXEDGIF] are disabled. If this field is 1, hardware interrupts are requested when STAT[RXEDGIF] is 1.<br>Changing the value of CTRL[LOOPS] or CTRL[RSRC] when this field (RXEDGIE) is 1 can cause STAT[RXEDGIF] to become 1.<br>0b - Disable<br>1b - Enable |
| 13<br>SBNS | Stop Bit Number Select<br>Determines whether data characters include one or two stop bits.<br>You must change the value of this field only when both the transmitter and receiver are disabled.<br>0b - One stop bit<br>1b - Two stop bits |
| 12-0<br>SBR | Baud Rate Modulo Divisor<br>Sets the modulo divide rate for the baud rate generator.<br>• If SBR is 0, baud rate generator is disabled.<br>• If SBR is 1–8191, baud rate = baud clock ÷ ((OSR + 1) × SBR). You must update the 13-bit baud rate setting [SBR12:SBR0] only when both the transmitter and receiver are disabled (both CTRL[RE] and CTRL[TE] are 0). |

## 73.6.7 Status (STAT)

**Offset**

| Register | Offset |
|---|---|
| STAT | 14h |

**Function**
Provides the module status.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | LBKDIF | RXEDGIF | MSBF | RXINV | RWUID | BRK13 | LBKDE | RAF | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF |
| W | W1C | W1C | | | | | | | | | | W1C | W1C | W1C | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | MA1F | MA2F | 0 | | | | 0 | | 0 | | | | | | AME | LBKFE |
| W | W1C | W1C | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>LBKDIF | LIN Break Detect Interrupt Flag<br>Indicates whether a LIN break character is detected.<br>This field becomes 1 when the LIN break detect circuitry is enabled and a LIN break character is detected.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not detected<br>1b - Detected<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 30<br>RXEDGIF | RXD Pin Active Edge Interrupt Flag<br>Indicates whether an active edge on the receive pin has occurred.<br>This field becomes 1 whenever the receiver is enabled and an active edge (falling if STAT[RXINV] is 0; rising if STAT[RXINV] is 1) on the RXD pin occurs.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not occurred<br>1b - Occurred<br>When writing<br>0b - No effect |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - Clear the flag |
| 29<br>MSBF | MSB First<br>Specifies the first bit that is transmitted after the start bit.<br>If this field is 0, LSB (bit 0) is the first bit transmitted after the start bit (which means, the first bit received after the start bit is identified as bit 0).<br>If this field is 1, MSB (identified as bit 9, bit 8, bit 7, or bit 6) is the first bit that is transmitted, after the start bit, depending on the settings of CTRL[M], CTRL[PE], and BAUD[M10].<br>Writing 1 to this field reverses the order of the bits that are transmitted and received on the wire. This field does not affect the polarity of the bits, the location of the parity bit, or the location of the start or stop bits. You must change the value of this field only when both the transmitter and receiver are disabled.<br>0b - LSB<br>1b - MSB |
| 28<br>RXINV | Receive Data Inversion<br>Specifies whether receive data is inverted.<br>Writing 1 to this field reverses the polarity of the received data input. You must change the value of this field only when the receiver is disabled.<br>**NOTE**<br>Writing 1 to this field inverts the RXD input for all cases: data bits, start and stop bits, break, and idle.<br>0b - Inverted<br>1b - Not inverted |
| 27<br>RWUID | Receive Wake Up Idle Detect<br>Controls, for CTRL[RWU] on idle character detection, whether the idle character that wakes up the receiver writes 1 to STAT[IDLE].<br>For address match wake-up, this field controls whether STAT[IDLE] = 1 when the address does not match. You must change the value of this field only when the receiver is disabled.<br>If this field is 0, during the Receive Standby state (CTRL[RWU] = 1), STAT[IDLE] does not become 1 upon detection of an idle character. During address match wake-up, STAT[IDLE] does not become 1 when an address does not match.<br>If this field is 1, during the Receive Standby state (CTRL[RWU] = 1), STAT[IDLE] becomes 1 upon detection of an idle character. During address match wake-up, STAT[IDLE] becomes 1 when an address does not match.<br>0b - STAT[IDLE] does not become 1<br>1b - STAT[IDLE] becomes 1 |
| 26<br>BRK13 | Break Character Generation Length<br>Selects the longer transmitted break character length. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | The state of this field does not affect the detection of a framing error. You must change the value of this field only when the transmitter is disabled. You can send a break character by writing 1 to CTRL[SBK], or by writing the transmit FIFO when DATA[FRETSC] is 1 and DATA[R9T9] is 0.<br>0b - 9 to 13 bit times<br>1b - 12 to 15 bit times |
| 25<br>LBKDE | LIN Break Detection Enable<br>Enables LIN break detection.<br>If this field is 0, LIN break detect is disabled, and only a normal break character can be detected.<br>If this field is 1, LIN break detect is enabled and the LIN break character is detected at a length of 11 bit times (if CTRL[M] is 0), 12 bit times (if CTRL[M] is 1), or 13 bit times (if BAUD[M10] is 1).<br>This field selects a longer break character detection length. When the field is 1, receive data is not stored in the receive FIFO.<br>**NOTE**<br>This field enables the LIN break detect circuit and disables writing receive data to FIFO. Therefore, it ignores all characters except a LIN break.<br>0b - Disable<br>1b - Enable |
| 24<br>RAF | Receiver Active Flag<br>Indicates whether the LPUART receiver is idle or active.<br>This field becomes 1 when the receiver detects the beginning of a valid start bit, and the field becomes 0 automatically when the receiver detects an idle line.<br>0b - Idle, waiting for a start bit<br>1b - Receiver active (RXD pin input not idle) |
| 23<br>TDRE | Transmit Data Register Empty Flag<br>Indicates whether the transmit FIFO level is greater than, equal to, or less than the watermark.<br>After the transmit FIFO is enabled, this field becomes 1 when the number of datawords in the transmit FIFO is equal to, or less than the number that WATER[TXWATER] indicates. To make the value of this field 0, write to it until the number of words in the transmit FIFO is greater than the number that WATER[TXWATER] indicates. After the transmit FIFO is disabled, this field becomes 1 to indicate that the FIFO level is less than the watermark. To make the value of this field 0 again, write to Data (DATA).<br>This register is not affected by a character that is in the process of being transmitted; it is updated at the start of each transmitted character.<br>0b - Greater than watermark<br>1b - Equal to or less than watermark |
| 22 | Transmission Complete Flag<br>Indicates whether the transmitter is active. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| TC | This field becomes 0 when a transmission is in progress or a preamble or break character is loaded; in other words, when the transmitter is active (sending data, a preamble, or a break). The field becomes 1 when the transmit buffer is empty and no data, preamble, or break character is being transmitted; in other words, when the transmission activity is complete. When this happens, the transmit data output signal becomes idle (logic 1). This field becomes 0 after you write to Data (DATA) to transmit new data, queuing a preamble by first writing 0 and then writing 1 to CTRL[TE], queuing a break character by writing 1 to CTRL[SBK].<br>0b - Transmitter active<br>1b - Transmitter idle |
| 21<br>RDRF | Receive Data Register Full Flag<br>Indicates whether the receive FIFO level is less than, equal to, or greater than the watermark.<br>This field becomes 1 when the number of datawords in the receive buffer is greater than the number that WATER[RXWATER] indicates and the receive FIFO is enabled. To write 0 to this field, read Data (DATA) until the number of datawords in the receive FIFO is equal to, or less than the number that WATER[RXWATER] indicates. When the receive FIFO is disabled, this field (RDRF) becomes 1 if the receive buffer (Data (DATA)) is full. To make this field 0, read Data (DATA).<br>A character that is in the process of being received does not cause a change in this field until the entire character is received. Even if this field is 1, the character continues to be received until an overrun condition occurs after the entire character is received.<br>0b - Equal to or less than watermark<br>1b - Greater than watermark |
| 20<br>IDLE | Idle Line Flag<br>Indicates whether an idle line is detected.<br>This field becomes 1 when the LPUART receive line becomes idle for a full character time after a period of activity. When CTRL[ILT] is 0, the receiver starts counting idle bit times after the start bit. If the receive character is all 1s, these bit times and the stop bit time count towards the full character time of logic high, 10 to 13 bit times, needed for the receiver to detect an idle line. After CTRL[ILT] becomes 1, the receiver does not start counting idle bit times until after the stop bits. The stop bits and any logic high bit times at the end of the previous character do not count towards the full character time of logic high needed for the receiver to detect an idle line.<br>For this field to become 0, write 1 to it. After the field becomes 0, you cannot write 1 to it again until after a new character is stored in the receive buffer or a LIN break character writes 1 to STAT[LBKDIF]. This field becomes 1 only once, even if the receive line remains idle for an extended period.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Idle line detected<br>1b - Idle line not detected<br>When writing |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No effect<br>1b - Clear the flag |
| 19<br>OR | Receiver Overrun Flag<br>Indicates whether there is receive overrun.<br>This field becomes 1 when you cannot prevent STAT[RDRF] from overflowing with data. The field becomes 1 immediately after the stop bit is completely received for the dataword that overflows the buffer and all the other error fields (STAT[FE], STAT[NF], and STAT[PF]) are prevented from becoming 1. The data in the shift register is lost, but the data already in the LPUART data registers is not affected. If STAT[LBKDE] is enabled and a LIN break is detected, this field becomes 1 if STAT[LBKDIF] is not 0 before the next data character is received.<br>When this field is 1, no additional data is stored in the receive FIFO even if sufficient room exists.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No overrun<br>1b - Receive overrun (new LPUART data is lost)<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 18<br>NF | Noise Flag<br>Indicates whether noise is detected in the received character of Data (DATA).<br>The advanced sampling technique used in the receiver takes three samples in each of the received bits. If some of these samples disagree with the rest of the samples within any bit time in the frame, then noise is detected for that character. This field becomes 1 whenever the next character to be read from Data (DATA) is received with noise detected within the character.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No noise detected<br>1b - Noise detected<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 17<br>FE | Framing Error Flag<br>Indicates whether a framing error is detected. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | This field becomes 1 whenever the next character to be read from Data (DATA) is received with logic 0 detected where a stop bit was expected.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No framing error detected (this does not guarantee that the framing is correct)<br>1b - Framing error detected<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 16<br>PF | Parity Error Flag<br>Indicates whether a parity error is detected.<br>This field becomes 1 whenever the next character to be read from Data (DATA) is received when parity is enabled (CTRL[PE] is 1) and the parity bit in the received character does not agree with the expected parity value.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No parity error detected<br>1b - Parity error detected<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 15<br>MA1F | Match 1 Flag<br>Indicates whether the received data is equal to MATCH[MA1].<br>This field becomes 1 whenever the next character to be read from Data (DATA) matches the value of MATCH[MA1].<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not equal to MA1<br>1b - Equal to MA1<br>When writing |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No effect<br>1b - Clear the flag |
| 14<br>MA2F | Match 2 Flag<br>Indicates whether the received data is equal to MATCH[MA2].<br>This field becomes 1 whenever the next character to be read from Data (DATA) matches the value of MATCH[MA2].<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - Not equal to MA2<br>1b - Equal to MA2<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 13-10<br>— | Reserved |
| 9-8<br>— | Reserved |
| 7-2<br>— | Reserved |
| 1<br>AME | Address Mark Enable<br>Configures the location of the address mark when configured for MSB first transfers.<br>This field has no effect when configured for LSB first and you must change the value of this field only when both the transmitter and receiver are disabled. If this field is 0, address mark in character is MSB. If this field is 1, the address mark is stored in Data (DATA) at MSB (or MSB-1 when the parity bit is enabled). In other words, the address mark in character is the last bit before the stop bit (or parity bit when enabled).<br>0b - Disable<br>1b - Enable |
| 0<br>LBKFE | LIN Break Flag Enable<br>Enables the LIN break flag to assert whenever a LIN break character is detected.<br>Unlike STAT[LBKDE], this does not impact data being stored in the receive data buffer, but does cause STAT[LBKDIF] to become 1 whenever a LIN break is detected. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Because a LIN break is longer than a normal character, the LIN break triggers a write to STAT[RDRF] with the data fields as 0 and STAT[FE] as 1. The character following the LIN break has DATA[LINBRK] as 1 to indicate that the previous character was a LIN break.<br>You must change the value of this field only when both the transmitter and receiver are disabled.<br>If this field is 1, the LIN break character is detected at a length of 11-bit times (if CTRL[M] is 0), 12 (if CTRL[M] is 1), or 13 (if BAUD[M10] is 1).<br>0b - Disable<br>1b - Enable |

## 73.6.8 Control (CTRL)

### Offset

| Register | Offset |
|---|---|
| CTRL | 18h |

### Function

Controls various optional features of the LPUART system.

You must write to the fields of this register only when both the transmitter and receiver are disabled.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | R8T9 | R9T8 | TXDIR | TXINV | ORIE | NEIE | FEIE | PEIE | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | MA1IE | MA2IE | 0 | 0 | M7 | IDLECFG | | | LOOPS | DOZEEN | RSRC | M | WAKE | ILT | PE | PT |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>R8T9 | Receive Bit 8 Transmit Bit 9<br>Contains R8 and T9 that correspond to different functions. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | R8 is the ninth data bit received after you configure LPUART for 9-bit or 10-bit data formats. When reading 9-bit or 10-bit data, read R8 before reading Data (DATA).<br>T9 is the tenth data bit transmitted after you configure LPUART for 10-bit data formats. When writing 10-bit data, write T9 before writing to Data (DATA). If T9 does not need to change from its previous value, such as when it is used to generate address mark or parity, then you need not write to it each time you write to Data (DATA).<br>**NOTE**<br>R8 is a read-only bit and T9 is a write-only bit; the value read is different from the value written. |
| 30<br>R9T8 | Receive Bit 9 Transmit Bit 8<br>Contains R9 and T8 that correspond to different functions.<br>R9 is the tenth data bit received after you configure LPUART for 10-bit data formats. When reading 10-bit data, read R9 before reading Data (DATA).<br>T8 is the ninth data bit transmitted after you configure LPUART for 9-bit or 10-bit data formats. When writing 9-bit or 10-bit data, write T8 before writing to Data (DATA). If T8 does not need to change from its previous value, such as when it is used to generate address mark or parity, then you need not write to it each time you write to Data (DATA).<br>**NOTE**<br>R9 is a read-only field and T8 is a write-only field; the value read is different from the value written. |
| 29<br>TXDIR | TXD Pin Direction in Single-Wire Mode<br>Determines the direction of data at the TXD pin, in Single-Wire mode, when LPUART is configured for a single-wire half-duplex operation (CTRL[LOOPS] and CTRL[RSRC] are 1). When writing 0 to this field, the transmitter finishes transmitting the current character (if any) before the receiver starts receiving data from the TXD pin.<br>0b - Input<br>1b - Output |
| 28<br>TXINV | Transmit Data Inversion<br>Specifies whether transmit data is inverted.<br>Writing 1 to this field reverses the polarity of the transmitted data output. This action inverts the TXD output for all cases: data bits, start and stop bits, break, and idle.<br>0b - Not inverted<br>1b - Inverted |
| 27<br>ORIE | Overrun Interrupt Enable<br>Enables STAT[OR] to generate hardware interrupt requests. When this field is 1, a hardware interrupt is requested. Use polling when OR interrupts are disabled. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Disable<br>1b - Enable |
| 26<br>NEIE | Noise Error Interrupt Enable<br>Enables STAT[NF] to generate hardware interrupt requests. When this field is 1, a hardware interrupt is requested. Use polling when NF interrupts are disabled.<br>0b - Disable<br>1b - Enable |
| 25<br>FEIE | Framing Error Interrupt Enable<br>Enables STAT[FE] to generate hardware interrupt requests. When this field is 1, a hardware interrupt is requested. Use polling when FE interrupts are disabled.<br>0b - Disable<br>1b - Enable |
| 24<br>PEIE | Parity Error Interrupt Enable<br>Enables STAT[PF] to generate hardware interrupt requests. When this field is 1, a hardware interrupt is requested. Use polling when PF interrupts are disabled.<br>0b - Disable<br>1b - Enable |
| 23<br>TIE | Transmit Interrupt Enable<br>Enables STAT[TDRE] to generate interrupt requests if STAT[TDRE] is 1.<br>0b - Disable<br>1b - Enable |
| 22<br>TCIE | Transmission Complete Interrupt Enable<br>Enables STAT[TC] to generate interrupt requests if STAT[TC] is 1.<br>0b - Disable<br>1b - Enable |
| 21<br>RIE | Receiver Interrupt Enable<br>Enables STAT[RDRF] to generate hardware interrupt requests if STAT[RDRF] is 1.<br>0b - Disable<br>1b - Enable |
| 20<br>ILIE | Idle Line Interrupt Enable<br>Enables hardware interrupts.<br>This field enables STAT[IDLE] to generate interrupt requests. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | If this field is 0, hardware interrupts from STAT[IDLE] are disabled and polling is used, and if this field is 1, hardware interrupts are enabled when STAT[IDLE] is 1.<br>0b - Disable<br>1b - Enable |
| 19<br>TE | Transmitter Enable<br>Enables the LPUART transmitter.<br>Using this field, you can also queue an idle preamble by first writing 0 and then writing 1 to this field. After this field becomes 0, the field reads 1 until the transmitter has completed the current character and the TXD pin is tristated.<br>You can also queue a single idle character by writing to the transmit FIFO with DATA[FRETSC] and DATA[R9T9] = 1.<br>0b - Disable<br>1b - Enable |
| 18<br>RE | Receiver Enable<br>Enables the LPUART receiver.<br>After you write 0 to this field, this field remains 1 until the receiver finishes receiving the current character (if any).<br>0b - Disable<br>1b - Enable |
| 17<br>RWU | Receiver Wake-Up Control<br>Specifies whether the LPUART receiver in standby is waiting for a wake-up condition.<br>You can write 1 to this field to place the LPUART receiver in a Standby state. The field becomes 0 automatically when an RWU event occurs, that is, in case of an idle event when CTRL[WAKE] is 0 or an address match when CTRL[WAKE] is 1 and STAT[RWUID] is 0.<br>**NOTE**<br>You must write 1 to this field only when CTRL[WAKE] is 0 (wake-up on idle), if the channel is currently not idle. You can determine this by the value of STAT[RAF]. If the field is 1 to wake up an idle event and the channel is already idle, LPUART, possibly, discards the data. This is because the data must be received or a LIN break is detected after an Idle condition is detected before the IDLE flag is allowed to be reasserted.<br>0b - Normal receiver operation<br>1b - LPUART receiver in standby, waiting for a wake-up condition |
| 16<br>SBK | Send Break<br>Specifies whether queue break character(s) are to be sent.<br>Writing 1 and then 0 to this field queues a break character in the transmit data stream. Additional break characters of 9 to 13 bits, or 12 to 15 bits if STAT[BRK13] is 1, and bit times of logic 0 are queued as long |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | as this field is 1. Depending on the timing when this field is 1 and 0, relative to the character currently being transmitted, a second break character may be queued before you write 0 to this field. If the time taken to write 0 to this field is too long, for example, if the field does not become 0 by the end of the first break character, a second break character is sent. This is compared to queuing a break character through the transmit FIFO that guarantees only one break character is sent.<br>You can also queue a single break character by writing to the transmit FIFO when DATA[FRETSC] is 1 and DATA[R9T9] is 0.<br>0b - Normal transmitter operation<br>1b - Queue break character(s) to be sent |
| 15<br>MA1IE | Match 1 (MA1F) Interrupt Enable<br>Enables the MA1F interrupt.<br>0b - Disable<br>1b - Enable |
| 14<br>MA2IE | Match 2 (MA2F) Interrupt Enable<br>Enables the MA2F interrupt.<br>0b - Disable<br>1b - Enable |
| 13<br>— | Reserved |
| 12<br>— | Reserved |
| 11<br>M7 | 7-Bit Mode Select<br>Specifies the data characters that the receiver and transmitter use.<br>You must change the value of this field only after both the transmitter and receiver are disabled.<br>0b - 8-bit to 10-bit<br>1b - 7-bit |
| 10-8<br>IDLECFG | Idle Configuration<br>Configures the number of idle characters that must be received before you write 1 to STAT[IDLE].<br>000b - 1<br>001b - 2<br>010b - 4<br>011b - 8<br>100b - 16 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 101b - 32<br>110b - 64<br>111b - 128 |
| 7<br>LOOPS | Loop Mode Select<br>Selects Loop mode.<br>After this field becomes 1, the RXD pin is disconnected from LPUART and the transmitter output is internally connected to the receiver input. The transmitter and receiver must be enabled to use the loop function. In Loop mode or Single-Wire mode, the transmitter outputs are internally connected to the receiver input (see CTRL[RSRC]).<br>0b - Normal operation: RXD and TXD use separate pins<br>1b - Loop mode or Single-Wire mode |
| 6<br>DOZEEN | Doze Mode<br>Enables LPUART in Doze mode.<br>If this field is 1, LPUART remains active when not in Doze mode.<br>0b - Enable<br>1b - Disable |
| 5<br>RSRC | Receiver Source Select<br>Determines the source of the receiver shift register input if CTRL[LOOPS] is 1.<br>This field has no effect unless CTRL[LOOPS] is 1.<br>If this field is 0, internal Loopback mode is selected. LPUART does not use the RXD pin.<br>If this field is 1, single-wire LPUART mode is selected where the TXD pin is connected to the transmitter output and receiver input.<br>0b - Internal Loopback mode<br>1b - Single-wire mode |
| 4<br>M | 9-Bit Or 8-Bit Mode Select<br>Specifies the data characters that the receiver and transmitter use.<br>0b - 8-bit<br>1b - 9-bit |
| 3<br>WAKE | Receiver Wake-Up Method Select<br>Determines which condition wakes up LPUART when CTRL[RWU] = 1 and BAUD[MATCFG] = 0 (this field must be 1 when BAUD[MATCFG] = 11):<br>• Address mark in the bit preceding the stop bit (or bit preceding the parity bit when parity is enabled) of the received data character |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • An idle condition on the receive pin input signal<br>If this field is 0, CTRL[RWU] is configured for idle line wake-up, and if this field is 1, CTRL[RWU] is configured with address mark wake-up.<br>0b - Idle<br>1b - Mark |
| 2<br>ILT | Idle Line Type Select<br>Determines when the receiver starts counting logic 1s as idle character bits.<br>The count begins either after a valid start bit or the stop bit. If the count begins after the start bit, a string of logic 1s preceding the stop bit can cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions.<br>**NOTE**<br>In case you write 1 to this field, a logic 0 is automatically shifted after a received stop bit, therefore resetting the idle count.<br>0b - After the start bit<br>1b - After the stop bit |
| 1<br>PE | Parity Enable<br>Enables hardware parity generation and checking.<br>If parity is enabled, the bit immediately before the stop bit is treated as the parity bit.<br>0b - Disable<br>1b - Enable |
| 0<br>PT | Parity Type<br>Selects the type of parity, even or odd, if parity is enabled (CTRL[PE] = 1):<br>• Odd parity means that the total number of logic 1 bits in the data character, including the parity bit, is odd.<br>• Even parity means that the total number of 1s in the data character, including the parity bit, is even.<br>0b - Even parity<br>1b - Odd parity |

## 73.6.9 Data (DATA)

**Offset**

| Register | Offset |
|---|---|
| DATA | 1Ch |

### Function

Supports 8-bit, 16-bit, or 32-bit writes, each type of write performing a separate function. An 8-bit write to DATA[7:0] pushes {CTRL[R8T9], CTRL[R9T8], DATA[7:0]} the transmit FIFO with TSC clear. A 16-bit or 32-bit write pushes the data written into the FIFO and does not update the value of CTRL[R8T9] or CTRL[R9T8].

Reads and writes of this register are also involved in the automatic flag clearing mechanisms for some of the LPUART status fields.

**NOTE**

Reads return the contents of the read-only receive FIFO and writes go to the write-only transmit FIFO, making this register work as a set of two separate registers.

### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | NOISY | PARITYE | FRETSC | RXEMPT | IDLINE | LINBRK | R9T9 | R8T8 | R7T7 | R6T6 | R5T5 | R4T4 | R3T3 | R2T2 | R1T1 | R0T0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15<br>NOISY | Noisy Data Received<br>Indicates whether the current received dataword contained in DATA[R9:R0] is received with noise.<br>0b - Received without noise<br>1b - Received with noise |
| 14<br>PARITYE | Parity Error<br>Indicates whether the current received dataword contained in DATA[R9:R0] is received with a parity error.<br>0b - Received without a parity error<br>1b - Received with a parity error |
| 13<br>FRETSC | Frame Error Transmit Special Character<br>Specifies the way the dataword is received. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | For reads, this field indicates that the current received dataword contained in DATA[R9:R0] is received with a frame error. For writes, the field indicates that a break or idle character is to be transmitted instead of the contents in DATA[T9:T0]. T9 indicates a break character when it is 0 and indicates an idle character when it is 1. The contents of DATA[T8:T0] must be 0.<br>0b - Received without a frame error on reads or transmits a normal character on writes<br>1b - Received with a frame error on reads or transmits an idle or break character on writes |
| 12<br>RXEMPT | Receive Buffer Empty<br>Indicates whether the receive buffer contains valid data.<br>This field becomes 1 when there is no data in the receive buffer. The field does not consider data in the receive shift register.<br>0b - Valid data<br>1b - Invalid data and empty |
| 11<br>IDLINE | Idle Line<br>Indicates whether the receiver line was idle before receiving the character in DATA[9:0]. It can be read as “1” on the first character when the receiver is first enabled. The difference between this field and STAT[IDLE] is that, STAT[IDLE] flag does not set on an idle line after the receiver is first enabled, it needs to receive a character before it can become set, whereas this field does not have this limitation and can be set on the first character received if an idle line is detected beforehand.<br>0b - Not idle<br>1b - Idle |
| 10<br>LINBRK | LIN Break<br>Indicates whether the receiver line detected a LIN break before receiving the character in DATA[9:0]. This field requires the value of STAT[LBKDIF] to be 1. If this field is 0, the LIN break detect circuitry is disabled.<br>0b - Not detected<br>1b - Detected |
| 9<br>R9T9 | Read receive FIFO bit 9 or write transmit FIFO bit 9 |
| 8<br>R8T8 | Read receive FIFO bit 8 or write transmit FIFO bit 8 |
| 7<br>R7T7 | Read receive FIFO bit 7 or write transmit FIFO bit 7 |
| 6<br>R6T6 | Read receive FIFO bit 6 or write transmit FIFO bit 6 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 5<br>R5T5 | Read receive FIFO bit 5 or write transmit FIFO bit 5 |
| 4<br>R4T4 | Read receive FIFO bit 4 or write transmit FIFO bit 4 |
| 3<br>R3T3 | Read receive FIFO bit 3 or write transmit FIFO bit 3 |
| 2<br>R2T2 | Read receive FIFO bit 2 or write transmit FIFO bit 2 |
| 1<br>R1T1 | Read receive FIFO bit 1 or write transmit FIFO bit 1 |
| 0<br>R0T0 | Read receive FIFO bit 0 or write transmit FIFO bit 0 |

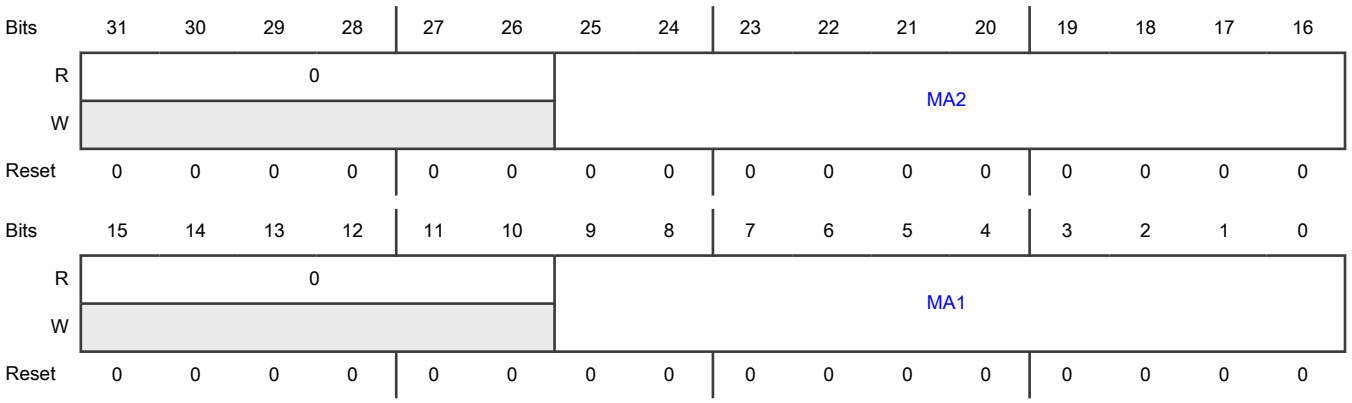
## 73.6.10 Match Address (MATCH)

### Offset

| Register | Offset |
|---|---|
| MATCH | 20h |

### Function
Provides addresses for address matching during the receiver operation.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25-16<br>MA2 | Match Address 2<br>Is compared to input data addresses when the most significant bit is 1 and the associated Baud Rate (BAUD) field is 1.<br>If a match occurs, the data that follows is transferred to Data (DATA). If a match fails, the data that follows is discarded. You must write to MATCH[MA1] and MATCH[MA2] only when the associated Baud Rate (BAUD) field is 0. |
| 15-10<br>— | Reserved |
| 9-0<br>MA1 | Match Address 1<br>Is compared to input data addresses when the most significant bit is 1 and the associated Baud Rate (BAUD) field is 1.<br>If a match occurs, the data that follows is transferred to Data (DATA). If a match fails, the data that follows is discarded. You must write to MATCH[MA1] and MATCH[MA2] fields only when the associated field in Baud Rate (BAUD) is 0. |

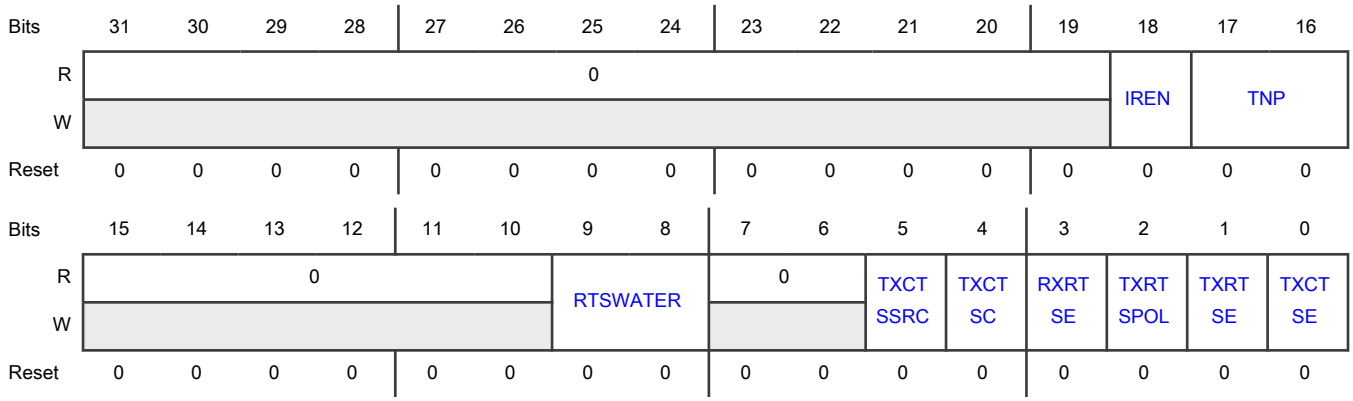
## 73.6.11 MODEM IrDA (MODIR)

### Offset

| Register | Offset |
|---|---|
| MODIR | 24h |

### Function
Controls options for setting the MODEM configuration.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18<br>IREN | IR Enable<br>Enables IR modulation and demodulation.<br>You must change the value of this field only when both the transmitter and receiver are disabled.<br>0b - Disable<br>1b - Enable |
| 17-16<br>TNP | Transmitter Narrow Pulse<br>Specifies whether LPUART transmits a 1 ÷ OSR, 2 ÷ OSR, 3 ÷ OSR, or 4 ÷ OSR narrow pulse when the IR pulse is enabled.<br>You must change the value of this field only when both the transmitter and receiver are disabled.<br>The IR pulse width must be configured to less than half of the OSR. Common pulse widths are 3 ÷ 16, 1 ÷ 16, 1 ÷ 32, or 1 ÷ 4 of the bit length. You can configure these by selecting the appropriate OSR and pulse width.<br>00b - 1 ÷ OSR<br>01b - 2 ÷ OSR<br>10b - 3 ÷ OSR<br>11b - 4 ÷ OSR |
| 15-10<br>— | Reserved |
| 9-8<br>RTSWATER | Receive RTS Configuration<br>Configures the assertion and negation of the receiver's RTS_B output.<br>The receiver's RTS_B output negates when the number of empty words in the receive FIFO is greater or equal to the value of this field. If this field is 0, the RTS_B pin negates when the receive FIFO is full. For the purpose of receive RTS_B generation, the number of words in the receive FIFO updates when a start bit is detected. This supports additional latency between RTS_B negation and the external transmitter ceasing transmission. If both receive RTS_B and address or data matching is enabled, RTS_B could assert at the end of a character if there exists no match.<br>You must change the value of this field only when the receiver is disabled. |
| 7-6<br>— | Reserved |
| 5<br>TXCTSSRC | Transmit CTS Source<br>Configures the source of the CTS input.<br>0b - The CTS_B pin<br>1b - An internal connection to the receiver address match result |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 4<br>TXCTSC | Transmit CTS Configuration<br>Configures whether the CTS state or input is checked or sampled at the start of each character or only when the transmitter is idle.<br>0b - Sampled at the start of each character<br>1b - Sampled when the transmitter is idle |
| 3<br>RXRTSE | Receiver RTS Enable<br>Allows the RTS output to control the CTS input of the transmitting device to prevent receiver overrun.<br>You must change the value of this field only when the receiver is disabled.<br>If this field is 0, the receiver has no effect on RTS.<br>If this field is 1, RTS is deasserted if STAT[RDRF] is 1 or a start bit is detected that causes STAT[RDRF] to become 1. RTS is asserted if STAT[RDRF] is 0 and has not detected a start bit that causes STAT[RDRF] to become 1.<br>**NOTE**<br>Do not write 1 to both MODIR[RXRTSE] and MODIR[TXRTSE].<br>0b - Disable<br>1b - Enable |
| 2<br>TXRTSPOL | Transmitter RTS Polarity<br>Controls the polarity of the transmitter RTS.<br>This field does not affect the polarity of the receiver RTS that remains negated in the active-low state unless MODIR[TXRTSE] is 1. You must change the value of this field only when the transmitter is disabled.<br>0b - Active low<br>1b - Active high |
| 1<br>TXRTSE | Transmitter RTS Enable<br>Controls the operation of RTS before and after a transmission.<br>You must change the value of this field only when the transmitter is disabled. If this field is 0, the transmitter has no effect on RTS, and if this field is 1, a character is placed into an empty transmit shift register. RTS asserts 1-bit time before the start bit is transmitted and deasserts 1-bit time after all characters in the transmitter FIFO and shift register are completely sent, including the last stop bit.<br>0b - Disable<br>1b - Enable |
| 0<br>TXCTSE | Transmitter CTS Enable<br>Enables the operation of the transmitter.<br>You can write 1 to this field irrespective of the states of MODIR[TXRTSE] and MODIR[RXRTSE]. If this field is 1, the transmitter checks the state of the CTS signal each time it is ready to send a character. If CTS is asserted, the character is sent. If CTS is deasserted, the TXD signal remains in the mark state and |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | transmission is delayed until CTS is asserted. Changes in CTS, when a character is being sent, do not affect its transmission.<br>0b - Disable<br>1b - Enable |

## 73.6.12 FIFO (FIFO)

### Offset

| Register | Offset |
|---|---|
| FIFO | 28h |

### Function

Provides you the ability to turn on and turn off the FIFO functionality.

This register also provides you the size of the FIFO that has been implemented. You can read this register at any time and must write to it only when CTRL[RE] and CTRL[TE] are 0 and the FIFO is empty.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | | | | | TXEMPT | RXEMPT | 0 | | | | TXOF | RXUF |
| W | | | | | | | | | | | | | | | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | RXIDEN | | | TXOFE | RXUFE | TXFE | TXFIFOSIZE | | | RXFE | RXFIFOSIZE | | |
| W | TXFLUSH | RXFLUSH | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

### Fields

| Field | Function |
|---|---|
| 31-24<br>— | Reserved |
| 23<br>TXEMPT | Transmit FIFO Or Buffer Empty<br>Indicates whether the transmit buffer is empty. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | This field becomes 1 when there is no data in the transmit FIFO or buffer. The field does not consider data in the transmit shift register.<br>0b - Not empty<br>1b - Empty |
| 22<br>RXEMPT | Receive FIFO Or Buffer Empty<br>Indicates whether the receive buffer is empty.<br>This field becomes 1 when there is no data in the receive FIFO or buffer. The field does not consider data in the receive shift register.<br>0b - Not empty<br>1b - Empty |
| 21-18<br>— | Reserved |
| 17<br>TXOF | Transmitter FIFO Overflow Flag<br>Indicates whether more data has been written to the transmit FIFO than it can hold.<br>If this field is 0, no transmit FIFO overflow has occurred since the last time the field was cleared, and if this field is 1, at least one transmit FIFO overflow has occurred since the last time the field was cleared.<br>This field becomes 1 regardless of the value of FIFO[TXOFE]. However, an interrupt is issued to the host only if FIFO[TXOFE] is 1.<br>**NOTE**<br>This field behaves differently for register reads and writes.<br>When reading<br>0b - No overflow<br>1b - Overflow<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 16<br>RXUF | Receiver FIFO Underflow Flag<br>Indicates whether more data has been read from the receive FIFO than was present.<br>If this field is 0, no receive FIFO underflow has occurred since the last time the field was cleared, and if this field is 1, at least one receive FIFO underflow has occurred since the last time the field was cleared.<br>This field becomes 1 regardless of the value of FIFO[RXUFE]. However, an interrupt is issued to the host only if FIFO[RXUFE] is 1.<br>**NOTE**<br>This field behaves differently for register reads and writes. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | When reading<br>0b - No underflow<br>1b - Underflow<br>When writing<br>0b - No effect<br>1b - Clear the flag |
| 15<br>TXFLUSH | Transmit FIFO Flush<br>Causes all data that is stored in the transmit FIFO to be flushed.<br>If you write 0 to this field, no flush operation occurs, and if you write 1 to this field, all data in the transmit FIFO or buffer clears out.<br>This does not affect data in the transmit shift register.<br>0b - No effect<br>1b - All data flushed out |
| 14<br>RXFLUSH | Receive FIFO Flush<br>Causes all data that is stored in the receive FIFO to be flushed.<br>If you write 0 to this field, no flush operation occurs, and if you write 1 to this field, all data in the receive FIFO or buffer clears out.<br>This does not affect data in the receive shift register.<br>0b - No effect<br>1b - All data flushed out |
| 13<br>— | Reserved |
| 12-10<br>RXIDEN | Receiver Idle Empty Enable<br>Enables STAT[RDRF] to become 1 when the receiver is idle for a number of idle characters and the FIFO is not empty.<br>000b - Disable STAT[RDRF] to become 1 because of partially filled FIFO when the receiver is idle<br>001b - Enable STAT[RDRF] to become 1 because of partially filled FIFO when the receiver is idle for one character<br>010b - Enable STAT[RDRF] to become 1 because of partially filled FIFO when the receiver is idle for two characters<br>011b - Enable STAT[RDRF] to become 1 because of partially filled FIFO when the receiver is idle for four characters<br>100b - Enable STAT[RDRF] to become 1 because of partially filled FIFO when the receiver is idle for eight characters |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 101b - Enable STAT[RDRF] to become 1 because of partially filled FIFO when the receiver is idle for 16 characters<br>110b - Enable STAT[RDRF] to become 1 because of partially filled FIFO when the receiver is idle for 32 characters<br>111b - Enable STAT[RDRF] to become 1 because of partially filled FIFO when the receiver is idle for 64 characters |
| 9<br>TXOFE | Transmit FIFO Overflow Interrupt Enable<br>Enables FIFO[TXOF] to generate an interrupt to the host.<br>0b - Disable<br>1b - Enable |
| 8<br>RXUFE | Receive FIFO Underflow Interrupt Enable<br>Enables FIFO[RXUF] to generate an interrupt to the host.<br>0b - Disable<br>1b - Enable |
| 7<br>TXFE | Transmit FIFO Enable<br>Enables the transmit FIFO.<br>If this field is 0, the transmit buffer operates as a FIFO of depth equal to 1 dataword, regardless of the value in FIFO[TXFIFOSIZE]. Both CTRL[TE] and CTRL[RE] must be 0 before you change the value of this field.<br>If this field is 1, the built-in FIFO structure for the transmit buffer is enabled. FIFO[TXFIFOSIZE] indicates the size of the FIFO structure.<br>0b - Disable<br>1b - Enable |
| 6-4<br>TXFIFOSIZE | Transmit FIFO Buffer Depth<br>Indicates the maximum number of transmit datawords (transmit FIFO buffer depth) that can be stored in the transmit buffer.<br>000b - 1<br>001b - 4<br>010b - 8<br>011b - 16<br>100b - 32<br>101b - 64<br>110b - 128<br>111b - 256 |
| 3 | Receive FIFO Enable |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| RXFE | Enables the receive FIFO.<br>If this field is 0, the receive buffer operates as a FIFO of depth equal to 1 dataword, regardless of the value in FIFO[RXFIFOSIZE]. Both CTRL[RE] and CTRL[TE] must be 0 before you change the value of this field.<br>If this field is 1, the built-in FIFO structure for the receive buffer is enabled. FIFO[RXFIFOSIZE] indicates the size of the FIFO structure.<br>0b - Disable<br>1b - Enable |
| 2-0<br>RXFIFOSIZE | Receive FIFO Buffer Depth<br>Indicates the maximum number of receive datawords (receive FIFO buffer depth) that can be stored in the receive buffer before an overrun occurs.<br>000b - 1<br>001b - 4<br>010b - 8<br>011b - 16<br>100b - 32<br>101b - 64<br>110b - 128<br>111b - 256 |

### 73.6.13 Watermark (WATER)

#### Offset

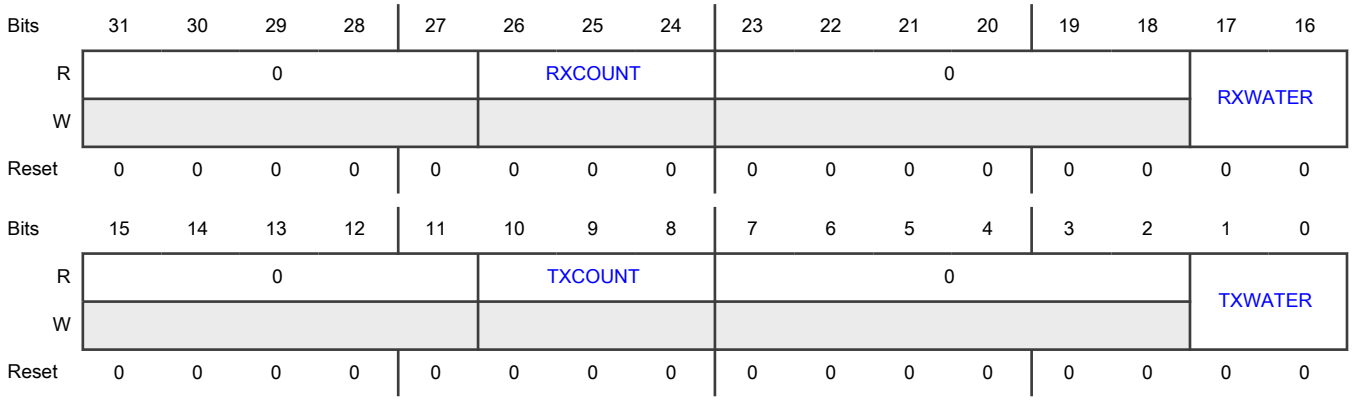
| Register | Offset |
|---|---|
| WATER | 2Ch |

#### Function

Provides the ability to set a programmable threshold for notification, or sets the programmable thresholds to indicate that transmit data can be written or receive data can be read.

You may read this register at any time but must write to it only when CTRL[TE] is 0.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26-24<br>RXCOUNT | Receive Counter<br>Indicates the number of datawords in the receive FIFO or buffer.<br>If a dataword is being received in the receive shift register, it is not included in the count. This value may be used in conjunction with FIFO[RXFIFOSIZE] to calculate the room left in the receive FIFO or buffer. |
| 23-18<br>— | Reserved |
| 17-16<br>RXWATER | Receive Watermark<br>Generates an interrupt or a DMA request if the number of datawords in the receive FIFO or buffer is greater than the value of this field.<br>For proper operation, the value of this field must be less than the size of the receive FIFO or buffer, as indicated by FIFO[RXFIFOSIZE] and FIFO[RXFE]. |
| 15-11<br>— | Reserved |
| 10-8<br>TXCOUNT | Transmit Counter<br>Indicates the number of datawords in the transmit FIFO or buffer.<br>If a dataword is being transmitted to the transmit shift register, it is not included in the count. This value may be used in conjunction with the value of FIFO[TXFIFOSIZE] to calculate the room left in the transmit FIFO or buffer. |
| 7-2<br>— | Reserved |
| 1-0 | Transmit Watermark |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| TXWATER | Generates an interrupt or a DMA request when the number of datawords in the transmit FIFO or buffer is equal to or less than the value of this field.<br>For proper operation, the value of this field must be less than the size of the transmit buffer or FIFO, as indicated by FIFO[TXFIFOSIZE] and FIFO[TXFE]. |

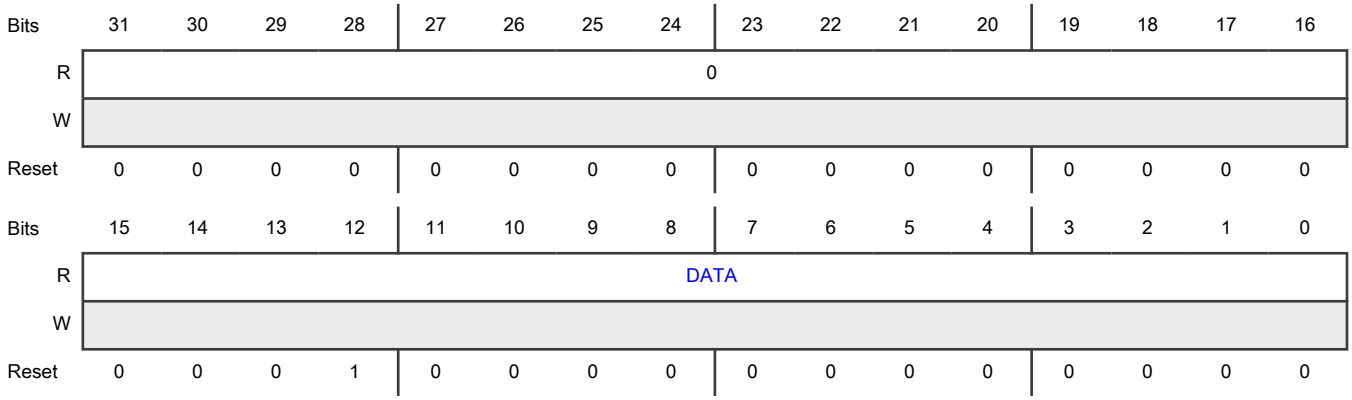
### 73.6.14 Data Read-Only (DATARO)

#### Offset

| Register | Offset |
|---|---|
| DATARO | 30h |

#### Function
Indicates the first entry in the receive FIFO, but does not pull data from the FIFO.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>DATA | Receive Data<br>Indicates the first entry from the receive FIFO.<br>This register has the same functionality as that of Data (DATA). |

## 73.7 Glossary

**Baud rate** Number of bits per second that LPUART transmits or receives

| | |
|---|---|
| **Break character** | Break character is generated when the transmitter is holding the data line at the space level for at least one character time |
| **Oversampling** | Number of times the receive circuitry samples the receive input per baud period (that is, per data bit) |

# Chapter 74
# Quad Serial Peripheral Interface (QuadSPI)

## 74.1 Chip-specific QuadSPI information

### 74.1.1 QuadSPI configuration

**Table 550. QuadSPI instances**

| Instance | MCXE31B | MCXE315/ MCXE316/MCXE317 |
|---|---|---|
| QuadSPI | Yes | No |

**Table 551. QuadSPI configuration details**

| Configuration | MCXE317/ MCXE31B |
|---|---|
| QuadSPI Tx FIFO size | 32 words |
| QuadSPI Rx FIFO size | 32 words |
| Look Up Table Size | 4 words |
| AHB Buffer Size | 256 Bytes |

The Quad Serial Peripheral Interface (QuadSPI) block acts as an interface to single serial flash devices, with up to four bidirectional data lines. The QuadSPI supports 4-pin Quad A interface only in SDR mode.

**NOTE**

Boot from QuadSPI is not supported but execution from external memory is supported.

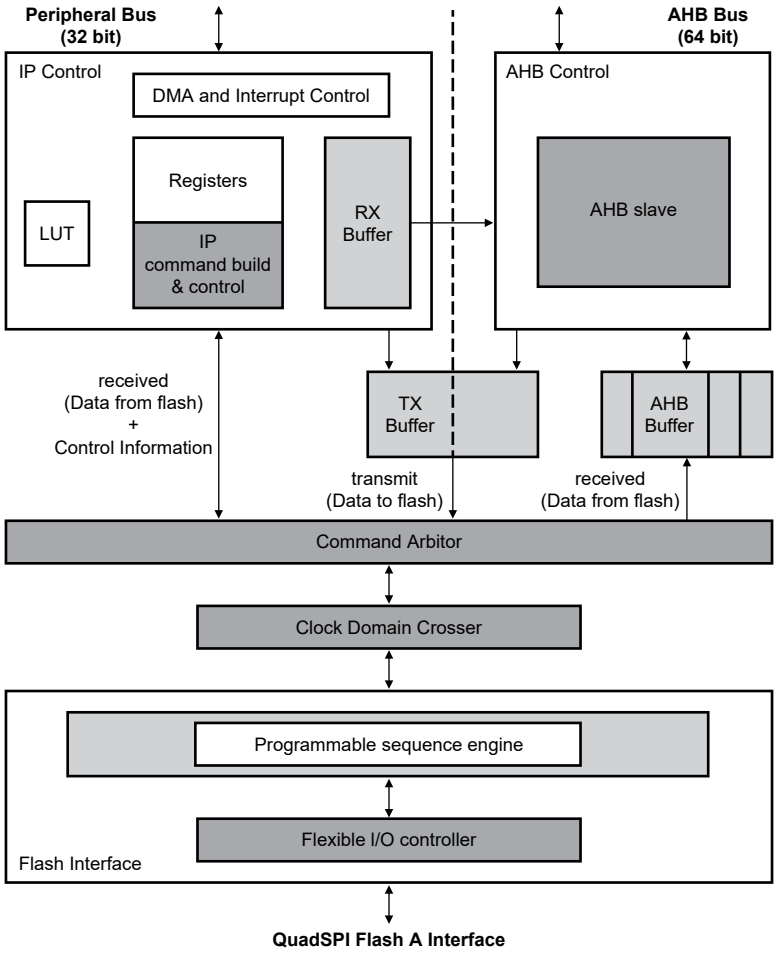



Figure 428. Block diagram

## 74.1.2 Supported read modes

The table below provides an overview of the QuadSPI read modes.

Table 552. QuadSPI read modes

| Read modes | | SDR support (QuadSPI_MCR [DDR_EN]=0) | QuadSPI_MCR [DQS_EN] | QuadSPI_MCR [DQS_FA_SEL] | DLL/Data learning support |
|---|---|---|---|---|---|
| DQS sampling method | Pad loopback | Yes | 1 | 01 | No |

## 74.1.3 QuadSPI initialization sequence

Following initialization sequence should be followed for proper QuadSPI operation:

- Enable QuadSPI module by MC_ME peripheral clock enable (register PRTNx_COFBy_CLKEN present within MC_ME). Refer MC_ME chapter for peripheral mapping.
- Configure the SIUL2 registers MSCR[OBE] as 1 and MSCR[SSS] as 0 for QuadSPI_SCKFA pin.
- Initialize QuadSPI SCKFA by writing a sequence of 1010 to the SIUL2 register GPDO[PDO_a] for QuadSPI_SCKFA pin.
- Configure the SIUL2 register MSCR[OBE] back as 0 for the pins.

- Initiate a dummy flash read to reset all DQS flops by itself and any crossing from DQS domain to IPS/AHB is taken care by CDC logic.
- Initiate a peripheral software reset to QuadSPI controller by writing to QuadSPI controller's MCR.
- Post this initialization sequence, the QuadSPI will work in intended deterministic manner.

**NOTE**

QuadSPI initialization is to be done before using QuadSPI after each functional reset.

### 74.1.4 Pad clock loopback

This chip supports pad clock loopback. The QuadSPI can be configured to use clock loopback to sample input data. SCK is delayed by the SCK pin output delay, plus the SCK pin input delay using pad loopback, and is configured by setting QuadSPI config registers SOCCR[SOCCFG] and MCR[DQS_FA_SEL]. Enabling the loopback version of SCK can inprove the setup time of the input data from the Flash.

For details of these register, see QuadSPI register descriptions.

### 74.1.5 QuadSPI SOC Configuration register SOCCR[SOCCFG] implementation

The QuadSPI SOC Configuration register QuadSPI_SOCCR[SOCCFG] register is used to control dummy loopback pads and obe_pull_timing_relax_b. Below is the description of it's bits:

**Table 553. SOCCR[SOCCFG] implementation**

| Bit | Description |
|---|---|
| Bit[0] | obe_pull_timing_relax_b : enables the timing relaxation by pulling obe for pad 1 for half cycle, if 0 then enabled else disabled. |
| Bit[1] | ibe of QSPIA_SCK_DUMMY pad. 0: Disable input reciever 1: Enable input reciever. |
| Bit[2] | obe of QSPIA_SCK_DUMMY pad. 0: Disable output driver 1: Enable output driver. |
| Bit[3] | dse of QSPIA_SCK_DUMMY pad. 0: Disable drive strength 1: Enable drive strength. |
| Bit[4] | pue of QSPIA_SCK_DUMMY pad. 0: Disable internal pullup or pulldown resistor 1: Enable internal pullup or pulldown resistor. |
| Bit[5] | pus of QSPIA_SCK_DUMMYpad. 0: Enable internal pulldown resistor if pue is set 1: Enable internal pullup resistor if pue is set. |
| Bit[6] | sre of QSPIA_SCK_DUMMY pad. 0: Disable slew rate control 1: Enable slew rate control. |
| Bit[31:7] | Reserved |

To Enable the Quadspi dummy PAD loopback use following settings

For Flash-A: MCR[DQS_FB_SEL] = 0x2

SOCCR[SOCCFG] = 0x0000000E (ibe=1, obe=1, dse=1 and sre=0)

### 74.1.6 QuadSPI AHB Buffer write access control

In MCXE31B and MCXE317 devices, the QuadSPI AHB buffer does not support writes. A write to QuadSPI AHB buffer results in error response from QuadSPI.

QuadSPI might behave unexpectedly in case if its AHB buffer is written with QuadSPI MCR[MDIS] =1. In cases wherein QuadSPI is disabled with above MDIS control bit, XRDC must also be appropriately configured to block the accesses to QSPI AHB buffer and indicate an error response.

## 74.2 Introduction

The QuadSPI module acts as an interface to a single serial flash memory device, with up to four bidirectional data lines.

### 74.2.1 Features

QuadSPI supports the following features:

- Flexible sequence engine to support various flash memory vendor devices. As there is no specific standard, the module supports various kinds of flash memories from different vendors. See Serial flash memory devices for example sequences.
- Single, dual, and quad modes of operation supported for Quad flash memories
- AHB master to read RX buffer data through AMBA AHB (64-bit width interface) or IPS registers space (32-bit access) and fill TX buffer via IPS registers space (32-bit access) or using AHB (64-bit width interface)
  - AHB master can be a DMA with a configurable inner loop size
- Multi-master accesses are allowed
  - Flexible and configurable buffer for each master—total available buffer size is 256 bytes.
- All AHB accesses to flash/RAM memory devices are directly memory mapped to the chip system memory
- Programmable sequence engine to cater to future command/protocol changes and ability to support all existing vendor commands and operations. The software needs to select the corresponding sequence according to the connected flash memory device.
  - Support for 3-byte and 4-byte addressing

### 74.2.2 RX buffer push event

To add the valid entries into the RX buffer

By default, each buffer push event adds two entries to the RX buffer because the interface to the serial clock domain is 64 bits in width. Depending on the number of bytes read from the serial flash memory device, it is possible for the very last buffer push event that only one entry is added.

RBSR[RDBFL] is incremented by the number of entries added to the RX buffer.

### 74.2.3 RX buffer POP event

To remove valid entries from the RX buffer

Each buffer POP event removes (RBCT[WMRK] + 1) valid entries from the buffer. BSR[RDBFL] is decremented by the same number and RBSR[RDCTR] is incremented accordingly.

### 74.2.4 Block diagram

The following figure shows a block diagram of the QuadSPI module.

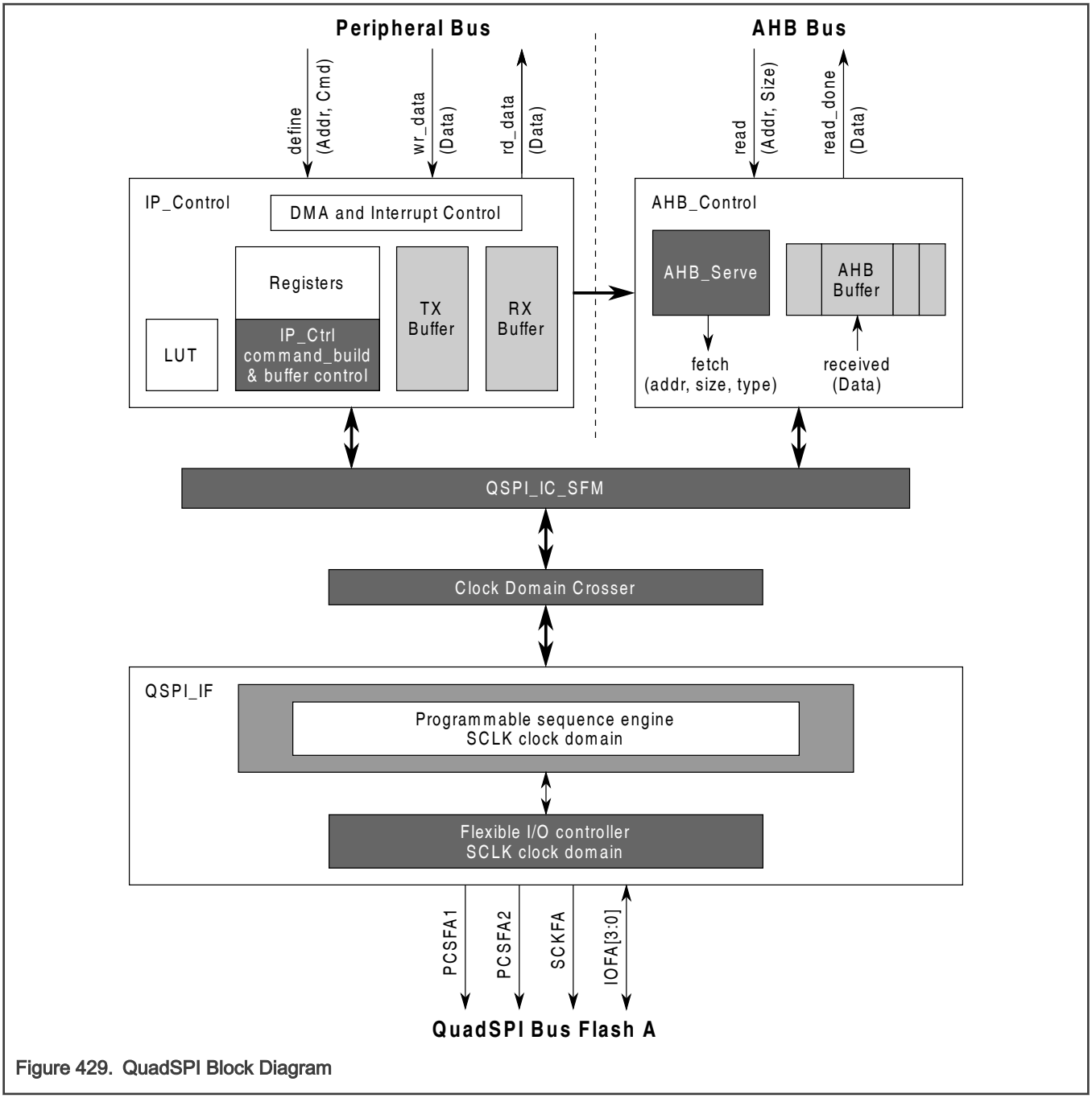



Figure 429. QuadSPI Block Diagram

## 74.2.5 QuadSPI modes of operation

QuadSPI supports the following modes of operation:

- Normal mode: You can use this mode for write or read accesses to an external serial flash memory device. See Normal mode for details.
  - Serial flash memory write: You can program data into the flash memory through the IP interface only. See Flash memory programming for details.
  - Serial flash memory read: Read the contents of the serial flash memory device. Two separate read channels are available through the RX buffer and AHB buffer. See Flash memory read for details.

- Module Disable mode: You can use this mode for disabling serial flash memory clock and AHB command. The clock to the non-memory mapped logic in QuadSPI can be stopped in the Module Disable mode. The module enters the mode by setting MCR[MDIS].

## 74.3 External signal description

This section provides the external signal information for the QuadSPI module.

The following table lists the external signals belonging to the module in conjunction with the different modes of operation.

**Table 554. Signal properties**

| Signal name | Function | Direction | Description |
|---|---|---|---|
| PCSFA1 | Peripheral Chip Select Flash Memory A1 | O | This signal is the chip select for the serial flash memory device A1 that represents the first of the two flash memory devices that share IOFA. |
| PCSFA2 | Peripheral Chip Select Flash Memory A2 | O | This signal is the chip select for the serial flash memory device A2 that represents the second of the two flash memory devices that share IOFA. |
| SCKFA | Serial Clock Flash Memory A | O | This signal is the serial clock output to the serial flash memory device A. |
| IOFA[3:0] | Serial I/O Flash memory A | I/O | These signals are the data I/O lines to/from the serial flash memory device A. See Driving external signals for details about the signal drive and timing behavior. Note that the signal pins of the serial flash memory device may change their function according to the SFM Command executed, leaving them as control inputs when single and dual instructions are executed. The module supports driving these inputs to dedicated values. In single I/O mode, QuadSPI drives data on IOFA[0] and expects data on IOFA[1]. |

**NOTE**

Please refer to chip specific information to check the configuration of QuadSPI block.

### 74.3.1 Driving external signals

Single/dual/quad instructions

Depending on the serial flash memory device connected to the QuadSPI module, there are instructions using a different number of data lines:

- Single pad: Single line I/O with one data out and one data in line to/from the serial flash memory device
- Dual pad: Dual line I/O with two bidirectional I/O lines, driven alternatively by the serial flash memory device or the QuadSPI module
- Quad pad: Quad line I/O with four bidirectional I/O lines, driven alternatively by the serial flash memory device or the QuadSPImodule

The different phases of the serial flash memory access scheme are shown in the following figure.

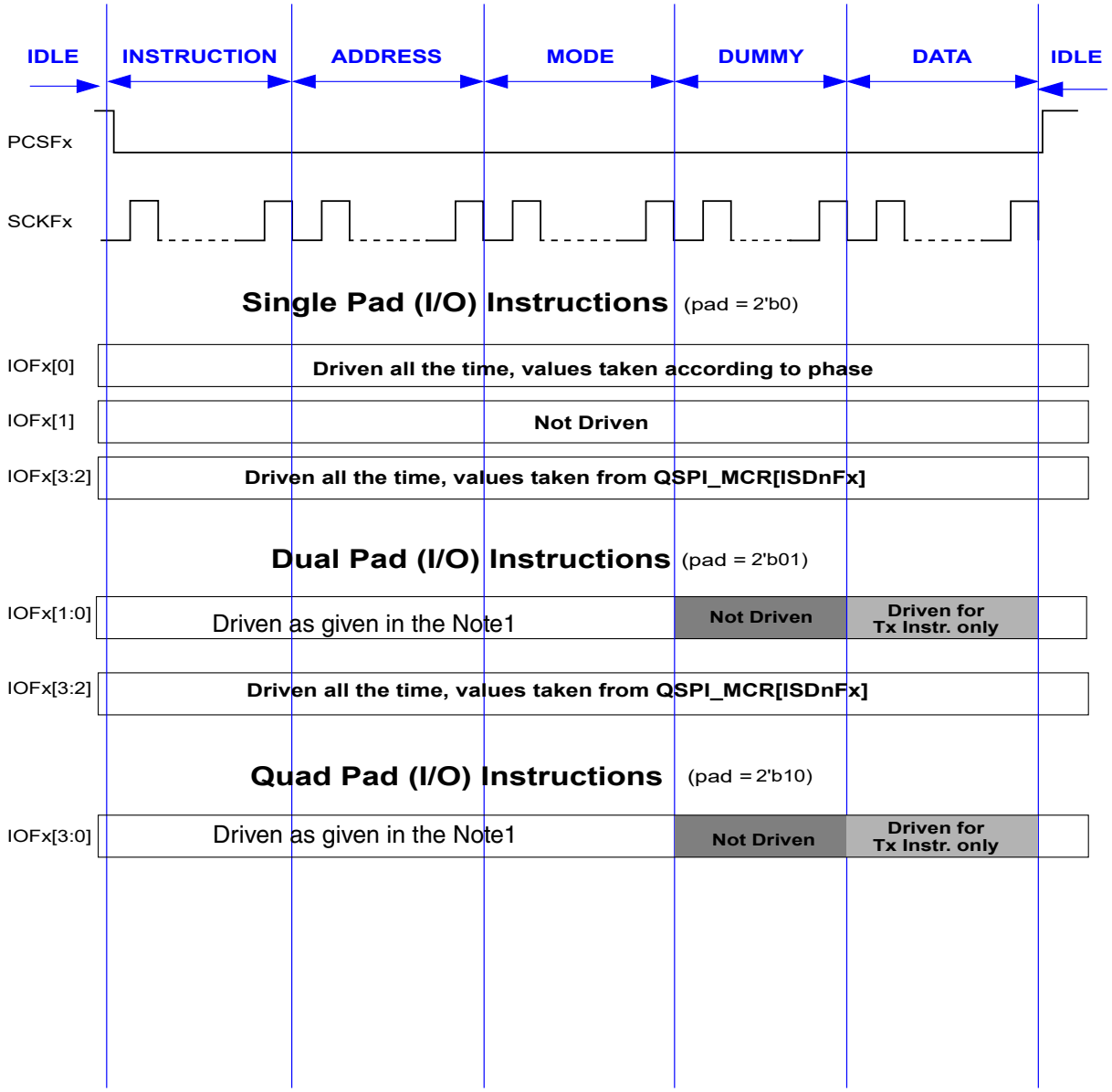



Figure 430. Serial flash memory access scheme

Note1:The IOs are driven from QuadSPI as per the number of pads configured for ongoing phase.

Note:The lines status will change based on command mode in case of instruction, address and mode phases. It can be either 1,2 or 4 lines

Following are the different phases and the I/O driving characteristics of the QuadSPI module:

- Idle: Serial flash memory device not selected – no interaction with the serial flash memory device and the IOFx signals are driven.
- Instruction: Serial flash memory device selected – the instruction is sent to the serial device and all the IOFx signals are driven.
- Address: Serial flash memory address is sent to the device – all the IOFx signals are driven and this phase is not applicable for all SFM commands.
- Mode: Mode bytes are sent to the serial flash memory device – all the IOFx signals are driven and this phase is not applicable for all SFM commands.

- Dummy: Dummy clocks are provided to the serial flash memory device. See Figure 430 for the IOFx signals driven. The actual data lines required for the SFM command executed are not driven for data read commands.

**NOTE**

— This phase is not applicable for all the SFM commands.

— All read commands in Dual pad, Quad pad modes must use a Dummy phase immediately before the Data phase. The Dummy phase pad configuration in the LUT must use the same number of pads as the subsequent Data phase. Note that this restriction is not applicable to Single-pad mode.

- Data: Serial flash memory data are sent to or received from the serial flash memory device. See the preceding figure for the IOFx signals driven. The actual data lines required for the SFM command executed are not driven for data read commands.

**NOTE**

This phase is not applicable for all the SFM commands.

The PCSFx and SCKFx signals are driven permanently throughout all the phases. In the individual flash memory mode, this applies to the selected flash memory device.

Access to a single, individual serial flash memory device

See Serial flash memory access schemes for details.

Read access to two serial flash memory devices attached to the QuadSPI module in parallel. See Serial flash memory access schemes for details.

## 74.4 Functional description

This section provides a functional description of the QuadSPI module.

### 74.4.1 Serial flash memory access schemes

In the individual flash memory mode, all supported commands are available.

### 74.4.2 Normal mode

This mode allows communication with an external serial flash memory device. Compared to the standard SPI protocol, this communication method uses up to four bidirectional data lines operating at high-data rates. The communication to the external serial flash memory device consists of an instruction code and optional address, mode, dummy, and data transfers. The flexible programmable core engine described below is immune to a wide variety of command or protocol differences in the serial flash memory devices provided by various flash memory vendors.

#### 74.4.2.1 Programmable sequence engine

The core of the QuadSPI module is a programmable sequence engine that works on "instruction-operand" pairs. The core controller executes each programmed instruction sequentially. The complete list of instructions and the corresponding operands are provided in the following table.

**Table 555. Instruction set**

| Instruction | Instruction encoding | Pins | Operand | Action on serial flash memories |
|---|---|---|---|---|
| CMD | 1d | N={0,1,2}d<br>0d - One pad<br>1d - Two pads<br>2d - Four pads | 8-bit command value | Provides the serial flash memory with the SFM command operand (Encoded) on the number of pads specified in STR mode. |

*Table continues on the next page...*

Table 555. Instruction set (continued)

| Instruction | Instruction encoding | Pins | Operand | Action on serial flash memories |
|---|---|---|---|---|
| ADDR | 2d | | Number of address bits to be sent (for example, 24d => 24 address bits required) | Provide the serial flash memory with address cycles according to the operand on the number of pads specified<br>The actual address to be provided is derived from the incoming address in case of AHB-initiated transactions and the value of SFAR in case of IPS-initiated transactions. |
| DUMMY | 3d | | Number of dummy clock cycles (should be <= 64 and > 2 cycles) | Provide the serial flash memory with dummy cycles according the operand<br>The PAD information defines the number of pads in input mode. For example, one pad implies that pad 1 is not driven, rest all are driven. |
| MODE | 4d | | 8-bit mode value | Provide the serial flash memory with 8-bit operand on the number of pads specified |
| MODE2 | 5d | N={0,1}d | 2-bit mode value | Provide the serial flash memory with 2-bit operand on the number of pads[1] specified |
| MODE4 | 6d | N={0,1,2}d | 4-bit mode value | Provide the serial flash memory with 4-bit operand on the number of pads[2] specified |
| READ | 7d | N={0,1,2}d<br>0d - One pad<br>1d - Two pads<br>2d - Four pads | Read data size in bytes (for AHB transactions, your application should ensure that data size is a multiple of 8 bytes) | Read data from flash memory on the number of pads specified<br>The data size can be overwritten by writing to the ADATSZ field of the BUFxCR registers for AHB-initiated transactions and to the IDATSZ field of the IP Configuration Register (IPCR) for IPS-initiated transactions. |
| WRITE | 8 | | Write data size in bytes | Write data on the number of pads specified<br>The data size can be overwritten by writing to the IDATSZ field of IP Configuration Register (IPCR). |
| JMP_ON_CS[3] | 9d | NA | Instruction number | Every time the CS is deasserted, jump to the instruction pointed to by the operand. This instruction allows the programmer to specify the behavior of the controller when a new read transaction is initiated following a CS deassertion. |
| STOP[3] | 0d | NA | NA | Stop execution; deassert CS |

1. For a one-pad instruction, MODE2 takes two serial flash memory clock cycles on the flash memory interface.
2. For a one-pad instruction, MODE4 takes four serial flash memory clock cycles on the flash memory interface. For a four-pad instruction, MODE4 takes one serial flash memory clock cycle on the flash memory interface.
3. Sequence ending with this instruction must have all remaining bits as 0s after it.

The programmable sequence engine allows you to configure the QuadSPI module according to the serial flash memory connected on board. This flexible structure is compatible with new command or protocol changes from different vendors.

#### 74.4.2.2 Flexible read xAHB buffers

To reduce the latency of the reads for AHB masters, the data read from serial flash memory is buffered in flexible AHB buffers. There are four such flexible buffers. The size of each of these buffers is configurable with the minimum size being 0 bytes and maximum size being the size of the complete buffer instantiated (256 bytes). The size of buffer 0 ranges from 0 to BUF0IND. The size of buffer 1 ranges from BUF0IND to BUF1IND, buffer2 from BUF1IND to BUF2IND and, buffer 3 ranges from BUF2IND to the size of the complete buffer (256 bytes).

Each flexible AHB buffer is associated with the following:

- An AHB master: Optionally, buffer3 may be configured as an "all master" buffer by writing 1 to BUF3CR[ALLMST]. When buffer3 is configured in such a way, any access from a master not associated with any other buffer is routed to buffer3.
- A datasize field representing the amount of data to be fetched from the flash memory on every missed access.

The master ID of every incoming request is checked and the data is returned or fetched into the corresponding associated buffer. See the chip-specific QuadSPI information for details about master IDs and their corresponding components. Every missed access to the buffer causes the controller to clear the buffer and fetch the BUFxCR[ADATSZ] amount of data from the serial flash memory. As such, you need not configure the buffer size to be greater than ADATSZ because the locations greater than ADATSZ are never used. For any AHB access, the sequence pointed to by BFGENCR[SEQID] is used for the initiated flash memory transaction. The data is returned to the master as soon as the requested amount is read from the serial flash memory. The controller; however, continues to prefetch the rest of the data in anticipation of a next consecutive request. See Figure 431 that shows flexible AHB buffers.

BFGENCR[SEQID] points to an index of the LUT. See LUT for details.

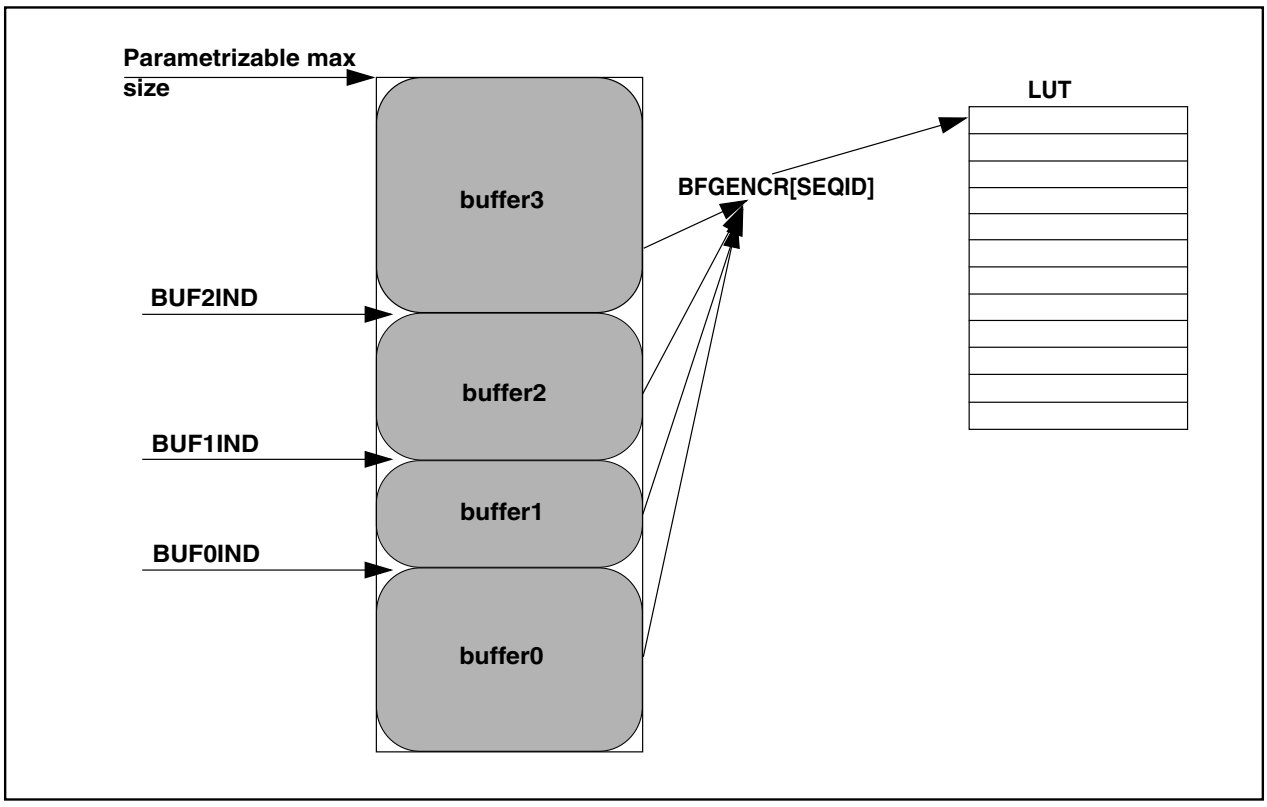



Figure 431. Flexible read AHB buffers

#### 74.4.2.3 Extended prefetch or Streaming mode

There is an option in QuadSPI to fetch more data than the maximum size of AHB prefetch buffer, which is 1024 bytes. This mode can be used in streaming use cases where the data is constantly read in incremental order.

To use this mode, configure buffer3 as an "all master" buffer by writing 1 to BUF3CR[ALLMST], and program BUF3CR[ADATSZ] greater than 1024 bytes. Also ensure that complete buffer is assigned to Buffer 3 Configuration Register (BUF3CR) and remaining buffer sizes should be programmed to 0 by writing 0 to Buffer 0 Top Index Register (BUF0IND), Buffer 1 Top Index Register (BUF1IND), and Buffer 2 Top Index Register (BUF2IND) registers. The maximum size that can be programmed for prefetch is 4096 bytes. In this mode, the complete 1K buffer size is used in rollover fashion, such that when the complete buffer is written, it starts writing the prefetch data again from starting location 0. So, if the AHB read is done continuously at sufficient speed, complete 4K data can be read without the buffer going full. In this mode, QuadSPI considers last AHB read address as the read pointer in AHB buffer, and when the prefeched data is being written in one location before the read pointer, it generates AHB full flag and prefetch is aborted. So, to efficiently utilize the streaming mode, ensure that AHB read is done in incremental/serial addresses with sufficient throughput. Also when the software is switching from streaming mode to normal modes, or vice-versa, it should clear the AHB buffers using software abort.

#### 74.4.2.4 Abort mechanism during AHB read

Any ongoing read transaction is aborted if a request from the same master arrives for a location other than the location at which the transaction is going on. The abort can happen at any point of time.

#### 74.4.2.5 HBURST support with AHB read

QuadSPI controller supports HBURST and HSIZE on the AHB read interface. HBURST indicates if the transfer forms part of a burst. Four, eight, and sixteen beat bursts are supported and the bursts might either be incrementing or wrapping. HSIZE indicates the size of the transfer, and supports 8-, 16-, 32-, and 64-bit data sizes. In case of WRAP accesses, QuadSPI generates aligned accesses to serial flash memory if there is no buffer hit for any incoming, non-sequential AHB read access. In case there is a buffer hit, the incoming address in the haddr line is latched as it is. If the total burst size is more than the data prefetch size, an error response is generated and the value of FR[AIBSEF] is configured as 1. The data prefetch size can be defined by BUFxCR[ADATSZ] or data size mentioned in the sequence pointed to by the SEQID field when ADATSZ is programmed as 0. In case of wrap burst, data prefetch size must be greater than or equal to the wrap burst size + 32 bytes. A few examples are shown in the figure below:

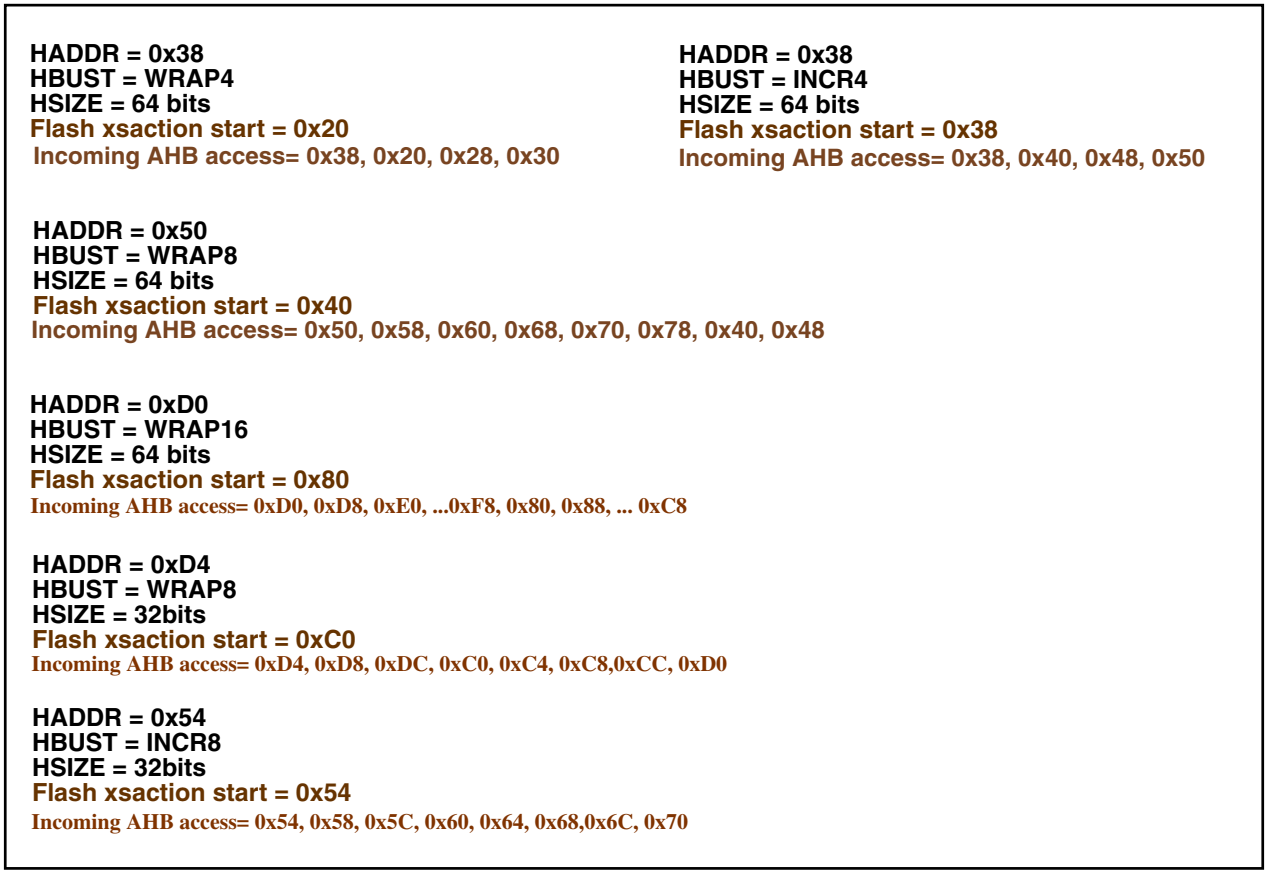



Figure 432. QuadSPI HBURST support

**NOTE**

The software must take care that the prefetch size should never be set less than the minimum data needed by any external interface to start processing.

**NOTE**

Whenever a core accesses QuadSPI memory with cache enabled, the prefetch size must be configured as equal or more than the cache line size; otherwise, FR[AIBSEF] error appears.

#### 74.4.2.6 LUT

The LUT consists of a number of pre-programmed sequences. Each sequence is basically a sequence of instruction-operand pairs, which when executed sequentially, generate a valid serial flash memory transaction. Each sequence can have a maximum of 10 instruction-operand pairs. The LUT can hold a maximum of sequences. The figure below shows the basic structure of the sequence in the LUT.

At reset, the index 0 of the LUT[0..4] is programmed with a basic read sequence as described in Read sequence. After reset, the complete LUT may be reprogrammed according to the chip connected on board. To protect its contents, during a code runover, the LUT might be locked, after which a write to the LUT will not be successful until it has been unlocked again. The key for locking or unlocking the LUT is 5AF05AF0h, and the associated processes are as follows:

**Locking the LUT**

1. Write the key 5AF05AF0h into the LUT Key Register (LUTKEY).
2. Write 0b01 to the LUT Lock Configuration Register (LCKCR). Note that this IPS transaction should immediately follow the above IPS transaction (no other IPS transaction can be issued). A successful write to this register locks the LUT.

**Unlocking the LUT**

1. Write the key 5AF05AF0h into the LUT Key Register (LUTKEY).
2. Write 0b10 to the LUT Lock Configuration Register (LCKCR). Note that this IPS transaction should immediately follow the above IPS transaction (no other IPS transaction can be issued in between). A successful write to this register unlocks the LUT.

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

Some example sequences are defined in Example sequences. After reset the instruction sequence 0 is populated with the default read sequence shown in the table below.

**Table 556. Read sequence**

| Instruction | Pad | Operand | Comment |
|---|---|---|---|
| CMD | 0h | 3h | Read data byte command on one pad |
| ADDR | 0h | 18h | 24 address bits to be sent on one pad |
| READ | 0h | 8h | Read 64 bits |
| JMP_ON_CS | 0h | 0h | Jump to instruction 0 (CMD) |

**NOTE**

If DLL is disabled then JMP_ON_CS or STOP instruction can be used else only STOP instruction can be used.

#### 74.4.2.7 Issuing SFM commands

Each access to the external device follows this sequence:

1. You must pre-populate the LUT with the serial flash memory command sequences that are required for the flash memory device being used.
2. The module executes the instructions in this sequence one by one. The transaction starts and the module configures the value of SR[BUSY].
3. Communication with the external serial flash memory device starts and the transaction executes.
4. After the transaction is complete (all transmit and receive operations with the external serial flash memory device are complete), the module resets SR[BUSY]. In case of an IP command, FR[TFF] is asserted.

For details, see Flash memory programming and Flash memory read.

You can trigger the processing of SFM commands in the QuadSPI module in one of the following ways:

**Using IP commands**

For IP commands, the required components need to be written into the following registers and in this sequence:

1. Write the serial flash memory address to be used as provided in the Serial Flash Address Register (SFAR). For IP commands not related to specific addresses, the base address of the related flash memory needs to be programmed. For example, for an instruction which does not require an address (that is, write enable instruction), the SFAR should be programmed with the base address of the memory the command is to be sent to.
2. Write the sequence ID and data size details in the IP Configuration Register (IPCR).
3. Note that writing a value to IPCR[SEQID] must be the last step of the sequence. It is possible to combine all the fields of the IPCR into one single write. See IP Configuration Register (IPCR) for details.

**Using AHB commands**

Any AHB memory-mapped access is routed to one of the buffers depending on the master ID of the request. If the access is a "miss," a new serial flash memory transaction is initiated. The transaction is based on the sequence pointed to by BFGENCR[SEQID] as described in Flexible read xAHB buffers.

An AHB access is considered memory mapped when the access is to the memory-mapped serial flash memories, as described in Memory Mapped Serial Flash Data - Individual Flash mode on Flash memory A.

#### 74.4.2.8 Flash memory programming

In all NOR Flash devices memory sector to be written needs to be erased first. The programming sequence is then initiated in the following way:

1. Check that SR[BUSY] is de-asserted or the value of the BUSY field is 0, also, check that the TX buffer is empty. If you need to discard the data present in the TX buffer (SR[TXNE]) then the TX buffer must be cleared by writing 1 to MCR[CLR_TXF].
2. Program the address related to the command in SFAR.
3. Provide initial data for the program command into the circular buffer through the TBDR. At least one words of data must be written into the TX buffer up to a maximum of 32 entries.
4. Program the IPCR to trigger the command. IPCR[SEQID] should point to an index of the LUT that has the flash memory program sequence pre-programmed. Write an appropriate value to IPCR[IDATSZ] to denote the size of the write in bytes.
5. Repeat step 3, depending on the amount of data required, until all of the required data is written to the TBDR. SR[TXFULL] can be used to check if the buffer is ready to receive more data. At any time, TBSR[TRCTR] can be read to check how many words have been written into the TX buffer.

After writing to IPCR[SEQID] (see step 4), the module starts executing the programmed sequence . The software ensures that the correct sequence is programmed into the LUT in accordance with the flash memory connected to the module. The data is fetched from the TX buffer. It consists of 32 entries of 32-bit sizes and is organized as a circular FIFO, the read pointer for which is incremented after each fetch. When all the data is transmitted, the QuadSPI module returns from the busy state to the idle state. However, this is not true for the external device because the internal programming is still ongoing. You may monitor the relevant status information available from the serial flash memory device and ensure that the programming is done appropriately.

#### 74.4.2.9 Flash memory read

Host access to the data stored in the external serial flash memory device is performed in two steps. First, the data must be read into the internal buffers and in the second step, these internal buffers can be read by the host.

**Reading serial flash memory data into the QuadSPImodule internal buffers**

A read access to the external serial flash memory device can be triggered in two different ways:

- IP command read: For reading flash memory data into the RX buffer, you must provide the correct sequence ID in IPCR[SEQID]. The sequence ID points to a sequence in the LUT. The software needs to ensure that a correct read sequence is programmed in the LUT in accordance with the serial flash memory device connected on board. You must program the SFAR , and IPCRs. All available read commands supported by the external serial flash memory are possible.

  Optionally, it is possible to clear the RX buffer pointer prior to triggering the IP command by writing a 1 to MCR[CLR_RXF]. This will invalidate the data currently present in the RX buffer and any new read data will overwrite the old one.

  Using these inputs, the complete transaction is built when IPCR[SEQID] is written to The transaction related to the read access starts and the requested number of bytes is fetched from the external serial flash memory device into the RX buffer. As the read access is triggered by an IP command, the value of both SR[IP_ACC] and SR[BUSY] is set to 1.. A count of the number of entries currently in the Rx buffer can be obtained from RBSR[RDBFL].

  Communication with the external serial flash memory stops if the specified number of bytes are read (on successful completion of the transaction).

- AHB command read: For reading flash memory data into the AHB buffer, you must:
  - Set up a read access by a master to the address range in the system memory map, which the external serial flash memory devices are mapped to.

—

— Program the buffer registers corresponding to the AHB master initiating the request.

— Provide the correct sequence ID in the BFGENCR. The software ensures that a correct read sequence is programmed in the LUT in accordance with the serial flash memory device connected on board. Flash memory device selection and access mode are determined by the address accessed in the AHB address space associated with the QuadSPI module (see Memory-mapped serial flash memory data—individual flash memory mode on flash memory A

On each AHB read access to the memory mapped area, the valid data in the AHB buffer is checked against the address requested in the actual read. When the AHB read request cannot be served from the content of the AHB buffer, the buffer is flushed and the controller executes the sequence pointed to by the sequence ID. The requested number of buffer entries defined in BUFxCR[ADATSZ] is then fetched from the external serial flash memory device into the internal AHB buffer. As the read access is triggered through the AHB bus, the value of SR[AHB_ACC] is set, driving SR[BUSY] in turn, until the transaction is complete. Communication with the external serial flash memory stops when the specified number of entries is filled.

### Data transfer from the QuadSPI module internal buffers

The data read out from the external serial flash memory device by the QuadSPI module is stored in the internal buffers. The means of accessing the data from the buffer differs depending on which buffer the data is loaded to. See Block diagram for details on the two available buffers, the RX buffer and the AHB buffer, in this module. The buffer appears transparent to you and is non-memory mapped. See the "Flexible AHB Buffer" section for details.

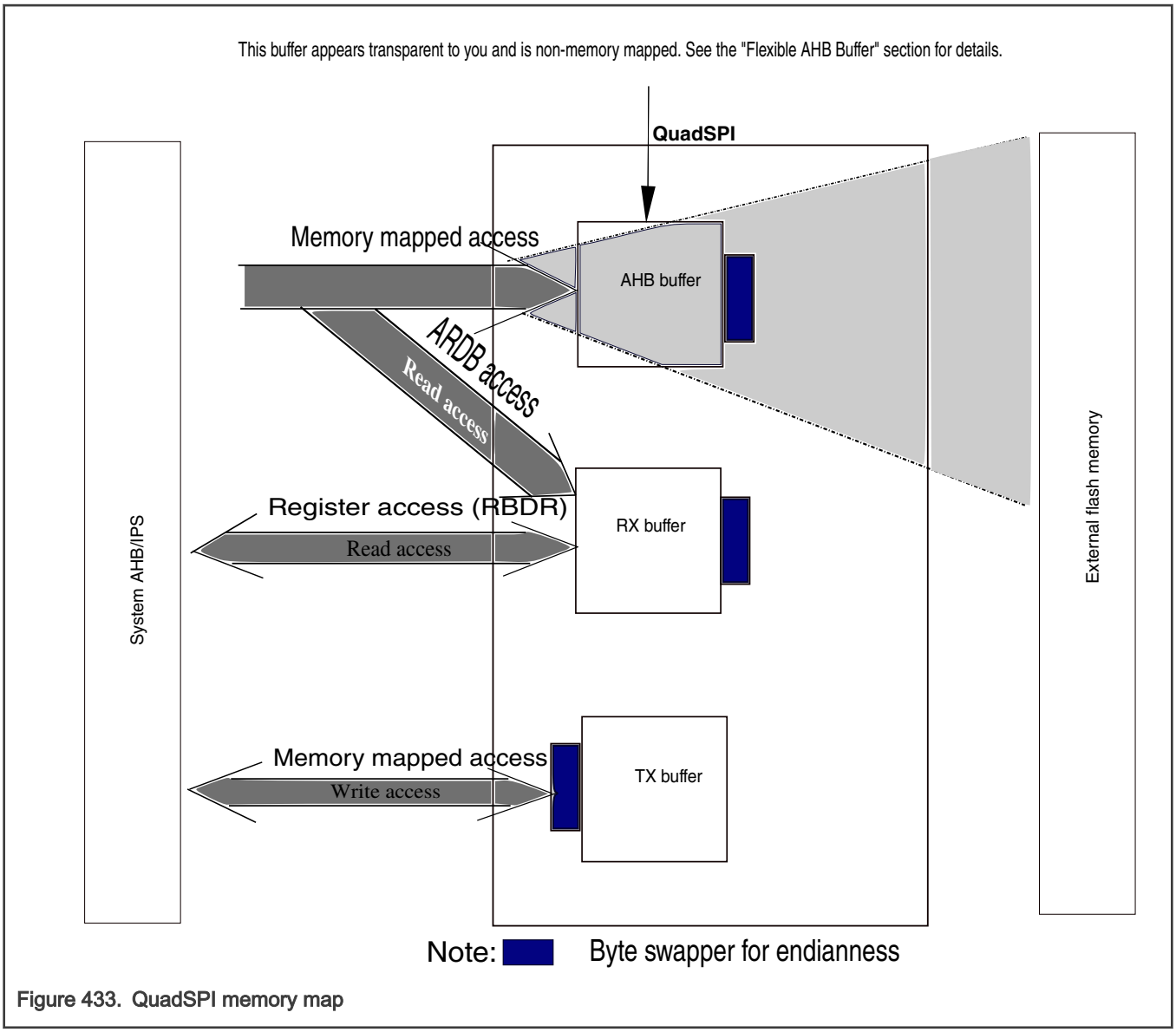



Figure 433. QuadSPI memory map

The RX buffer is implemented as FIFO of depth 32 entries of 4 bytes. Its content is accessible in two different address areas, both referring to identical data and the same physical memory:

- In the IPS address space in the area associated with RX Buffer Data Register (RBDR0 - RBDR31).
- In the AHB address space in the area associated with AHB RX Data Buffer Register (ARDB0 - ARDB127).Two successive entries are accessed with one single 64-bit AHB read operation.

The RX buffer operation can be summarized as follows:

- RBCT[WMRK] determines at which fill level SR[RXWE] is asserted and how many entries are removed from the RX buffer on each buffer POP operation.
- SR[RXWE] indicates that the configured number of data entries is available in the RX buffer and RBSR[RDBFL] indicates how many valid entries are available in total.
- The first entry (RBDR0 or ARDB0) always corresponds to the first valid entry in the RX buffer.

For details, see RX Buffer Data Register (RBDR0 - RBDR31) and AHB RX Data Buffer Register (ARDB0 - ARDB127).

- Flag-based data read of the RX buffer is performed by polling SR[RXWE]. When it is asserted, the valid entries can be read either via the IPS address space (RBDRn) or the AHB address space (ARDBn). A buffer POP operation must be

triggered by the application by writing a 1 to FR[RBDF]. This automatically updates the FIFO to point to the next entry as defined by RBCT[WMRK]. For example, if WMRK is set to 3, then the buffer discards 16 bytes of data.

- DMA-controlled data read of the RX buffer is performed by using the DMA module. The application must ensure that the DMA controller of the related chip is programmed appropriately, as described in DMA usage.
- DMA-controlled read out is triggered fully automatically by the assertion of SR[RXWE]. The related buffer POP operation is also handled completely inside the QuadSPI module. As in the case explained here, accessing the RX buffer content either on RBDRn or ARDBn related addresses is equivalent.
- AHB buffer data read via memory-mapped access: This kind of access is performed by reading one of the addresses assigned to the external serial flash memory device(s) within the range specified in Table 578. If this is not the case, a memory-mapped AHB command read is triggered as described above. If the requested data is already available in the AHB buffer, it is provided directly to the host.

When an AHB access is made to the flash memory mapped address, the data is fetched and returned to the AHB interface. The AHB interface is stalled until the data is fetched. As soon as the data from the requested address is read by the QuadSPI module, the AHB read access is served. Therefore, it is possible to run sequential reads from the AHB buffer at arbitrary speed without the need to monitor any information about the availability of the data. Nevertheless, this access scheme stalls the AHB bus for the time required to read the data from the serial flash memory device. If you know that the access is sequential, a better way is to have a prefetch enabled by programming the value of BUFxCR[ADATSZ] so that the data is fetched into the buffer before the next sequential AHB access.

As long as the host restricts its accesses to the data present in the buffer and to the data currently fetched from the serial flash memory, it is possible to run the host read from the AHB buffer simultaneously with the serial flash memory read into the AHB buffer.

#### 74.4.2.10 Byte ordering of serial flash memory read data

The basic scheme is that the first byte read out of the serial flash memory device, which is addressed by SFAR[SFADR], corresponds to RBDR0[31:24] for IP command read. Similarly, to send a single byte it should be positioned in TBDR[0:7]. In contrast to that for AHB command read, the bytes are always positioned according to the byte ordering of the AHB bus.

- Byte ordering in individual flash memory mode

  The following table provides the byte ordering scheme of how the byte oriented data space of the serial flash memory device is mapped into one single 32-bit entry of the RX buffer or the AHB buffer. The table is valid within the following context:

  — Flash memory A in individual flash memory mode

  — All AHB data read commands with 32-bit access size

**Table 557. Byte ordering in individual flash memory mode**

| Serial flash memory byte numbering | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| **Buffer entry bit position [31:0] (32-bit data width)** | [31:24] | [23:16] | [15:8] | [7:0] |

**NOTE**

For IP commands, the read size can be specified as number of bytes. If this number is not a multiple of four, then the last buffer entry is not completely filled with the missing higher numbered bytes at undefined values.

  For AHB commands and reads, starting from an address not aligned to 32-bit boundaries, the requested bytes are given at the appropriate positions according to the AMBA AHB specification.

- Buffer entry ordering for 64-bit read access

  For read access via the AHB interface, a 64-bit access is possible. Each 64-bit access reads two 32-bit entries, simultaneously. The ordering of these 32-bit entries within the 64-bit word is provided in the following table.

Table 558. 64-bit read access buffer entry ordering

| AHB read data bit position [63:0] | [63:32] | [31:0] |
|---|---|---|
| Buffer entry # | Odd (1, 3, 5, ...) | Even (0, 2, 4, ...) |

#### 74.4.2.11 Normal mode interrupt and DMA requests

The QuadSPI module has different flags that can only generate interrupt requests and one flag that can generate an interrupt as well as DMA requests. The following table lists the eight conditions. Note that the flags mentioned in the table are associated with the Flag Register (FR).

Table 559. Interrupt and DMA request conditions

| Condition | Flag (FR) | DMA |
|---|---|---|
| TX buffer fill | TBFF | - |
| TX buffer underrun | TBUF | - |
| Illegal instruction error | ILLINE | - |
| RX buffer drain | RBDF | X |
| RX buffer overflow | RBOF | - |
| AHB buffer overflow | ABOF | - |
| AHB sequence error | ABSEF | - |
| AHB illegal transaction error | AITEF | - |
| AHB illegal burst size error | AIBSEF | - |
| IP command trigger during AHB access error | IPAEF | - |
| IP command trigger could not be executed error | IPIEF | - |
| IP command related transaction finished | TFF | - |

Each condition has a corresponding field in Flag Register (FR) and a request enable field in DMA Request Select and Enable Register (RSER). FR[RBDF] has separate enable fields for generating IRQ and DMA requests. Note that not all the fields have an individual IRQ line. See the chip's Interrupt Vector table for details.

- Transmit buffer fill interrupt request

  This indicates that the TX buffer can accept new data. The buffer is asserted if FR[TBFF] is asserted and if the value of the corresponding enable field, RSER[TBFIE], is 1. See TX buffer Operation for details on the assertion of FR[TBFF].

  Apart from IRQ, it is possible to handle the TX buffer fill by using the DMA. If the value of RSER[TBFDE] is 1, a DMA request is triggered when the number of available space in the TX buffer is more than the TBCT[WMRK] valid entries and value of SR[TXWA] is set. The application must configure the environment appropriately (for example, the DMA controller) for the DMA transfer.

- Receive buffer drain interrupt or DMA request

  This is derived from FR[RBDF], indicating that the RX buffer of the QuadSPI module has data available from the serial flash memory device to be read by the host. It remains set as long as RBSR[RXWE] is configured. Also, RSER[RBDIE] enables the related IRQ.

Apart from the IRQ, it is possible to handle the RX buffer drain by using the DMA. If the value of RSER[RBDDE] is 1, a DMA request is triggered when the RX buffer contains more than RBCT[WMRK] valid entries. The application must set the environment appropriately (for example, the DMA controller) for the DMA transfers.

- Buffer overflow/underrun interrupt request

  This is a combination of the following fields (all located in the Flag Register (FR) with the related enable bits in the DMA Request Select and Enable Register (RSER)):

  — TBUF - TX buffer underrun, enabled by TBUIE

  — RBOF - RX buffer overflow, enabled by RBOIE

  — ABOF - AHB buffer overflow, enabled by ABOIE

  — The transmit buffer underrun indicates that an underrun condition in the TX buffer has occurred. It is generated when a write instruction is triggered whilst the TX buffer is empty and the value of RSER[TBUIE] is 1.

  — The receive buffer overflow indicates that an overflow condition in the RX buffer has occurred. It is generated when the RX buffer is full, an additional read transfer attempts to write into the RX buffer, and the value of RSER[RBOIE] is 1.

  — The AHB buffer overflow indicates that an overflow condition in the AHB buffer has occurred. It is generated when the AHB buffer is full, an additional read transfer attempts to write into the AHB buffer and the value of RSER[ABOIE] is 1.

  — The data from the transfers that generated the individual overflow conditions is ignored.

- Serial flash memory command error interrupt request

  If the IPAEF, IPIEF fields in the FR are set, and the related interrupt enable bits in the RSER are also set, then an interrupt is requested.

- Transaction finished interrupt request

  The IP command transaction finished IRQ indicates the completion of the current IP command. It is triggered by FR[TFF] and is masked by RSER[TFIE].

#### 74.4.2.12 TX buffer operation

The TX buffer provides the data used for page programming. For proper operation, it is required to provide at least one entry in the TX buffer prior to starting the execution of the page programming command. The application must ensure that the required number of data bytes is written into the TX buffer fast enough as long as the command is executed without a TX buffer overflow or underrun.

The QuadSPI module sets the FR[TBFF] field as long as the TX buffer is not full and can accept more data. At the end of write through TX buffer, you must clear FR[TBFF] to avoid unnecessary last TX buffer fill interrupt. However, there would always be a pending request asserted from QuadSPI controller at the end of any DMA transfer. When external DMA finishes transfer iteration, this request from QuadSPI is kept asserted for the next iteration loop.

**NOTE**

Even if the generation of DMA requests for filling the TX buffer is disabled by using RSER[TBFDE], the TX buffer still accepts a DMA transfer because of the last asserted pending request.

Disabling of DMA transfer should be controlled by an external DMA master.

When the QuadSPI module tries to pull data out of an empty TX buffer, FR[TBUB] signals the TX buffer underrun. The TX buffer underrun flag is also asserted when the TX buffer contains less than 32-bits of data and the QuadSPI module tries to pull out data from it. The current IP command leading to the underrun condition is continued until the specified number of bytes is sent to the serial flash memory device. Also, the data that is transferred is in the Fs[10] format. This means, after the underrun flag is set under this condition, it returns Fs until the required number of bytes are not sent. This has been done to ensure that the software does not erase the whole sector after underrun and just reprogramming from failure point serves the purpose. When this sequence command is complete, FR[TBFF] is asserted, indicating that the TX buffer is ready to be written again.

The TX buffer overflow is not signaled explicitly, but TBSR[TRBFL] can monitor the TX buffer fill level.

[10] Fs represents a value of FFFF_FFFF.

For more information, see TX Buffer Status Register (TBSR) and Flag Register (FR).

#### 74.4.2.13 Address scheme

Earlier, serial flash memories supported only a 24-bit address space, restricting the maximum memory size of the serial flash memory to 16 MB. The new memory specification supports two types of 32-bit addressing mode in addition to the legacy 24-bit address mode.

**Extended address mode**

In this mode, the legacy 24-bit commands are converted to accept 32-bit address commands. The flash memory needs to be configured for the 32-bit address mode. Also, while programming the LUT sequence in QuadSPI for 32-bit mode, the ADDR commands should be programmed with 32d as the operand value . By default, QuadSPI is in 24-bit legacy address mode. Each of the memory vendors have a different way of enabling this mode (see the memory specification from memory vendors). For example, the command B7h sent to the Macronix flash memory enables it for the 32-bit address mode.

**Extended address register**

In this mode, the upper 8 bits of the 32-bit register are provided by the Extended address register in the memory, which provides a specific register that is updated according to the address to be accessed. This effectively converts the legacy 24-bit address command into 32-bit address commands. The memories greater in size than 16 MB consist of banks of 16 MB each. The 8 bits occupied in the extended address register effectively enable a bank. For example, in Spansion memory, when the extended address register is updated with a value of 1h, with the help of the 17h command, it opens Bank1 of the memory. The consequent 24-bit address commands lead to Bank1. The extended address register needs to be updated with the respective value for access to other banks. This effectively converts the legacy 24-bit address command into 32-bit address commands.

### 74.4.3 Module Disable mode

Module Disable mode is a block-specific mode that the QuadSPI can enter to disable serial flash memory clock and AHB command. This mode can be entered by:

- The host software: by writing a '1' to MCR[MDIS]

Below are the condition that must be fulfilled to enter the Module Disable mode:

- SR[BUSY] = 0
- SR[AHBTRN] = 0
- RBSR[RDBFL] = 0
- SR[RXDMA] = 0
- SR[TXDMA] = 0
- None of the flags in FR are enabled as interrupts is set

The conditions mentioned above ensures the following:

- There is no SFM command currently being executed.
- All the data read into the RX buffer from the serial flash memory have been fetched by the application.
- There is no current AHB access.
- There is no active DMA request.
- There is no enabled interrupt that is pending.

Certain read or write operations have a different effect when the QuadSPI is in the Module Disable mode. In the Module Disable mode, not all of the status and flag bits of the QuadSPI module are updated, and writing to them has no effect. Interrupt and DMA request signals cannot be cleared while in the Module Disable mode.

**NOTE**

It is illegal to issue a new SFM command starting two clock cycles prior to raising the request of entering the Module Disable mode until the QuadSPI stays in this mode.

### 74.4.4 Leaving Module Disable mode

In the Module Disable mode, the serial flash memory clock and AHB command to the QuadSPI module are switched off.

After the QuadSPI has left this mode and has returned to Normal mode, the execution of the first SFM command is deferred until the clock to drive that part of the module related to the serial flash memory device is available. Depending upon the point in time when the first SFM command is triggered, the actual execution of the command starts with a delay, respective with the re-enabling of the flash memory clock signal.

## 74.5 Initialization/application information

This section provides the initialization and application information of the QuadSPI module.

### 74.5.1 Power up and reset

The serial flash memory devices connected to the QuadSPI module might require special voltage characteristics of their inputs during power up or reset. The application must ensure this.

**CAUTION**

Erase or program commands should be completed before issuing a reset or power cycle to avoid corrupted flash pages. The application shall ensure there is a backup of critical data stored at a different location to enable recovery from corrupted flash pages.

**Example: Flash reset sequence**

Use the following sequence to reset flash A:

1. Make sure that the flash supports a reset for the condition CS#=high and IOF[3]=low.
2. Set MCR[SWRSTSD] and MCR[SWRSTHD] fields and then clear them.
3. Set MCR[MDIS] field.
4. Reset MCR[ISD3FA] field for flash A.
5. Clear MCR[MDIS] field.
6. Set MCR[MDIS] field.
7. Set MCR[ISD3FA] field for flash A.
8. Clear MCR[MDIS] field.

### 74.5.2 Available status/flag information

This section provides an overview of the different flags and statuses available, and their interdependencies for different use cases. The SR and FR are the related registers.

#### 74.5.2.1 IP commands

See IP Configuration Register (IPCR) for additional details not explicitly covered in this paragraph.

- IP commands—normal operation

  Writing to IPCR[SEQID] triggers the execution of a new IP command. Given that this is a legal command, SR[IPACC] and SR[BUSY] are asserted simultaneously, immediately after the execution starts.

  After the instruction on the serial flash memory device is complete, these field deassert and FR[TFF] is configured.

- IP commands—error situations

See Overview_of_Error_Flags for details.

#### 74.5.2.2 AHB commands

See the "Reading serial flash memory data into the QuadSPI module internal buffers" topic in theFlash Memory Read section for details.

- AHB commands—normal operation

  Memory-mapped read access to a serial flash memory address not contained in the AHB buffer triggers the execution of an AHB command. Given that this is a legal command, SR[AHB_ACC] and SR[BUSY] are asserted simultaneously, immediately after the execution starts. After the instruction on the serial flash memory device is complete, these fields are deasserted.

- IP commands—error situations

  See Overview of FR error flags for details.

#### 74.5.2.3 SFM commands

An SFM command consists of an instruction code and all other parameters (for example, size or mode bytes) needed for that specific instruction code. Triggering a command either initiates a transaction on the external serial flash memory or results in an error. See Table 560 for details on errors.

#### 74.5.2.4 Overview of error flags

The following table provides an overview of the different error flags in the FR and additional error-related details.

**Table 560. Overview of FR error flags**

| Error category | Error flag in FR | Command execution on serial flash memory device<br>TFF behavior (in case of IP commands only) | Description |
|---|---|---|---|
| AHB error flag | ABOF | Flash memory transaction continues until it finishes | Set when the module tries to push data into the AHB buffer that exceeds the size of the AHB buffer. Occurs only because of the wrong programming of BUFxCR[ADATSZ]. |
| AHB error flag | AIBSEF | Flash memory transaction is aborted | Total burst size of the AHB transaction is greater than prefetch data size. |
| AHB error flag | AITEF | Flash memory transaction is aborted | No response is generated from QuadSPI to AHB bus in case of illegal transaction. Also, the watchdog timer expires. |
| Miscellaneous error flag | ILLINE | Flash memory transaction aborted | Illegal instruction error flag - set when an illegal instruction is encountered by the controller in any of the sequences. |
| Command arbitration error | IPIEF | TFF not asserted in conjunction with that command | IP command error - caused when IP access is currently in progress (IP_ACC is set) and during: |

*Table continues on the next page...*

Table 560. Overview of FR error flags (continued)

| Error category | Error flag in FR | Command execution on serial flash memory device<br>TFF behavior (in case of IP commands only) | Description |
|---|---|---|---|
| | | | • Write attempt to IPCR register<br>• Write attempt to SFAR register<br>• Write attempt to RBCT register |
| Command arbitration error | IPAEF | TFF not asserted in conjunction with that command | • AHB command already running, another IP command could not be executed<br>• AHB command already running, write attempt to IPCR[SEQID] |
| Buffer-related error | RBOF | TFF is asserted on completion | • RX buffer overrun |
| Buffer-related error | TBUF | TFF is asserted on completion | • TX buffer underrun |

Note that only the buffer-related errors are associated with a transaction on the external serial flash memory. All the other errors do not trigger an actual transaction.

#### 74.5.2.5 IP bus and AHB access command collisions

There are following flags related to this topic: FR[IPAEF] and FR[IPIEF]. See the "Reading serial flash memory data into the QuadSPI module internal buffers" topic of the Flash Memory read section for a description of the flags.

### 74.5.3 Flash memory device selection

Regardless of the SFM command (IP or AHB), the access mode is selected by specifying the 32-bit address value for the following SFM command.

For IP commands, the access mode is selected with the address programmed into the SFAR register. See Serial Flash Address Register (SFAR) for details.

For AHB commands, the access mode is determined by the memory-mapped address. See AMBA Bus Register Memory Map for details.

### 74.5.4 DMA usage

For a complete description of the DMA module, see the related DMA Controller chapter. This section only provides QuadSPI-specific DMA usage details.

#### 74.5.4.1 DMA usage in normal mode

##### 74.5.4.1.1 Bandwidth considerations

Careful consideration of the throughput rate of the entire chain (serial flash memory -> AHB bus / IP bus -> DMA controller) involved in the read/write data process is essential for a proper operation. Such analysis must take into account not only the data rate provided by the serial flash memory but also the data rate of the AHB bus and the performance of the DMA controller in reading/writing data from/to the RX/TX buffer.

Two figures must match for a proper operation, which means that the data rate provided by the serial flash memory device must not exceed the average RX buffer readout data rate. Otherwise, the longer this state persists, it results into an RX buffer overflow. Similarly, the data consumed by the serial flash memory device must not exceed the average TX buffer fill rate. If this persists, it leads to an underrun.

#### AHB bus side (data read)

The total number of bus cycles for each DMA minor loop completion is added from the following components:

The following table provides certain examples for typical use cases:

Case 1: DMA needs to read 4 bytes from SRAM and provide to QuadSPI. It costs total four bus clock cycles. Then, DMA handshake adds additional six bus clock cycles, resulting in a total of [6 + 4 * (32/4) = 38] bus clock cycles.

**Table 561. Access duration examples for bus clock side**

| TBCT[WMRK] | Number of bytes per DMA loop | Number of bus clock cycles for DMA minor loop | Time duration of DMA minor loop for 60 MHz bus clock frequency |
|---|---|---|---|
| 3 | 16 | 6+(16/4)*4 = 22 | ~366ns |
| 7 | 32 | 6+(32/4)*4 = 38 | ~633ns |
| 11 | 48 | 6+(48/4)*4 = 54 | ~900ns |
| 15 | 64 | 6+(64/4)*4 = 70 | ~1166ns |

Case 2: DMA needs to read 32 bytes from SRAM and provide to QuadSPI. DMA handshake takes an additional six bus cycles, with 32 bytes DMA read from SRAM costs (8 + 3) core clock cycles. DMA writes 32 bytes to QuadSPI, takes 2 * (32/4) = 16 bus cycles with one additional CPU access to QuadSPI, costing two bus clock cycles. This results in a total 6 + (8+3)/2 + 2 * (32/4) +2 = 30 bus clock cycles.

**Table 562. Access duration examples for bus clock side**

| TBCT[WMRK] | Number of bytes per DMA loop > | Number of bus clock cycles for DMA minor loop | Time duration of DMA minor loop for 80 MHz bus clock frequency |
|---|---|---|---|
| 3 | 16 | 6 + (4+3) /2 + (16/4)*2 + 2 = 20 | ~333ns |
| 7 | 32 | 6 + (8+3) /2 + (32/4)*2 + 2 = 30 | ~500ns |
| 11 | 48 | 6 + (4+3)/2*3 + (48/4)*2 + 2 = 44 | ~733ns |
| 15 | 64 | 6 + (8+3) + (64/4)*2 + 2 = 51 | ~810ns |

**NOTE**

This table figure represents an ideal scenario; actual performance depends on how the chip integrates DMA and QuadSPI modules.

#### Serial flash memory device side (data read)

The number of serial flash memory cycles can be determined in the following way:

- Number of serial flash memory clock cycles is required to read 4 bytes, corresponding to one RX buffer entry (setup of command and address not considered): , eight cycles for quad mode (SDR) instructions in individual flash memory mode, and so on.
- Overhead because of clock domain crossing: one cycle

The following table lists the number of clock cycles required to read the data from the serial flash memory corresponding to the different settings of RBCT[WMRK]:

**Table 563. Access duration examples for serial flash memory side**

| RBCT[WMRK] setting | Num bytes per DMA loop [1] | Num SCKFx for 80 MHz SCKFx | | Time duration of flash memory data readout for 80 MHz SCKFx (~12.5ns period) | |
|---|---|---|---|---|---|
| | | IFM [2] quad | IFM quad DDR | IFM quad | IFM quad DDR |
| 0 | 4 | 8 | 4 | ~100ns | ~50ns |
| 1 | 8 | 16 | 8 | ~200ns | ~100ns |
| 3 | 16 | 32 | 16 | ~400ns | ~200ns |
| 7 | 32 | 64 | 32 | ~800ns | ~400ns |
| 11 | 48 | 96 | 48 | ~1200ns | ~600ns |

1. DMA loop refers to one minor loop completion that is equivalent to one major loop iteration.
2. Individual flash memory mode

**NOTE**

The table figure represents an ideal scenario; actual performance depends on how the chip integrates with DMA and QuadSPI modules.

A complementary example is when the watermark is set to be too high. In such a case, the time taken by the DMA to read out the RX buffer entries should be lesser than the time taken by the controller to push in the remaining entries in the buffer.

**IPS bus side (data write)**

The total number of bus cycles for each DMA minor loop completion are added from the following components:

- Overhead for each minor loop, given by DMA controller: assume 10 cycles
- Overhead because to clock domain crossing: assume two cycles
- Number of bus clock cycles required for 16 bytes (128-bit write size): assume four cycles (read/write sequence of DMA controller)

Note that the size of the minor loop is determined by the size of TBCT[WMRK]; therefore, the overhead specified above distributes among (TBCT[WMRK]+1) write accesses of 32-bit each.

The following table provides some examples for typical use cases:

**Table 564. Access duration examples for bus clock side**

| TBCT[WMRK] | Number of bytes per DMA loop [1] | Number of bus clock cycles for DMA minor loop | Time duration of DMA minor loop for 80 MHz bus clock frequency |
|---|---|---|---|
| 3 | 16 | 12+4 = 16 | ~200ns |
| 7 | 32 | 12+8 = 20 | ~250ns |
| 11 | 48 | 12+12 = 24 | ~300ns |
| 15 | 64 | 12+16 = 28 | ~350ns |
| 19 | 80 | 12+20 = 32 | ~400ns |

1. DMA loop refers to one minor loop completion that is equivalent to one.

**NOTE**

The table figure represents an ideal scenario; actual performance depends on how the chip integrates with DMA and QuadSPI modules.

**Serial flash memory device side (data write)**

The number of serial flash memory cycles can be determined in the following way:

- Number of serial flash memory clock cycles required to write 16 bytes, corresponding to four TX buffer entry (setup of command and address not considered): 32 cycles for quad SDR writes in individual flash memory mode.
- Overhead due to clock domain crossing: one cycle

The following table lists the number of clock cycles required to read the data from the serial flash memory corresponding to the different settings of TBCT[WMRK]:

**Table 565. Access duration examples for serial flash memory side**

| TBCT[WMRK] setting | Num bytes per DMA loop [1] | Num SCKFx | | Time duration for consuming data at flash memory interface 100 MHz SCKFx (10 ns period)[2] | | Time for FIFO to get empty[3] | |
|---|---|---|---|---|---|---|---|
| | | IFM [4] quad | [5] single IO SDR mode | IFM quad | PFM single IO SDR | IFM quad | PFM single IO SDR |
| 3 | 16 | 32 | 64 | 320ns | 640ns | 2240ns | 4480ns |
| 7 | 32 | 64 | 128 | 640ns | 1280ns | 1920ns | 3840ns |
| 15 | 64 | 128 | 256 | 1280ns | 2560ns | 1280ns | 2560ns |
| 23 | 96 | 192 | 384 | 1920ns | 3840ns | 640ns | 1280ns |

1. DMA loop refers to one minor loop completion that is equivalent to one major loop iteration.
2. Not all flash memory devices support writes at 100 MHz. See the flash memory data sheet for the actual page program frequency supported.
3. The assumption for these timings is that the TX Fifo is full when the transaction is initiated
4. Individual flash memory mode
5. Parallel flash memory mode

**NOTE**

The tables mentioned above are only examples which must be correlated with the DMA in the system.

Considering the examples provided in the two tables above for TX FIFO, it is evident that depending on the relationship between the bus clock and serial flash memory clock frequencies, there are settings possible where the serial flash memory consumes data faster than the IPS bus can write data in TX buffer. In these cases, a TX buffer underrun situation occurs. To avoid TX buffer underrun, the data transaction size should be large enough.

### 74.5.5 Flash memory devices address mapping

QuadSPI is configured in Single mode for the supported flash memory port A

The sizes of the flash memory devices are mapped with the system memory space based on the configurations of the following registers:

- SFA1AD
- SFA2AD
- SFB1AD
- SFB2AD

The total memory region for the flash memory devices is mapped between QuadSPI_AMBA_BASE and TOP_ADDR_MEMB2 such that the corresponding CS is asserted based on SFA1AD, SFA2AD and SFB1AD register configurations.

#### 74.5.5.1 Single mode

For single mode configuration, you must write the same value to SFB1AD and SFB2AD registers that you write to the SFA2AD register.

For dual-die flash memories, the values you write to SFB1AD and SFB2AD registers corresponds to the mapped top addresses of each die.

For single-die flash memories, you must write the same value to SFA2AD register that you write to the SFA1AD register.

Following is a programming example for single mode single-die flash memory:

- QuadSPI_AMBA_BASE - 1000_0000h
- SFA1AD[TPADA1] - 2000_0000h
- SFA2AD[TPADA2] - 2000_0000h
- SFB1AD[TPADB1] - 2000_0000h
- SFB2AD[TPADB2] - 2000_0000h

The following figure illustrates the memory mapping for single mode QuadSPI configuration.

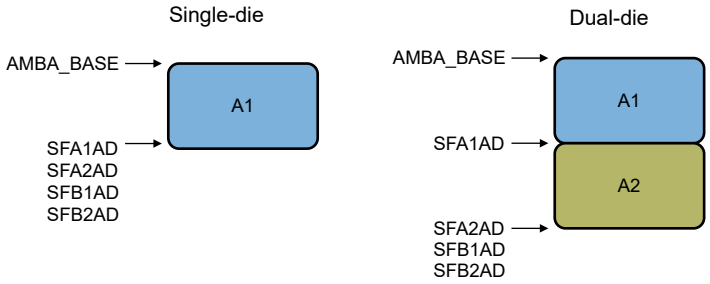



**Figure 434. Memory map for Single mode**

## 74.6 Byte ordering – endianness

The following topics show the byte ordering in 64-bit LE configuration for AHB buffer and 32-bit LE for TX/RX buffer.

### 74.6.1 Programming flash memory data

CPU writes instructions to the TBDR register, such as:

- Write TBDR: 4_03_02_01h
- Write TBDR: 8_07_06_05h

The following table shows the content against each TX buffer entry.

**Table 566. Example of QuadSPI TX buffer**

| TX buffer entry | Content |
|---|---|
| 0 | 4_03_02_01h |
| 1 | 8_07_06_05h |

Programming the TX buffer into the external serial flash memory device results in the following byte order to be sent to the serial flash memory:

- 01...02...03...04...05...06...07...08

### 74.6.2 Reading flash memory data into the RX buffer

Reading the content from the same address provides the following sequence of bytes, identical to the write case:

- 01...02...03...04...05...06...07...08

The following table shows the content against each TX buffer entry.

**Table 567. Resulting RX buffer content**

| RX buffer entry | Content |
|---|---|
| 0 | 4_03_02_01h |
| 1 | 8_07_06_05h |

#### 74.6.2.1 Readout of the RX buffer through RBDRn

The RX buffer content appears at CPU read access through the peripheral bus interface in the following order:

- Read RBDR0: 4_03_02_01h
- Read RBDR1: 8_07_06_05h

#### 74.6.2.2 Readout of the RX buffer through ARDBn

The RX buffer content appears at read access on the AMBA AHB interface at the QuadSPI module boundary:

- 32-bit access: Read ARDB0: 4_03_02_01h
- 32-bit access: Read ARDB1: 8_07_06_05h
- 64-bit access: Read ARDB0: 8_07_06_05_04_03_02_01h

### 74.6.3 Reading flash memory data into the AHB buffer

Reading the content from the same address as it was written to provides the following sequence of bytes, identical to the write case:

- 01...02...03...04...05...06...07...08

The following table shows the content against each TX buffer entry.

**Table 568. Resulting AHB buffer content**

| AHB buffer entry | Content |
|---|---|
| 0 | 8_07_06_05_04_03_02_01h |

#### 74.6.3.1 Readout of the AHB buffer through memory-mapped read

The AHB buffer content appears at read access on the AMBA AHB interface at the QuadSPI module boundary:

- 32-bit read access: 4_03_02_01h
- 32-bit read access: 8_07_06_05h
- 64-bit read access: 8_07_06_05_04_03_02_01h

## 74.7 Driving flash memory control signals in single and dual modes

In single and dual modes, the serial flash memory devices that can connect to the QuadSPI module expect additional control signals on the inputs, which are connected to IOFA[3], IOFA[2] in the quad mode. For easy interfacing, the outputs IOFA[3:2] for flash memory A are driven to the logic state given by the configuration fields MCR[ISD3FA], MCR[ISD2FA].

These outputs are driven all the time to the logic level programmed in the MCR except the time when quad commands of the serial flash memory are executed. See the specifications of the related serial flash memory device for details about the inactive level.

## 74.8 Serial flash memory devices

Several different vendors make flash memory devices with a QuadSPI interface. At present, there is no set standard for the QuadSPI instruction set. The most common commands currently have the same instruction code for all vendors; however, some commands are unique to specific vendors. Some of the example sequences are provided in the following sections.

### 74.8.1 Example sequences

This section provides the example sequences of the QuadSPI module.

**Table 569. Exit 4 x I/O read enhance performance mode (XIP) (Macronix) and read status**

| Instruction | Pad | Operand | Description |
|---|---|---|---|
| CMD | 0h | EBh | 4xIO read command |
| ADDR | 2h | 18h | 24-bit address to be sent on four pads |
| MODE | 2h | 0h | 2 mode cycles (exit XIP) |
| DUMMY | 2h | 4h | 4 dummy cycles |
| READ | 2h | 8h | Read 64 bits |
| CMD | 0h | 5h | Read Status register |
| READ | 0h | 1h | Status register data |
| STOP | 0h | 0h | Stop, instruction over |

#### 74.8.1.1 Fast read sequence (Macronix/Numonyx/Spansion/Winbond)

The following table shows the fast read sequence for Macronix/Numonyx/Spansion/Winbond flash memories.

**Table 570. Fast read sequence**

| Instruction | Pad | Operand | Description |
|---|---|---|---|
| CMD | 0h | Bh | Fast read command = 0Bh |
| ADDR | 0h | 18h | 24 address bits to be sent on one pad |
| DUMMY | 0h | 8h | Eight dummy cycles |
| READ | 0h | 4h | Read 32 bits on one pad |
| JMP_ON_CS | 0h | 0h | Jump to instruction 0 (CMD) |

**NOTE**

If DLL is disabled then JMP_ON_CS or STOP instruction can be used else only STOP instruction can be used.

#### 74.8.1.2 Fast read quad output (Winbond)

The following table shows the fast read quad output sequence for Winbond memories

**Table 571. Fast read quad output sequence**

| Instruction | Pad | Operand | Description |
|---|---|---|---|
| CMD | 0h | 6Bh | Fast read quad output command = 6Bh |
| ADDR | 0h | 18h | 24 address bits to be sent on one pad |
| DUMMY | 2h | 8h | Eight dummy cycles |
| READ | 2h | 4h | Read 32 bits on four pads |
| JMP_ON_CS | 0h | 0h | Jump to instruction 0 (CMD) |

**NOTE**

If DLL is disabled then JMP_ON_CS or STOP instruction can be used else only STOP instruction can be used.

#### 74.8.1.3 4 x I/O read enhance performance mode (XIP) (Macronix)

The following table shows the 4 x I/O read enhance performance mode for Macronix flash memories. The enhanced performance mode is also known as XIP mode.

**Table 572. Fast read quad output sequence**

| Instruction | Pad | Operand | Description |
|---|---|---|---|
| CMD | 0h | EBh | 4xI/O read command = EBh |
| ADDR | 2h | 18h | 24 address bits to be sent on four pads |
| MODE | 2h | A5h | Two mode cycles |
| DUMMY | 2h | 4h | Four dummy cycles |
| READ | 2h | 4h | Read 32 bits on four pads |
| JMP_ON_CS | 0h | 1h | Jump to instruction 1 (ADDR) |

When in XIP mode, the software must ensure that all the flash memories connected to the controller are in this mode. As a part of initializing the controller, all the flash memories might be enabled with XIP by carrying out dummy reads.

#### 74.8.1.4 Dual command page program (Numonyx)

The following table shows the dual command page program sequence for Numonyx flash memories.

**Table 573. Dual command page program sequence**

| Instruction | Pad | Operand | Description |
|---|---|---|---|
| CMD | 1h | 2h | Dual command page program = 02h on 2 pads |

*Table continues on the next page...*

Table 573. Dual command page program sequence (continued)

| Instruction | Pad | Operand | Description |
|---|---|---|---|
| ADDR | 1h | 18h | 24 address bits to be sent on two pads |
| WRITE | 1h | 20h | Write 32 bytes on two pads |
| STOP | 0h | 0h | Stop, instruction over |

#### 74.8.1.5 Sector erase (Macronix/Numonyx/Spansion)

The following table shows the Sector erase sequence for the Macronix/Numonyx/Spansion flash memories.

Table 574. Sector erase sequence

| Instruction | Pad | Operand | Description |
|---|---|---|---|
| CMD | 0h | 20h | Sector erase command = 20h |
| ADDR | 0h | 18h | 24 address bits to be sent on one pad |
| STOP | 0h | 0h | Stop, instruction over |

#### 74.8.1.6 Read status register (Macronix/Numonyx/Spansion/Winbond)

The following table shows the read status register sequence for Macronix/Numonyx/Spansion/Winbond flash memories.

Table 575. Read status register sequence

| Instruction | Pad | Operand | Description |
|---|---|---|---|
| CMD | 0h | 0h5 | Read status register command = 05h |
| READ | 0h | 0h1 | Read status register data |
| STOP | 0h | 0h | Stop, instruction over |

## 74.9 Sampling of serial flash memory input data

### 74.9.1 Basic description

QuadSPI is used to read data from the serial flash memory device. Depending on the actual implementation, there is a delay between the internal clocking in the QuadSPI module and the external serial flash memory device. See the following figure for an overview of this scheme.

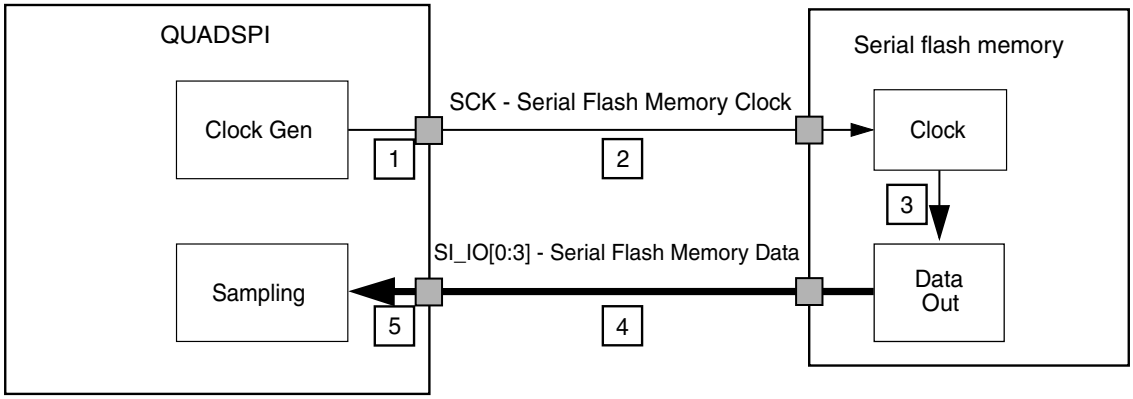



**Figure 435. Serial flash memory sampling clock overview**

The rising edge of the internal reference clock is taken as timing reference for the data output of the serial flash memory. After a time of $t_{total_delay}$ the data arrives at the internal sampling stage of the QuadSPI module. Considering the figure provided here, the following parts of the delay chain contribute to $t_{total_delay}$:

- Output delay of the serial flash memory clock output of the device containing the QuadSPI module
- Wire delay of application/PCB from the device containing the QuadSPI module to the external serial flash memory device
- Clock to data out delay of the external serial flash memory device, including input and output delays
- Wire delay of application/PCB from the external serial flash memory device to the device containing the QuadSPI module
- Device delay corresponding to the input data

**NOTE**

The $t_{total_delay}$ is specific to the characteristics of the actual implementation. Also, the serial flash memory device clock (SCK) is inverted with respect to the QuadSPI internal reference clock.

## 74.9.2 DQS sampling method

### 74.9.2.1 Basic description

In the DQS mode, the data strobe signal (DQS/RWDS) is used to sample the read data. Here, both DQS and the data sent by the flash memory move in the same direction; therefore, it is relatively easier to achieve at higher frequencies.

When using DQS for SDR reads, QuadSPI internally samples the incoming data on the rising edge of the strobe signal.

The next figure shows the sampling read data in the SDR mode using the DQS.

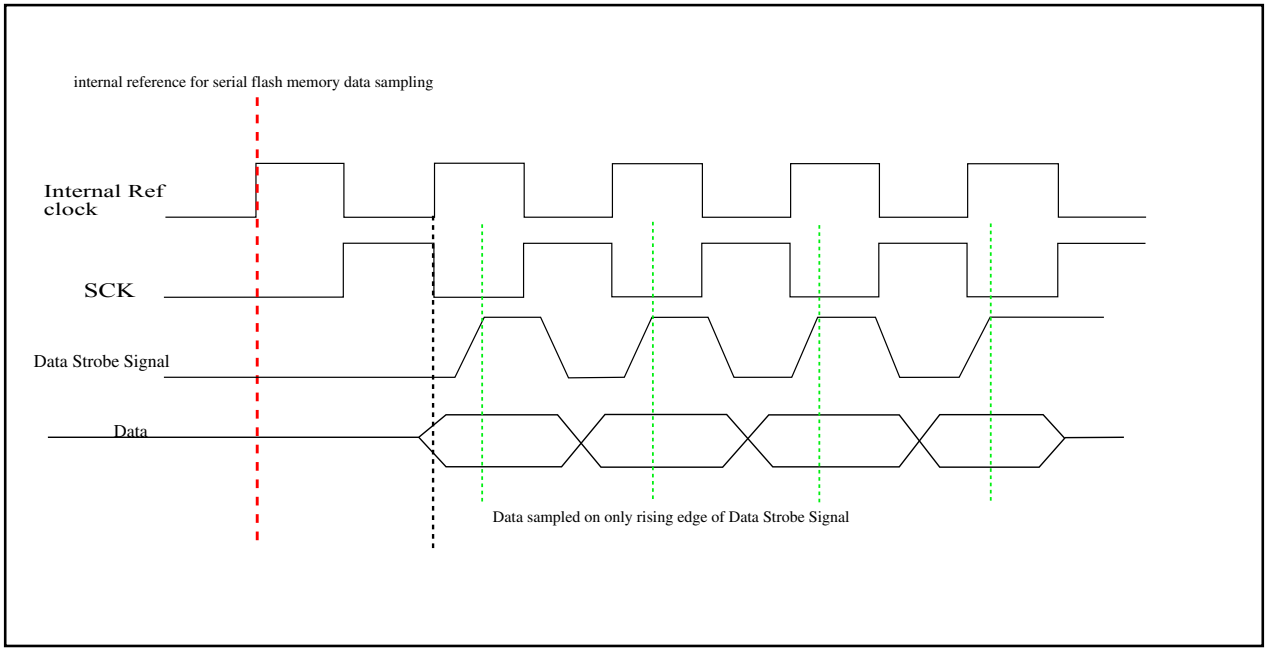



**Figure 436. Data strobe functionality in SDR mode for read operation.**

**NOTE**

Consider "Data Strobe Signal" as "Data Strobe Signal driven by memory" and "Data" as "Data from memory".

**NOTE**

Refer to the Datasheet for specific timing waveforms of QuadSPI

**NOTE**

For Specific details - Refer to the Data Sheet specification of QuadSPI module.

### 74.9.2.2 Dummy Pad loopback

The internal clock is loop-backed from the dummy internal pad to compensate data pad delays. This can be enabled by configuring the value of MCR[DQS_FA_SEL] as "01" for flash memory A. This mode can be used with the following configuration:

- High/low frequency delay chain manual programming in bypass mode using DLLCRA[SLV_DLY_COARSE] and DLLCR[FREQEN].

**NOTE**

Refer to Auto-DataLearning (4x Sampling method) section with DLL for further details

**NOTE**

This mode may not be available on the chip. See the "Supported read modes" section in the chip-specific QuadSPI information for the read modes that this chip supports.

## 74.10 Delay chain usage

**Slave delay chain programming sequence—**

Following is the programming sequence for DLL bypass mode.

1. Program DLLCRA[SLV_EN]=1, DLLCRA[SLV_DLL_BYPASS]=1, and DLLCRA[SLAVE_AUTO_UPDT]=0.

2. Program the following fields to provide the desired DQS delay for sampling: DLLCRA[SLV_FINE_OFFSET], DLLCRA[SLV_DLY_COARSE], and DLLCR[FREQEN]. See the chip-specific QuadSPI information for the supported programming settings.
3. Program DLLCRA[SLV_UPD]=1 to load these values in the slave delay chain.
4. Check the slave delay chain update status by polling DLLSR[SLVA_LOCK]=1 and clear DLLCRA[SLV_UPD] after confirming the update state.

## 74.11 Memory map and register definition

This section provides the memory map and register definitions for the QuadSPI module.

### 74.11.1 Register write access

Following are the write access restriction terms that apply to all the registers:

- Register write access restriction

  For each register field, the write access conditions are specified in the detailed register description.

  The following table provides a description of the write access conditions. If, for a specific register bit or field, none of the given write access conditions is fulfilled, any write attempt to this register bit or field is ignored without any notification. The values of the bits or fields are not changed.

  The condition term [A or B] indicates that the register or field can be written to if at least one of the conditions is fulfilled.

**Table 576. Register write access restrictions**

| Condition | Description |
|---|---|
| Anytime | No write access restriction |
| Disabled mode | Write access only if MCR[MDIS] = 1 |
| Normal mode | Write access only if the module is in the normal mode |

- Register write access requirements

  You can access all registers using 8-bit, 16-bit, and 32-bit wide operations. For some of the registers, at least a 16-bit or 32-bit wide write access is required to ensure correct operation. This write access requirement is stated in the detailed register description for each affected register.

### 74.11.2 QUADSPI register descriptions

This section provides the memory map and register definitions for the QuadSPI module.

Access to the following addresses does not result in a transfer error:

- 50h
- 64h
- 104h
- 120h
- 138h
- 168h
- 188h
- 18Ch

### 74.11.2.1 QuadSPI memory map

QuadSPI base address: 404C_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Module Configuration Register (MCR) | 32 | RW | 000F_404Ch |
| 8h | IP Configuration Register (IPCR) | 32 | RW | 0000_0000h |
| Ch | Flash Memory Configuration Register (FLSHCR) | 32 | RW | 0000_0303h |
| 10h | Buffer 0 Configuration Register (BUF0CR) | 32 | RW | 0000_0003h |
| 14h | Buffer 1 Configuration Register (BUF1CR) | 32 | RW | 0000_0002h |
| 18h | Buffer 2 Configuration Register (BUF2CR) | 32 | RW | 0000_0001h |
| 1Ch | Buffer 3 Configuration Register (BUF3CR) | 32 | RW | 8000_0000h |
| 20h | Buffer Generic Configuration Register (BFGENCR) | 32 | RW | 0000_0000h |
| 24h | SOC Configuration Register (SOCCR) | 32 | RW | 0000_0000h |
| 30h | Buffer 0 Top Index Register (BUF0IND) | 32 | RW | 0000_0000h |
| 34h | Buffer 1 Top Index Register (BUF1IND) | 32 | RW | 0000_0000h |
| 38h | Buffer 2 Top Index Register (BUF2IND) | 32 | RW | 0000_0000h |
| 60h | DLL Flash Memory A Configuration Register (DLLCRA) | 32 | RW | 0120_0000h |
| 100h | Serial Flash Memory Address Register (SFAR) | 32 | RW | 0000_0000h |
| 108h | Sampling Register (SMPR) | 32 | RW | FF00_0000h |
| 10Ch | RX Buffer Status Register (RBSR) | 32 | R | 0000_0000h |
| 110h | RX Buffer Control Register (RBCT) | 32 | RW | 0000_0000h |
| 134h | Data Learning Status Flash Memory A Register (DLSR_FA) | 32 | R | 0000_0000h |
| 150h | TX Buffer Status Register (TBSR) | 32 | R | 0000_0000h |
| 154h | TX Buffer Data Register (TBDR) | 32 | RW | 0000_0000h |
| 158h | TX Buffer Control Register (TBCT) | 32 | RW | 0000_0000h |
| 15Ch | Status Register (SR) | 32 | R | 0200_3800h |
| 160h | Flag Register (FR) | 32 | RW | 0800_0000h |
| 164h | Interrupt and DMA Request Select and Enable Register (RSER) | 32 | RW | 0000_0000h |
| 16Ch | Sequence Pointer Clear Register (SPTRCLR) | 32 | RW | 0000_0000h |
| 180h | Serial Flash Memory A1 Top Address Register (SFA1AD) | 32 | RW | 7000_0000h |
| 184h | Serial Flash Memory A2 Top Address Register (SFA2AD) | 32 | RW | 7000_0000h |
| 188h | Serial Flash Memory B1 Top Address Register (SFB1AD) | 32 | RW | 7000_0000h |
| 18Ch | Serial Flash Memory B2 Top Address Register (SFB2AD) | 32 | RW | 7000_0000h |
| 200h - 27Ch | RX Buffer Data Register (RBDR0 - RBDR31) | 32 | R | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 300h | LUT Key Register (LUTKEY) | 32 | RW | 5AF0_5AF0h |
| 304h | LUT Lock Configuration Register (LCKCR) | 32 | RW | 0000_0002h |
| 310h | LUT Register (LUT0) | 32 | RW | 0818_0403h |
| 314h | LUT Register (LUT1) | 32 | RW | 2400_1C08h |
| 318h - 35Ch | LUT Register (LUT2 - LUT19) | 32 | RW | 0000_0000h |

### 74.11.2.2 Module Configuration Register (MCR)

#### Offset

| Register | Offset |
|---|---|
| MCR | 0h |

#### Function

This register holds configuration data associated with the QuadSPI operation.

Special write-access is permitted in different modes:

- DQS_FA_SEL: Disabled mode
- ISD3FA, ISD2FA: Disabled mode
- All other fields: Anytime

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | Reserved | Reserved | | Reserved | Reserved | DQS_FA_SEL | | Reserved | Reserved | 0 | | Reserved | Reserved | ISD3FA | ISD2FA |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | MDIS | Reserved | Reserved | 0 | 0 | 0 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | | SWRS THD | SWRS TSD |
| W | | | | | CLR_TXF | CLR_RXF | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>— | Reserved |
| 29-28<br>— | Reserved |
| 27<br>— | Reserved |
| 26<br>— | Reserved |
| 25-24<br>DQS_FA_SEL | DQS clock for sampling read data at flash memory A<br>Selects DQS clock for sampling read data at flash memory A QuadSPI port<br>00b - Reserved<br>01b - Pad loopback<br>10b - Reserved<br>11b - Reserved |
| 23<br>— | Reserved |
| 22<br>— | Reserved |
| 21-20<br>— | Reserved |
| 19<br>— | Reserved |
| 18<br>— | Reserved |
| 17<br>ISD3FA | Idle signal drive IOFA[3] flash memory A<br>Determines the logic level that the IOFA[3] output of the QuadSPI module is driven to in the inactive state. See Driving flash memory control signals in single and dual modes for details.<br>0b - IOFA[3] is driven to logic L |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 1b - IOFA[3] is driven to logic H |
| 16<br>ISD2FA | Idle signal drive IOFA[2] flash memory A<br>Determines the logic level that the IOFA[2] output of the QuadSPI module is driven to in the inactive state. See Driving flash memory control signals in single and dual modes for details.<br>0b - IOFA[2] is driven to logic L.<br>1b - IOFA[2] is driven to logic H. |
| 15<br>— | Reserved |
| 14<br>MDIS | Module disable<br>Allows the clock to the non-memory mapped logic in the QuadSPI to be stopped.<br>0b - Enable QuadSPI clocks<br>1b - Allow external logic to disable QuadSPI clocks |
| 13<br>— | Reserved |
| 12<br>— | Reserved |
| 11<br>CLR_TXF | Clear TX FIFO/buffer<br>This is a self-clearing field that invalidates the TX buffer content.<br>**NOTE**<br>Software must wait for at least five system cycles and three flash cycles after writing '1' to this field.<br>0b - No action<br>1b - Read and write pointers of the TX buffer are reset to 0 and TBSR[TRCTR] is reset to 0. |
| 10<br>CLR_RXF | Clear RX FIFO<br>This is a self-clearing field that invalidates the RX buffer content.<br>0b - No action<br>1b - Read and write pointers of the RX buffer are reset to 0 and RBSR[RDBFL] is reset to 0. |
| 9<br>— | Reserved |
| 8<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 7<br>— | Reserved |
| 6<br>— | Reserved |
| 5<br>— | Reserved |
| 4<br>— | Reserved |
| 3-2<br>— | Reserved |
| 1<br>SWRSTHD | Software reset for AHB domain<br>0b - De-assert Software reset<br>1b - AHB domain flops are reset. This field does not reset configuration registers. It is advisable to reset both the serial flash memory domain and AHB domain at the same time. Resetting only one domain might lead to side effects.<br>**NOTE**<br>The software resets need the clock to be running to propagate to the design. The value of MCR[MDIS] should be 0 when the software reset bits are asserted. Also, before they can be deasserted again (by setting MCR[SWRSTHD] to 0), it is recommended to set the value of MCR[MDIS] to 1. After the software resets have been deasserted, the normal operation can be started by setting MCR[MDIS] to 0.<br>**NOTE**<br>Software must wait for at least three system cycles and three flash cycles after changing the value of this field. |
| 0<br>SWRSTSD | Software reset for serial flash memory domain<br>0b - De-assert Software reset<br>1b - Serial flash memory domain flops are reset. This field does not reset configuration registers. It is advisable to reset both the serial flash memory domain and AHB domain at the same time. Resetting only one domain might lead to side effects. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>The software resets need the clock to be running to propagate to the design. The value of MCR[MDIS] should therefore be 0 when the software reset bits are asserted. Also, before they can be deasserted again (by specifying 0 as the value for MCR[SWRSTSD]), it is recommended to specify 1 as the value for MCR[MDIS]. After the software resets are deasserted, the normal operation can be started by specifying 0 as the value for MCR[MDIS].<br>**NOTE**<br>Software must wait for at least three system cycles and three flash cycles after changing the value of this field. |

### 74.11.2.3 IP Configuration Register (IPCR)

#### Offset

| Register | Offset |
|---|---|
| IPCR | 8h |

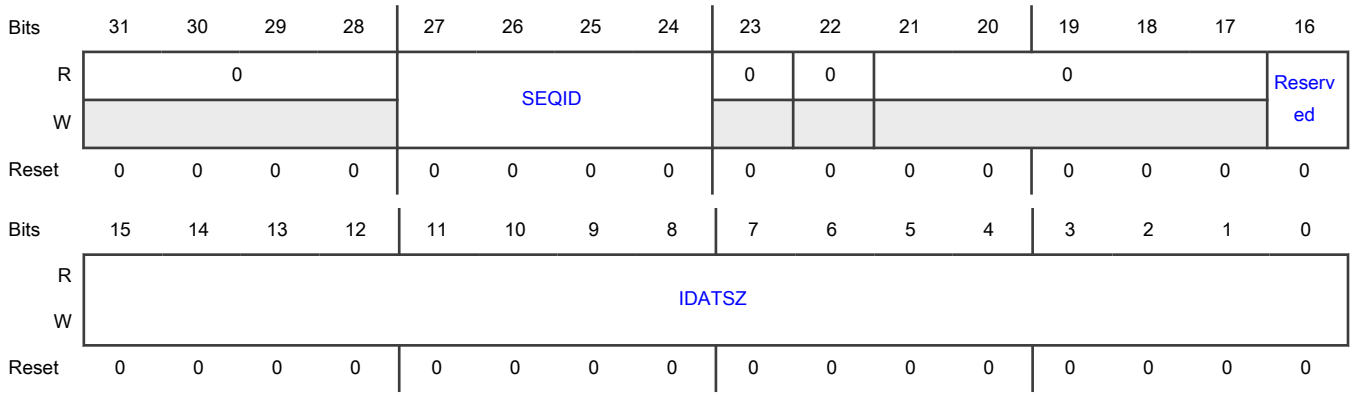
#### Function

This register provides all the configuration required for an IP-initiated command, which can be triggered by writing in the SEQID field of this register. If the SEQID field is written successfully, a new command to the external serial flash memory is initiated per the sequence pointed to by this field. See Normal mode for details on command triggering and command execution.

Special write-access is permitted if:

- SR[IP_ACC]=0

#### Diagram



**Fields**

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-24<br>SEQID | Points to a sequence in the LUT<br>This field contains the sequence index of the LUT. See LUT for details.<br>Each sequence index can accommodate up to 10 instructions (2 instructions per register).<br>A write to this field triggers a transaction on the serial flash memory interface. |
| 23<br>— | Reserved |
| 22<br>— | Reserved |
| 21-17<br>— | Reserved |
| 16<br>— | Reserved |
| 15-0<br>IDATSZ | IP data transfer size<br>This field defines the data transfer size, in bytes, of the IP command. |

### 74.11.2.4 Flash Memory Configuration Register (FLSHCR)

**Offset**

| Register | Offset |
|---|---|
| FLSHCR | Ch |

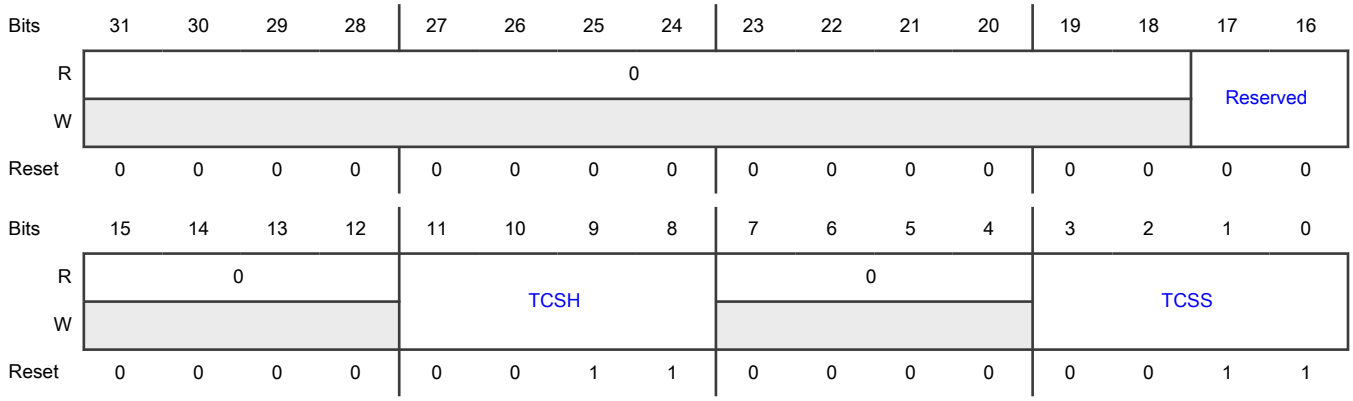
**Function**

This register contains the timings that are specific to the flash memory device. The QuadSPI controller must meet these timings for the device to function correctly.

Special write-access is permitted if:

- SR[AHB_ACC] = 0
- SR[IP_ACC] = 0

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-18<br>— | Reserved |
| 17-16<br>— | Reserved |
| 15-12<br>— | Reserved |
| 11-8<br>TCSH | Serial flash memory CS hold time<br>This hold time is in terms of serial flash memory clock cycles, and it must be greater than or equal to five flash memory clock cycles. Refer the chip datasheet for the exact value. |
| 7-4<br>— | Reserved |
| 3-0<br>TCSS | Serial flash memory CS setup time<br>This setup time is in terms of serial flash memory clock cycles, and it must be greater than or equal to two flash memory clock cycles. Refer the chip Datasheet for the exact value. |

## 74.11.2.5 Buffer 0 Configuration Register (BUF0CR)

### Offset

| Register | Offset |
|---|---|
| BUF0CR | 10h |

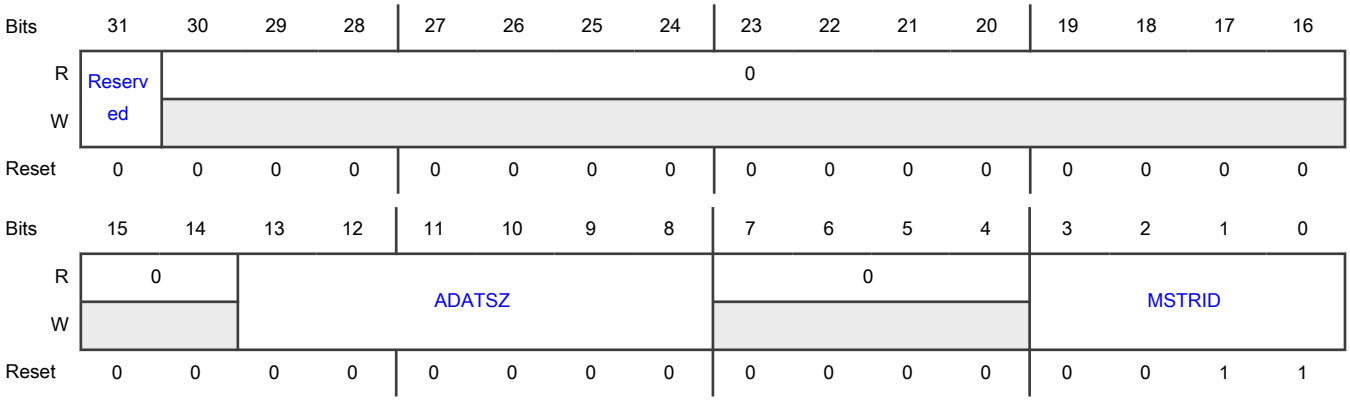
### Function

This register provides the configuration for any read access routed to buffer0, which happens when the master ID of the incoming AHB request matches BUF0CR[MSTRID]. Any buffer "miss" leads to a serial flash memory transaction being triggered per the sequence pointed to by BFGENCR[SEQID].

Special write-access is permitted if:

- SR[AHB_ACC] = 0

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-14<br>— | Reserved |
| 13-8<br>ADATSZ | AHB data transfer size<br>Defines the read data transfer size in 8 bytes of an AHB triggered read access to serial flash memory. For example, a value of 0x2 sets transfer size to 16 bytes. When ADATSZ = 0, the data size mentioned in the sequence pointed to by the SEQID field overrides this value. The software should ensure that this transfer size is not greater than the size of the buffer. |
| 7-4<br>— | Reserved |
| 3-0<br>MSTRID | Master ID<br>ID of the AHB master associated with BUFFER 0<br>Any AHB read access with this master ID is routed to this buffer. You must ensure that the master IDs associated with all buffers are different.<br>**NOTE**<br>See the chip-specific QuadSPI information for details about master IDs and their corresponding components. |

### 74.11.2.6 Buffer 1 Configuration Register (BUF1CR)

#### Offset

| Register | Offset |
|---|---|
| BUF1CR | 14h |

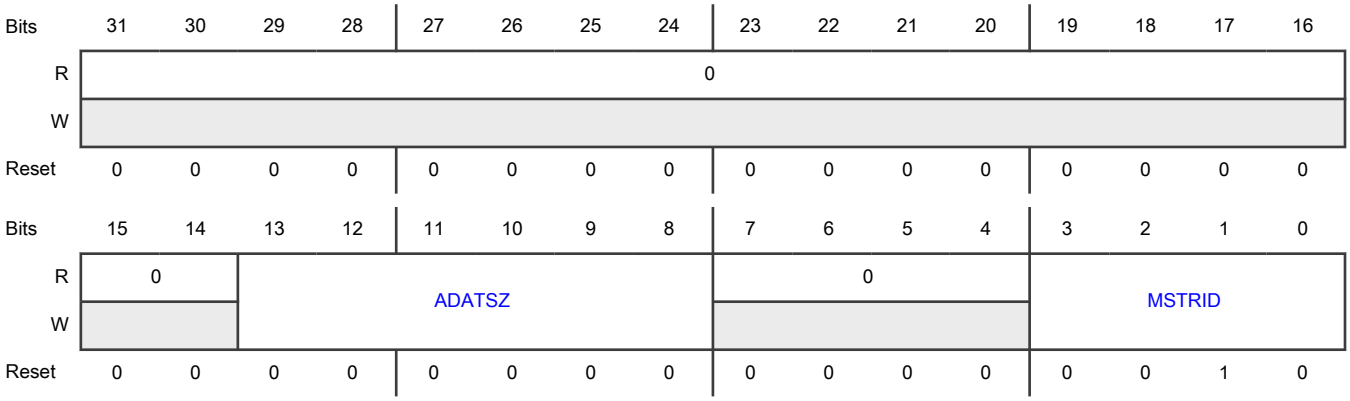
#### Function

This register provides the configuration for any access routed to buffer 1, which happens when the master ID of the incoming AHB request matches the MSTRID field of this register. Any buffer "miss" leads to the buffer being flushed and a serial flash memory transaction being triggered per the sequence pointed to by BFGENCR[SEQID].

Special write-access is permitted if:

- SR[AHB_ACC] = 0

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-14<br>— | Reserved |
| 13-8<br>ADATSZ | AHB data transfer size<br>This field defines the read data transfer size in 8 bytes of an AHB triggered read access to serial flash memory. For example, a value of 0x2 sets the transfer size to 16 bytes. When ADATSZ = 0, the data size mentioned in the sequence pointed to by the SEQID field overrides this value. Software should ensure that this transfer size is not greater than the size of this buffer. |
| 7-4<br>— | Reserved |
| 3-0<br>MSTRID | Master ID<br>ID of the AHB master associated with BUFFER 1 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Any AHB read access with this master ID is routed to this buffer. You must ensure that the master IDs associated with all buffers are different.<br>**NOTE**<br>See the chip-specific QuadSPI information for details about master IDs and their corresponding components. |

### 74.11.2.7 Buffer 2 Configuration Register (BUF2CR)

#### Offset

| Register | Offset |
|---|---|
| BUF2CR | 18h |

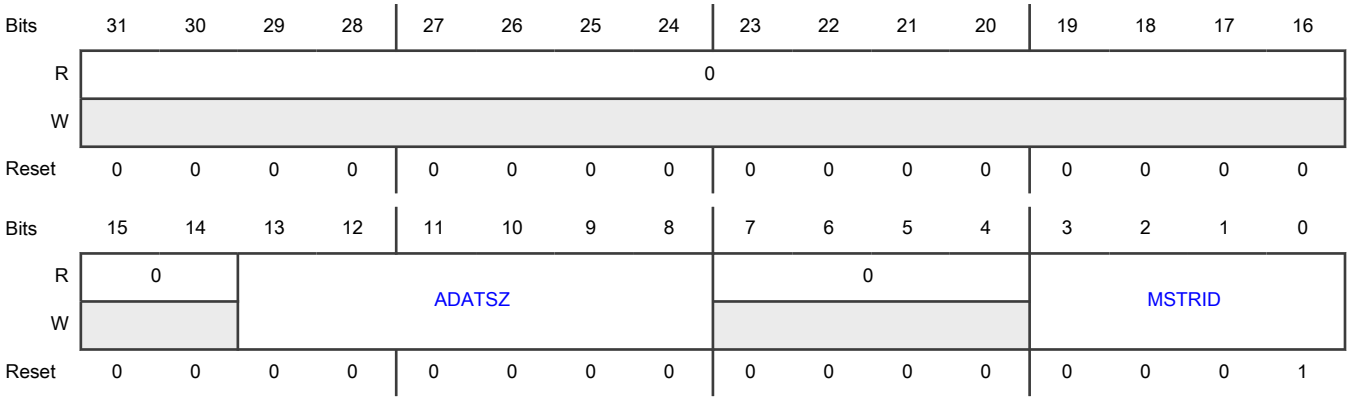
#### Function

This register provides the configuration for any access routed to buffer 2, which happens when the master ID of the incoming AHB request matches the MSTRID field of this register. Any buffer "miss" leads to the buffer being flushed and a serial flash memory transaction being triggered per the sequence pointed to by BFGENCR[SEQID].

Special write-access is permitted if:

- SR[AHB_ACC] = 0

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-14<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 13-8<br>ADATSZ | AHB data transfer size<br>This field defines the read data transfer size in 8 bytes of an AHB triggered read access to the serial flash memory. For example, a value of 0x2 sets transfer size to 16 bytes. When ADATSZ = 0, the data size mentioned in the sequence pointed to by the SEQID field overrides this value. The software should ensure that this transfer size is not greater than the size of this buffer. |
| 7-4<br>— | Reserved |
| 3-0<br>MSTRID | Master ID<br>The ID of the AHB master associated with BUFFER2. Any AHB read access with this master ID is routed to this buffer.<br>It must be ensured that the master IDs associated with all buffers are different.<br>**NOTE**<br>See the chip-specific QuadSPI information for details about master IDs and their corresponding components. |

### 74.11.2.8 Buffer 3 Configuration Register (BUF3CR)

#### Offset

| Register | Offset |
|---|---|
| BUF3CR | 1Ch |

#### Function

This register provides the configuration for any access to buffer 3.

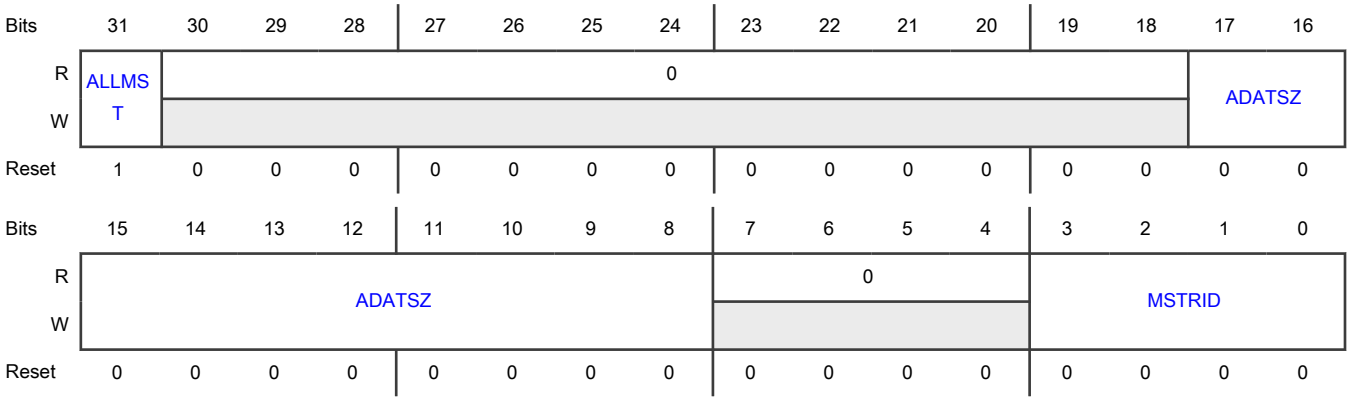
An access is routed to buffer 3 when the master ID of the incoming AHB request matches the MSTRID field of BUF3CR. Any buffer "miss" leads to the buffer being flushed and a serial flash memory transaction being triggered per the sequence pointed to by BFGENCR[SEQID].

In case the value of the ALLMST field is not 1, any such transaction (where master ID does not match any of the MSTRID fields) is returned with an ERROR response.

Special write-access is permitted if:

- SR[AHB_ACC] = 0

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>ALLMST | All master enable<br>When set, buffer3 acts as an all-master buffer. Any AHB access with a master ID not matching with the master ID of buffer0, buffer1, or buffer2 is routed to buffer3. When set, the MSTRID field of this register is ignored. |
| 30-18<br>— | Reserved |
| 17-8<br>ADATSZ | AHB data transfer size<br>Defines the read data transfer size in 8 bytes of an AHB triggered read access to serial flash memory. When ADATSZ = 0, the data size mentioned in the sequence pointed to by the SEQID field overrides this value. The software should ensure that this transfer size is not greater than the size of this buffer. If extended prefetch till 4K bytes has to be done then it can be programmed greater than the buffer size but should be always less than 512 just make sure that in this case ALLMST bit is also 1 and complete buffer is mapped to BUF3. |
| 7-4<br>— | Reserved |
| 3-0<br>MSTRID | Master ID<br>ID of the AHB master associated with BUFFER 3. Any AHB read access with this master ID is routed to this buffer. You must ensure that the master IDs associated with all buffers are different.<br>**NOTE**<br>See the chip-specific QuadSPI information for details about master IDs and their corresponding components. |

### 74.11.2.9 Buffer Generic Configuration Register (BFGENCR)

#### Offset

| Register | Offset |
|---|---|
| BFGENCR | 20h |

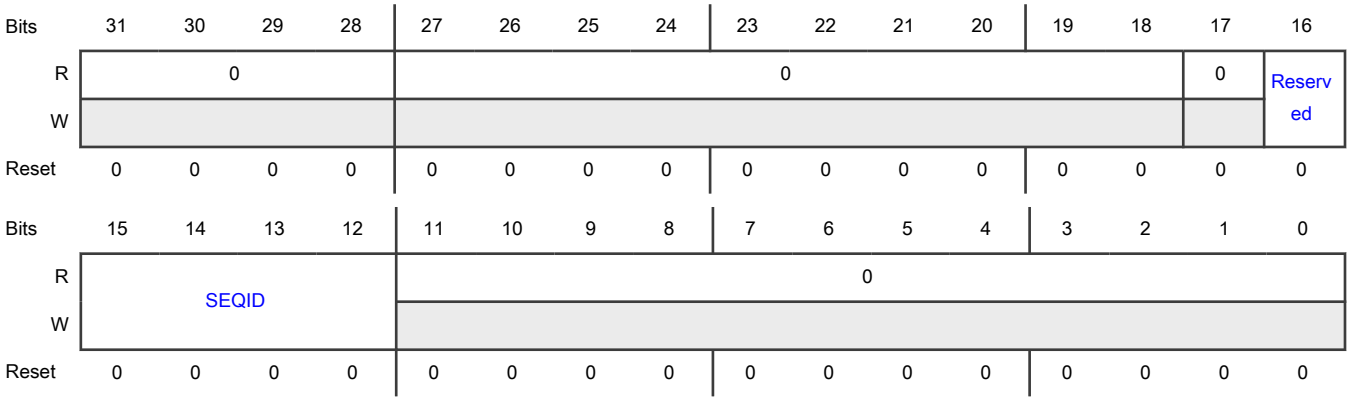
#### Function

This register provides generic configuration to any of the buffer accesses. Any buffer "miss" leads to the buffer being flushed and a serial flash memory transaction being triggered per the sequence pointed to by the SEQID field.

Special write-access is permitted if:

- SR[AHB_ACC] = 0

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-28<br>— | Reserved |
| 27-18<br>— | Reserved |
| 17<br>— | Reserved |
| 16<br>— | Reserved |
| 15-12<br>SEQID | Points to a sequence in the LUT.<br>This field contains the sequence index of the LUT... See LUT. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | **NOTE**<br>If the sequence pointer differs in the new and the previous sequences, you should reset it. See sequence pointer clear register for more information. |
| 11-0<br>— | Reserved |

### 74.11.2.10 SOC Configuration Register (SOCCR)

#### Offset

| Register | Offset |
|---|---|
| SOCCR | 24h |

#### Function

This register is programmed at the chip level for QuadSPI configuration. For details, see chip-specific QuadSPI information.

Special write-access is permitted if:

- SR[AHB_ACC] = 0

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SOCCFG | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SOCCFG | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-0<br>SOCCFG | SOC configuration<br>This field configuration is specific to chip. For details, see chip-specific QuadSPI information. |

### 74.11.2.11 Buffer 0 Top Index Register (BUF0IND)

#### Offset

| Register | Offset |
|---|---|
| BUF0IND | 30h |

#### Function

This register specifies the top index for buffer 0, which defines its size. Note that the three LSBs of this register are set to 0. This ensures that the buffer is 64-bit aligned because each buffer entry is 64-bits long.

The register value should be set to the desired number of bytes. For example, setting BUF0IND[31:3] to 0 gives 0 bytes, setting the value to 1 gives 8 bytes, and so on.

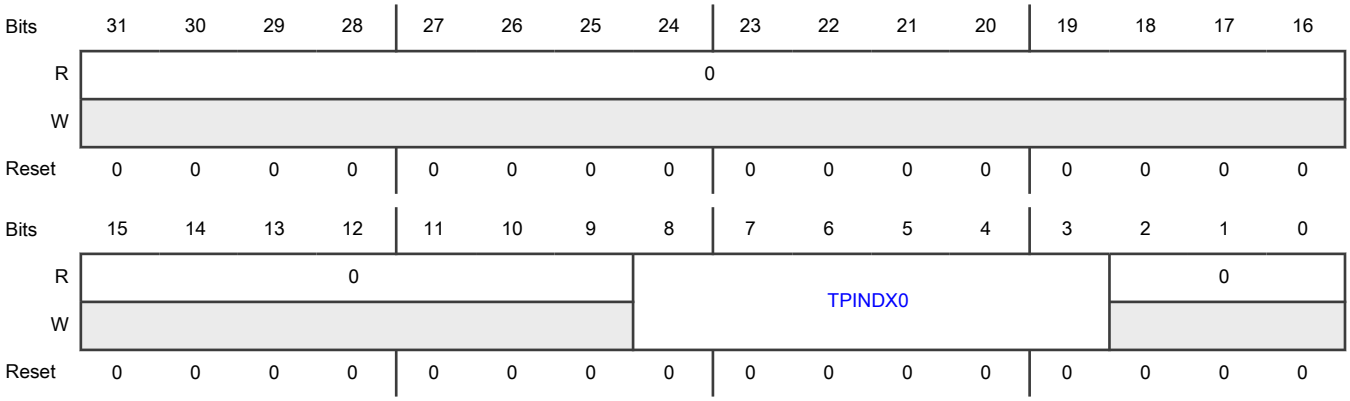
The size of buffer 0 is the difference between BUF0IND and 0.

The software must ensure that the value of TPINDX0 is not greater than the size of buffer 0.

Special write-access is permitted if:

- SR[AHB_ACC] = 0

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8-3<br>TPINDX0 | Top index of buffer 0 |
| 2-0<br>— | Reserved |

### 74.11.2.12 Buffer 1 Top Index Register (BUF1IND)

#### Offset

| Register | Offset |
|---|---|
| BUF1IND | 34h |

#### Function

This register specifies the top index of buffer 1, which defines its size. Note that the three LSBs of this register are set to 0. This ensures that the buffer is 64-bit aligned because each buffer entry is 64-bits long.

The size of buffer 1 is the difference between BUF1IND and BUF0IND. The register value should be entered in bytes. For example, if BUF0IND = 0x100, then setting BUF1IND = 0x130 sets the size of buffer 1 to 0x30 bytes.

The software must ensure that the value of TPINDX1 is not greater than the size of buffer 1.

Special write-access is permitted if:

- SR[AHB_ACC] = 0

#### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | TPINDX1 | | | | | | | | 0 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8-3<br>TPINDX1 | Top index of buffer 1 |
| 2-0<br>— | Reserved |

### 74.11.2.13 Buffer 2 Top Index Register (BUF2IND)

#### Offset

| Register | Offset |
|---|---|
| BUF2IND | 38h |

#### Function

This register specifies the top index of buffer 2, which defines its size. Note that the three LSBs of this register are set to 0. This ensures that the buffer is 64-bit aligned because each buffer entry is 64-bits long.

The size of buffer 2 is the difference between BUF2IND and BUF1IND. The register value should be entered in bytes. For example, if BUF1IND = 0x130 then setting BUF2IND = 0x180 sets the size of buffer 2 to 0x50 bytes.

The software must ensure that the value of TPINDX2 is not greater than the size of buffer 2.

Special write-access is permitted if:

- SR[AHB_ACC] = 0

#### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | | TPINDX2 | | | | | | | | 0 | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8-3<br>TPINDX2 | Top index of buffer 2 |
| 2-0<br>— | Reserved |

### 74.11.2.14 DLL Flash Memory A Configuration Register (DLLCRA)

#### Offset

| Register | Offset |
|---|---|
| DLLCRA | 60h |

#### Function

This register configures slave delay chain for flash memory A.

See Delay chain usage for the programming sequence.

**NOTE**

See the chip data sheet for information on programming register fields.

**NOTE**

Please see the chip specific section of QuadSPI DLLCRA register for delay elements data

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | FREQEN | 0 | | Reserved | | | | Reserved | | | | SLV_FINE_OFFSET | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | SLV_DLY_OFFSET | | | SLV_DLY_COARSE | | | | Reserved | | | 0 | Reserved | SLV_EN | SLV_DLL... | SLV_UPD |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>FREQEN | Frequency enable<br>0b - Selects delay chain for low frequency of operation<br>1b - Selects delay chain for high frequency of operation |
| 29-28<br>— | Reserved |
| 27-24 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 23-20<br>— | Reserved |
| 19-16<br>SLV_FINE_OFFSET | Fine offset delay elements in incoming DQS<br>This field sets the number of fine offset delay elements up to 16 in incoming DQS, and the default must be 1 element. |
| 15<br>— | Reserved |
| 14-12<br>SLV_DLY_OFFSET | T/16 offset delay elements in incoming DQS<br>This field sets the number of T/16 offset delay elements in incoming DQS; default is 0. |
| 11-8<br>SLV_DLY_COARSE | Delay elements in each delay tap<br>This field sets the number of delay elements in each delay tap. The field is used to overwrite DLL-generated delay values and works when the value of SLV_DLL_BYPASS is 1. Note : Please refer to the QuadSPI datasheet for more details. |
| 7-5<br>— | Reserved |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>SLV_EN | Slave enable<br>0b - DLL slave logic remains in reset, and its value should be 0 for at least three flash memory clock cycles for reset.<br>1b - Enables DQS slave delay chain, and should be 1 before any slave configuration settings take place. |
| 1<br>SLV_DLL_BYPASS | Slave DLL bypass<br>This field enables selection of the number of delays in each slave delay tap.<br>0b - Disables manual selection of coarse delays in the slave delay chain.<br>1b - Enables selection of number of delays in each slave delay tap, based on DLLCRA[SLV_DLY_COARSE]. |
| 0 | Slave update |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| SLV_UPD | You must program this field only after slave delay chain configuration takes place.<br>0b - Disables any further update on DQS slave delay chain.<br>1b - Updates the DQS slave delay chain with either ref-delay or bypass slave delay value, and should be set in the absence of the DQS clock. |

### 74.11.2.15 Serial Flash Memory Address Register (SFAR)

#### Offset

| Register | Offset |
|---|---|
| SFAR | 100h |

#### Function

The module automatically translates this address on the memory map to the address on the flash memory. When operating in a 24-bit mode, only bits 23-0 are sent to the flash memory. In the 32-bit mode, bits 27-0 are used with bits 31-28 driven to 0 . See Table 577 for the mapping between the access mode and the SFAR content and Normal mode for details on command triggering and command execution. The software must ensure that the serial flash memory address provided in the SFAR register lies in the valid flash memory address range, as defined in Table 577.

Special write-access is permitted if:

- SR[IP_ACC] = 0

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SFADR | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | SFADR | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-0<br>SFADR | Serial flash memory address |

### 74.11.2.16 Sampling Register (SMPR)

#### Offset

| Register | Offset |
|---|---|
| SMPR | 108h |

#### Function

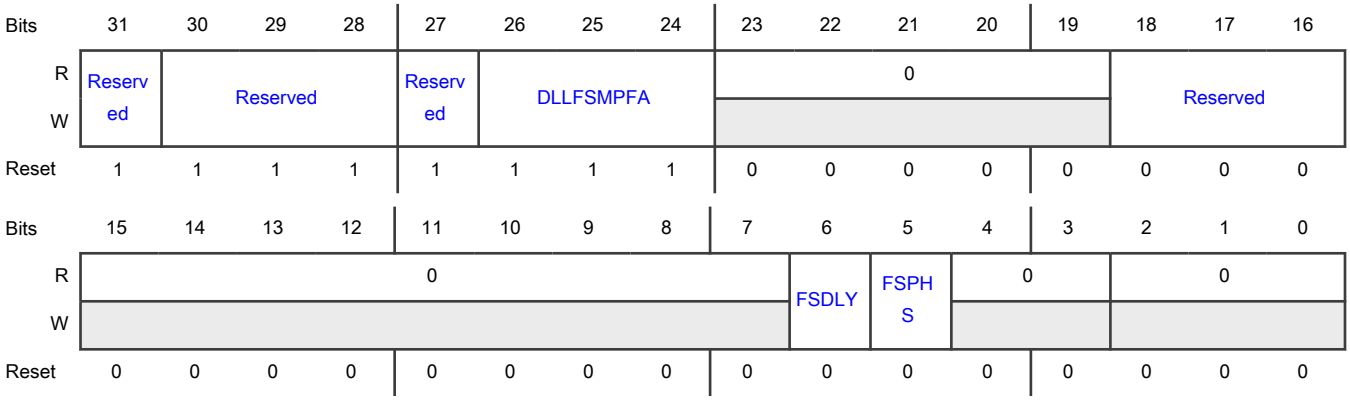
This register allows configuration of how the incoming data from the external serial flash memory devices is sampled in the QuadSPI module.

**NOTE**

See the chip data sheet for programming the register fields.

Special write-access is permitted in the disabled mode.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-28<br>— | Reserved |
| 27<br>— | Reserved |
| 26-24<br>DLLFSMPFA | Selects the nth tap provided by slave delay chain for flash memory A<br>The value of *n* can vary from 0 to 7, with each tap delay based on the DLLCRA register. |
| 23-19 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 18-16<br>— | Reserved |
| 15-7<br>— | Reserved |
| 6<br>FSDLY | Full speed delay selection for SDR instructions<br>Select the delay with respect to the reference edge for the sample point valid for full speed commands.<br>0b - One clock cycle delay<br>1b - Two clock cycles delay |
| 5<br>FSPHS | Full-speed phase selection for SDR instructions<br>This field selects the edge of the sampling clock valid for full-speed commands.<br>0b - Select sampling at non-inverted clock<br>1b - Select sampling at inverted clock |
| 4-3<br>— | Reserved |
| 2-0<br>— | Reserved |

### 74.11.2.17 RX Buffer Status Register (RBSR)

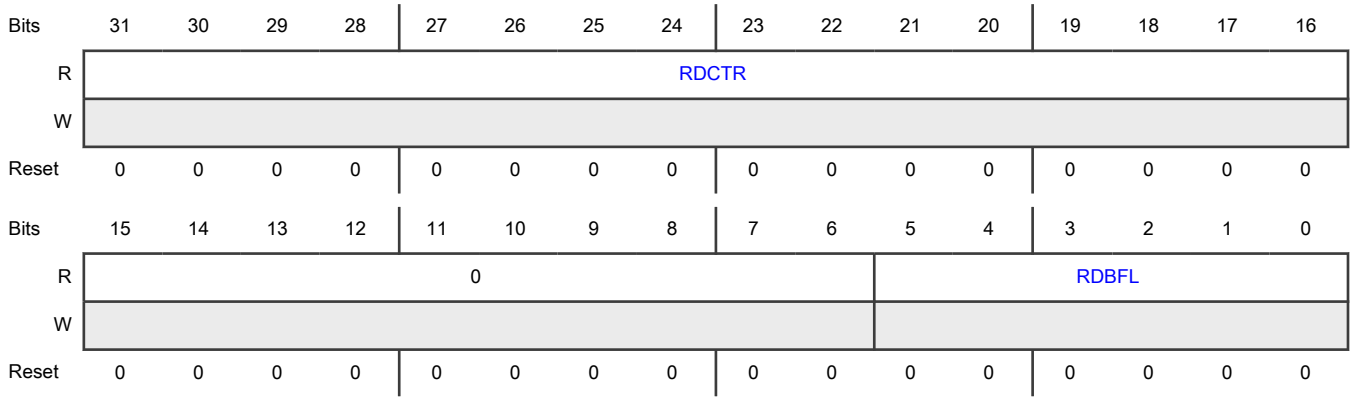
#### Offset

| Register | Offset |
|---|---|
| RBSR | 10Ch |

#### Function

This register contains information related to the receive data buffer.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>RDCTR | Read counter<br>Indicates the number of 4-byte entries removed from the RX buffer. For example, a value of 0x2 indicates that 8 bytes have been removed.<br>It is incremented by the number (RBCT[WMRK] + 1) on RX buffer POP event. The RX buffer can be popped using DMA or FR[RBDF]. The RSER[RBDDE] defines which pop should be pursued. For details, see AHB RX Data Buffer Register (ARDB0 - ARDB127) and Data Transfer from the QuadSPI Module Internal Buffers. |
| 15-6<br>— | Reserved |
| 5-0<br>RDBFL | RX buffer fill level<br>Indicates the number of 4-byte entries available in the RX buffer. For example, a value of 0x2 indicates 8 bytes are available. |

## 74.11.2.18 RX Buffer Control Register (RBCT)

### Offset

| Register | Offset |
|---|---|
| RBCT | 110h |

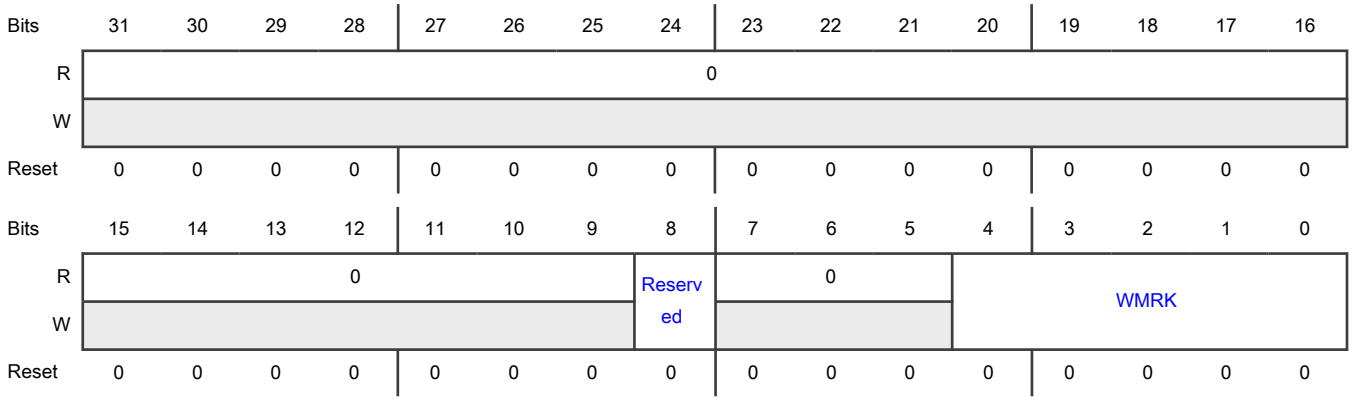
### Function

This register contains control data related to the receive data buffer.

Special write-access is permitted if:

- SR[IP_ACC] = 0

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-9<br>— | Reserved |
| 8<br>— | Reserved |
| 7-5<br>— | Reserved |
| 4-0<br>WMRK | RX buffer watermark<br>This field determines when the readout action of the RX buffer is triggered. When the number of valid entries in the RX buffer is equal to or greater than the number provided by (WMRK+1), the SR[RXWE] flag is asserted. The value should be entered as the number of 4-byte entries minus 1. For example, a value of 0x0 sets the watermark to 4 bytes, 1 to 8bytes, 2 to 12 bytes, and so on.<br>For details, see DMA usage.<br>**NOTE**<br>This field should never be programmed above 31 because there are only 32 memory mapped RBDR registers. If watermark is programmed above 31, data above 32 words will be lost. |

## 74.11.2.19 Data Learning Status Flash Memory A Register (DLSR_FA)

### Offset

| Register | Offset |
|---|---|
| DLSR_FA | 134h |

### Function

This register shows sampling point selected by data learning algorithm when the value of DLSR_FA[DLPFFA] is 0. Otherwise, it shows the pattern matching outline.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31<br>DLPFFA | Data learning pattern fail<br>This field asserts when data learning fails at flash memory A. |
| 30-16<br>— | Reserved |
| 15-8<br>POS_EDGE | DLP positive edge match signature for flash memory A |
| 7-0<br>NEG_EDGE | DLP negative edge match signature for flash memory A |

## 74.11.2.20 TX Buffer Status Register (TBSR)

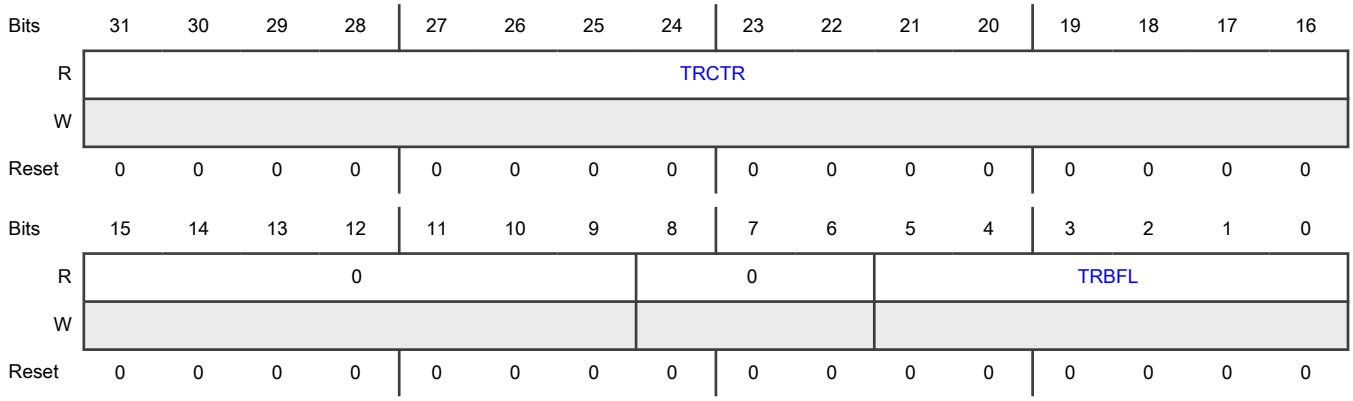
### Offset

| Register | Offset |
|---|---|
| TBSR | 150h |

### Function

This register contains information related to the transmit data buffer.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-16<br>TRCTR | Transmit counter<br>This field indicates how many entries of 4 bytes have been written into the TX buffer by host accesses. It is reset to 0 when a 1 is written to MCR[CLR_TXF]. It is incremented on each write access to the TBDR register when another word has been pushed onto the TX buffer. When it is not cleared, the TRCTR field wraps around to 0. See TX Buffer Data Register (TBDR) for details. |
| 15-9<br>— | Reserved |
| 8-6<br>— | Reserved |
| 5-0<br>TRBFL | TX buffer fill level<br>This field contains the number of entries of 4 bytes each available in the TX buffer for the QuadSPI module to transmit to the serial flash memory device. The value of this field can reach maximum up to the total TX buffer size. |

## 74.11.2.21 TX Buffer Data Register (TBDR)

### Offset

| Register | Offset |
|---|---|
| TBDR | 154h |

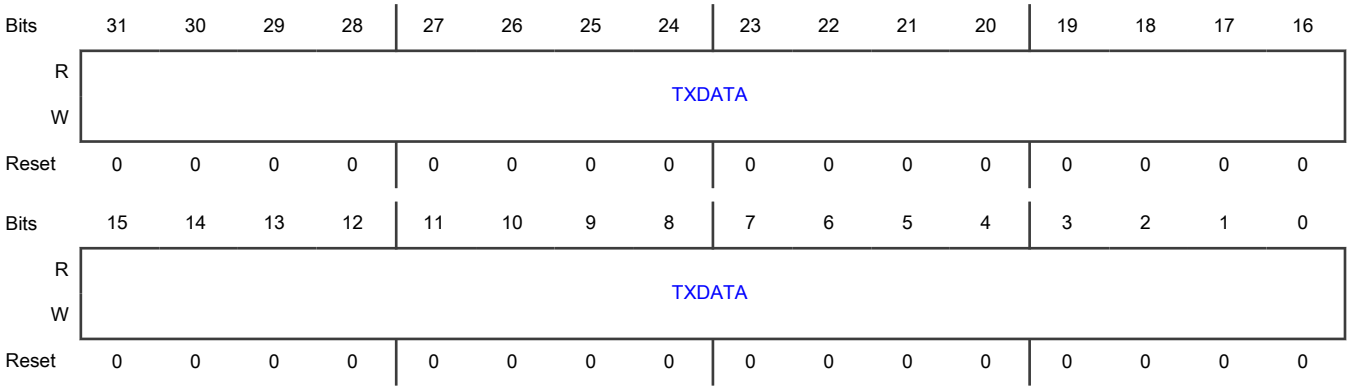
### Function

This register provides access to the circular TX buffer of depth 32 , so the total size is 32 * 4 bytes. This buffer provides the data written into it as write data for the page programming commands to the serial flash memory device. See Table 557 for the byte ordering scheme. A write transaction on the flash memory with data size of less than 32 bits leads to the removal of one data entry from the TX buffer. The valid bits are used and the rest of the bits are discarded.

Special write-access is permitted if:

- SR[TXFULL] = 0

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>TXDATA | TX data<br>On write access, the data is written to the next available entry of the TX buffer and TBSR[TRBFL] is updated accordingly.<br>On a read access, the last data written to the register is returned. |

## 74.11.2.22 TX Buffer Control Register (TBCT)

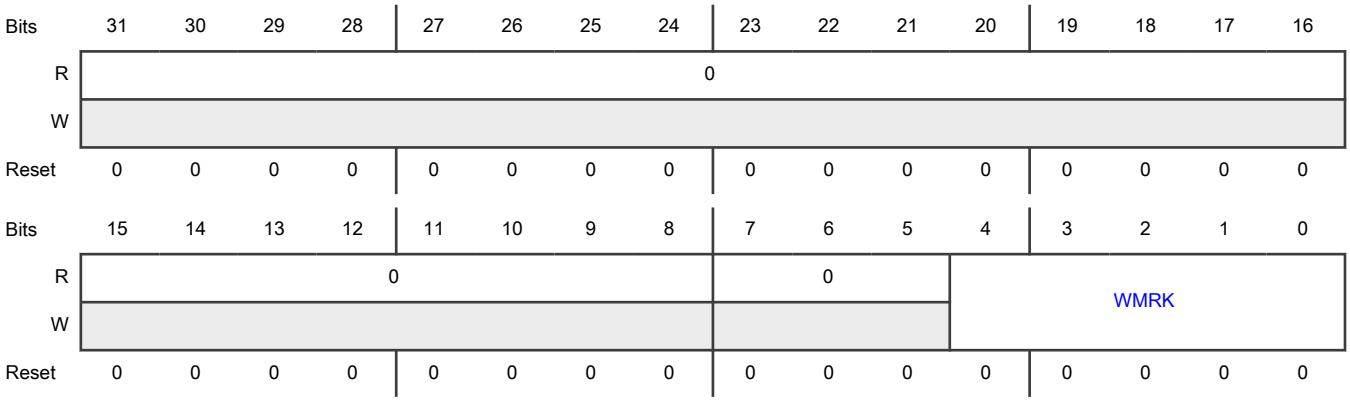
### Offset

| Register | Offset |
|---|---|
| TBCT | 158h |

### Function

This register contains control information for transmit data buffer.

### Diagram



### Fields

| Field | Function |
|---|---|
| 31-8<br>— | Reserved |
| 7-5<br>— | Reserved |
| 4-0<br>WMRK | Watermark for TX buffer<br>Determines the watermark for the TX buffer<br>When the number of available space in the TX buffer is greater than or equal to the number provided by WMRK (number of 4-byte entries), SR[TXWA] is asserted. For example, a value of 0x1 sets the watermark to 4 bytes, 0x2 sets it to 8 bytes, 0x3 sets it to 12 bytes, and so on. For details, see DMA usage.<br>WMRK = 0 is invalid. |

## 74.11.2.23 Status Register (SR)

### Offset

| Register | Offset |
|---|---|
| SR | 15Ch |

### Function

This register provides all the available status information about SFM command execution and arbitration, the RX buffer, TX buffer, and the AHB buffer.

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | | | 0 | TXFULL | TXDMA | TXWA | TXNE | RXDMA | 0 | | | RXFULL | 0 | | RXWE |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | AHB3FUL | AHB2FUL | AHB1FUL | AHB0FUL | AHB3NE | AHB2NE | AHB1NE | AHB0NE | AHBTRN | Reserved | 0 | Reserved | AHB_ACC | IP_ACC | BUSY |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31-29<br>— | Reserved |
| 28<br>— | Reserved |
| 27<br>TXFULL | TX buffer full<br>Asserted when the FIFO level reaches 39 (that is, TX buffer size of 32 + async FIFO size of 7) |
| 26<br>TXDMA | TX DMA<br>Asserted when the TXFIFO fill via DMA is active and DMA is requested or running |
| 25<br>TXWA | TX buffer watermark available<br>Asserted when the number of available spaces in the TX buffer is greater than or equal to the value provided by TBCT[WMRK]<br>Example: When TBCT[WMRK]=1, SR[TXWA] is de-asserted when TX FIFO has 32+7(size of async FIFO) entries |
| 24<br>TXNE | TX buffer not empty<br>Asserted when TX buffer contains data |
| 23<br>RXDMA | RX buffer DMA<br>Asserted when RX buffer read out via DMA is active; that is, when DMA is requested or running |
| 22-20<br>— | Reserved |
| 19<br>RXFULL | RX buffer full<br>Asserted when the RX buffer is full; that is, when RBSR[RDBFL] is equal to 32 |
| 18-17<br>— | Reserved |
| 16<br>RXWE | RX buffer watermark exceeded<br>Asserted when the number of valid entries in the RX buffer exceeds the number provided in RBCT[WMRK] |
| 15<br>— | Reserved |
| 14<br>AHB3FUL | AHB 3 buffer full<br>Asserted when AHB 3 buffer is full |
| 13 | AHB 2 buffer full |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| AHB2FUL | Asserted when AHB 2 buffer is full |
| 12<br>AHB1FUL | AHB 1 buffer full<br>Asserted when the AHB 1 buffer is full |
| 11<br>AHB0FUL | AHB 0 buffer full<br>Asserted when the AHB 0 buffer is full |
| 10<br>AHB3NE | AHB 3 buffer not empty<br>Asserted when the AHB 3 buffer contains data |
| 9<br>AHB2NE | AHB 2 buffer not empty<br>Asserted when the AHB 2 buffer contains data |
| 8<br>AHB1NE | AHB 1 buffer not empty<br>Asserted when the AHB 1 buffer contains data |
| 7<br>AHB0NE | AHB 0 buffer not empty<br>Asserted when the AHB 0 buffer contains data |
| 6<br>AHBTRN | AHB access transaction pending<br>Asserted when there is a pending request on the AHB interface. See Flash memory mapped AMBA bus. |
| 5<br>— | Reserved |
| 4<br>— | Reserved |
| 3<br>— | Reserved |
| 2<br>AHB_ACC | AHB read access<br>Asserted when the currently executed transaction is initiated by the AHB bus |
| 1<br>IP_ACC | IP access<br>Asserted when transaction currently executed is initiated by the IP bus |
| 0<br>BUSY | Module busy<br>Asserted when module is currently busy handling a transaction to an external flash memory device |

### 74.11.2.24 Flag Register (FR)

#### Offset

| Register | Offset |
|---|---|
| FR | 160h |

#### Function

This register provides all available flags about SFM command execution and arbitration, which may serve as the source for the generation of interrupt service requests. Note that the error flags in this register do not relate directly to the execution of the transaction in the serial flash memory device itself but only to the behavior and conditions visible in the QuadSPI module.

Special write-access is permitted in the enabled mode.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | 0 | Reserved | 0 | TBFF | TBUF | 0 | 0 | ILLINE | 0 | | 0 | 0 | | RBOF | RBDF |
| W | | | | | W1C | W1C | | | W1C | | | | | | W1C | W1C |
| Reset | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | AAEF | AITEF | AIBSEF | ABOF | Reserved | CRCAEF | 0 | 0 | IPAEF | IPIEF | 0 | 0 | 0 | | | TFF |
| W | W1C | W1C | W1C | W1C | W1C | W1C | | | W1C | W1C | | | | | | W1C |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>— | Reserved |
| 29<br>— | Reserved |
| 28<br>— | Reserved |
| 27<br>TBFF | TX buffer fill flag |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | Before writing to the TX buffer, this field should be cleared. Then, it should be read back. If it is set, the TX buffer can include more data. If the field remains cleared, the TX buffer can be considered as full. See TX buffer operation for details. |
| 26<br>TBUF | TX buffer underrun flag<br>This field is set if the module tries to pull data when the TX buffer is empty.. The IP command leading to the TX buffer underrun is continued (data sent to the serial flash memory device is undefined ). Here, a valid underrun means that it should have occurred during the transaction so that few bytes (that is, less than 4 bytes) are left in FIFO and the remaining are filled with "FFFFh". This field does not set if transfer is less than 128 bits. The application must clear the TX buffer in response to this event by writing a 1 to MCR[CLR_TXF]. The application must clear the TX buffer in response to this event by writing a 1 to MCR[CLR_TXF]. |
| 25<br>— | Reserved |
| 24<br>— | Reserved |
| 23<br>ILLINE | Illegal instruction error flag<br>This field is set when an illegal instruction is encountered by the controller in any of the sequences. As soon as the field is set, you must assert MCR[SWRSTSD] and MCR[SWRSTHD]. That is, reset the flash memory and AHB domain after reconfiguring the correct sequence instruction. See Table 555 for a list of legal instructions. |
| 22-21<br>— | Reserved |
| 20<br>— | Reserved |
| 19-18<br>— | Reserved |
| 17<br>RBOF | RX buffer overflow flag<br>This field is set when no more data can be pushed into the RX buffer from the serial flash memory device.<br>The IP command leading to this condition is continued until the number of bytes in IPCR[IDATSZ] are read from the serial flash memory device.<br>The content of the RX buffer remains unchanged. |
| 16<br>RBDF | RX buffer drain flag<br>This field is set if SR[RXWE] is asserted.<br>Writing 1 to this field triggers one of the following actions: |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • If the RX buffer has up to RBCT[WMRK] valid entries, then the flag is cleared.<br>• If the RX buffer has more than RBCT[WMRK] valid entries and the RSER[RBDDE] field is not set (flag driven mode), an RX buffer POP event is triggered.<br>The flag remains set if the RX buffer contains more than RBCT[WMRK] valid entries after the RX buffer POP event is complete.<br>The flag is cleared if the RX buffer contains less than or equal to RBCT[WMRK] valid entries after the RX buffer POP event is complete.<br>See the "Receive Buffer Drain Interrupt or DMA Request" section in Normal mode interrupt and DMA requests for details. |
| 15<br>AAEF | AHB abort error flag<br>This field is set in case software abort is asserted in an ongoing AHB transaction. The field should be cleared before starting any new AHB transaction. |
| 14<br>AITEF | AHB illegal transaction error flag<br>This is set whenever there is no response generated from QuadSPI to AHB bus in case of an illegal transaction and the watchdog timer expires. The timer value is considered as a parameter. |
| 13<br>AIBSEF | AHB illegal burst size error flag<br>This is set whenever the total burst size (size x beat) of an AHB transaction is greater than the prefetch data size, which is defined by BUFxCR[ADATSZ] or data size mentioned in the sequence pointed to by the SEQID field in case ADATSZ = 0. See HBURST support with AHB read details on HBURST feature. |
| 12<br>ABOF | AHB buffer overflow flag<br>ABOF has different behavior depending on which AHB buffer caused this flag. This is set when the size of the AHB access exceeds the size of the AHB buffer (0-2). This condition can occur if BUFxCR[ADATSZ] is programmed incorrectly. The AHB command leading to this condition is continued until the number of entries according to BUFxCR[ADATSZ] have been read from the serial flash memory device.The content of the AHB buffer is not changed. AHB buffer 3 has support for extended prefetch/streaming mode. It's not mandatory that ABOF bit is set for this buffer.If write location of prefetched data from flash in the AHB buffer equals the location of last AHB data read from the AHB buffer, AHB transaction is aborted. |
| 11<br>— | Reserved |
| 10<br>CRCAEF | Sets when there is CRC or ECC error for flash memory A<br>0b - CRCEF interrupt is not generated.<br>1b - CRCEF interrupt is generated. |
| 9<br>— | Reserved |
| 8<br>— | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 7<br>IPAEF | IP command trigger during AHB access error flag<br>This is set when the following condition occurs:<br>• A write access occurs to IPCR[SEQID] and the SR[AHB_ACC] field is set. Any command leading to the assertion of the IPAEF field is ignored. |
| 6<br>IPIEF | IP command trigger could not be executed error flag<br>This is set when the SR[IP_ACC] and SR[AWRACC] fields are set (that is, an IP triggered command is currently executing) and any of the following conditions occurs:<br>• Write access to the IPCR. Any command leading to the assertion of the IPIEF flag is ignored.<br>• Write access to the SFAR<br>• Write access to the RBCT |
| 5<br>— | Reserved |
| 4<br>— | Reserved |
| 3-1<br>— | Reserved |
| 0<br>TFF | IP command transaction finished flag<br>This field is set after the QuadSPI module completes a running IP command. If an error occurs, and the related error flags are valid in the same clock cycle, the TFF flag is asserted. |

### 74.11.2.25 Interrupt and DMA Request Select and Enable Register (RSER)

#### Offset

| Register | Offset |
|---|---|
| RSER | 164h |

#### Function

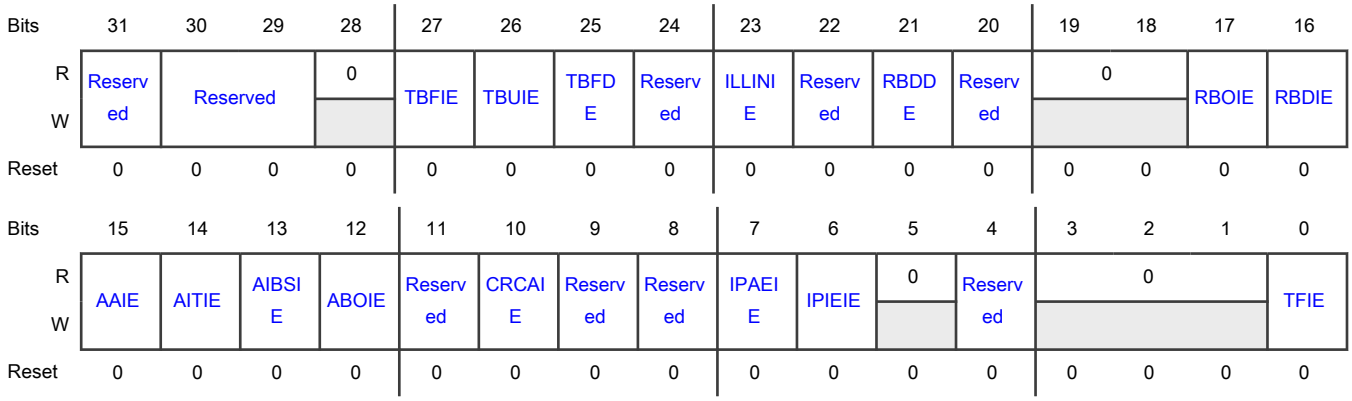
This register provides enables and selectors for the interrupts in the QuadSPI module.

**NOTE**

Each field of the FR enabled as source for an interrupt prevents the QuadSPI module from entering the Stop mode or Module Disable mode when this flag is set.

Special write-access is permitted in the "Anytime" mode.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30-29<br>— | Reserved |
| 28<br>— | Reserved |
| 27<br>TBFIE | TX buffer fill interrupt enable flag<br>This field indicates the TX buffer fill interrupt enable flag.<br>0b - No TBFF interrupt is generated.<br>1b - TBFF interrupt is generated. |
| 26<br>TBUIE | TX buffer underrun interrupt enable flag<br>This field indicates the TX buffer underrun interrupt enable flag.<br>0b - No TBUF interrupt is generated<br>1b - TBUF interrupt is generated |
| 25<br>TBFDE | TX buffer fill DMA enable<br>Enables generation of DMA requests for TX buffer fill. When the value of this field is 1, DMA requests are generated as long as number of available spaces in the TX buffer is greater than or equal to the value provided by TBCT[WMRK].<br>**NOTE**<br>After you write 1 to this field (to enable DMA transfers), writing 0 does not disable DMA transfers. You must perform a software reset for the AHB domain by using MCR[SWRSTHD] to disable DMA transfers. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No DMA request is generated<br>1b - DMA request is generated |
| 24<br>— | Reserved |
| 23<br>ILLINIE | Illegal instruction error interrupt enable<br>Triggered by the ILLINE flag in FR<br>0b - No ILLINE interrupt is generated.<br>1b - ILLINE interrupt is generated. |
| 22<br>— | Reserved |
| 21<br>RBDDE | RX buffer drain DMA enable<br>This field enables generation of DMA requests for RX buffer drain. When the value of this field is 1, the DMA requests are generated as long as SR[RXWE] is set.<br>**NOTE**<br>After you write 1 to this field (to enable DMA transfers), writing 0 does not disable DMA transfers. You must perform a software reset for the AHB domain by using MCR[SWRSTHD] to disable DMA transfers.<br>0b - No DMA request is generated.<br>1b - DMA request is generated. |
| 20<br>— | Reserved |
| 19-18<br>— | Reserved |
| 17<br>RBOIE | RX buffer overflow interrupt enable<br>This field indicates the RX buffer overflow interrupt enable flag.<br>0b - No RBOF interrupt is generated.<br>1b - RBOF interrupt is generated. |
| 16<br>RBDIE | RX buffer drain interrupt enable<br>This field enables generation of IRQ requests for RX buffer drain. When the value of this field is 1, the interrupt is asserted as long as SR[RBDF] is set.<br>0b - No RBDF interrupt is generated.<br>1b - RBDF Interrupt is generated. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 15<br>AAIE | AHB abort error interrupt enable<br>Triggered by AAEF flags in FR.<br>0b - No AAEF interrupt is generated<br>1b - AAEF interrupt is generated |
| 14<br>AITIE | AHB illegal transaction interrupt enable flag<br>This field indicates the AHB illegal transaction interrupt enable flag.<br>0b - No AITEF interrupt is generated.<br>1b - AITEF interrupt is generated. |
| 13<br>AIBSIE | AHB illegal burst size interrupt enable flag<br>This field indicates the AHB illegal burst size interrupt enable flag.<br>0b - No AIBSEF interrupt is generated.<br>1b - AIBSEF interrupt is generated. |
| 12<br>ABOIE | AHB buffer overflow interrupt enable flag<br>This field indicates the AHB buffer overflow interrupt enable flag.<br>0b - No ABOF interrupt is generated.<br>1b - ABOF interrupt is generated. |
| 11<br>— | Reserved |
| 10<br>CRCAIE | CRC and ECC interrupt enable for flash memory A<br>0b - CRCAEF interrupt is not generated.<br>1b - CRCAEF interrupt is generated. |
| 9<br>— | Reserved |
| 8<br>— | Reserved |
| 7<br>IPAEIE | IP command trigger during AHB read access error interrupt enable flag<br>This field indicates IP command trigger during AHB read access error interrupt enable flag.<br>0b - No IPAEF interrupt is generated<br>1b - IPAEF interrupt is generated |
| 6<br>IPIEIE | IP command trigger during IP access error interrupt enable flag<br>This field indicates IP command trigger during IP access error interrupt enable flag. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - No IPIEF interrupt is generated<br>1b - IPIEF interrupt is generated |
| 5<br>— | Reserved |
| 4<br>— | Reserved |
| 3-1<br>— | Reserved |
| 0<br>TFIE | Transaction finished interrupt enable flag<br>This field indicates the transaction finished interrupt enable flag.<br>0b - No TFF interrupt is generated.<br>1b - TFF interrupt is generated. |

#### 74.11.2.26 Sequence Pointer Clear Register (SPTRCLR)

**Offset**

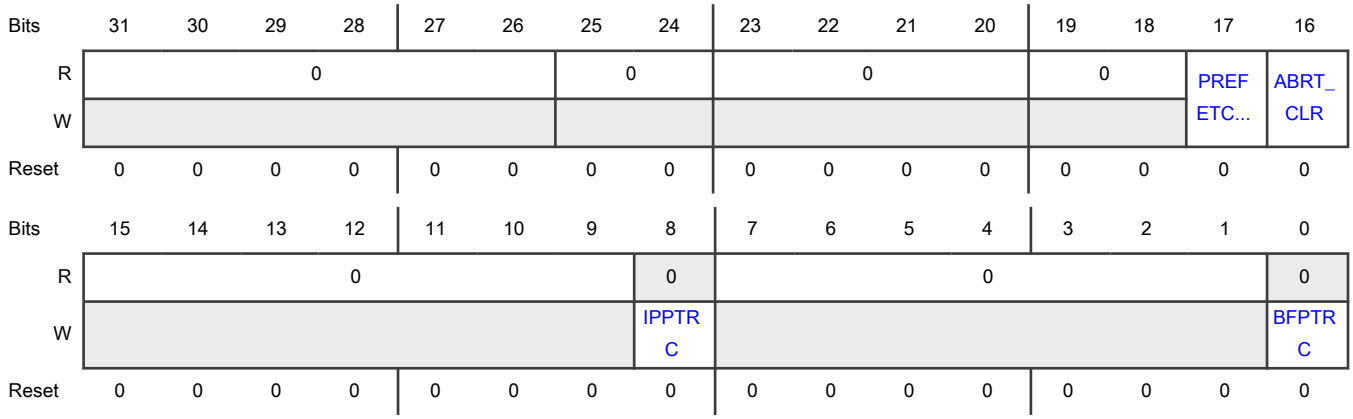
| Register | Offset |
|---|---|
| SPTRCLR | 16Ch |

**Function**

This register provides fields to reset the IP and buffer sequence pointers. The sequence pointer contains the index of instructions within the LUT entry that is to be executed next. For example, if the LUT entry ends on a JMP_ON_CS value of 2, the index is stored as 2.

The software should reset the sequence pointers defined by JMP_ON_CS operand whenever the sequence ID is required to be changed by updating the SEQID field in the IPCR or BFGENCR.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-26<br>— | Reserved |
| 25-24<br>— | Reserved |
| 23-20<br>— | Reserved |
| 19-18<br>— | Reserved |
| 17<br>PREFETCH_DIS | Prefetch disable<br>This field should be configured to disable the prefetch mechanism of receiver. It is not based on dynamic programming. Therefore, it should be programmed initially at once. When this field is set, then during an ongoing flash memory read, any subsequent AHB read is checked for buffer hit. However, after the end of flash memory read, as soon as chip select is deasserted, any subsequent AHB read results in flushing of the current AHB buffer data and issues fresh flash memory transaction if the AHB buffer data is not updated with the flash memory. |
| 16<br>ABRT_CLR | Flash memory Abort/AHB buffer clear<br>This is a dynamic field. Writing a 1 to it, irrespective of the prefetch disable, clears the AHB buffer pointers and also aborts any ongoing flash memory transaction (if any) and rejects any ongoing AHB read with an error response (if any).<br>QuadSPI sets this field to 0 after clearing the AHB buffer pointers. |
| 15-9<br>— | Reserved |
| 8 | IP pointer clear |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| IPPTRC | This is a self-clearing field.<br>1b - Clears the sequence pointer for IP accesses as defined in IPCR. |
| 7-1<br>— | Reserved |
| 0<br>BFPTRC | Buffer pointer clear<br>This is a self-clearing field.<br>1b - Clears the sequence pointer for AHB read accesses as defined in BFGENCR. |

### 74.11.2.27 Serial Flash Memory A1 Top Address Register (SFA1AD)

#### Offset

| Register | Offset |
|---|---|
| SFA1AD | 180h |

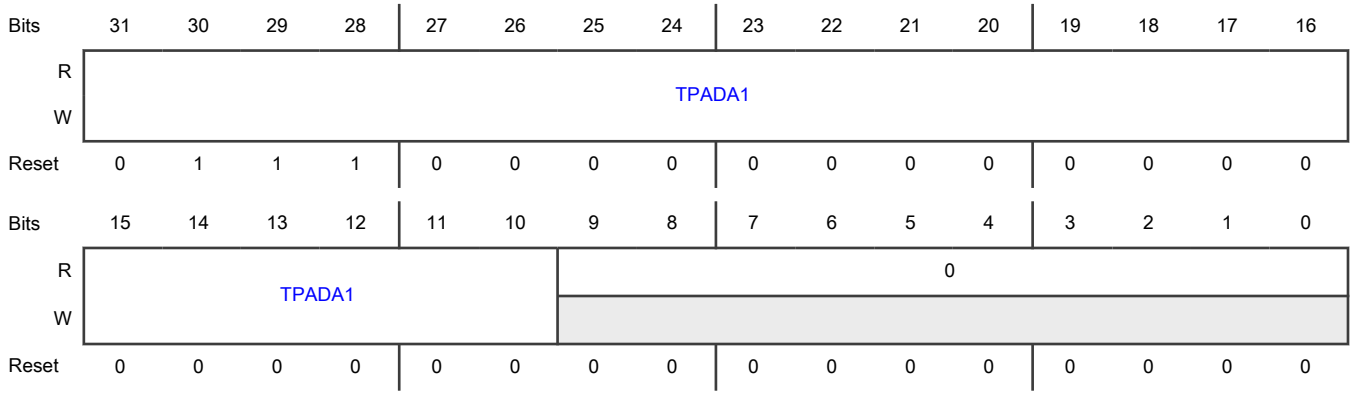
#### Function

This register provides the address mapping for serial flash memory A1. The difference between SFA1AD[TPADA1] and AMBA_BASE defines the size of the memory map for serial flash memory A1.

Special write-access is permitted if:

- SR[IP_ACC] = 0
- SR[AHB_ACC] = 0

#### Diagram



### Fields

| Field | Function |
|---|---|
| 31-10<br>TPADA1 | Top address for serial flash memory A1<br>In effect, TPADxx is the first location of the next memory. |
| 9-0<br>— | Reserved |

## 74.11.2.28 Serial Flash Memory A2 Top Address Register (SFA2AD)

### Offset

| Register | Offset |
|---|---|
| SFA2AD | 184h |

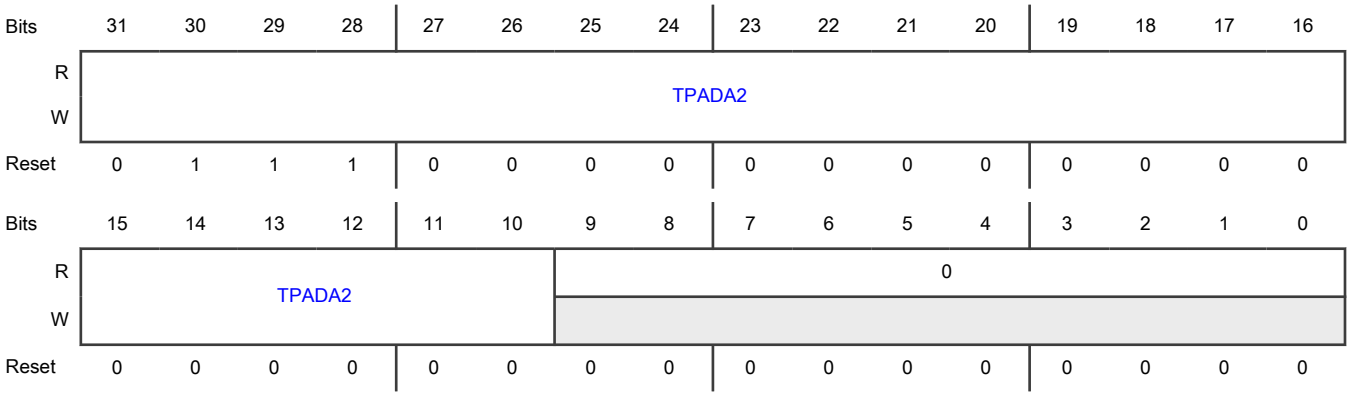
### Function

This register provides the address mapping for serial flash memory A2. The difference between SFA2AD[TPADA2] and SFA1AD[TPADA1] defines the size of the memory map for serial flash memory A2.

Special write-access is permitted if:

- SR[IP_ACC] = 0
- SR[AHB_ACC] = 0

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-10<br>TPADA2 | Top address for serial flash memory A2<br>In effect, TPxxAD is the first location of the next memory. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| 9-0<br>— | Reserved |

### 74.11.2.29 Serial Flash Memory B1 Top Address Register (SFB1AD)

#### Offset

| Register | Offset |
|---|---|
| SFB1AD | 188h |

#### Function

This register provides the address mapping for serial flash memory B1. The difference between SFB1AD[TPADB1] and SFA2AD[TPADA2] defines the size of the memory map for serial flash memory B1.

Special write-access is permitted if:

- SR[IP_ACC] = 0
- SR[AHB_ACC] = 0

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TPADB1 | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TPADB1 | | | | | | | | 0 | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-10<br>TPADB1 | Top address for serial flash memory B1.<br>In effect, TPxxAD is the first location of the next memory. |
| 9-0<br>— | Reserved |

### 74.11.2.30 Serial Flash Memory B2 Top Address Register (SFB2AD)

#### Offset

| Register | Offset |
|---|---|
| SFB2AD | 18Ch |

#### Function

This register provides the address mapping for serial flash memory B2. The difference between SFB2AD[TPADB2] and SFB1AD[TPADB1] defines the size of the memory map for serial flash memory B2.

Special write-access is permitted if:

- SR[IP_ACC] = 0
- SR[AHB_ACC] = 0

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | TPADB2 | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TPADB2 | | | | | | 0 | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-10<br>TPADB2 | Top address for serial flash memory B2.<br>In effect, TPxxAD is the first location of the next memory. |
| 9-0<br>— | Reserved |

### 74.11.2.31 RX Buffer Data Register (RBDR0 - RBDR31)

#### Offset

For a = 0 to 31:

| Register | Offset |
|---|---|
| RBDRa | 200h + (a × 4h) |

### Function

These registers provide access to individual entries in the RX buffer. See Table 557 for the byte ordering scheme.

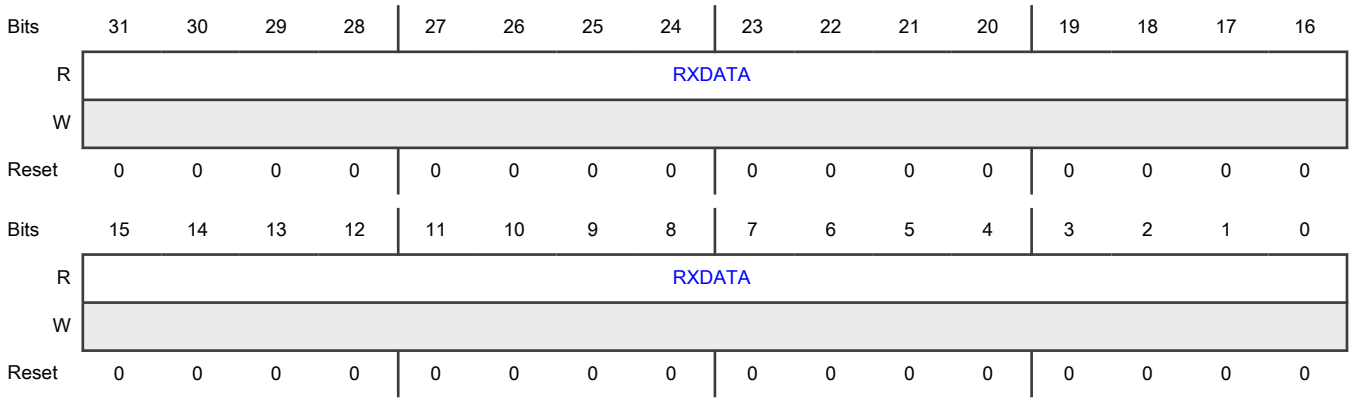
RBDR0 corresponds to the actual position of the read pointer within the RX buffer. The number of valid entries available depends on the number of RX buffer entries implemented and on the number of valid buffer entries available in the RX buffer.

Example 1 - RX buffer filled completely with 32 words: In this case, the address range for valid read access extends from RBDR0 to RBDR31 RBDR63.

Example 2 - RX buffer filled with five valid words: RX buffer fill level of RBSR[RDBFL] is 5. In this case, access to RBDR4 provides the last valid entry.

Any access beyond the range of valid RX buffer entries provides undefined results.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>RXDATA | RX data<br>This field contains the data associated with the related RX buffer entry. For data format and byte ordering, see Byte ordering of serial flash memory read data. |

## 74.11.2.32 LUT Key Register (LUTKEY)

### Offset

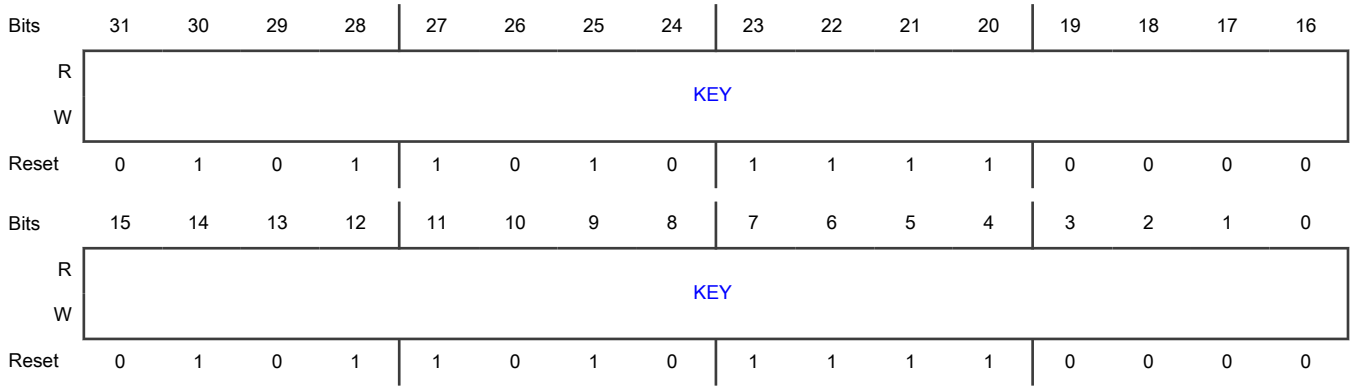
| Register | Offset |
|---|---|
| LUTKEY | 300h |

### Function

This register contains the key to lock and unlock the LUT. See LUT for details.

Special write-access is permitted in the "Anytime" mode. Write access is not permitted if MGC[11]=1b.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>KEY | Key to lock or unlock the LUT<br>The key is 0x5AF05AF0 and the read value is always 0x5AF05AF0. Write access is not permitted if MGC[11]=1b. |

## 74.11.2.33 LUT Lock Configuration Register (LCKCR)

### Offset

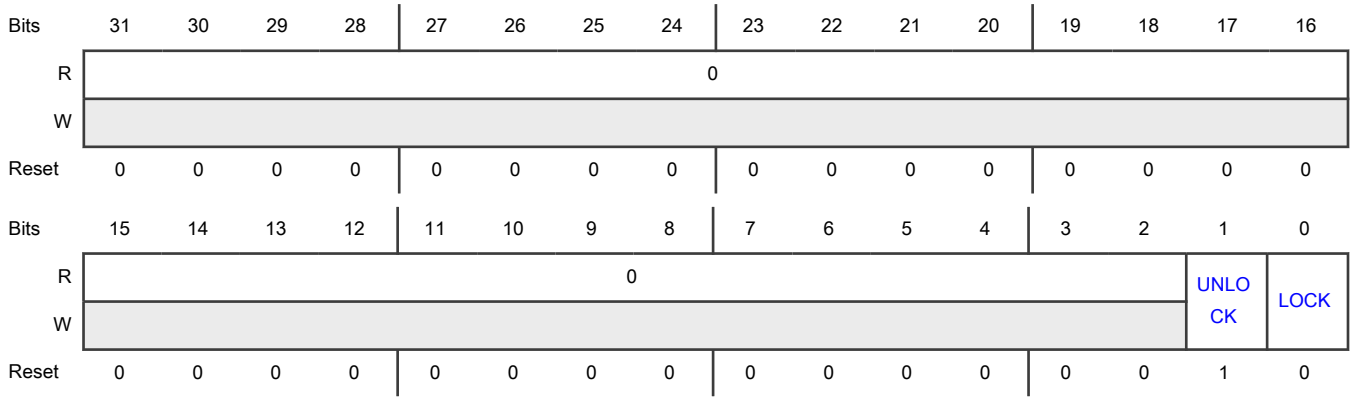
| Register | Offset |
|---|---|
| LCKCR | 304h |

### Function

This register is used along with the LUTKEY register to lock or unlock the LUT. This register should be written immediately after the LUTKEY register for the lock or unlock operation to be successful. See LUT for details. Setting both the LOCK and UNLOCK bits as "00" or "11" is not allowed.

Special write access is permitted after writing the LUT key register. Write access is not permitted if MGC[11]=1b.

### Diagram



#### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>UNLOCK | Unlock LUT<br>Unlocks the LUT when the following two conditions are met:<br>• This register is written just after the LUT Key Register (LUTKEY).<br>• The LUT key register was written with the 0x5AF05AF0 key. |
| 0<br>LOCK | Lock LUT<br>Locks the LUT when the following conditions are met:<br>• This register is written just after the LUT Key Register (LUTKEY).<br>• The LUT key register is written with the 0x5AF05AF0 key. |

### 74.11.2.34 LUT Register (LUT0)

#### Offset

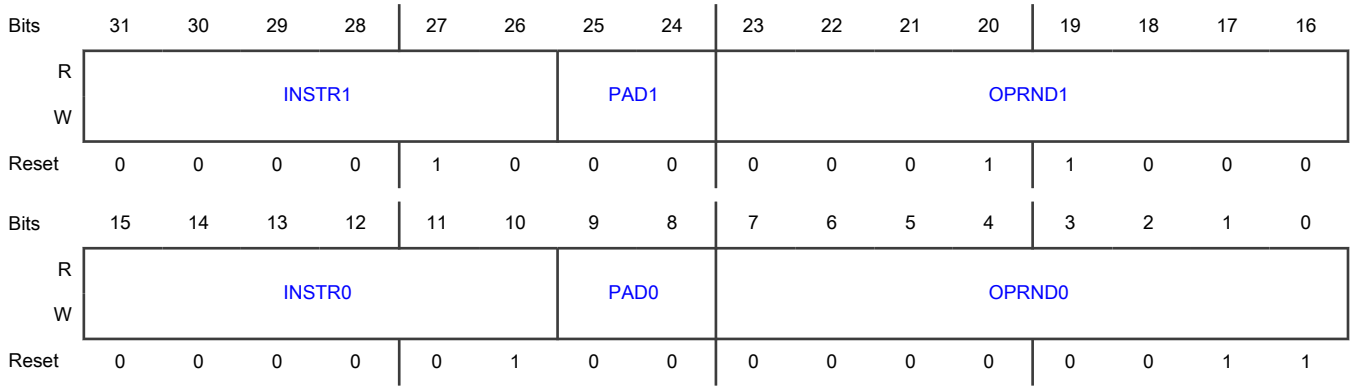
| Register | Offset |
|---|---|
| LUT0 | 310h |

#### Function

A sequence of instruction-operand pairs may be pre-populated in the LUT according to the device connected on board. Each instruction-operand pair is of 16 bits (2 bytes) each. Every sequence preprogrammed by Program Sequence Engine in the LUT is referred to by its index. The LUT registers act as lookup tables for sequences of instructions. The programmable sequence engine executes the instructions in these sequences to generate a valid serial flash memory transaction. There are a total of 20 LUT registers. These 20 registers are divided into groups of 5 registers that make a valid sequence. Therefore, LUT[0], LUT[5], LUT[10] ..... LUT[15] are the starting registers of a valid sequence. Each of these sets of 5 registers can have a maximum of 10 instructions. Reset value of the register shown below is only applicable to LUT2 to LUT19. A maximum of 4 sequences can be defined at one time. See LUT that describes the LUT registers in detail.

Special write-access is permitted if the LUT is unlocked.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-26<br>INSTR1 | Instruction 1 |
| 25-24<br>PAD1 | Pad information for INSTR1<br>00b - 1 Pad<br>01b - 2 Pads<br>10b - 4 Pads<br>11b - NA |
| 23-16<br>OPRND1 | Operand for INSTR1 |
| 15-10<br>INSTR0 | Instruction 0 |
| 9-8<br>PAD0 | Pad information for INSTR0<br>00b - 1 Pad<br>01b - 2 Pads<br>10b - 4 Pads<br>11b - NA |
| 7-0<br>OPRND0 | Operand for INSTR0 |

### 74.11.2.35 LUT Register (LUT1)

#### Offset

| Register | Offset |
|---|---|
| LUT1 | 314h |

#### Function

A sequence of instruction-operand pairs may be pre-populated in the LUT according to the device connected on board. Each instruction-operand pair is of 16 bits (2 bytes) each. Every sequence preprogrammed by Program Sequence Engine in the LUT is referred to by its index. The LUT registers act as lookup tables for sequences of instructions. The programmable sequence engine executes the instructions in these sequences to generate a valid serial flash memory transaction. There are a total of 20 LUT registers. These 20 registers are divided into groups of 5 registers that make a valid sequence. Therefore, LUT[0], LUT[5], LUT[10] ..... LUT[15] are the starting registers of a valid sequence. Each of these sets of 5 registers can have a maximum of 10 instructions. Reset value of the register shown below is only applicable to LUT2 to LUT19. A maximum of 4 sequences can be defined at one time. See LUT that describes the LUT registers in detail.

Special write-access is permitted if the LUT is unlocked.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | INSTR1 | | | | | | PAD1 | | OPRND1 | | | | | | | |
| Reset | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R<br>W | INSTR0 | | | | | | PAD0 | | OPRND0 | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31-26<br>INSTR1 | Instruction 1 |
| 25-24<br>PAD1 | Pad information for INSTR1<br>00b - 1 Pad<br>01b - 2 Pads<br>10b - 4 Pads<br>11b - NA |
| 23-16 | Operand for INSTR1 |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| OPRND1 | |
| 15-10<br>INSTR0 | Instruction 0 |
| 9-8<br>PAD0 | Pad information for INSTR0<br>00b - 1 Pad<br>01b - 2 Pads<br>10b - 4 Pads<br>11b - NA |
| 7-0<br>OPRND0 | Operand for INSTR0 |

### 74.11.2.36 LUT Register (LUT2 - LUT19)

#### Offset

For a = 2 to 19:

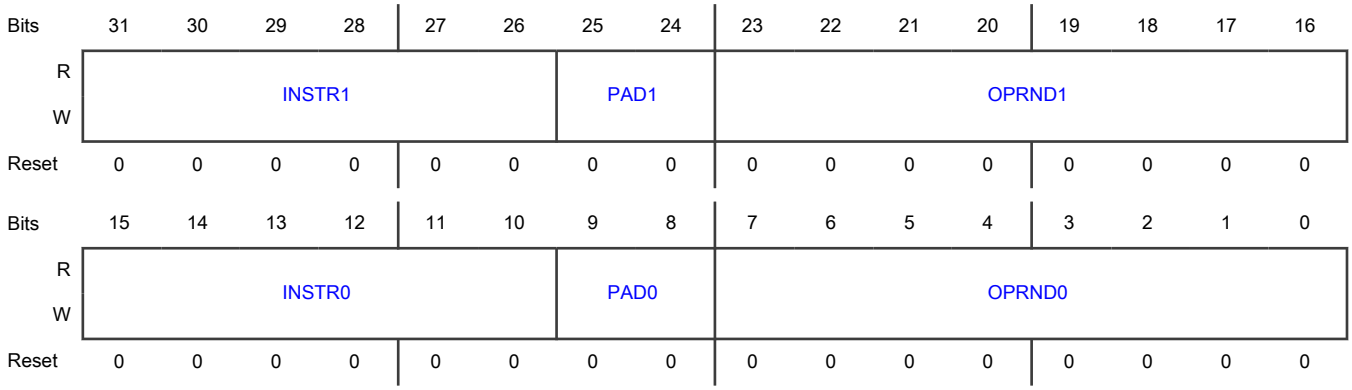
| Register | Offset |
|---|---|
| LUTa | 310h + (a × 4h) |

#### Function

A sequence of instruction-operand pairs may be pre-populated in the LUT according to the device connected on board. Each instruction-operand pair is of 16 bits (2 bytes) each. Every sequence preprogrammed by Program Sequence Engine in the LUT is referred to by its index. The LUT registers act as lookup tables for sequences of instructions. The programmable sequence engine executes the instructions in these sequences to generate a valid serial flash memory transaction. There are a total of 20 LUT registers. These 20 registers are divided into groups of 5 registers that make a valid sequence. Therefore, LUT[0], LUT[5], LUT[10] ..... LUT[15] are the starting registers of a valid sequence. Each of these sets of 5 registers can have a maximum of 10 instructions. Reset value of the register shown below is only applicable to LUT2 to LUT19. A maximum of 4 sequences can be defined at one time. See LUT that describes the LUT registers in detail.

Special write-access is permitted if the LUT is unlocked.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-26<br>INSTR1 | Instruction 1 |
| 25-24<br>PAD1 | Pad information for INSTR1<br>00b - 1 Pad<br>01b - 2 Pads<br>10b - 4 Pads<br>11b - NA |
| 23-16<br>OPRND1 | Operand for INSTR1 |
| 15-10<br>INSTR0 | Instruction 0 |
| 9-8<br>PAD0 | Pad information for INSTR0<br>00b - 1 Pad<br>01b - 2 Pads<br>10b - 4 Pads<br>11b - NA |
| 7-0<br>OPRND0 | Operand for INSTR0 |

## 74.11.3 Serial flash memory address assignment

The serial flash memory address assignment can be modified by writing into Serial Flash Memory A1 Top Address Register (SFA1AD) and Serial Flash Memory A2 Top Address Register (SFA2AD) for device A

**Table 577. Serial flash memory address assignment**

| Parameter | Function | Access mode |
|---|---|---|
| QuadSPI_AMBA_BASE ((31:10) - 22 bits) | QuadSPI AHB base address<br>First address of the serial flash memory device as presented to the QuadSPI controller. This might be the base of the serial flash memory in the system address map or it may be a remapping, for instance to 0h, performed by the system. (See the system address map file attached to this document) | |
| QuadSPI_ARDB_BASE | First address of the QuadSPI Rx buffer on system memory map | |
| TOP_ADDR_MEMA1(TPADA1) | Top address for the external flash memory A1 (the first of the two independent flash memories sharing the IOFA) | Any access to the address space between TOP_ADDR_MEMA1 and QuadSPI_AMBA_BASE is routed to serial flash memory A1. |
| TOP_ADDR_MEMA2(TPADA2) | Top address for the external flash memory A2 (the second of the two independent flash memories sharing the IOFA) | Any access to the address space between TOP_ADDR_MEMA2 and TOP_ADDR_MEMA1 is routed to serial flash memory A2. |

## 74.12 Flash memory mapped AMBA bus

QuadSPI_AMBA_BASE defines the address to be used as the start address of the serial flash memory device, as defined by the system memory map. Note that this may be a remapping of the physical address of the serial flash memory in the system. See the system address map for details.

**Table 578. QuadSPI AMBA bus memory map**

| Address | Register name |
|---|---|
| QuadSPI_AMBA_BASE to (TOP_ADDR_MEMA2 - 1h) | • See Memory-mapped serial flash memory data—individual flash memory mode on flash memory A.<br>• For information about byte ordering, see Table 557 and Table 558. |
| QuadSPI_ARDB_BASE to... (32 * 4 Byte) QuadSPI_ARDB_BASE + 1FFh | • See AHB RX Data Buffer Register (ARDB0 - ARDB127).<br>• For information about the byte ordering, see Table 557. |

**NOTE**

Any read access to non-implemented addresses provides undefined results.

In individual flash memory modes, the 3/4 address bytes (as programmed in the instruction/operand in the sequence) available for the flash memory address are determined by SFADR[23:0] or SFADR[31:0] as provided in the table shown above.

### 74.12.1 AHB bus access read considerations

Note that all logic in the QuadSPI module implementing the AHB bus access is designed to read the content of an external serial flash memory device. Therefore, the following restrictions apply to the QuadSPI module with respect to accesses to the AHB bus:

- Any AHB command resulting in the assertion of the FR[ABSEF] flag is answered with the ERROR condition according to the AMBA_AHB specification. The resulting AHB command is ignored.
- AHB bus read access types—NONSEQ and BUSY—are fully supported.
- AHB read access type—SEQ—is treated in the same way as NONSEQ. See Flash memory mapped AMBA bus for details.
- Early burst termination is not supported for AHB transactions.
- An AHB error response occurs when FR[AITEF] bit is set
- An AHB bus stall along with error response occurs when FR[AAEF] bit is set

### 74.12.2 Memory-mapped serial flash memory data—individual flash memory mode on flash memory A

Starting with address QuadSPI_AMBA_BASE, the content of the first external serial flash memory device is mapped into the address space of the device containing the QuadSPI module. Serial flash memory byte address 0h corresponds to bus address, QuadSPI_AMBA_BASE, in an increasing order. . See the following table for the address mapping. The byte ordering for 32-bit access is provided in Table 557 and for 64-bit read access, the byte ordering is provided in Table 558.

**Table 579. Memory-mapped individual flash memory mode—flash memory A address scheme**

| Memory-mapped address 32-bit access | Memory-mapped address 64-bit access | Serial flash memory byte address | Flash memory device |
|---|---|---|---|
| QuadSPI_AMBA_BASE + 0h | QuadSPI_AMBA_BASE + 0h | 0h to 3h | A1 |
| QuadSPI_AMBA_BASE + 4h | | 4h to 7h | |
| ... | ... | ... | |
| TOP_ADDR_MEMA1 - 8h | TOP_ADDR_MEMA1 - 8h | (TOP_ADDR_MEMA1- 8h) to (TOP_ADDR_MEMA1 - 0h4 -0h1) | |
| TOP_ADDR_MEMA1 - 4h | | (TOP_ADDR_MEMA1 - 4h) to (TOP_ADDR_MEMA1 - 0h1) | |
| TOP_ADDR_MEMA1 + 4h | TOP_ADDR_MEMA1 + 0h | 0h to 3h | A2 |
| TOP_ADDR_MEMA1 + 0h4 | | 4h to 7h | |
| ... | ... | ... | |
| TOP_ADDR_MEMA2 - 8h | TOP_ADDR_MEMA2 - 8h | (TOP_ADDR_MEMA2 - 8h) to (TOP_ADDR_MEMA2 - 4h - 1h) | |
| TOP_ADDR_MEMA2 - 4h | | (TOP_ADDR_MEMA2 - 4h) to (TOP_ADDR_MEMA2 - 1h) | |

The available address range depends on the size of the external serial flash memory device. Any access beyond the size of the external serial flash memory provides undefined results.

For details concerning the read process, see Flash memory read.

## 74.12.3 ARDB register descriptions

**NOTE**

See the system memory map in this document for the base address of the QuadSPI AHB RX data buffer.

### 74.12.3.1 ARDB memory map

QuadSPI_ARDB base address: 6800_0000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h - 1FCh | AHB RX Data Buffer Register (ARDB0 - ARDB127) | 32 | R | 0000_0000h |

### 74.12.3.2 AHB RX Data Buffer Register (ARDB0 - ARDB127)

#### Offset

For a = 0 to 127:

| Register | Offset |
|---|---|
| ARDBa | 0h + (a × 4h) |

#### Function

This register is used to read the buffer content of the RX buffer from successive addresses. ARDB0 corresponds to the RX buffer register entry corresponding to the current value of the read pointer in an increasing order.

The increment of the read pointer depends on the access scheme (DMA or flag-driven). See Flash memory read, RX buffer, data read through register interface, and AHB read for the description of successive accesses to the RX buffer content. See Byte Ordering of Serial Flash Memory Read Data for the byte ordering scheme.

Valid address range accessible in the ARDB*n* range depends on the number of RX buffer entries implemented and on the number of valid buffer entries available in the RX buffer.

- Example 1 - RX buffer filled completely with 128 words: In this case, the address range for valid read access extends from ARDB0 to ARDB127.
- Example 2 - RX buffer filled with five valid words; RX buffer fill level RBSR[RDBFL] is 5. In this case, an access to ARDB4 provides the last valid entry.

#### Diagram

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

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

**Fields**

| Field | Function |
|---|---|
| 31-0<br>ARXD | ARDB provided RX buffer data<br>Byte order (endianness) is identical to the RX buffer data registers. |

## 74.13 Glossary

| | |
|---|---|
| **AHB** | Advanced high-performance bus, a version of AMBA |
| **AMBA** | Advanced microcontroller bus architecture |
| **APB** | Advanced peripheral bus |
| **BE** | Big endian byte ordering |
| **CRS** | Center aligned read strobe |
| **CS** | Chip select |
| **FRAD** | Flash region access descriptor |
| **I/O** | Input output, I/O lines are also referred to as pads in this chapter |
| **IFM** | Individual flash memory mode |
| **LE** | Little endian |
| **MDAD** | Master domain access descriptors |
| **MGID** | Master-Group identifier |
| **PCS** | Peripheral chip select |
| **SCK** | Serial communications clock |
| **SFM** | Serial flash memory |

# Chapter 75
# Debug Subsystem

## 75.1 Introduction

This chapter discusses the debug and trace architecture of the chip that is based on the specifications provided in the *Arm® CoreSight™ SoC-400 Technical Reference Manual*. See References for a link to this document and to other related documentation available on the Arm website.

The chip architecture includes debug and trace modules. See Features for details on the debug and trace features that Cortex-M7 core clusters support.

The debug components that the Cortex-M7 core supports are accessible via the Arm DAP controller-based architecture. The DAP controller works in parallel with the system JTAGC. Both these controllers share the JTAG port and JTAG instruction set.

The system components are similar for different chips in the same family and include the primary core debug interfaces, the chip-level debug interfaces, and chip-level trace interfaces. The accelerator components include debug and trace circuits. These circuits are necessary for any application-specific accelerators required for the different application spaces that a chip supports. They vary from one chip to another and could include only trace components.

## 75.2 Interfaces supported in MCX E31 family

**Table 580. Interfaces supported in MCXE31 family**

| MCXE315/MCXE316/MCXE317 | MCXE31B |
|---|---|
| Cortex-M7_0 (CTI, DWT, BPU, ITM) | Cortex-M7_0 (CTI, DWT, BPU, ETM, ITM) |
| - | - |
| - | - |
| Cortex-M0+ | Cortex-M0+ |
| - | HTM (CTI+DMA/EMAC TRACE) |
| SWO | SWO |
| - | TPIU |
| - | ETF (4) |
| - | FUNNELs (3) |
| CTI (2) | CTI (3) |
| CTM (1) | CTM (1) |
| Timestamp | Timestamp |
| MDM_AP | MDM_AP |
| SDA_AP | SDA_AP |

## 75.3 Block diagram

This figure illustrates the DAP architecture of the MCX E31 family.

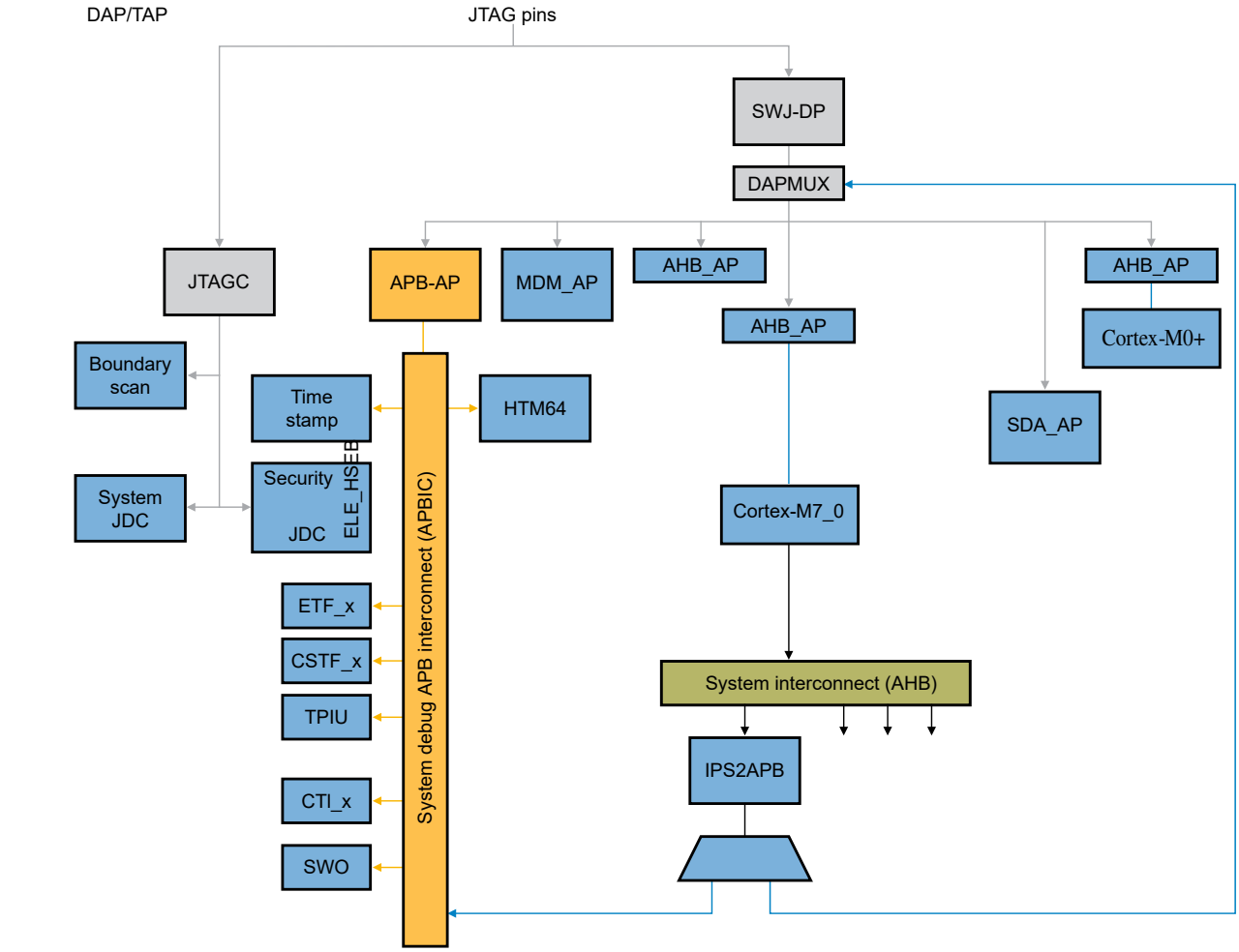



Figure 437. DAP architecture

**NOTE**

Debug components other than the Cortex-M0+ core are accessible through software path (CORE > AXBS > AIPS > DAPMUX > APs), that is, by directly specifying the memory mapped addresses without doing challenge response.

## 75.4 Features

This chip includes these features that support the functions listed under each feature type:

- Debug control
  - Is implemented via IEEE 1149.1-compliant JTAG
- Test control
  - Is implemented via IEEE 1149.1-compliant JTAG
  - Uses IEEE 1149.6 extension to IEEE 1149.1
- Trace interface
  - Includes four high-speed data pads and one clock pad @120 MHz (120 MB/s throughput per pad)
  - Includes 16 low-speed data pads and one clock pad @25 MHz (100 MB/s throughput per pad)
- Debug security

  - Includes a DAP/TAP interface that controls all debug features and is gated by Cortex M0+ JDC module
  - Includes security modes as described in the "Security" chapter
- Debugging and run control
  - Is implemeted through stopping points, starting points, breakpoints, and watchpoints
  - Cortex-M7 core supports eight instruction comparators and a watchpoint unit having four watchpoints
- Trace source
  - Includes Cortex-M7:
    - ETM that provides instruction and data trace
    - ITM that provides DWT, time-stamping, and diagnostic information
- Cross-triggering support
  - Controls run-control options of cores based on other cores
  - Provides a matrix having connections to:
    - HTM via system CTI
    - Cortex-M7 CPUs
- Ability to view and modify all memory-mapped areas that are not otherwise blocked—includes a system bus debug access port
- Performance monitoring of cores—implements a performance monitoring unit (PMU) on each Cortex core
- Safety
  - Supports monitoring of debug signals to avoid common mode faults
  - Allows checking of erroneous activation of debugging, especially if intrusive (for example, a CPU entering Debug state)
- Timestamps
  - Generates a timestamp bus for distribution to the trace sources
  - Includes a 48-bit binary timestamp bus
  - Clocks timestamp generator by a frequency of 160 MHz

## 75.5 Debug

### 75.5.1 TAP connectivity

This figure shows a detailed view of TAP connectivity. JTAG select, SWJ-DP, and JTAG-DP are parts of DAP.

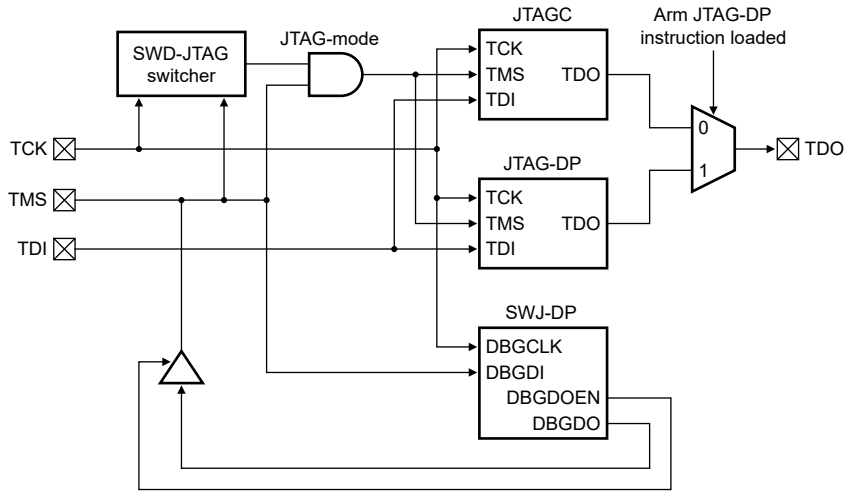



Figure 438. TAP connectivity

The debug port comes out of reset in a standard JTAG mode. It switches to another mode by using the change sequences. After the mode changes, unused debug pins can be reassigned to any of their alternative muxed functions.

JTAG-DP and JTAGC are connected in an overlay scheme and both have an Instruction Register (IR) length of 8 bits.

### 75.5.2 DAP TAP

DAP is an Arm component that provides multiple-master driving ports. A single external interface port accesses and controls these ports to provide system-wide debug.

The DAP Instruction Register (DAP IR) overlays with the system JTAG Instruction Register (JTAGC IR). Table 581 presents DAP instructions. In addition to the four codes listed in the table, DAP uses BYPASS, which is identical to JTAGC BYPASS and is therefore not shown in the table.

Table 581. DAP IR codes

| Code | DAP IR |
|---|---|
| 1111_1000b | ABORT |
| 1111_1010b | DPACC |
| 1111_1011b | APACC |
| 1111_1110b | IDCODE |

DAP offers an AHB master interface to access system buses. It also exports the internal DAP bus to extend the access ports. For more information on DAP TAP, see the *Arm Debug Interface Architecture Specification* document available in References.

In this chip:

- The AHB slave ports of all Cortex-M*x* cores provide debugger access to all memory units and registers in the system.
- The new pass-through approach allows the debugger to see a cache coherent view of the memory map for that core.
- The debugger must access the corresponding AHB_AP port to access the AHB-S port of that subsystem.
- XRDC controls system access.
- A core can access the debug components of another core through a DAPMUX to the bus interconnect.
- The exported DAP bus hosts AHB_AP and MDM_AP.

— MDM_AP hosts system-level JTAG status and control registers (see MDM_AP register descriptions). These registers can be used for cross triggering, synchronized debug, and other miscellaneous control and status functions.

— APB_AP uses an APSEL value of 1h to access any APB-mapped debug modules.

Table 585 describes the access addresses of APB-mapped debug modules. The value of the DAP select signal in the APSEL field of SWJ-DP's Select register selects the access ports in the DAP. Table 582 shows APSEL decoding.

Table 582. DAP master address mapping

| DAP master # | DAP component | DAPBUS base address (access from debugger) [1] | System memory map address (access from cores) | | Selected port or master | Applicability |
|---|---|---|---|---|---|---|
| | | | Page number | Base address | | |
| 0h | Reserved | 0000_0000h | 0 | 4025_0000h | Reserved | - |
| 1h | APB_AP to all APB-mapped debug modules (for example, ETFs, HTM, and trace funnels) | 0100_0000h | 0 | 4025_0100h | APB_AP to all APB-mapped debug modules (for example, ETFs, HTM, CSTF, and so on) | MCXE31B (for ETF, HTM, and CSTF see Table 580) |
| 2h | Reserved | 0200_0000h | 0 | 4025_0200h | Reserved | - |
| 4h | AHB_AP to Cortex-M7_0 debug modules and chip system memory map | 0400_0000h | 0 | 4025_0400h | AHB_AP to Cortex-M7_0 debug modules and chip system memory map | All |
| 5h | AHB_AP to Cortex-M7_1 debug modules and chip system memory map | 0500_0000h | 0 | 4025_0500h | AHB_AP to Cortex-M7_1 debug modules and chip system memory map | MCXE317 |
| 6h | MDM_AP | 0600_0000h | 0 | 4025_0600h | MDM_AP | All |
| 7h | SDA_AP | 0700_0000h | 0 | 4025_4700h | SDA_AP used for challenge-response (CR) in SWJ-DP mode | All |
| 8h—FFh | Reserved (default AP response) | — | — | — | — | - |

1. For example, 4000_XXYYh address is provided when accessing access ports (APs) via the debugger. Here, XX shows the AP select number and YY shows the address to be accessed. If you access SDA_AP via the debugger on 80h, you must provide the address 4000_0780h.

**NOTE**

Accessing SDA_AP simultaneously from the core and the debugger is prohibited. If you try to do so, you may receive unpredictable responses to core-initiated transactions (for example, incorrect data read, a failed attempt to write to the register, or a transfer error). Debugger-initiated transactions proceed correctly.

### 75.5.3 System JTAGC

JTAGC connects in parallel with the TAP controller (JTAG-DP of DAP), which has an IR length of 8 bits. The JTAGC IR codes overlay the ones of the DAP controller. DAP uses four instructions and JTAGC uses the remaining ones. The TAP outputs (TPOs) are multiplexed based on the selected IR code. This chip is fully JTAG-compliant and appears as a single TAP to the JTAG chain.

This table shows the JTAGC IR codes. The instructions that are used by Arm DAP TAP are shown in Table 581.

**Table 583. JTAG instructions for JTAGC**

| Code | JTAGC IR |
|---|---|
| 0000_0000b | IDCODE |
| 0000_0001b | Reserved |
| 0000_0010b | SAMPLE/PRELOAD |
| 0000_0011b | SAMPLE |
| 0000_0100b | EXTEST |
| 0000_0101b | HI-Z |
| 0000_0110b | Reserved |
| 0000_0111b | Reserved |
| 0000_1000b—0000_1001b | Reserved |
| 0000_1010b—0000_1011b | Reserved |
| 0000_1100b | CLAMP |
| 0000_1101b | ENABLE_SOC_DATA1 |
| 0000_1110b | Reserved |
| 0000_1111b | Reserved |
| 1000_0000b | Reserved |
| 1000_0001b | Reserved |
| 1000_0010b | Reserved |
| 1000_0011b | Reserved |
| 1000_0100b | Reserved |
| 1000_0101b | Reserved |

*Table continues on the next page...*

Table 583. JTAG instructions for JTAGC (continued)

| Code | JTAGC IR |
|---|---|
| 1000_0110b—1000_0111b | Reserved |
| 1000_1000b | Reserved |
| 1000_1001b | Reserved |
| 1000_1010b—1000_1111b | Reserved |
| 1001_0000b | Security JDC |
| 1001_0001b | System JDC |
| 1001_0010b—1001_0111b | Reserved for other system auxiliary clients |
| 1001_1000b—1001_1011b | Reserved |
| 1001_1100b—1011_1111b | Reserved for other system auxiliary clients |
| 1100_0000b | Reserved |
| 1100_0001b | Reserved |
| 1100_0010b—1110_1111b | Reserved |
| 1111_1000b | ABORT (Arm) |
| 1111_1001b | Reserved |
| 1111_1010b | DPACC (Arm) |
| 1111_1011b | APACC (Arm) |
| 1111_1100b | Reserved |
| 1111_1101b | Reserved |
| 1111_1110b | IDCODE (Arm) |
| 1111_1111b | BYPASS |

#### 75.5.3.1 Chip JTAG/Target ID

Each chip in the MCXE31 family includes a unique JTAG ID, which must be changed when instantiated with a different die in an SiP. The next table shows the JTAGC ID for this chip.

Table 584. JTAG/Target ID

| Chip | PRN | DC | PIN | MIC | IDCODE ID | JTAG ID | Target ID |
|---|---|---|---|---|---|---|---|
| MCXE31B | 0 | 26h (38d) | 160h (352d) | Eh (14d) | 1 | 0996_001Dh | 0996_001Dh |
| MCXE315/ MCXE316 | 0 | 26h (38d) | 16Ch (364d) | Eh (14d) | 1 | 0996_C01Dh | 0996_C01Dh |
| MCXE317 | 0 | 26h (38d) | 168h (360d) | Eh (14d) | 1 | 0996_801Dh | 0996_801Dh |

#### 75.5.3.2 JDC

JDC allows you to access two 32-bit data registers by using the JTAG interface and by software running on one of the CPUs in the chip. These registers exchange data between an internal CPU and an external debug tool.

### 75.5.4 Peripheral IPG debug implementation

MCXE31 is a multi-core chip. Any core can control a peripheral instance individually. You define the core's control over a peripheral during application development.

For peripheral halt, all individual cores have the same capability and are gated by a dedicated MDM_AP core halt register field. For more information, see the configurations defined in MDM_AP register descriptions.

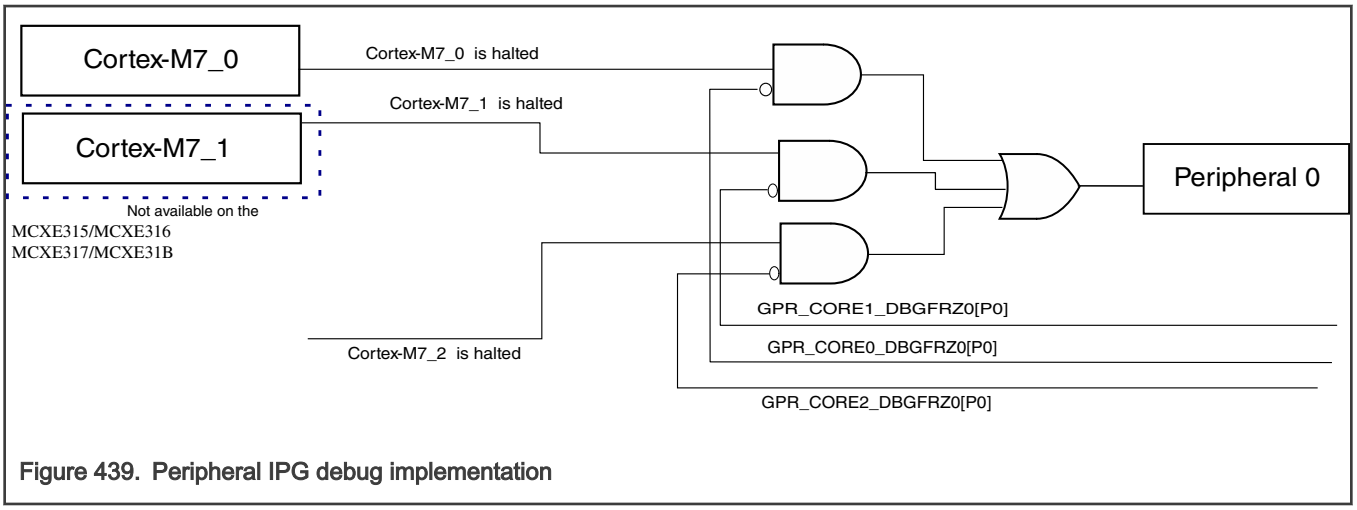



Figure 439. Peripheral IPG debug implementation

### 75.5.5 Application debugging

This section covers the following two cases:

- Case 1: Debugger connected—application debugging from the first instruction
- Case 2: Debugger not connected

#### 75.5.5.1 Application debugging from first instruction

This chip supports debugging from the first instruction on system power-up, destructive reset, functional reset, and standby exit. However, by default, debugging from the first instruction is disabled.

The next figure shows the timing diagram of application debug implementation from first instruction on system power-up or destructive reset.

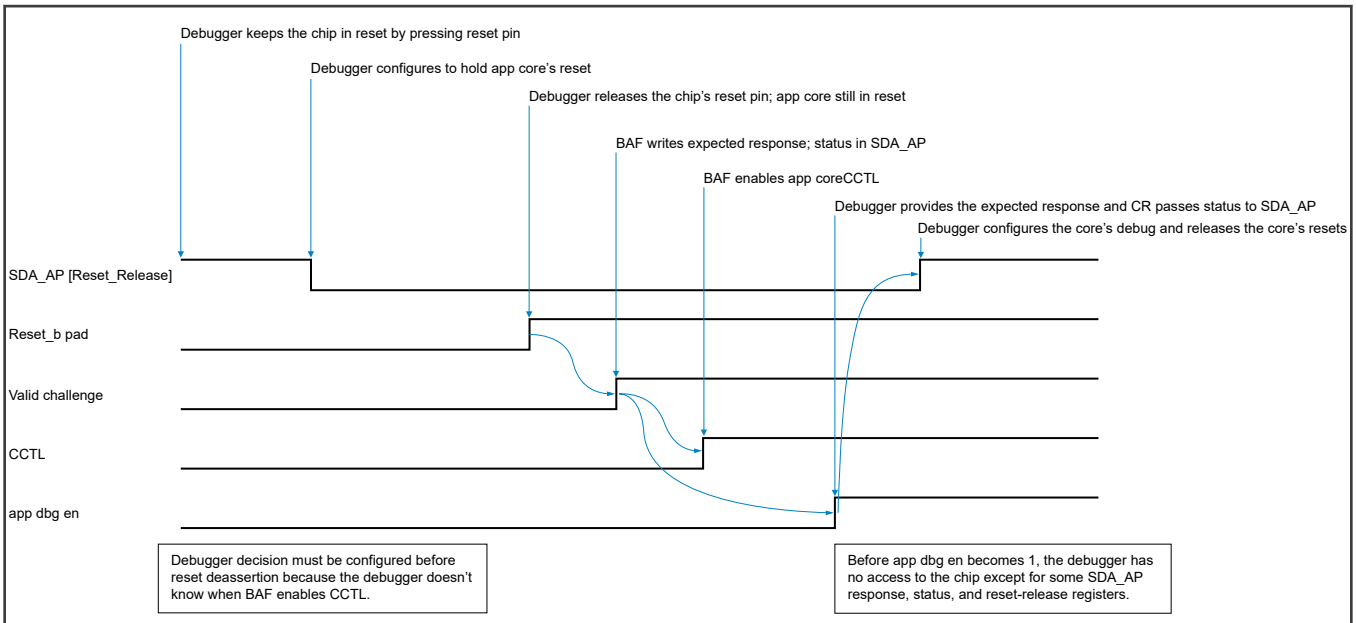



**Figure 440. Application debug implementation from first instruction**

These are the steps involved in enabling application debug from the first instruction on system power-up or destructive reset:

1. After the reset is applied, the debugger provides an option to debug from the first instruction. To do this:
   a. Write 1 to SDAAPRSTCTRL[RSTRELTLCM70] and SDAAPRSTCTRL[RSTRELTLCM7].
   b. Set the reset value of Reset Control (SDAAPRSTCTRL).
2. Cortex M0+ starts after the reset pin is released.
3. Debugger challenge-response (CR) starts.
4. Cortex-M7_0 and Cortex-M7_1 remain in reset until MC_ME's PRTN0_CORE0_PCONF[CCE] field becomes 1 for Cortex-M7_0 and PRTN0_CORE1_PCONF[CCE] field becomes 1 for Cortex-M7_1.
5. When BAF or FW writes 1 to MC_ME's PRTN0_CORE0_PCONF[CCE] field for Cortex-M7_0 and to PRTN0_CORE1_PCONF[CCE] field for Cortex-M7_1, and the debugger CR has passed:
   - If the value of SDAAPRSTCTRL[RSTRELTLCM71] and SDAAPRSTCTRL[RSTRELTLCM70] is 0 for their respective cores, the debugger configures the core debug registers and then enables these fields that start the code execution.
   - If the value of SDAAPRSTCTRL[RSTRELTLCM71] and SDAAPRSTCTRL[RSTRELTLCM70] is 1 for their respective cores, the debugger configures the core debug registers and then enables these fields that start the code execution.

To debug from the first instruction during the low-power debug protocol, the debugger:

1. Writes 0 to SDAAPRSTCTRL[RSTRELTLCM71] and SDAAPRSTCTRL[RSTRELTLCM70] for their respective cores.
2. Writes 1 to MDMAPWIRREL[WTRSTRGM].

### 75.5.5.2 Debugger not connected

If the debugger is not connected, the value of the DBGPWRUP_ACK signal is 0 as there is no DBGPWRUP_REQ. This results in the following scenario:

1. The booting core writes 1 to MC_ME's PRTN0_CORE0_PCONF[CCE] field for Cortex-M7_0 and to PRTN0_CORE1_PCONF[CCE] field for Cortex-M7_1.

2. The application core starts running. In case of Lock-step mode, the checker core executes the same code after a delay of two cycles.

### 75.5.6 Debugger considerations while flash program/erase

The flash programming can be done by application cores as well as debugger. The debugger can program/erase flash using either of the following two options:

1. Loading the program/erase code to SRAM from the debugger and then executing the program/erase sequence by application code from SRAM.
   - Debugger is connected and authenticated.
   - The debugger loads the code in the form of application binary image into the on-chip SRAM.
   - The debugger then initiates the application core to execute the binary from SRAM.
2. Debugger executing the program/erase sequence.
   - Debugger is connected and authenticated.
   - The debugger then initiates the flash programming by reading/writing the registers involved and following the program/erase sequence.

First option is the recommended because of less execution time. In second option, the debugger is the master and since the debug interface is serial, the execution takes more time.

In some cases, you might experience flash program/erase failure due to flash watchdog timeout when debugger executes the flash program/erase sequence due to the serial interface. Therefore, it is recommended to use the first option. In case second option is required, it should be performed in reduced clocking options (Option E and E2 only. For details, refer to the **Clocking details** section in the **Clocking** chapter).

### 75.5.7 SWJ-DP sequence for debug authentication

To perform the SWJ-DP sequence for debug authentication, the debugger:

1. Polls the AUTHSTTS[CHALRDY] field until it indicates the challenge status.
2. Reads Key Challenge (KEYCHAL0 - KEYCHAL7) if the challenge is valid and creates an authenticated 256-bit response key.
3. Writes a 256-bit response key to Key Response (KEYRESP0 - KEYRESP7).
4. Indicates that the response is ready by configuring AUTHCTL[HSEAUTHREQ].
5. Checks the status of Authentication Status (AUTHSTTS) to evaluate if the operation is successful.

   If the authentication process is successful and the debugger has write access to Debug Enable Control (DBGENCTRL), the challenge or response is considered successful too.

**NOTE**

SDA_AP supports challenge and response based on both JTAG and SWJ-DP modes. See SDAAPGENCTRL0[JTAG_CR_EN] for details.

## 75.6 APB memory map

You can access the debug registers via the APB_AP bus. The next table shows all the addresses for the CoreSight APB components and the addressing used for accessing the DAP components via the memory interface. You can also access all APB registers from the processing cores.

Table 585. APB components mapping

| Debug APB component | APBIC base address (access from debugger) | System memory map address (access from cores) | | Memory allocation (KB) | APB-AP0 slot number | Applicability |
|---|---|---|---|---|---|---|
| | | Page number | Base address | | | |
| APB_AP ROM table | F8h | 0 | 4024_00F8h | 4 | APBIC base | All |
| Funnel 0 | 1000h | 0 | 4024_1000h | 4 | 0 | ETM/ITM: MCXE31B<br>ITM: MCXE315/ MCXE316 and MCXE317 |
| Funnel 1 | 2000h | 0 | 4024_2000h | 4 | 1 | MCXE31B |
| Funnel 2 | 3000h | 0 | 4024_3000h | 4 | 2 | MCXE31B |
| CM7_cluster_ETF_ETMI | 4000h | 1 | 4024_4000h | 4 | 3 | MCXE31B |
| CM7_cluster_ETF_ETMD | 5000h | 1 | 4024_5000h | 4 | 4 | MCXE31B |
| HTM ETF | 6000h | 1 | 4024_6000h | 4 | 5 | MCXE31B |
| Shared_system_ETF | 7000h | 1 | 4024_7000h | 4 | 6 | MCXE31B |
| HTM 0 | 8000h | 2 | 4024_8000h | 4 | 7 | MCXE31B |
| HTM 0 CTI | 9000h | 2 | 4024_9000h | 4 | 8 | All |
| TPIU | A000h | 2 | 4024_A000h | 4 | 9 | MCXE31B |
| System SWO | B000h | 2 | 4024_B000h | 4 | 10 | All |
| Timestamp CTL | C000h | 3 | 4024_C000h | 4 | 11 | All |

## 75.7 Trace

### 75.7.1 Trace modules and connectivity

The Trace subsystem:

- Combines trace data from all internal clients that generate trace information
- Includes a 32-bit TPIU
- Includes these components:
  - ATB
  - ATBR
  - CSTF
  - Debug APB
  - ETF
  - TPIU

Multiple options for trace output allow parallel tracing. Trace information can be read from the trace interface or the DAP interface, and traces can be alternatively read out from ETF at a slow speed via APB_AP. Table 586 shows EFT sizes.

**Table 586. ETF sizes**

| FIFO | Memory interface data width (in bits) | Size (KB) |
|---|---|---|
| Cortex-M7 ETM/ITM cluster ETF | 64 | 1 |
| Cortex-M7 ETMD cluster ETF | 128 | 2 |
| HTM ETF | 64 | 1 |
| Shared system ETF | 64 | 2 |

**NOTE**

- The DMA-HTM trace supports four words at 80 MHz. For HTM trace, both DBGENCTRL[GSPNIDEN] and DBGENCTRL[GSPIDEN] must be 1.
- Enabling any one of data trace causes overflow inside ETM and trace packets get dropped. Enabling instruction trace of both the core simultaneously does not cause any overflow.

Control (MDMAPCTL) provides fields to override the speed control (see Table 587 for details) from some of the trace sinks (and TPIU). The complete trace pipeline bandwidth is limited by the slowest sink component. The default settings of these fields allow maximum bandwidth for the TPIU to trace. When tracing to memory (if supported), the fields may need to be changed.

**Table 587. Trace output overrides**

| Trace destination | SWO_override | TPIU_override |
|---|---|---|
| TPIU | 1 | 0 |

The trace sources of the chip are the cores and their related modules. Trace funnels exist for all possible trace clients. For more information on the various components in the trace bus connectivity, see the *CoreSight Components Technical Reference Manual* (available in References). See Table 588 for details on funnel assignments.

The ATBR is integrated to send a shared funnel output across TPIU. This figure illustrates the chip's detailed trace architecture.

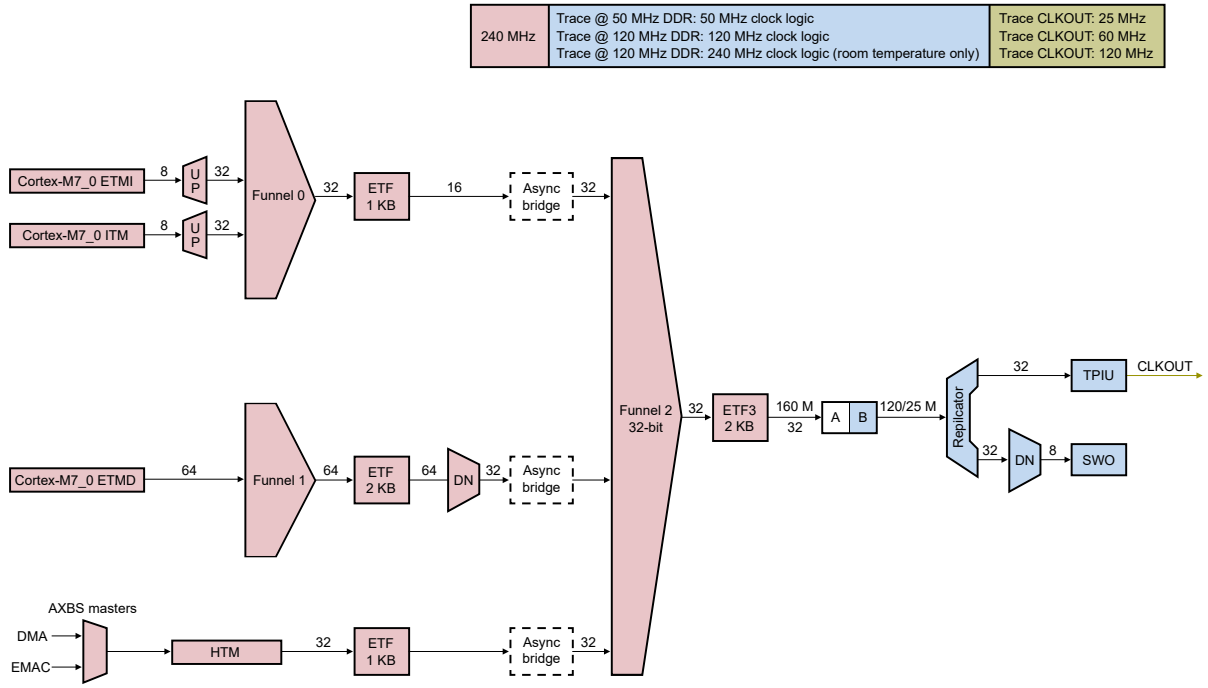



Figure 441. MCXE31B detailed trace architecture

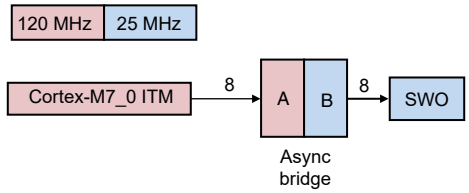



Figure 442. MCXE315/MCXE316 and MCXE317 detailed trace architecture

Table 588. Funnel assignments

| Funnel | Port | Input | MCXE314 Frquency (MHz) | Applicability |
|---|---|---|---|---|
| 0 | 0 | Cortex-M7_0 ETMI | 160 | MCXE31B |
| 0 | 3 | Cortex-M7_1 ITM | 160 | All |
| 0 | 4 | Cortex-M7_2 ETMI | - | MCXE314 |
| 0 | 6-7 | Reserved | - | - |
| 1 | 0 | Cortex-M7_0 ETMD | 160 | MCXE314, MCXE31B |
| 1 | 3-7 | Reserved | - | - |
| 2 | 0 | Cortex-M7 instruction ETF through an asynchronous bridge | 160 | MCXE314, MCXE31B |
| 2 | 1 | Cortex-M7 data ETF through an asynchronous bridge | 160 | MCXE314, MCXE31B |

*Table continues on the next page...*

Table 588. Funnel assignments (continued)

| Funnel | Port | Input | MCXE314 Frquency (MHz) | Applicability |
|---|---|---|---|---|
| 2 | 2 | HTM ETF through an asynchronous bridge | 160 | MCXE314, MCXE31B |
| 2 | 3-7 | Reserved | - | - |

#### 75.7.1.1 Chip's bus trace client

Chips in the MCXE31 family include a bus trace client.

HTM provides the address and data trace information about AXBS buses. The information from an HTM can be used with the debugger to enable easy, accurate debugging on AXBS-based embedded systems. The chip implements an HTM64 configuration to trace 64-bit AXBS masters in the system. To simplify the implementation, instead of tracing the individual ports of various AXBS masters and slaves, which may be running at different frequencies, HTM64 snoops the AXBS crossbar master ports that are all synchronous with a 160 MHz system clock. This table provides details related to the HTM input connectivity.

Table 589. HTM connections

| HTM64 port | AHB crossbar port |
|---|---|
| HTMBUSSELECT0 | M0 AXBS_Lite XBIC (DMA) |
| HTMBUSSELECT1 | M3 AXBS (Ethernet) |

#### 75.7.1.2 TPIU interface

A standard 16-bit parallel TPIU is integrated into the debug subsystem. Chips in the MCXE31 family generate the trace via the trace port.

To prevent instruction trace from being dropped in a multi-core environment, a TPIU throughput of 55.2 MB/s or higher must be maintained. You could use these pin and frequency combinations:

- Four high-speed data + one clock pads @120 MHz (240 Mbit/s throughput per pad) totaling 120 MB/s
- 16 low-speed data + one clock pads @25 MHz (50 Mbit/s throughput per pad) totaling 100 MB/s

High-end chips from the MCXE31 family support at least 56 MB/s of trace throughput on the TPIU interface.

Table 590. Throughput

| Core | Throughput value | Applicability |
|---|---|---|
| Cortex-M7_0 | 27.6 MB/s | All |
| Cortex-M7_1 | 27.6 MB/s | MCXE31B |
| Total | 82.8 MB/s | |

#### 75.7.1.3 TPIU flush

To allow the chip to enter Standby mode, software executes an WFI instruction indicating Standby entry. When the chip enters this mode, MC_RGM asserts a trace flush request to the TPIU and waits for a trace flush done signal from the TPIU before requesting MC_PCU to proceed. The TPIU and debug infrastructure clocks can be gated after this. Because the TPIU is sourced from system clock, the clock must not be gated until this point.

### 75.7.2 Trace assignments

ETF combines trace information from multiple clients into a single stream of trace data. This table shows the CoreSight funnel input assignments.

**Table 591. CoreSight funnel assignments**

| Instance name | Port assignment | Applicability |
|---|---|---|
| Funnel 0 (CSTF0) | Cortex-M7_0 ETMI | MCXE31B |
| | Cortex-M7_0 ITM | All |
| | Cortex-M7_1 ETMI | MCXE31B |
| | Cortex-M7_1 ITM | MCXE317 |
| Funnel 1 (CSTF1) | Cortex-M7_0 ETMD | MCXE31B |
| Funnel 2 (CSTF2) | Funnel 0 | MCXE31B |
| | Funnel 1 | MCXE31B |
| | HTM | MCXE31B |

## 75.8 Embedded cross trigger (ECT)

ECT allows multi-core run control and trace cross-triggering, such as synchronous stop-start for all cores or trigger trace on a trigger event from another core or module. See the *CoreSight Components Technical Reference Manual* (available in References) for detailed information on ECT.

ECT architecture involves CTMs and CTIs. The CTIs provide a cross-triggering interface between the cores and other debug and trace modules. The channels of these CTIs are interconnected using CTMs, as shown in this figure.

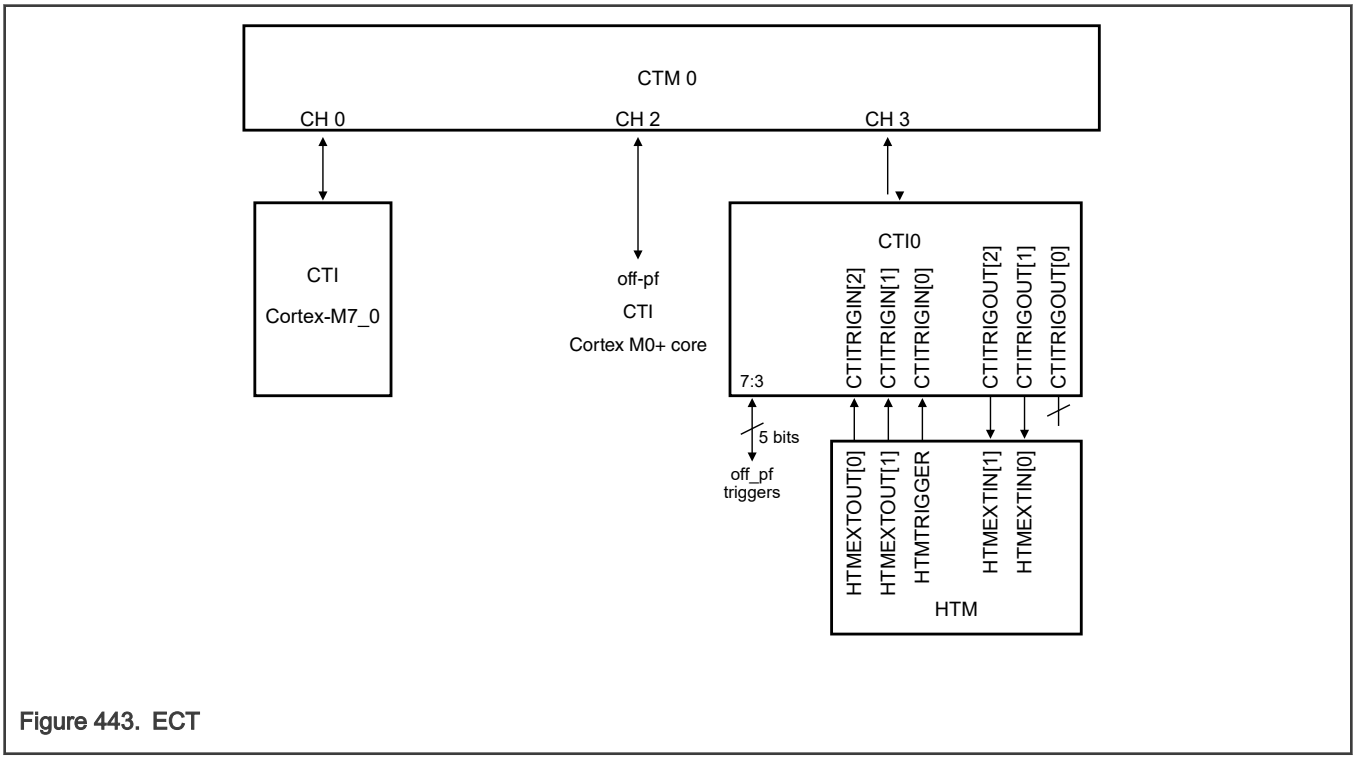



**Figure 443. ECT**

### 75.8.1 CTI assignments

**Table 592. CTI assignments**

| CTI instance | Trigger number | Trigger in | Trigger out |
|---|---|---|---|
| CTI_Cortex-M7_0 | 7 | ETM event output 3 | Processor restart |
| | 6 | ETM event output 2 | ETM event input 3 |
| | 5 | ETM event output 1 | ETM event input 2 |
| | 4 | ETM event output 0 | ETM event input 1 |
| | 3 | DWT comparator output 2 | ETM event input 0 |
| | 2 | DWT comparator output 1 | Interrupt request 1 |
| | 1 | DWT comparator output 0 | Interrupt request 0 |
| | 0 | Processor halted | Processor debug request |
| CTI_Cortex-M7_1 | 7 | ETM event output 3 | Processor restart |
| | 6 | ETM event output 2 | ETM event input 3 |
| | 5 | ETM event output 1 | ETM event input 2 |
| | 4 | ETM event output 0 | ETM event input 1 |
| | 3 | DWT comparator output 2 | ETM event input 0 |
| | 2 | DWT comparator output 1 | Interrupt request 1 |
| | 1 | DWT comparator output 0 | Interrupt request 0 |
| | 0 | Processor halted | Processor debug request |
| CTI_Cortex-M7_2 | 7 | ETM event output 3 | Processor restart |
| | 6 | ETM event output 2 | ETM event input 3 |
| | 5 | ETM event output 1 | ETM event input 2 |
| | 4 | ETM event output 0 | ETM event input 1 |
| | 3 | DWT comparator output 2 | ETM event input 0 |
| | 2 | DWT comparator output 1 | Interrupt request 1 |
| | 1 | DWT comparator output 0 | Interrupt request 0 |
| | 0 | Processor halted | Processor debug request |
| CTI_0 | 7 | Reserved (grounded) | Reserved (no connection) |
| | 6 | ETF_3 full | ETF_3 trigger input |
| | 5 | ETF_2 full | ETF_2 trigger input |
| | 4 | ETF_1 full | ETF_1 trigger input |
| | 3 | ETF_0 full | ETF_0 trigger input |
| | 2 | HTM trigger out 2 | HTM trigger in 2 |

*Table continues on the next page...*

Table 592. CTI assignments (continued)

| CTI instance | Trigger number | Trigger in | Trigger out |
|---|---|---|---|
| | 1 | HTM trigger out 1 | HTM trigger in 1 |
| | 0 | Reserved | HTM trigger in 0 |

## 75.9 Low-power debug handshake protocol

The debugger must perform this sequence to enter or exit Standby mode, if the debugger handshake is enabled:

1. Power on DAP.

   The debugger connection is established.

2. Configure the Debug subsystem for the relevant operations.
3. Write 1 to MDMAPWIREN[LWPWREN] to gate entry into Standby mode with the debugger handshake.
4. Disable POR_WDG for monitoring the entry to or exit from Standby mode by writing 1 to DCM's DCMRWP1[8] field. This is required for low-power debug handshake because debugger configurations can be more time consuming than the POR_WDG threshold levels and can raise a false POR_WDG event.
5. Write 1 to DCMRWF1[STANDBY_IO_CONFIG] in the Device Configuration Module (DCM). This causes padkeeping to be disabled on standby entry itself without any software dependency. This is needed in case of low-power debug since otherwise the padkeeping on TDO would result in no debugger communication. In other cases, DCMRWF1[STANDBY_IO_CONFIG] should be configured as 0 before standby entry.
6. Initiate entry into Standby mode. See the "Power Management" chapter for the Standby mode entry sequence and configurations.
7. Identify whether the low-power debug is enabled on the low-power entry by reading MC_RGM's LPDEBUG[LP_DBG_EN] field.
   - If low-power debug is enabled and TPIU is enabled too, trace flush starts, and the debugger acknowledges the following:
     - Low-power debug traces
     - DAP-related configuration reception and context saving

       After this, the chip enters Standby mode.
   - If TPIU is disabled:
     - The debugger acknowledges DAP-related configuration reception and context saving by writing 1 to MDMAPWIRREL[PRVNTRSTRGM].

       After this, the chip enters Standby mode.
   - If low-power debug is disabled, the chip enters Standby mode without waiting for debugger acknowledgment.

   On any wakeup event, the chip starts the Standby mode exit sequence described in the "Power Management" chapter. After the chip exits Standby mode, the debugger connection is restored. By this time, the debugger is already aware that it has enabled the low-power debug handshake. In this case, the debugger must poll MDMAPSTTS[DESTRST] to check whether:
   - The debug infrastructure is out of reset.
   - The DAP connection can be established.

   After DAP is powered on (the debugger connection is established), the debugger:
   a. Reconfigures the debug and trace attributes using the fields in DBGENCTRL
   b. Writes 1 to MDMAPWIRREL[WTRSTRGM] after the debugger trace context is restored

      The chip exits Standby mode.

If low-power debug is not configured, the chip exits Standby mode without waiting for the debugger to write to MDMAPWIRREL[WTRSTRGM].

8. Reconfigure the debug and trace configuration in SDA_AP register descriptions after the chip exits Standby mode.

**NOTE**

If the chip wakes up from Standby mode through pad reset, the debugger must perform a CR again.

## 75.10 Debug resets

The debug subsystem follows this sequence on the source of reset:

1. POR resets the complete debug logic.
2. The destructive reset resets all types of debug logic except JTAGC.

Conversely, the debug subsystem can generate a system reset using these mechanisms:

- System destructive reset defined in Control (MDMAPCTL) that allows the debugger to provide the destructive reset to the system. The debugger loses connection to the system with this reset.
- System functional reset defined in Control (MDMAPCTL) that allows the debugger to hold the system in functional reset.

To program various debug registers, the functional clocks must be enabled. All debug and trace components must be on destructive reset.

## 75.11 Debug across device LifeCycles

The debug access to the system is available in early lifecycles and subsequently based on NVM configuration settings.

After the debug is set to 'Trusted', the system access is allowed after a successful authorization step between the debugger and the system.

The authorization provides provisions to allow debug for Cortex M0+ core and Cortex M7 cores.

Following table shows the debug access based on various configuration bits.

**Table 593. Debug access based on LifeCycle and bit configurations**

| LifeCycle | Configuration bits | | | | | | | | | Debug access | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | HSE_DIS_CR_MCU_PROD | HSE_DIS_CUS | DBG_EN_OEM | APP_DIS_OEM | HSE_DIS_OEM | DBG_EN_FLD | APP_DIS_FLD | HSE_DIS_FLD | HSE_DIS_PRE_FA | HSE Debug | Appl Core Debug |
| MCU_PROD | 0 | - | - | - | - | - | - | - | - | Trusted | Open |
| | 1 | - | - | - | - | - | - | - | - | Open | Open |
| CUST_DEL | - | 0 | - | - | - | - | - | - | - | Trusted | Open |
| | - | 1 | - | - | - | - | - | - | - | Disabled | Open |

*Table continues on the next page...*

Table 593. Debug access based on LifeCycle and bit configurations (continued)

| LifeCycle | Configuration bits | | | | | | | | | Debug access | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | HSE_DIS_CR_MCU_PROD | HSE_DIS_CUS | DBG_EN_OEM | APP_DIS_OEM | HSE_DIS_OEM | DBG_EN_FLD | APP_DIS_FLD | HSE_DIS_FLD | HSE_DIS_PRE_FA | HSE Debug | Appl Core Debug |
| OEM_PROD | - | - | 0 | - | - | - | - | - | - | Closed | Closed |
| | - | - | 1 | 0 | 0 | - | - | - | - | Trusted | Trusted |
| | - | - | 1 | 0 | 1 | - | - | - | - | Disabled | Trusted |
| | - | - | 1 | 1 | - | - | - | - | - | Disabled | Disabled |
| IN_FIELD | - | - | - | - | - | 0 | - | - | - | Closed | Closed |
| | - | - | - | - | - | 1 | 0 | 0 | - | Trusted | Trusted |
| | - | - | - | - | - | 1 | 0 | 1 | - | Disabled | Trusted |
| | - | - | - | - | - | 1 | 1 | - | - | Disabled | Disabled |
| PRE_FA | - | - | - | - | - | - | - | - | 0 | Trusted | Trusted |
| | - | - | - | - | - | - | - | - | 1 | Disabled | Trusted |
| FA | - | - | - | - | - | - | - | - | - | Open | Open |

**NOTE**

1. **Closed:** Debug is not possible in this state (not even if keys are known - no authentication is possible)
2. **Open:** Debug is always possible
3. **Trusted:** Debug is possible after successful authentication (Challenge/Response handshake with correct credentials)
4. **Disabled:** Debug has been explicitly disabled via burning of a fuse in that LC, otherwise behaves as Closed
5. HSE_DIS_CR_MCU_PROD bit disables the challenge-response based mechanism, making the debug open when asserted.
6. During functional reset, the debug-enable will retain its last status, while DCM scans NVM for lifecycle and DCF values. Debug status will be re-evaluated once dcm_done gets 1.
7. During temporary advancement of lifecycle, debug-en will immediately reflect the status based on updated lifecycle/DCF bits.

## 75.12 Pin interface

This table presents a summary of functional and power pins that are used for debugging purposes.

Table 594. Pin interface

| Pin type | Pins | Number of pins (balls)[1] | Nominal voltage |
|---|---|---|---|
| Functional JTAG | JCOMP, TCK, TMS, TDI, TDO | Five (only TDI and TDO can be multiplexed with GPIO or other functions) | 3.3 V, 5 V |
| Functional trace pins (parallel trace) | TRACE_CLK | One | 3.3 V, 5 V |
| | TRACE_D[15:0] | 16 | 3.3 V, 5 V |
| Ground | VSS | See the IOMUX file attached to this document for information on the number of VSS pins in various packages. | 0 V |

1. The terms pins and balls are used interchangeably. Some chip packages include pins and others include balls.

**NOTE**

Fast Trace pad TRACE_D[15:0] is available only in MCXE314.

### 75.12.1 Debug port and pin descriptions

The pads to which the debug signals are mapped operate using the JTAG functionality out of reset but can later be reassigned to their alternate functionalities. TDI and TDO can operate as alternate GPIO functions. See this table for pin assignments in different modes.

Table 595. Debug port pins

| Pin name | JTAG debug port | | Internal pullup or pulldown logic |
|---|---|---|---|
| | Type | Description | |
| TMS | I/O | JTAG test mode selection (TMS) | Pullup |
| TCK/SWCLK | I | JTAG test clock (TCK) | Pulldown |
| TDI | I | JTAG test data input (TDI) | Pullup |
| TDO/TRACESWO | O | JTAG test data output (TDO) | Not connected |

### 75.12.2 Trace port pin descriptions

This chip generates trace via TPIU that transmits the trace data out of the chip over a parallel trace port. The trace port consists of an ETM trace clock and 16 parallel trace data outputs. The Arm optional trace port control (TRACECTRL), debug request (DBGRQ), and debug acknowledge (DBGACK) signals are not implemented.

Table 596. Trace output port pins

| Pin name | Description |
|---|---|
| TRACE_DATA00 | ETM parallel trace data output 01 |
| TRACE_DATA01 | ETM parallel trace data output 01 |
| TRACE_DATA02 | ETM parallel trace data output 02 |
| TRACE_DATA03 | ETM parallel trace data output 03 |

*Table continues on the next page...*

Table 596. Trace output port pins (continued)

| Pin name | Description |
|---|---|
| TRACE_DATA04 | ETM parallel trace data output 04 |
| TRACE_DATA05 | ETM parallel trace data output 05 |
| TRACE_DATA06 | ETM parallel trace data output 06 |
| TRACE_DATA07 | ETM parallel trace data output 07 |
| TRACE_DATA08 | ETM parallel trace data output 08 |
| TRACE_DATA09 | ETM parallel trace data output 09 |
| TRACE_DATA10 | ETM parallel trace data output 10 |
| TRACE_DATA11 | ETM parallel trace data output 11 |
| TRACE_DATA12 | ETM parallel trace data output 12 |
| TRACE_DATA13 | ETM parallel trace data output 13 |
| TRACE_DATA14 | ETM parallel trace data output 14 |
| TRACE_DATA15 | ETM parallel trace data output 15 |
| TRACE_CLK | ETM parallel trace clock output |

**NOTE**

ETM is supported in MCXE314 only

**NOTE**

By default, Rx pins float and are not pulled inside. An internal active pulldown logic exists only when you enable Rx via software (IBE).

## 75.13 Timestamp distribution network

The timestamp distribution network uses CoreSight timestamp components to generate a 48-bit timestamp value for the trace sources, as shown in the next figure. The CoreSight timestamp generator generates a 64-bit counter value, but only the least significant 48 bits are distributed to the trace sources. A 7-bit narrow timestamp is derived from the 48-bit timestamp and distributed to the trace client, where a decoder regenerates the 48-bit timestamp. The generator must be programmed, and you could find the related programming information in the *CoreSight Components Technical Reference Manual* (available in References).

This chip supports 48-bit timestamping. The generator operates at a frequency of 160 MHz and gives a 64-bit timestamp value. The Cortex-M7_0 and Cortex-M7_1 cores operate at a frequency of 160 MHz.

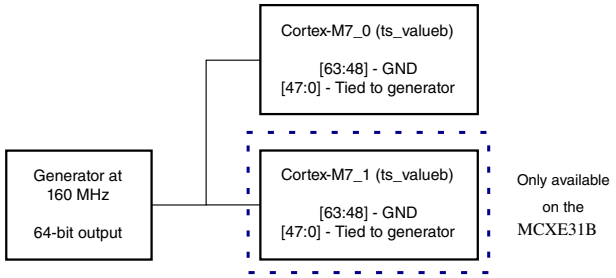



Figure 444. Timestamp distribution network

## 75.14 Peripheral debug freeze register descriptions

Implement the logic provided in this section for each peripheral instance supporting the debug operation.

**NOTE**

Debug freeze is enabled for all the peripherals so that as soon as the core halts, peripherals can be frozen to support debugging from the first instruction.

For register details, refer to the following registers in the Device Configuration Module (DCM):

- Read Write GPR On Destructive Reset Register (DCMRWD6)
- Read Write GPR On Destructive Reset Register (DCMRWD7)
- Read Write GPR On Destructive Reset Register (DCMRWD8)
- Read Write GPR On Destructive Reset Register (DCMRWD9)

## 75.15 MDM_AP register descriptions

The debugger has access to the status and control elements implemented as registers in MDM_AP, which is selected by APSEL (6h) on the DAP bus. These registers provide additional control and status information for typical debug, cross-triggering, and run-control scenarios. Also, the register fields provide a way for the debugger to get the updated status of the core without initiating a bus transaction across the crossbar switch, thus remaining less intrusive during a debug session.

MDM_AP is accessible as DAP (see DAP TAP for details).

### 75.15.1 MDM_AP memory map

MDM_AP base address: 4025_0600h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Status (MDMAPSTTS) | 32 | R | 5000_0000h |
| 4h | Control (MDMAPCTL) | 32 | RW | 0640_0000h |
| 30h | WIR Enable (MDMAPWIREN) | 32 | RW | 0000_0000h |
| 34h | WIR Status (MDMAPWIRSTTS) | 32 | R | 0000_0000h |
| 38h | WIR Release (MDMAPWIRREL) | 32 | RW | 0000_0000h |
| FCh | Identity (ID) | 32 | R | 001C_0030h |

### 75.15.2 Status (MDMAPSTTS)

**Offset**

| Register | Offset |
|---|---|
| MDMAPSTTS | 0h |

### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | Reserved | Reserved | CM70 DBG... | Reserved | | | | Reserved | Reserved | Reserved | CM70 SLP... | Reserved | Reserved | Reserved | CM70 DPS... |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | Reserved | Reserved | CM70 HLT | Reserved | | | | | | | | | FUNC RST | DEST RST | Reserved |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>— | Reserved |
| 29<br>— | Reserved |
| 28<br>CM70DBGRSTRD | Cortex-M7_0 Debug Restarted<br>Indicates if Cortex-M7_0 has returned to Normal mode from Debug mode.<br>0b - In Debug mode<br>1b - In Normal mode |
| 27-24<br>— | Reserved |
| 23<br>— | Reserved |
| 22<br>— | Reserved |
| 21<br>— | Reserved |
| 20<br>CM70SLPNG | Cortex-M7_0 Sleeping<br>Indicates if Cortex-M7_0 is in Sleep mode. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Not in Sleep mode<br>1b - In Sleep mode |
| 19<br>— | Reserved |
| 18<br>— | Reserved |
| 17<br>— | Reserved |
| 16<br>CM70DPSLP | Cortex-M7_0 Deep Sleep<br>Indicates if Cortex-M7_0 is in Deep Sleep mode.<br>0b - Not in Deep Sleep mode<br>1b - In Deep Sleep mode |
| 15<br>— | Reserved |
| 14<br>— | Reserved |
| 13<br>— | Reserved |
| 12<br>CM70HLT | Cortex-M7_0 Halted<br>Indicates if Cortex-M7_0 is halted because of entry into Debug mode.<br>0b - Core is not halted<br>1b - Core is halted |
| 11-3<br>— | Reserved |
| 2<br>FUNCRST | Functional Reset<br>Indicates the system reset state.<br>0b - Not in functional reset<br>1b - In functional reset |
| 1<br>DESTRST | Destructive Reset<br>Indicates the system reset state. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Not in destructive reset<br>1b - In destructive reset |
| 0<br>— | Reserved |

### 75.15.3 Control (MDMAPCTL)

#### Offset

| Register | Offset |
|---|---|
| MDMAPCTL | 4h |

#### Function

Allows the debugger:

- To give the destructive reset to the system. A system destructive reset enables this. The debugger also loses connection to the system with this reset.
- To hold the system in functional reset. A system functional reset enables this.

**NOTE**

The trace functionality on trace pins is selected with this register and is retained across the functional reset.

#### Diagram

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Reserved | Reserved | Reserved | CM70DBG... | Reserved | | | | | SWOOVRD | Reserved | Reserved | Reserved | | Reserved | Reserved |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | POR_WDG... | Reserved | Reserved | | Reserved | Reserved | Reserved | CM70DBG... | Reserved | | SYSFUNC... | SYSRESE... | Reserved | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### Fields

| Field | Function |
|---|---|
| 31 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 30<br>— | Reserved |
| 29<br>— | Reserved |
| 28<br>CM70DBGRSRT | Cortex-M7_0 Debug Restart<br>Indicates if a request to Cortex-M7_0 to leave the debug halt state is asserted.<br>0b - Normal operation<br>1b - Request asserted |
| 27-23<br>— | Reserved |
| 22<br>SWOOVRD | SWO Override<br>Indicates if the SWO trace response is overriden. When TPIU is not the selected trace sink target, you must override the trace response.<br>0b - Not overridden, and SWO generates the trace response<br>1b - Is overridden |
| 21<br>— | Reserved |
| 20<br>— | Reserved |
| 19-18<br>— | Reserved |
| 17<br>— | Reserved |
| 16<br>— | Reserved |
| 15<br>POR_WDG_DIS | Power Watchdog Status<br>When a 0 is written to this field, the power watchdog is disabled in case of MDM FUNC reset or JTAG reset. When a 1 is written to this field, the power watchdog is enabled in case of MDM FUNC reset or JTAG reset. |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | 0b - Power watchdog is disabled<br>1b - Power watchdog is enabled |
| 14<br>— | Reserved |
| 13-12<br>— | Reserved |
| 11<br>— | Reserved |
| 10<br>— | Reserved |
| 9<br>— | Reserved |
| 8<br>CM70DBGREQ | Cortex-M7_0 Debug Request<br>Drives the EDBGREQ input for Cortex-M7_0 and indicates if the debug request is generated. When the core goes into debug state, the field acknowledges that with a halted output signal (see MDMAPSTTS[CM70HLT]). If the core is in Stop mode, this field is used to wake up the core and transition it to the debug halt state.<br>0b - Debug request is not generated<br>1b - Debug request is generated |
| 7-6<br>— | Reserved |
| 5<br>SYSFUNCRST | System Functional Reset<br>Asserts or deasserts functional reset to the chip. The debugger maintains connection with the chip.<br>0b - Deasserted<br>1b - Asserted |
| 4<br>SYSRESETREQ | System Destructive Reset<br>Asserts or deasserts destructive reset to the chip. The debugger also loses connection with this reset. When this field is reset, the entire system comes out of reset.<br>0b - Deasserted<br>1b - Asserted |
| 3-0 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

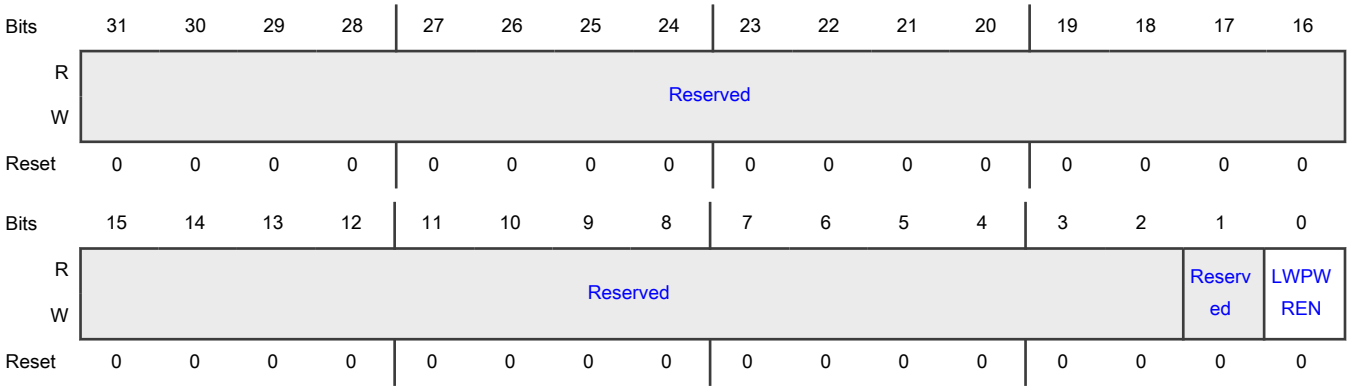
| Field | Function |
|---|---|
| — | |

## 75.15.4 WIR Enable (MDMAPWIREN)

### Offset

| Register | Offset |
|---|---|
| MDMAPWIREN | 30h |

### Diagram




### Fields

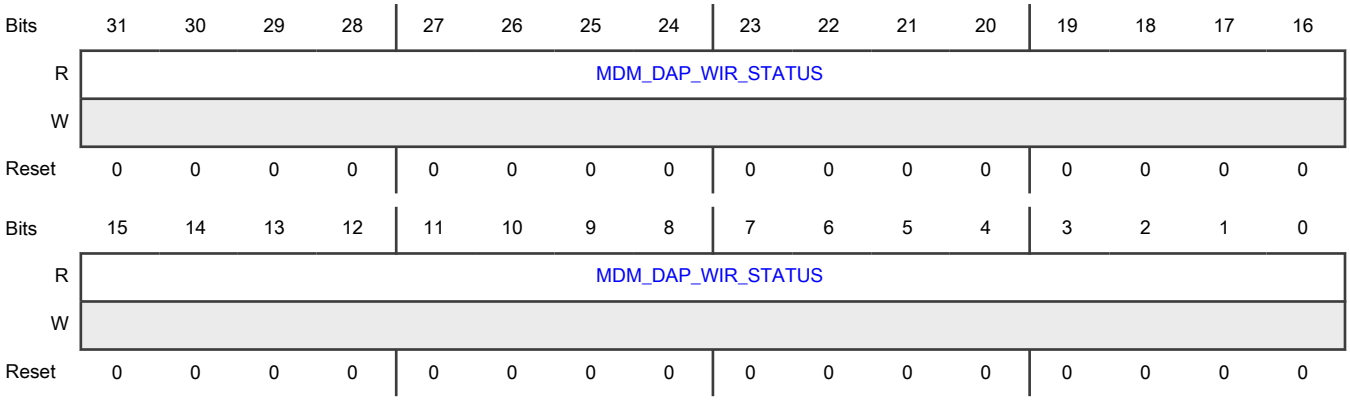
| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>— | Reserved |
| 0<br>LWPWREN | Low Power Debug Enable<br>Enables or disables low-power debug.<br>When the debugger writes 1 to this field, MC_RGM's LPDEBUG[LP_DBG_EN] field becomes 1 too.<br>0b - Disabled<br>1b - Enabled |

## 75.15.5 WIR Status (MDMAPWIRSTTS)

### Offset

| Register | Offset |
|---|---|
| MDMAPWIRSTTS | 34h |

### Diagram




### Fields

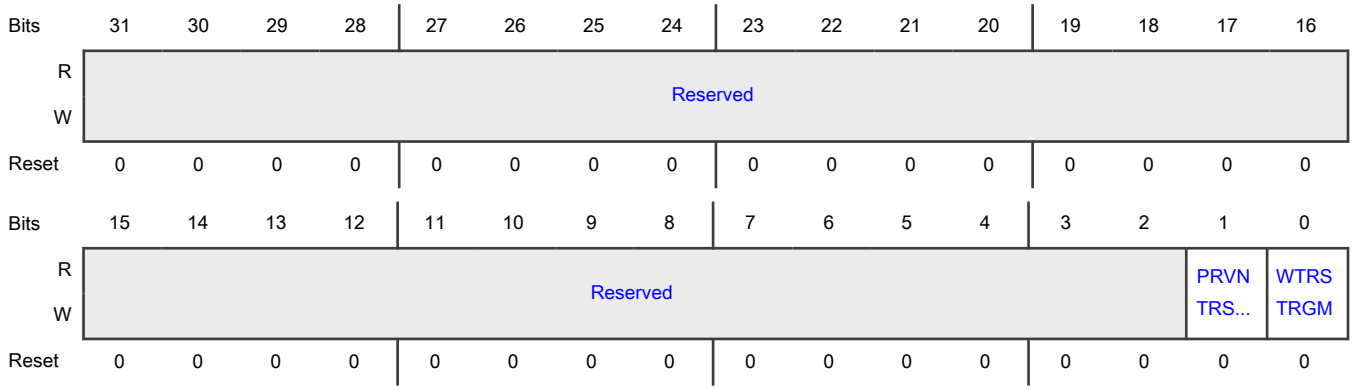
| Field | Function |
|---|---|
| 31-0<br>MDM_DAP_WIR_STATUS | MDM_AP WIR Status<br>0b - Indicates WIR status<br>1b - Does not indicate WIR status |

## 75.15.6 WIR Release (MDMAPWIRREL)

### Offset

| Register | Offset |
|---|---|
| MDMAPWIRREL | 38h |

### Diagram




### Fields

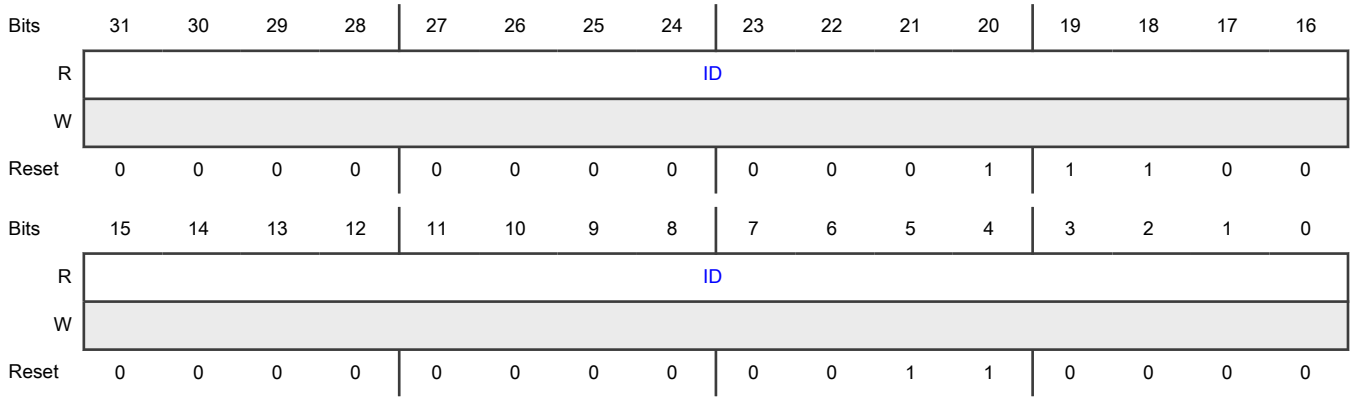
| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>PRVNTRSTRGM | Prevent Reset<br>Indicates if MC_RGM is prevented from generating reset.<br>After TPIU flush, this field prevents MC_RGM from generating reset even if MC_RGM receives an acknowledge response from TPIU. This is valid for low-power entry.<br>0b - Normal operation<br>1b - MC_RGM prevented |
| 0<br>WTRSTRGM | Wait In Reset B<br>Indicates if waiting of MC_RGM from generating reset is supported.<br>On exiting Standby mode, MC_RGM waits until the debugger writes to another field in the MDM_AP register to allow it to exit reset.<br>0b - Normal operation<br>1b - Wait supported |

## 75.15.7 Identity (ID)

### Offset

| Register | Offset |
|---|---|
| ID | FCh |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>ID | Identity |

## 75.16 SDA_AP register descriptions

The debugger and system have access to secure authentication control and status, implemented as registers in SDA_AP on the DAP bus. These registers provide authentication control, authentication status, key exchange information, and debug enable controls.

### 75.16.1 SDA_AP memory map

SDA_AP base address: 4025_4700h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Authentication Status (AUTHSTTS) | 32 | R | 6000_0004h |
| 4h | Authentication Control (AUTHCTL) | 32 | RW | 0000_0000h |
| 10h - 2Ch | Key Challenge (KEYCHAL0 - KEYCHAL7) | 32 | R | 0000_0000h |
| 40h - 5Ch | Key Response (KEYRESP0 - KEYRESP7) | 32 | RW | 0000_0000h |
| 70h | User Identification 0 (UID0) | 32 | R | 0000_0000h |
| 74h | User Identification 1 (UID1) | 32 | R | 0000_0000h |
| 80h | Debug Enable Control (DBGENCTRL) | 32 | RW | See section |
| 90h | Reset Control (SDAAPRSTCTRL) | 32 | RW | 0600_0000h |
| A0h | SDA_AP Generic Status (SDAAPGENSTATUS0) | 32 | R | 0000_0000h |
| A4h | Generic Control 0 (SDAAPGENCTRL0) | 32 | RW | 0000_0000h |
| B0h | SDA_AP Generic Status (SDAAPGENSTATUS1) | 32 | R | 0000_0000h |

*Table continues on the next page...*

*Table continued from the previous page...*

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| C0h | SDA_AP Generic Status (SDAAPGENSTATUS2) | 32 | R | 0000_0000h |
| D0h | SDA_AP Generic Status (SDAAPGENSTATUS3) | 32 | R | 0000_0000h |
| E0h | SDA_AP Generic Status (SDAAPGENSTATUS4) | 32 | R | 0000_0000h |
| FCh | Identity (ID) | 32 | R | 001C_0040h |

## 75.16.2 Authentication Status (AUTHSTTS)

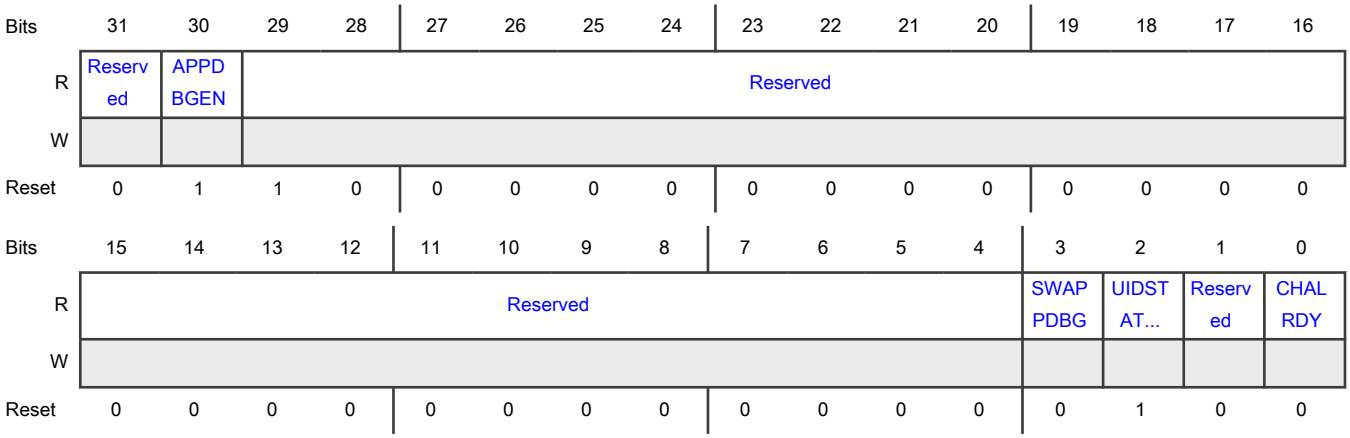
### Offset

| Register | Offset |
|---|---|
| AUTHSTTS | 0h |

### Function

Indicates the status of the authentication process.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31<br>— | Reserved |
| 30<br>APPDBGEN | Application Debug Enabled or Disabled<br>Indicates: |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| | • The status of application debug<br>• Whether CR is satisfied<br>• Whether access to other APs is allowed<br>0b - Application debug disabled<br>1b - Application debug enabled |
| 29-4<br>— | Reserved |
| 3<br>SWAPPDBG | Software Application Debug<br>Indicates:<br>• The status of software application debug<br>• Whether access to debug controls is allowed<br>0b - Software application debug disabled<br>1b - Software application debug enabled |
| 2<br>UIDSTATUS | User Identification Status<br>Indicates:<br>• The status of UID<br>• Whether DCM has finished reading the flash memory user section<br>0b - UID is not ready and is invalid<br>1b - UID is ready and is valid |
| 1<br>— | Reserved |
| 0<br>CHALRDY | Challenge Ready<br>Indicates:<br>• The status of challenge ready when the value of export control is 0<br>• The status of the DCM_DONE signal if the value of export control is 1<br>0b - Challenge is not ready<br>1b - Challenge is ready |

## 75.16.3 Authentication Control (AUTHCTL)

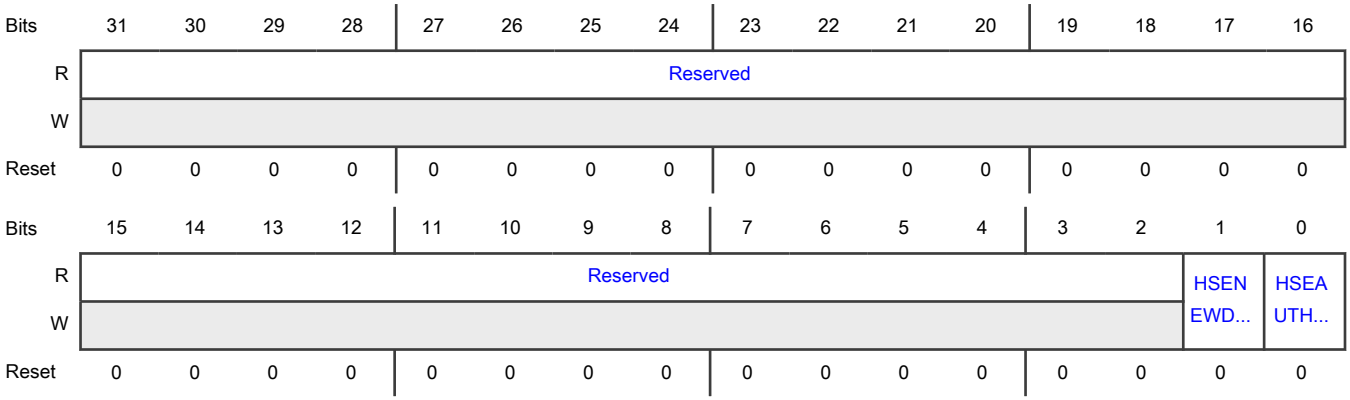
### Offset

| Register | Offset |
|---|---|
| AUTHCTL | 4h |

### Function

Controls the authentication process.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>HSENEWDATACTL | New Data Control<br>Indicates that the debugger has consumed the data registers. It is alright for the core to provide new data.<br>0b - Does not indicate that the debugger has consumed the data registers<br>1b - Indicates that the debugger has consumed the data registers |
| 0<br>HSEAUTHREQ | Debug Enablement Authentication Request<br>Indicates that all key values are written and the chip can start the authentication request.<br>0b - Does not start the authentication request<br>1b - Starts the authentication request |

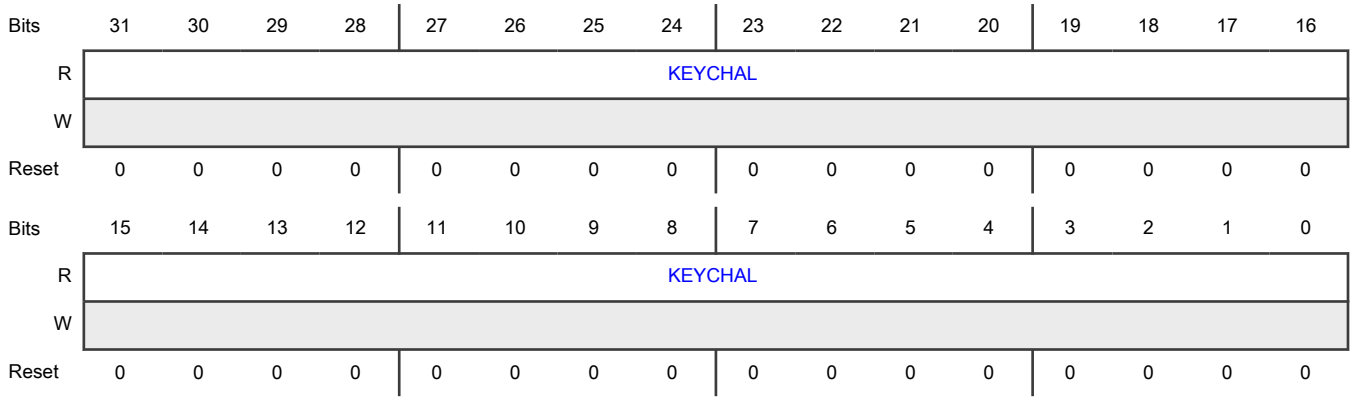
## 75.16.4 Key Challenge (KEYCHAL0 - KEYCHAL7)

### Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| KEYCHALa | 10h + (a × 4h) |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>KEYCHAL | Debug Enablement Key Challenge |

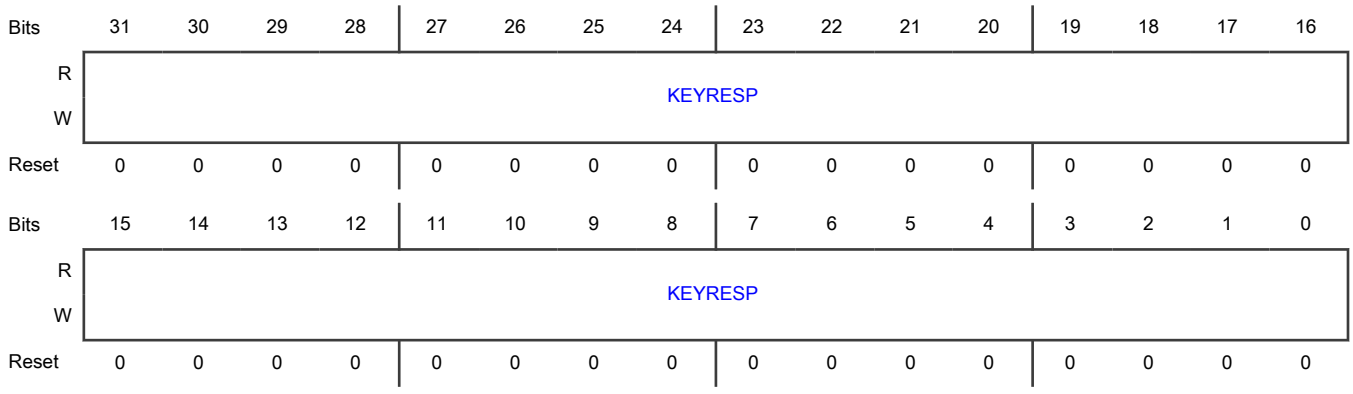
## 75.16.5 Key Response (KEYRESP0 - KEYRESP7)

### Offset

For a = 0 to 7:

| Register | Offset |
|---|---|
| KEYRESPa | 40h + (a × 4h) |

### Diagram



### Fields

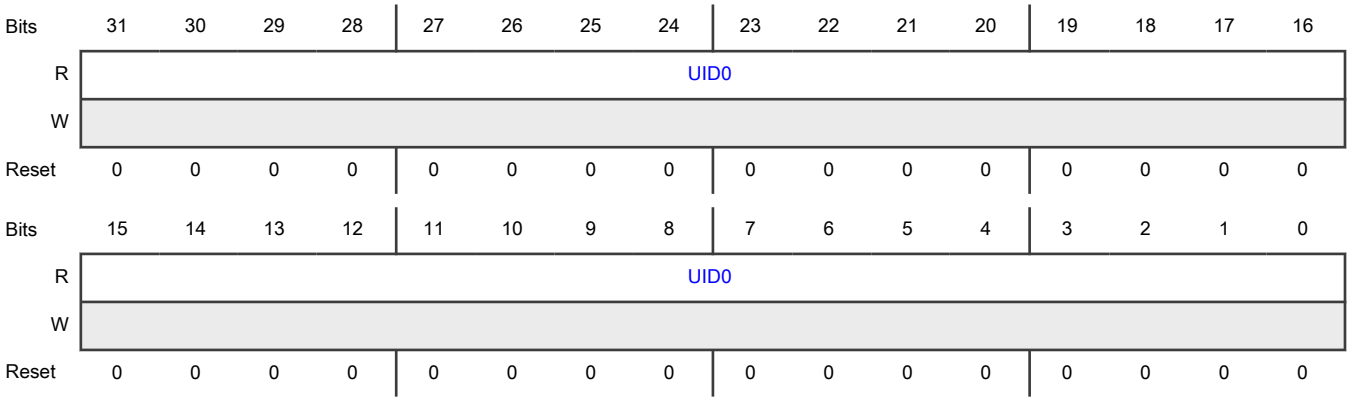
| Field | Function |
|---|---|
| 31-0<br>KEYRESP | Debug Enablement Key Response |

## 75.16.6 User Identification 0 (UID0)

### Offset

| Register | Offset |
|---|---|
| UID0 | 70h |

### Diagram




### Fields

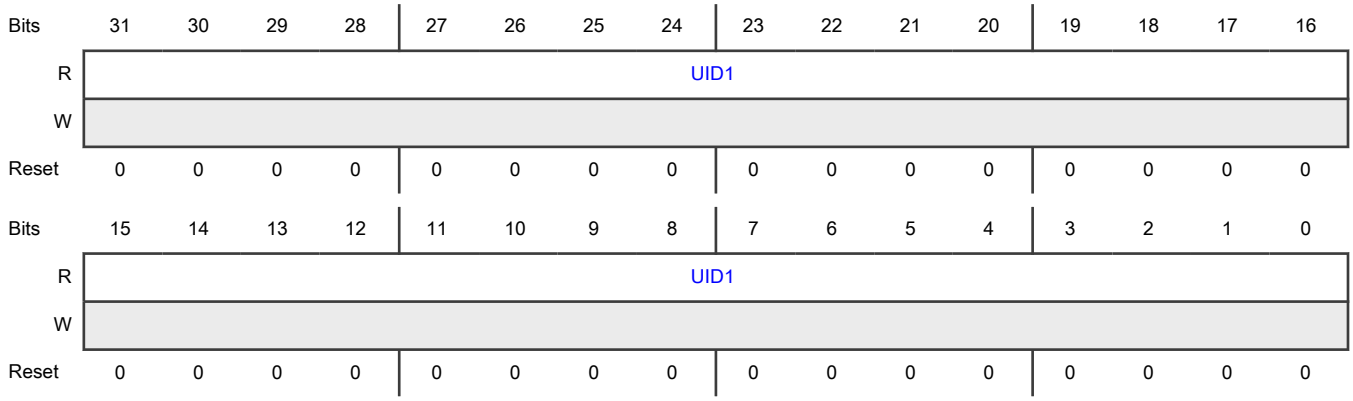
| Field | Function |
|---|---|
| 31-0<br>UID0 | User ID 0<br>Indicates the JTAG user ID bits of the lower word. |

## 75.16.7 User Identification 1 (UID1)

### Offset

| Register | Offset |
|---|---|
| UID1 | 74h |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>UID1 | User ID 1<br>Indicates the JTAG user ID bits of the upper word. |

## 75.16.8 Debug Enable Control (DBGENCTRL)

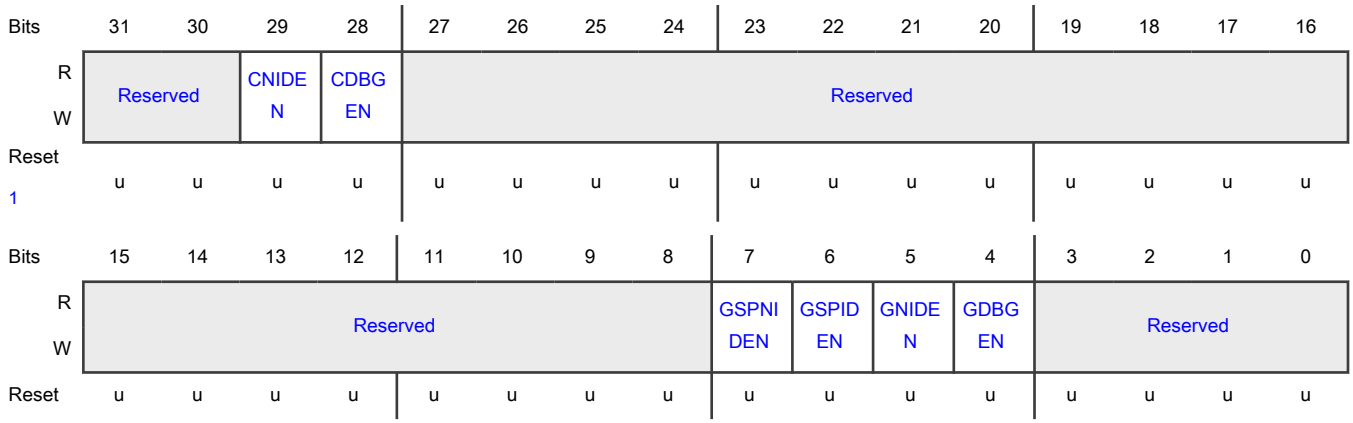
### Offset

| Register | Offset |
|---|---|
| DBGENCTRL | 80h |

### Function

Includes a special protection that allows access from Cortex-M0+ only if bit 30 of Authentication Status (AUTHSTTS) is 1.

### Diagram




1. The reset value is controlled by an export control enable input. If export control is enabled, the reset value of this register is FFFF_FFF0h. Otherwise, it is 0000_0000h.

### Fields

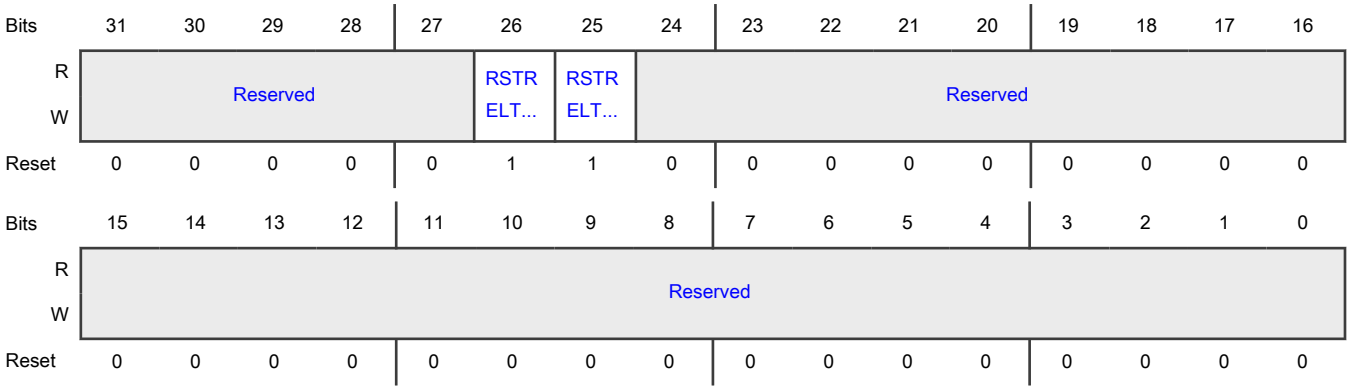
| Field | Function |
|---|---|
| 31-30<br>— | Reserved |
| 29<br>CNIDEN | Core Non-Invasive Debug Enable<br>Controls CNIDEN of debug blocks coupled with the Cortex-M7 core.<br>0b - Disabled<br>1b - Enabled |
| 28<br>CDBGEN | Core Debug Enable<br>Controls CDBGEN of debug blocks coupled with the Cortex-M7 core.<br>0b - Disabled<br>1b - Enabled |
| 27-8<br>— | Reserved |
| 7<br>GSPNIDEN | Global Secure Privileged Non-Invasive Debug Enable<br>Controls GSPNIDEN of debug blocks coupled with Cortex-M7's subsystems, ETM, ITM, and CTI.<br>0b - Disabled<br>1b - Enabled |
| 6<br>GSPIDEN | Global Secure Privileged Debug Enable<br>Controls GSPIDEN of debug blocks coupled with Cortex-M7's subsystems, ETM, ITM, and CTI.<br>0b - Disabled<br>1b - Enabled |
| 5<br>GNIDEN | Global Non-Invasive Debug Enable<br>Controls GNIDEN of debug blocks coupled with Cortex-M7's subsystems, ETM, ITM, and CTI.<br>0b - Disabled<br>1b - Enabled |
| 4<br>GDBGEN | Global Debug Enable<br>Controls GDBGEN of debug blocks coupled with Cortex-M7's subsystems, ETM, ITM, and CTI.<br>0b - Disabled<br>1b - Enabled |
| 3-0<br>— | Reserved |

## 75.16.9 Reset Control (SDAAPRSTCTRL)

### Offset

| Register | Offset |
|---|---|
| SDAAPRSTCTRL | 90h |

### Diagram




### Fields

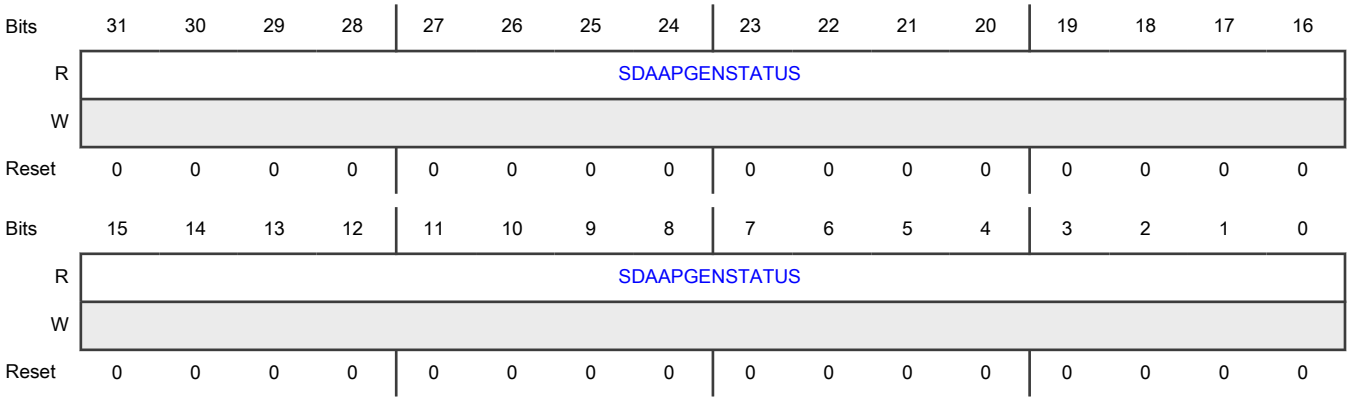
| Field | Function |
|---|---|
| 31-27<br>— | Reserved |
| 26<br>RSTRELTLCM71 | Reset Release Cortex-M7_1<br>Indicates if the control signal released the reset for Cortex-M7_1. The reset is released to debug the core from first instruction.<br>The default value of this field is 1.<br>0b - Core is in reset<br>1b - Reset is released |
| 25<br>RSTRELTLCM70 | Reset Release Cortex-M7_0<br>Indicates if the control signal released the reset for Cortex-M7_0. The reset is released to debug the core from the first instruction.<br>The default value of this field is 1.<br>0b - Core is in reset<br>1b - Reset is released |
| 24-0<br>— | Reserved |

## 75.16.10 SDA_AP Generic Status (SDAAPGENSTATUS0 - SDAAPGENSTATUS4)

### Offset

| Register | Offset |
|---|---|
| SDAAPGENSTATUS0 | A0h |
| SDAAPGENSTATUS1 | B0h |
| SDAAPGENSTATUS2 | C0h |
| SDAAPGENSTATUS3 | D0h |
| SDAAPGENSTATUS4 | E0h |

### Diagram




### Fields

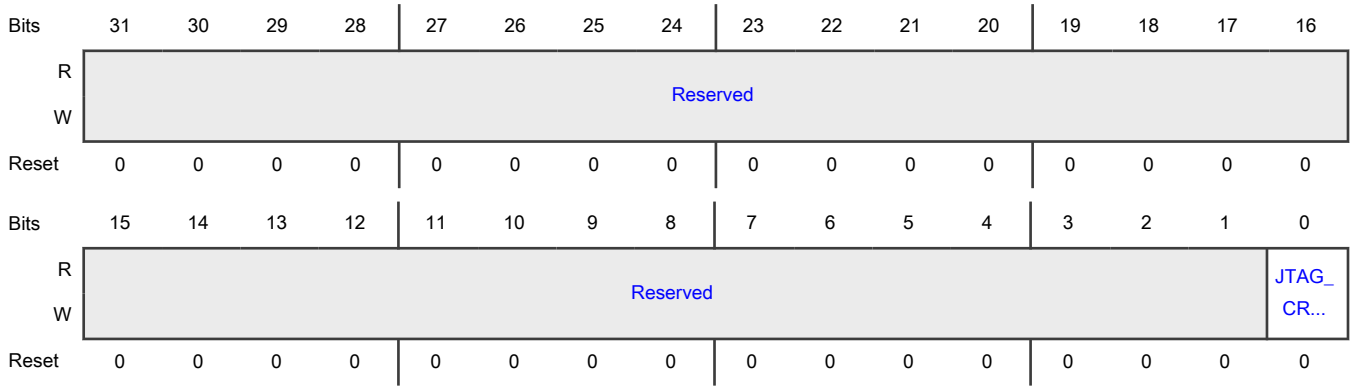
| Field | Function |
|---|---|
| 31-0<br>SDAAPGENSTATUS | DAP Generic Status<br>Is a generic status field for future use.<br>0b - Does not show generic status<br>1b - Shows generic status |

## 75.16.11 Generic Control 0 (SDAAPGENCTRL0)

### Offset

| Register | Offset |
|---|---|
| SDAAPGENCTRL0 | A4h |

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-1<br>— | Reserved |
| 0<br>JTAG_CR_EN | JTAG CR Enable<br>Performs CR or password comparison based on SWJ-DP mode or JTAG mode. If you write 1 to this field, this function is performed using JTAG irrespective of SWJ-DP mode of the debugger.<br>0b - Function performed on the basis of SWJ-DP mode<br>1b - Function performed on the basis of JTAG mode |

## 75.16.12 Identity (ID)

### Offset

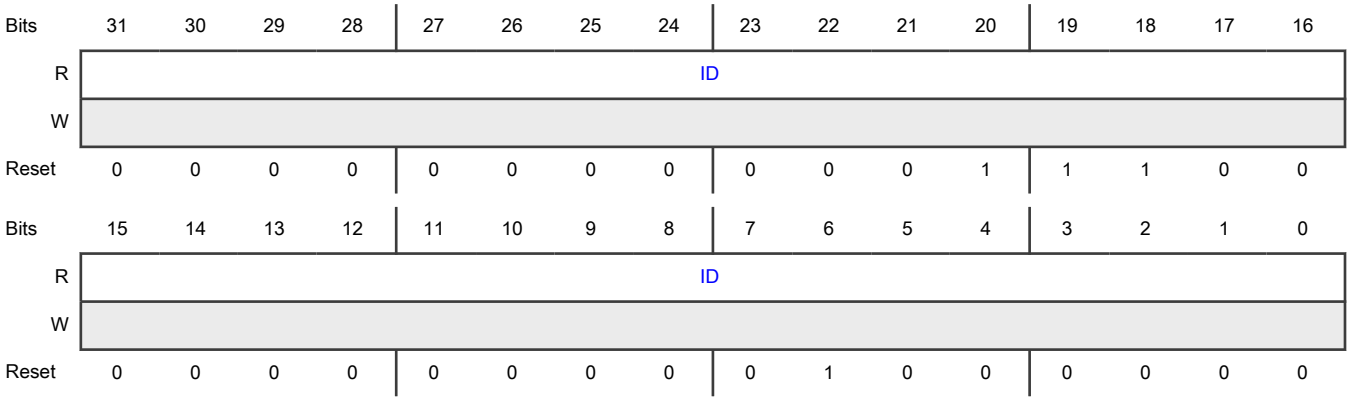
| Register | Offset |
|---|---|
| ID | FCh |

### Function

**NOTE**

This register does not generate the bus error on performing the write operation.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>ID | Identity |

## 75.17 Glossary

| | |
|---|---|
| **AHB** | Advanced high-performance bus |
| **AHB_AP** | Advanced high-performance bus access port |
| **APB** | Advanced peripheral bus |
| **APB_AP** | Advanced peripheral bus access port |
| **ATB** | Advanced trace bus |
| **ATBR** | ATB replicator |
| **CSTF** | CoreSight trace funnel |
| **DAP** | Debug access port |
| **DC** | Design center |
| **ETF** | Embedded CoreSight funnels |
| **JTAG-DP** | JTAG debug port |
| **MDM_AP** | Miscellaneous debug module access port |
| **MIC** | Manufacturer identity code |
| **PIN** | Part identification number |
| **PRN** | Part revision number |
| **SDA_AP** | Serial data access port |
| **SiP** | System-in-package |
| **SWJ-DP** | Serial wire/JTAG debug port |
| **TAP** | Test and debug access port |
| **TPIU** | Trace port interface unit |

## 75.18 References

- Arm CoreSight SoC-400 Technical Reference Manual

  https://static.docs.arm.com/100536/0302/coresight_soc400_technical_reference_manual_100536_0302_01_en.pdf
- Arm CoreSight Architecture Specification

  http://infocenter.arm.com/help/topic/com.arm.doc.ihi0029d/IHI0029D_coresight_architecture_spec_v2_0.pdf
- CoreSight Components Technical Reference Manual

  http://infocenter.arm.com/help/topic/com.arm.doc.ddi0314h/DDI0314H_coresight_components_trm.pdf
- Arm Debug Interface Architecture Specification

  https://static.docs.arm.com/ihi0031/c/IHI0031C_debug_interface_as.pdf

# Chapter 76
# JTAG Controller (JTAGC)

## 76.1 Chip-specific JTAGC information

The following figure shows the detailed TAP connectivity. SWD/JTAG SELECT, SWDP and JTAG-DP are part of ARM DAP

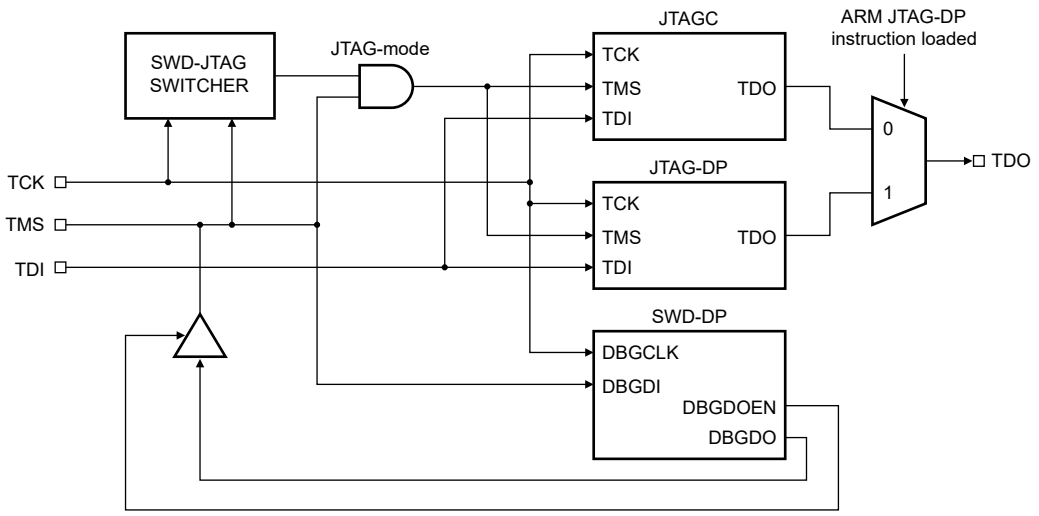



**Figure 445. Test and Debug Access Port (TAP) connectivity**

The debug port comes out of reset in a standard JTAG mode. It is switched to SWD mode by the change sequences described in JTAG to SWD change sequence . Once the mode has been changed, unused debug pins can be reassigned to any of their alternative muxed functions

JTAG-to-SWD change sequence:

1. Send more than 50 TCK cycles with TMS (SWDIO) =1
2. Send the 16-bit sequence on TMS (SWDIO) = 0111_1001_1110_0111 (MSB transmitted first)
3. Send more than 50 TCK cycles with TMS (SWDIO) =1

**NOTE**

In case of any reset event, JTAGC is moved to Test Logic Reset (TLR) state first and then JTAGC should be used.

**NOTE**

If JTAG pins are used for alternative functionality and reset event arrives, then these pins need to be reconfigured to required functionality before using it, in order to avoid unintentional entry in JTAG mode.

## 76.2 Overview

JTAGC allows you to test chip functionality and connectivity while remaining transparent to system logic when not in test mode. Testing is performed via a boundary scan technique, as defined in the IEEE 1149.1-2001 standard. All data input to and output from JTAGC is communicated in serial format.

### 76.2.1 Block diagram

The following is a simplified block diagram of the JTAG Controller (JTAGC) block. See the chip-specific configuration information within this chapter as well as Register description for more information about the JTAGC registers.

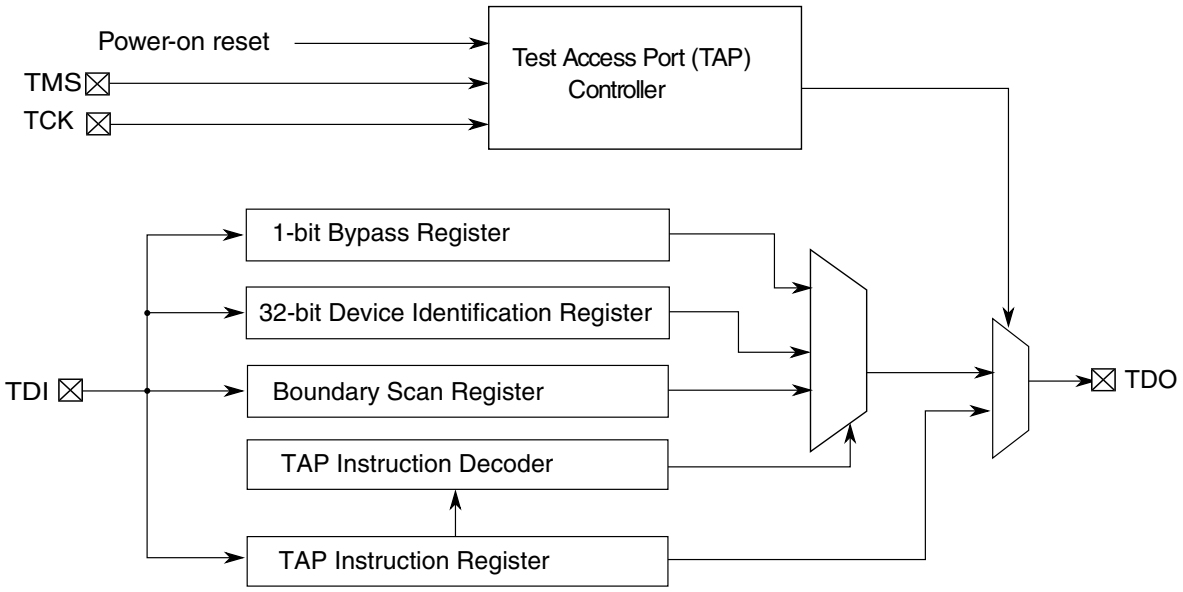



Figure 446. JTAG (IEEE 1149.1) block diagram

### 76.2.2 Features

The JTAGC block is compliant with the IEEE 1149.1-2001 standard, and supports the following features:

- IEEE 1149.1-2001 TAP interface
  - Four pins (TDI, TMS, TCK, and TDO)
- Instruction register that supports several IEEE 1149.1–2001 defined instructions as well as several public and private device-specific instructions (see JTAGC block instructions for a list of supported instructions).
- Sharing of the TAP with other TAP controllers via ACCESS_AUX_x instructions
- Bypass register, boundary scan register, and device identification register
- TAP controller state machine that controls the operation of the data registers, instruction register, and associated circuitry

## 76.3 Functional description

### 76.3.1 Modes of operation

The JTAGC block uses a power-on reset indication as its primary reset signals. Several IEEE 1149.1–2001 defined test modes are supported, as well as a bypass mode.

#### 76.3.1.1 IEEE 1149.1–2001 defined test modes

The JTAGC block supports several IEEE 1149.1–2001 defined test modes. A test mode is selected by loading the appropriate instruction into the instruction register when the JTAGC is enabled. Supported test instructions include EXTEST, HIGHZ, CLAMP, PRELOAD.

Each instruction defines the set of data register(s) that may operate and interact with the on-chip system logic when the instruction is current. Only one test data register path is enabled to shift data between TDI and TDO for each instruction.

The boundary scan register is enabled for serial access between TDI and TDO when the EXTEST, PRELOAD instructions are active. The single-bit bypass register shift stage is enabled for serial access between TDI and TDO when the following instructions are active:

- BYPASS

- HIGHZ
- CLAMP

The functionality of each test mode is explained in more detail in JTAGC block instructions.

#### 76.3.1.2 Bypass mode

When no test operation is required, the BYPASS instruction can be loaded to place the JTAGC block into bypass mode. When in bypass mode, the single-bit bypass shift register is used to provide a minimum-length serial path to shift data between TDI and TDO.

### 76.3.2 IEEE 1149.1-2001 (JTAG) TAP

The JTAGC block uses the IEEE 1149.1-2001 TAP for accessing registers. This port can be shared with other TAP controllers on the MCU. Ownership of the port is determined by the value of the currently loaded instruction. For more detail on TAP sharing via JTAGC instructions, see ACCESS_AUX_x instructions .

Data is shifted between TDI and TDO through the selected register (instruction, test data, or the bypass register) starting with the least significant bit, as illustrated in the following figure.

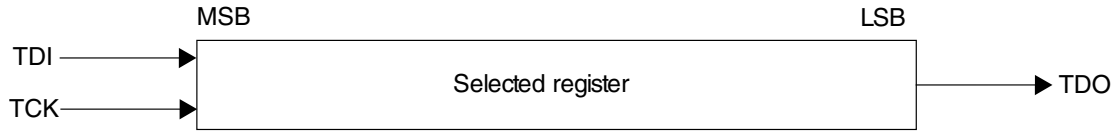
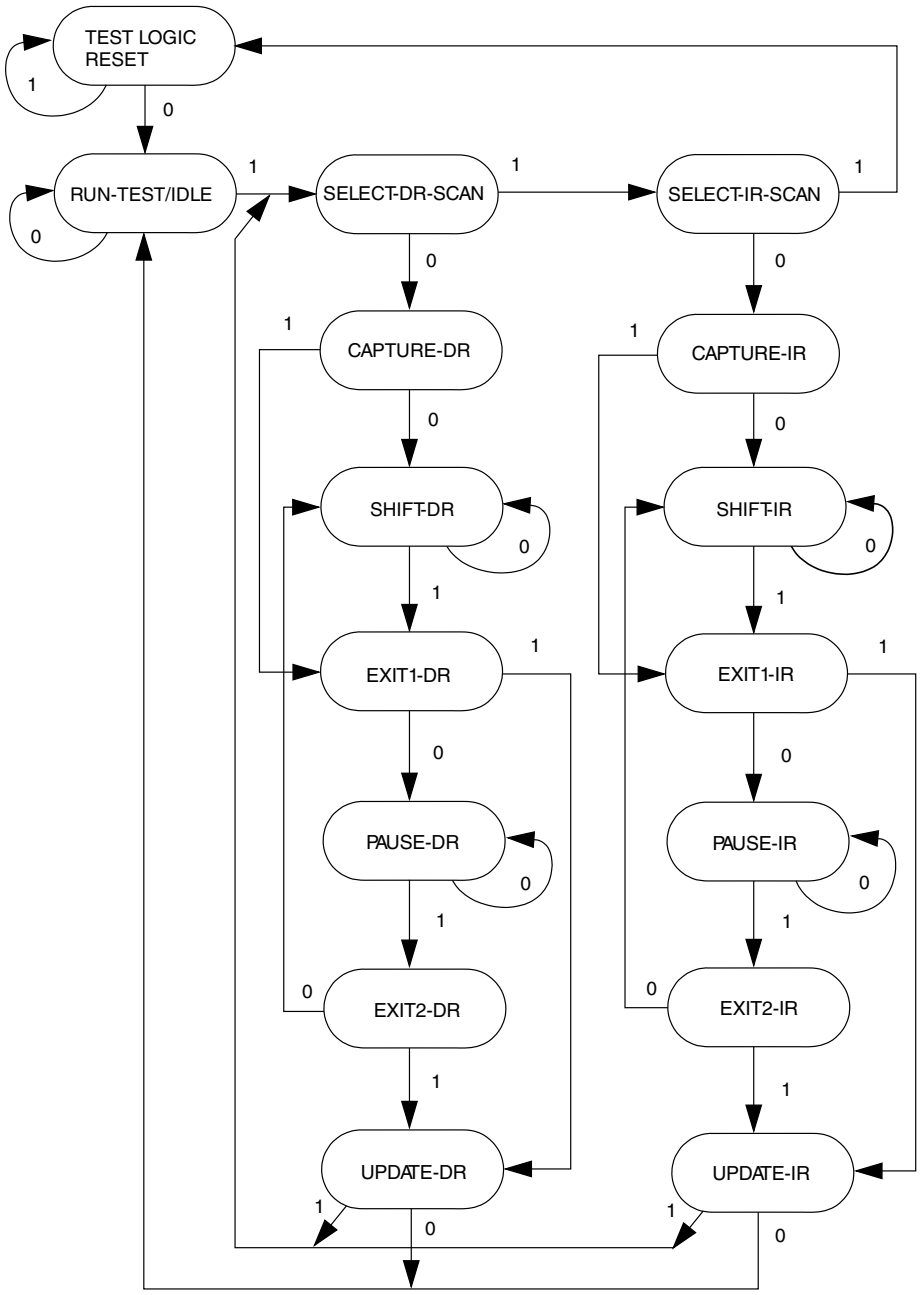



**Figure 447. Shifting data through a register**

### 76.3.3 TAP controller state machine

The TAP controller is a synchronous state machine that interprets the sequence of logical values on the TMS pin.

The following figure shows the machine's states. The value shown next to each state is the value of the TMS signal sampled on the rising edge of the TCK signal. As the figure shows, holding TMS at logic 1 when clocking TCK through a sufficient number of rising edges also causes the state machine to enter the Test-Logic-Reset state.




The value shown adjacent to each state transition in this figure represents the value of TMS at the time of a rising edge of TCK.

**Figure 448. IEEE 1149.1-2001 TAP controller finite state machine**

### 76.3.3.1 Enabling the TAP controller

The JTAGC TAP controller is enabled by setting the JTAGC enable to a logic 1 value.

### 76.3.3.2 Selecting an IEEE 1149.1-2001 register

Access to the JTAGC data registers is achieved by loading the instruction register with any of the JTAGC block instructions when the JTAGC is enabled. Instructions are shifted in via the Select-IR-Scan path and loaded in the Update-IR state. At this point, all data register access is performed via the Select-DR-Scan path.

The Select-DR-Scan path is used to read or write the register data by shifting in the data (LSB first) during the Shift-DR state. When reading a register, the register value is loaded into the IEEE 1149.1-2001 shifter during the Capture-DR state. When writing a register, the value is loaded from the IEEE 1149.1-2001 shifter to the register during the Update-DR state. When reading a register, there is no requirement to shift out the entire register contents. Shifting may be terminated after the required number of bits have been acquired.

### 76.3.4 JTAGC block instructions

The JTAGC implements instructions defined in IEEE 1149.1-2001. See the chip-specific JTAGC information for the JTAGC instructions implemented on this chip. See the IEEE 1149.1-2001 standard for additional information. All undefined opcodes are reserved.

#### 76.3.4.1 IDCODE instruction

IDCODE selects the 32-bit device identification register as the shift path between TDI and TDO. This instruction allows interrogation of the MCU to determine its version number and other part identification data. IDCODE is the instruction placed into the instruction register when the JTAGC block is reset.

#### 76.3.4.2 PRELOAD instruction

The PRELOAD instruction has two functions:

- The PRELOAD portion of the instruction initializes the boundary scan register cells before selecting the EXTEST or CLAMP instructions to perform boundary scan tests. This is achieved by shifting in initialization data to the boundary scan register during the Shift-DR state. The initialization data is transferred to the parallel outputs of the boundary scan register cells on the falling edge of TCK in the Update-DR state. The data is applied to the external output pins by the EXTEST or CLAMP instruction. System operation is not affected.

#### 76.3.4.3 EXTEST external test instruction

EXTEST selects the boundary scan register as the shift path between TDI and TDO. It allows testing of off-chip circuitry and board-level interconnections by driving preloaded data contained in the boundary scan register onto the system output pins. Typically, the data is loaded into the boundary scan register using the PRELOAD instruction before the selection of EXTEST. EXTEST asserts the internal system reset for the MCU to force a predictable internal state when performing external boundary scan operations.

#### 76.3.4.4 TEST_LEAKAGE instruction

The TEST_LEAKAGE instruction forces the jtag_leakage output signal to high. It is intended to tristate all output pad buffers and disable all of the part's pad input buffers except JCOMP and TEST. The jtag_leakage signal is asserted at the falling edge of TCK following the TAP controller state machine transition from the Update-IR state to the Run-Test-Idle state. When asserted, the part disables TCK, TMS, and TDI inputs and forces them to a logic 1. The TAP controller state machine remains in the Run-Test-Idle state until the JCOMP input is set to a value other than the JTAGC enable encoding. TEST_LEAKAGE also asserts the internal system reset for the MCU to force a predictable internal state.

#### 76.3.4.5 HIGHZ instruction

HIGHZ selects the bypass register as the shift path between TDI and TDO. When HIGHZ is active all output drivers are placed in an inactive drive state (for example, high impedance). HIGHZ also asserts the internal system reset for the MCU to force a predictable internal state.

#### 76.3.4.6 CLAMP instruction

CLAMP allows the state of signals driven from MCU pins to be determined from the boundary scan register when the bypass register is selected as the serial path between TDI and TDO. CLAMP enhances test efficiency by reducing the overall shift path to a single bit (the bypass register) when conducting an EXTEST type of instruction through the boundary scan register. CLAMP also asserts the internal system reset for the MCU to force a predictable internal state.

#### 76.3.4.7 ACCESS_AUX_x instructions

The JTAGC is configurable to allow other TAP controllers on the device to share the port with it. This is done by providing ACCESS_AUX_x instructions for each of these TAP controllers.

When this instruction is loaded, control of the JTAG pins is transferred to the selected TAP controller. Any data input via TDI and TMS is passed to the selected TAP controller, and any TDO output from the selected TAP controller is sent back to the JTAGC to be output on the pins.

The JTAGC gets back control of the JTAG port during the UPDATE-DR state if the PAUSE-DR state was entered. Auxiliary TAP controllers are held in RUN-TEST/IDLE when they are inactive. Instructions not used to access an auxiliary TAP controller on a device are treated like the BYPASS instruction.

#### 76.3.4.8 BYPASS instruction

BYPASS selects the bypass register, creating a single-bit shift register path between TDI and TDO. BYPASS enhances test efficiency by reducing the overall shift path when no test operation of the MCU is required. This allows more rapid movement of test data to and from other components on a board that are required to perform test functions. When the BYPASS instruction is active the MCU operates normally.

### 76.3.5 Boundary scan

The boundary scan technique allows signals at component boundaries to be controlled and observed through the shift-register stage associated with each pad. Each stage is part of a larger boundary scan register cell, and cells for each pad are interconnected serially to form a shift-register chain around the border of the design. The boundary scan register consists of this shift-register chain, and is connected between TDI and TDO when the EXTEST, PRELOAD instructions are loaded. The shift-register chain contains a serial input and serial output, as well as clock and control signals.

### 76.3.6 Clocking

There is one clock for this module. For more information See External signals.

### 76.3.7 Interrupts

This module has no interrupts.

### 76.3.8 Reset

The JTAGC block is placed in reset when:

- Power-on reset is asserted
- TMS input is held high for enough consecutive rising edges of TCK to sequence the TAP controller state machine into the Test-Logic-Reset state

Holding TMS high for five consecutive rising edges of TCK guarantees entry into the Test-Logic-Reset state regardless of the current TAP controller state. Asserting power-on reset results in asynchronous entry into the reset state.

When in reset, the following actions occur:

- The TAP controller is forced into the Test-Logic-Reset state, thereby disabling the test logic and allowing normal operation of the MCU.
- The instruction register is loaded with the IDCODE instruction.

## 76.4 External signals

JTAGC includes a set of signals that connect to off-chip development tools and allow access to test support functions. The JTAGC signals are outlined in the following table and described in the following sections.

Table 597. JTAG signal properties

| Name | I/O | Function | Reset state |
|---|---|---|---|
| TCK | Input | Test clock | Weak pulldown |
| TDI | Input | Test data in | Weak pullup |
| TDO | Output | Test data out | High Z[1] |
| TMS | Input | Test mode select | Weak pullup |

1. TDO output buffer is disabled when the JTAGC is not in the Shift-IR or Shift-DR states. A weak pull may be implemented at the TDO pad for use when JTAGC is inactive.

### 76.4.1 Test clock input (TCK)

Test Clock Input (TCK) is an input pin used to synchronize the test logic and control register access through the TAP.

### 76.4.2 Test data input (TDI)

Test Data Input (TDI) is an input pin that receives serial test instructions and data. TDI is sampled on the rising edge of TCK.

### 76.4.3 Test data output (TDO)

Test Data Output (TDO) is an output pin that transmits serial output for test instructions and data. TDO is tristateable and is actively driven only in the Shift-IR and Shift-DR states of the TAP controller state machine, which is described in TAP controller state machine.

### 76.4.4 Test mode select (TMS)

Test Mode Select (TMS) is an input pin used to sequence the IEEE 1149.1-2001 test control state machine. TMS is sampled on the rising edge of TCK.

## 76.5 Initialization

To initialize the JTAGC block and enable access to registers, the following sequence is required:

1. Place the JTAGC in reset through TAP controller state machine transitions controlled by TMS.
2. Load the appropriate instruction for the test or action to be performed.

## 76.6 Application information

The test logic is a static logic design, and TCK can be stopped in either a high or low state without loss of data. However, the system clock is not synchronized to TCK internally. Any mixed operation using both the test logic and the system functional logic requires external synchronization.

## 76.7 Register description

This section provides a detailed description of the JTAGC block registers accessible through the TAP interface, including data registers and the instruction register. Individual bit-level descriptions and reset states of each register are included. These registers are not memory-mapped and can only be accessed through the TAP.

### 76.7.1 Instruction register

The JTAGC block uses a 8-bit instruction register as shown in the following figure.

The instruction register allows instructions to be loaded into the block to select the test to be performed, or the test data register to be accessed, or both. Instructions are shifted in through TDI when the TAP controller is in the Shift-IR state, and latched on the falling edge of TCK in the Update-IR state. The latched instruction value can only be changed in the Update-IR and Test-Logic-Reset TAP controller states.

Synchronous entry into the Test-Logic-Reset state results in the IDCODE instruction being loaded on the falling edge of TCK. Asynchronous entry into the Test-Logic-Reset state results in asynchronous loading of the IDCODE instruction. During the Capture-IR TAP controller state, the instruction shift register is loaded with the value 0b1, making this value the register's read value when the TAP controller is sequenced into the Shift-IR state.

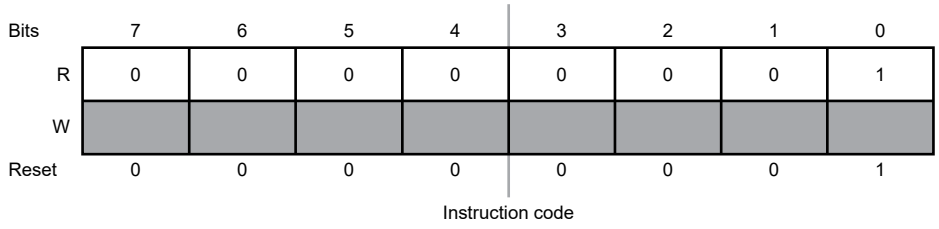



**Figure 449. Instruction register**

## 76.7.2 Bypass register

The bypass register is a single-bit shift register path selected for serial data transfer between TDI and TDO when the following instructions are active:

- BYPASS
- HIGHZ
- CLAMP

After entry into the Capture-DR state, the single-bit shift register is set to a logic 0. Therefore, the first bit shifted out after selecting the bypass register is always a logic 0.

## 76.7.3 Device identification register

The device identification (JTAG ID) register, shown in the following figure, allows the revision number, part number, manufacturer, and design center responsible for the design of the part to be determined through the TAP.

The device identification register is selected for serial data transfer between TDI and TDO when the IDCODE instruction is active. Entry into the Capture-DR state when the device identification register is selected loads the IDCODE into the shift register to be shifted out on TDO in the Shift-DR state. No action occurs in the Update-DR state.

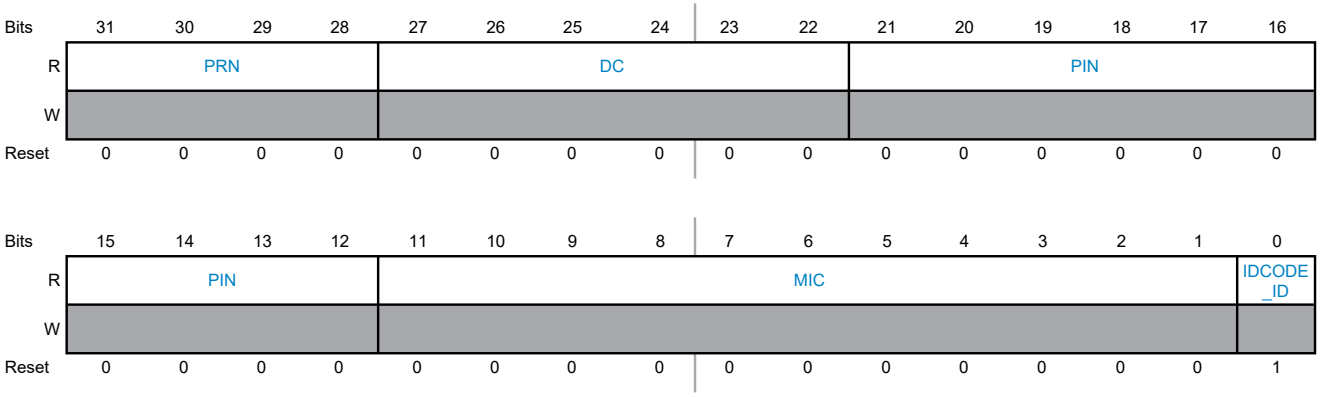



**Figure 450. Device identification register**

The following table describes the device identification register functions. The device identification register values are described in the chip-specific JTAGC information.

**Table 598. Device identification register field descriptions**

| Field | Function |
|---|---|
| PRN | Part revision number<br>Contains the revision number of the part. |
| DC | Design center<br>Indicates the design center. |
| PIN | Part identification number<br>Contains the part number of the device. |
| MIC | Manufacturer identity code<br>Contains the reduced Joint Electron Device Engineering Council (JEDEC) ID . |
| IDCODE ID | IDCODE register ID<br>Identifies this register as the device identification register and not the bypass register. Always set to 1. |

### 76.7.4 Boundary scan register

The boundary scan register is connected between TDI and TDO when the EXTEST, PRELOAD instructions are active. It is used to:

- Capture input pin data
- Force fixed values on output pins
- Select a logic value and direction for bidirectional pins

Each bit of the boundary scan register represents a separate boundary scan register cell, as described in the IEEE 1149.1-2001 standard and discussed in Boundary scan. The size of the boundary scan register and bit ordering is device-dependent and can be found in the device BSDL file.

## 76.8 Glossary

| | |
|---|---|
| **TAP** | Test access port |
| **TCK** | Test clock |
| **TDI** | Test data in |
| **TDO** | Test data out |
| **TMS** | Test mode select |
| **JCOMP** | JTAG compliance |

# Chapter 77
# JTAG Data Communication (JDC)

## 77.1 Overview

JDC module provides the capability to move register data between the IPS and JTAG domains. This facilitates communication between internal resources that access memory-mapped register space and an external tool that accesses the JTAG port.

The JDC module consists of:

- IPS-accessible registers
- JTAG-accessible registers
- Associated logic to coordinate movement of data from one register domain to another

The JDC implements the following IPS data registers, occupying separate memory space:

- 1 32-bit memory mapped register that can be read or written via IPS (JTAG Output Data Register (JOUT_IPS)), and whose contents are ported out for capture into a JTAG register (JOUT) to be read via the JTAG port.
- 1 32-bit memory mapped register that can only be read via IPS (JTAG Input Data Register (JIN_IPS)), and whose contents are loaded from a JTAG register (JIN).

JDC indicates when:

- New data has been shifted in via the JTAG port and is ready to be read from the JTAG Input Data Register (JIN_IPS) register
- New data has been written to the JTAG Output Data Register (JOUT_IPS) register and is ready to be read via the JTAG port

A Module Status Register (MSR) captures the state of these flags and also provides the flags to a JTAG register (JOUT) for tool visibility.

### 77.1.1 Block diagram

There is a single bus interface to port register data out to the JTAGC, and a single bus interface to port data in from the JTAGC. The following figure is the JDC module block diagram.

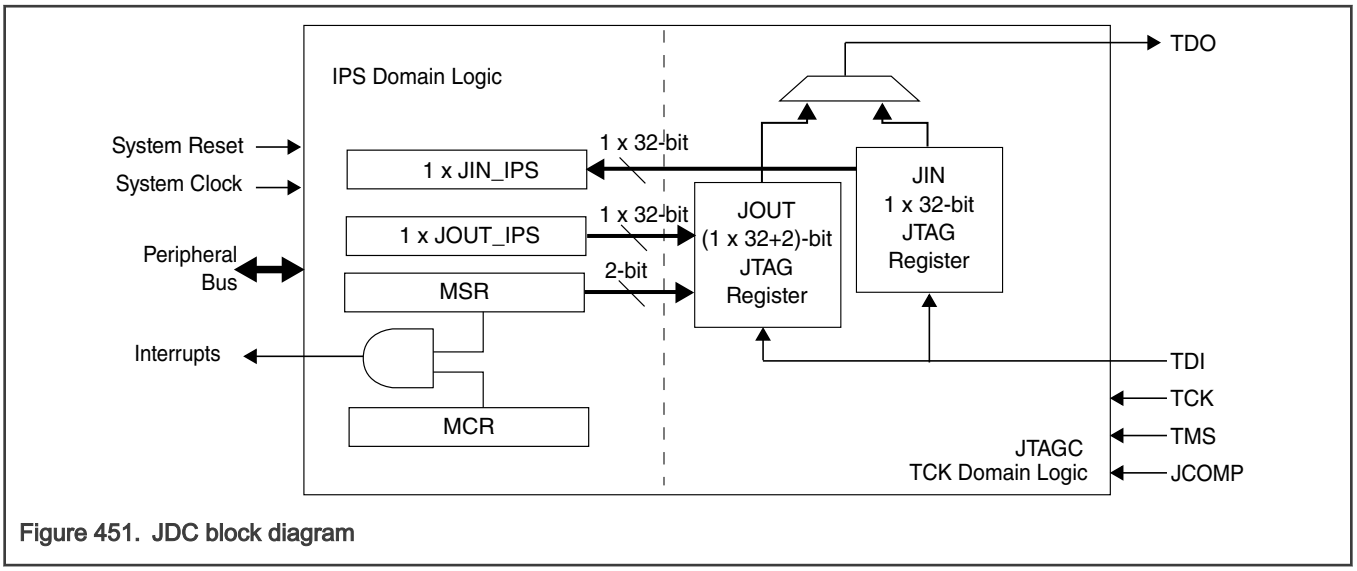



**Figure 451. JDC block diagram**

## 77.2 Functional description

The JDC module provides the ability to shift in data via the JTAG port and capture that data in memory-mapped register that can be accessed via IPS. It also provides the ability to capture data written to memory-mapped register into a JTAG shift register for output via the JTAG port.

### 77.2.1 JDC functionality

An overview of the module functionality is described here:

- Software write to the JOUT_IPS register:
  1. The JDC sets the JOUT_RDY flag bit, indicating new data is available to be read from the JOUT register via the JTAG port.
  2. The MSR reflects the state of the JOUT_RDY bit.
  3. The JDC also captures the state of the JOUT_RDY bit in the JOUT register.
  4. A JTAG read of the JOUT register via execution of the JOUT_READ instruction with a JOUT_RDY bit whose value is logic 1 indicates that the register contains new data.
  5. The JDC clears the JOUT_RDY flag bit upon exit of the Capture-DR JTAG state during execution of the JOUT_READ instruction.
  6. Clearing the JOUT_RDY bit indicates to software that a new data value can be written to the JOUT_IPS register.
- JTAG write to the JIN register via execution of the JIN_WRITE JTAG instruction:
  1. The JDC updates the contents of the JIN_IPS register upon exit of the Update-DR state.
  2. An update of the JIN_IPS register sets the JIN_RDY flag bit, indicating new data is available to be read via IPS.
  3. The MSR register reflects the state of the JIN_RDY bit.
  4. The JDC also captures the state of the JIN_RDY bit in the JOUT register.
  5. The JDC clears the JIN_RDY flag bit upon software read of the JIN_IPS register.
  6. A JTAG read of the JOUT register with a JIN_RDY value of logic 0 indicates that new data can be written to the JIN register.

### 77.2.2 JTAG register access

See the JTAGC documentation for information on how to access the JTAG registers.

#### 77.2.2.1 JDC block instructions

The JDC block implements the instructions listed in Table 599. This section gives an overview of each instruction. All undefined opcodes are reserved.

**Table 599. JTAG instructions**

| Instruction | Code[4:0] | Instruction summary |
|---|---|---|
| Reserved | 00001 | Factory debug reserved. |
| JOUT_READ | 00010 | Selects JOUT data register. The JDC captures data from JOUT_IPS into JOUT data register upon entry to Capture-DR state when JOUT_READ is active. |
| JIN_WRITE | 01110 | Selects JIN data register. The JDC captures data from JIN into JIN_IPS upon exit of Update-DR state when JIN_WRITE is active. |

*Table continues on the next page...*

Table 599. JTAG instructions (continued)

| Instruction | Code[4:0] | Instruction summary |
|---|---|---|
| BYPASS[1] | 11111 | Selects bypass register for data operations. |
| Reserved | All other opcodes | Decoded to select bypass register. |

1. This is an IEEE 1149.1-2001 defined instruction. See the IEEE 1149.1-2001 standard for more details.

### 77.2.3 Clocking

JDC module uses TCK and inverted TCK as clock source.

### 77.2.4 Interrupts

This module has no interrupts.

## 77.3 External signals

JDC signals are listed in the following table.

Table 600. Signal properties

| Name | Function | I/O | Reset |
|---|---|---|---|
| TCK | Test clock input | I | — |
| TMS | Test mode select | I | 0 |
| TDI | Test data in | I | 0 |
| JCOMP | JTAG compliance pin | I | 0 |
| TDO | Test data out | O | HiZ |

## 77.4 Initialization

This module does not require initialization.

## 77.5 Secure challenge/response connection procedure

A summary of how the JDC can be used to interact with a security module to perform challenge/response password authorization is as follows:

1. After exit of reset, all registers are at their reset values. JIN_RDY bit value of 0 indicates that the tool may write data to the JIN register. JOUT_RDY bit value of 0 indicates that software may write data to the JOUT_IPS register.
2. The MCR register is programmed to enable JOUT and JIN interrupts, or not.
3. Once the tool is ready to start the authorization process, it writes a value to the JIN register via execution of the JTAGC JIN_WRITE instruction.
4. Software monitors the state of the JIN_RDY bit or enables the JIN interrupt and uses the interrupt assertion as a trigger to start the authorization process.
5. Once the authorization process is started, software writes the first JOUT value to the JOUT_IPS register. This sets the JOUT_RDY bit value to 1.

6. Following the initial tool write to the JIN register, the tool polls the JOUT register to determine when software has provided a challenge word. Upon reading the JOUT register with JOUT_RDY bit set, the tool records the data value to be used to generate the appropriate response. The read of the JOUT register clears the JOUT_RDY bit. The clearing of the JOUT_RDY bit also sets the JOUT_INT bit and asserts the JOUT interrupt if the MCR[JOUT_IEN] is set.
7. With the JOUT_RDY bit cleared, software may now provide the next challenge word.
8. Once the tool has generated a response word, it uses the value of the JIN_RDY bit from the JOUT register read to determine when it can perform a write to the JIN register to provide the response.
9. The security module can provide additional challenge words and read back the corresponding response words as needed, repeating the process.

Simply providing a password without using the challenge/response protocol can be done in a similar way. The tool provides 32-bits of the password at a time with each write to the JIN register, and monitors the JIN_RDY bit value of the JOUT register to determine when the next JIN register value can be written. There is no need for software to write the JOUT_IPS register in this case.

For details related to the lifecycle stage at which this process is performed, see the "Hardware Security Engine" chapter.

## 77.6 JDC register descriptions

The following sections describe the implemented 32-bit registers. Only 32-bit accesses are valid. The effects of access that are not 32 bits are not defined.

### 77.6.1 JDC memory map

JDC base address: 4039_4000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | Module Configuration Register (MCR) | 32 | RW | 0000_0000h |
| 4h | Module Status Register (MSR) | 32 | RW | 0000_0000h |
| 8h | JTAG Output Data Register (JOUT_IPS) | 32 | RW | 0000_0000h |
| Ch | JTAG Input Data Register (JIN_IPS) | 32 | R | 0000_0000h |

### 77.6.2 Module Configuration Register (MCR)

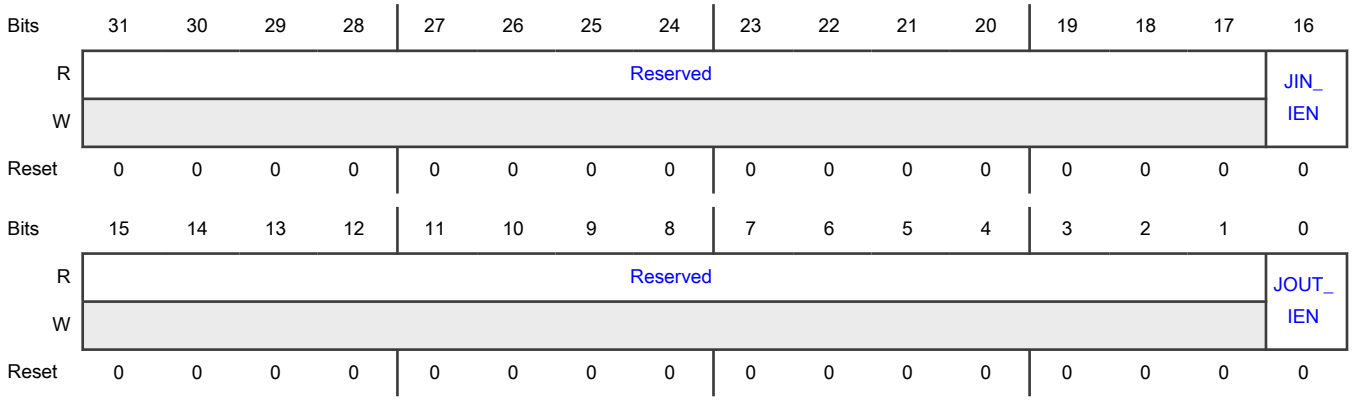
#### Offset

| Register | Offset |
|---|---|
| MCR | 0h |

#### Function

The MCR enables the interrupt outputs of the JDC. This register is reset by system destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-17<br>— | Reserved |
| 16<br>JIN_IEN | JIN Interrupt Enable<br>0b - Setting MSR[JIN_INT] bit does not assert the JIN interrupt<br>1b - Setting MSR[JIN_INT] bit asserts the JIN interrupt |
| 15-1<br>— | Reserved |
| 0<br>JOUT_IEN | JOUT Interrupt Enable<br>0b - Setting MSR[JOUT_INT] bit does not assert the JOUT interrupt<br>1b - Setting MSR[JOUT_INT] bit asserts the JOUT interrupt |

## 77.6.3 Module Status Register (MSR)

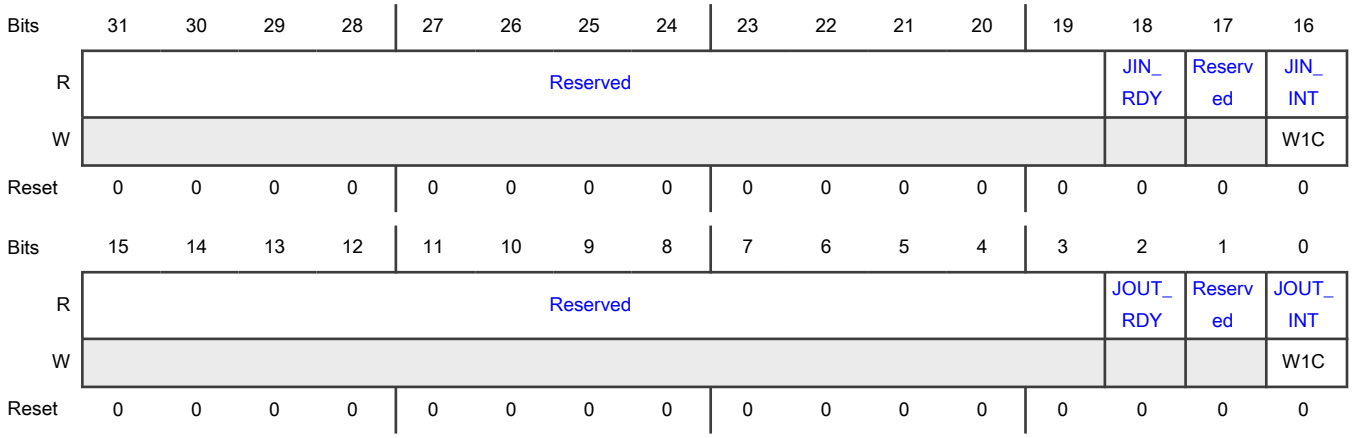
### Offset

| Register | Offset |
|---|---|
| MSR | 4h |

### Function

The MSR holds the JTAG register status and interrupt bits. This register is reset by system destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-19<br>— | Reserved |
| 18<br>JIN_RDY | JIN Ready (read only)<br>0b - Cleared upon software read of JIN_IPS contents via IPS<br>1b - Set when new data is written to the JIN_IPS register |
| 17<br>— | Reserved |
| 16<br>JIN_INT | JIN Interrupt<br>0b - Cleared by writing logic 1<br>1b - Set when new data is written to the JIN_IPS register |
| 15-3<br>— | Reserved |
| 2<br>JOUT_RDY | JOUT Ready (read only)<br>0b - Cleared upon tool read of JOUT register via JTAG port<br>1b - Set when new data is written to the JOUT_IPS register |
| 1<br>— | Reserved |
| 0<br>JOUT_INT | JOUT Interrupt<br>0b - Cleared by writing logic 1<br>1b - Set when JOUT_RDY bit is cleared by tool reading JOUT register |

## 77.6.4 JTAG Output Data Register (JOUT_IPS)

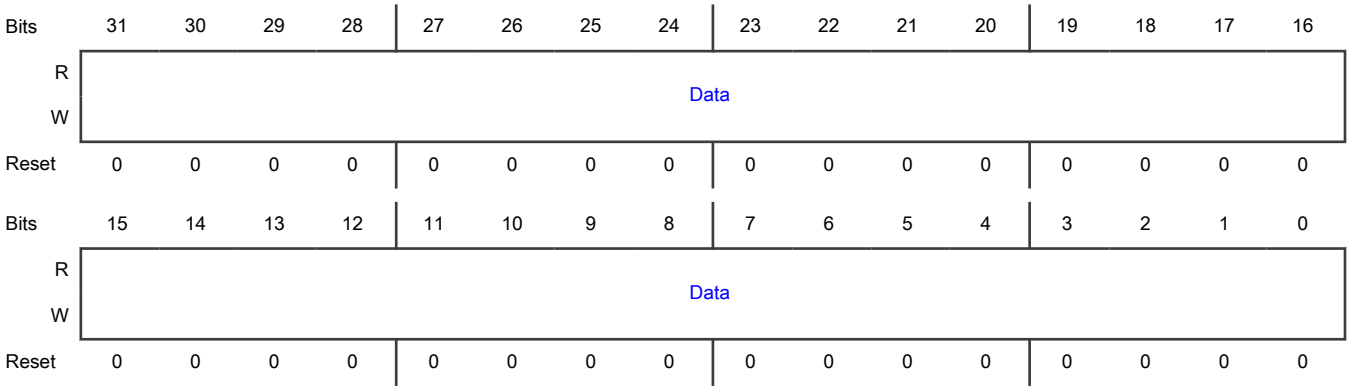
### Offset

| Register | Offset |
|---|---|
| JOUT_IPS | 8h |

### Function

The JOUT_IPS register holds data written via IPS. The JDC captures the JOUT_IPS contents into the JOUT register to be read via the JTAG port. This register is reset by system destructive reset.

### Diagram




### Fields

| Field | Function |
|---|---|
| 31-0<br>Data | JOUT_IPS Data |

## 77.6.5 JTAG Input Data Register (JIN_IPS)

### Offset

| Register | Offset |
|---|---|
| JIN_IPS | Ch |

### Function

The JIN_IPS register holds data written to the JTAG input data register (JIN) via the JTAG port, where the data can be read via IPS. Any IPS write to the JIN_IPS register returns a transfer error. This register is reset by system destructive reset.

### Diagram

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

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

### Fields

| Field | Function |
|---|---|
| 31-0<br>Data | JIN_IPS data |

## 77.7 Non-memory-mapped register definition

The JDC also implements two JTAG accessible registers that are not memory-mapped. One JTAG register shifts in data to be placed in the JIN_IPS register. The other JTAG register captures data from the JOUT_IPS register plus ready status from the MSR, to be shifted out via the JTAG port.

### 77.7.1 JTAG output data register (JOUT)

The JOUT register captures data from the JOUT_IPS register upon execution of the JOUT_READ JTAG instruction. It also holds the JIN_RDY and JOUT_RDY status bits. This register is reset by system destructive reset and JTAG reset. The reset value of the JOUT register is 0. The following figure shows the format of the JOUT register.

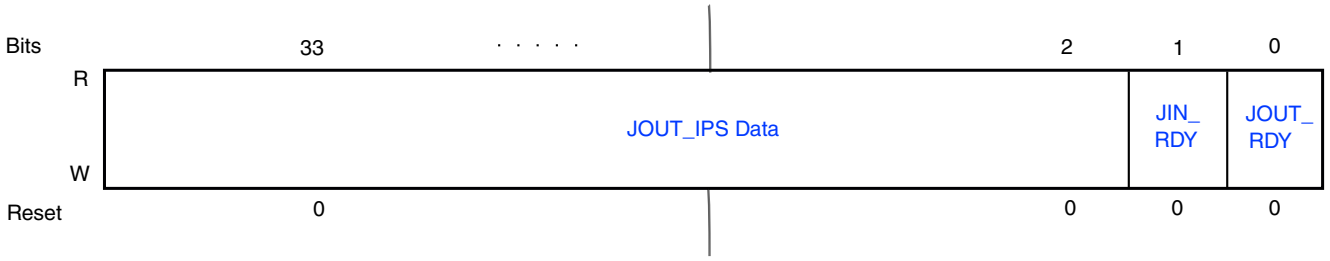



Figure 452. JOUT

The JOUT register is described in the following table.

Table 601. JOUT JTAG register field descriptions

| Field | Function |
|---|---|
| JOUT_IPS Data | JOUT_IPS Data<br>Data value from JOUT_IPS register |

*Table continues on the next page...*

Table 601. JOUT JTAG register field descriptions (continued)

| Field | Function |
|---|---|
| JIN_RDY | JIN_RDY<br>State of JIN_RDY bit from MSR |
| JOUT_RDY | JOUT_RDY<br>State of JOUT_RDY bit from MSR |

### 77.7.2 JTAG input data register (JIN)

The external tool writes data to the JIN register via JTAG during execution of the JIN_WRITE JTAG instruction. The JDC later captures JIN data in the JIN_IPS register to be read via IPS. This register is reset by system destructive reset and JTAG reset. The reset value of the JIN register is 0. The following figure shows the format of the JIN register.

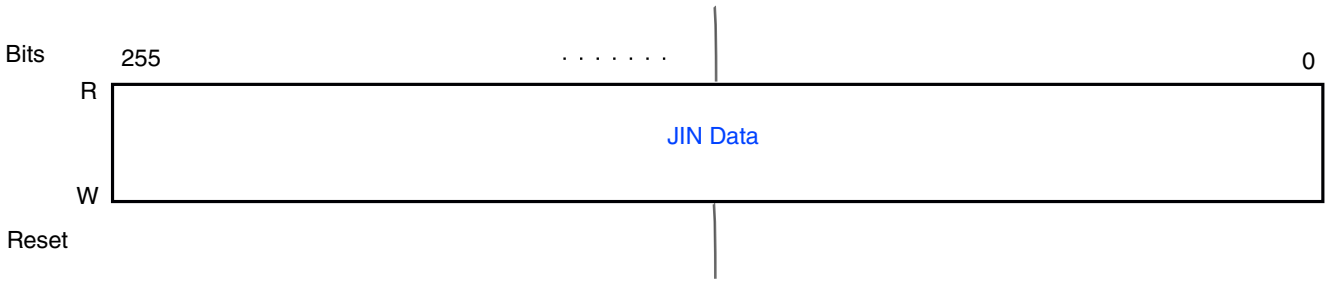



Figure 453. JIN

The JIN register is described in the following table.

Table 602. JIN JTAG register field descriptions

| Field | Description |
|---|---|
| JIN Data | Contains data to be captured in JIN_IPS register upon exit of Update-DR state when executing WRITE_JIN JTAG instruction. |

## 77.8 Glossary

| | |
|---|---|
| **IPS** | Internal peripheral system |
| **XBAR** | Crossbar bus interface |

# Chapter 78
# Temperature Sensor (TEMPSENSE)

## 78.1 Overview

TEMPSENSE is used to measure the temperature on the chip through an ADC and to flag if an overtemperature condition occurs.

The TEMPSENSE module contains control registers for use with the Engineering temperature sensor (ETS) hard block and the flag generation hard block. These registers use a peripheral bus interface to communicate with the chip.

### 78.1.1 Block diagram

The functional structure of the TEMPSENSE can be seen in the block diagram in Figure 454.

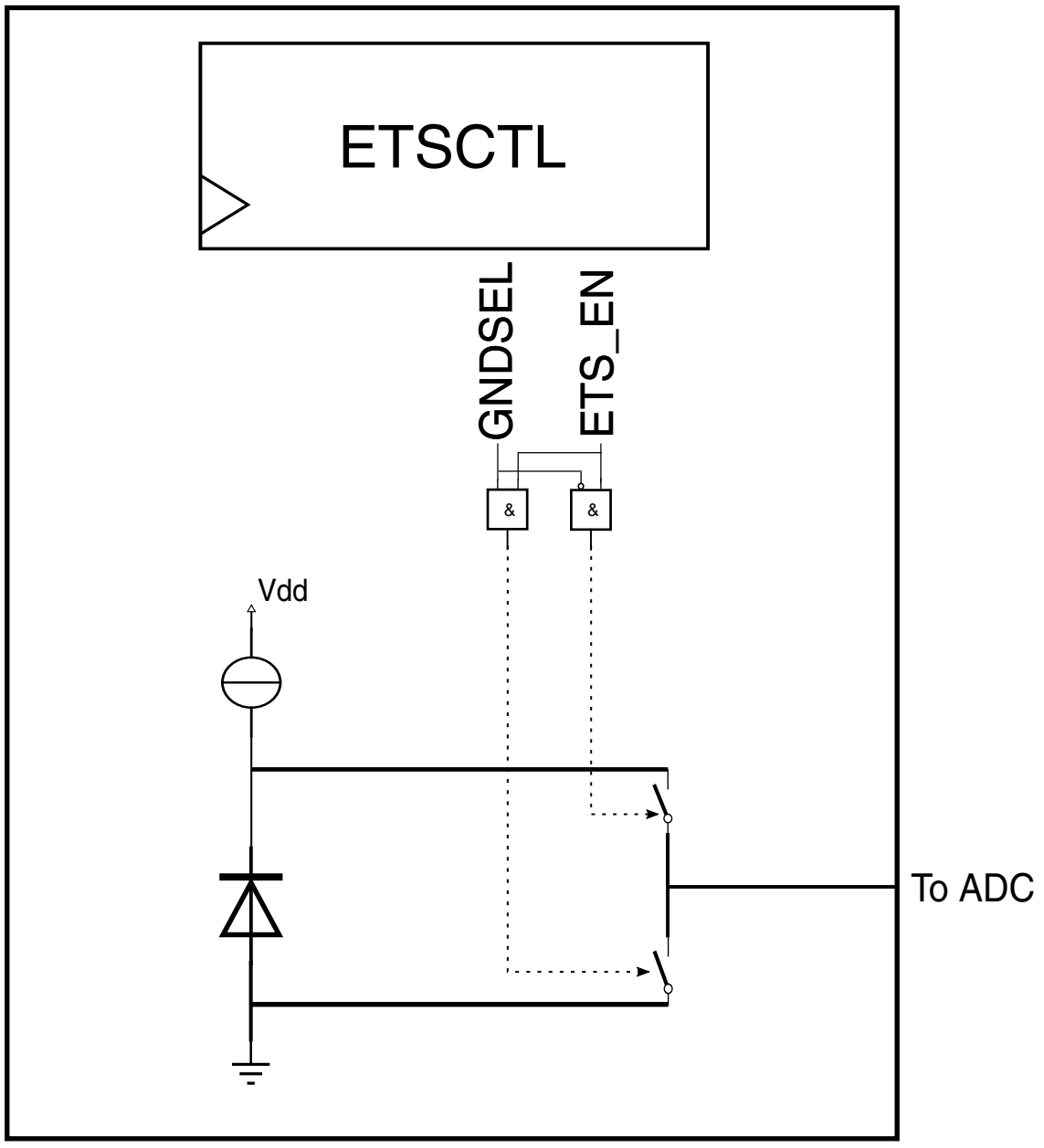



Figure 454. TEMPSENSE block diagram

### 78.1.2 Features

The TEMPSENSE includes the following features:

- Provide a voltage proportional to the temperature which will be read out by ADC
- Ground selection for improved precision in the ETS. The ETS ground could be different from the ADC ground. By exposing it to the ADC, the temperature calculation could be improved as the voltage value will be more precise.

## 78.2 Functional description

### 78.2.1 Clocking

This module has no clocking considerations.

### 78.2.2 Interrupts

This module has no interrupts.

### 78.2.3 General

The TEMPSENSE module is designed to measure the temperature on the chip.

The TEMPSENSE output can be read by an ADC on demand so that the software can determine the current die temperature. The software should enable the TEMPSENSE by setting ETSCTL[ETS_EN].

The TEMPSENSE output can be read by an ADC on demand so that the software can determine the current die temperature..

Coefficients are used to allow a simple and accurate calculation of linearized temperature directly from the ADC. The increase of the sampling time of the ADC from the minimum sampling value is increasing the temperature accuracy.

### 78.2.4 Initialization

#### 78.2.4.1 Conversion from voltage to temperature

Solve the equation for the conversion from voltage to temperature, where $V_{ETS}$ is the difference of $V_{be}$ and $V_{GND}$. $V_{GND}$ is expose on the ADC output when ETSCTL[GNDSEL] is set.

$$T_{ETS} = TCA_0 + TCA_1 \cdot V_{ETS} + TCA_2 \cdot V_{ETS}^2$$

The coefficients TCA0, TCA1 and TCA2 are read from the corresponding registers. They are stored in a signed fixed-point format as the following TCAx(11,4) (1 bit for the sign, 11 bits for the integer part and 4 bits for the decimal part).

The calculation of the temperature should be done with the actual coefficient values provided in the TCAx fields. See the example below for an ambient temperature of 25C, and an ADC reference voltage ($V_{REFH}$) of 5V. See the ADC section of the device data sheet for more details on $V_{REFH}$ and N-bit level resolution.

$$TCA0 = (000110100100.1110)_2 = (000110100100)_2 + (1110)_2 \times \frac{1}{2^4} = +((420)_{10} + (14)_{10} \times \frac{1}{2^4}) = +420.8750\ C$$

$$TCA1 = (100010101111.1110)_2 = (100010101111)_2 + (1110)_2 \times \frac{1}{2^4} = -((175)_{10} + (14)_{10} \times \frac{1}{2^4}) = -175.8750\ C/V$$

$$TCA2 = (100000011101.1110)_2 = (100000011101)_2 + (1110)_2 \times \frac{1}{2^4} = -((29)_{10} + (14)_{10} \times \frac{1}{2^4}) = -29.8750\ C\ /V^2$$

$$V_{ETS} = (010110001011)_2 = (1419)_{10} \times \frac{V_{REFH}}{2^N}$$

By using the formula, the junction temperature calculated is 26.5838 C.

The maximal calculation error is 0.0313 C.

## 78.3 External signals

There are no external signals for this module.

## 78.4 Memory map and register definition

### 78.4.1 Transfer error description

The following actions will cause a transfer error and will not change the register content:

- Any access to an unused register address
- Write access to a read-only register (TCAx)

### 78.4.2 TEMPSENSE register descriptions

#### 78.4.2.1 TEMPSENSE memory map

TEMPSENSE base address: 4037_C000h

| Offset | Register | Width (In bits) | Access | Reset value |
|---|---|---|---|---|
| 0h | ETS Control (ETSCTL) | 32 | RW | 0000_0000h |
| 8h | Temperature Coefficient (TCA0) | 32 | R | See section |
| Ch | Temperature Coefficient (TCA1) | 32 | R | See section |
| 10h | Temperature Coefficient (TCA2) | 32 | R | See section |

#### 78.4.2.2 ETS Control (ETSCTL)

**Offset**

| Register | Offset |
|---|---|
| ETSCTL | 0h |

**Function**

This register contains control bits that control ETS.

**Diagram**

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

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

**Fields**

| Field | Function |
|---|---|
| 31-2<br>— | Reserved |
| 1<br>GNDSEL | Ground selection<br>0b - No exposure of the ground<br>1b - Expose ground on the ADC output if ETSCTL[ETS_EN]=1 |
| 0<br>ETS_EN | Temperature Sensor enable<br>Power up the ETS.<br>0b - Power down<br>1b - Functional mode |

### 78.4.2.3 Temperature Coefficient (TCA0)

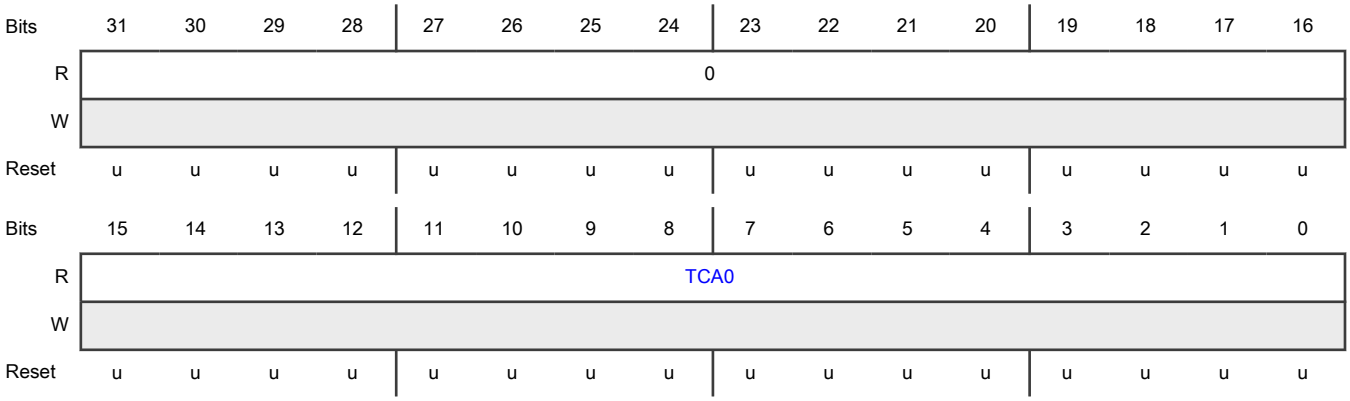
**Offset**

| Register | Offset |
|---|---|
| TCA0 | 8h |

**Function**

This register contains the coefficient TCA0 needed to calculate the temperature. the reset value is specific for each chip.

**Diagram**




**Fields**

| Field | Function |
|---|---|
| 31-16 | Reserved |

*Table continues on the next page...*

*Table continued from the previous page...*

| Field | Function |
|---|---|
| — | |
| 15-0<br>TCA0 | Temperature coefficient A0<br>see Conversion from voltage to temperature for the usage of this coefficient |

### 78.4.2.4 Temperature Coefficient (TCA1)

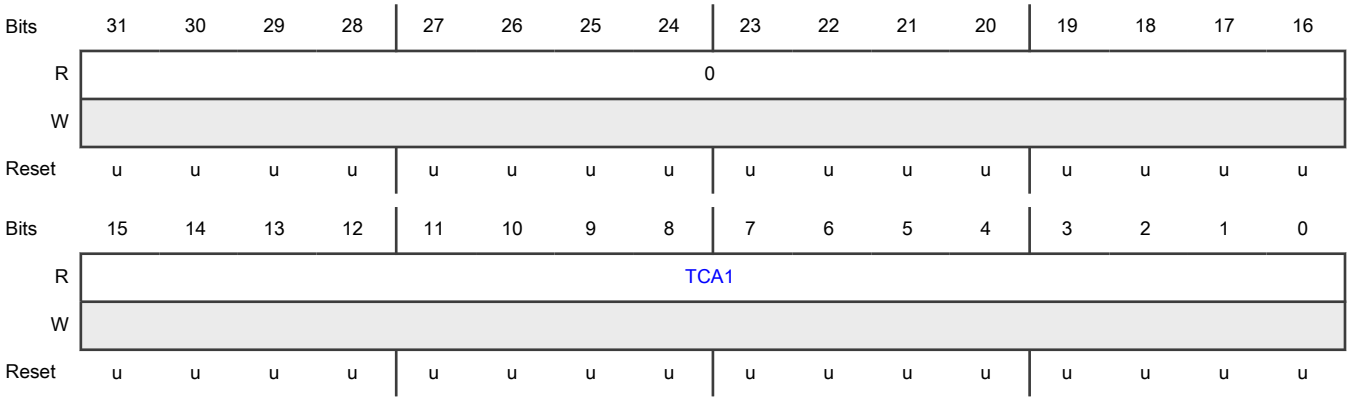
#### Offset

| Register | Offset |
|---|---|
| TCA1 | Ch |

#### Function

This register contains the coefficient TCA1 needed to calculate the temperature. the reset value is specific for each chip.

#### Diagram




#### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>TCA1 | Temperature coefficient A1<br>see Conversion from voltage to temperature for the usage of this coefficient |

### 78.4.2.5 Temperature Coefficient (TCA2)

#### Offset

| Register | Offset |
|---|---|
| TCA2 | 10h |

#### Function

This register contains the coefficient TCA2 needed to calculate the temperature. the reset value is specific for each chip.

#### Diagram

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

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | TCA2 | | | | | | | | | | | | | | | |
| W | | | | | | | | | | | | | | | | |
| Reset | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u | u |

#### Fields

| Field | Function |
|---|---|
| 31-16<br>— | Reserved |
| 15-0<br>TCA2 | Temperature coefficient A2<br>see Conversion from voltage to temperature for the usage of this coefficient |

## 78.5 Glossary

| | |
|---|---|
| **ADC** | Analog to digital converter |
| **ETS** | Engineering temperature sensor |
| **CTS** | Customer temperature sensor |
| **$V_{be}$** | Base emitter voltage |
| **$V_{ETS}$** | Engineering temperature sensor voltage |
| **$V_{GND}$** | Ground voltage |

# Appendix A
# Release notes

## A.1 About This Manual changes

| |
|---|
| No substantial content changes. |

## A.2 Introduction changes

| |
|---|
| No substantial content changes. |

## A.3 Memory map module changes

| |
|---|
| No substantial content changes. |

## A.4 Signal Multiplexing module changes

| |
|---|
| No substantial content changes. |

## A.5 Cortex-M7 changes

| |
|---|
| No substantial content changes. |

## A.6 Miscellaneous Control Module (MCM)

### A.6.1 Chip-specific MCM information changes

| |
|---|
| No substantial content changes. |

### A.6.2 MCM module changes

| |
|---|
| • In Local Memory Descriptor 0 (LMEM_DESC_0), updated the note to "The reserved bits can be read and written (instead of read as zero and write ignored). Setting any of these bits does not have any functional impact". |
| • Changed the bullet from "FPU invalid operation interrupt is enabled (FDZCE) and an invalid operation occurs (FDZC)" to "FPU invalid operation interrupt is enabled (FIOCE) and an invalid operation occurs (FIOC)" in Interrupts. |
| • Added PID register.<br>• Added more restrictions to note in Memory map and register descriptions. |
| • Updated with inclusive language. |

## A.7 Miscellaneous System Control Module (MSCM)

### A.7.1 Chip-specific MSCM information changes

| |
|---|
| No substantial content changes. |

### A.7.2 MSCM module changes

| |
|---|
| No substantial content changes. |

## A.8 Virtualization Wrapper (VIRT_WRAPPER)

### A.8.1 Chip-specific VIRT WRAPPER information changes

| |
|---|
| No substantial content changes. |

### A.8.2 VIRT_WRAPPER changes

| |
|---|
| • Updated the note from "Access to reserved spaces outside the register bank generates the transfer error. However, access to the reserved spaces within the register bank does not have any effect" to "Access to reserved spaces outside the register bank and holes (unimplemented registers) within register bank generates the transfer error.".<br>• Added the note "Any master can access holes (unimplemented registers) of SIUL2 through PDAC0." for REG_A and REG_B.<br>• Corrected the syntax to show the reserve RW Implemented hole bits within the registers REG_A and REG_B using MSCR_MAPPED parameter for REG_A and REG_B. |
| • Changed from PDAC to PDAC slot as applicable throughout the document.<br>• Added a note, "PDAC and PDAC slot are interchangeable terms." in Overview. |
| • Added section Initialization, Clocking, Application Information, and External signals. |
| • Made editorial updates. |
| • In Virtualization of interrupt control registers, added "The PDAC that...16 KB space" and "Access to mirrored...region".<br>• In Modes of operation, added see "Chip specific... for more details". |
| • Replaced "master" with "initiator" throughout the document.<br>• Removed, "This process is called ANY_MASTER" from PDAC-based protection scheme. |
| • Replaced "that is programmed by PDAC4 ... accessing PDAC0" with "that is programmed by specific PDAC ... accessing PDAC0" from Overview.<br>• Added a table in Register group mapping for SIUL2. |

## A.9 System Integration Unit Lite2 (SIUL2)

### A.9.1 Chip-specific SIUL2 information changes

| |
|---|
| No substantial content changes. |

### A.9.2 SIUL2 module changes

| |
|---|
| • In the note "For the array of 16-bit registers PGPDO*n* and PGPDI*n*": rephrased the second bullet-item. |

*Table continues on the next page...*

| |
|---|
| • Added field values 'Low' and 'High' for PGPDOn and PGPDIn registers. |
| • In External interrupt initialization added a new step 1 and added a note "When you chose external interrupt inputs for ...". |
| • Changed access type of reserved field from ROZ to RO in IFCPR and IFMCR*n* registers. |
| • Updated Figure 16.<br>• In the final note in register descriptions of MSCR*n* and IMCR*n*, added list items:<br>— "These registers are a part of SIUL memory map but physical implementation of these register is a part of IOMUX RTL.<br>— SIUL2 interprets accesses to MSCRn at module level." |
| • Throughout register descriptions, changed every instance of the character Á to × |
| • Added External signal description<br>• Moved block diagram related information to a new section: Block diagram<br>• Changed section title form "External interrupts/DMA requests (REQ Pins)" to External interrupts<br>• Added Clocking and has the following content: "This module has no clocking considerations".<br>• Added Initialization and has the following content: "This module does not require initialization".<br>• Updated DMA requests<br>• Throughout the register section, removed 'Register' from the register full names. |
| • In , replaced the existing content with: "See the IOMUX file attached to this document for more information". |
| • In Block diagram, mentioned that REQ pins mean interrupt request pins.<br>• In External interrupts, mentioned that IRC means Internal RC Oscillator. |
| • Updated the Block diagram. |
| • IFCPR register is removed. |

## A.10 Touch Sensing Pin Coupling (TSPC)

### A.10.1 Chip-specific TSPC information changes

| |
|---|
| No substantial content changes. |

### A.10.2 TSPC module changes

| |
|---|
| • Made editorial updates:<br>— Rephrased Overview.<br>— Added the following in Functional description "*... if any OBE within a group transitions from high to low, it enables that group of OBEs for simultaneous transition from high to low ...*". |
| • Made editorial changes. |

*Table continues on the next page...*

| |
|---|
| • Moved TSPC register descriptions to the end of TSPC chapter, just before Glossary. |
| • In Logic block description, replaced "gated output buffer enables" with "gated OBEs". |
| • Added topic Submodules.<br>• Added topic Operation.<br>• Added topic Clocking.<br>• Added topic External signals.<br>• Added topic Application information. |
| • Added sentence "See the chip-specific TSPC configuration for group of pads information" in Features. |

## A.11 Crossbar Switch (AXBS)

### A.11.1 Chip-specific AXBS information changes

| |
|---|
| No substantial content changes. |

### A.11.2 AXBS module changes

| |
|---|
| • Replaced "Master with Initiator" and "Slave with Target". |

## A.12 Peripheral Bridge (AIPS_Lite)

### A.12.1 Chip-specific AIPS_Lite information changes

| |
|---|
| No substantial content changes. |

### A.12.2 AIPS_Lite module changes

| |
|---|
| No substantial content changes. |

## A.13 Direct Memory Access Multiplexer (DMAMUX)

### A.13.1 Chip-specific DMAMUX information changes

| |
|---|
| No substantial content changes. |

### A.13.2 DMAMUX module changes

| |
|---|
| • Split the memory map within the DMAMUX register descriptions section into 2 sets of registers: one for channels having triggering capability (having TRIG bit) and the other for channels that don't have triggering capability (and hence no TRIG bit). |

*Table continues on the next page...*

| |
|---|
| • Split the memory map within the DMAMUX register descriptions section into 2 sets of registers: one for channels having triggering capability (having TRIG bit) and the other for channels that don't have triggering capability (and hence no TRIG bit).<br>• Added the footnote within the memory map as follows: "In this array, the index and offset values of the registers do not increment in direct alignment. For details, see the register description." |
| • Added the section Glossary and cross-referenced the first-time usage of the terms. |
| • Updated Glossary. |
| • Updated the section Functional description. |
| • Updated Block diagram.<br>• Updated the image "DMAMUX triggered channels" (Figure 24).<br>• Added a short introduction in Modes of operation. |

## A.14 Enhanced Direct Memory Access (eDMA)

### A.14.1 Chip-specific eDMA information changes

| |
|---|
| No substantial content changes. |

### A.14.2 eDMA3 module changes

| |
|---|
| No substantial content changes. |

## A.15 INTM module changes

| |
|---|
| • Added Register links all over the document.<br>• Updated Figure 34.<br>• Updated Registers, Initialization, Functional description topics.<br>• Topic Glossary removed. |
| • Updated Figure 34. |
| • Updated Functional description with a note to mention only level type interrupts are supported. |
| • Added description of the term FCCU in bullet no.5 of the topic Initialization. |
| • Updated Figure 34, added INTM_IRQSEL. |
| • Updated note in Functional description section. |
| • Removed ITRC occurence from Functional description.<br>• For INTM_TIMER0-INTM_TIMER3 registers, updated access of TIMER bit field to read only |
| • For INTM_TIMER0-INTM_TIMER3 registers, updated access of TIMER bit field to Read Write |

## A.16 Semaphores2 (SEMA42)

### A.16.1 Chip-specific SEMA42 information changes

| |
|---|
| No substantial content changes. |

### A.16.2 SEMA42 module changes

| |
|---|
| • Moved the block diagram to a new section Block diagram.<br>• Added following sections:<br>— Clocking<br>— Interrupts<br>— External signals<br>— Initialization<br>• Marked RSTGT_R[ROZ] as a reserved field in the field description table.<br>• Updated Functional description and Features.<br>• Updated Glossary.<br>• Updated GATEn state machine. |
| • Moved the Multi-core programming: software gates section to Functional description.<br>• Clarified the Function and field descriptions of Reset Gate Read (RSTGT_R) and Reset Gate Write (RSTGT_W) register. |
| • Updated Clocking section. |
| • Moved the "Initialization" and "External signals" section above "Memory map/register defination" section. |
| • Added NOTE in Overview section. |

## A.17 Crossbar Integrity Checker (XBIC)

### A.17.1 Chip-specific XBIC information changes

| |
|---|
| No substantial content changes. |

### A.17.2 XBIC module changes

| |
|---|
| • ESR.VLD: Updated the access to W1C |
| • Changed "Unsupported access type—an attempted write to a read-only register, for example" to "Writes to read-only registers generate a transfer error" in "XBIC register description" section<br>• Added text "This module has no interrupts." in the Interrupts section<br>• Added text "XBIC has no external interface signals." in the External signals section<br>• Added text "This module does not require initialization." in the Initialization section |

*Table continues on the next page...*

| |
|---|
| • Added Glossary section. Glossary<br>• Removed section "Signal description" and moved the text "XBIC has no external interface signals." to External signals section |
| • Removed "Writes to read-only registers generate a transfer error" from "XBIC register descriptions" section.<br>• Added note "Writes to read-only registers generate a transfer error" in the description of register XBIC Error Address (EAR)EAR. |
| • Changed the access type of bitfield "VLD" in register ESR from "RO" to "RW".<br>• Updated note in register XBIC Error Address (EAR) to "An attempted write to this read-only register results in a transfer error". |
| • Updated "Overview" section. |
| • Changed the access type of bitfield "VLD" of register ESR from "RW" to "RO". |
| • Updated the "Features". |
| • Replaced "Master with Initiator" and "Slave with Target". |

## A.18 Extended Resource Domain Controller (XRDC)

### A.18.1 Chip-specific XRDC information changes

| |
|---|
| No substantial content changes. |

### A.18.2 XRDC module changes

| |
|---|
| • Updated description of MRGD_W3_x[VLD] |
| • Overview:<br>— Added 'If the value of MDACFG*m*[NMDAR] is 1 ... rights granted for this DID.' |
| • Updated Process Identifier register.<br>• Added a bullet point "Register settings for DFMT1 direct domain assignment transactions" in Overview.<br>• UpdatedPID-based domain assignment and Transaction request sources.<br>• Added a bullet point "DFMT1 direct domain assignment example-single MDA" in XRDC transaction examples.<br>• Added DFMT1 direct domain assignment example-single MDA and its sub-sections. |
| • In MDA_W*r_m*_DFMT1[PA] field description Note, changed "secure" to "privileged" |
| • Added the text "This module has no clocking considerations" in Clocking<br>• Added Interrupts<br>• Added a note in MDA register structure |
| • Updated description of (MDA_Wn_m_DFMT0[VLD]) and (MDA_Wn_m_DFMT1[VLD]) |

*Table continues on the next page...*

| |
|---|
| • Added Exceptions and violations. |
| • Removed "Zipwire" and "FlexRay" from Transaction protection. |
| • Updated Handling domain access violation errors.<br>• Updated the register descriptions for DERR_W0_i, DERR_W1_i, and DERR_W3_i. |
| • Updated the register descriptions for DERR_W0_i and DERR_W1_i. |
| • Replaced "master" to "initiator" throughout the document. |
| • Added a note for deprecated terms in Overview. |
| • Updated Block descriptions, Peripheral ACP evaluation example, and Memory ACP evaluation example. |

## A.19 Memories and Memory Interfaces changes

| |
|---|
| No substantial content changes. |

## A.20 Embedded Flash Memory (c40asf)

### A.20.1 Chip-specific C40asf information changes

| |
|---|
| No substantial content changes. |

### A.20.2 Embedded Flash Memory (c40asf)

| |
|---|
| No substantial content changes. |

## A.21 Flash Memory Controller (PFLASH)

### A.21.1 Chip-specific PFLASH information changes

| |
|---|
| No substantial content changes. |

### A.21.2 PFLASH controller module changes

| |
|---|
| • Added PFAPR[M3AP] bit. |

## A.22 RAM Controller (PRAMC)

### A.22.1 Chip-specific PRAMC information changes

| |
|---|
| No substantial content changes. |

### A.22.2 PRAMC module changes

- Divided the Overview section to three sections: Overview, Block diagram, and Features.
- Added section for clocking, Clocking, and reset, Reset .
- Added Reset.
- Added information that the module has no external signals in the section External signals.
- Added information that the module has no interrupts in the section Interrupts.

- Block diagram - Removed internal content from the section.

## A.23 Clocking module changes

No substantial content changes.

## A.24 Clock Generation Module (MC_CGM)

### A.24.1 Chip-specific MC_CGM information changes

No substantial content changes.

### A.24.2 MC_CGM module changes

- Added following note in the MUX_n_DC_m register: Software-controlled clock multiplexer dividers are not expected to return to the default state on the hardware transitions and handshakes occurring as part of the functional reset entry sequence (only hardware-controlled clock multiplexer dividers return to the default state).

- Added following note in the MUX_n_DIV_UPD_STAT[DIV_STAT] field: This field clears once divider configuration is updated or on destructive reset. If functional reset comes when this field is 1 then it can remain fixed to 1 until divider input clock is restored.

## A.25 FIRC module changes

- Added text "This module has no external signals." in External signals section
- Added text "This module does not require initialization." to Initialization section

## A.26 SIRC module changes

- Added text "This module has no external signals" in the External signals section
- Added text "This module does not require initialization." in the Initialization section

## A.27 Fast Crystal Oscillator Digital Controller (FXOSC)

### A.27.1 Chip-specific FXOSC information changes

No substantial content changes.

### A.27.2 FXOSC module changes

| |
|---|
| • Added the CTRL[ALC_D] field. |
| • Updated Overview section as follows:<br>— Changed the title from "Introduction" to "Overview" in overview.dita file.<br>— Added "The Fast Crystal Oscillator (FXOSC) generates a clock which can be used at the SoC level. The FXOSC has a digital interface to control and configure the oscillator to generate multiple frequencies. When FXOSC is powered down at any time, it is designed not to generate any glitch at the output clock. A counter inside FXOSC handles different stabilization times." in the overview section.<br>— Commented out "This module provides the fast crystal oscillator to the chip." in overview section.<br>• Removed the note "For bypass mode applications, the EXTAL pin should be driven low when FXOSC is in off/disabled state" from "Functional Description" section.<br>• Removed the following notes from "Initialization section"<br>— "When the FXOSC is used in Bypass mode, the external clock source can only be enabled after the FXOSC is enabled."<br>— "When the FXOSC is used in Bypass mode, the external clock source must already be inactive before disabling the FXOSC."<br>• Added the new section Block diagram, , Clocking, External signals and Interrupts. |
| • Made changes in block diagram<br>• Replaced the content of Clocking section with "This module has no clocking considerations."<br>• Removed “to generate multiple frequencies” from the Overview section<br>• Added "CG cell is clock gating cell, it gates the clock till stabilization time." in Overview section<br>• Added note "When the FXOSC is used in Bypass mode, the clock from the external source can only be used after the FXOSC is enabled" in Initialization section. |
| • Updated Block diagram section |
| • Added long description to CTRL[ALC_D].<br>• In Initialization- Added a note "See Hardware design guide for further details and the recommended circuit for each mode." |
| • Updated the section Clock generation in Single-Input Bypass mode. |

## A.28 SXOSC module changes

| |
|---|
| • Updated the reset value of CTRL register. |
| • Updated the bit field value of CURR_PRG_COMP, CURR_PRG_SF and GM_SEL. |
| • As per TKT0613245, parameterized the Bypass mode(to hide it for S32K3xx) and other points are already updated as per TKT0611802. |
| • Fixed the shared review comments received on S32K5 RM. |

## A.29 PLL Digital Interface (PLLDIG)

### A.29.1 Chip-specific PLLDIG information changes

| |
|---|
| No substantial content changes. |

### A.29.2 PLLDIG module changes

| |
|---|
| • Note "The value of this field is 0b when the module is under reset." added in PLLSR[CGE_OUT2 and CGE_OUT1] register. |
| • Maximum possible modulation depth formula added in "Frequency modulation programming" topic. |
| • Maximum possible modulation depth formula added for "When PLL[RDIV] is 0" in "Frequency modulation programming" topic.<br>• LOL abbreviation added in LOL topic. |

## A.30 Reset module changes

| |
|---|
| No substantial content changes. |

## A.31 Boot Overview changes

| |
|---|
| No substantial content changes. |

## A.32 Reset Generation Module (MC_RGM)

### A.32.1 Chip-specific MC_RGM information changes

| |
|---|
| No substantial content changes. |

### A.32.2 MC_RGM module changes

| |
|---|
| • Added following note in section Reset state machine: See chip-specific MC_RGM information for chip-specific reset sequence details. |
| • Added following note in DES[F_POR] field: All reset status in the FES and DES registers must be ignored at power up if this field is set. |
| • Removed the following content from Functional reset entry timer: See the chip-specific section for the default value of timeout value and the details on the chip behavior in case of functional reset entry timer timeout.<br>• Added section: Alternate event generation. |
| • Removed the following statement from RESET_B pin assertion and pin safe state control: See STCU2 chapter for more details on its usage.<br>• Added following note in Functional reset escalation: Functional counter increments for each reset source for which escalation is enabled. For details, see "Reset sources" table in the Reset chapter. |

*Table continues on the next page...*

| |
|---|
| • Added following note in Destructive reset escalation: Destructive counter increments for each reset source for which escalation is enabled. For details, see "Reset sources" table in the Reset chapter. |
| • Editorial changes.<br>• Updated Figure 133.<br>• In Features, removed the first three features and added the following:<br>— Destructive and functional reset management<br>— Capturing the reset sources for each reset sequence (reset status flags)<br>— Assertion the Reset_B pin to propagate the reset sequence out of chip<br>• Changed the "Reset value" from 0 to 1 in Table 182.<br>• Updated the description in Table 182.<br>• Updated the last statement in RESET_B pin assertion and pin safe state control |
| • In Functional reset entry sequence, updated the second last paragraph for clarity.<br>• In Functional reset exit sequence, updated the second bullet.<br>• Moved content from External reset to RESET_B assertion control.<br>• Renamed the "External reset assertion control" section to RESET_B assertion control.<br>• Updated the second paragraph of the Destructive reset escalation for clarity.<br>• Updated the description of FREC, FRET[FRET] and DRET[DRET]. |
| • In Reset sources, removed "and the MC_RGM itself" from the last statement of the third paragraph.<br>• In Reset sources, removed "(for example, debug modules, flash modules)" from the last statement of the last paragraph. |
| • Updated the Figure 134. |
| • Updated the note in Functional /External Reset Status Register (FES).<br>• In External signal description, updated the reset value of "RESET_B" signal. |
| • Updated the note in Functional /External Reset Status Register (FES).<br>• Updated Figure 133 and Figure 134.<br>• Removed "(except for a small portion, for example FIRC, JTAG interface)" from Reset sources.<br>• Updated references to reset chapter to make them generic. |
| • Updated the note in DES[F_POR].<br>• Updated the note in Functional /External Reset Status Register (FES). |
| • Updated the note in Functional /External Reset Status Register (FES). |
| • Updated the description of the Functional /External Reset Status Register (FES). |
| • Removed the fields: MODE_CONF[POR_RST], MODE_CONF[FULL_WKUP], MODE_CONF[Standby]. |
| • Updated Figure 134. |

*Table continues on the next page...*

| |
|---|
| • Removed Standby mode related information from the chapter. |
| • Removed "LPRR_CONF" register. |

## A.33 POR_WDG module changes

| |
|---|
| No substantial content changes. |

## A.34 Security overview changes

| |
|---|
| No substantial content changes. |

## A.35 HSE_B module changes

| |
|---|
| No substantial content changes. |

## A.36 DCF module changes

| |
|---|
| No substantial content changes. |

## A.37 Chip-specific DCM information changes

| |
|---|
| No substantial content changes. |

## A.38 DCM module changes

| |
|---|
| • Removed the content related to CUST_DEL, NXP_PWD and MCU_PROD from the chapter. Removed the DCMLCC[DCMFLC] and DCMLCC[DCMLCFN] bits. |
| • Changed the access of Reserved bits 3,8 and 9 from ROz to RW in DCMLCC register. |
| • Changed the access of Reserved bits 0 to 11 from ROz to W1C in DCMLCS register. |
| • Revised Overview, Figure 139, and Features.<br>• Revised DCF mechanism, removed the content duplicate with DCF chapter of RM.<br>• Revised DCF client error mechanism, changed CERS to DCMCERS in figure title.<br>• Revised LC to update information of valid and invalid field types, and Table 196 to change PFA to Pre_FA.<br>• Revised Glossary, added OTP and DCF.<br>• Deleted DCMSTAT[9] DCMOTAS.<br>• Revised the description of DCM Status (DCMSTAT), LC and LC Control (DCMLCC), LC Scan Status (DCMLCS), LC Scan Status 2 (DCMLCS_2), DCMSTAT[DCMLCST], , DCMMISC[DCMDBGE], DCMMISC[DCMDBGT], DCF Error Count (DCMEC), DCMEC[DCMECT], DCMSRRn, DCMSRR1[DCMESF1], and DCMSRR1[DCMDCFF1]. |
| • Revised DCM error detection mechanism and Figure 141. Also, changed DCF to DCM in topic and figure title. |

- Revised DCF client error mechanism, Figure 140 and changed its title to "DCF error handling".
- Revised Features, updated debug related information.
- In LC, revised Table 196.
- In Glossary, added PFC.
- Revised description of LC and LC Control (DCMLCC).
- Revised DCMLCS[DCMLCE5], DCMDEB[DCM_APPDBG_STAT], DCMSRR1[DCMDCFF1] .

## A.39 Messaging Unit (MU)

### A.39.1 Chip-specific MU information changes

No substantial content changes.

### A.39.2 MU module changes

- Updated the reset value of the SR[MURS] register bit field as undefined.

- Revised Figure 143 - changed "Processor A side" and "Processor B side" to "MUA" and "MUB" respectively.
- Added new topics Submodules and Initialization.
- Updated Glossary.
- Added Clocking - "This module has no clocking considerations.".

- In Glossary and Features, corrected MUA and MUB from Message Unit A and Message Unit B to Messaging Unit A and Messaging Unit B respectively.

- Changed "external memory", and "memory array" to "shared memory" in Functional description.
- Added new clock "Bus clock" in "Clocking".
- Removed topic "Wake up core in low power mode".
- Revised Interrupts topic.
- Revised Table 201.
- Revised , TR0-TR3[TR_DATA], RR0-RR3[RR_DATA].
- Revised Table 206.

- Revised access type and reset value of CCR0[2].
- Revised access type and reset value of CCR0[4].

- Added note in MUA side and MUB side.

- Updated Asynchronous system reset.

- Deleted topic "Sending messages with interrupt" and revised Initialization to add information "MU does not require initialization.".

- Updated topics MUA side and MUB side, revised notes.

*Table continues on the next page...*

| |
|---|
| • Updated the description of SR[TEP] and RSR[RFn]. |
| • Updated receive and transmit interrupts related points in Interrupts. |

## A.40 Power management changes

| |
|---|
| No substantial content changes. |

## A.41 PMC changes

| |
|---|
| No substantial content changes. |

## A.42 PMC changes

| |
|---|
| No substantial content changes. |

## A.43 Mode Entry Module (MC_ME)

### A.43.1 Chip-specific MC_ME information changes

| |
|---|
| No substantial content changes. |

### A.43.2 MC_ME module changes

| |
|---|
| • Added following note in Partition 0 Process Configuration Register (PRTN0_PCONF), Partition 0 Process Update Register (PRTN0_PUPD), Partition 0 Status Register (PRTN0_STAT): See chip-specific MC_ME information to check if this register is implemented on chip. |
| • Added Main Core ID Register (MAIN_COREID). |

## A.44 MC_PCU changes

| |
|---|
| • Updated Table 217. |
| • Updated Power sequence FSM. |
| • Updated first statement in the second paragraph of Introduction. |

## A.45 Wakeup Unit (WKPU)

### A.45.1 Chip-specific WKPU information changes

| |
|---|
| No substantial content changes. |

### A.45.2 WKPU module changes

| |
|---|
| No substantial content changes. |

## A.46 Safety overview changes

| |
|---|
| No substantial content changes. |

## A.47 Error Injection Module (EIM)

### A.47.1 Chip-specific EIM information changes

| |
|---|
| No substantial content changes. |

### A.47.2 EIM module changes

| |
|---|
| • In Functional description, added information about enabling/disabling error injection. |
| • Added Initialization section stating that this module does not require initialization. |
| • Updated the definition of SEC-DED. |
| • Updated the first paragraph of Overview. |
| • Updated revision history title to "EIM module changes". |
| • Added NOTE in Overview section. |

## A.48 Error Reporting Module (ERM)

### A.48.1 Chip-specific ERM information changes

| |
|---|
| No substantial content changes. |

### A.48.2 ERM module changes

| |
|---|
| • Updated the description of the SR registers. |
| • Added following note in the "ERM register descriptions" section: To access the channel registers, corresponding channel clocks must be enabled. |
| • Added following note in the Non-correctable error events: Parity errors can be mapped to non-correctable errors where error attributes like SYNDROME, ADDRESS are not provided.<br>• Updated the last statement of the Overview as follows: ERM does not capture syndrome or error address for cache memories or memory with parity instead of ECC.<br>• Added following note in the "ERM Memory a Correctable Error Count Register": Non-correctable errors are considered a serious fault, so the ERM does not provide any mechanism to count non-correctable errors. Only correctable errors are counted. |
| • Removed the following statement from the CR*n*[ENCIE*m*] and CR*n*[ESCIE*m*] fields: This field is initialized by hardware reset. |

*Table continues on the next page...*

| |
|---|
| • Added following note in the "ERM Memory a Error Address Register": This register provides the offset of error location and not the absolute address. For example, if memory is SRAM0 (which starts from 0x20400000) if error occurs at 0x20400010 the EAR register gives value 0x10 and not 0x20400010. |
| • Removed following note from the "ERM Memory a Error Address Register": This register provides the offset of error location and not the absolute address. For example, if memory is SRAM0 (which starts from 0x20400000) if error occurs at 0x20400010 the EAR register gives value 0x10 and not 0x20400010. |
| • In introductory text of "ERM register descriptions", added information related to the XFR error behavior at the IPS interface.<br>• Added Figure 178. |

## A.49 Fault Collection and Control Unit (FCCU)

### A.49.1 Chip-specific FCCU information changes

| |
|---|
| No substantial content changes. |

### A.49.2 FCCU module changes

| |
|---|
| • Note "Software reset of FCCU by MC_RGM_PRST5[FCCU_RST] bit is optional since FCCU registers are not initialized during functional reset. MC_RGM_PRST5[FCCU_RST] has the same effect on FCCU as power on reset. If the bit is activated, it can change the state of the error pin, if the current state of the pin is different from default." removed from topic "Use cases and limitations". |
| • Reset interface topic moved into chip specific section. |
| • The equation changed from "Tmin = 250 µs + DELTA_T to "Tmin = (250us + DELTA_T) * (SAFE_CLK_FREQ/CLKSAFE freq). |
| • Added a topic External signals. |
| • Added EIN input signal in block diagram. |
| • Note "The FCCU is not reset by short functional resets" deleted from BG. |
| • Updated the description of "Not Fault-Toggle" value of Timer state in FAULT state of Fault-Output Timer (TMR_ETMR) register. |
| • Updated the External signals. |
| • For EINOUT, updated the bit field description for EINOUT[EIN1] and EINOUT[EIN0]. |

## A.50 SELFTEST_GPR module changes

| |
|---|
| No substantial content changes. |

## A.51 Self-Test Control Unit (STCU2)

### A.51.1 Chip-specific STCU2 information changes

| |
|---|
| No substantial content changes. |

### A.51.2 STCU2 module changes

| |
|---|
| • In Example: Multi-phase BIST sequence, changed the BIST number from 38 to 30 for phase 0. |
| • Removed MTR from the function of "MBIST interface" in Table 240. |
| • Updated the description of the STCU2 Online LBIST Unrecoverable FM (LBUFM0). |
| • Removed instances of "INT_FLG" from Peripheral bus interface. |
| • Added title to Figure 189. |
| • Updated the description of STCU2 SK Code (SKC). |
| • Removed topic "Use cases and limitations".<br>• Revised FSM description, changed state machine to FSM.<br>• Revised Peripheral bus interface and Glossary.<br>• Revised description for CFG[PTR], CFG[CLK_CFG] , CFG[LB_DELAY] , LB_CTRL0[PTR], and MB_CTRLn[PTR].<br>• Revised description for MBSSWn and MBESWn. |

## A.52 Register Protection (REG_PROT)

### A.52.1 Chip-specific REGPROT information changes

| |
|---|
| No substantial content changes. |

### A.52.2 REG_PROT module changes

| |
|---|
| No substantial content changes. |

## A.53 Clock Monitoring Unit – Frequency Check (CMU_FC)

### A.53.1 Chip-specific CMU_FC information changes

| |
|---|
| No substantial content changes. |

### A.53.2 CMU_FC module changes

| |
|---|
| • Updated Features, to remove clocking information.<br>• Added separate section for clocking, Clocking. |

*Table continues on the next page...*

| |
|---|
| • Added Reset.<br>• Added information that the module has no external signals in the section External signals. |
| • In Features<br>— Changed "Timeout functionality generates an FLL event if a monitored clock stops functioning" to "Timeout functionality generates an FLL event (if enable) when monitored clock is very slow or stop functioning." |
| • Updated the Features. |
| • Updated the Operating modes. |

## A.54 Clock Monitoring Unit – Frequency Meter (CMU_FM)

### A.54.1 Chip-specific CMU_FM information changes

| |
|---|
| No substantial content changes. |

### A.54.2 CMU_FM module changes

| |
|---|
| • Updated Features, to remove clocking information.<br>• Added separate section for clocking, Clocking.<br>• Added Reset.<br>• Added information that the module has no external signals in the section External signals. |
| • Fixed the Minimum RCCR[REF_CNT] formula calculation in Programming RCCR[REF_CNT]. |

## A.55 Cyclic Redundancy Check (CRC)

### A.55.1 Chip-specific CRC information changes

| |
|---|
| No substantial content changes. |

### A.55.2 CRC module changes

| |
|---|
| • Renamed the section "Introduction" to Overview<br>• Added the section External signals.<br>• Moved the section Modes of operation within the Functional description section.<br>• Moved the section Initialization after the Functional description section. |
| • Updated the following sections: Features, Calculating a 16-bit CRC, Memory map and register descriptions, Expected read data fields, and CTRL programming |
| • Updated the following section: Memory map and register descriptions. |
| • Updated Calculating a 32-bit CRC. |

*Table continues on the next page...*

| |
|---|
| • Added Clocking and Interrupts.<br>• Renamed section title "CRC initialization and reinitialization" to "Initialization".<br>• Updated the long description of CTRL[WAS]. |
| • Updated Clocking. |
| • In section Calculating a 16-bit CRC cross reference in step 2 “Program the transpose and complement options in Data (DATA) as required for the CRC calculation” )" changed to "Control (CTRL)".<br>• In section Calculating a 32-bit CRC cross reference in step 7 “CRC is computed on every data value write and the intermediate CRC result is stored back into DATA[LU] and DATA[LL]” changed to "DATA".<br>• In register DATA[HL] long description updated to "Generates CRC checksum in both 16-bit and 32-bit CRC modes if CTRL[WAS] = 0." |
| • Updated the description of Data (DATA) register.<br>• Updated the description of DATA[LL]. |
| • Updated Low-power modes and Clocking |

## A.56 PCMC changes

| |
|---|
| No substantial content changes. |

## A.57 Analog-to-Digital Converter (ADC)

### A.57.1 Chip-specific ADC information changes

| |
|---|
| No substantial content changes. |

### A.57.2 ADC module changes

| |
|---|
| • In AMSIO register: reserved bit AMSIO[16] was made visible to customer as AMSIO[CMPCTRL0], deleted text from function "you must write 3d to HSEN" and updated the description of field 18-17.<br>• In Clock section, the table "Clock configuration for highest conversion speeds" was updated with the new recommendation, that is for freq > 80MZ the setting AMSIO[HSEN]=01b and AMSIO[CMPCTRL0]=1b shall be used.<br>• In Clock section, deleted text "For some conversion clock frequencies, you must write 3d to AMSIO[HSEN]".<br>• In Conversion time section, change the text from "The evaluation time of a single bit is 4 AD_clk cycles if AMSIO[HS] = 0h, and 5 AD_clk cycles if AMSIO[HS] = 1h." to "The evaluation time of a single bit is 4 AD_clk cycles if AMSIO[HSEN] = 0h and AMSIO[CMPCTRL0] = 0b. The evaluation time of a single bit is 6 AD_clk cycles if AMSIO[HSEN] = 1h and AMSIO[CMPCTRL0] = 1b." |
| • Deleted statement about analog watchdog programming from topic Analog watchdog configuration (up to 16 watchdogs) "The threshold values to be selected for a channel must be programmed only once in CWSELRaIn." |
| • CDATA bit field description updated to "Contains conversion data from standard input n, determined by the SAR algorithm. The conversion data bit count is dependent on the conversion resolution selected (CALBISTREG[RESN]). Depending on the value of MCR[WLSIDE], the conversion data MSB bits start from 14 (MCR[WLSIDE] = 0) or 15 (MCR[WLSIDE] = 1). |

## A.58 Low Power Comparator (LPCMP)

### A.58.1 Chip-specific LPCMP information changes

| |
|---|
| No substantial content changes. |

### A.58.2 LPCMP module changes

| |
|---|
| • Removed references to "hard block" from CCR2[HYSTCTR] in register file for clarity.<br>• In Functional description, edited the title. |
| • Edited register file. |
| • Added Glossary of terms. |
| • Changed the name of the "Clocks" section to Clocking.<br>• Changed the name of the "Trigger Mode" sectionto Round-robin trigger mode and clarified the full name of the Round Robin Trigger mode.<br>• Clarified Latency issues.<br>• Added register links to Initialization. |
| • Updated External signal descriptions with RR_ACTIVE related description.<br>• Updated Block diagram with RR_ACTIVE and Round_robin FSM.<br>• Updated Round-robin trigger mode with RR_ACTIVE wave form. |
| • Updated Block diagram, including input channels and register references using REG[FIELD].<br>• Updated Initialization.<br>• Updated Application information. |
| • Reserved CCR3 register in LPCMP register descriptions. |
| • Updated LPCMP register descriptions by adding a note to Comparator Status (CSR). |
| • Updated Round-robin clock (RCLK) frequency requirement by changing "ms" to "µs". |
| • In LPCMP register descriptions, updated "ACMP" with "LPCMP" in the Note of CSR register. |
| • Added Version ID (VERID) register. |

## A.59 Logic Control Unit (LCU)

### A.59.1 Chip-specific LCU information changes

| |
|---|
| No substantial content changes. |

### A.59.2 LCU module

| |
|---|
| • Updated block diagram to show outputs that go to IO pads |
| • LC Diagram Revised the LC Diagram to update the LC0_FCTRL[FORCE_MODE0] mux values from 0 1 2 3 to 0 2 1 3.<br>• LC n Sync Control Revised the conditions for bit 8 and 9 to show bit 9 as Reserved whenever the input sync value is 1 or 2.<br>• Mux Select Added note "Access to Address space 200h + (a x 4h) does not generate a transfer error". |
| • Updated description of CFG[WP].<br>• Updated the bit information in title for Mapping of 12-bit input, output, and state fields to reflect the correct bit value. |
| • Changed bit 9 of LC n Sync Control (LC0_SCTRL - LC2_SCTRL) to RW from ROZ. |
| • Added note in LC output filters.<br>• Added new topic DMA.<br>• Revised Interrupts. |
| • Revised the long description for LC0_FFILT[FORCE_POL]. |
| • Removed "Glossary" section. |
| • Revised Features, LC diagram.<br>• Revised Figure 251, and Figure 252 to remove sync 2 and 3 inputs.<br>• Revised the long description of DBGEN[DBGEN]. |
| • Changed 1h to Eh in Table 299 for LUTCTRL0.<br>• Revised long description for LC0_SCTRL[SW_MODE].<br>• Revised Features to update register information.<br>• Revised Figure 251 |
| • Revised Block diagram to remove the note "3 LCs are shown in this block diagram.".<br>• Revised Figure 252. |

## A.60 Enhanced Modular IO Subsystem (eMIOS)

### A.60.1 Chip-specific eMIOS information changes

| |
|---|
| No substantial content changes. |

### A.60.2 eMIOS module changes

| |
|---|
| • Changed bit position from 0 to 31 for AS2 in Effect of C*n*[EDPOL] on edge capturing and SAIC submodes and MODE field values<br>• Added bitfield A*n*[RISE_FALL]. |

*Table continues on the next page...*

| |
|---|
| • Editorial updates. |
| • Deleted counter bus E from figures in UC block diagram and UC control and datapath. |

## A.61 Body Cross-triggering Unit (BCTU)

### A.61.1 Chip-specific BCTU information changes

| |
|---|
| No substantial content changes. |

### A.61.2 BCTU module changes

| |
|---|
| • Topic External Signals removed. |
| • Minor correction in figures of Trigger timing and Simultaneous trigger priority management.<br>• Updated Implementation. |
| • Updated no of triggers and ADC to (n) in Block diagram. |

## A.62 Trigger MUX (TRGMUX)

### A.62.1 Chip-specific TRGMUX information changes

| |
|---|
| No substantial content changes. |

### A.62.2 TRGMUX module changes

| |
|---|
| • Renamed the "Introduction" section to Overview.<br>• Nested Features within the "Overview" section. |
| • In #unique_3756/unique_3756_Connect_42_bitFieldTable, updated values for the SELx field from binary to hex format.<br>• In #unique_3757, updated description for SELn fields.<br>• Added Block diagram<br>• Updated Features to remove block diagram.<br>• Added Functional description |
| • Removed "Memory map and register definition" section. |
| • Removed "Memory map and register definition" section:<br>— Moved following content from the "Memory map and register definition" section to "TRGMUX memory map" sections:<br>"You can only write to TRGMUX registers in Supervisor mode."<br>— Removed redundant content "The TRGMUX registers contain fields for selecting trigger sources for peripheral modules." |

*Table continues on the next page...*

| |
|---|
| • Editorial updates:<br>— Renamed all the registers from "TRGMUX *Reg_Name* Register (*Reg_Name*)" to "TRGMUX *Reg_Name* (*Reg_Name*)", here *Reg_Name* refers to the register name.<br>— Updated the description of all the fields in all the registers. |
| • In Block diagram, updated reference to chip-specific TRGMUX information.<br>• In #unique_3757, removed reference to chip-specific TRGMUX information. |

## A.63 Software Watchdog Timer (SWT)

### A.63.1 Chip-specific SWT information changes

| |
|---|
| No substantial content changes. |

### A.63.2 SWT module changes

| |
|---|
| • In CR[FRZ], replaced "chip" with "core." |
| • In Behavior in different chip and core modes, replaced "chip" with "core." |
| • Edited throughout.<br>• Added Block diagram section.<br>• Added Clocking section.<br>• Added Reset section.<br>• Added Interrupts section. |
| • Updated Block diagram section.<br>• Updated Reset section. |
| • Updated Initialize the SWT section to remove the bullet "Select the clock source (if there are more than one possible sources)".<br>• Removed 'Select the clock source' section.<br>• Updated Clocking section to add the following details "See the chip-specific SWT information to find the clock source that drives the countdown timer." |
| • Updated terminology in this chapter as follows:<br>— Updated "Master" to "Initiator" |
| • In Clocking, added reference to the chip-specific SWT information for the clock source of the counter and reset logic. |

## A.64 System Timer Module (STM)

### A.64.1 Chip-specific STM information changes

| |
|---|
| No substantial content changes. |

### A.64.2 STM module changes

| |
|---|
| • Renamed "Introduction" to Overview.<br>• Restructured to conform to BGAC outline.<br>• Added the Clocking topic.<br>• Added the Interrupts topic.<br>• Restored the External signals topic.<br>• Added the Initialization topic. |

## A.65 Periodic Interrupt Timer (PIT)

### A.65.1 Chip-specific PIT information changes

| |
|---|
| No substantial content changes. |

### A.65.2 PIT module changes

| |
|---|
| • Updated description of MCR[MDIS]. Changed the sentence “You must write 1 to this field before setting up anything” to “You must ensure that the clock is disabled (MDIS = 1) before configuring PIT. After configuration, user must enable clock (MDIS=0).”<br>• Updated Modes of operation to include Debug mode. |
| • Added STOP mode section. |
| • Rearranged feature list for clarity and added an item for the number of timers.<br>• Rewrote all register and field descriptions for clarity and usability except LTMR64L. |
| • Removed STOP mode section. |
| • Removed unnecessary "Preface" section.<br>• Rewrote the following sections for clarity and usability.<br>  — LTMR64H and LTMR64L register and field descriptions<br>  — Interrupts<br>• Renamed "Introduction" to Overview and rewrote for clarity.<br>• In Modes of operation, replaced bulleted list with a table.<br>• Added new Clocking topic.<br>• Added new External signals topic. |
| • Corrected the link present in note of register RTI_LDVAL_STAT. |
| • Moved Modes of operation under Functional description.<br>• Updated in the sub-sections of Functional description.<br>• Updated in Application information and its sub-sections. |

*Table continues on the next page...*

- Updated the access row in "PIT memory map".
- Updated register function description in PIT Module Control (MCR).
- Updated description in RTI_LDVAL_STAT[RT_STAT].
- Updated description in RTI_TFLG[TIF].
- Added text "See LTMR64H and LTMR64L." in TCTRL0[CHN]."
- Updated description in TFLG0[TIF].

- In Example configuration using chained timers:
  - Updated the example configuration to (60 s/10 ns).
  - Updated the step 6 and 7 in the configuring PIT.
  - Updated the example code for Disable PIT to "PIT_MCR = 0x02"
- Updated the example code for Disable PIT to "PIT_MCR = 0x02" in Example configuration using the lifetimer.
- Updated the example code for Disable PIT to "PIT_MCR = 0x02" in Example configuration for general timers.

- Updated the section headings of Restart counter period, Change current timer period and Dynamically setting a new timer period.
- Replaced "...when it expires (CVALn - 0)" to "...when it expires (CVALn = 0)" in Chained timers.

- In PIT register descriptions, changed PIT_RTI to PIT.
- Moved the statements about chip-specific PIT information from PIT register descriptions to Functional description and rewrote it.

- In Example configuration using chained timers, corrected "(60 s/10 ns)" to "(60 s/100 ns)".

- In Clocking, added reference of the Clocking chapter.

## A.66 Real Time Clock (RTC)

### A.66.1 Chip-specific RTC information changes

No substantial content changes.

### A.66.2 RTC module changes

- Updated last para of RTC to :In case, RTC wakeup's are used as a wakeup source, bus clock (system clock or register interface clock of RTC) should be disabled (to save power in standby mode) after enabling the required wakeup and ensuring sufficient time gap (3-6 bus clock or RTC clock (rtc_clock) cycles, whichever is slower) between bus clock disabling and wakeup event. To ensure writing the register to enable the wakeup, the bus clock must be active, hence it should not be disabled before enabling wakeup. Correct operation is not guaranteed if the specification is not followed.

- Changed "ACMP" into "CMP" in TRIG_EN.

- Modified note in CLKSEL.

- Added a note "See the chip-specific RTC information and clocking chapter for mapping of clock sources" in Features .

## A.67 Low Power Serial Peripheral Interface (LPSPI)

### A.67.1 Chip-specific LPSPI information changes

| |
|---|
| No substantial content changes. |

### A.67.2 LPSPI module changes

| |
|---|
| • In Controller mode, Transmit and command FIFO commands, and Transmit Command (TCR), clarified that LPSPI must be enabled to update command word.<br>• Updated Features. |
| • Updated Timing parameters.<br>• Added a warning on writing data to Clock Configuration (CCR). |
| • Moved "SPI bus transfers" content from Transmit and command FIFO commands to subtopic<br>• Moved "Circular FIFO" content from Transmit and command FIFO commands to subtopic |
| • Updated CR[MEN]. |
| • Updated Timing parameters.<br>• Updated CCR[SCKDIV]. |
| • Updated Features. |
| • Updated CFGR1[SAMPLE]. |
| • In LPSPI memory map, the access display changed to "See description" for some registers that have mixed access types.<br>• Updated LPSPI register descriptions description. |
| • Updated CFGR1[PARTIAL]. |
| • Updated CFGR1[MATCFG] description and notes in Data Match 0 (DMR0) and Data Match 1 (DMR1). |
| • Removed Low-power mode.<br>• Updated Interrupts and DMA requests.<br>• Removed paragraph from introduction to register descriptions, content is in Overview.<br>• Updated IER[REIE] and IER[TEIE].<br>• Updated CFGR1[PCSPOL].<br>• Updated CFGR1[SAMPLE] description.<br>• Updated RSR[SOF]. |
| • Updated Transmit Command Burst (TCBR), Transmit Data Burst (TDBR0 - TDBR127), and Receive Data Burst (RDBR0 - RDBR127).<br>• Added DMA support registers. |

*Table continues on the next page...*

| |
|---|
| • Replaced "master" and "slave" with "controller" and "peripheral" throughout chapter. |
| • Updated note on CFGR1[PINCFG]. |
| • Clarified note on CFGR1[PINCFG]. |
| • Clarified description in Timing parameters, to say that you can only change CCR when LPSPI is disabled (CR[MEN] = 0), previously it said you cannot change CCR when LPSPI is busy. |

## A.68 Low Power Inter-Integrated Circuit (LPI2C)

### A.68.1 Chip-specific LPI2C information changes

| |
|---|
| No substantial content changes. |

### A.68.2 LPI2C module changes

| |
|---|
| • Changed MCFGR0[1] to reserved. |
| • Added MCFGR0[HRPOL]. |
| • Updated Low-power modes, Interrupts and DMA requests, Controller mode, and Target mode.<br>• Added cross references to LPI2C register descriptions. |
| • Updated MCFGR1[IGNACK]. |
| • Changed title of External signals |
| • Updated Low-power modes. |
| • Changed MCFGR0[3] reserved field to RW from ROZ. |
| • Updated Low-power modes. |
| • Updated Controller Configuration 1 (MCFGR1) register. |
| • Updated access type on MCRGR0[3] in Controller Configuration 0 (MCFGR0) register. |
| • In External signals, updated I2C Four-wire serial bus figure. |
| • Updated Timing parameters. |
| • Updated MCR[RST] description. |
| • Moved Interrupts and DMA requests and Peripheral triggers. |
| • Updated Overview. |
| • Added Initialization and Application information.<br>• Updated Timing parameters. |

*Table continues on the next page...*

| |
|---|
| • Text moved from Memory map and registers to LPI2C register descriptions.<br>• In memory map table, the access changed for some registers that have mixed access types.<br>• In Version ID (VERID), removed "Read-only field." from field descriptions.<br>• In Controller Status (MSR), updated the descriptions of fields: DMF, PLTF, FEF, ALF, NDF, SDF, and EPF.<br>• Updated MCFGR1[TIMECFG].<br>• Updated Controller Configuration 2 (MCFGR2), MCFGR2[FILTSDA], and MCFGR2[FILTSCL].<br>• Updated Controller Configuration 3 (MCFGR3) description and MCFGR3[PINLOW] description.<br>• Updated MDMR[MATCH1] and MDMR[MATCH0].<br>• Updated Controller Clock Configuration 0 (MCCR0), Controller Clock Configuration 1 (MCCR1), and Controller FIFO Control (MFCR).<br>• Updated Controller FIFO Control (MFCR) description, MFCR[RXWATER], and MFCR[TXWATER].<br>• Updated MTDR[CMD] and MTDR[DATA].<br>• Updated SCR[RRF] and SCR[RTF].<br>• In Target Status (SSR), updated description in fields: FEF, BEF, SDF, RSF, and AVF.<br>• Updated Target Configuration 1 (SCFGR1) description, SCFGR1[RXCFG], and SCFGR1[TXCFG] descriptions.<br>• Updated descriptions in Target Configuration 2 (SCFGR2), Target Address Match (SAMR), and Target Address Status (SASR). |
| • Updated Error conditions. |
| • Updated Features.<br>• Updated Address matching.<br>• Updated Timing parameters.<br>• Updated Controller operations.<br>• Updated Pin configuration. |
| • Updated MCFGR0[HRPOL] description. |
| • Updated Pin configuration. |
| • Updated Controller FIFO Control (MFCR) description.<br>• Updated Controller Control (MCR) fields, RRF and RTF. |
| • Updated Target Configuration 1 (SCFGR1) references to SSR register.<br>• Updated Target DMA Enable (SDER) references to SSR register. |

## A.69 Flexible I/O (FlexIO)

### A.69.1 Chip-specific FlexIO information changes

| |
|---|
| No substantial content changes. |

### A.69.2 FLEXIO module changes

| |
|---|
| No substantial content changes. |

## A.70 CAN (FlexCAN)

### A.70.1 Chip-specific FlexCAN information changes

| |
|---|
| No substantial content changes. |

### A.70.2 FlexCAN module changes

| |
|---|
| • In Overview, removed the text "See the chip-specific FlexCAN information for the actual number of message buffers configured in the chip". |
| • In Features, Overview and , updated the ISO 11898-1 to ISO 11898-1:2015.<br>• In Protocol timing, added text "It is the smallest time unit for all configuration values". |
| • Updated the ISO 11898-1 to ISO 11898-1:2015 throughout the chapter. |
| • In Bus interface added the following registers:<br>— EPRS<br>— ENCBT<br>— EDCBT<br>— ETDC<br>• In Enhanced RX FIFO Point 5, changed EFRCR[NFE] to ERFCR[NFE]. |
| • In FlexCAN initialization sequence, added text "FlexCAN memory must be initialized before switching to functional mode". |
| • Changed the topic title for External signal descriptions from "FlexCAN signal descriptions" to "External signal descriptions". |
| • In Enhanced RX FIFO, corrected register EFRCR to ERFCR. |
| • Revised Reset to add information about hard and soft reset. |
| • Interchangeable terms "message buffer" and "mailbox" consolidated to "message buffer" throughout chapter.<br>• In Clocks, clock signal names changed from "ipg_clk*" to "MODULE_CLK*". |
| • Added note in register section.<br>• Revised topic Reset. |
| • Added "NOTE" in MCR[SOFTRST]. |
| • Updated Features.<br>• Updated Freeze mode. |

*Table continues on the next page...*

| |
|---|
| • Removed the column "Affected by hard reset" from the table #unique_2735/unique_2735_Connect_42_id-25951b. |
| • Updated "NOTE" in MCR[LPMACK] and MCR[FRZACK]. |
| • Updated long description of ERFSR[ERFCLR]. |
| • Updated the section legacy RX FIFO structure (corrected the extended ID range in RXIDB_0 of format B from "296-" to "29-16"). |
| • Updated the section Reset. |
| • Combined notes into one in Overview. |

## A.71 Synchronous Audio Interface (SAI)

### A.71.1 Chip-specific SAI information changes

| |
|---|
| No substantial content changes. |

### A.71.2 SAI module changes

| |
|---|
| • Updated TCR2[BCI] and RCR2[BCI] with note about enabling buffers. |
| • Updated field descriptions: TCSR[FR], TCSR[SR], TCSR[FWDE], TCR1[TFW], TCR5[WNW], TCR5[W0W], TCR5[FBT]. RCSR[FR], RCSR[SR], RCSR[FWDE], RCR1[RFW], RCR3[WDFL], RCR4[FRSZ], RCR4[SYWD], RCR5[WNW], RCR5[W0W], and RCR5[FBT].<br>• In Receive Configuration 5 (RCR5), updated description. |
| • Updated access type on RCR4[5]. |
| • Updated reserved fields TCSR[30] and RCSR[30].<br>• Updated reserved field RCR4[5]. |
| • Added Initialization.<br>• Moved Interrupts and DMA requests. |
| • Updated Synchronous modes. |
| • Corrected bit field descriptions for TCR2[BCP] and RCR2[BCP]. |
| • Updated TCR1[TRW]. |
| • In Block diagram, changed input clock name to MCLK.<br>• Throughout, changed audio controller clock to audio clock or audio clock (MCLK). |

## A.72 Ethernet Media Access Controller (EMAC)

### A.72.1 Chip-specific EMAC information changes

| |
|---|
| No substantial content changes. |

### A.72.2 ENET_Controller module changes

| |
|---|
| • MAC_VLAN_HASH_TABLE register: Updated description as 'The upper four bits of the calculated CRC value are used to index the contents of the VLAN hash table'<br>• MAC_VLAN_Tag.DOVLTC: Added 'The VLAN filter is bypassed when VLAN type ... VLAN type in the VLAN filter' |
| • Renamed "Signal descriptions" to External signals.<br>• Renamed "Introduction" to Overview.<br>• Renamed "Functional block diagram" to Block diagram. |
| • Corrected spelling errors |
| • Removed SGMII from figure: System-level block diagram<br>• TDES3 context descriptor: Added note 'This error is categorized as an abnormal event ...' |
| • Features: Added note: "Only byte-aligned ..." |
| • Updated registers MAC_Hash_Table_Reg0 and MAC_Hash_Table_Reg1 to add the note "In this chip, upper 6 bits should be used for 64-bit Hash Table".<br>• Deleted DMA_Safety_Interrupt_Status[DECIS] and DMA_Safety_Interrupt_Status[DEUIS] and added DMA_Safety_Interrupt_Status[1:0] as reserved. |

## A.73 Low Power Universal Asynchronous Receiver/Transmitter (LPUART)

### A.73.1 Chip-specific LPUART information changes

| |
|---|
| No substantial content changes. |

### A.73.2 LPUART module changes

| |
|---|
| • Updated the following sections: Receiver wake-up operation, and Peripheral triggers. |
| • Updated the description of Watermark (WATER).<br>• Updated the bit field descriptions of PF and TXDIR. |
| • In BAUD[OSR], added a note to read "BAUD[OSR] in this bit field results an OSR of BAUD[OSR]+1, for example, BAUD[OSR] = 0_0101b results in final divide by 6". |
| • In Receiver functional description, updated "the data character is transferred to receive FIFO, resulting the bit STAT[RDRF] to 1".<br>• In Transmitter functional description, updated "the value waiting transmit FIFO is transferred to the transmit shift register...". Replaced "STAT[TDRE]" with "transmit FIFO". |
| • Updated 0b and 1b value for STAT[RDRF].<br>• In WATER[TXWATER] and WATER[RXWATER], added "generates an interrupt". This now reads as: "Generates an interrupt, or a DMA request when the number of datawords in the transmit FIFO or buffer is equal to or less than the value of this field". |

*Table continues on the next page...*

| |
|---|
| • Updated SBR 0 in BAUD[SBR]. |
| • Updated Clocking.<br>• Added Reset.<br>• Added Initialization.<br>• Added functions in the following registers: Global (GLOBAL), Pin Configuration (PINCFG), Baud Rate (BAUD), Status (STAT).<br>• Updated BAUD[SBR]. |
| • Replaced the instance of "IrDA" with "infrared data association (IrDA)" in Overview section.<br>• Updated the following sections: Receiver functional description, Data sampling technique, Receiver wake-up operation.<br>• Replaced section title "Interrupts and status fields" with "Interrupts". |
| • Replaced reserved field access type "RW" with "ROZ" in bit 12 of Control (CTRL) register. |
| • Updated Baud rate generation section. |
| • Added Baud rate tolerance and Calculating baud rate tolerance. |
| • Replaced "receive data" with "Receive data (from RxD)" in Figure 424.<br>• Replaced 'sever to 10 bits' with 'along with N (7, 8, 9, 10) bits' in Receiver functional description.<br>• Replaced "When CTRL[LOOPS] is 1,..... (CTRL[RSRC] = 1)." with "Enable Loop mode.....CTRL[RSRC] = 0" in Loop mode.<br>• Replaced "When CTRL[LOOPS] is 1.....(CTRL[RSRC] = 1)" with "Enable single.....CTRL[RSRC] = 1" in Single-Wire mode. |
| • Replaced the word ‘DTS’ with ‘CTS’ in Hardware flow control.<br>• Changed the line ‘This is called resynchronization’ with ‘This Synchronization.....as Resynchronization’ in Data sampling technique.<br>• Updated Figure 427.<br>• Updated the long description of DATA[IDLINE]. |
| • Replaced the term ‘Internal Slave peripheral bus‘ with ‘Internal chip peripheral bus’ in Figure 423 and Figure 424 . |
| • Updated Hardware flow control. |
| • In Baud rate generation, added statement "There is a relationship between the required baud rate and the input clock frequency, SBR, and OSR. See the below figure for details". |

## A.74 Quad Serial Peripheral Interface (QuadSPI)

### A.74.1 Chip-specific QuadSPI information changes

| |
|---|
| No substantial content changes. |

### A.74.2 QuadSPI module changes

| |
|---|
| • Updated field size ofBUF3CR[ADATSZ]. |
| • Added a new sectionExtended prefetch or Streaming mode.<br>• Updated field description ofFR[ABOF].<br>• Added footnote on JMP_ON_CSand STOP instructions inTable 555.<br>• Updated field description ofBUF3CR[ADATSZ]. |
| • UpdatedExtended prefetch or Streaming mode. |
| • Removed note saying "The hardware does not provide any protection against illegal programming" from register descriptions ofBuffer 0 Top Index Register (BUF0IND),Buffer 1 Top Index Register (BUF1IND), andBuffer 2 Top Index Register (BUF2IND). |
| • Removed the RSER[TBFIE] field.<br>• Added following information in the FRAD0_WORD2[MD0ACP]: If EAL is being written in same cycle (10 or 11), or if FRAD_WORD3 [LOCK] bits are written as 11. if LOCK bits are written as 10 then this field can be programmed only if master ID matches the ID check of respective MDAD queue. If MDAD descriptor is not valid then respective MDxACP also becomes read-only.<br>• Updated the description of the FRAD0_WORD3[VLD], FRAD0_WORD3[LOCK], FRAD0_WORD5[CMP_ERR] fields.<br>• Updated the descriptions of the TG0SFARS[VLD] and TG0SFARS[CLR] fields.<br>• Updated the descriptions of the TG0IPCRS[VLD] and TG0IPCRS[CLR], and FLSEQREQ[FRAD] fields.<br>• Updated the description of the value 11 of the FRAD0_WORD3[EAL] field.<br>• Updated the description of the value 00 of the TG0MDAD[SA] field.<br>• Added caution in the sectionPower up and reset.<br>• Changes access of the following reserved fields to RW: DLCR[25], FR[30], FR[31], FR[13], FR[14], FR[4], RSER[25]<br>• Changes access of the following reserved fields to RO: FR[29], SR[25], SR[26], SR[5], SR[3] |
| • Changed the access of the following reserved fields to RW: RSER[27].<br>• Changed the access of the following reserved fields to ROZ: FR[4].<br>• Changed the access of the following reserved field to ROZ: FR[30]. |
| • Added following note inRSER[WMRK]field: This field should never be programmed above 63 because there are only 64 memory mapped RBDR registers. If watermark is programmed above 63, data above 64 words will be lost.<br>• Changed the access of the following reserved field to ROZ: BFGENCR[17]. |
| • Changed the access of the following reserved field to ROZ: BFGENCR[28]. |
| • Added "Example: Flash reset sequence" inPower up and reset.<br>• Added following fields:MCR[ISD3FA]andMCR[ISD2FA]. |
| • Replaced all instances of "master/domain" ID with domain ID. |

*Table continues on the next page...*

| |
|---|
| • Added following information in theExtended prefetch or Streaming modesection: Also ensure that complete buffer is assigned toBuffer 3 Configuration Register (BUF3CR)and remaining buffer sizes should be programmed to 0 by writing 0 toBuffer 0 Top Index Register (BUF0IND),Buffer 1 Top Index Register (BUF1IND), andBuffer 2 Top Index Register (BUF2IND)registers.<br>• InFlash memory programming, updated the second statement of the fourth step as follows: Write an appropriate value to IPCR[IDATSZ] to denote the size of the write in bytes. |
| • Added RSER[TBFIE] field. |
| • Updated Basic description section<br>• In Table 554, removed row INTB<br>• In section Dummy Pad loopback, added following notes:<br>— "Refer to Auto-DataLearning... details"<br>• "Spansion hyperflash" changed to "hyperflash"<br>• Note added, "If DLL is disabled....can be used for JMP_ON_CS instruction " for sections<br>— Fast read quad output (Winbond)<br>— Fast read sequence (Macronix/Numonyx/Spansion/Winbond)<br>—<br>— LUT<br>• Figure 430 updated |
| • Replaced 'master port number' with 'master ID'<br>• In BUF0CR[MSTRID], added a Note, "See the chip-specific... components.<br>• Updated description of IPCR[SEQID] field.<br>• Access changed to reserved for RBCT[RXBRD] bitfield 8.<br>• In BFGENCR[SEQID] Function section, removed "This field....LUT index". |
| • In Flash memory read, removed "and this depends on the configuration of RBCT[RXBRD]" |
| • Figure title changed to "Data strobe functionality in SDR mode for read operation" in Figure 436.<br>• In Figure 430, added a note, "The IOs are driven ...ongoing phase"<br>• In Figure 430, removed 2nd and 3rd note<br>• Updated Figure 430 with Note1.<br>• In section Dummy Pad loopback, added content "DLLCR[FREQEN]" and "or DLLCRB[SLV_DLY_COARSE]" for high and low frequency delay chain manual programming |
| • In Basic description, changed the variable name "tDel,total" to "ttotal_delay" |
| • In section Features, added phrase "or using AHB (64-bit width interface)"<br>• In SR[RXFULL], updated description<br>• In MCR[SWRSTSD], updated description of 0b bit<br>• In MCR[SWRSTHD], updated description of 0b bit |

*Table continues on the next page...*

| |
|---|
| • In SR[TXWA], updated Function section |
| • In RSER[CRCAIE], updated description<br>• In FR[CRCAEF], updated description |
| • In Section Dummy Pad loopback, Title updated |
| • In Programmable sequence engine, updated the Instruction set entry for CMD instruction<br>• In LUT, updated the LUT section statement for Reset Sequence Table<br>• In Flash memory programming, description updated for Flash Programming memory section<br>• In Flash memory read, for Flash memory read section, description added "A read access to... the different ways"<br>• In Byte ordering of serial flash memory read data, description added in section "Byte ordering of serial flash memory read data" |
| • In DLLCRA[FREQEN], Note removed |
| • In Sequence Pointer Clear Register (SPTRCLR), Function section updated |
| • In SR[TXWA], updated Function section |
| • Removed RSER[ABSEIE], Reserved bitfield |
| • In Address scheme, removed sections:<br>— Separation of address into rows and columns<br>— Word addressable mode for flash memory<br>• FR[CRCBEF] access updated to Reserved<br>• RSER[CRCBIE] access updated to Reserved |
| • In Flexible read xAHB buffers, content updated |
| • Removed SPTRCLR[STREAM_EN] |
| • Updated note on MSTRID to "See the chip-specific... corresponding components" |
| • Updated Note on Master IDs |
| • In RBCT[WMRK], Note added. |
| • In SOC Configuration Register (SOCCR), updated Function section |
| • Removed from feature list, "Support for HyperFlash memory" |
| • In Module Disable mode, removed text "Software needs to ensure.... is requested." |
| • Section AHB bus access read considerations updated<br>• In Flash memory programming removed text, "If required,....addressable" |
| • Removed '4h' from section QUADSPI register descriptions |

*Table continues on the next page...*

| |
|---|
| • Driving external signals<br>— Note revised, "All read commands in Dual pad, Quad pad.... applicable to Single-pad mode" |
| • TGnSFARS[ERR]<br>— Bitfield description updated |
| • Revised Pad values of "DUMMY" instruction in the following tables:<br>— Table 571<br>— Table 569<br>— Table 572 |
| • FR[ABOF]<br>— Bitfield description updated |
| • Updated bitfield description of FR[TBUF]<br>• Updated bitfield description ofFR[ABOF] |
| • Function section updated in the following registers:<br>— In LUT*n* register, added text, "Write access.... if MGC[11]=1b"<br>— In register LUT Key Register (LUTKEY), added text, "Write access.... if MGC[11]=1b"<br>— In register LUT Lock Configuration Register (LCKCR), added text, "Write access.... if MGC[11]=1b"<br>— Buffer Generic Configuration Register (BFGENCR), added text, "Write access.... if MGC[11]=1b" |
| • Revised note in Dummy Pad loopback. |
| • Revised TX buffer operation: a footnote is added to Fs |
| • Updated title of Table 556 to "Read sequence" |

## A.75 Debug changes

| |
|---|
| No substantial content changes. |

## A.76 JTAG Controller (JTAGC)

### A.76.1 Chip-specific JTAGC information changes

| |
|---|
| No substantial content changes. |

### A.76.2 JTAGC module changes

| |
|---|
| • Marked SAMPLE instruction as internal.<br>• Marked SAMPLE keyword as internal in EXTEST external test instruction, IEEE 1149.1–2001 defined test modes, Boundary scan register, and Boundary scan.<br>• Marked SAMPLE keyword as internal in PRELOAD instruction |

*Table continues on the next page...*

| |
|---|
| • Renamed "Initialization/application information" to Initialization.<br>• Reordered and reorganized various subsections in Functional description.<br>• Added Clocking, Interrupts, and Application information. |
| • Moved "IDCODE instruction", "SAMPLE/PRELOAD instruction", "EXTEST external test instruction", "HIGHZ instruction", "ENABLE_SOC_DATA1 instruction" , "CLAMP instruction", "ACCESS_AUX_x instructions" and "BYPASS instruction" into JTAGC block instructions. |
| • Moved "TEST_LEAKAGE instruction" into JTAGC block instructions section. |

## A.77 JDC module changes

| |
|---|
| • Added Clocking section.<br>• Added Interrupts section containing text "This module has no interrupts".<br>• Added JDC functionality section.<br>• Renamed "Signal description" to External signals.<br>• Renamed "Introduction" to Overview. |
| • Added Initialization section. |
| • Updated Clocking section. |

## A.78 TEMPSENSE module changes

| |
|---|
| • Renamed Introduction to Overview.<br>• Added sections Clocking, External signals, and interrupt, to convey that there are no clock sources, external signals, and interrupts for this module.<br>• Added the section head "Initialization" to Conversion from voltage to temperature. |
| • Updated Overview and Conversion from voltage to temperature |
| • Added ipg_scan_mode signal in .<br>• Improved LVIX signal description in and changed the coefficient unit from C2/V2 to C/V2 in Conversion from voltage to temperature. |

# Legal information

## Definitions

**Draft** — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

## Disclaimers

**Limited warranty and liability** — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

**Right to make changes** — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

**Suitability for use** — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

**Applications** — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

**Terms and conditions of commercial sale** — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

**Export control** — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

**Suitability for use in non-automotive qualified products** — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

**HTML publications** — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

**Translations** — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

**Security** — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

**NXP B.V.** — NXP B.V. is not an operating company and it does not distribute or sell products.

## Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

**NXP** — wordmark and logo are trademarks of NXP B.V.

**AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile** — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

**Bluetooth** — the Bluetooth wordmark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by NXP Semiconductors is under license.

**eIQ** — is a trademark of NXP B.V.

**I2C-bus** — logo is a trademark of NXP B.V.

**SafeAssure** — is a trademark of NXP B.V.

**SafeAssure** — logo is a trademark of NXP B.V.

**SuperFlash** — This product uses SuperFlash® technology. SuperFlash® is a registered trademark of Silicon Storage Technology, Inc.

**Synopsys & Designware** — are registered trademarks of Synopsys, Inc.

**Synopsys** — Portions Copyright © 2018-2022 Synopsys, Inc. Used with permission. All rights reserved.

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© 2026 NXP B.V. All rights reserved.

For more information, please visit: https://www.nxp.com

Date of release: 26 May 2026
Document identifier: MCXE31XRM