NXP Semiconductors
Reference Manual

Document identifier: LPC553xRM
Rev. 3, 07/2023

LPC553x Reference Manual

Contents

Figures

Tables

Chapter 1
About this manual

1.1 About This Document

1.1.1 Audience

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

1.1.2 Organization

This document is organized into two main sets of chapters.

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

1.1.3 Suggested Reading

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

1.1.3.1 General Information

For information about the Arm Cortex processor see:

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

1.1.3.2 Related Documentation

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

1.1.4 Conventions

This document uses the following notational conventions:

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

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

1.1.5 Register Access

1.1.5.1 Register Diagram Field Access Type Legend

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

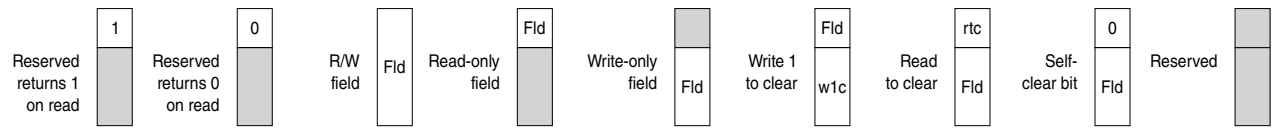


Figure 1. Register Field Conventions

NOTE

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

1.1.5.2 Register Macro Usage

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

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

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

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

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

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

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

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

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

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

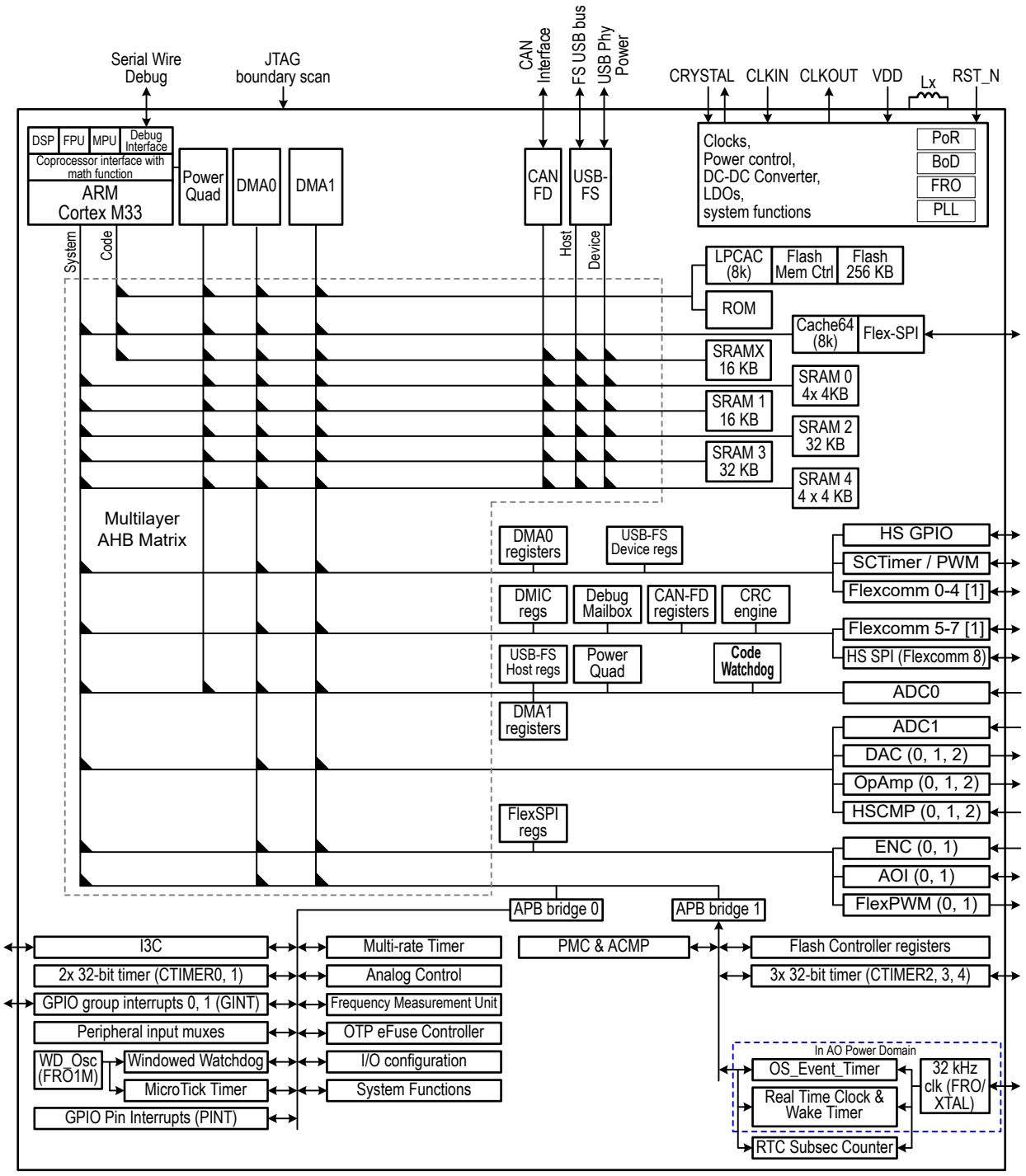
Register	Address
CCM_ANALOG_PLL_ARM	20C_8000h
CCM_ANALOG_PLL_ARM_SET	20C_8004h
CCM_ANALOG_PLL_ARM_CLR	20c_8008h
CCM_ANALOG_PLL_ARM_TOG	20C_800Ch

Chapter 2
Introduction

2.1 Overview

The LPC553x is an Arm Cortex-M33 based microcontroller for embedded applications. These devices include up to 256 KB on-chip flash, up to 128 KB of on-chip SRAM, FlexSPI with cache full-speed USB host and device interface with crystal-less operation for full-speed, CAN FD, five general-purpose timers, one SCTimer/PWM, one RTC/alarm timer, one 24-bit Multi-Rate Timer (MRT), a Windowed Watchdog Timer (WWDT), one OS Timer, one Micro-tick timer, eight flexible serial communication peripherals (which can be configured as a USART, SPI, high speed SPI, I2C, or I2S interface), one QuadFlash Filter, one DMIC, one I3C interface, two 16-bit 2.0 Msamples/sec ADCs capable of four simultaneous conversions, four comparators, two temperature sensors, three 12-bit 1 Msample/sec DACs, three OpAmps, two FlexPWM timers, and two QEIs.

2.1.1 Block Diagram



Notes: [1] : each FlexComm includes USART, SPI, and I2C. Flexcomms 6 and 7 include 4 channel-pairs of I2S, Flexcomms 0-5 each include 1 channel-pair of I2S.

Figure 2. LPC553xx Block Diagram

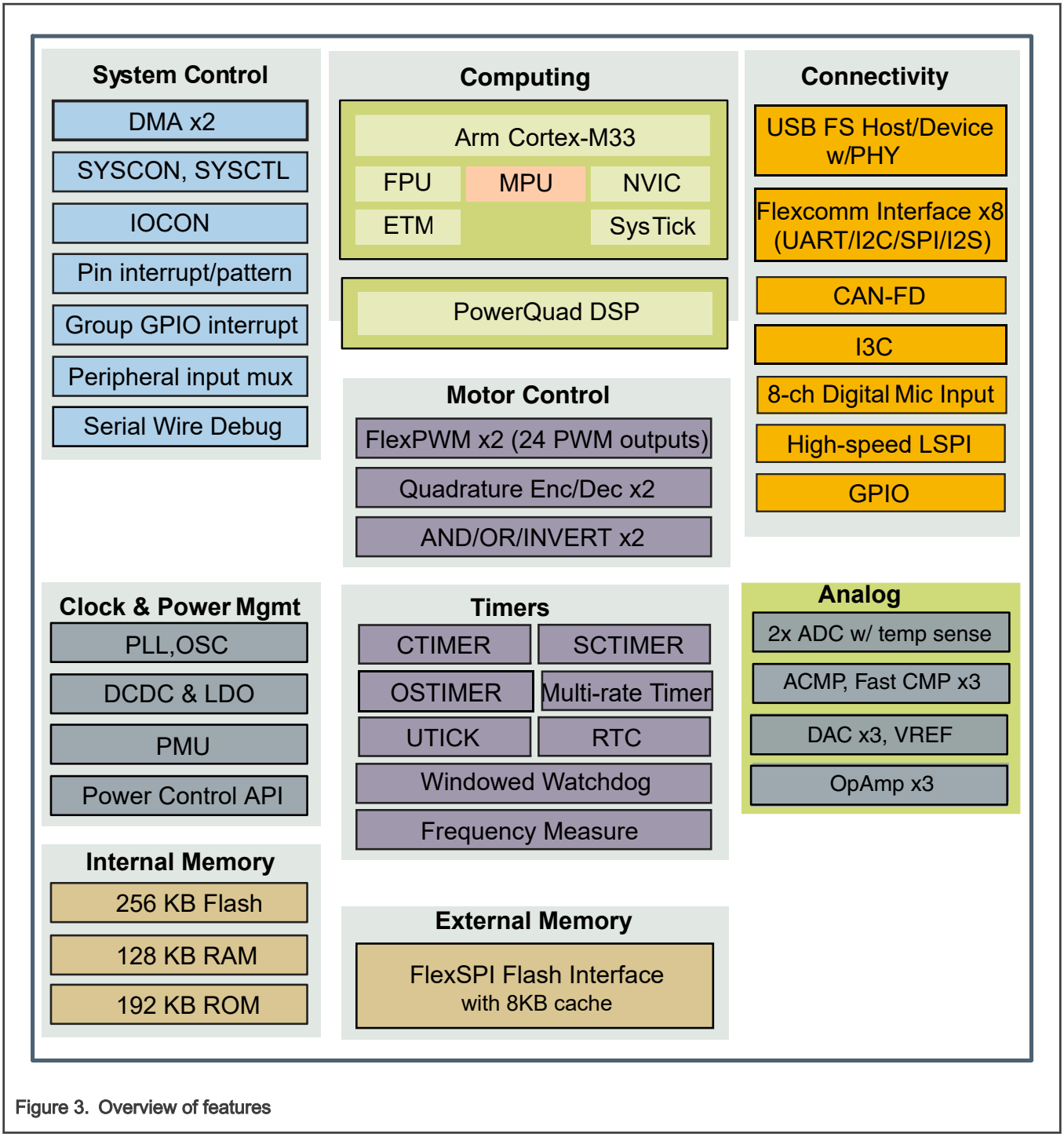


Figure 3. Overview of features

2.2 Features and benefits

- Computing:
  - Arm Cortex-M33 core (r0p4) running at a frequency of up to 150 MHz.
  - Integrated digital signal processing (DSP) instructions.
  - Floating Point Unit (FPU) and Memory Protection Unit (MPU).
  - Arm Cortex M33 built-in Nested Vectored Interrupt Controller (NVIC).

  - Non-maskable Interrupt (NMI) input with a selection of sources.
  - Serial Wire Debug with eight breakpoints and four watch points. Includes Serial Wire Output for enhanced debug capabilities and trace (ETM).
  - System tick timer.
- A hardware DSP accelerator for fixed and floating point DSP functions (PowerQuad). The PowerQuad uses a bank of four dedicated 4 KB SRAMs. .
- On-chip memory and memory interfaces:
  - Up to 256 KB on-chip flash program memory, with flash accelerator and 512 byte page erase and write.
  - Up to 128 KB total SRAM consisting of 16 KB SRAM on Code Bus, 112 KB SRAM on System Bus (112 KB is contiguous).
  - Parity support on RAM (ECC support on 16KB instance).
  - FlexSPI flash interface for external flash with 8 KB cache and dynamic decryption for execute-in-place and supports DMA. The FlexSPI includes 1 port: high speed channel A which supports quad or octal operation. Support dual image via address remapping.
  - OTP-eFuse programmable memory.
- On-chip ROM bootloader supports:
  - Booting of images from on-chip flash and external flash.
  - Flash programming through In System Programming (ISP) commands over following interfaces: USB0 interfaces using HID Class device, UART interface (Flexcomm 0) with auto baud, High-Speed SPI slave interfaces (Flexcomm 8) using mode 3 (CPOL = 1 and CPHA = 1), I2C slave interface (Flexcomm 1), and CAN-FD ISP.
  - ROM API functions: Flash programming API, Power control API, OTP-eFuse programming API, Protected Flash Region (PFR) programming.
  - Dual images (boot latest version) from on-chip flash using re-map feature.
  - Loading image to RAM from external Octal/QuadSPI device.
  - Booting Execute-in-Place (XIP) images present on Octal/QuadSPI devices.
  - Dual Execute-in-Place (XIP) images in Octal/QuadSPI flash through flash address remap feature.
  - Load-to-RAM boot mode from 1-bit SPI flash devices connected to FlexComm (selectable by PFR) as normal boot option and recovery boot option.
  - USB Device DFU Connection (Device only).
  - Code Read protection (CRP).
  - Crystal-less USB ISP mode.
- Connectivity:
  - Flexcomm Interface contains up to eight serial peripherals (Flexcomm Interface 0-7). Each Flexcomm Interface can be selected by software to be a USART, SPI, I2C, and I2S interface. Each Flexcomm Interface includes a FIFO that supports USART, SPI, and I2S. A variety of clocking options are available to each Flexcomm Interface, including a fractional baud-rate generator, and a time-out feature.Flexcomm interfaces 0 to 5 each provide one channel pair of I2S and Flexcomm interfaces 6 to 7 each provide four channel pairs of I2S.
  - I2C-bus interfaces support Fast-mode and Fast-mode Plus with data rates of up to 1Mbit/s and with multiple address recognition and monitor mode. Two sets of true I2C pads also support high-speed Mode (3.4 Mbit/s) as a slave.
  - High Speed SPI (Flexcomm 8, 50MHz for both master and slave).
  - A digital microphone interface supporting up to two channels with associated decimators and Voice Activation Detect. One pair of channels can be streamed directly to I2S. The DMIC supports DMA.
  - One I3C bus interface.

  - One CAN FD module with dedicated DMA controller.
  - USB 2.0 full speed host/device controller with on-chip PHY and dedicated DMA controller supporting crystal-less operation in device mode.
- System control and chip I/O:
  - DMA0 controller with 52 channels and up to 53 programmable triggers; DMA1 controller with 16 channels and up to 25 programmable triggers, able to access all memories and DMA-capable peripherals.
  - Up to 66 General-Purpose Input/Output (GPIO) pins.
  - GPIO registers are located on the AHB for fast access. The DMA supports GPIO ports.
  - A group of up to 8 GPIO pins can be selected for boolean pattern matching, which can generate interrupts and/or drive a pattern-match output.
  - Up to eight GPIOs can be selected as pin interrupts (PINT), triggered by rising, falling or both input edges.
  - I/O pin configuration with support for up to 16 signal options.
  - FlexSPI flash interface for external flash with 8 KB cache and dynamic decryption for execute-in-place and supports DMA. The FlexSPI includes 1 port: high speed channel A which supports quad or octal operation. Support dual image via address remapping.
  - Two AOI (AND/OR/Invert) combinatorial logic modules with dedicated set of input and output signals. Each AOI has 4 outputs that feed to different peripheral muxes to individual peripherals.
- Timers:
  - Five 32-bit standard general purpose asynchronous timers/counters, which support up to four capture inputs and four compare outputs, PWM mode, and external count input. Specific timer events can be selected to generate DMA requests.
  - One SCTimer/PWM with 8 input and 10 output functions (16 capture and match registers). Inputs and outputs can be routed to or from external pins and internally to or from selected peripherals. Internally, the SCTimer/PWM supports 16 captures/matches, 16 events, 32 states, and a Dither engine for improved average resolution of pulse edges.
  - 32-bit Real-time clock (RTC) with calendar feature and 1 s resolution running in the always-on power domain. Another timer in the RTC can be used for wake-up from all low power modes including deep power-down, with 1 ms resolution. The RTC is clocked by the 32 kHz FRO or 32.768 kHz external crystal.
  - Multiple-channel multi-rate 24-bit timer (MRT) for repetitive interrupt generation at up to four programmable, fixed rates.
  - Windowed Watchdog Timer (WWDT) with FRO 1 MHz as clock source.
  - Code Watchdog for detecting code flow integrity.
  - The Micro-Tick Timer running from the watchdog oscillator can be used to wake-up the device from sleep and deep-sleep modes. Includes 4 capture registers with pin inputs.
  - 42-bit free running OS Timer as continuous time-base for the system, available in any reduced power modes. It has a selectable clock source. When a 32 kHz clock is slelected, allows a count period of more than 4 years.
- Motor Control Subsystem: 2x FlexPWM with 4 sub-modules, providing 24 PWM outputs, it supports two 3-phase motors PWM outputs and 2 Quadrature Encoder Interface (QEI).
- Analog peripherals:
  - Two 16-bit 2.0 Msamples/sec (two 12-bit 3.3 Msamples/sec) with eight differential channel pairs, (or 16 single-ended channels), with multiple internal and external trigger inputs and sample rates of up to 2.0 MSamples/sec. The ADC supports four independent simultaneous conversions sequences.
  - Integrated temperature sensor connected to both the ADCs.
  - One comparator in always-on domain with up to four input pins and internal reference voltage. Can be used as a wake-up source from low power modes.

  - Three High Speed Comparators with up to five input pins and internal reference voltage.
  - Three 12-bit DACs with sample rates of up to 1.0 MSample/sec.
  - Three OpAmps with programmable VREF.
- Clock generation:
  - Internal Free Running Oscillator (FRO).
  - 32 kHz Internal Free Running Oscillator FRO.
  - Internal low power oscillator (FRO 1 MHz) trimmed to +/- 15% accuracy over the entire voltage and temperature range.
  - Crystal oscillator with an operating frequency of 16 MHz to 32 MHz. Option for external clock input (bypass mode) for clock frequencies of up to 25 MHz.
  - Crystal oscillator with 32.768 kHz operating frequency.
  - PLL0 and PLL1 allows CPU operation up to the maximum CPU rate without the need for a high-frequency external clock. PLL0 and PLL1 can run from the internal FRO 12 MHz output, the external oscillator, internal FRO 1 MHz output, or the 32.768 kHz RTC oscillator.
  - Clock output function with divider to monitor internal clocks.
  - Frequency measurement unit for measuring the frequency of on-chip or off-chip clock signals.
  - Each crystal oscillator has one embedded capacitor bank, where each can be used as an integrated load capacitor for the crystal oscillators. Using APIs, the capacitor banks on each crystal pin can tune the frequency for crystals with a Capacitive Load (CL) leading to conserving board space and reducing costs.
- Power-saving modes and wake-up:
  - Integrated PMU (Power Management Unit) to minimize power consumption.
  - Low power modes: Sleep, deep-sleep with RAM retention, power-down with RAM retention and CPU retention, and deep power-down with RAM retention.
  - Configurable wake-up options from peripherals interrupts.
  - The Micro-Tick Timer running from the watchdog oscillator can be used to wake-up the device from sleep and deep-sleep modes.
  - The Real-Time Clock (RTC) running from the 32.768 kHz clock can be used to wake-up the device from sleep, deep-sleep, power-down, and deep power-down modes.
  - Power-On Reset (POR).
  - Brown-Out Detectors (BOD) for external VDD_MAIN and internal VDD_CORE with separate thresholds for forced reset.
- Operating from internal DC-DC converter or selectable LDO such that DC-DC converter can be bypassed.

2.3 Target Applications

This microcontroller targets motor control applications in the appliance and industrial markets, specifically white goods and appliances, industrial and building control, and voice-operated devices.

2.4 Endianness Support

The chip supports only the Little Endian mode.

Chapter 3
PowerQuad DSP Coprocessor and Accelerator (PowerQuad)

3.1 Chip-specific PowerQuad information

Table 1. Reference links to related information

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

3.1.1 Module instances

This device has only one instance of the PowerQuad module, POWERQUAD0.

3.2 Overview

The PowerQuad is a hardware accelerator for common calculations in DSP applications. It consists of seven internal computation engines:

- Transform
- Transcendental function
- Trigonometry function
- Dual biquad infinite impulse response (IIR) filter
- Matrix accelerator
- Finite impulse response (FIR) filter
- Coordinate rotation digital computer (CORDIC)

3.2.1 Block diagram

This block diagram is a simplified representation of the PowerQuad.

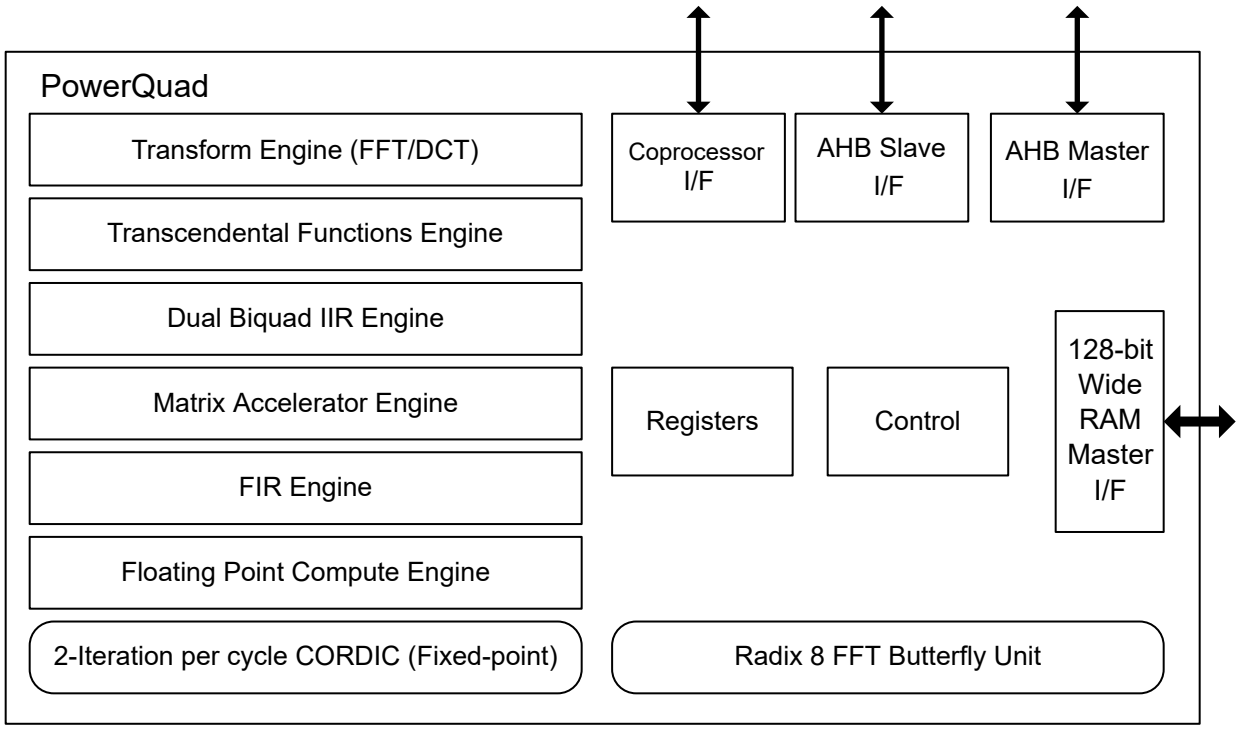


Figure 4. PowerQuad architecture

3.2.2 Features

The key features of the PowerQuad are:

- Four single-precision floating-point multiplier accumulator content (MAC) units
- AHB DMA to read/write data for input, computations, and results. The PowerQuad handles 128-bit wide RAM for input, computations, and results.
- Coprocessor interface for tightly-coupled opcodes: sin(x), cos(x), ln(x), e^(x), e^(-x), 1/(x), 1/sqrt(x), sqrt(x), biquad(x). Using two MACs, it can run two opcodes in parallel.
- FFT/iFFT/DCT/iDCT machine
- Matrix operations: add, sub, dot, prod, mult, inverse, transpose, scale
- Convolution/Correlation/FIR
- Arctan/Arctanh (can be customized to compute any CORDIC function)
- When using private RAM, it is not required to align operators. For example, for a FIR filter five taps long, the taps do not have to be placed at a location which is eight-word aligned.
- FFT engine supports pre-scaling with a positive shift value (shift left) or right shift (use same register to control).

The PowerQuad can free the Arm Cortex-M33 core perform other tasks. While the PowerQuad executes an assigned computation task, the Cortex-M33 core can prepare the next PowerQuad task, resulting in a pipeline of PowerQuad tasks.

3.3 Functional description

3.3.1 PowerQuad operation

After the clock is applied to the PowerQuad, one of the computation engines is activated based on the requested operation. An event or interrupt informs the Cortex-M33 core when the result is ready, or the Cortex-M33 core can poll the PowerQuad status.

All PowerQuad operations use one of two access types: via the coprocessor interface (CP), and via the AHB slave interface (AHB).

- When accessing the PowerQuad via the coprocessor interface, the execution of a properly formatted coprocessor instruction launches the operation, and the execution of a second coprocessor instruction retrieves the result.
- When accessing the PowerQuad via the AHB slave interface, writing to the PowerQuad launches the operation. Completion of an operation can generate an interrupt, or the PowerQuad can be polled via CONTROL[INST_BUSY] (cleared to zero). Results of the operation are then retrieved via one or more AHB read operations.

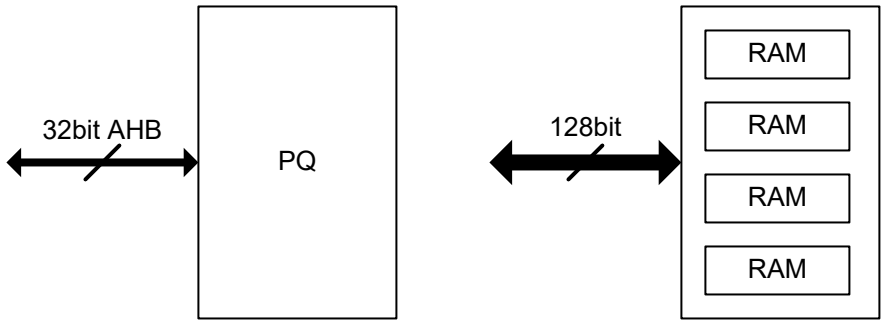


Figure 5. PowerQuad using RAM Bank 4 (16 kB) as the 128-bit wide RAM scratch pad

3.3.1.1 PowerQuad coprocessor operation

The coprocessor interface on the Arm v8-M architecture (Arm Cortex-M33 core) allows PowerQuad access via MCR (Move from Coprocessor to Register) and MRC (Move from Register to Coprocessor) opcodes. Using this method, up to two registers can be transferred between the Cortex-M33 core and the PowerQuad.

After submitting the data and/or opcodes to a coprocessor, the Cortex-M33 core can continue executing other tasks while the coprocessor computes in parallel.

In the PowerQuad, this interface is used for short (several cycles) operations for which the Cortex-M33 core provides data from one of its registers and the result is stored back in the Cortex-M33 core register.

When the data provided to the PowerQuad is in fixed-point or floating-point format, it is internally handled as floating point (conversion from fixed to floating point is done automatically if needed).

The result of PowerQuad operations can be fixed or floating point (converting from floating to fixed point, if needed).

The functions handled in this way are: sin(x), cos(x), ln(x), e^(x), e^(-x), 1/(x), 1/sqrt(x), sqrt(x), biquad(x).

3.3.1.2 PowerQuad AHB operation

Large data operations are managed in a memory-mapped fashion. The Cortex-M33 core writes to the PowerQuad Control registers to select the function, to provide four memory regions with source and/or destination addresses, and to provide a temporary scratch address for intermediate results (where applicable, such as FFT and Matrix Inversion). These regions are: Input A, Input B, Output, and Temp.

Table 2. Register overview

Address	Name	Description	Access	Comments
0x000	OUTBASE	Base address register for Output region	RW	Input A, B and Output region settings

Table continues on the next page...

Table 2. Register overview (continued)

Address	Name	Description	Access	Comments
0x004	OUTFORMAT	Data format for Output region	RW	
0x008	TMPBASE	Base address register for Temporary region	RW	
0x00C	TMPFORMAT	Data format for Temporary region	RW	
0x010	INABASE	Base address register for Input A region	RW	
0x014	INAFORMAT	Data format for Input A region	RW	
0x018	INBBASE	Base address register for Input B region	RW	
0x01C	INBFORMAT	Data format for Input B region	RW	
0x100	CONTROL	PowerQuad control register	RW	Command register
0x104	LENGTH	Length register	RW	Maximum row/column length is 16 by 16 for matrix operations. For vector operations, the maximum size is 256 (vector operations have only one dimension for length).
0x108	CPPRE	Coprocessor pre-scale register	RW	0
0x10C	MISC	Miscellaneous use register	RW	0
0x110	CURSORY	Cursory register	RW	CORDIC engine input and output registers
0x180	CORDIC_X	CORDIC input X register	RW	
0x184	CORDIC_Y	CORDIC input Y register	RW	
0x188	CORDIC_Z	CORDIC input Z register	RW	
0x18C	ERRSTAT	Read/write register where error statuses are captured (sticky)	RW1C	Interrupt and status flags
0x190	INTREN	Determines which conditions will assert the interrupt output	RW	
0x194	EVENTEN	Determines which conditions will assert the event trigger output	RW	
0x198	INTRSTAT	Interrupt status register	RW1C	
0x200 - 0x23C	GPREGS[16]	General purpose register bank (16 x 32 bits)	RW	Computing registers

Table continues on the next page...

Table 2. Register overview (continued)

Address	Name	Description	Access	Comments
0x240 - 0x25C	COMPREGS[8]	Compute register bank (8 x 32 bits)	RW	

Completion of an operation can generate an interrupt, or the PowerQuad can be polled via CONTROL[INST_BUSY] (cleared to zero).

The functions handled in this way are:

- Matrix (add, sub, scale, invert, transpose, multiply, dot, product)
- FFT/IFFT/DCT/IDCT (note that DCT is partial in the PowerQuad and requires some adjustments)
- FIR/convolution/correlation

3.3.2 Using the PowerQuad with the Arm Cortex-M33 core

In a conventional MCU without PowerQuad, the main CPU performs the computation work. When a more critical task needs to be serviced, the processor has to save registers in the stack, service the task, then retrieve registers from the stack and continue computation work. Using the PowerQuad overcomes this single-task flow.

While the conventional approach is to program compute tasks sequentially and anticipate conditional branching, you are encouraged to treat the PowerQuad as a coprocessor, performing multiple tasks by rearranging them and reducing the branching to speed up computational processing. The PowerQuad engines compute the assigned functions faster than using the Cortex-M33 core alone.

3.3.3 Clocking

Enable the clock to PowerQuad through an AHB clock control register.

3.3.4 Interrupts

This module has no interrupts.

3.4 External signals

This module has no external signals.

3.5 Initialization

To Initialize PowerQuad, enable the clock to PowerQuad through an AHB clock control register.

3.6 Application information

3.6.1 PowerQuad API functions

Driver APIs remove the need for you to write code to implement PowerQuad functions.

A comprehensive list of PowerQuad driver functions is available for download in the SDK at www.nxp.com. PowerQuad application notes give example code for various subsystems that show how PowerQuad driver functions do the work.

PowerQuad operations are grouped into four families: Math, Filtering, Matrix, and Transform. The operations implemented in each family are listed in Table 3.

The driver function table uses the following terms:

- CP = coprocessor
- Xform = Transform

- Tran = Transcendental
- Trig = Trigonometry
- IIR = Infinite Impulse Response
- FIR = Finite Impulse Response
- FP = 32-bit single-precision floating point
- N.A.=Not Applicable
- Fix-16 = 16-bit fixed point
- Fix-32 = 32-bit Fixed Point
- Q1.31 = signed 32-bit fixed point with 31 bits to the right of the binary point

Table 3. Summary of PowerQuad driver functions

Operation	Driver function	Access type	Input/Output data formats	InputA region usage	InputB region usage	Output region usage	Fixed point input/output scalers	Engine	Uses GPREGs/ COMPREGs
1/x	PQ_InvFixed PQ_InvF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
sqrt(x)	PQ_SqrtFixed PQ_SqrtF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
1/sqrt(x)	PQ_InvSqrtFixed PQ_InvSqrtF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
ln(x)	PQ_LnFixed PQ_LnF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
e^(x)	PQ_EtoxFixed PQ_EtoxF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
e^(-x)	PQ_EtonxFixed PQ_EtonxF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
x1 / x2	PQ_DivF32	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	cppre_in cppre_out	Tran	No
sin(x)	PQ_SinFixed PQ_SinF32	CP	FP, Q1.31 (in radians, normalized)	N.A.	N.A.	N.A.	cppre_in cppre_out	Trig	No
cos(x)	PQ_CosFixed PQ_CosF32	CP	FP, Q1.31 (in radians, normalized)	N.A.	N.A.	N.A.	cppre_in cppre_out	Trig	No
IIR Filter (x)	PQ_*Biquad*	CP	FP, Fix-16, Fix-32	N.A.	N.A.	N.A.	cppre_in cppre_out	Biquad	Yes
arctan(x)	PQ_ArctanFixed	AHB	Fix-16, Fix-32	CORDIC_X	CORDIC_Y	CORDIC_Z	N.A.	CORDIC	No
arctanh(x)	PQ_ArctanhFixed	AHB	Fix-16, Fix-32	CORDIC_X	CORDIC_Y	CORDIC_Z	N.A.	CORDIC	No
FIR Filter	PQ_FIR	AHB	FP, Fix-16, Fix-32	Input data	Filter coefficients	Output data	Ina_scaler Inb_scaler Out_scaler	FIR	No

Table continues on the next page...

Table 3. Summary of PowerQuad driver functions (continued)

Operation	Driver function	Access type	Input/Output data formats	InputA region usage	InputB region usage	Output region usage	Fixed point input/output scalers	Engine	Uses GPREGs/ COMPREGs
FIR Filter Incremental	PQ_FIR	AHB	FP, Fix-16, Fix-32	Input data	Filter coefficients	Output data	Ina_scaler Inb_scaler Out_scaler	FIR	No
Convolution	PQ_FIR	AHB	FP, Fix-16, Fix-32	Input data	Filter coefficients	Output data	Ina_scaler Inb_scaler Out_scaler	FIR	No
Correlation	PQ_FIR	AHB	FP, Fix-16, Fix-32	Input data	Filter coefficients	Output data	Ina_scaler Inb_scaler Out_scaler	FIR	No
Matrix Addition	PQ_MatrixAddition	AHB	FP, Fix-16, Fix-32	Matrix M1	Matrix M2	Result Matrix	Ina_scaler Inb_scaler Out_scaler	Matrix	No
Matrix Subtraction	PQ_MatrixSubtraction	AHB	FP, Fix-16, Fix-32	Matrix M1	Matrix M2	Result matrix	Ina_scaler Inb_scaler Out_scaler	Matrix	No
Matrix Hadamard Product	PQ_MatrixMultiplication	AHB	FP, Fix-16, Fix-32	Matrix M1	Matrix M2	Result matrix	Ina_scaler Inb_scaler Out_scaler	Matrix	No
Matrix Product	PQ_MatrixProduct	AHB	FP, Fix-16, Fix-32	Matrix M1	Matrix M2	Result matrix	Ina_scaler Inb_scaler Out_scaler	Matrix	No
Matrix Invert	PQ_MatrixInversion	AHB	FP, Fix-16, Fix-32	Matrix M1	N.A.	Result matrix	Ina_scaler Inb_scaler Out_scaler	Matrix	No
Matrix Transpose	PQ_MatrixTranspose	AHB	FP, Fix-16, Fix-32	Matrix M1	N.A.	Result matrix	Ina_scaler Inb_scaler Out_scaler	Matrix	No
Matrix Scale	PQ_MatrixScale	AHB	FP, Fix-16, Fix-32	Matrix M1	N.A. (Scale factor in	Result matrix	Ina_scaler Inb_scaler Out_scaler	Matrix	No

Table continues on the next page...

Table 3. Summary of PowerQuad driver functions (continued)

Operation	Driver function	Access type	Input/Output data formats	InputA region usage	InputB region usage	Output region usage	Fixed point input/output scalers	Engine	Uses GPREGs/ COMPREGs
					MISC register)				
Vector Dot Product	PQ_VectorDotProduct	AHB	FP, Fix-16, Fix-32	Vector A	Vector B	Scaler result	Ina_scaler Inb_scaler Out_scaler	Matrix	No
FFT of complex-valued input sequence	PQ_TransformCFFT	AHB	Fix-16, Fix-32	Input data	N.A.	Output data	Ina_scaler Inb_scaler Out_scaler	Xform	Yes
FFT of real-valued input sequence	PQ_TransformRFFT	AHB	Fix-16, Fix-32	Input data	N.A.	Output data	Ina_scaler Inb_scaler Out_scaler	Xform	Yes
Inverse FFT	PQ_TransformIFFT	AHB	Fix-16, Fix-32	Input data	N.A.	Output data	Ina_scaler Inb_scaler Out_scaler	Xform	Yes
DCT of complex-valued input sequence	PQ_TransformCDCT	AHB	Fix-16, Fix-32	Input data	N.A.	Output data	Ina_scaler Inb_scaler Out_scaler	Xform	Yes
DCT of real-valued input sequence	PQ_TransformRDCT	AHB	Fix-16, Fix-32	Input data	N.A.	Output data	Ina_scaler Inb_scaler Out_scaler	Xform	Yes
Inverse DCT	PQ_TransformIDCT	AHB	Fix-16, Fix-32	Input data	N.A.	Output data	Ina_scaler Inb_scaler Out_scaler	Xform	Yes

3.7 Memory map and register definition

This section includes the PowerQuad module memory map and detailed descriptions of all registers.

3.7.1 PowerQuad register descriptions

3.7.1.1 PowerQuad memory map

POWERQUAD0 base address: 400A_6000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Output Base (OUTBASE)	32	RW	0000_0000
4	Output Format (OUTFORMAT)	32	See section	0000_0000
8	Temporary Base (TMPBASE)	32	RW	0000_0000
C	Temporary Format (TMPFORMAT)	32	See section	0000_0000
10	Input A base (INABASE)	32	RW	0000_0000
14	Input A format (INAFORMAT)	32	See section	0000_0000
18	Input B base (INBBASE)	32	RW	0000_0000
1C	Input B format (INBFORMAT)	32	See section	0000_0000
100	Control (CONTROL)	32	See section	0000_0000
104	Length (LENGTH)	32	RW	0000_0000
108	Coprocessor Pre-scale (CPPRE)	32	See section	0000_0000
10C	Miscellaneous (MISC)	32	RW	0000_0000
110	Cursory (CURSORY)	32	See section	0000_0000
180	CORDIC input X (CORDIC_X)	32	RW	0000_0000
184	CORDIC input Y (CORDIC_Y)	32	RW	0000_0000
188	CORDIC input Z (CORDIC_Z)	32	RW	0000_0000
18C	Error Status (ERRSTAT)	32	See section	0000_0000
190	Interrupt Enable (INTREN)	32	See section	0000_0000
194	Event Enable (EVENTEN)	32	See section	0000_0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
198	Interrupt Status (INTRSTAT)	32	See section	0000_0000
200 - 23C	General Purpose Register Bank n (GPREG0 - GPREG15)	32	RW	0000_0000
240 - 25C	Compute Register Bank n (COMPREG0 - COMPREG7)	32	RW	0000_0000

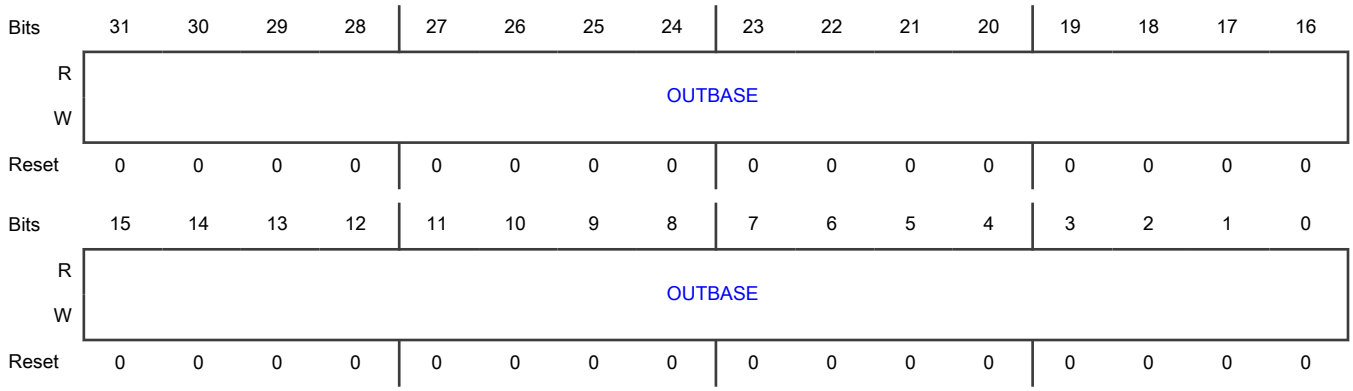
3.7.1.1.1 Output Base (OUTBASE)

Base address register for output region

Offset

Register	Offset
OUTBASE	0h

Diagram



Fields

Field	Description
31-0 OUTBASE	Base address for the output region

3.7.1.1.2 Output Format (OUTFORMAT)

Data format for output region

Offset

Register	Offset
OUTFORMAT	4h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	OUT_SCALER								Reserved		OUT_FORMAT EXT		Reserved		OUT_FORMATI NT	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-16 —	Reserved
15-8 OUT_SCALER	Output scaler value (for scaled 'q31' formats)
7-6 —	Reserved
5-4 OUT_FORMAT EXT	Output external format 00 - q15 01 - q31 10 - float
3-2 —	Reserved
1-0 OUT_FORMATI NT	Output internal format 00 - q15 01 - q31 10 - float

3.7.1.1.3 Temporary Base (TMPBASE)

Base address register for temporary region

Offset

Register	Offset
TMPBASE	8h

Diagram

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

Fields

Field	Description
31-0 TMPBASE	Base address for the temporary region

3.7.1.1.4 Temporary Format (TMPFORMAT)

Data format for temporary region

Offset

Register	Offset
TMPFORMAT	Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	TMP_SCALER								Reserved		TMP_FORMAT EXT		Reserved		TMP_FORMATI NT	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-16 —	Reserved
15-8 TMP_SCALER	Temporary scaler value (for scaled 'q31' formats)
7-6 —	Reserved
5-4 TMP_FORMATEXT	Temporary external format 00 - q15 01 - q31 10 - float
3-2 —	Reserved
1-0 TMP_FORMATINT	Temporary internal format 00 - q15 01 - q31 10 - float

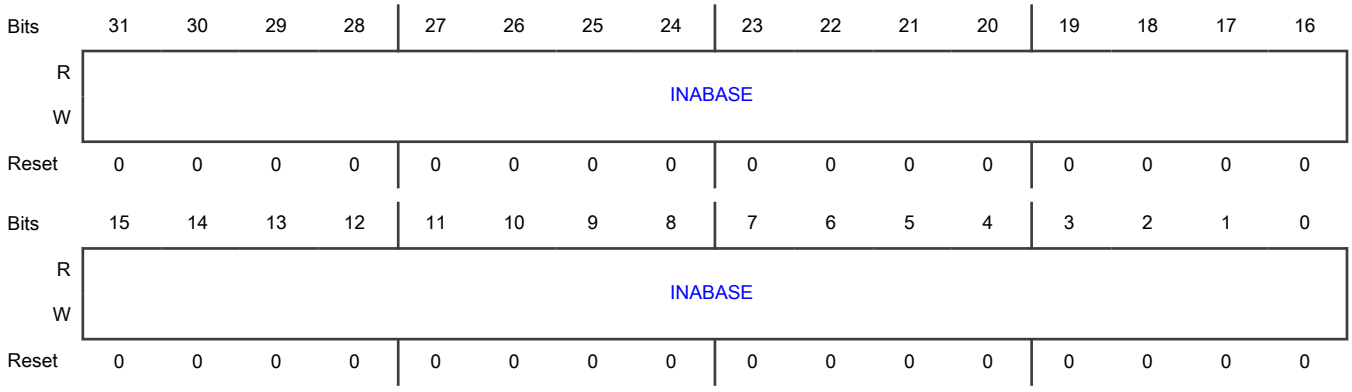
3.7.1.1.5 Input A base (INABASE)

Base address register for input A region

Offset

Register	Offset
INABASE	10h

Diagram



Fields

Field	Description
31-0 INABASE	Input A base Base address for the input A region

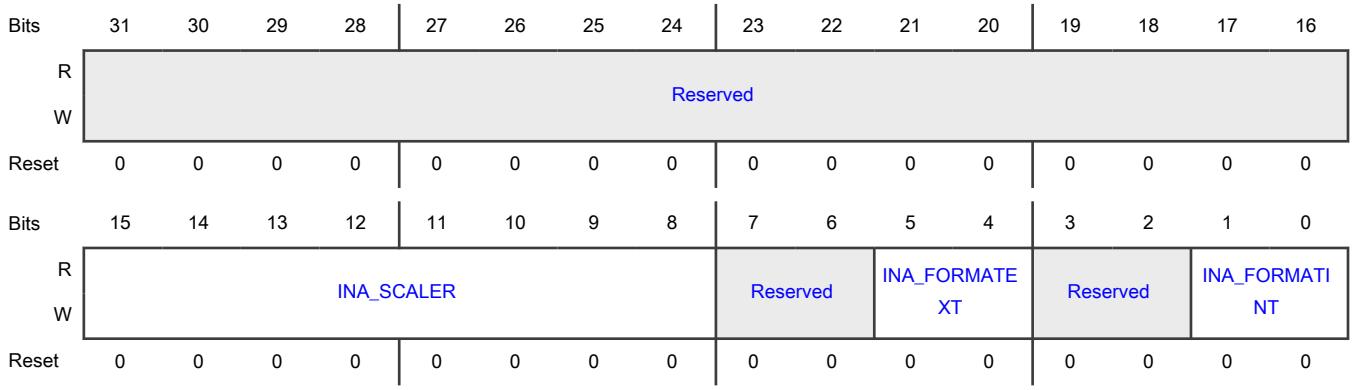
3.7.1.1.6 Input A format (INAFORMAT)

Data format for region input A

Offset

Register	Offset
INAFORMAT	14h

Diagram



Fields

Field	Description
31-16 —	Reserved
15-8 INA_SCALER	Input A scaler value (for scaled 'q31' formats)
7-6 —	Reserved
5-4 INA_FORMATEXT	Input A external format 00 - q15 01 - q31

Table continues on the next page...

Table continued from the previous page...

Field	Description
	10 - float
3-2 —	Reserved
1-0 INA_FORMATINT	Input A internal format 00 - q15 01 - q31 10 - float

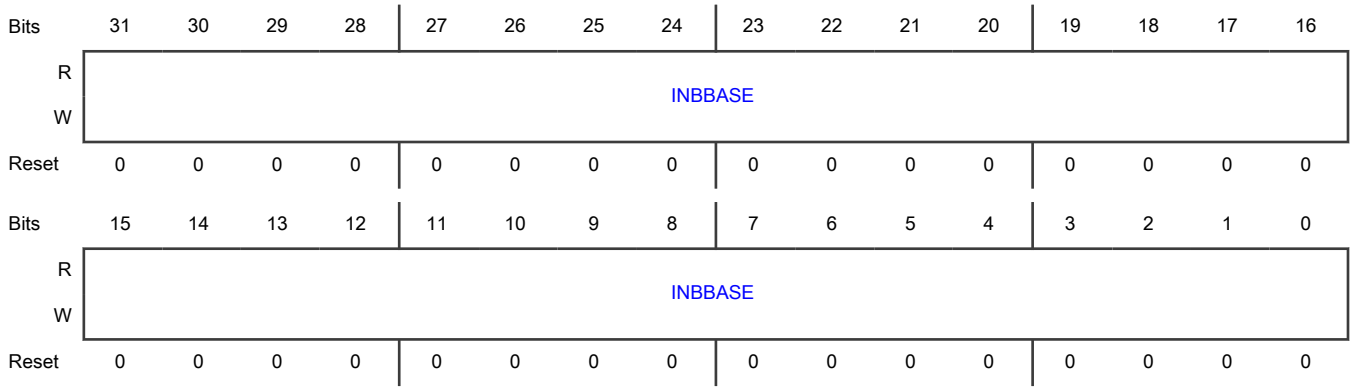
3.7.1.1.7 Input B base (INBBASE)

Base address register for input B region

Offset

Register	Offset
INBBASE	18h

Diagram



Fields

Field	Description
31-0 INBBASE	Input B base Base address for the input B region

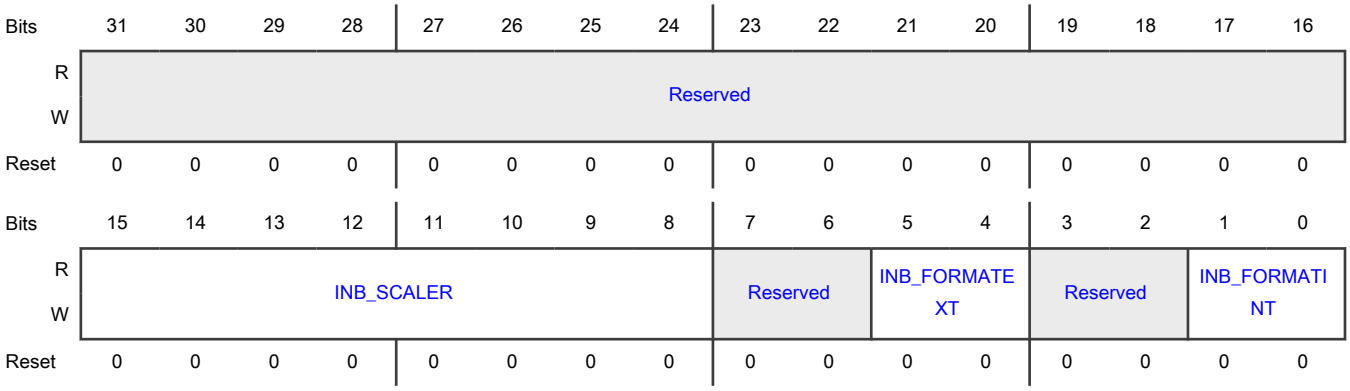
3.7.1.1.8 Input B format (INBFORMAT)

Data format for region input B

Offset

Register	Offset
INBFORMAT	1Ch

Diagram



Fields

Field	Description
31-16 —	Reserved
15-8 INB_SCALER	Input B scaler value (for scaled 'q31' formats)
7-6 —	Reserved
5-4 INB_FORMATEXT	Input B external format 00 - q15 01 - q31 10 - float
3-2 —	Reserved
1-0 INB_FORMATINT	Input B internal format 00 - q15 01 - q31 10 - float

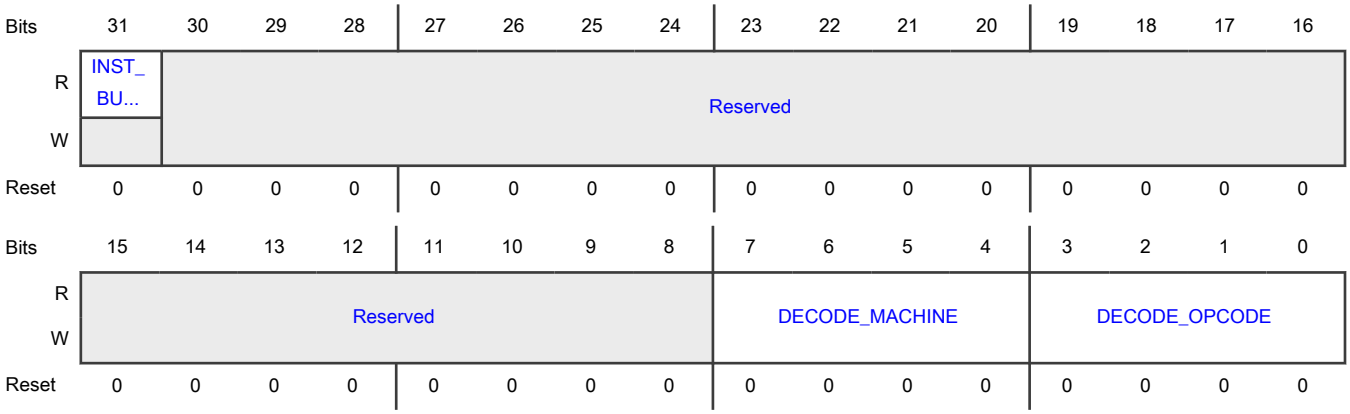
3.7.1.1.9 Control (CONTROL)

PowerQuad control register

Offset

Register	Offset
CONTROL	100h

Diagram



Fields

Field	Description
31 INST_BUSY	Instruction busy 0 - Not busy 1 - busy
30-8 —	Reserved
7-4 DECODE_MACHINE	Decode machine 0 : Coprocessor , 1 : matrix , 2 : fft , 3 : fir , 4 : stat , 5 : cordic , 6 -15 : NA
3-0 DECODE_OPCODE	Decode opcode Opcode specific to decode_machine

3.7.1.1.10 Length (LENGTH)

Length register

Offset

Register	Offset
LENGTH	104h

Diagram

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

Fields

Field	Description
31-0 INST_LENGTH	Instruction length When FIR : fir_xlength = inst_length[15:0] , fir_tlength = inst_len[31:16]. When MTX : rows_a = inst_length[4:0] , cols_a = inst_length[12:8] , cols_b = inst_length[20:16] **NOTE** Maximum row/column is 16 by 16, regardless of 5-bit allowance. For vector operations, the maximum size is 256 (vector operations have only one dimension for length).

3.7.1.1.11 Coprocessor Pre-scale (CPPRE)

Pre-scale register

Offset

Register	Offset
CPPRE	108h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved														CPPRE_S...	CPPRE_S...
Reset	0	0	0	0	0	0	0	0	0	0	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	CPPRE_OUT								CPPRE_IN							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-18 —	Reserved
17 CPPRE_SAT8	Saturation 8 0 - 8 bits 1 - 16 bits
16 CPPRE_SAT	Saturation Sub-32 bit saturation 0 - No saturation 1 - Forces sub-32 bit saturation
15-8 CPPRE_OUT	Output Coprocessor fixed-point output
7-0 CPPRE_IN	Input Coprocessor scaling of input

3.7.1.1.12 Miscellaneous (MISC)

Used for the scaling factor for matrix operations

Offset

Register	Offset
MISC	10Ch

Diagram

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

Fields

Field	Description
31-0 INST_MISC	For matrix operations used for scaling factor

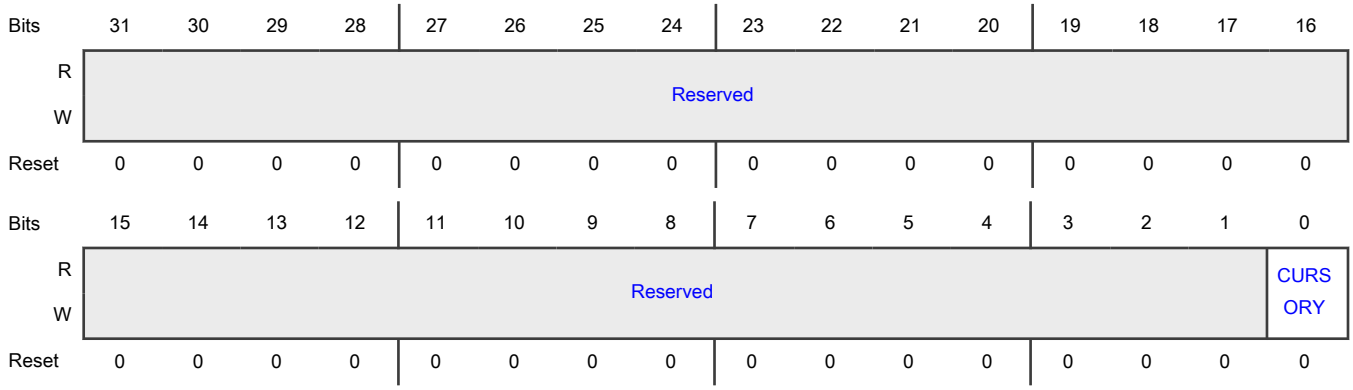
3.7.1.1.13 Cursory (CURSORY)

Cursory register

Offset

Register	Offset
CURSORY	110h

Diagram



Fields

Field	Description
31-1 —	Reserved
0 CURSORY	Cursory Mode 0 - Disable cursory mode 1 - Enable cursory Mode

3.7.1.1.14 CORDIC input X (CORDIC_X)

Offset

Register	Offset
CORDIC_X	180h

Diagram

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

Fields

Field	Description
31-0 CORDIC_X	CORDIC input x

3.7.1.1.15 CORDIC input Y (CORDIC_Y)

Offset

Register	Offset
CORDIC_Y	184h

Diagram

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

Fields

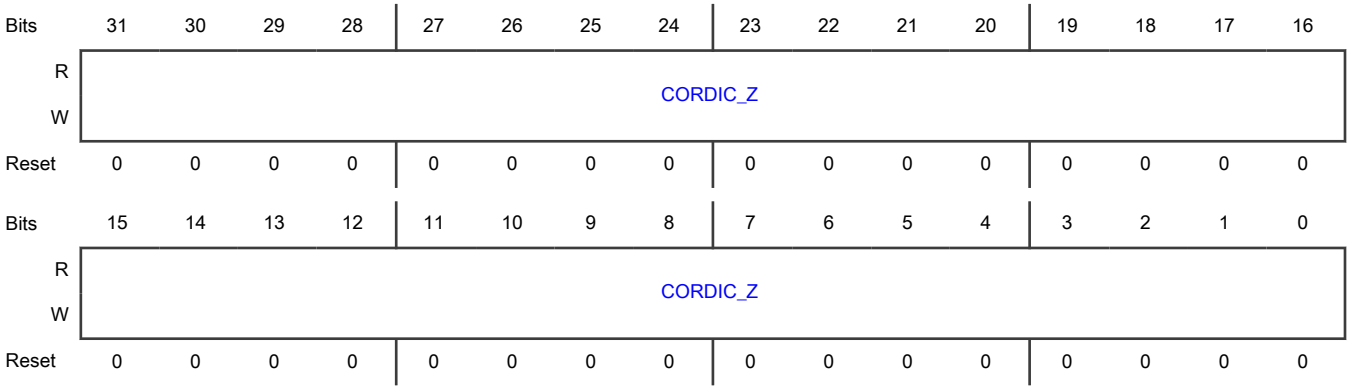
Field	Description
31-0 CORDIC_Y	CORDIC input y

3.7.1.1.16 CORDIC input Z (CORDIC_Z)

Offset

Register	Offset
CORDIC_Z	188h

Diagram



Fields

Field	Description
31-0 CORDIC_Z	CORDIC input z

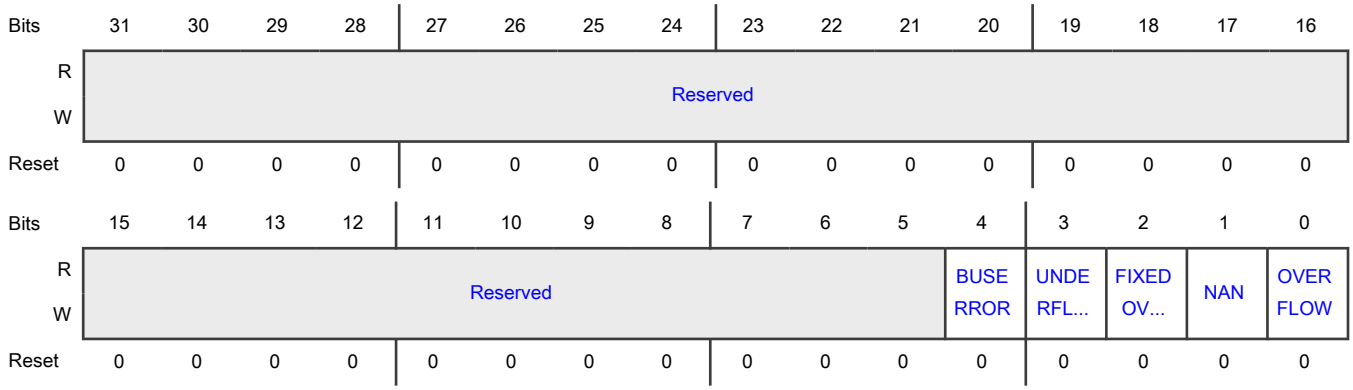
3.7.1.1.17 Error Status (ERRSTAT)

Read/Write register where error statuses are captured (sticky)

Offset

Register	Offset
ERRSTAT	18Ch

Diagram



Fields

Field	Description
31-5 —	Reserved
4 BUSERROR	Bus error 0 - No error 1 - Error on bus
3 UNDERFLOW	Underflow 0 - No error 1 - Error on underflow
2 FIXEDOVERFLOW	Fixed point overflow 0 - No error 1 - Error on fixed point overflow
1 NAN	Floating Point NaN 0 - No error 1 - Error on Floating Point NaN
0 OVERFLOW	Floating point overflow 0 - No error 1 - Error on floating point overflow

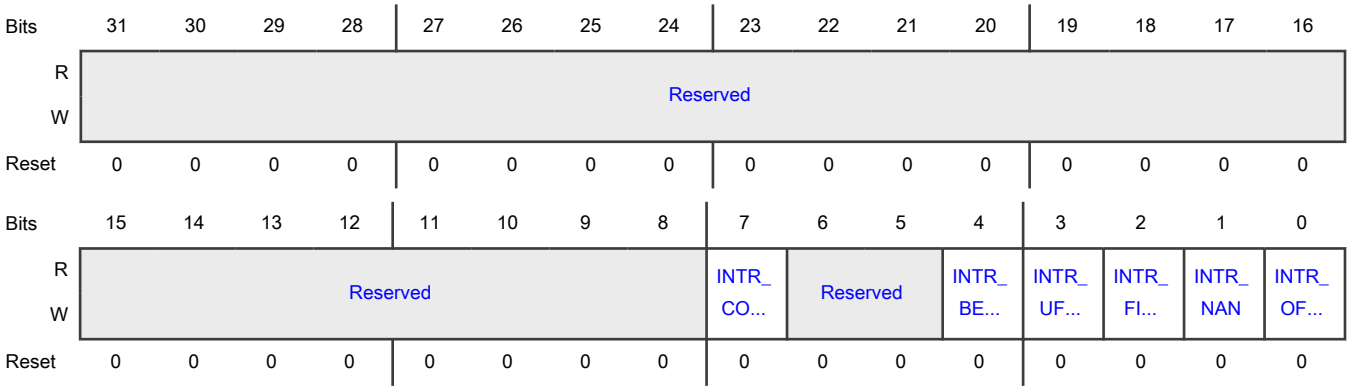
3.7.1.1.18 Interrupt Enable (INTREN)

Determines which conditions assert the interrupt output

Offset

Register	Offset
INTREN	190h

Diagram



Fields

Field	Description
31-8 —	Reserved
7 INTR_COMP	Interrupt on instruction completion 0 - Disable interrupt 1 - Enable interrupt
6-5 —	Reserved
4 INTR_BERR	Interrupt on AHBM bus error 0 - Disable interrupt 1 - Enable interrupt
3 INTR_UFLOW	Interrupt on subnormal truncation 0 - Disable interrupt 1 - Enable interrupt
2 INTR_FIXED	Interrupt on fixed-point overflow 0 - Disable interrupt 1 - Enable interrupt
1	Interrupt floating point NaN 0 - Disable interrupt

Table continues on the next page...

Table continued from the previous page...

Field	Description
INTR_NAN	1 - Enable interrupt
0 INTR_OFLOW	Interrupt floating point overflow 0 - Disable interrupt 1 - Enable interrupt

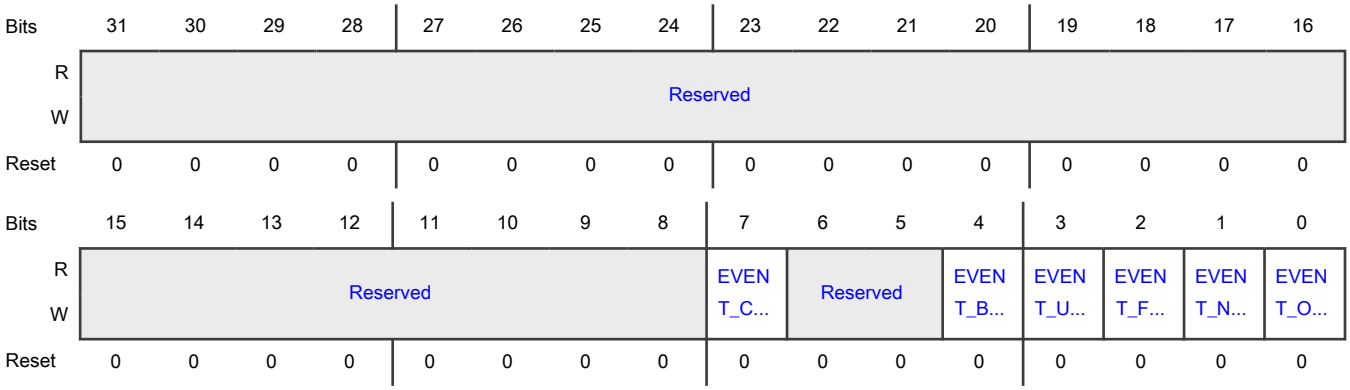
3.7.1.1.19 Event Enable (EVENTEN)

Determines which conditions assert the event trigger output.

Offset

Register	Offset
EVENTEN	194h

Diagram



Fields

Field	Description
31-8 —	Reserved
7 EVENT_COMP	Event trigger on instruction completion 0 - Disable event trigger 1 - Enable event trigger
6-5 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
4 EVENT_BERR	Event trigger on AHBM bus error 0 - Disable event trigger 1 - Enable event trigger
3 EVENT_UFLOW	Event trigger on subnormal truncation 0 - Disable event trigger 1 - Enable event trigger
2 EVENT_FIXED	Event trigger on fixed-point overflow 0 - Disable event trigger 1 - Enable event trigger
1 EVENT_NAN	Event trigger on floating point NaN 0 - Disable event trigger 1 - Enable event trigger
0 EVENT_OFLOW	Event trigger on floating point overflow 0 - Disable event trigger 1 - Enable event trigger

3.7.1.1.20 Interrupt Status (INTRSTAT)

A write of any value clears the Interrupt Status bit.

Offset

Register	Offset
INTRSTAT	198h

Diagram

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

Fields

Field	Description
31-1 —	Reserved
0 INTR_STAT	Interrupt Status 0 - No new interrupt 1 - Interrupt captured

3.7.1.1.21 General Purpose Register Bank n (GPREG0 - GPREG15)

The General Purpose Register Bank is n x 32 bits.

Offset

For n = 0 to 15:

Register	Offset
GPREGn	200h + (n × 4h)

Diagram

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

Fields

Field	Description
31-0 GPREG	General Purpose Bank

3.7.1.1.22 Compute Register Bank n (COMPREG0 - COMPREG7)

The Compute Register Bank is n x 32 bits.

Offset

For n = 0 to 7:

Register	Offset
COMPREGn	240h + (n × 4h)

Diagram

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

Fields

Field	Description
31-0 COMPREG	Compute bank

Chapter 4
Memory Maps

4.1 Memory system overview

This section introduces the memory architecture of the chip.

4.2 System memory map

Table 4. CPU0 memory map

Start address	Default [1]	Description	Alias	Size	Access	Cached
0000_0000	Non-Secure	Flash memory on CM33 code bus on, AHB slave port 0.		256 KB	Core, DMA, selected masters	Yes
0004_0000		Reserved	—	—	—	—
0300_0000	Non-Secure	Boot ROM, on CM33 code bus, AHB slave port 0.		192 KB	Core, DMA, selected masters	No
0303_0000		Reserved	—	—	—	—
0400_0000	Non-Secure	SRAM X on CM33 code bus, on AHB slave port 1.		16 KB	Core, DMA, selected masters	No
0400_4000		Reserved	—	—	—	—
0800_0000	Non-Secure	External Quad/Octal Flash (FlexSPI) on AHB slave port 2.		128 MB	Core, DMA, selected masters	Yes
1000_0000		Reserved	—	—	—	—
2000_0000	Non-Secure	SRAM 0 on CM33 data bus, on AHB slave port 3. First 4 KB: Retention-braindump. Second 4KB: Stack. Upper 8 KB: shared with PKC.		16 KB	Core, DMA, selected masters	No
2000_4000		SRAM 1 on CM33 data bus, on AHB slave port 4.		16 KB	Core, DMA, selected masters	No
2000_8000		SRAM 2 on CM33 data bus, on AHB slave port 5.		32 KB	Core, DMA, selected masters	No
2001_0000		SRAM 3 on CM33 data bus, on AHB slave port 6.		32 KB	Core, DMA, selected masters	No
2001_8000		SRAM 4 on CM33 data bus, on AHB slave port 7.		16 KB	Core, DMA, selected masters	No

Table continues on the next page...

Table 4. CPU0 memory map (continued)

Start address	Default [1]	Description	Alias	Size	Access	Cached
2001_C000		Reserved	—	—	—	—
4000_0000	Non-Secure	AHB to APB bridge 0 on slave port 8. See APB peripheral table.		128 KB	Core, DMA, selected masters	No
4002_0000		AHB to APB bridge 1 on slave port 8. See APB peripheral table.		128 KB	Core, DMA, selected masters	No
4004_0000		Reserved	—	—	—	—
4008_0000	Non-Secure	AHB peripherals on slave port 9. See AHB peripheral table.		64 KB	Core, DMA, selected masters	No
4009_0000		AHB peripherals on slave port 10. See AHB peripheral table.		64 KB	Core, DMA, selected masters	No
400A_0000		AHB peripherals on slave port 11. See AHB peripheral table.		64 KB	Core, DMA, selected masters	No
400B_0000		AHB peripherals on slave port 12. See AHB peripheral table.		64 KB	Core, DMA, selected masters	No
400C_0000		AHB peripherals on slave port 13. See AHB peripheral table.		64 KB	Core, DMA, selected masters	No
400D_0000		Reserved	—	—	—	—
E000_0000	Non-Secure	Private Peripheral Bus (Internal). Includes NVIC and SYSTICK timer.		256 MB	Core only	No
F000_0000		Reserved	—	—	—	—

1. This is the default Targetted Security Attribute for the memory region at reset. It is set by the IDAU and the SAU configuration. The attribute listed is the intended use case for users of TrustZone-M (TZM) security function.

4.3 Peripheral memory map

The following table provides details of the addresses for APB peripherals.

4.3.1 APB Peripheral Bridge 0 (PBRIDGE0)

Table 5. APB Peripheral Bridge 0 (PBRIDGE0)

Base	Slot	Module	Alias
4000_0000	0	Syscon	SYSCON0
4000_1000	1	IOCON pin function selection and pin control	IOCON

Table continues on the next page...

Table 5. APB Peripheral Bridge 0 (PBRIDGE0) (continued)

Base	Slot	Module	Alias
4000_2000	2	Group GPIO input interrupt 0 (GINT0)	GINT0
4000_3000	3	Group GPIO input interrupt 1 (GINT1)	GINT1
4000_4000	4	GPIO_INT (PINT0)	PINT0
4000_5000	5	Secure GPIO Interrupt (PINT1)	PINT1
4000_6000	6	Peripheral Input Muxes	INPUTMUX0
4000_7000		Reserved	
4000_8000	8	CTimer0 (standard counter/timer 0)	CTIMER0
4000_9000	9	CTimer1 (standard counter/timer 1)	CTIMER1
4000_A000		Reserved	
4000_C000	12	WWDT0 (windowed watchdog timer 0)	WWDT0
4000_D000	13	MRT (Multi-Rate Timer)	MRT0
4000_E000	14	Utick (micro-tick timer)	UTICK0
4000_F000		Reserved	
4001_0000		Reserved	
4001_3000	19	Analog Controls and Frequency Measurement function	ANACTRL0/ FREQME
4001_4000		Reserved	
4001_6000	22	I3C Interface 0	I3C0
4001_7000		Reserved	
4001_B000		Reserved	
4001_C000		Reserved	
4001_D000		Reserved	

4.3.2 APB Peripheral Bridge 1 (PBRIDGE1)

Table 6. APB Peripheral Bridge 1 (PBRIDGE1)

Base	Slot	Module	Alias
4002_0000	0	Power Management Controller (PMC)	PMC
4002_1000		Reserved	
4002_2000		Reserved	
4002_3000	3	Sysctl (I2S signal sharing)	SYSCTL0
4002_4000		Reserved	
4002_5000		Reserved	
4002_8000	8	CTimer2 (standard counter/timer 2)	CTIMER2

Table continues on the next page...

Table 6. APB Peripheral Bridge 1 (PBRIDGE1) (continued)

Base	Slot	Module	Alias
4002_9000	9	CTimer3 (standard counter/timer 3)	CTIMER3
4002_A000	10	CTimer4 (standard counter/timer 4)	CTIMER4
4002_B000		Reserved	
4002_C000	12	RTC and Wake-up timer	RTC
4002_D000	13	OS_Event Timer	OSTIMER_CM33
4002_E000	14	Cache64 Policy Select	CACHE64_POLSEL0
4002_F000		Reserved	
4003_0000		Reserved	
4003_1000		Reserved	
4003_2000		Reserved	
4003_3000		Reserved	
4003_4000	20	Flash controller	FLASH
4003_5000		Reserved	
4003_6000		Reserved	
4003_A000		Reserved	
4003_B000		Reserved	
4003_C000		Reserved	
4003_D000		Reserved	
4003_E000		Reserved	

4.4 AHB Peripheral memory map

The following table provides details of the base addresses for AHB peripherals.

4.4.1 AHB slave port 9

Table 7. AHB slave port 9

Base	Module	Alias
4008_0000	Reserved	
4008_2000	DMA Controller 0 registers	DMAC0
4008_3000	Reserved	
4008_4000	USB Full Speed Device Registers	USBFSD
4008_5000	SCTimer/PWM	SCT0
4008_6000	Flexcomm Interface 0	FLEXCOMM0
4008_7000	Flexcomm Interface 1	FLEXCOMM1

Table continues on the next page...

Table 7. AHB slave port 9 (continued)

Base	Module	Alias
4008_8000	Flexcomm Interface 2	FLEXCOMM2
4008_9000	Flexcomm Interface 3	FLEXCOMM3
4008_A000	Flexcomm Interface 4	FLEXCOMM4
4008_B000	Reserved	
4008_C000	High-Speed GPIO	GPIO_HS

4.4.2 AHB slave port 10

Table 8. AHB slave port 10

Base	Module	Alias
4009_0000	DMIC0 (8 channel PDM digital microphone interface) 0	DMIC0
4009_1000	Reserved	
4009_5000	CRC Engine	CRC
4009_6000	Flexcomm Interface 5	FLEXCOMM5
4009_7000	Flexcomm Interface 6	FLEXCOMM6
4009_8000	Flexcomm Interface 7	FLEXCOMM7
4009_9000	Reserved	
4009_C000	Debug Mailbox (DM-AP)	DEBUGGER_MAILBOX0
4009_D000	CAN FD (Controller Area Network Flexible Data, MCAN)	CAN
4009_E000	Reserved	
4009_F000	High Speed SPI	FLEXCOMM8

4.4.3 AHB slave port 11

Table 9. AHB slave port 11

Base	Module	Alias
400A_0000	Analog to Digital Converter 0 (ADC0)	ADC0
400A_1000	Reserved	
400A_2000	USB Full Speed Host Registers	USBFSH
400A_3000	Reserved	
400A_6000	PowerQuad	POWERQUAD0
400A_7000	DMA0 registers	DMA0
400A_8000	Reserved	
400A_C000	Reserved	

4.4.4 AHB slave port 12

Table 10. AHB slave port 12

Base	Module	Alias
400B_0000	Reserved	
400B_1000	Analog to Digital Converter 1 (ADC1)	ADC1
400B_2000	Digital to Analog Converter 0 (DAC0)	DAC0
400B_3000	High Speed Comparator 0 (HSCMP0)	HSCMP0
400B_4000	Operational Amplifier 0 (OPAMP0)	OPAMP0
400B_5000	Voltage Reference (VREF)	VREF0
400B_6000	Digital to Analog Converter 1 (DAC1)	DAC1
400B_7000	High Speed Comparator 1 (HSCMP1)	HSCMP1
400B_8000	Operational Amplifier 1 (OPAMP1)	OPAMP1
400B_9000	Digital to Analog Converter 2 (DAC2)	DAC2
400B_A000	High Speed Comparator 2 (HSCMP2)	HSCMP2
400B_B000	Operational Amplifier 2 (OPAMP2)	OPAMP2

4.4.5 AHB slave port 13

Table 11. AHB slave port 13

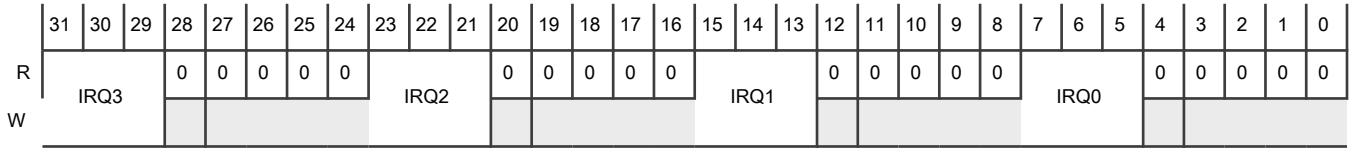
Base	Module	Alias
400C_0000	FlexSPI	FLEXSPI0
400C_1000	Reserved	
400C_2000	Reserved	
400C_3000	Enhanced Flex Pulse Width Modulator 0 (eFlexPWM0)	PWM0
400C_4000	Quadrature Encoder Interface 0 (ENC0)	ENC0
400C_5000	Enhanced Flex Pulse Width Modulator 1 (eFlexPWM1)	PWM1
400C_6000	Quadrature Encoder Interface 1 (ENC1)	ENC1
400C_7000	And-Or-Inverter 0 (AO0)	AOI0
400C_8000	And-Or-Inverter 1 (AO1)	AOI1

Chapter 5
Interrupt and DMA Assignments

5.1 Nested Vectored Interrupt Controller (NVIC)

5.1.1 Interrupt priority levels

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



5.1.2 Non-Maskable Interrupt (NMI) configuration

The Non-Maskable Interrupt (NMI) request to the NVIC is controlled by the external NMI_b signal. The pin the NMI_b signal ismultiplexed on, must be configured for the Non-Maskable Interrupt function to generate the non-maskable interrupt request.

5.1.3 Interrupts

Table 13. Interrupt Vector Assignments

Address	Vector	IRQ	NVIC non-IPR register number	NVIC IPR register number	Alias	Source Description
0000_0000	0	-	-	-	SP initial value	Initial Stack Pointer value
0000_0004	1	-	-	-	Reset	Reset exception
0000_0008	2	-	-	-	NMI	Non-Maskable Interrupt
0000_000C	3	-	-	-	Hard Fault	Error during normal or exception processing
0000_0010	4	-	-	-	MemManage Fault	Memory protection violation
0000_0014	5	-	-	-	NBus Fault	Memory-related violation
0000_0018	6	-	-	-	Usage	Undefined instruction, unaligned access, etc.
0000_001C	7	-	-	-	SecureFault	Security-related fault condition
0000_0020	8	-	-	-	Reserved	(reserved)
0000_0024	9	-	-	-	Reserved	(reserved)
0000_0028	10	-	-	-	Reserved	(reserved)
0000_002C	11	-	-	-	SVCall	Supervisor Call

Table continues on the next page...

Table 13. Interrupt Vector Assignments (continued)

Address	Vector	IRQ	NVIC non-IPR register number	NVIC IPR register number	Alias	Source Description
0000_0030	12	-	-	-	DebugMonitor	Debug Monitor exception
0000_0034	13	-	-	-	Reserved	(reserved)
0000_0038	14	-	-	-	PendSV	Asynchronous request for system-level service
0000_003C	15	-	-	-	SysTick	SysTick timer timed out
0000_0040	16	0	0	0	SYS IRQ	System interrupts: watchdog and brownout detect
0000_0044	17	1	0	0	SDMA0_IRQ	DMA controller 0
0000_0048	18	2	0	0	GPIO_INTA	GPIO interrupt 0
0000_004C	19	3	0	0	GPIO_INTB	GPIO interrupt 1
0000_0050	20	4	0	1	PIN_INT0	Pin interrupt 0 or pattern match engine slice 0
0000_0054	21	5	0	1	PIN_INT1	Pin interrupt 0 or pattern match engine slice 1
0000_0058	22	6	0	1	PIN_INT2	Pin interrupt 0 or pattern match engine slice 2
0000_005C	23	7	0	1	PIN_INT3	Pin interrupt 0 or pattern match engine slice 3
0000_0060	24	8	0	2	UTICK0_IRQ	Micro-tick timer
0000_0064	25	9	0	2	MRT0_IRQ	Multi-rate timer
0000_0068	26	10	0	2	CTIMER0_IRQ	Standard counter/timer CTIMER0
0000_006C	27	11	0	2	CTIMER1_IRQ	Standard counter/timer CTIMER1
0000_0070	28	12	0	3	SCT0_IRQ	SCTimer/PWM
0000_0074	29	13	0	3	CTIMER3_IRQ	Standard counter/timer CTIMER3
0000_0078	30	14	0	3	FLEXCOMM0_IRQ	Flexcomm Interface 0 (USART, SPI, I2C, I2S)
0000_007C	31	15	0	3	FLEXCOMM1_IRQ	Flexcomm Interface 1 (USART, SPI, I2C, I2S)
0000_0080	32	16	0	4	FLEXCOMM2_IRQ	Flexcomm Interface 2 (USART, SPI, I2C, I2S)
0000_0084	33	17	0	4	FLEXCOMM3_IRQ	Flexcomm Interface 3 (USART, SPI, I2C, I2S)
0000_0088	34	18	0	4	FLEXCOMM4_IRQ	Flexcomm Interface 4 (USART, SPI, I2C, I2S)
0000_008C	35	19	0	4	FLEXCOMM5_IRQ	Flexcomm Interface 5 (USART, SPI, I2C, I2S)

Table continues on the next page...

Table 13. Interrupt Vector Assignments (continued)

Address	Vector	IRQ	NVIC non-IPR register number	NVIC IPR register number	Alias	Source Description
0000_0090	36	20	0	5	FLEXCOMM 6_IRQ	Flexcomm Interface 6 (USART, SPI, I2C, I2S)
0000_0094	37	21	0	5	FLEXCOMM 7_IRQ	Flexcomm Interface 7 (USART, SPI, I2C, I2S)
0000_0098	38	22	0	5	ADC0_IRQ	ADC 0
0000_009C	39	23	0	5	ADC1_IRQ	ADC 1
0000_00A0	40	24	0	6	ACMP0_IRQ	Low Power analog comparator 0
0000_00A4	41	25	0	6	D_MIC	Digital microphone
0000_00A8	42	26	0	6	HWVAD	Hardware voice activity detect
0000_00AC	43	27	0	6	USB0_NEEDCLK	USB 0 clock request
0000_00B0	44	28	0	7	USB0_IRQ	USB 0 interrupt
0000_00B4	45	29	0	7	RTC_IRQ	RTC alarm and wakeup interrupts
0000_00B8	46	30	0	7	Reserved	(reserved)
0000_00BC	47	31	0	7	WAKEUP_IRQ	Wakeup
0000_00C0	48	32	1	8	PIN_INT4	Pin interrupt 0 or pattern match engine slice 4
0000_00C4	49	33	1	8	PIN_INT5	Pin interrupt 0 or pattern match engine slice 5
0000_00C8	50	34	1	8	PIN_INT6	Pin interrupt 0 or pattern match engine slice 6
0000_00CC	51	35	1	8	PIN_INT7	Pin interrupt 0 or pattern match engine slice 7
0000_00D0	52	36	1	9	CTIMER2_IRQ	Standard counter/timer CTIMER2
0000_00D4	53	37	1	9	CTIMER4_IRQ	Standard counter/timer CTIMER4
0000_00D8	54	38	1	9	OS_EVENT_TIMER_IRQ	OS event timer
0000_00DC	55	39	1	9	FlexSPI	FlexSPI
0000_00E0	56	40	1	10	Reserved	(reserved)
0000_00E4	57	41	1	10	Reserved	(reserved)
0000_00E8	58	42	1	10	Reserved	(reserved)
0000_00EC	59	43	1	10	CAN0_IRQ0	CAN 0 interrupt 0
0000_00F0	60	44	1	11	CAN0_IRQ1	CAN 0 interrupt 1
0000_00F4	61	45	1	11	Reserved	(reserved)

Table continues on the next page...

Table 13. Interrupt Vector Assignments (continued)

Address	Vector	IRQ	NVIC non-IPR register number	NVIC IPR register number	Alias	Source Description
0000_00F8	62	46	1	11	Reserved	(reserved)
0000_00FC	63	47	1	11	Reserved	(reserved)
0000_0100	64	48	1	12	Reserved	(reserved)
0000_0104	65	49	1	12	Reserved	(reserved)
0000_0108	66	50	1	12	Reserved	(reserved)
0000_010C	67	51	1	12	Reserved	(reserved)
0000_0110	68	52	1	13	FREQMEASURE	Frequency Measurement
0000_0114	69	53	1	13	Reserved	(reserved)
0000_0118	70	54	1	13	Reserved	(reserved)
0000_011C	71	55	1	13	Reserved	(reserved)
0000_0120	72	56	1	14	Reserved	(reserved)
0000_0124	73	57	1	14	PQ_IRQ	PowerQuad math coprocessor
0000_0128	74	58	1	14	SDMA1_IRQ	DMA controller 1
0000_012C	75	59	1	14	HS_SPI_IRQ	High-speed SPI
0000_0130	76	60	1	15	Reserved	(reserved)
0000_0134	77	61	1	15	Reserved	(reserved)
0000_0138	78	62	1	15	I3C0_IRQ	I3C 0 interrupt
0000_013C	79	63	1	15	Reserved	(reserved)
0000_0140	80	64	2	16	Reserved	(reserved)
0000_0144	81	65	2	16	Reserved	(reserved)
0000_0148	82	66	2	16	RTC_Tamper_IRQ	RTC tamper interrupt
0000_014C	83	67	2	16	Reserved	(reserved)
0000_0150	84	68	2	17	Reserved	(reserved)
0000_0154	85	69	2	17	Reserved	(reserved)
0000_0158	86	70	2	17	Reserved	(reserved)
0000_015C	87	71	2	17	Reserved	(reserved)
0000_0160	88	72	2	18	Reserved	(reserved)
0000_0164	89	73	2	18	Reserved	(reserved)
0000_0168	90	74	2	18	DAC0_IRQ	DAC 0

Table continues on the next page...

Table 13. Interrupt Vector Assignments (continued)

Address	Vector	IRQ	NVIC non-IPR register number	NVIC IPR register number	Alias	Source Description
0000_016C	91	75	2	18	DAC1_IRQ	DAC 1
0000_0170	92	76	2	19	DAC2_IRQ	DAC 2
0000_0174	93	77	2	19	HS_CMP0_IRQ	High-Speed comparator 0
0000_0178	94	78	2	19	HS_CMP1_IRQ	High-Speed comparator 1
0000_017C	95	79	2	19	HS_CMP2_IRQ	High-Speed comparator 2
0000_0180	96	80	2	20	FlexPWM0_capture_IRQ	FlexPWM0 input capture (all submodules)
0000_0184	97	81	2	20	FlexPWM0_fault_IRQ	FlexPWM0 Fault input interrupt
0000_0188	98	82	2	20	FlexPWM0_reload_error_IRQ	FlexPWM0 reload error (all submodules)
0000_018C	99	83	2	20	FlexPWM0_compare0_IRQ	FlexPWM0 submodule 0 compare
0000_0190	100	84	2	21	FlexPWM0_reload0_IRQ	FlexPWM0 submodule 0 reload
0000_0194	101	85	2	21	FlexPWM0_compare1_IRQ	FlexPWM0 submodule 1 compare
0000_0198	102	86	2	21	FlexPWM0_reload1_IRQ	FlexPWM0 submodule 1 reload
0000_019C	103	87	2	21	FlexPWM0_compare2_IRQ	FlexPWM0 submodule 2 compare
0000_01A0	104	88	2	22	FlexPWM0_reload2_IRQ	FlexPWM0 submodule 2 reload
0000_01A4	105	89	2	22	FlexPWM0_compare3_IRQ	FlexPWM0 submodule 3 compare
0000_01A8	106	90	2	22	FlexPWM0_reload3_IRQ	FlexPWM0 submodule 3 reload
0000_01AC	107	91	2	22	FlexPWM1_capture_IRQ	FlexPWM1 input capture (all submodules)

Table continues on the next page...

Table 13. Interrupt Vector Assignments (continued)

Address	Vector	IRQ	NVIC non-IPR register number	NVIC IPR register number	Alias	Source Description
0000_01B0	108	92	2	23	FlexPWM1_fault_IRQ	FlexPWM1 Fault input interrupt
0000_01B4	109	93	2	23	FlexPWM1_reload_error_IRQ	FlexPWM1 reload error (all submodules)
0000_01B8	110	94	2	23	FlexPWM1_compare0_IRQ	FlexPWM1 submodule 0 compare
0000_01BC	111	95	2	23	FlexPWM1_reload0_IRQ	FlexPWM1 submodule 0 reload
0000_01C0	112	96	3	24	FlexPWM1_compare1_IRQ	FlexPWM1 submodule 1 compare
0000_01C4	113	97	3	24	FlexPWM1_reload1_IRQ	FlexPWM1 submodule 1 reload
0000_01C8	114	98	3	24	FlexPWM1_compare2_IRQ	FlexPWM1 submodule 2 compare
0000_01CC	115	99	3	24	FlexPWM1_reload2_IRQ	FlexPWM1 submodule 2 reload
0000_01D0	116	100	3	25	FlexPWM1_compare3_IRQ	FlexPWM1 submodule 3 compare
0000_01D4	117	101	3	25	FlexPWM1_reload3_IRQ	FlexPWM1 submodule 3 reload
0000_01D8	118	102	3	25	ENC0_Compare_IRQ	Quadarature Encoder Interface 0 Compare match interrupt
0000_01DC	119	103	3	25	ENC0_Home_IRQ	Quadarature Encoder Interface 0 HOME signal transition interrupt
0000_01E0	120	104	3	26	ENC0_WDG_IRQ	Quadarature Encoder Interface 0 watchdog or Simultaneous PHASEA & PHASEB change
0000_01E4	121	105	3	26	ENC0_IDX_IRQ	Quadarature Encoder Interface 0 INDEX signal transition or roll-over/under interrupt
0000_01E8	122	106	3	26	ENC1_Compare_IRQ	Quadarature Encoder Interface 1 Compare match interrupt
0000_01EC	123	107	3	26	ENC1_Home_IRQ	Quadarature Encoder Interface 1 HOME signal transition interrupt

Table continues on the next page...

Table 13. Interrupt Vector Assignments (continued)

Address	Vector	IRQ	NVIC non-IPR register number	NVIC IPR register number	Alias	Source Description
0000_01F0	124	108	3	27	ENC1_WDG_IRQ	Quadarature Encoder Interface 1 watchdog timeout interrupt
0000_01F4	125	109	3	27	ENC1_IDX_IRQ	Quadarature Encoder Interface 1 INDEX signal transition or roll-over/under interrupt
0000_01F8	126	110	3	27	Reserved	(reserved)
0000_01FC	127	111	3	27	Reserved	(reserved)
0000_0200	128	112	3	28	Reserved	(reserved)
0000_0204	129	113	3	28	Reserved	(reserved)
0000_0208	130	114	3	28	PVTVF0_AMBER_IRQ	PVT0 monitor amber alert (internal feature)
0000_020C	131	115	3	28	PVTVF0_RED_IRQ	PVT0 monitor red alert (internal feature)
0000_0210	132	116	3	29	PVTVF1_AMBER_IRQ	PVT1 monitor amber alert (internal feature)
0000_0214	133	117	3	29	PVTVF1_RED_IRQ	PVT1 monitor red alert (internal feature)
0000_0218	134	118	3	29	FLASH0_IRQ	Flash controller interrupt
0000_021C	135	119	3	29	RAM_PARITY_ECC_ERR	RAM parity error

5.2 DMA request configuration

The two DMA controllers, DMA0 and DMA1, each receive the same DMA requests from peripherals, as shown in the table below. DMA request enables are provided to allow controlling which DMA controller (if any) receives each request.

5.3 DMA request assignments

Table 14. DMA0 request assignments

Slot number	Alias	Description
0	(no DMA request connected)[1]	
1	(no DMA request connected)[2]	
2	HS_LSPI_RX_DMA	HS SPI DMA receive request
3	HS_LSPI_TX_DMA	HS SPI DMA transmit request
4	FLEXCOM0_RX_DMA	FLEXCOM0 DMA receive request / I2C Slave[3]
5	FLEXCOM0_TX_DMA	FLEXCOM0 DMA transmit request / I2C Master[3]

Table continues on the next page...

Table 14. DMA0 request assignments (continued)

Slot number	Alias	Description
6	FLEXCOM1_RX_DMA	FLEXCOM1 DMA receive request / I2C Slave[3]
7	FLEXCOM1_TX_DMA	FLEXCOM1 DMA transmit request/ I2C Master[3]
8	FLEXCOM3_RX_DMA	FLEXCOM3 DMA receive request / I2C Slave[3]
9	FLEXCOM3_TX_DMA	FLEXCOM3 DMA transmit request/ I2C Master[3]
10	FLEXCOM2_RX_DMA	FLEXCOM2 DMA receive request / I2C Slave[3]
11	FLEXCOM2_TX_DMA	FLEXCOM2 DMA transmit request/ I2C Master[3]
12	FLEXCOM4_RX_DMA	FLEXCOM4 DMA receive request / I2C Slave[3]
13	FLEXCOM4_TX_DMA	FLEXCOM4 DMA transmit request/ I2C Master[3]
14	FLEXCOM5_RX_DMA	FLEXCOM5 DMA receive request / I2C Slave[3]
15	FLEXCOM5_TX_DMA	FLEXCOM5 DMA transmit request/ I2C Master[3]
16	FLEXCOM6_RX_DMA	FLEXCOM6 DMA receive request / I2C Slave[3]
17	FLEXCOM6_TX_DMA	FLEXCOM6 DMA transmit request/ I2C Master[3]
18	FLEXCOM7_RX_DMA	FLEXCOM7 DMA receive request / I2C Slave[3]
19	FLEXCOM7_TX_DMA	FLEXCOM7 DMA transmit request/ I2C Master[3]
20	DAC0_DMA	DAC0 DMA request
21	ADC0_DMA0	ADC0 DMA0 request
22	ADC0_DMA1	ADC0 DMA1 request
23	DMIC0_CH0	DMIC0 channel0 DMA request
24	DMIC0_CH1	DMIC0 channel1 DMA request
25	I3C0_RX	I3C0 DMA receive request
26	I3C0_TX	I3C0 DMA transmit request
27	ADC1_DMA0	ADC1 DMA0 request
28	ADC1_DMA1	ADC1 DMA1 request

Table continues on the next page...

Table 14. DMA0 request assignments (continued)

Slot number	Alias	Description
29	DAC1_DMA	DAC1 DMA request
30	DAC2_DMA	DAC2 DMA request
31	unused	
32	unused	
33	unused	
34	unused	
35	unused	
36	(no DMA request connected)	
37	(no DMA request connected)	
38	(no DMA request connected)	
39	(no DMA request connected)	
40	(no DMA request connected)	
41	(no DMA request connected)	
42	(no DMA request connected)	
43	(no DMA request connected)	
44	(no DMA request connected)	
45	(no DMA request connected)	
46	(no DMA request connected)	
47	(no DMA request connected)	
48	(no DMA request connected)	
49	(no DMA request connected)	
50	(no DMA request connected)	
51	(no DMA request connected)	

1. If the FlexSPI RX DMA trigger is used, it must be used with this DMA channel.
2. If the FlexSPI TX DMA trigger is used, it must be used with this DMA channel.
3. See DMA with I2C monitor mode for information about DMA for the I2C monitor function.

Table 15. DMA1 request assignments

Slot number	Alias	Description
0	unused (tie-low)	
1	unused (tie-low)	
2	HS_LSPI_RX_DMA	HS SPI receive DMA request
3	HS_LSPI_TX_DMA	HS SPI transmit DMA request

Table continues on the next page...

Table 15. DMA1 request assignments (continued)

Slot number	Alias	Description
4	FLEXCOM0_RX_DMA	FLEXCOM0 receive DMA request / I2C Slave[1]
5	FLEXCOM0_TX_DMA	FLEXCOM0 transmit DMA request / I2C Master[1]
6	FLEXCOM1_RX_DMA	FLEXCOM1 receive DMA request / I2C Slave[1]
7	FLEXCOM1_TX_DMA	FLEXCOM1 transmit DMA request / I2C Master[1]
8	FLEXCOM3_RX_DMA	FLEXCOM3 receive DMA request / I2C Slave[1]
9	FLEXCOM3_TX_DMA	FLEXCOM3 transmit DMA request / I2C Master[1]
10	DMIC0_CH0	DMIC0 channel0 DMA request
11	DMIC0_CH1	DMIC0 channel1 DMA request
12	I3C0_RX	I3C0 receive request
13	I3C0_TX	I3C0 transmit request
14	(no DMA request connected)[2]	
15	(no DMA request connected)[2]	

1. See DMA with I2C monitor mode for information about DMA for the I2C monitor function.
2. If the FlexSPI TX DMA trigger is used, it must be used with this DMA channel.

5.3.1 DMA with I2C monitor mode

The I2C monitor function may be used with DMA if one of the channels related to the same Flexcomm Interface is available.

Table 16. DMA with the I2C

I2C Master DMA	I2C Slave DMA	I2C monitor DMA
Not enabled	-	If I2C Monitor DMA is enabled, it will use the DMA channel for the Master function other same Flexcomm Interface.
Enabled	Not enabled	If I2C Monitor is DMA enabled, it will use the DMA channel for the Slave function of the same Flexcomm Interface.
Enabled	Enabled	The I2C Monitor function cannot use DMA.

5.4 DMA trigger input multiplexing

The DMA can use trigger input multiplexing to sequence DMA transactions without the use of interrupt service routines. The trigger input multiplexing for each DMA controller is configured as shown in Figure 6.

In each DMA controller, four of these input triggers are selected from the DMA trigger outputs, controlled by the DMA_OTRIG_INMUX registers. See DMA output triggers for DMA0 and DMA1 output trigger multiplexers.

With the DMA trigger input mux registers (DMAC_ITRIG_SEL), one trigger input can be selected for each of the DMA channels from the potential internal sources. By default, none of the triggers are selected. DMA input trigger assignments shows the peripheral connections for DMA0 and DMA1 trigger inputs.

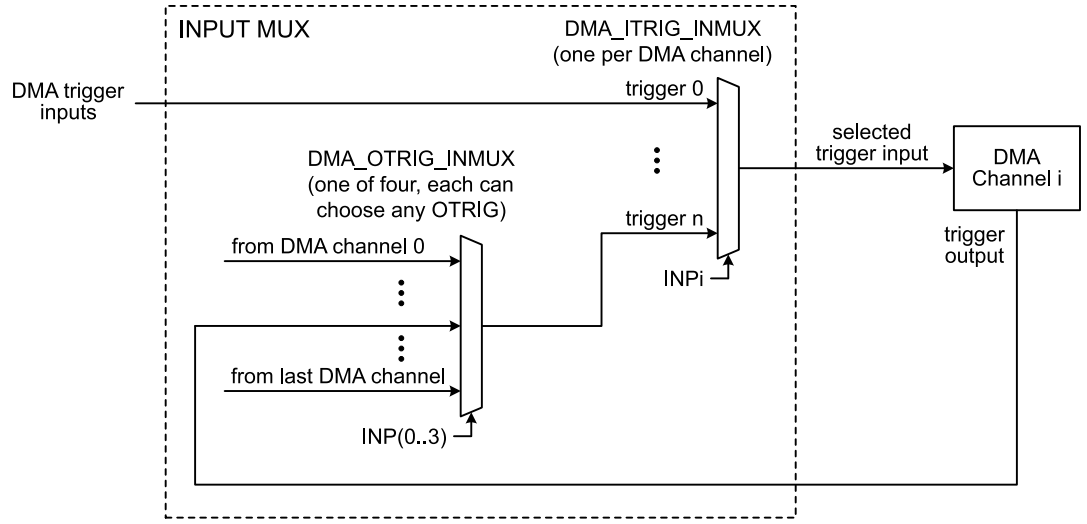


Figure 6. DMA trigger multiplexing

5.4.1 DMA input trigger assignments

Table 17. DMA0 input trigger multiplexing assignments

Slot Number	Alias	Source Description
0	FlexSPI_RX	FlexSPI receive
1	FlexSPI_TX	FlexSPI transmit
2	GPIO_INT0	General Purpose Input-Output Interrupt 0
3	GPIO_INT1	General Purpose Input-Output Interrupt 1
4	GPIO_INT2	General Purpose Input-Output Interrupt 2
5	GPIO_INT3	General Purpose Input-Output Interrupt 3
6	CTIMER0_DMAREQ_M0	CTIMER0_DMAREQ_M0
7	CTIMER0_DMAREQ_M1	CTIMER0_DMAREQ_M1
8	CTIMER1_DMAREQ_M0	CTIMER1_DMAREQ_M0
9	CTIMER1_DMAREQ_M1	CTIMER1_DMAREQ_M1
10	CTIMER2_DMAREQ_M0	CTIMER2_DMAREQ_M0
11	CTIMER2_DMAREQ_M1	CTIMER2_DMAREQ_M1
12	CTIMER3_DMAREQ_M0	CTIMER3_DMAREQ_M0

Table continues on the next page...

Table 17. DMA0 input trigger multiplexing assignments (continued)

Slot Number	Alias	Source Description
13	CTIMER3_DMAREQ_M1	CTIMER3_DMAREQ_M1
14	CTIMER4_DMAREQ_M0	CTIMER4_DMAREQ_M0
15	CTIMER4_DMAREQ_M1	CTIMER4_DMAREQ_M1
16	ACMP0_OUT	ACMP0_OUT
17	DMA0_TRIGOUT_A	DMA Trigger output A
18	DMA0_TRIGOUT_B	DMA Trigger output B
19	DMA0_TRIGOUT_C	DMA Trigger output C
20	DMA0_TRIGOUT_D	DMA Trigger output C
21	SCT_DMA0	SCTimer DMA request 0
22	SCT_DMA1	SCTimer DMA request 1
23	ADC0_tcomp[0]	ADC0_TCOMP[0]
24	ADC1_tcomp[0]	ADC1_TCOMP[1]
25	HS-CMP0	HS-CMP0
26	HS-CMP1	HS-CMP1
27	HS-CMP2	HS-CMP2
28	AOI0_OUT0	AOI0_OUT0
29	AOI0_OUT1	AOI0_OUT1
30	AOI0_OUT2	AOI0_OUT2
31	AOI0_OUT3	AOI0_OUT3
32	AOI1_OUT0	AOI1_OUT0
33	AOI1_OUT1	AOI1_OUT1
34	AOI1_OUT2	AOI1_OUT2
35	AOI1_OUT3	AOI1_OUT3
36	FlexPWM0_req_capt0	FlexPWM0_req_capt0
37	FlexPWM0_req_capt1	FlexPWM0_req_capt1
38	FlexPWM0_req_capt2	FlexPWM0_req_capt2
39	FlexPWM0_req_capt3	FlexPWM0_req_capt3
40	FlexPWM0_req_val0	FlexPWM0_req_val0
41	FlexPWM0_req_val1	FlexPWM0_req_val1
42	FlexPWM0_req_val2	FlexPWM0_req_val2
43	FlexPWM0_req_val3	FlexPWM0_req_val3
44	FlexPWM1_req_capt0	FlexPWM1_req_capt0

Table continues on the next page...

Table 17. DMA0 input trigger multiplexing assignments (continued)

Slot Number	Alias	Source Description
45	FlexPWM1_req_capt1	FlexPWM1_req_capt1
46	FlexPWM1_req_capt2	FlexPWM1_req_capt2
47	FlexPWM1_req_capt3	FlexPWM1_req_capt3
48	FlexPWM1_req_val0	FlexPWM1_req_val0
49	FlexPWM1_req_val1	FlexPWM1_req_val1
50	FlexPWM1_req_val2	FlexPWM1_req_val2
51	FlexPWM1_req_val3	FlexPWM1_req_val3
52	TMPR_OUT	TMPR_OUT

Table 18. DMA1 input trigger multiplexing assignments

Slot Number	Alias	Source Description
0	GPIO_INT0	GPIO_INT0
1	GPIO_INT1	GPIO_INT1
2	GPIO_INT2	GPIO_INT2
3	GPIO_INT3	GPIO_INT3
4	T0_DMAREQ_M0	T0_DMAREQ_M0
5	T0_DMAREQ_M1	T0_DMAREQ_M1
6	T2_DMAREQ_M0	T2_DMAREQ_M0
7	T4_DMAREQ_M0	T4_DMAREQ_M0
8	SDMA1_TRIGOUT_A	SDMA1_TRIGOUT_A
9	SDMA1_TRIGOUT_B	SDMA1_TRIGOUT_B
10	SDMA1_TRIGOUT_C	SDMA1_TRIGOUT_C
11	SDMA1_TRIGOUT_D	SDMA1_TRIGOUT_D
12	SCT_DMA_REQ0	SCT_DMA_REQ0
13	SCT_DMA_REQ1	SCT_DMA_REQ1
14	FlexSPI_RX	FlexSPI_RX
15	FlexSPI_TX	FlexSPI_TX
16	AOI0_OUT0	AOI0_OUT0
17	AOI0_OUT1	AOI0_OUT1
18	AOI0_OUT2	AOI0_OUT2
19	AOI0_OUT3	AOI0_OUT3
20	AOI1_OUT0	AOI1_OUT0
21	AOI1_OUT1	AOI1_OUT1

Table continues on the next page...

Table 18. DMA1 input trigger multiplexing assignments (continued)

Slot Number	Alias	Source Description
22	AOI1_OUT2	AOI1_OUT2
23	AOI1_OUT3	AOI1_OUT3
24	TMPR_OUT	TMPR_OUT

5.5 DMA output triggers

Table 19. DMA0 output trigger assignments

Slot Number	Alias	Source Description
0	SDMA0_CH0_TRIGOUT	DMA0 channel 0 output trigger
1	SDMA0_CH1_TRIGOUT	DMA0 channel 1 output trigger
2	SDMA0_CH2_TRIGOUT	DMA0 channel 2 output trigger
3	SDMA0_CH3_TRIGOUT	DMA0 channel 3 output trigger
4	SDMA0_CH4_TRIGOUT	DMA0 channel 4 output trigger
5	SDMA0_CH5_TRIGOUT	DMA0 channel 5 output trigger
6	SDMA0_CH6_TRIGOUT	DMA0 channel 6 output trigger
7	SDMA0_CH7_TRIGOUT	DMA0 channel 7 output trigger
8	SDMA0_CH8_TRIGOUT	DMA0 channel 8 output trigger
9	SDMA0_CH9_TRIGOUT	DMA0 channel 9 output trigger
10	SDMA0_CH10_TRIGOUT	DMA0 channel 10 output trigger
11	SDMA0_CH11_TRIGOUT	DMA0 channel 11 output trigger
12	SDMA0_CH12_TRIGOUT	DMA0 channel 12 output trigger
13	SDMA0_CH13_TRIGOUT	DMA0 channel 13 output trigger
14	SDMA0_CH14_TRIGOUT	DMA0 channel 14output trigger
15	SDMA0_CH15_TRIGOUT	DMA0 channel 15 output trigger
16	SDMA0_CH16_TRIGOUT	DMA0 channel 16 output trigger
17	SDMA0_CH17_TRIGOUT	DMA0 channel 17 output trigger
18	SDMA0_CH18_TRIGOUT	DMA0 channel 18 output trigger
19	SDMA0_CH19_TRIGOUT	DMA0 channel 19 output trigger
20	SDMA0_CH20_TRIGOUT	DMA0 channel 20 output trigger
21	SDMA0_CH21_TRIGOUT	DMA0 channel 21 output trigger
22	SDMA0_CH22_TRIGOUT	DMA0 channel 22 output trigger
23	SDMA0_CH23_TRIGOUT	DMA0 channel 23 output trigger
24	SDMA0_CH24_TRIGOUT	DMA0 channel 24 output trigger

Table continues on the next page...

Table 19. DMA0 output trigger assignments (continued)

Slot Number	Alias	Source Description
25	SDMA0_CH25_TRIGOUT	DMA0 channel 25 output trigger
26	SDMA0_CH26_TRIGOUT	DMA0 channel 26 output trigger
27	SDMA0_CH27_TRIGOUT	DMA0 channel 27 output trigger
28	SDMA0_CH28_TRIGOUT	DMA0 channel 28 output trigger
29	SDMA0_CH29_TRIGOUT	DMA0 channel 29 output trigger
30	SDMA0_CH30_TRIGOUT	DMA0 channel 30 output trigger
31	SDMA0_CH31_TRIGOUT	DMA0 channel 31 output trigger
32	SDMA0_CH32_TRIGOUT	DMA0 channel 32 output trigger
33	SDMA0_CH33_TRIGOUT	DMA0 channel 33 output trigger
34	SDMA0_CH34_TRIGOUT	DMA0 channel 34 output trigger
35	SDMA0_CH35_TRIGOUT	DMA0 channel 35 output trigger
36	SDMA0_CH36_TRIGOUT	DMA0 channel 36 output trigger
37	SDMA0_CH37_TRIGOUT	DMA0 channel 37 output trigger
38	SDMA0_CH38_TRIGOUT	DMA0 channel 38 output trigger
39	SDMA0_CH39_TRIGOUT	DMA0 channel 39 output trigger
40	SDMA0_CH40_TRIGOUT	DMA0 channel 40 output trigger
41	SDMA0_CH41_TRIGOUT	DMA0 channel 41 output trigger
42	SDMA0_CH42_TRIGOUT	DMA0 channel 42 output trigger
43	SDMA0_CH43_TRIGOUT	DMA0 channel 43 output trigger
44	SDMA0_CH44_TRIGOUT	DMA0 channel 44 output trigger
45	SDMA0_CH45_TRIGOUT	DMA0 channel 45 output trigger
46	SDMA0_CH46_TRIGOUT	DMA0 channel 46 output trigger
47	SDMA0_CH47_TRIGOUT	DMA0 channel 47 output trigger
48	SDMA0_CH48_TRIGOUT	DMA0 channel 48 output trigger
49	SDMA0_CH49_TRIGOUT	DMA0 channel 49 output trigger
50	SDMA0_CH50_TRIGOUT	DMA0 channel 50 output trigger
51	SDMA0_CH51_TRIGOUT	DMA0 channel 51 output trigger
52	SDMA0_CH52_TRIGOUT	DMA0 channel 52 output trigger

Table 20. DMA1 output trigger assignments

Slot Number	Alias	Source Description
0	SDMA1_CH0_TRIGOUT	DMA1 channel0 output trigger
1	SDMA1_CH1_TRIGOUT	DMA1 channel1 output trigger

Table continues on the next page...

Table 20. DMA1 output trigger assignments (continued)

Slot Number	Alias	Source Description
2	SDMA1_CH2_TRIGOUT	DMA1 channel2 output trigger
3	SDMA1_CH3_TRIGOUT	DMA1 channel3 output trigger
4	SDMA1_CH4_TRIGOUT	DMA1 channel4 output trigger
5	SDMA1_CH5_TRIGOUT	DMA1 channel5 output trigger
6	SDMA1_CH6_TRIGOUT	DMA1 channel6 output trigger
7	SDMA1_CH7_TRIGOUT	DMA1 channel7 output trigger
8	SDMA1_CH8_TRIGOUT	DMA1 channel8 output trigger
9	SDMA1_CH9_TRIGOUT	DMA1 channel9 output trigger
10	SDMA1_CH10_TRIGOUT	DMA1 channel10 output trigger
11	SDMA1_CH11_TRIGOUT	DMA1 channel11 output trigger
12	SDMA1_CH12_TRIGOUT	DMA1 channel12 output trigger
13	SDMA1_CH13_TRIGOUT	DMA1 channel13 output trigger
14	SDMA1_CH14_TRIGOUT	DMA1 channel14 output trigger
15	SDMA1_CH15_TRIGOUT	DMA1 channel15 output trigger

Chapter 6
Signal Muxing and Pinouts

6.1 Introduction

To optimize functionality in small packages, pins have several functions available via signal multiplexing. This chapter illustrates which of this device's signals are multiplexed on which external pin.

See the attached spreadsheet to know how the pins available on this device are configured. The available device packages and pinout diagrams are also provided in the attached spreadsheet.

6.1.1 Signal multiplexing constraints

- A given module signal must be assigned to a maximum of one package pin. Do not program the same function to more than one pin.
- To ensure the best signal timing for a given peripheral's interface, choose the pins in closest proximity to each other.

6.2 Pinout

See the attached "LPC553x Pin Functions" spreadsheet for pinouts.

Chapter 7
I/O Pin Configuration (IOCON)

7.1 Chip-specific IOCON information

Table 21. Reference links to related information

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

7.1.1 Module instances

This device has one instance of the IOCON module, IOCON.

7.2 Overview

The IOCON module controls the functions and properties of external pins on the chip. Each port pin PIOm_n has one IOPCTL register assigned to control the characteristics of the pin. Registers for pins that are not available on a specific package are reserved.

7.2.1 Features

All pins are standard MFIO(Multi Function Input Output) port pins except PIO_13 andPIO_14 pins which are combo I^2C/MFIO port pins. The following electrical properties are configurable for standard port pins:

- Pull-up/pull-down resistor.
- High-speed mode.
- Open-drain mode.
- Inverted function.

Pins PIO0_13, PIO0_14, can be set either as standard port pins or as true open-drain pins that can be configured for different I^2C-bus speeds. Configuration options are somewhat different for these pins, as described in this chapter. See the device datasheet for more details.

SCTimer/PWM inputs, frequency measure, JTAG signals, and ADC triggers are not selected through IOCON. The connections for these signals are described in INPUTMUX.

7.3 Functional description

The following sections describe functional details of the IOCON module.

7.3.1 Pin Configuration

The figure below shows the standard GPIO pin configuration. Some pins are configured as digital inputs and digital outputs. Some are configured as analog IO with analog switch. For more details on availability of analog signals for each GPIO refer Analog/Digital mode

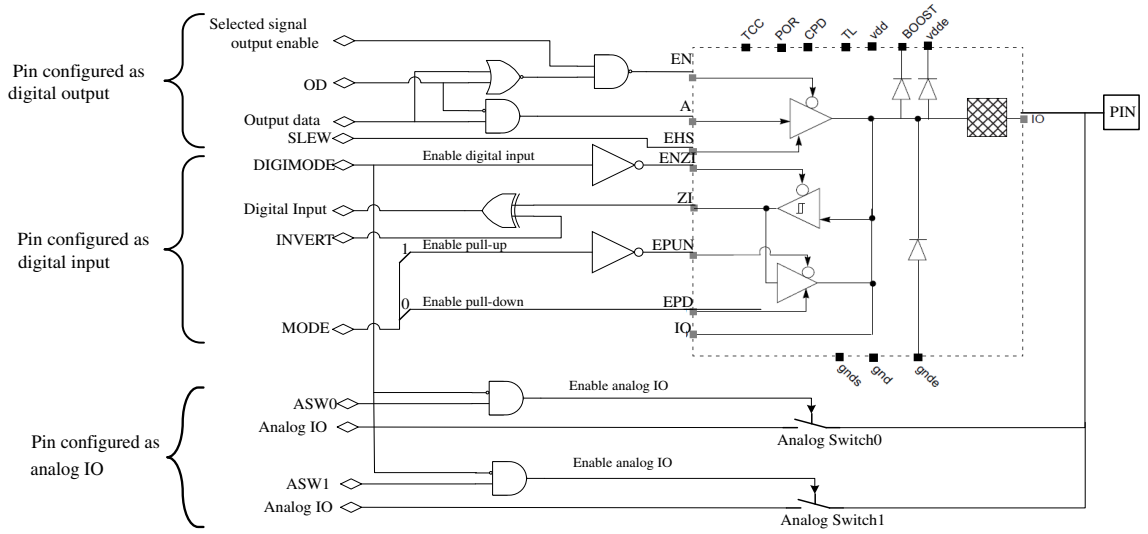


Figure 7. Standard GPIO pin configuration

The figure below shows the configuration of a combo I²C/MFIO pin. ASW input signal is not represented since there is no analog IO associated with it.

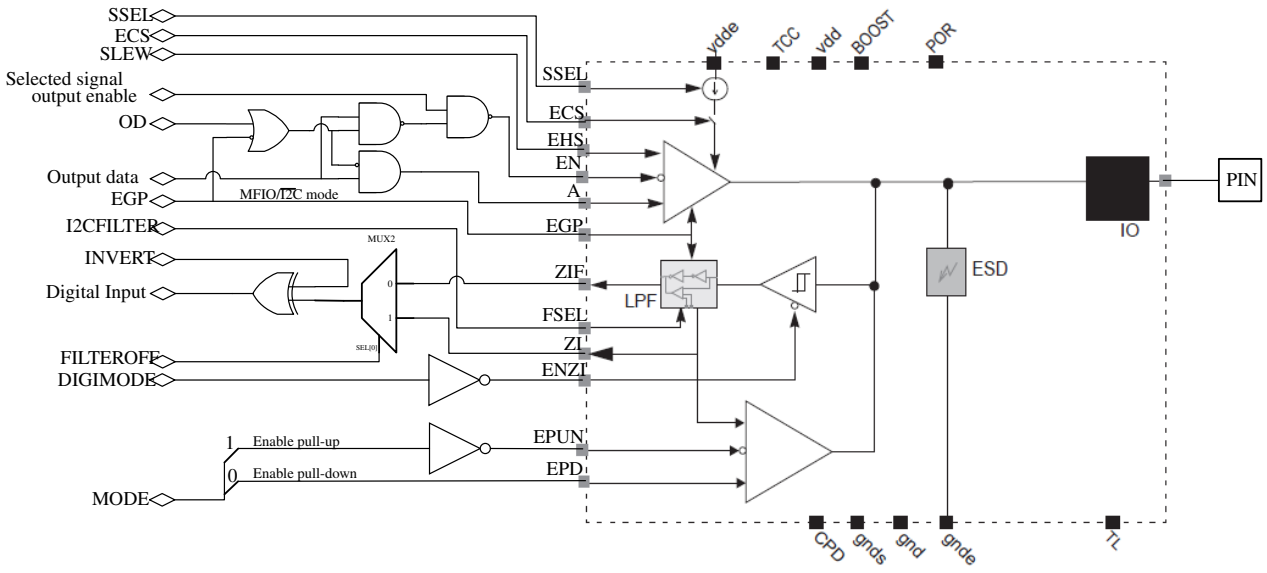


Figure 8. Combo I²C/GPIO pin configuration

7.3.2 Pin mode

The possible on-chip resistor configurations are

- Pull-up enabled
- Pull-down enabled
- No pull-up/pull-down

The default value for most of the pins are no pull-up/down and input disabled (tristated) except:

- Pull-up enable, Pull-down disable, Input enable: SWDIO - PIO_9, ISPSelect - PIO_7
- Pull-up disable, Pull-down enable, Input enable: SWCLK - PIO_0, TRSTN - PIO_2, TCK - PIO_3, TMS - PIO_4

The repeater mode enables the pull-up resistor if the pin is high and enables the pull-down resistor if the pin is low. This causes the pin to retain its last known state if it is configured as an input and is not driven externally. This state retention is not applicable to the deep power-down mode. Repeater mode may typically be used to prevent a pin from floating and potentially using significant power if it floats to an indeterminate state if it is temporarily not driven.

7.3.3 Analog/Digital mode

A pin can be set in analog mode when DIGIMODE bit is set to 0 and analog switch ASW bit is set to 1 in the Control register. A pin can be set in digital mode when DIGIMODE bit is set to 1 and analog switch ASW bit is set to 0 in the Control register.

NOTE

Analog/Digital pins with no switch control do not support ASW bit in the Control register. Analog/Digital pins with one switch control support one ASW bit and two switch control support two ASW bits in the Control register.

This protects the analog input from voltages outside the range of the analog power supply and reference that may sometimes be present on digital pins. All pin types include this control, even if they do not support any analog signals. The digital output can be disabled by selecting GPIO (FUNC bit set to 0 in Control register) and the corresponding DIR bit set to 1 (See GPIO).

In order to use a pin that has an analog IO (ADC, Comparator, and so on) option for that purpose, select GPIO function (FUNC bit set to 0 in Control register) and the corresponding DIR bit set to 0 (See GPIO); disable the digital pin function (DIGIMODE bit set to 0) and enable the analog switch (ASW bit set to 1). The MODE field should also be set to 0.

In analog mode, the MODE field should be set to 0. The INVERT, FILTEROFF, and OD settings have no effect. For an unconnected pin that has an analog signal, the ASW bit should be set to 0 (analog IO disabled), DIGIMODE bit should be set to 0 (digital input disabled) and the MODE bit should be set to no pull-up no pull-down mode. It isolates the pin from the circuit inside and saves power.

PIN	Analog Signals		
	Direct Connect	Analog Switch 0 (ASW0)	Analog Switch 1 (ASW1)
ANALOG PAD			
ADC1IN1A	ADC1IN1A		
ADC1IN1B	ADC1IN1B		
DAC0_OUT	DAC0_OUT/ADC1IN4A/ ADC0IN4A/HSCMP0_IN5		
OPAMP0_INN	OPAMP0_INN		
OPAMP1_INN	OPAMP1_INN		
OPAMP2_INN	OPAMP2_INN		
VREF_OUT	VREF_OUT/ADC1IN5A/ ADC0IN5A		
VREFP	VREFP		
VREFN	VREFN/ADC1IN5B/ ADC0IN5B		
GPIO PAD			
PIO0_0		ACMP0_A	

Table continues on the next page...

Table continued from the previous page...

PIN	Analog Signals		
	Direct Connect	Analog Switch 0 (ASW0)	Analog Switch 1 (ASW1)
PIO0_1	ADC1IN2B		
PIO0_7		HSCMP1_IN0	
PIO0_8	OPAMP0_INP		
PIO0_9		ACMP0_B	
PIO0_10	ADC0IN1A		
PIO0_11	ADC1IN2A		
PIO0_12	ADC1IN3A		
PIO0_15	ADC0IN3A		
PIO0_16	ADC0IN3B		
PIO0_17		HSCMP2_IN0	
PIO0_18		ACMP0_C	
PIO0_23		ADC0IN8B	
PIO0_24		HSCMP0_IN0	
PIO0_27	OPAMP1_INP		
PIO0_31		ADC0IN8A	
PIO1_0	ADC1IN0B		
PIO1_5		HSCMP0_IN3	
PIO1_7	ADC1IN3B		
PIO1_9	ADC0IN0A/OPAMP0_OUT	HSCMP0_IN4	
PIO1_10		HSCMP1_IN3	
PIO1_12		HSCMP0_IN1	
PIO1_13		VREF_COMP	
PIO1_14		ACMP0_D	
PIO1_19	DAC1_OUT/ADC0IN4B/ HSCMP1_IN5		
PIO1_20		ADC1IN8A	
PIO1_22		HSCMP1_IN1	DAC0_OUT
PIO1_23		HSCMP2_IN1	
PIO1_24		ADC1IN8B	
PIO2_0		ADC0IN9A	

Table continues on the next page...

Table continued from the previous page...

PIN	Analog Signals		
	Direct Connect	Analog Switch 0 (ASW0)	Analog Switch 1 (ASW1)
PIO2_1	OPAMP2_INP		
PIO2_2	ADC1IN0A/ADC0IN2A/ ADC0IN2B/OPAMP2_OUT	HSCMP2_IN4	
PIO2_4	DAC2_OUT/ADC1IN4B/ HSCMP2_IN5		
PIO2_14	ADC0IN0B/OPAMP1_OUT	HSCMP1_IN4	

7.3.4 I²C modes

Pins that support I²C (PIO0_13 and PIO0_14) have additional configuration bits. These have multiple configurations to support I²C variants. These pins can be used for non-I²C signals.

For non-I²C operation, these pins can be open-drain or not, as standard GPIO pins.

Table 22. IOCON settings for I²C signals

Mode	IOCON register bit						
	15: I²CFILTER	14: I²CDRIVE	13: ECS	12: FILTEROFF	8: DIGIMODE	7: INVERT	6: SLEW
GPIO low-speed mode	-	1	-	0 [1]	1 [2]	0	0
GPIO high-speed mode	-	1	-	1 [1]	1 [2]	0	1
Standard-mode I²C	1	0	0	0	1	0	0
Fast-mode Plus I²C	0	0	0	0	1	0	0
High-speed slave I²C	-	0	1	1	1	0	0

1. The input filter may be turned off by setting FILTEROFF if it is not needed.
2. The input may be turned off by clearing DIGIMODE if it is not needed.

7.3.5 Hysteresis

The input buffer for digital signals has built-in hysteresis. See device data sheet for details.

7.3.6 Invert pin

This option is included to avoid having to include an external inverter on an input that is meant to be the opposite polarity of the external signal. By default this option is disabled.

7.3.7 Input filter

Some pins include a filter that can be selectively disabled by setting the FILTEROFF bit. This concerns combo I²C/MFIO pins. The filter suppresses input pulses smaller than about 3ns in MFIO mode and smaller than 10 ns or 50 ns in I²C mod, depending on the value of I²CFILTER field.

7.3.8 Output slew rate

The SLEW bits of digital outputs that do not need to switch state should be set to "standard". This setting allows multiple outputs to switch simultaneously without noticeably degrading the power/ground distribution of the device, and has a small effect on signal transition time. This is particularly important if analog accuracy is significant to the application. See the relevant specific device data sheet for more details.

7.4 Initialization

Enable the clock to the IOCON in the AHBCLKCTRL0 register, see CLKCTL. Once the pins are configured, the IOCON clock can be disabled in order to conserve power.

7.5 Pin Multiplexing

See attached Niobe4Analog pinout spreadsheet.

7.6 Memory Map and register definition

The IOCON registers control the signal selection and electrical properties of pins. Each GPIO pin has a dedicated control register to select its unique signal and characteristics. Pins that have an I2C signal can be configured for different I2C-bus modes, while pins that have an analog alternate signal have an analog mode that can be selected.

A particular peripheral signal may be allowed on more than one pin, so it is possible to configure more than one pin to have the same signal. If a peripheral output signal is configured to appear on more than one pin, it will be routed to those pins. If a peripheral input signal is coming from more than one source, the values will be logically combined, possibly resulting in incorrect peripheral operation.

7.6.1 IOCON register descriptions

Each port pin PIO has one IOCON register assigned to control the electrical characteristics of the pin.

There are five types of pin functions:

- Digital only pins
- Analog/digital pins (no switch control)
- Analog/digital pins (one switch control)
- Analog/digital pins (two switch control)
- I2C pins

7.6.1.1 IOCON memory map

IOCON.IOCON base address: 4000_1000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Analog/Digital I/O control for port (PIO0_0)	32	See section	0000_0119
4	Analog/Digital I/O control for port (PIO0_1)	32	See section	0000_0000
8 - 10	Digital I/O control for port (PIO0_2 - PIO0_4)	32	See section	0000_0110

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
14	Digital I/O control for port (PIO0_5)	32	See section	0000_0120
18	Digital I/O control for port (PIO0_6)	32	See section	0000_0000
1C	Analog/Digital I/O control for port (PIO0_7)	32	See section	0000_0120
20	Analog/Digital I/O control for port (PIO0_8)	32	See section	0000_0000
24	Analog/Digital I/O control for port (PIO0_9)	32	See section	0000_0129
28 - 30	Analog/Digital I/O control for port (PIO0_10 - PIO0_12)	32	See section	0000_0000
34 - 38	I2C control for port (PIO0_13 - PIO0_14)	32	See section	0000_5000
3C - 40	Analog/Digital I/O control for port (PIO0_15 - PIO0_16)	32	See section	0000_0000
44 - 48	Analog/Digital I/O control for port (PIO0_17 - PIO0_18)	32	See section	0000_0000
4C - 58	Digital I/O control for port (PIO0_19 - PIO0_22)	32	See section	0000_0000
5C - 60	Analog/Digital I/O control for port (PIO0_23 - PIO0_24)	32	See section	0000_0000
64 - 68	Digital I/O control for port (PIO0_25 - PIO0_26)	32	See section	0000_0000
6C	Analog/Digital I/O control for port (PIO0_27)	32	See section	0000_0000
70 - 78	Digital I/O control for port (PIO0_28 - PIO0_30)	32	See section	0000_0000
7C	Analog/Digital I/O control for port (PIO0_31)	32	See section	0000_0000
80 - 90	Digital I/O control for port (PIO1_0 - PIO1_4)	32	See section	0000_0000
94	Analog/Digital I/O control for port (PIO1_5)	32	See section	0000_0000
98	Digital I/O control for port (PIO1_6)	32	See section	0000_0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
9C	Analog/Digital I/O control for port (PIO1_7)	32	See section	0000_0000
A0	Digital I/O control for port (PIO1_8)	32	See section	0000_0000
A4 - A8	Analog/Digital I/O control for port (PIO1_9 - PIO1_10)	32	See section	0000_0000
AC	Digital I/O control for port (PIO1_11)	32	See section	0000_0000
B0 - B8	Analog/Digital I/O control for port (PIO1_12 - PIO1_14)	32	See section	0000_0000
BC - C8	Digital I/O control for port (PIO1_15 - PIO1_18)	32	See section	0000_0000
CC	Analog/Digital I/O control for port (PIO1_19)	32	See section	0000_0000
D0	Analog/Digital I/O control for port (PIO1_20)	32	See section	0000_0000
D4	Digital I/O control for port (PIO1_21)	32	See section	0000_0000
D8	Analog/Digital I/O control for port (PIO1_22)	32	See section	0000_0000
DC - E0	Analog/Digital I/O control for port (PIO1_23 - PIO1_24)	32	See section	0000_0000
E4 - FC	Digital I/O control for port (PIO1_25 - PIO1_31)	32	See section	0000_0000
100	Analog/Digital I/O control for port (PIO2_0)	32	See section	0000_0000
104	Analog/Digital I/O control for port (PIO2_1)	32	See section	0000_0000
108	Analog/Digital I/O control for port (PIO2_2)	32	See section	0000_0000
10C	Digital I/O control for port (PIO2_3)	32	See section	0000_0000
110	Analog/Digital I/O control for port (PIO2_4)	32	See section	0000_0000
114 - 118	Analog/Digital I/O control for port (PIO2_5 - PIO2_6)	32	See section	0000_0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
11C	Digital I/O control for port (PIO2_7)	32	See section	0000_0000
120	Analog/Digital I/O control for port (PIO2_8)	32	See section	0000_0000
124 - 128	Digital I/O control for port (PIO2_9 - PIO2_10)	32	See section	0000_0000
12C - 138	Analog/Digital I/O control for port (PIO2_11 - PIO2_14)	32	See section	0000_0000
13C - 148	Digital I/O control for port (PIO2_15 - PIO2_18)	32	See section	0000_0000
14C - 150	Analog/Digital I/O control for port (PIO2_19 - PIO2_20)	32	See section	0000_0000
154 - 158	Digital I/O control for port (PIO2_21 - PIO2_22)	32	See section	0000_0000
15C - 160	Analog/Digital I/O control for port (PIO2_23 - PIO2_24)	32	See section	0000_0000
164 - 198	Digital I/O control for port (PIO2_25 - PIO3_6)	32	See section	0000_0000

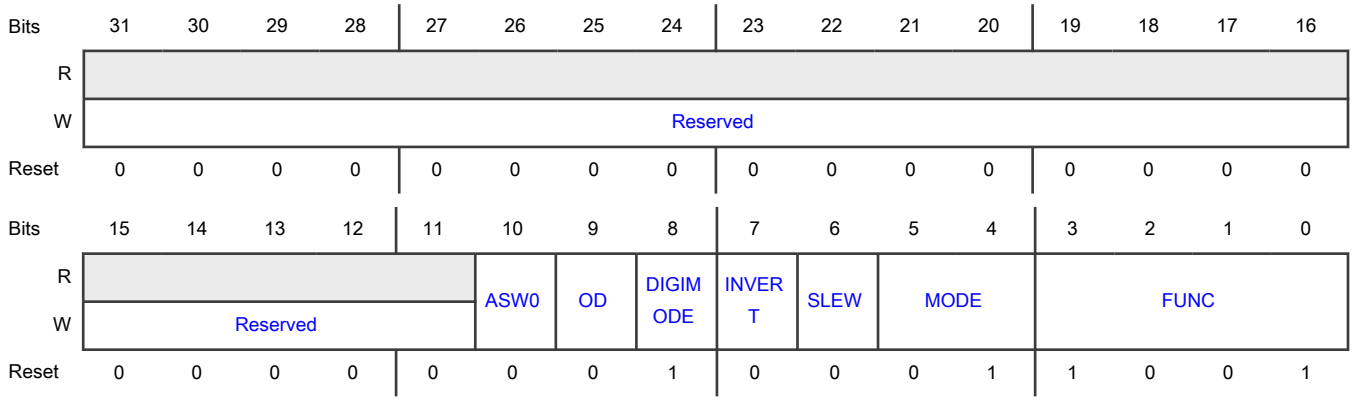
7.6.1.1.1 Analog/Digital I/O control for port (PIO0_0)

Type A registers.

Offset

Register	Offset
PIO0_0	0h

Diagram



Fields

Field	Description
31-11 —	Reserved Read value is undefined, only zero should be written.
10 ASW0	Analog switch input control Usable only if DIGIMODE = 0. To prevent digital noise, set FUNC = 0000 and MODE = 00. 0 - Analog switch is open. (disable) 1 - Analog switch is closed. (enable)
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

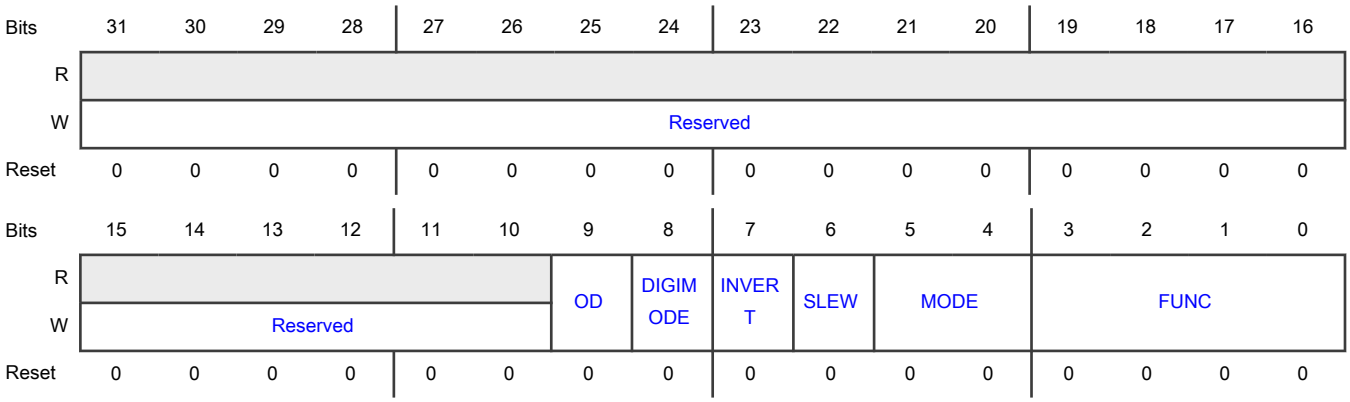
7.6.1.1.2 Analog/Digital I/O control for port (PIO0_1)

Type A registers.

Offset

Register	Offset
PIO0_1	4h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4	Mode select (on-chip pull-up/pull-down resistor control)

Table continues on the next page...

Table continued from the previous page...

Field	Description
MODE	These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

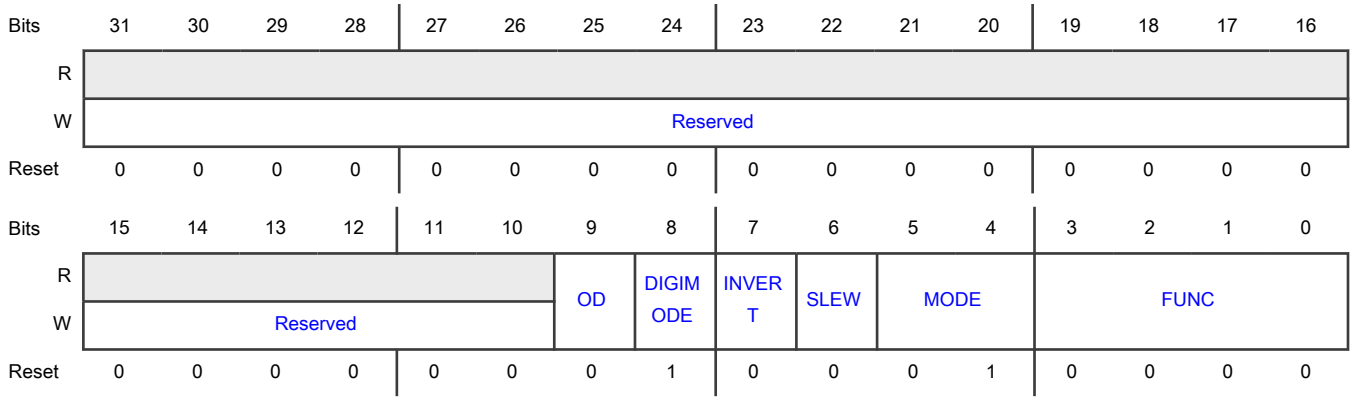
7.6.1.1.3 Digital I/O control for port (PIO0_2 - PIO0_4)

Type D registers.

Offset

Register	Offset
PIO0_2	8h
PIO0_3	Ch
PIO0_4	10h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

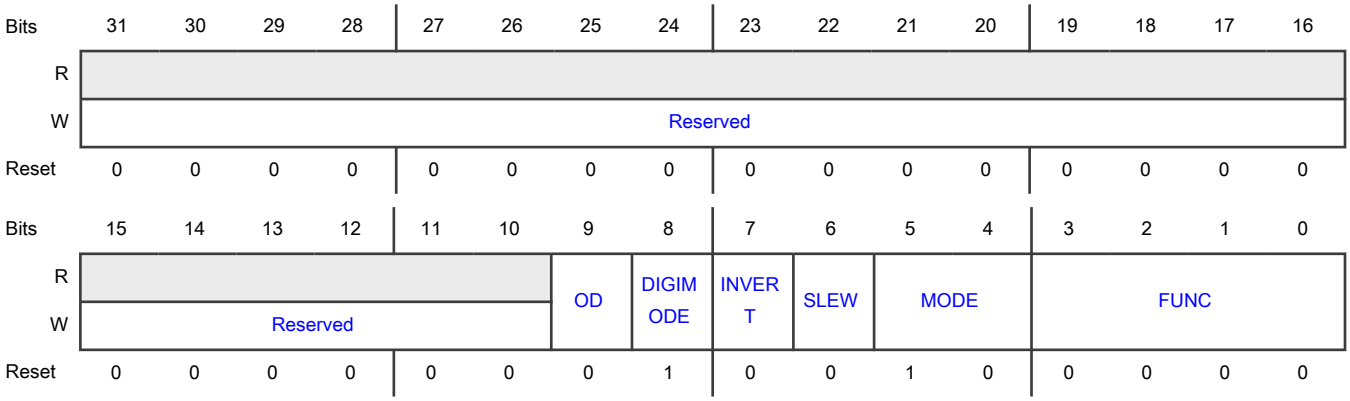
7.6.1.1.4 Digital I/O control for port (PIO0_5)

Type D registers.

Offset

Register	Offset
PIO0_5	14h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled).

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

7.6.1.1.5 Digital I/O control for port (PIO0_6)

Type D registers.

Offset

Register	Offset
PIO0_6	18h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
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							OD	DIGIMODE	INVERT	SLEW	MODE		FUNC			
W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9	Controls open-drain mode

Table continues on the next page...

Table continued from the previous page...

Field	Description
OD	0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

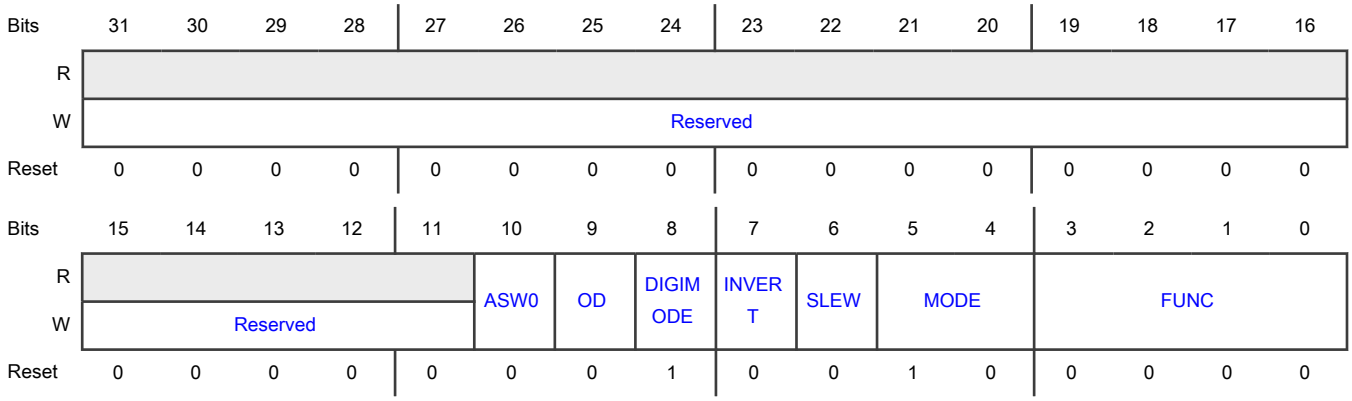
7.6.1.1.6 Analog/Digital I/O control for port (PIO0_7)

Type A registers.

Offset

Register	Offset
PIO0_7	1Ch

Diagram



Fields

Field	Description
31-11 —	Reserved Read value is undefined, only zero should be written.
10 ASW0	Analog switch input control Usable only if DIGIMODE = 0. To prevent digital noise, set FUNC = 0000 and MODE = 00. 0 - Analog switch is open. (disable) 1 - Analog switch is closed. (enable)
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

7.6.1.1.7 Analog/Digital I/O control for port (PIO0_8)

Type A registers.

Offset

Register	Offset
PIO0_8	20h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
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							OD	DIGIMODE	INVERT	SLEW	MODE		FUNC			
W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

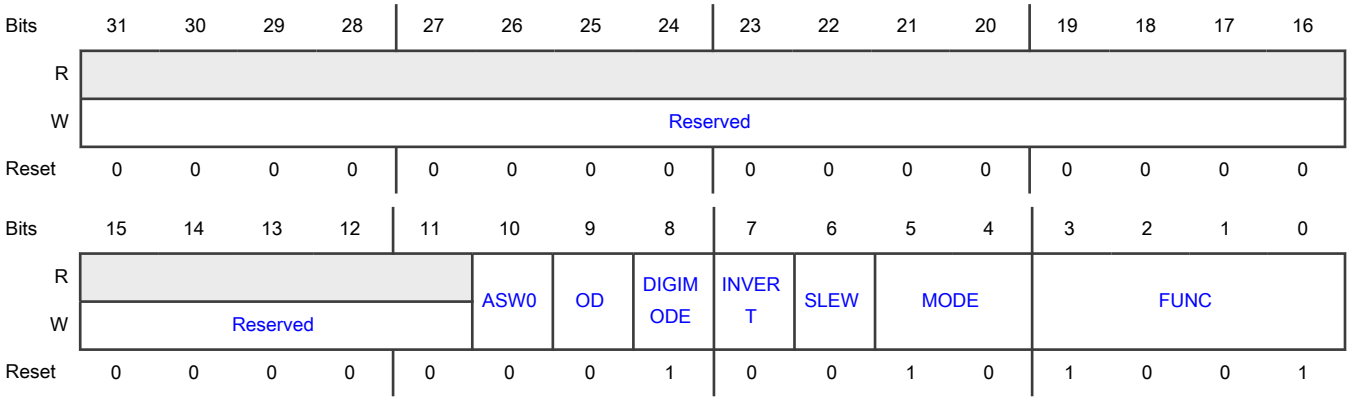
7.6.1.1.8 Analog/Digital I/O control for port (PIO0_9)

Type A registers.

Offset

Register	Offset
PIO0_9	24h

Diagram



Fields

Field	Description
31-11 —	Reserved Read value is undefined, only zero should be written.
10 ASW0	Analog switch input control Usable only if DIGIMODE = 0. To prevent digital noise, set FUNC = 0000 and MODE = 00. 0 - Analog switch is open. (disable) 1 - Analog switch is closed. (enable)
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

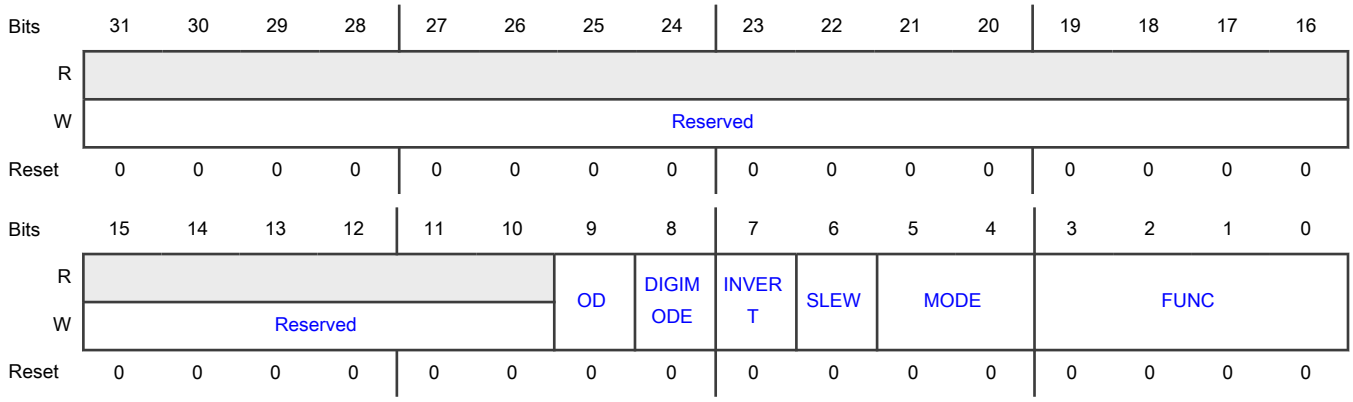
7.6.1.1.9 Analog/Digital I/O control for port (PIO0_10 - PIO0_12)

Type A registers.

Offset

Register	Offset
PIO0_10	28h
PIO0_11	2Ch
PIO0_12	30h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

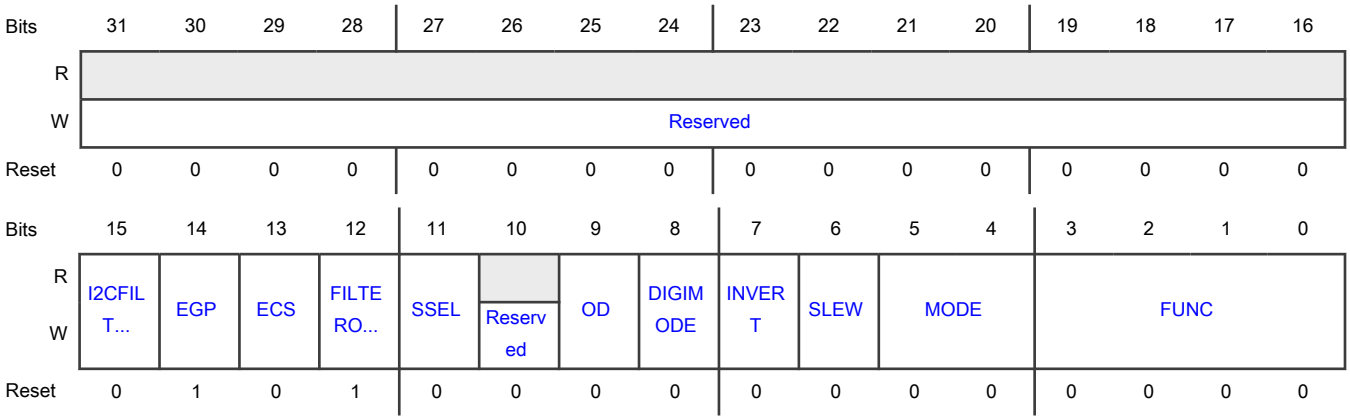
7.6.1.1.10 I2C control for port (PIO0_13 - PIO0_14)

Type registers.

Offset

Register	Offset
PIO0_13	34h
PIO0_14	38h

Diagram



Fields

Field	Description
31-16 —	Reserved Read value is undefined, only zero should be written.
15 I2CFILTER	Configures I2C features for standard mode, fast mode, and Fast Mode Plus operation and High-Speed mode operation. 0 - I2C 50 ns glitch filter enabled. Typically used for Standard-mode, Fast-mode and Fast-mode Plus I2C. 1 - I2C 10 ns glitch filter enabled. Typically used for High-speed mode I2C.
14 EGP	Switch between GPIO mode and I2C mode 0 - I2C mode 1 - GPIO mode.
13 ECS	Pull-up current source enable in I2C mode 0 - Disabled. IO is in open drain cell. 1 - Enabled. Pull resistor is conencted.
12 FILTEROFF	Controls input glitch filter 0 - Filter enabled. Noise pulses below approximately 3 ns are filtered out in GPIO mode (EGP = 1). In I2C mode (EGP = 0), noise pulses below approximately 10 ns or 50 ns are filtered out, depending on I2CFILTER bit field value. 1 - Filter disabled. No input filtering is done.

Table continues on the next page...

Table continued from the previous page...

Field	Description
11 SSEL	Supply Selection bit. 0 - 3V3 Signaling in I2C Mode. 1 - 1V8 Signaling in I2C Mode.
10 —	Reserved
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

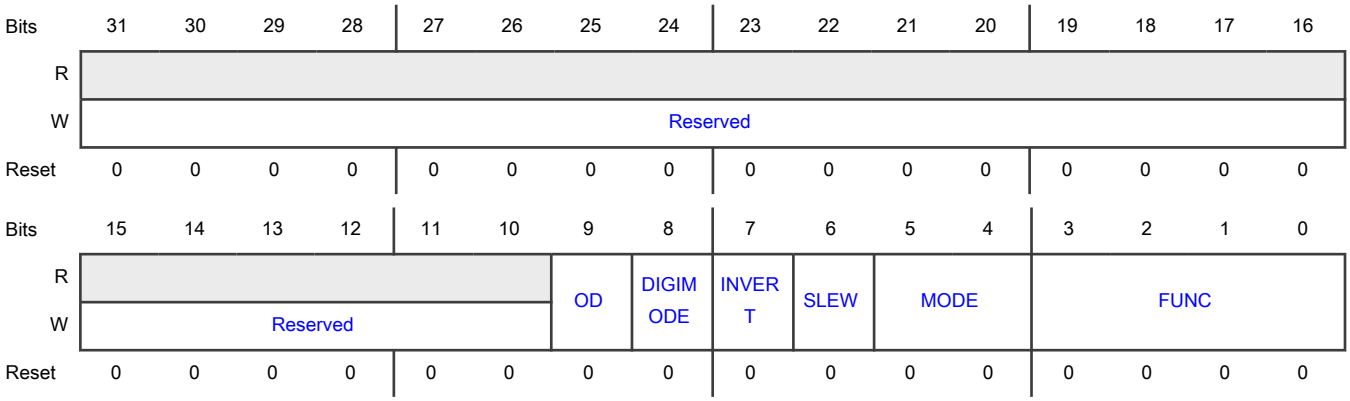
7.6.1.1.11 Analog/Digital I/O control for port (PIO0_15 - PIO0_16)

Type A registers.

Offset

Register	Offset
PIO0_15	3Ch
PIO0_16	40h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

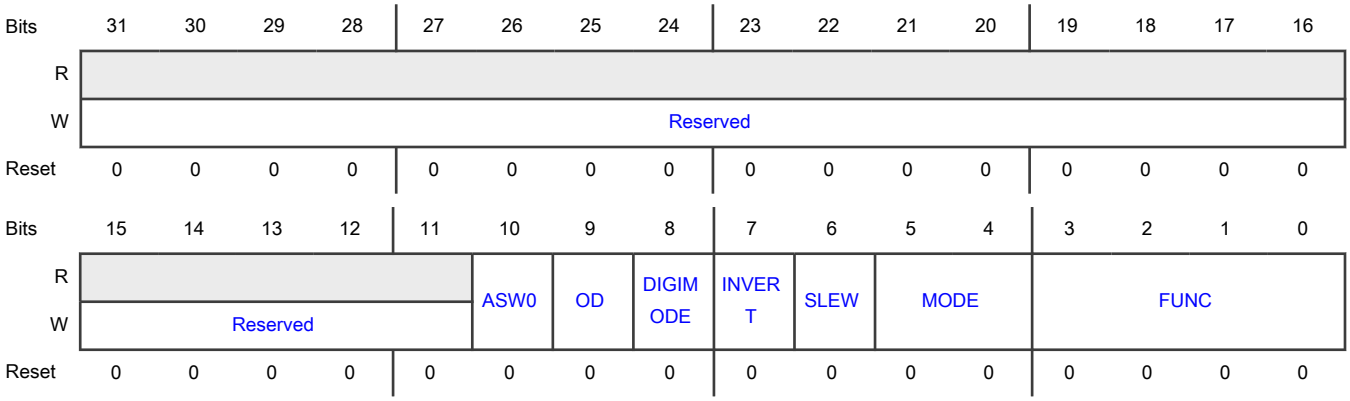
7.6.1.1.12 Analog/Digital I/O control for port (PIO0_17 - PIO0_18)

Type A registers.

Offset

Register	Offset
PIO0_17	44h
PIO0_18	48h

Diagram



Fields

Field	Description
31-11	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
10 ASW0	Analog switch input control Usable only if DIGIMODE = 0. To prevent digital noise, set FUNC = 0000 and MODE = 00. 0 - Analog switch is open. (disable) 1 - Analog switch is closed. (enable)
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

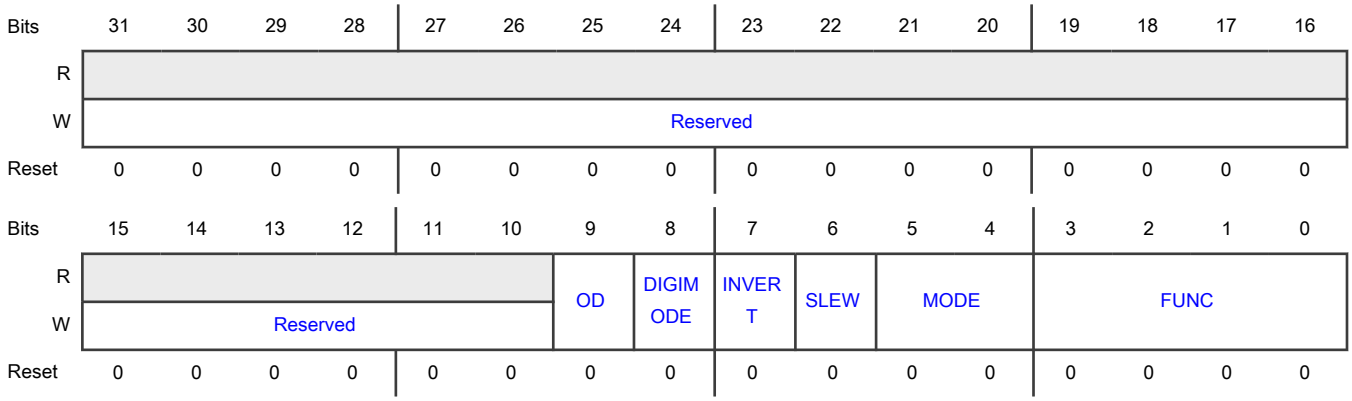
7.6.1.1.13 Digital I/O control for port (PIO0_19 - PIO0_22)

Type D registers.

Offset

Register	Offset
PIO0_19	4Ch
PIO0_20	50h
PIO0_21	54h
PIO0_22	58h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.

Table continues on the next page...

Table continued from the previous page...

Field	Description
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

7.6.1.1.14 Analog/Digital I/O control for port (PIO0_23 - PIO0_24)

Type A registers.

Offset

Register	Offset
PIO0_23	5Ch
PIO0_24	60h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
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						ASW0	OD	DIGIMODE	INVERT	SLEW	MODE		FUNC			
W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-11 —	Reserved Read value is undefined, only zero should be written.
10 ASW0	Analog switch input control Usable only if DIGIMODE = 0. To prevent digital noise, set FUNC = 0000 and MODE = 00. 0 - Analog switch is open. (disable) 1 - Analog switch is closed. (enable)
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

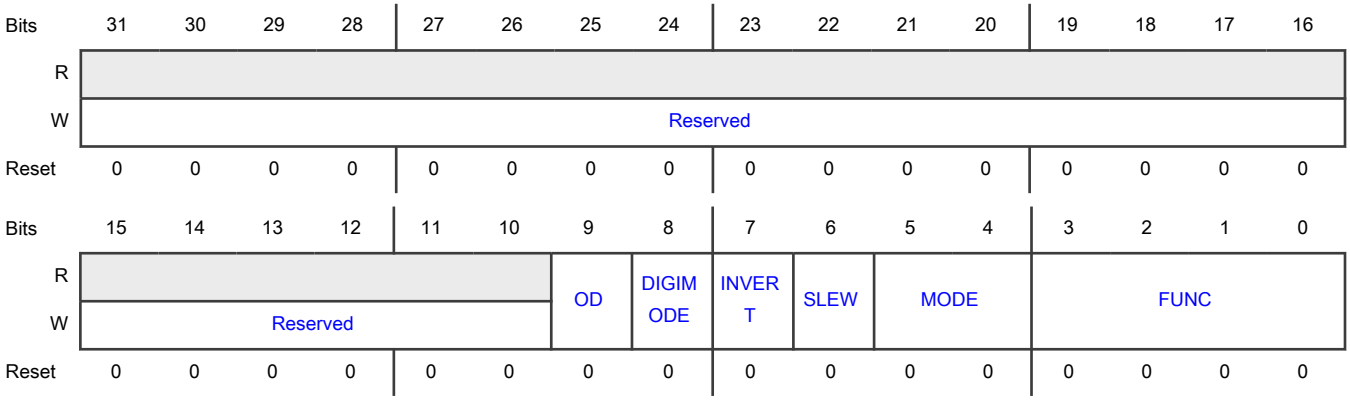
7.6.1.1.15 Digital I/O control for port (PIO0_25 - PIO0_26)

Type D registers.

Offset

Register	Offset
PIO0_25	64h
PIO0_26	68h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.

Table continues on the next page...

Table continued from the previous page...

Field	Description
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

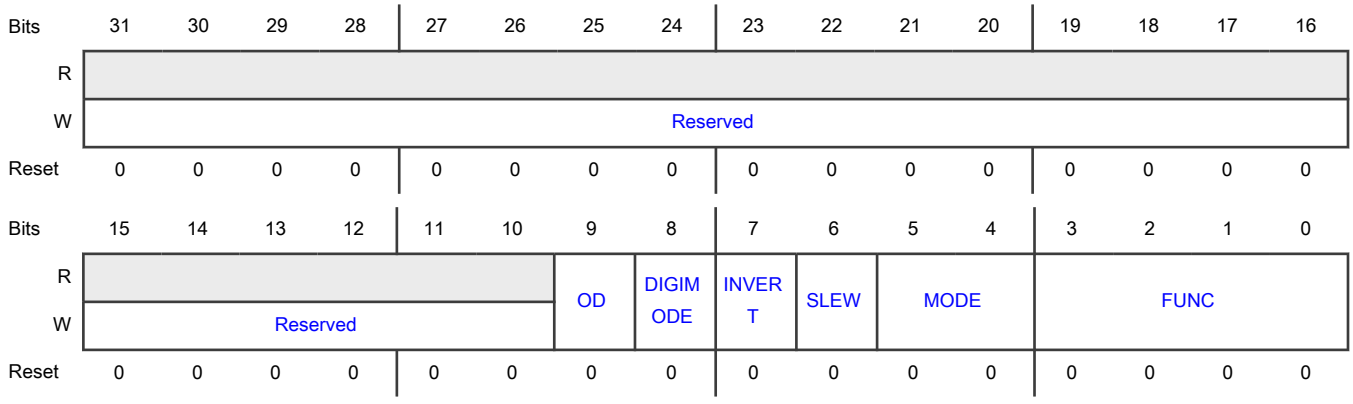
7.6.1.1.16 Analog/Digital I/O control for port (PIO0_27)

Type A registers.

Offset

Register	Offset
PIO0_27	6Ch

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

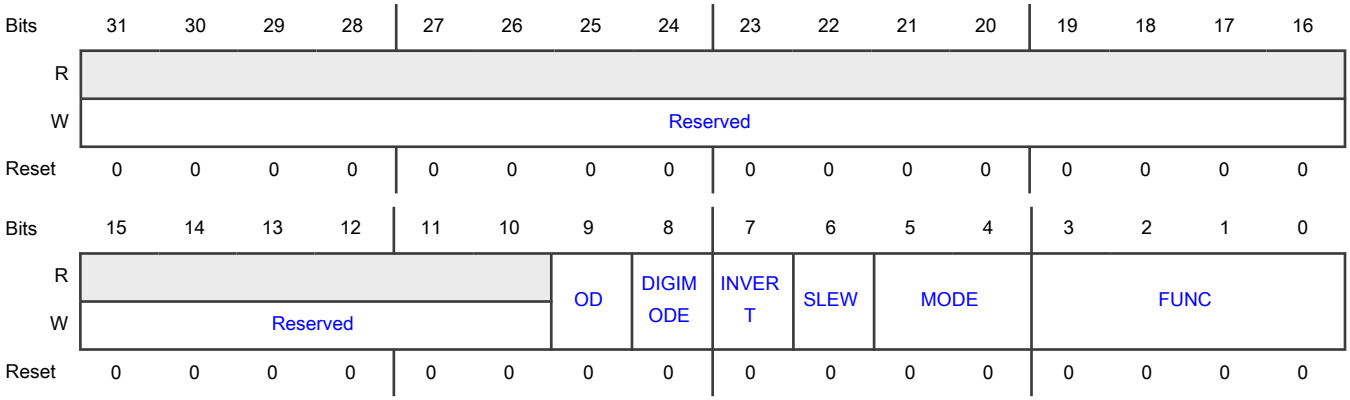
7.6.1.1.17 Digital I/O control for port (PIO0_28 - PIO0_30)

Type D registers.

Offset

Register	Offset
PIO0_28	70h
PIO0_29	74h
PIO0_30	78h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4	Mode select (on-chip pull-up/pull-down resistor control)

Table continues on the next page...

Table continued from the previous page...

Field	Description
MODE	These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

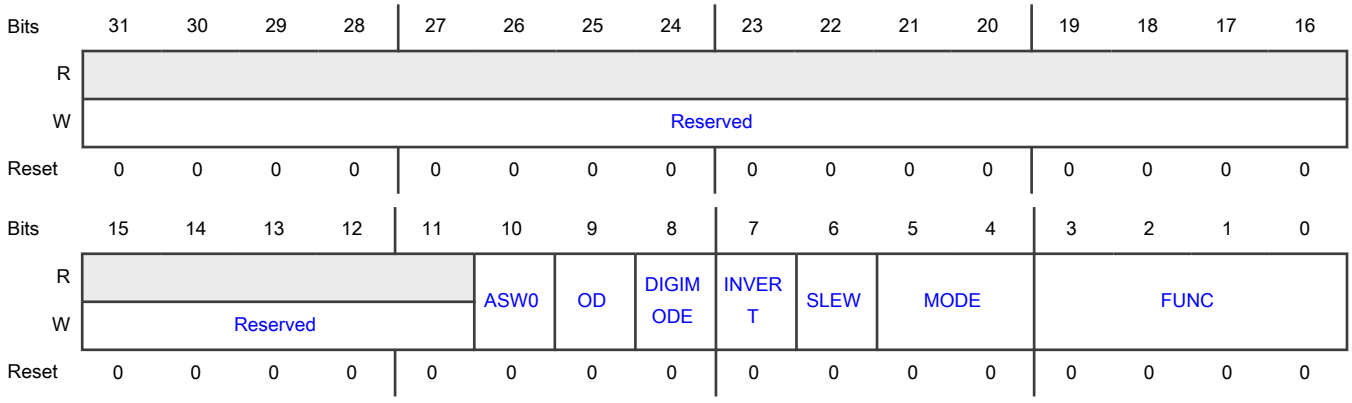
7.6.1.1.18 Analog/Digital I/O control for port (PIO0_31)

Type A registers.

Offset

Register	Offset
PIO0_31	7Ch

Diagram



Fields

Field	Description
31-11 —	Reserved Read value is undefined, only zero should be written.
10 ASW0	Analog switch input control Usable only if DIGIMODE = 0. To prevent digital noise, set FUNC = 0000 and MODE = 00. 0 - Analog switch is open. (disable) 1 - Analog switch is closed. (enable)
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

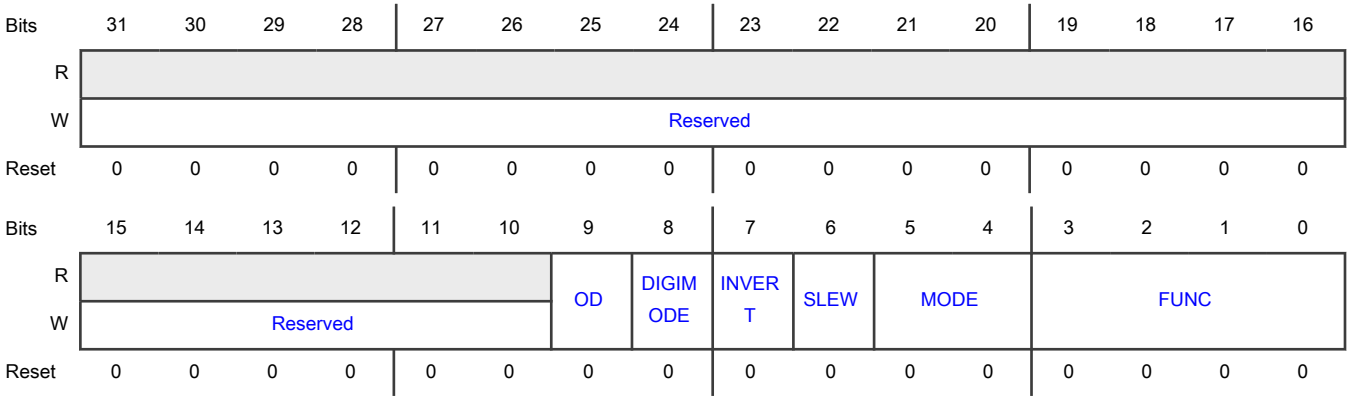
7.6.1.1.19 Digital I/O control for port (PIO1_0 - PIO1_4)

Type D registers.

Offset

Register	Offset
PIO1_0	80h
PIO1_1	84h
PIO1_2	88h
PIO1_3	8Ch
PIO1_4	90h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.

Table continues on the next page...

Table continued from the previous page...

Field	Description
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

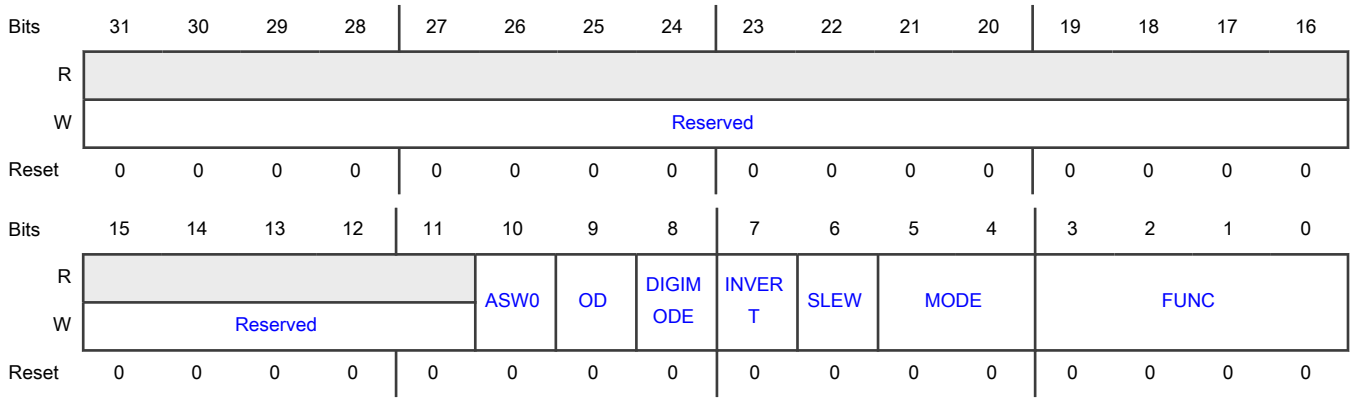
7.6.1.1.20 Analog/Digital I/O control for port (PIO1_5)

Type A registers.

Offset

Register	Offset
PIO1_5	94h

Diagram



Fields

Field	Description
31-11 —	Reserved Read value is undefined, only zero should be written.
10 ASW0	Analog switch input control Usable only if DIGIMODE = 0. To prevent digital noise, set FUNC = 0000 and MODE = 00. 0 - Analog switch is open. (disable) 1 - Analog switch is closed. (enable)
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

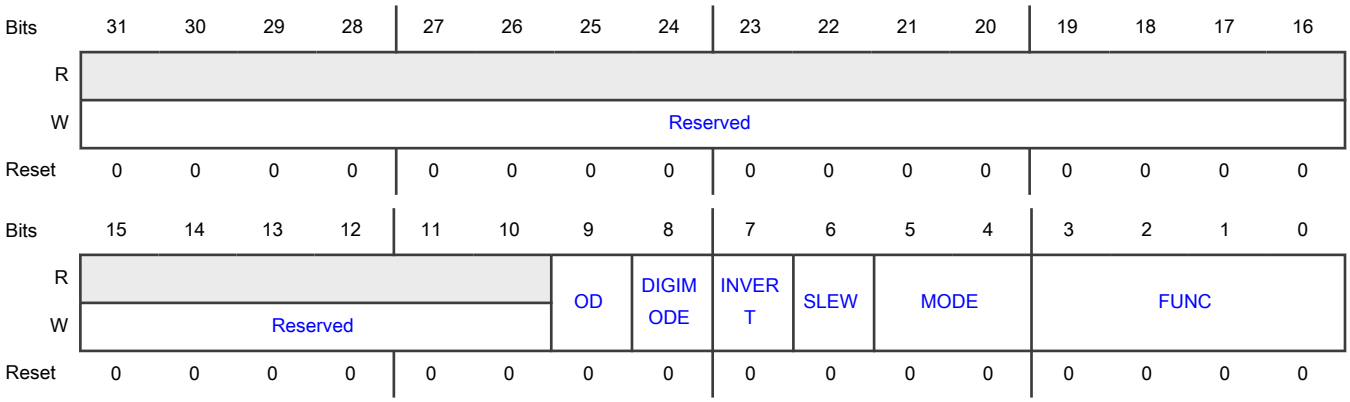
7.6.1.1.21 Digital I/O control for port (PIO1_6)

Type D registers.

Offset

Register	Offset
PIO1_6	98h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4	Mode select (on-chip pull-up/pull-down resistor control)

Table continues on the next page...

Table continued from the previous page...

Field	Description
MODE	These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

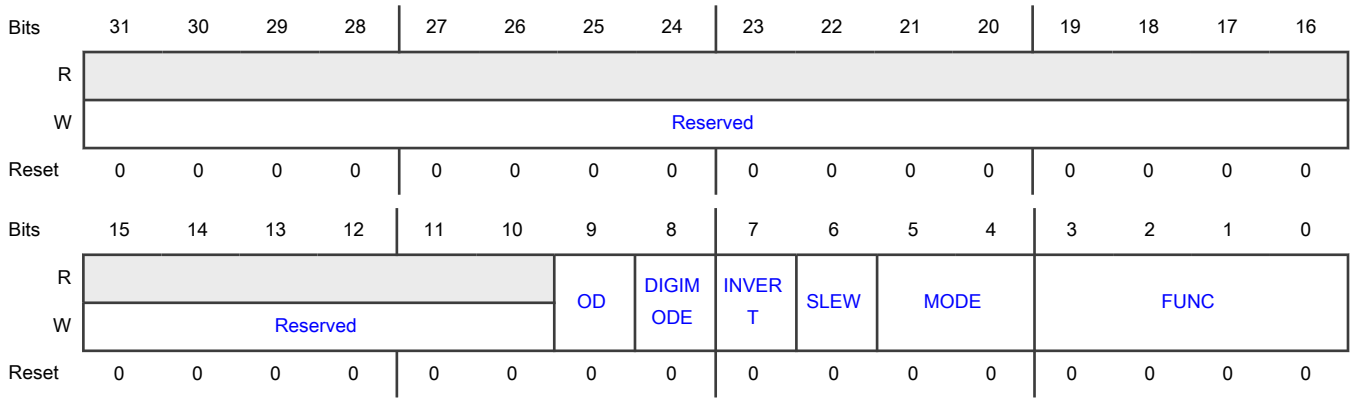
7.6.1.1.22 Analog/Digital I/O control for port (PIO1_7)

Type A registers.

Offset

Register	Offset
PIO1_7	9Ch

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

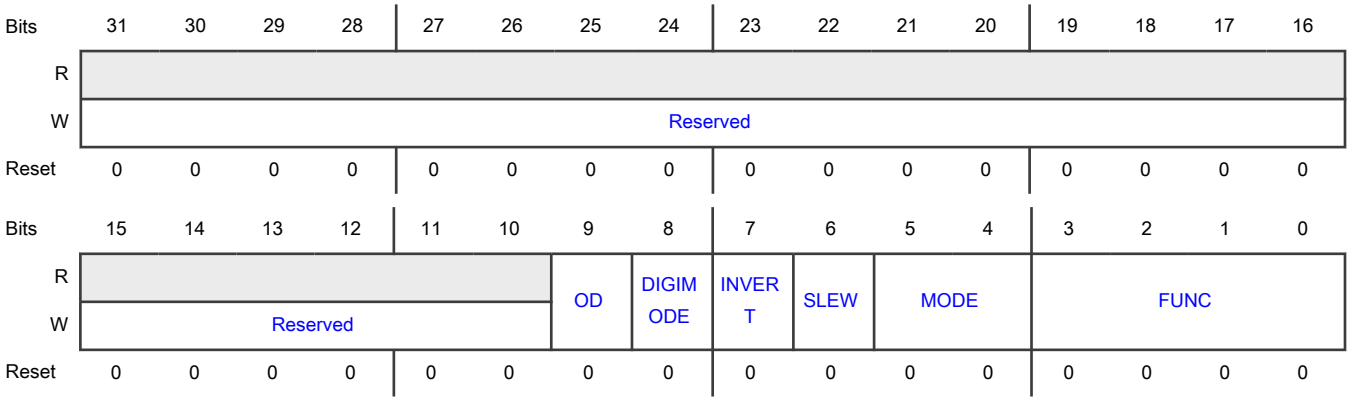
7.6.1.1.23 Digital I/O control for port (PIO1_8)

Type D registers.

Offset

Register	Offset
PIO1_8	A0h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled).

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

7.6.1.1.24 Analog/Digital I/O control for port (PIO1_9 - PIO1_10)

Type A registers.

Offset

Register	Offset
PIO1_9	A4h
PIO1_10	A8h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
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						ASW0	OD	DIGIMODE	INVERT	SLEW	MODE		FUNC			
W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-11 —	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
10 ASW0	Analog switch input control Usable only if DIGIMODE = 0. To prevent digital noise, set FUNC = 0000 and MODE = 00. 0 - Analog switch is open. (disable) 1 - Analog switch is closed. (enable)
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

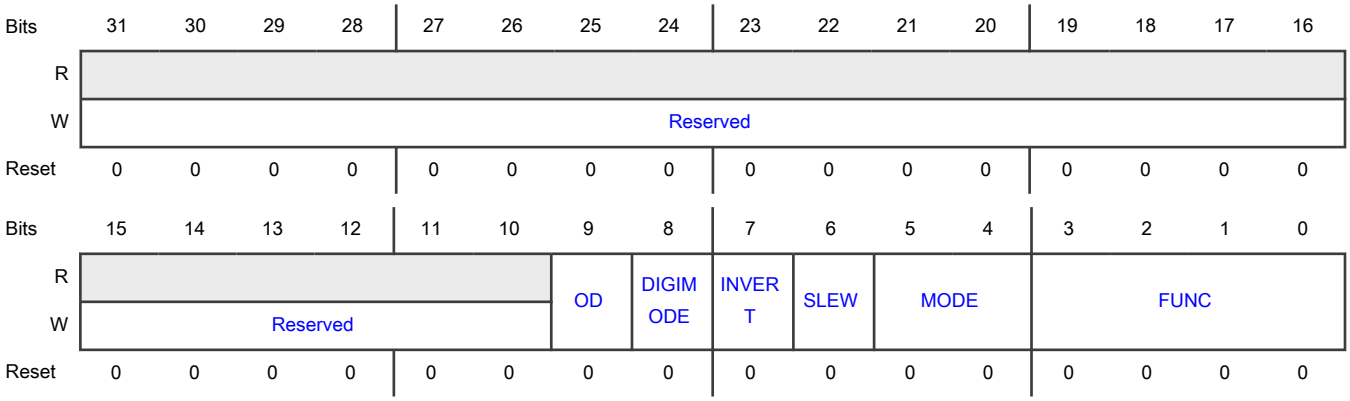
7.6.1.1.25 Digital I/O control for port (PIO1_11)

Type D registers.

Offset

Register	Offset
PIO1_11	ACh

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled).

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

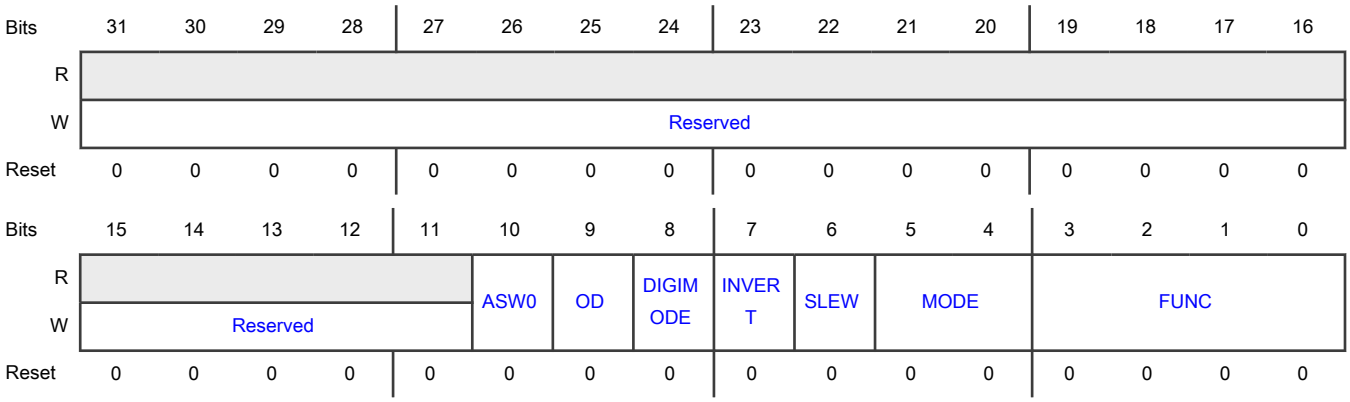
7.6.1.1.26 Analog/Digital I/O control for port (PIO1_12 - PIO1_14)

Type A registers.

Offset

Register	Offset
PIO1_12	B0h
PIO1_13	B4h
PIO1_14	B8h

Diagram



Fields

Field	Description
31-11	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
10 ASW0	Analog switch input control Usable only if DIGIMODE = 0. To prevent digital noise, set FUNC = 0000 and MODE = 00. 0 - Analog switch is open. (disable) 1 - Analog switch is closed. (enable)
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

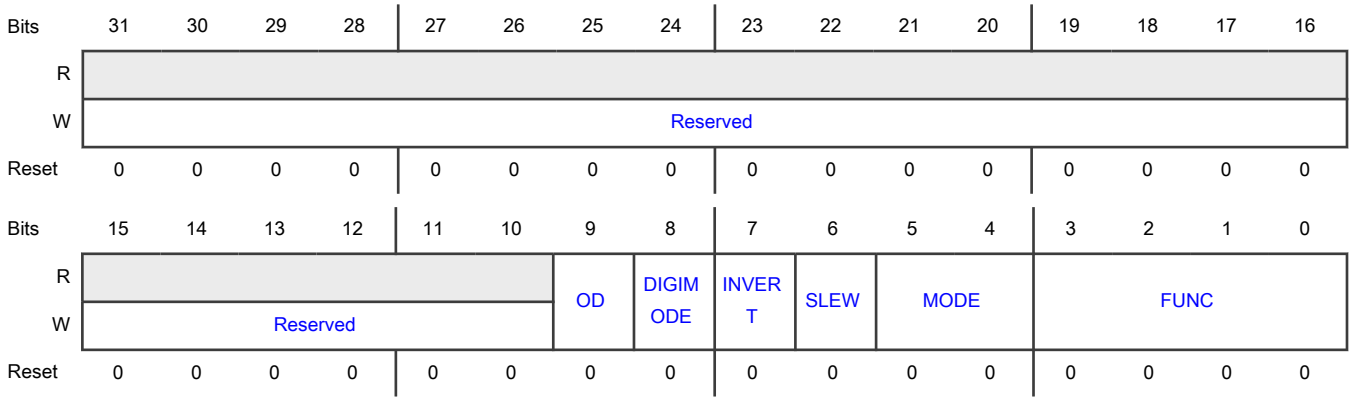
7.6.1.1.27 Digital I/O control for port (PIO1_15 - PIO1_18)

Type D registers.

Offset

Register	Offset
PIO1_15	BCh
PIO1_16	C0h
PIO1_17	C4h
PIO1_18	C8h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.

Table continues on the next page...

Table continued from the previous page...

Field	Description
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

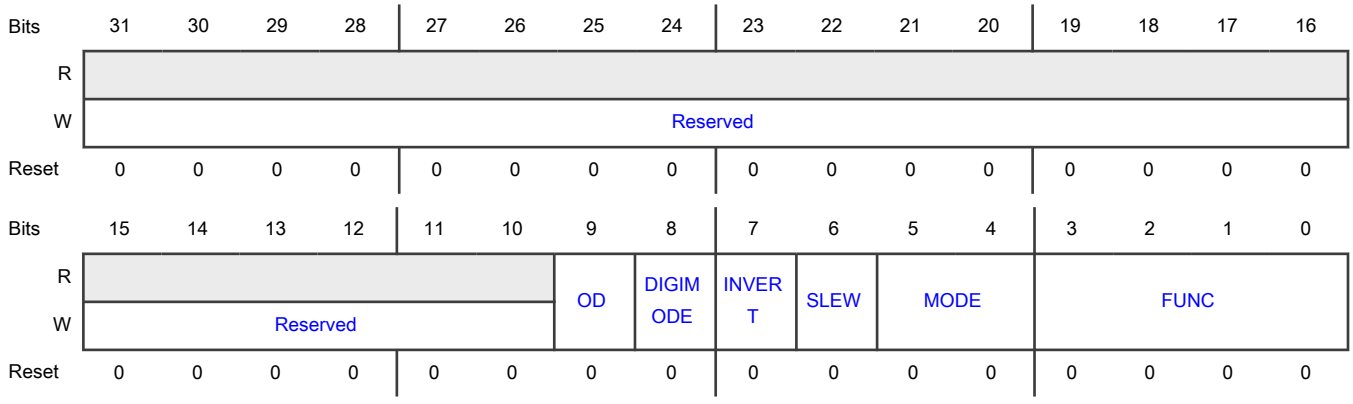
7.6.1.1.28 Analog/Digital I/O control for port (PIO1_19)

Type A registers.

Offset

Register	Offset
PIO1_19	CCh

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

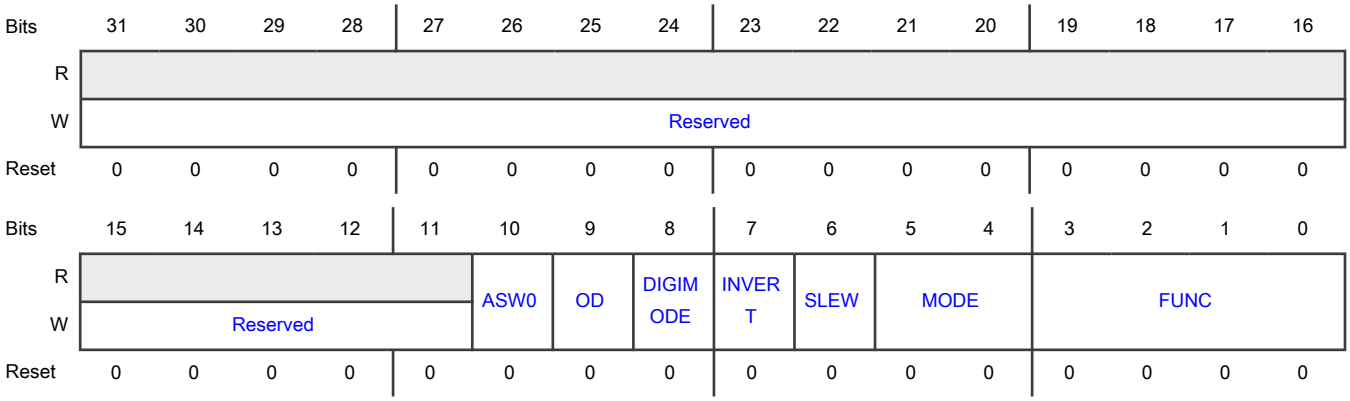
7.6.1.1.29 Analog/Digital I/O control for port (PIO1_20)

Type A registers.

Offset

Register	Offset
PIO1_20	D0h

Diagram



Fields

Field	Description
31-11 —	Reserved Read value is undefined, only zero should be written.
10 ASW0	Analog switch input control Usable only if DIGIMODE = 0. To prevent digital noise, set FUNC = 0000 and MODE = 00. 0 - Analog switch is open. (disable) 1 - Analog switch is closed. (enable)
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously.

Table continues on the next page...

Table continued from the previous page...

Field	Description
SLEW	1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

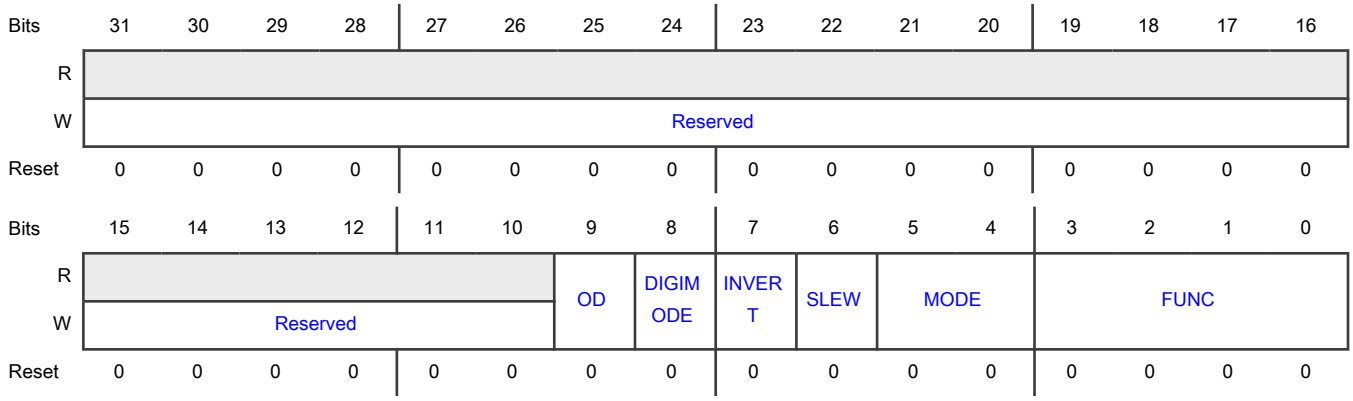
7.6.1.1.30 Digital I/O control for port (PIO1_21)

Type D registers.

Offset

Register	Offset
PIO1_21	D4h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

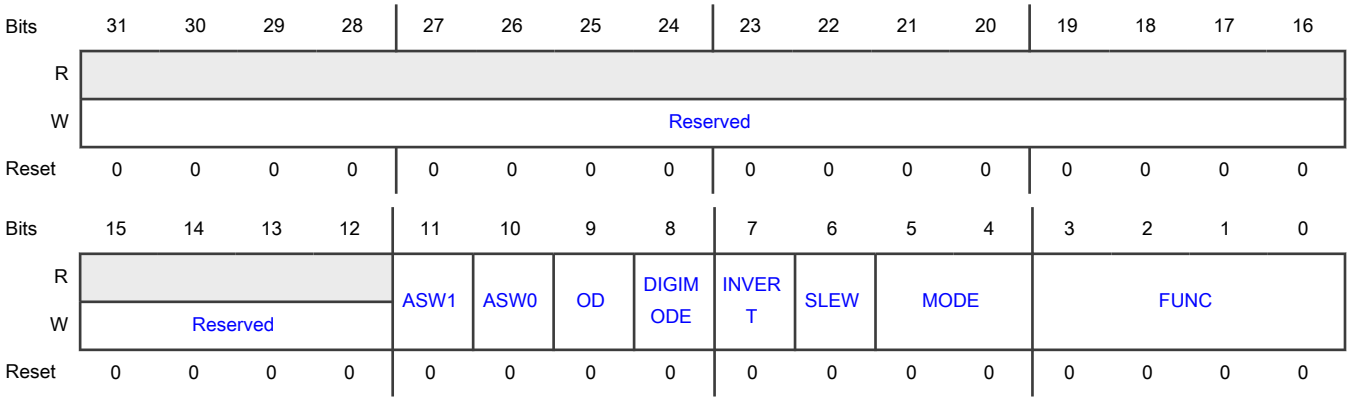
7.6.1.1.31 Analog/Digital I/O control for port (PIO1_22)

Type A registers.

Offset

Register	Offset
PIO1_22	D8h

Diagram



Fields

Field	Description
31-12 —	Reserved Read value is undefined, only zero should be written.
11 ASW1	Analog switch input control Usable only if DIGIMODE = 0. To prevent digital noise, set FUNC = 0000 and MODE = 00. 0 - Analog switch is open. (disable) 1 - Analog switch is closed. (enable)
10 ASW0	Analog switch input control Usable only if DIGIMODE = 0. To prevent digital noise, set FUNC = 0000 and MODE = 00. 0 - Analog switch is open. (disable) 1 - Analog switch is closed. (enable)
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

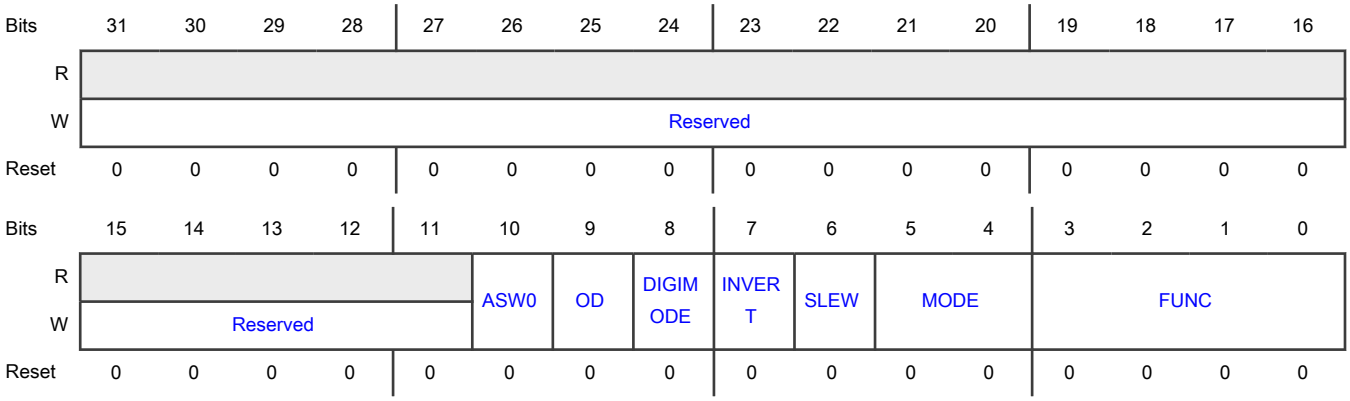
7.6.1.1.32 Analog/Digital I/O control for port (PIO1_23 - PIO1_24)

Type A registers.

Offset

Register	Offset
PIO1_23	DCh
PIO1_24	E0h

Diagram



Fields

Field	Description
31-11 —	Reserved Read value is undefined, only zero should be written.
10 ASW0	Analog switch input control Usable only if DIGIMODE = 0. To prevent digital noise, set FUNC = 0000 and MODE = 00. 0 - Analog switch is open. (disable) 1 - Analog switch is closed. (enable)
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

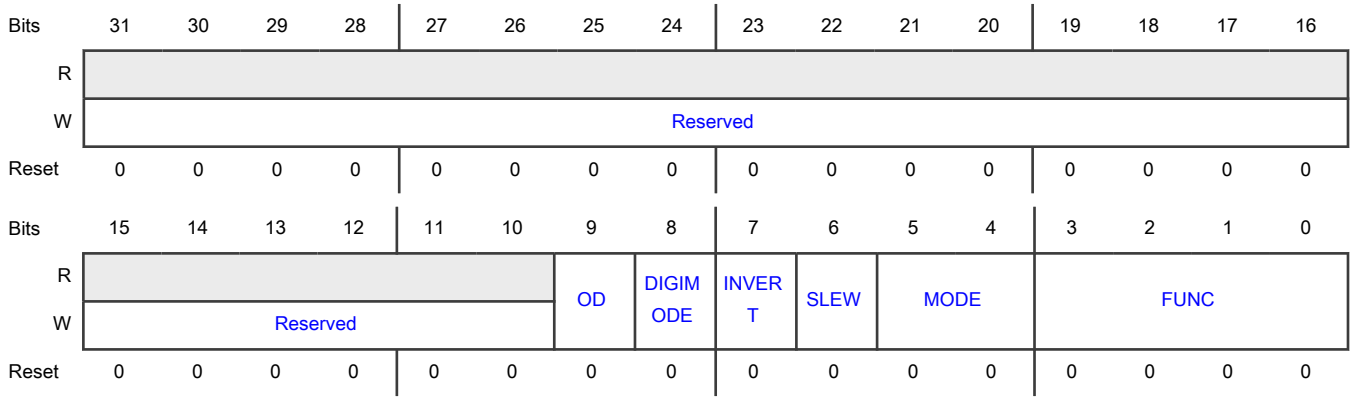
7.6.1.1.33 Digital I/O control for port (PIO1_25 - PIO1_31)

Type D registers.

Offset

Register	Offset
PIO1_25	E4h
PIO1_26	E8h
PIO1_27	ECh
PIO1_28	F0h
PIO1_29	F4h
PIO1_30	F8h
PIO1_31	FCh

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

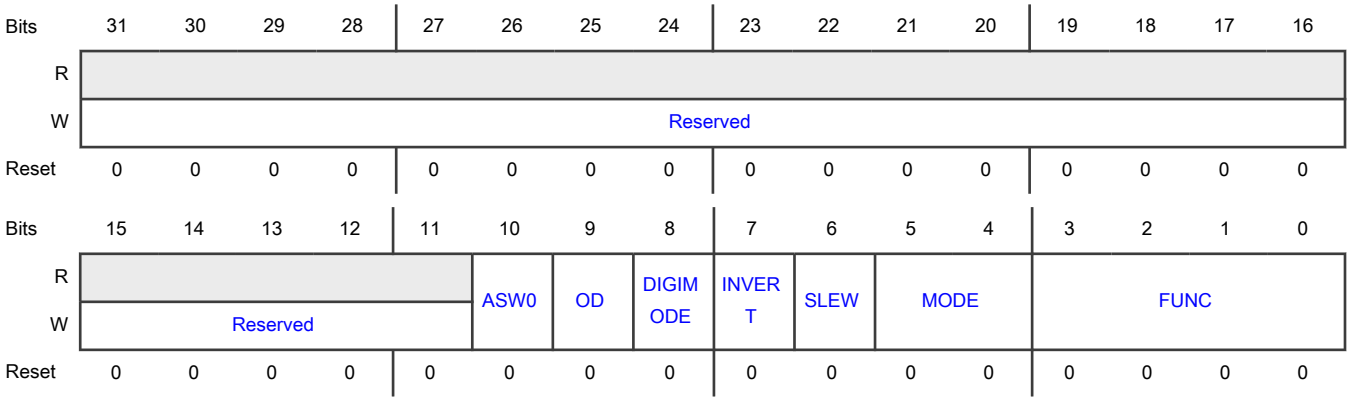
7.6.1.1.34 Analog/Digital I/O control for port (PIO2_0)

Type A registers.

Offset

Register	Offset
PIO2_0	100h

Diagram



Fields

Field	Description
31-11 —	Reserved Read value is undefined, only zero should be written.
10 ASW0	Analog switch input control Usable only if DIGIMODE = 0. To prevent digital noise, set FUNC = 0000 and MODE = 00. 0 - Analog switch is open. (disable) 1 - Analog switch is closed. (enable)
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously.

Table continues on the next page...

Table continued from the previous page...

Field	Description
SLEW	1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

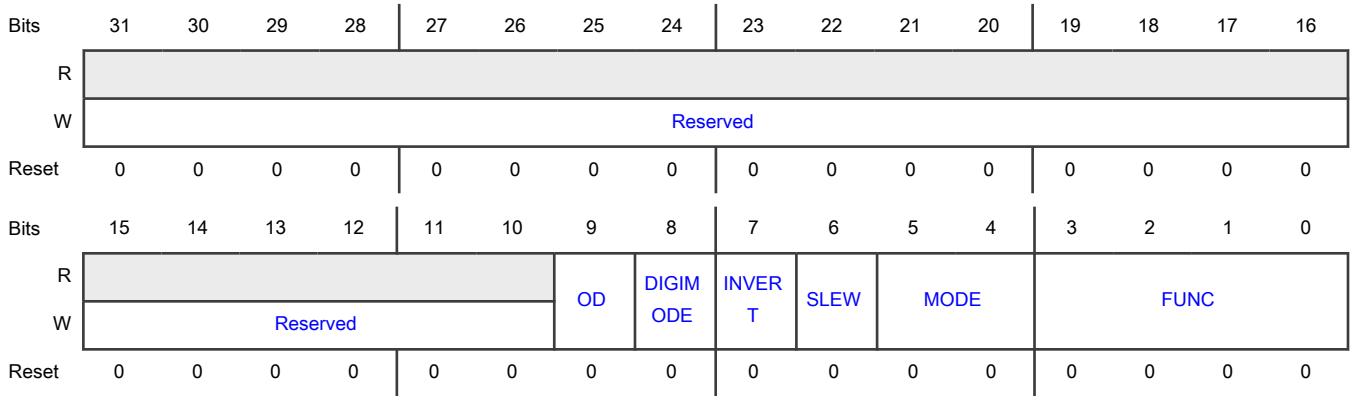
7.6.1.1.35 Analog/Digital I/O control for port (PIO2_1)

Type A registers.

Offset

Register	Offset
PIO2_1	104h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

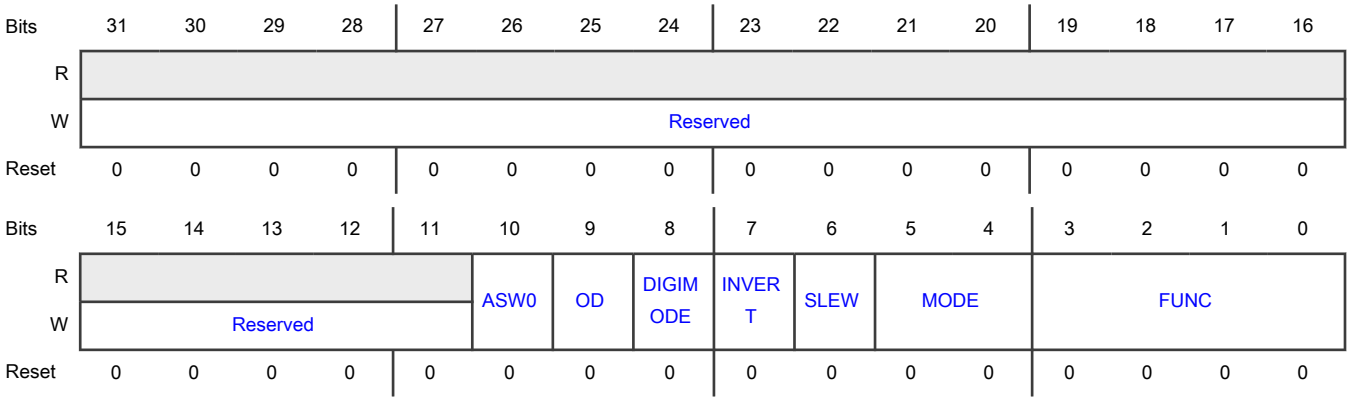
7.6.1.1.36 Analog/Digital I/O control for port (PIO2_2)

Type A registers.

Offset

Register	Offset
PIO2_2	108h

Diagram



Fields

Field	Description
31-11 —	Reserved Read value is undefined, only zero should be written.
10 ASW0	Analog switch input control Usable only if DIGIMODE = 0. To prevent digital noise, set FUNC = 0000 and MODE = 00. 0 - Analog switch is open. (disable) 1 - Analog switch is closed. (enable)
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously.

Table continues on the next page...

Table continued from the previous page...

Field	Description
SLEW	1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

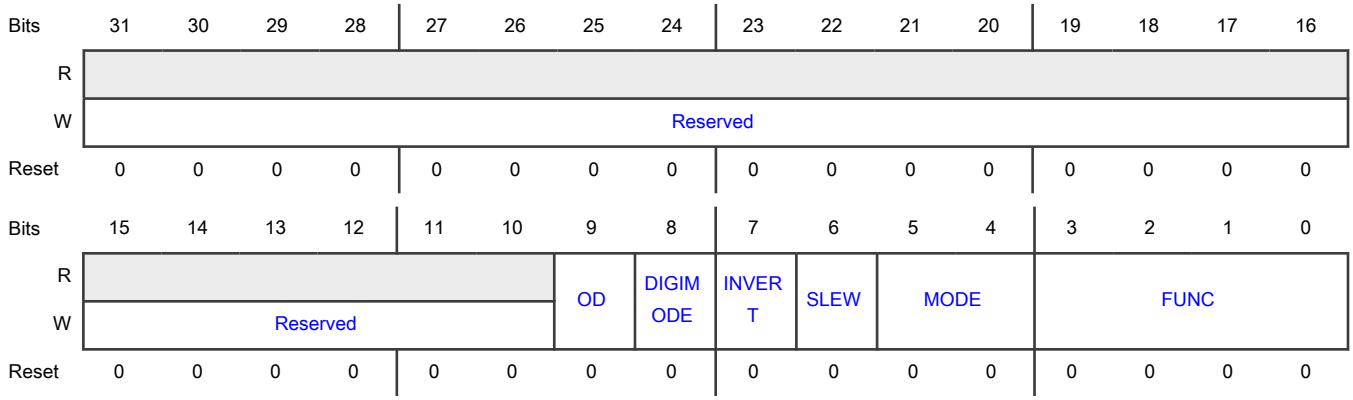
7.6.1.1.37 Digital I/O control for port (PIO2_3)

Type D registers.

Offset

Register	Offset
PIO2_3	10Ch

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

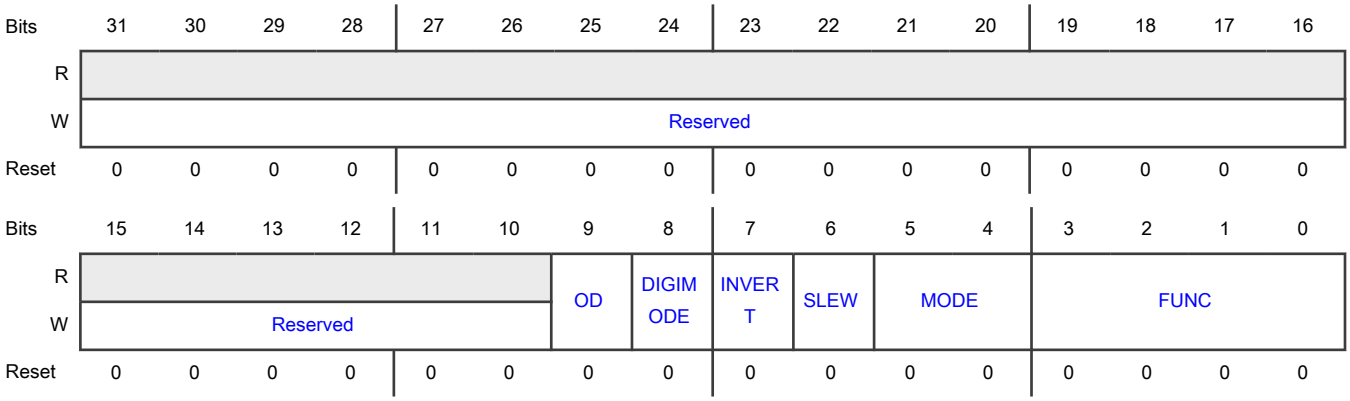
7.6.1.1.38 Analog/Digital I/O control for port (PIO2_4)

Type A registers.

Offset

Register	Offset
PIO2_4	110h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled).

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

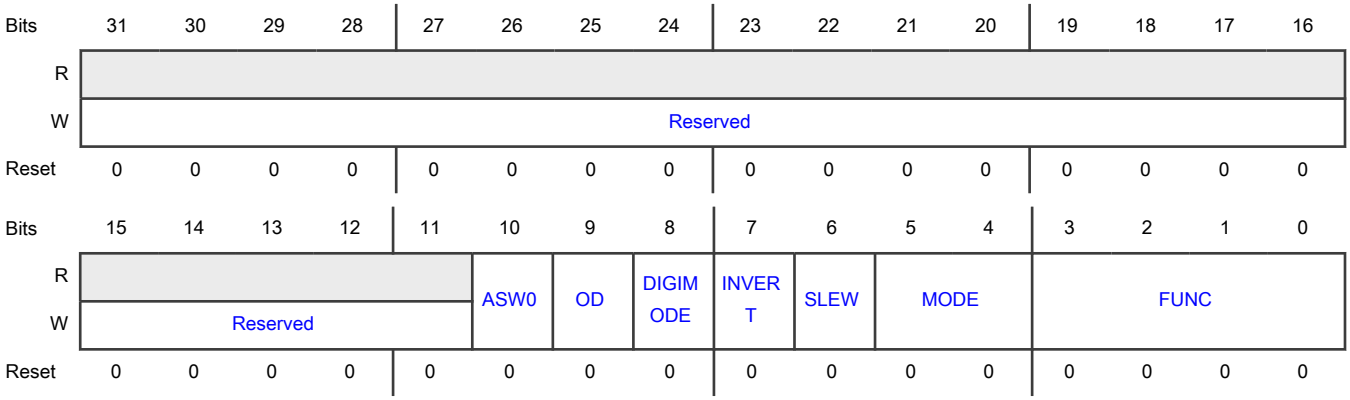
7.6.1.1.39 Analog/Digital I/O control for port (PIO2_5 - PIO2_6)

Type A registers.

Offset

Register	Offset
PIO2_5	114h
PIO2_6	118h

Diagram



Fields

Field	Description
31-11 —	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
10 ASW0	Analog switch input control Usable only if DIGIMODE = 0. To prevent digital noise, set FUNC = 0000 and MODE = 00. 0 - Analog switch is open. (disable) 1 - Analog switch is closed. (enable)
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

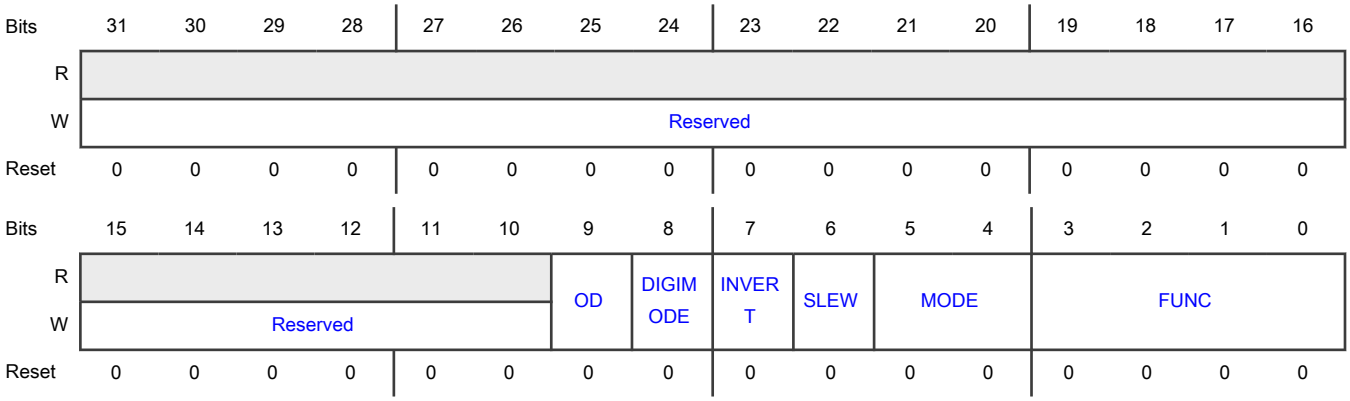
7.6.1.1.40 Digital I/O control for port (PIO2_7)

Type D registers.

Offset

Register	Offset
PIO2_7	11Ch

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled).

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

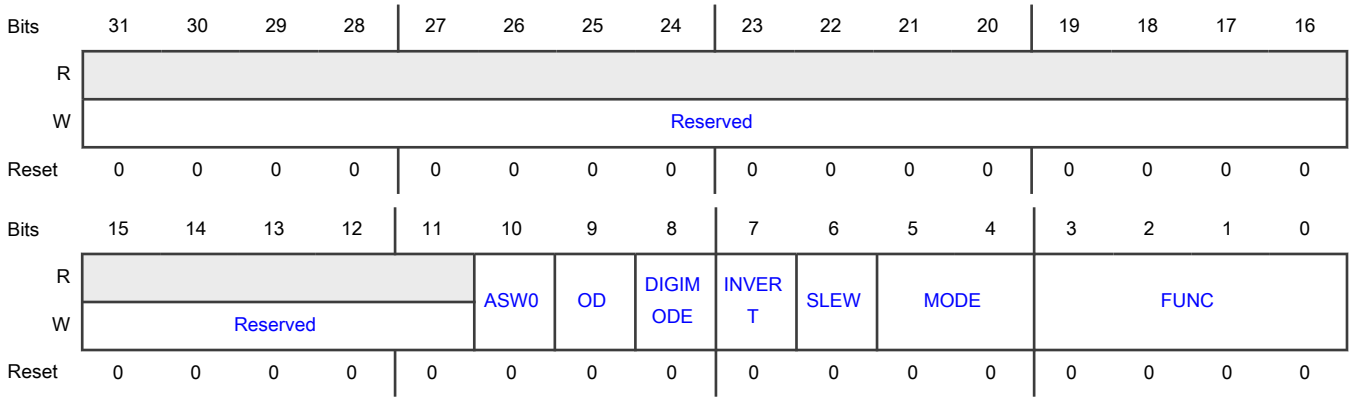
7.6.1.1.41 Analog/Digital I/O control for port (PIO2_8)

Type A registers.

Offset

Register	Offset
PIO2_8	120h

Diagram



Fields

Field	Description
31-11 —	Reserved Read value is undefined, only zero should be written.
10	Analog switch input control

Table continues on the next page...

Table continued from the previous page...

Field	Description
ASW0	Usable only if DIGIMODE = 0. To prevent digital noise, set FUNC = 0000 and MODE = 00. 0 - Analog switch is open. (disable) 1 - Analog switch is closed. (enable)
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

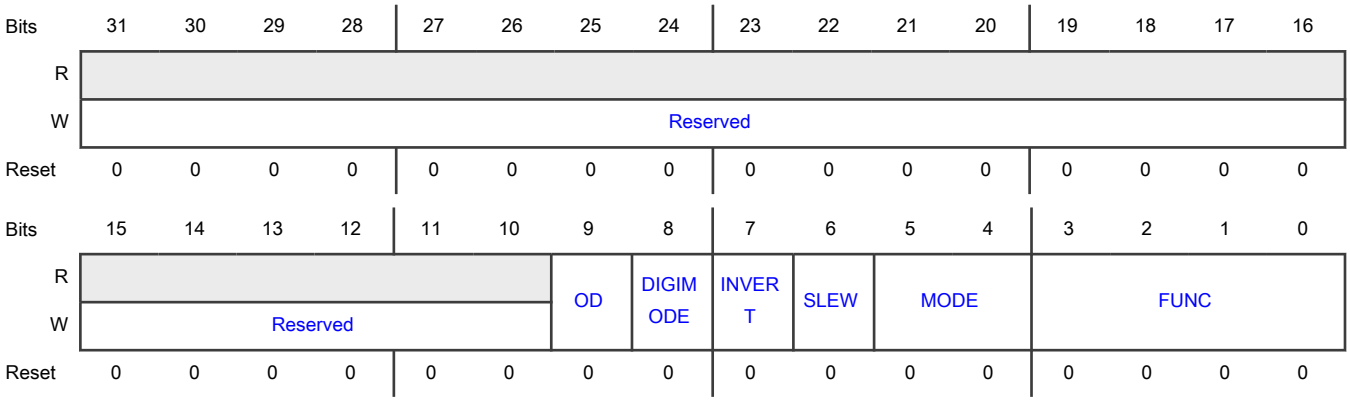
7.6.1.1.42 Digital I/O control for port (PIO2_9 - PIO2_10)

Type D registers.

Offset

Register	Offset
PIO2_9	124h
PIO2_10	128h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

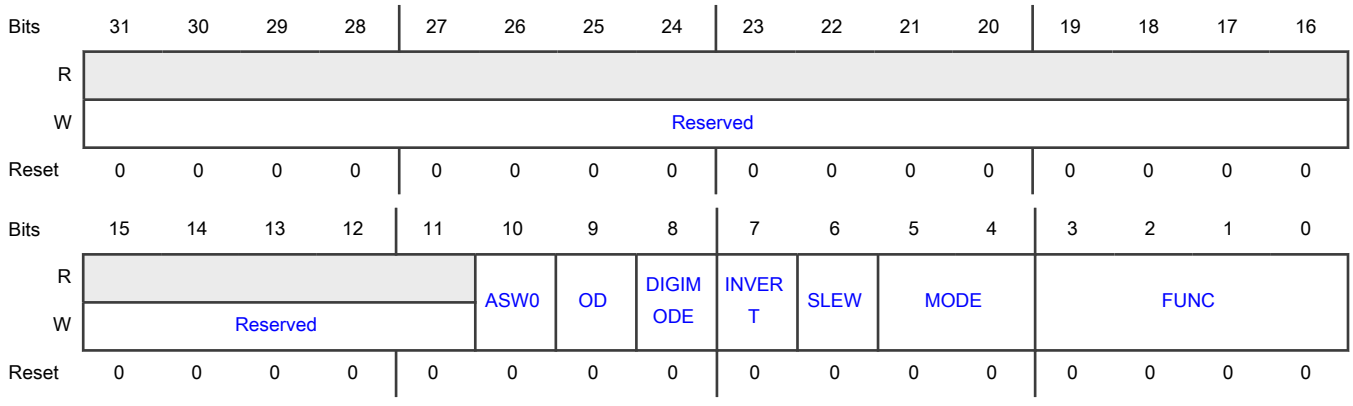
7.6.1.1.43 Analog/Digital I/O control for port (PIO2_11 - PIO2_14)

Type A registers.

Offset

Register	Offset
PIO2_11	12Ch
PIO2_12	130h
PIO2_13	134h
PIO2_14	138h

Diagram



Fields

Field	Description
31-11 —	Reserved Read value is undefined, only zero should be written.
10 ASW0	Analog switch input control Usable only if DIGIMODE = 0. To prevent digital noise, set FUNC = 0000 and MODE = 00. 0 - Analog switch is open. (disable) 1 - Analog switch is closed. (enable)
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

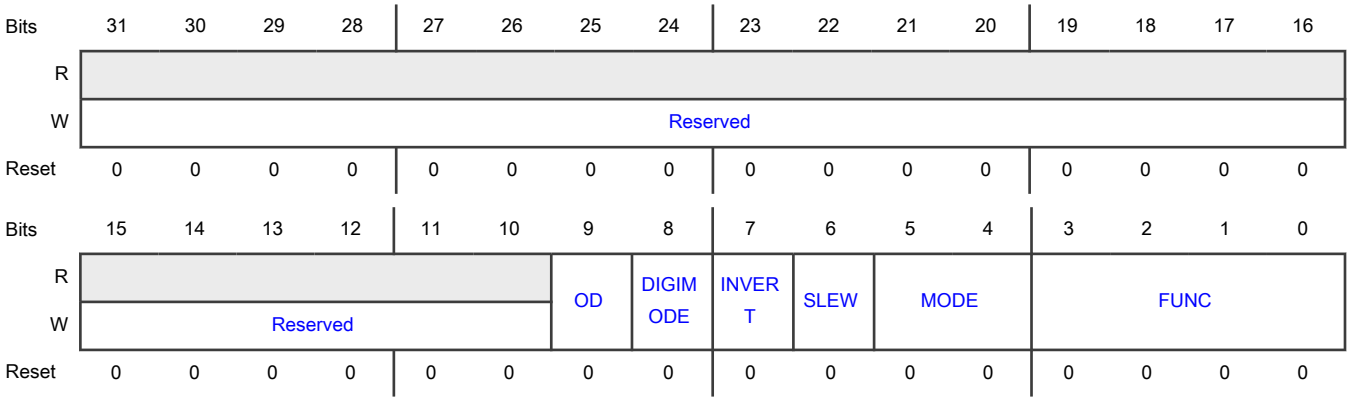
7.6.1.1.44 Digital I/O control for port (PIO2_15 - PIO2_18)

Type D registers.

Offset

Register	Offset
PIO2_15	13Ch
PIO2_16	140h
PIO2_17	144h
PIO2_18	148h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6	Driver slew rate

Table continues on the next page...

Table continued from the previous page...

Field	Description
SLEW	0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

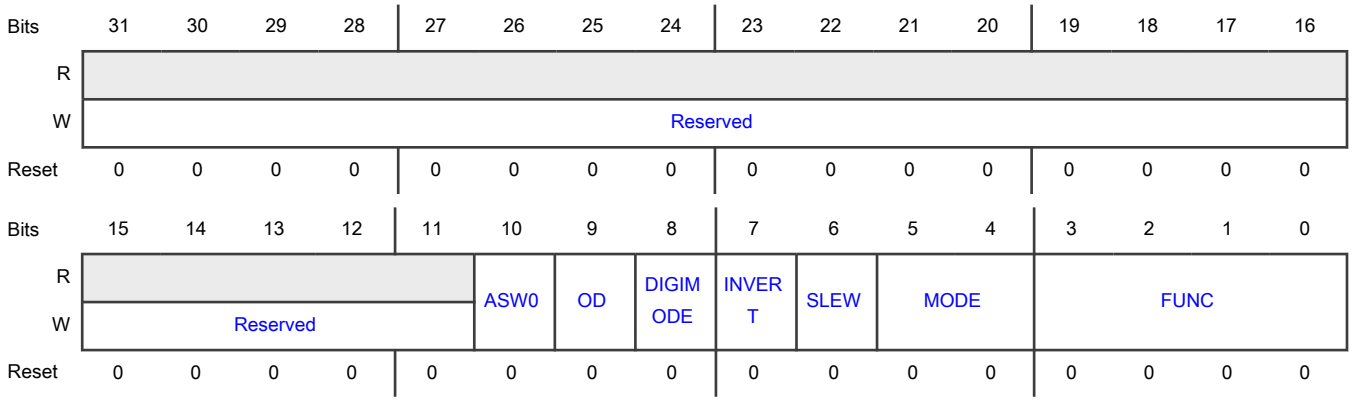
7.6.1.1.45 Analog/Digital I/O control for port (PIO2_19 - PIO2_20)

Type A registers.

Offset

Register	Offset
PIO2_19	14Ch
PIO2_20	150h

Diagram



Fields

Field	Description
31-11 —	Reserved Read value is undefined, only zero should be written.
10 ASW0	Analog switch input control Usable only if DIGIMODE = 0. To prevent digital noise, set FUNC = 0000 and MODE = 00. 0 - Analog switch is open. (disable) 1 - Analog switch is closed. (enable)
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

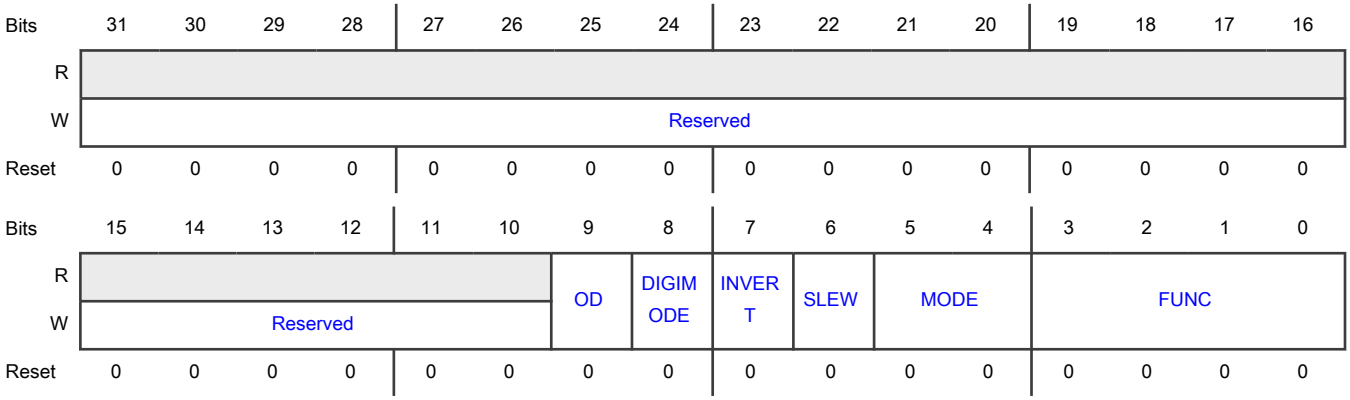
7.6.1.1.46 Digital I/O control for port (PIO2_21 - PIO2_22)

Type D registers.

Offset

Register	Offset
PIO2_21	154h
PIO2_22	158h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.

Table continues on the next page...

Table continued from the previous page...

Field	Description
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

7.6.1.1.47 Analog/Digital I/O control for port (PIO2_23 - PIO2_24)

Type A registers.

Offset

Register	Offset
PIO2_23	15Ch
PIO2_24	160h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
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						ASW0	OD	DIGIMODE	INVERT	SLEW	MODE		FUNC			
W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-11 —	Reserved Read value is undefined, only zero should be written.
10 ASW0	Analog switch input control Usable only if DIGIMODE = 0. To prevent digital noise, set FUNC = 0000 and MODE = 00. 0 - Analog switch is open. (disable) 1 - Analog switch is closed. (enable)
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

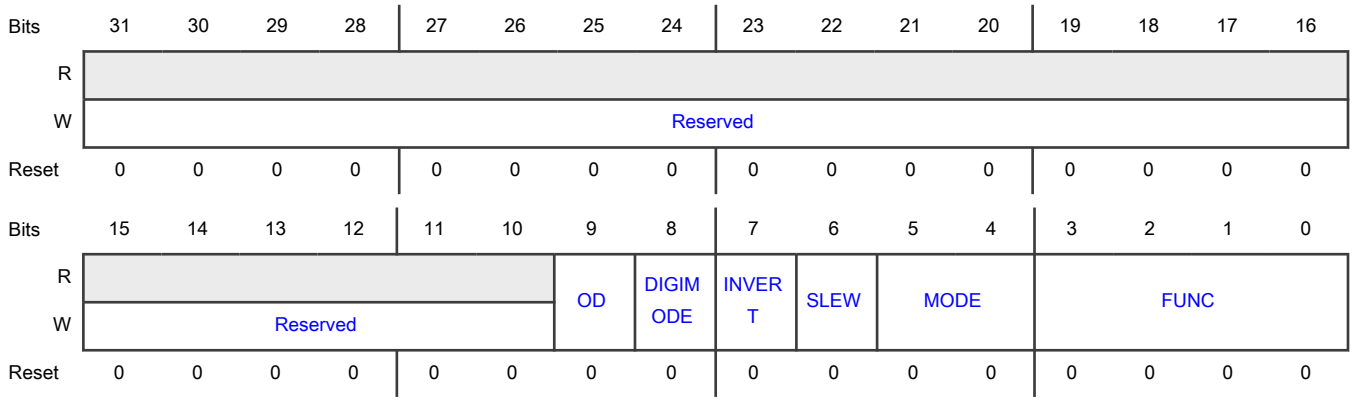
7.6.1.1.48 Digital I/O control for port (PIO2_25 - PIO3_6)

Type D registers.

Offset

Register	Offset
PIO2_25	164h
PIO2_26	168h
PIO2_27	16Ch
PIO2_28	170h
PIO2_29	174h
PIO2_30	178h
PIO2_31	17Ch
PIO3_0	180h
PIO3_1	184h
PIO3_2	188h
PIO3_3	18Ch
PIO3_4	190h
PIO3_5	194h
PIO3_6	198h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
9 OD	Controls open-drain mode 0 - Normal. Normal push-pull output 1 - Open-drain. Simulated open-drain output (high drive disabled).
8 DIGIMODE	Select Digital mode 0 - Disable digital mode. Digital input set to 0. 1 - Enable Digital mode. Digital input is enabled.
7 INVERT	Invert polarity of input signal 0 - Don't invert the signal. 1 - Invert the signal.
6 SLEW	Driver slew rate 0 - Standard-mode, output slew rate is slower. More outputs can be switched simultaneously. 1 - Fast-mode, output slew rate is faster. Refer to the appropriate specific device data sheet for details.
5-4 MODE	Mode select (on-chip pull-up/pull-down resistor control) These bits allow the selection of on-chip pull-up or pull-down resistors for each pin or select the plain input mode or the repeater mode. 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
3-0 FUNC	Signal(function) select Please refer Pin Multiplexing for more details. These bits can be set to GPIO (value 0) or to a peripheral signal. The default value is 0 (GPIO) except for P0_11 and P0_12 where default is FUNC = 6 (resp swclk and swdio special functions). For pins set to GPIO, the DIR registers in GPIO determine whether the pin is configured as an input or output. For peripheral signals, the pin direction is controlled automatically depending on the signal. The DIR registers have no effect for peripheral signals.

Chapter 8
SYSCON

8.1 Chip-specific SYSCON information

Table 23. Reference links to related information

Topic	Related module	Reference
Full description	SYSCON	SYSCON
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Power control API		Power control API
NVIC		NVIC assignments
Signal multiplexing	Port control	Signal multiplexing

8.1.1 Module instances

This device has one instance of the SYSCON module, SYSCON0.

8.1.2 Configuration

Configure the SYSCON block as follows:

- No clock configuration is needed. The clock to the SYSCON block is always enabled. By default, the SYSCON block is clocked by the FRO 12 MHz (fro_12m).
- The SYSCON block controls use of the CLKOUT pin, which must also be configured through IOCON. See Signals. RESET_B is a dedicated pin.

8.1.3 Set up the PLL0

The PLL0 creates a stable output clock at a higher frequency than the input clock. If a main clock is needed with a frequency higher than the FRO 12 MHz clock and the FRO 96 MHz clock (fro_hf) is not appropriate, use the PLL to boost the input frequency.

8.1.4 Set up the PLL1

The PLL1 creates a stable output clock at a higher frequency than the input clock. If a main clock is needed with a frequency higher than the FRO 12 MHz clock and the FRO 96 MHz clock (fro_hf) is not appropriate, use the PLL to boost the input frequency.

8.1.5 Configure the main clock and system clock

The clock source for the registers and memories is derived from main clock. The main clock can be selected from the sources listed in step 1 below.

The main clock, after being optionally divided by the CPU Clock Divider, is called the system clock and clocks the core, the memories, and the peripherals (register interfaces and peripheral clocks).

1. Select the main clock. The following options are available:
   - FRO 12 MHz output (fro_12m) from internal oscillator. This clock is divided down from FRO high-speed.

- On power-up, the LPC553x boots up using FRO high-speed output at 96 MHz.
- External oscillator.
- FRO 1 MHz output (fro_1m) from internal oscillator.
- The output of the PLL0.
- The output of the PLL1.
- The RTC 32 kHz oscillator.

NOTE

See MAINCLKSELA and MAINCLKSELB registers in SYSCON register memory map.

2. Select the divider value for the system clock AHBCLKDIV register.
3. Enable the clock for the memories and peripherals used in the application.
4. Peripheral clock must be disabled (AHBCLKCTRLx registers) when performing software reset on the peripherals via (PRESETCTRLx registers). Once reset assertion and de-assertion is completed, the respective peripheral clock can be enabled.

8.1.6 Clock generation

The system control block facilitates the clock generation. Many clocking variations are possible. Figure 9 gives an overview of potential clock options. Table 24 describes signals on the clocking diagram. The maximum clock frequency is 150 MHz.

NOTE

The indicated clock multiplexers shown in Figure 9 are synchronized. In order to operate, the currently selected clock must be running, and the clock to be switched to must also be running so the multiplexer can gracefully switch between the two clocks without glitches. Other clock multiplexers are not synchronized. The output divider can be stopped and restarted gracefully during switching if a glitch-free output is needed.

The low-power oscillator provides a frequency in the range of 1 MHz. The accuracy of this clock is limited to +/- 15% over temperature, voltage, and silicon processing variations after trimming made during assembly. To determine the actual Watchdog oscillator output, use the frequency measure block. See Chip-specific Frequency Measurement information.

The device contains two PLLs (PLL0 and PLL1) that can be configured to use a number of clock inputs and produce an output clock in the range of 1.2 MHz up to the maximum chip frequency, and can be used to run most on-chip functions. The output of the PLL can be monitored through the CLKOUT pin.

NOTE

The maximum allowed frequency for the main clock and system clock (to CPU0, AHB bus, Sync, etc.,) is 150 MHz. The POWER_SetVoltageForFreq API call must always be used when setting or switching the frequency. See Power Profiles/Power Control API.

Table 24. Clocking diagram signal name descriptions

Name	Description
32k_osc	The 32 kHz clock source. It is selected as either FRO32K or XTAL32K in the RTCOSC32K register.
AHB_clk	The AHB bus clock. Used by the CPU, AHB bus, APB bus, and others. Derived from main_clk.
clk_in	It is the internal clock that comes from the external oscillator.
CLKOUT	Many on-chip clocks can potentially be selected to be output on the CLKOUT function. This can be used for debugging purposes or to drive some external logic (see data sheet for limitations on pin output frequency. If used, the CLKIN pin function must be connected to a pin by selecting it in the IOCON block.

Table continues on the next page...

Table 24. Clocking diagram signal name descriptions (continued)

Name	Description
frg_clk	The output of each Fractional Rate Generator to Flexcomm clock. Each FRG and its source selection is shown in Figure 10.
fro_12m	12 MHz divided down from the currently selected on-chip FRO_192 oscillator.
fro_1m	The output of the low power oscillator.
fro_hf	The currently selected FRO_192 high speed output at 96 MHz. FRO_HF clock is the output of the FRO_192 divided by 2 (96 MHz). Note that this clock can only be used for USB device and is not reliable for USB host timing requirements of the data signaling rate.
fro_hf_div	This is fro_hf potentially divided by the FRO_HF Divider. Used in places where fro_hf is faster than needed in order to save power.
FRO32K	Output of the on-chip 32 kHz FRO.
main_clk	The main clock used by the CPU and AHB bus, and potentially many others. The main clock and its source selection are shown in Figure 9.
mclk_in	The MCLK input function, when it is connected to a pin by selecting it in the IOCON block.
"none"	A tied-off source that should be selected to save power when the output of the related multiplexer is not used.
pll_clk_div	The output of either PLL, potentially divided down. The raw PLL output may be too high a frequency to be used directly, or can be divided down to save power.
pll0_clk	The output of the PLL0. The PLL0 and its source selection is shown in Figure 9.
pll1_clk	The output of the PLL1. The PLL1 and its source selection is shown in Figure 9.
wdt_clk	The clock to the watchdog timer.
xtal32MHz_freqme	Output of the Main crystal oscillator provided to the frequency measurement function.
xo_to_adc_clk	Output of the Main crystal oscillator provided to the ADC.
XTAL32K	Output of the 32 kHz crystal oscillator.
Xtalin, xtalout	Pins for the main crystal oscillator.

NOTE

The maximum frequency of OSTimer clock is 100 MHz.

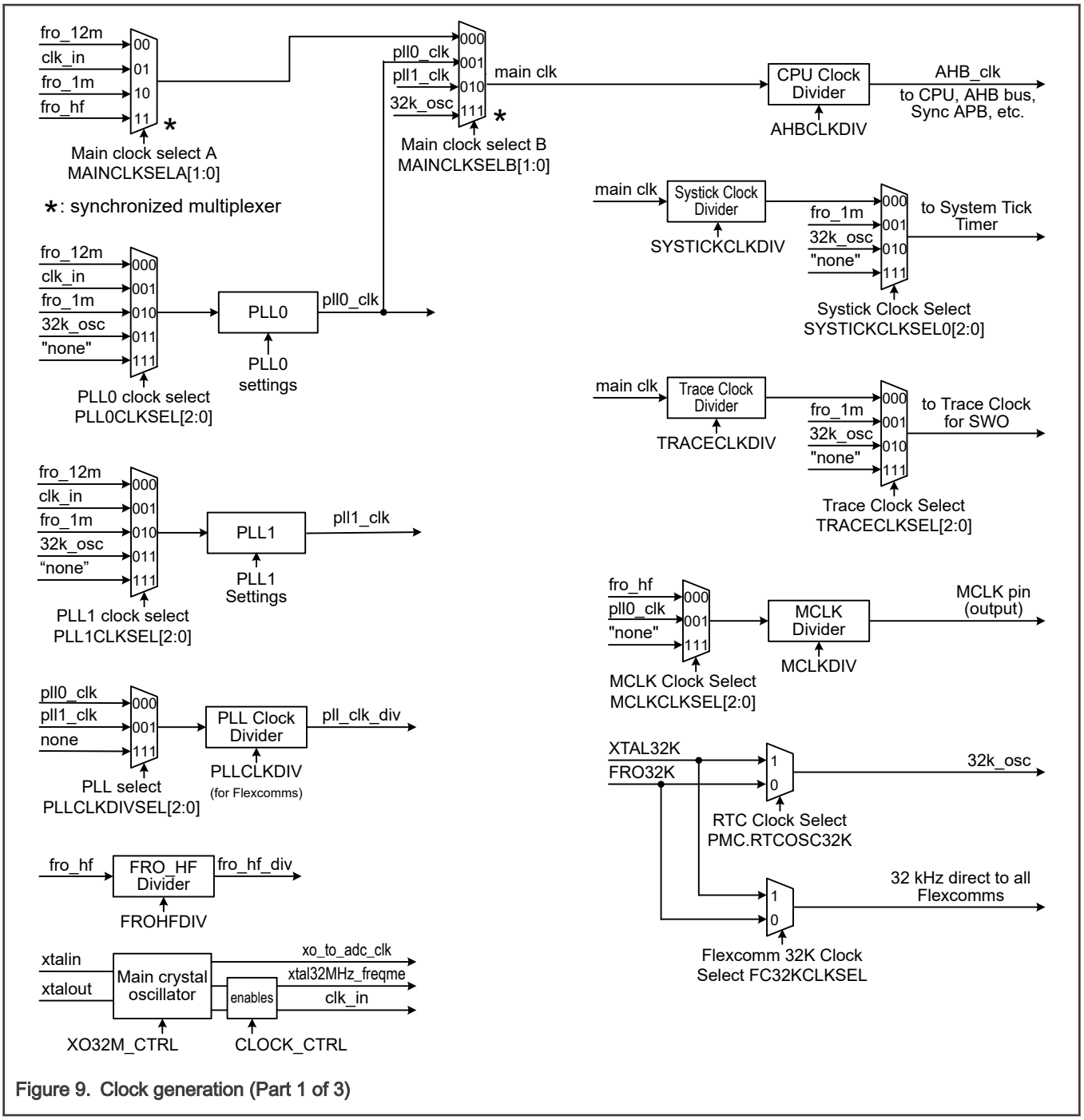


Figure 9. Clock generation (Part 1 of 3)

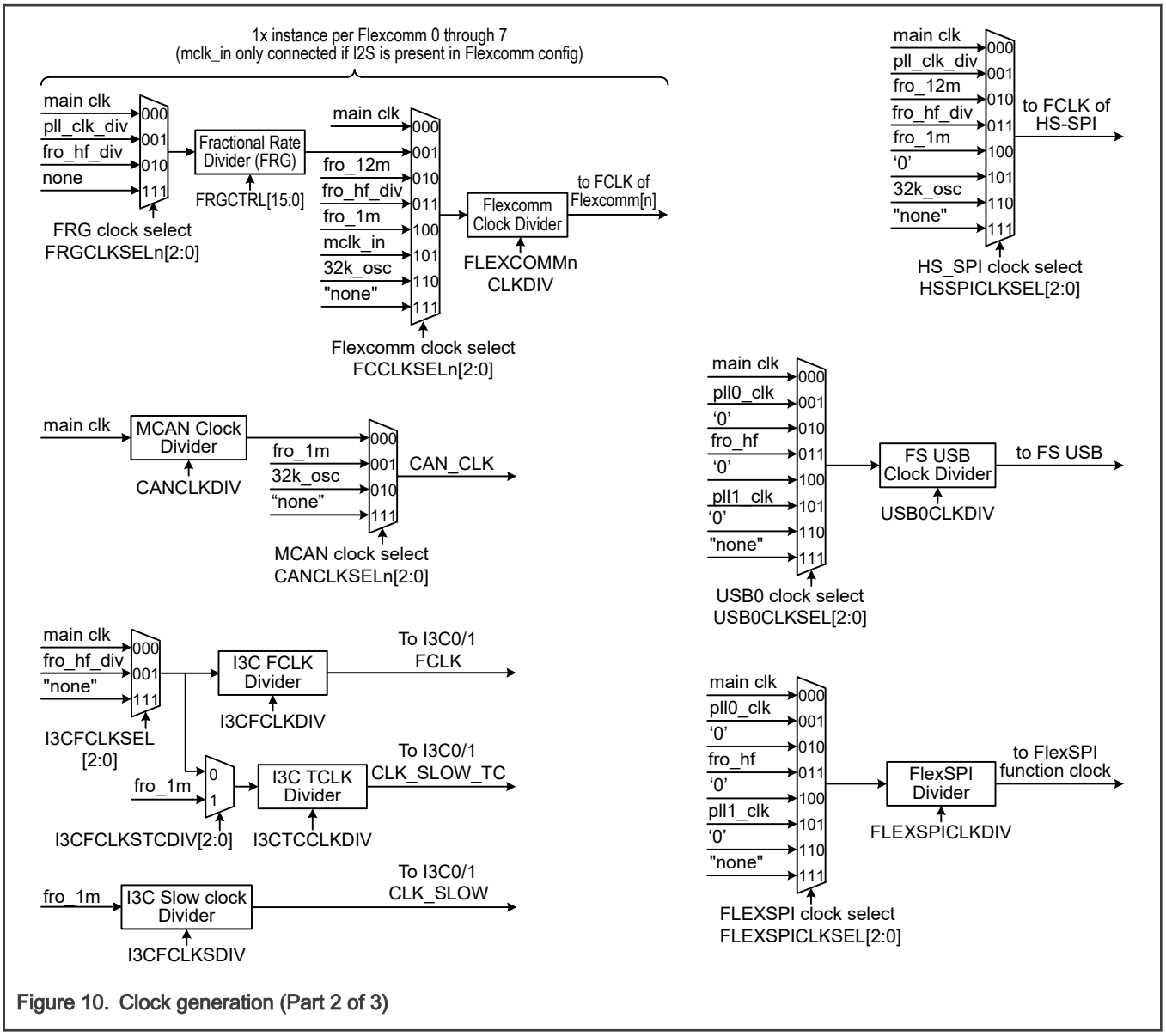


Figure 10. Clock generation (Part 2 of 3)

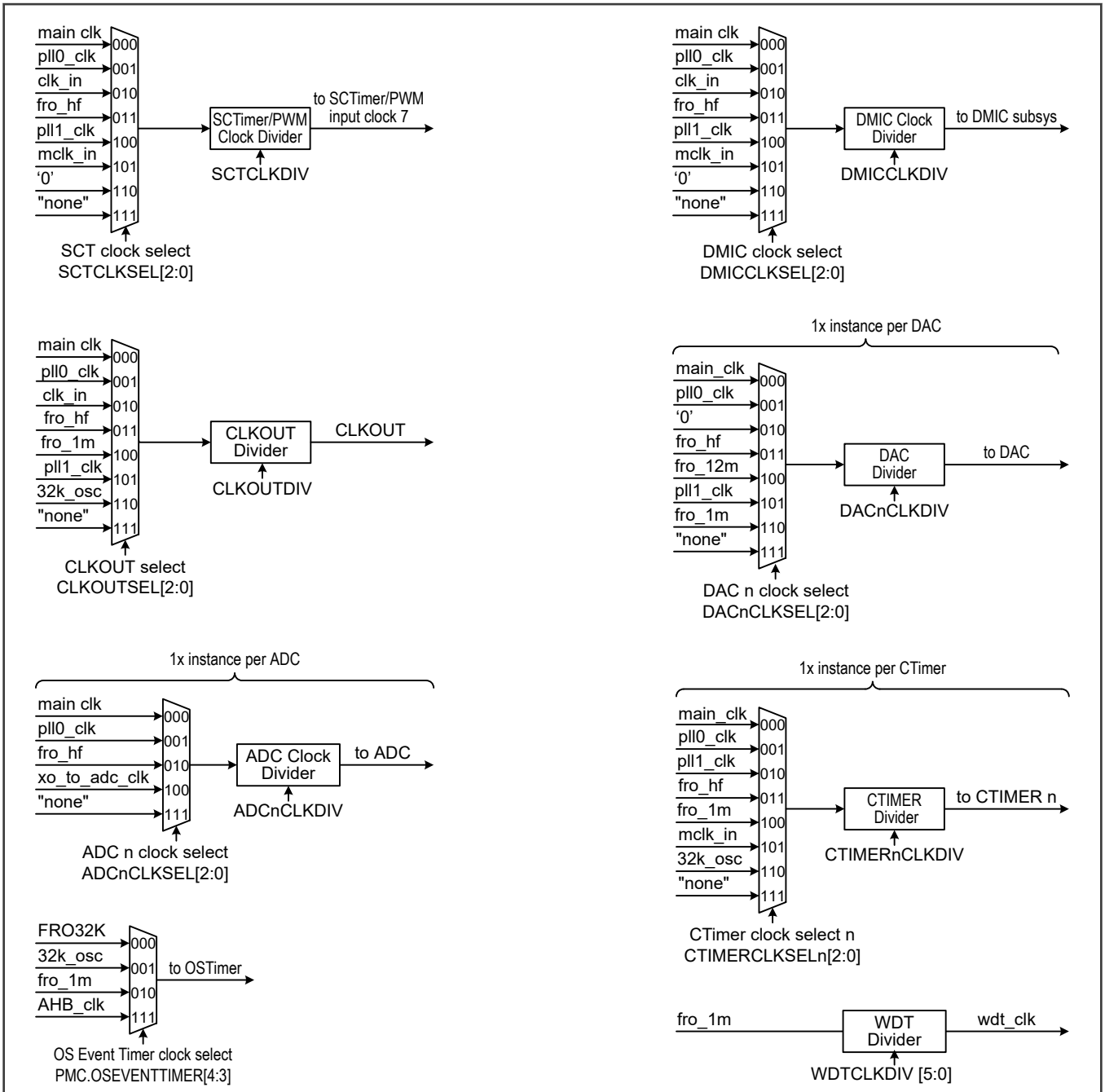


Figure 11. Clock generation (Part 3 of 3)

8.2 Overview

This section provides an introduction to the SYSCON module.

8.2.1 Features

- System and bus configuration.
- Clock select and control.
- PLL0 and PLL1 configuration.

- Reset control.
- Wake-up control.
- High-accuracy frequency measurement function for on-chip and off-chip clocks.
- Uses a selection of on-chip clocks as the reference clock.
- Device ID register.

8.3 Functional description

8.3.1 Clock

The main clock select multiplexers are implemented with glitch-free logic. All the other clock multiplexers described in this chapter cannot be considered as glitch-free, thus it is necessary to pay attention during clock switching. All the dividers can be halted and restarted during clock switching, to provide a glitch free output.

8.3.2 Reset

Reset has the following sources:

- The RESET_B pin.
- Watchdog reset.
- Power-On Reset (POR).
- 2x (VDD_MAIN, VDD_CORE) Brown Out Detect output with selectable ranges, programable to use as interrupt or reset.
- Individual SW Reset control in Syscon for many peripherals.
- Arm system reset.
- ISP-AP debug reset.
- Software reset.

Assertion of the POR or the BOD reset, once the operating voltage attains a usable level, starts the FRO_192. After the FRO-start-up time, the FRO_192 provides a stable clock output. The reset remains asserted until the external reset is released, the oscillator is running, and the flash controller has completed its initialization.

On the assertion of any reset source (Arm system reset, POR, BOD reset, external reset, watchdog reset, and Software Reset), the following processes are initiated:

1. The FRO is enabled or starts up if not running.
2. The flash wake-up starts. This takes approximately 40µs.
3. The boot code in the ROM starts. The boot code performs the boot tasks and may jump to the flash.

When the internal reset is removed, the processor begins executing at address 0, which is initially the reset vector mapped from the boot block. At that point, all of the processor and peripheral registers have been initialized to predetermined values.

The matrix in below table describes the various reset types and shows which associated domains/components get reset by each type. "Rst" indicates that the sub-domain is reset by the associated reset source, and "Act" indicates that it remains active and does not change state in response to that reset source:

Table 25. Resets

Chip reset/wakeup for the following components:	POR (Power on Reset)	nRESET	BODRESET	SYSTEMRESET	WDTRESET	SWRRESET	DPDRESET_WAKEUPIO	DPDRESET_RTC	DPDRESET_OSTIMER	CDOGRESET	Wake up from Power Down	Wake up from Deep Sleep	Wake up from Sleep w
SRAM Retention Memory	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act	Act	Rst	Act	Act	Act
SRAM Memory	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
FLASH Memory	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
Flash controller	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
WDT	Rst	Rst	Rst	Rst	Act	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
UTICK	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
MRT	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
UTIMER	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
cdog	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
Debug mailbox	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
cpu0	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act	Act
CRC Random Generator	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
FLEXSPI SS	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act	Act
DMA	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
analog_ctrl	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
ADC	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
USB0-FS	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
ctimers (32bit)	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
flexcomm (all except 3)	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
INPUTMUX	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
GPIO	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
PINT	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
CAN	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act

Table continues on the next page...

Table 25. Resets (continued)

Chip reset/wakeup for the following components:	POR (Power on Reset)	nRESET	BODRESET	SYSTEMRESET	WDTRESET	SWRRESET	DPDRESET_WAKEUPIO	DPDRESET_RTC	DPDRESET_OSTIMER	CDOGRESET	Wake up from Power Down	Wake up from Deep Sleep	Wake up from Sleep w
i2s sharing	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
CAN FD	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
powerquad	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act
Motocon Subsystem	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Rst	Act	Act

8.3.3 Start-up behavior

The FRO 12 MHz oscillator provides the default clock at reset and provides a clean system clock shortly after the supply pins reach operating voltage. See the device data sheet for details of start-up timing.

Note: The ROM boot code might switch to a higher frequency (either 24 MHz, or 48 MHz) based on the settings in the Boot_Speed (BOOT_CFG < 8:7 >) bits of CMPA Protected Flash Area (PFR).

8.3.4 Brown-out detection

This device includes two Brown-out detector to monitor the voltage VDD_MAIN, VDD_CORE. If the voltage falls below one of the selected voltages, the BOD asserts an interrupt to the NVIC or issues a reset, see BODVDDMAIN and RESETCTRL registers in PMC.

The interrupt signal can be enabled for interrupt in the interrupt enable register in the NVIC, see Nested Vectored Interrupt Controller (NVIC), to cause a CPU interrupt; if not, software can monitor the signal by reading a dedicated status register.

If the BOD interrupt is enabled, the BOD interrupt can wake up the chip from a reduced power mode, not including power-down and deep power-down.

If the BOD reset is enabled, the forced BOD reset can wake up the chip from reduced power modes, not including power-down and deep power-down.

8.3.5 Flash accelerator functional description

The flash accelerator is also known as the Flash Memory Controller, or FMC. The FMC is distinct from, and interfaces with the Flash Controller.

Flash cache engine with 8 KB RAM (LPCAC) is to enable 0-latency flash access. LPCAC Cache is placed between AHB Slave port and FMC. LPCAC is a low-power instruction cache, that implements LRU based replacement scheme. Cache RAM size should be 8 KB.

The flash accelerator block allows maximization of the performance of the CPU when it is running code from flash memory, while also saving power. The flash accelerator also provides speed and power improvements for data accesses to the flash memory.

The flash accelerator is divided into several functional blocks:

- AHB matrix interface, accessible by all bus masters that have a connection to the matrix slave port used for flash memory.
- An array of eight 128-bit buffers.
- Flash accelerator control logic, including address compare and flash control.
- A flash memory interface.

Below diagram shows a simplified diagram of the flash accelerator blocks and data paths.

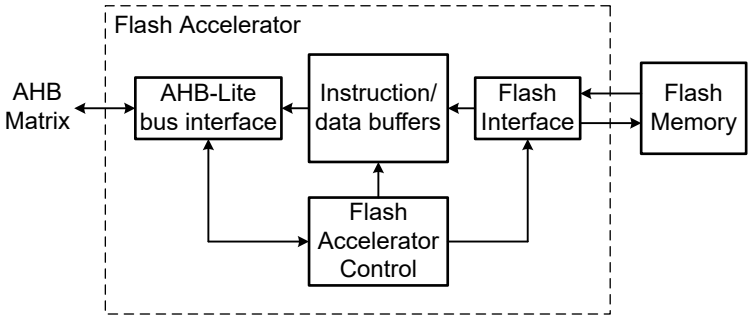


Figure 12. Simplified block diagram of the flash accelerator

In the following descriptions, the term *fetch* applies to an explicit flash read request from the CPU.

8.3.5.1 Flash memory bank

Flash programming operations are not controlled by the flash accelerator, but are handled as a separate function. The boot code includes flash programming functions that may be called as part of the application program, as well as loaders that may be used to accomplish initial flash programming.

8.3.5.2 Flash programming constraints

Since the flash memory does not allow accesses during programming and erase operations, it is necessary for the flash accelerator to force the CPU to wait if a memory access to a flash address is requested while the flash memory is busy with a programming operation. Under some conditions, this delay could result in a Watchdog time-out. The user will need to be aware of this possibility and take steps to insure that an unwanted Watchdog reset does not cause a system failure while programming or erasing the flash memory. Application code, especially interrupts, can continue to run from other memories during flash erase/write operations.

To preclude the possibility of stale data being read from the flash memory, the flash accelerator buffers are automatically invalidated at the beginning of any flash programming or erase operation. Any subsequent read from a flash address will cause a new fetch to be initiated after the flash operation has completed.

NOTE

Flash ERASE and PROGRAM operations must be performed with a system clock below or equal to 100 MHz.

8.3.6 PLL0 and PLL1 functional description

The PLL is typically used to create a frequency that is higher than other on-chip clock sources, and used to operate the CPU and/or other on-chip functions. It may also be used to obtain a specific clock that is otherwise not available. For example, a source clock with a frequency of any integer MHz (for example, the 12 MHz FRO) can be divided down to 1 MHz, then multiplied up to any other integer MHz (for example, 13, 14 and15).The PLL can be set up by calling an API supplied by NXP Semiconductors. Also see PLL1 and PLL0 Registers.

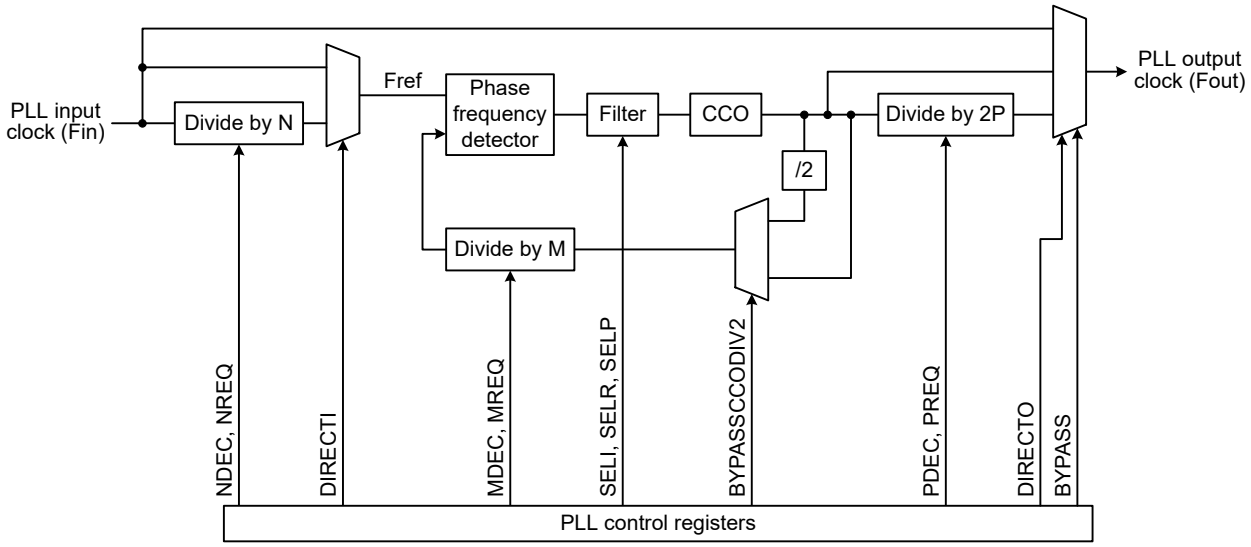


Figure 13. System PLL block diagram showing typical operation

8.3.6.1 PLL features

- Integrated PLL with no external components for clock generation.
- Large input range at the phase detector: 2 kHz - 150 MHz.
- CCO frequency: 275 MHz - 550 MHz.
- Output clock (clkout) range: 4.3 MHz to 550 MHz (max limited to 150 MHz).
- Programmable:
  - Pre-divider N, (N, 1 to 2^8-1)
  - Feedback-divider M, (M, 1 to 2^{16}-1)
  - Post-divider P * 2 (P, 1 to 2^5-1)
- Programmable bandwidth (integrating action, proportional action, high frequency pole).
- Real-time adjustment of the clock (dividers with handshake control).
- Positive edge clocking.
- Frequency limiter to avoid *hang-up* of the PLL.
- Lock detector.
- Power-down mode.
- Possibility to bypass whole PLL.
- Possibility to bypass the post-divider.
- Possibility to bypass the pre-divider.
- Possibility to disable the output clock.
- Spread Spectrum mode (only on PLL0).

8.3.6.2 PLL description

A number of sources may be used as an input to the PLL, see Clock generation for more information. In addition, a block diagram of the PLL is shown in PLL0 and PLL1 functional description. The PLL input, in the range: 2 kHz to 150 MHz, may initially be divided down by a value *N*, which may be in the range of 1 to 255. This input division provides a greater number of possibilities in providing a wide range of output frequencies from the same input frequency.

Following the PLL input divider is the PLL multiplier. The multiplier can multiply the input divider output through the use of a Current Controlled Oscillator (CCO) by a value *M*, in the range of 1 through 65,535. The resulting frequency must be in the range of 275 MHz to 550 MHz. The multiplier works by dividing the CCO output by the value of M, then using a phase-frequency detector to compare the divided CCO output to the multiplier input. The error value is filtered and used to adjust the CCO frequency.

The PLL output may further be divided by a value *2P* if desired, where P is value in the range of 1 to 31.

All of the dividers that are part of the PLL use an encoded value, not the binary divide value. The LPCOpen Chip_POWER_SetPLL API, see POWER_EnterSleep, can adjust the value for the main feedback divider (the M divider), but does not accept pre- and post-divider values. See section PLL operating modes and PLL usage for information on how to obtain divider values.

There are additional dividers in the clocking system to bring the PLL output frequency down to what is needed for the CPU, USB, and other peripherals. The PLL output dividers are described in the Clock Dividers section following the PLL description.

8.3.6.2.1 Lock detector

The lock detector measures the phase difference between the rising edges of the input and feedback clocks. Only when this difference is smaller than the so called *lock criterion* for more than seven consecutive input clock periods, the lock output switches from low to high. A single too large phase difference immediately resets the counter and causes the lock signal to drop (if it was high). Requiring seven phase measurements in a row to be below a certain figure ensures that the lock detector will not indicate lock until both the phase and frequency of the input and feedback clocks are very well aligned. This effectively prevents false lock indications, and thus ensures a glitch free lock signal.

The PLL lock indicator is not reliable when F_{ref} is below 100 kHz or above 20 MHz. Instead, software should use a 6 ms time interval to insure the PLL will be stable.

For PLL0, spread spectrum mode, the PLL will generally not lock, software should use a 6 ms time interval to insure the PLL will be stable. See Procedure for determining PLL settings.

8.3.6.2.2 Power-down

To reduce the power consumption when the PLL clock is not needed, a PLL power-down mode has been incorporated. This mode is enabled by setting the PDEN_PLLn (where n indicates PLL number) bit to one in the power configuration register PDRUNCFG0 (see PMC chapter). In this mode, the internal current reference will be turned off, the oscillator and the phase-frequency detector will be stopped and the dividers will enter a reset state. While in PLL power-down mode, the lock output will be low to indicate that the PLL is not in lock.

When the PLL power-down mode is terminated by setting the PDEN_PLLn (where n indicates PLL number) bit to zero, the PLL will resume its normal operation and will make the lock signal high once it has regained lock on the input clock. While in this state, new divider values may be entered, which will be used when the PLL power-down state is exited by clearing PDEN_PLLn (where n indicates PLL number).

8.3.6.3 PLL operating modes

The PLL includes several main operating modes, and a power-down mode. These are summarized in Table 26 and detailed in the following sections.

Table 26. PLL operating mode summary

Mode	PDEN_PLLn (where n indicates PLL number) bit in PDRUNCFG0	Bits in SYSPLLCTRL:			SEL_EXT bit in PLL0SSCG0	PD bit in PLL0SSCG1
		BYPASS	UPLIMOFF	BANDSEL		
Normal	0	0	0	1	1	1
Spread spectrum (only for PLL0)	0	0	1	0	0	0
Power-down	1	x [1]	x	x	x	1

1. Use 1 if the PLL output is used even though the PLL is not altering the frequency.

8.3.6.3.1 Normal modes

Typical operation of the PLL includes an optional pre-divide of the PLL input, followed by a frequency multiplication, and finally an optional post-divide to produce the PLL output.

Notations used in the frequency equations:

- Fin = the input to the PLL.
- Fout = the output of the PLL.
- Fref = the PLL reference frequency, the input to the phase frequency detector.
- N = optional pre-divider value.
- M = feedback divider value, which represents the multiplier for the PLL. Note that an additional divide-by-2 may optionally be included in the divider path.
- P = optional post-divider value. An additional divide-by-2 is included in the post-divider path.

A block diagram of the PLL as used in normal modes is shown in Figure 14.

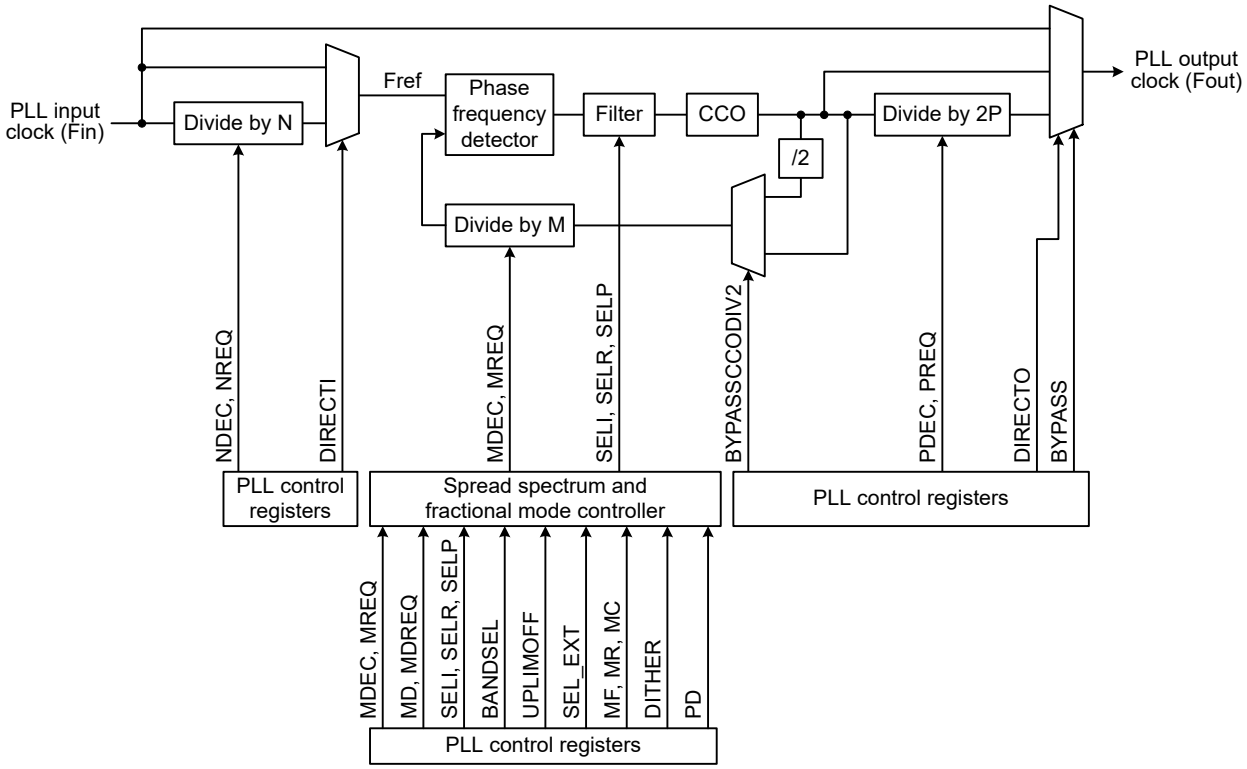


Figure 14. System PLL block diagram showing spread spectrum and fractional divide operation

Mode 1a: Normal operating mode without post-divider and without pre-divider

In normal operating mode 1a the post-divider and pre-divider are bypassed. The operating frequencies are:

$F_{out} = F_{cco} = M \times F_{in} \wedge (275\ MHz \leq F_{cco} \leq 550\ MHz, 2\ kHz \leq F_{in} \leq 150\ MHz)$

The feedback ratio is programmable:

- Feedback-divider M (M, 1 to $2^{16} - 1$)

Mode 1b: Normal operating mode with post-divider and without pre-divider

In normal operating mode 1b the pre-divider is bypassed. The operating frequencies are:

$F_{out} = F_{cco} / (2 \times P) = M / (2 \times P) \times F_{in} \wedge (275\ MHz \leq F_{cco} \leq 550\ MHz, 2\ kHz \leq F_{in} \leq 150\ MHz)$

The divider ratios are programmable:

- Feedback-divider M (M, 1 to $2^{16} - 1$)
- Post-divider P (P, 1 to $2^5 - 1$)

Mode 1c: Normal operating mode without post-divider and with pre-divider

In normal operating mode 1c the post-divider with divide-by-2 divider is bypassed. The operating frequencies are:

$F_{out} = F_{cco} = M/N \times F_{in} \wedge (275\ MHz \leq F_{cco} \leq 550\ MHz, 2\ kHz \leq F_{in}/N \leq 150\ MHz)$

The divider ratios are programmable:

- Pre-divider N (N, 1 to 2^8 - 1)
- Feedback-divider M (M, 1 to 2^{16} - 1)

Mode 1d: Normal operating mode with post-divider and with pre-divider

In normal operating mode 1d none of the dividers are bypassed. The operating

frequencies are:

$F_{out} = F_{cco}/2xP = M/ (N \times 2 \times P) \times F_{in} \wedge (275\ MHz \leq F_{cco} \leq 550\ MHz, 2\ kHz \leq F_{in}/N \leq 150\ MHz)$

The divider ratios are programmable:

- Pre-divider N (N, 1 to 2^8 - 1)
- Feedback-divider M (M, 1 to 2^{16} - 1)
- Post-divider P (P, 1 to 2^5 - 1)

8.3.6.3.2 Selecting the bandwidth

In normal applications the bandwidth must be calculated manually by using the equations below for seli and selp. In that case the PLL will be automatically stable. In normal applications pin band_direct has to be low (‘0’) in this case the bandwidth is changed together with the M-divider value.

For normal applications the value for selp[4:0] must be calculated using the following equation:

selp = floor(M/4) + 1

Where:

- Feedback-divider M (M, 1 to 2^{16} - 1)
- If selpcalculated >= 31 then selp[4:0] = 31

For normal applications the value for seli[5:0] must be calculated using one of the following equations depending on the value of the feedback divider M:

if (M >= 8000) => seli = 1

if (8000 > M >= 122) => seli = floor(8000/M)

if (122 > M >= 1) => seli = 2 * floor(M/4) + 3

Where:

- Feedback-divider M (M, 1 to 2^{16} - 1)
- If seli >= 63 then seli[5:0] = 63.

For normal applications the value for selr[3:0] must be kept 0.

For frequencies at the phase detector smaller than 50 kHz (Fin/N ≤ 50kHz) please consult NXP.

In some applications, it is preferable to change the bandwidth directly on the PLL. In such an application, Bit BWDIRECT in the PLLxCTRL register must be set high (‘1’).

8.3.6.3.3 Spread spectrum mode

The spread spectrum functionality can be used to modulate the PLL output frequency automatically, in a programmable manner. It can decrease electromagnetic interference (EMI) in an application.

The spread spectrum clock generator can be used in several ways:

- It can encode M-divider values between 1 and 255 to produce the MDEC value used directly by the PLL, saving the need for executing encoding algorithm code, or hard-coding predetermined values into an application.
- It can provide a fractional rate feature to the PLL.

- It can be set up to automatically alter the PLL CCO frequency on an ongoing basis to decrease electromagnetic interference (EMI).

A block diagram of the PLL as used in fractional mode is shown in Figure 14.

If the spread spectrum mode is enabled, choose N to ensure 3 MHz < Fin/N < 5 MHz. Spread spectrum mode cannot be used when Fin = 32 kHz.

When the modulation (MR) is set to zero, the PLL becomes a fractional PLL.

Triangular wave modulation

For the center spread triangular waveform modulation with a modulation frequency depth δfmodpk-pk and a modulation frequency fm, the clock cycle displacement and spectral tone reduction ΔP can be calculated. The theoretical maximum clock cycle displacement (peak-to-peak) can be expressed with the following equation below:

if $directo_{PLL}$ = 1:

$$\Delta n_{max;theoretically} = (N_{ss} \times k)/16$$

if $directo_{PLL}$ = 0, P_{PLL} = 1:

$$\Delta n_{max;theoretically} = (N_{ss} \times k)/(32 \times P_{PLL})$$

In practice, the clock cycle displacement could be larger. So, for safety reasons (buffer overflow) use:

if $directo_{PLL}$ = 1:

$$\Delta n_{max;practically} = (N_{ss} \times k)/8$$

if $directo_{PLL}$ = 0, P_{PLL} = 1:

$$\Delta n_{max;practically} = (N_{ss} \times k)/(16 \times P_{PLL})$$

The spectral tone reduction/EMI reduction ΔP at F_{out} is approximately:

if $directo_{PLL}$ = 1:

$$\Delta P \quad 10\log(N_{ss} \times k)/(2)$$

if $directo_{PLL}$ = 0, P_{PLL} = 1:

$$\Delta P \quad 10\log(N_{ss} \times k)/(4 \times P_{PLL})$$

See below table for the spectral tone reduction and clock cycle displacement for $directo_{PLL}$ = 0 and P_{PLL}= 1.

Table 27. Values for different settings, $directo_{PLL}$ = 0, P_{PLL} = 1

Table values are: ΔP Δn_{max}	mf[2:0]=000 N_{SS} = 512	mf[2:0]=001 N_{SS} ≈ 384	mf[2:0]=010 N_{SS} = 256	mf[2:0]=011 N_{SS} = 128	mf[2:0]=100 N_{SS} = 64	mf[2:0]=101 N_{SS} = 32	mf[2:0]=110 N_{SS} ≈ 24	mf[2:0]=111 N_{SS} = 16
mr[2:0]=000, k≈0	0 dB 0	0 dB 0	0 dB 0	0 dB 0	0 dB 0	0 dB 0	0 dB 0	0 dB 0
mr[2:0]=001, k≈1	21 dB 32	20 dB 24	18 dB 16	15 dB 8	12 dB 4	9 dB 2	8 dB 1.5	6 dB 1
mr[2:0]=010, k≈1.5	23 dB 48	22 dB 32	20 dB 24	17 dB 12	14 dB 6	11 dB 3	10 dB 2.2	8 dB 1.5

Table continues on the next page...

Table 27. Values for different settings, directo$_{PLL}$ = 0, P$_{PLL}$ = 1 (continued)

Table values are: ΔP Δn$_{max}$	mf[2:0]=000 N$_{SS}$ = 512	mf[2:0]=001 N$_{SS}$ ≈ 384	mf[2:0]=010 N$_{SS}$ = 256	mf[2:0]=011 N$_{SS}$ = 128	mf[2:0]=100 N$_{SS}$ = 64	mf[2:0]=101 N$_{SS}$ = 32	mf[2:0]=110 N$_{SS}$ ≈ 24	mf[2:0]=111 N$_{SS}$ = 16
mr[2:0]=011, k≈2	24 dB 64	23 dB 48	21 dB 32	18 dB 16	15 dB 8	12 dB 4	11 dB 3	9 dB 2
mr[2:0]=100, k≈3	26 dB 96	25 dB 64	25 dB 48	20 dB 24	17 dB 12	14 dB 6	13 dB 4.5	12 dB 4
mr[2:0]=101, k≈4	27 dB 128	26 dB 96	24 dB 64	21 dB 32	18 dB 16	15 dB 8	14 dB 6	12 dB 4
mr[2:0]=110, k≈6	28 dB 192	28 dB 128	26 dB 96	23 dB 48	20 dB 24	17 dB 12	16 dB 9	14 dB 6
mr[2:0]=111, k≈8	30 dB 256	29 dB 192	27 dB 128	24 dB 64	21 dB 32	18 dB 16	17 dB 12	15 dB 8

8.3.6.3.4 PLL power-down mode

If the PLL is not used, or if it there are cases where it is turned off in a running application, power can be saved by putting the PLL in power-down mode. Before this is done, the CPU and any peripherals that are not meant to be stopped as well, must be running from some other clock source.

8.3.6.4 PLL related registers

The PLL is controlled by registers described in Memory map and register definition summarized below.

Table 28. Summary of PLL related registers

Register	Description
PLLxCTRL	PLL control.
PLLxSTAT	PLL status.
PLLxNDEC	PLL pre-divider.
PLLxPDEC	PLL post-divider.
PLL0SSCTRL	PLL spread spectrum control 0.
PLL0SSCTRL1	PLL spread spectrum control 1.

8.3.6.5 PLL usage

As previously noted, the PLL divider settings used in the PLL registers are not simple binary values, they are encoded as shown in the PLL register descriptions. The divider values and their encoding can be found by calculation using the information in this document. For simple PLL usage with no pre-divide or post-divide, the LPCOpen Chip_POWER_SetPLL API can be used, see Power Control API. Also, a PLL setting calculator can be found on the NXP website. The latter two possibilities are recommended in order to avoid PLL setup issues.

8.3.6.5.1 Procedure for determining PLL settings

In general, PLL configuration values may be found as follows:

1. Identify a desired PLL output frequency. This may depend on a specific interface frequency needed or be based on expected CPU performance requirements, and may be limited by system power availability.

2. Determine which clock source to use as the PLL input. This can be influenced by the power or accuracy that is required, or by the potential to obtain the desired PLL output frequency.
3. Identify PLL settings to obtain the desired output from the selected input. The Fcco frequency must be either the actual desired output frequency, or the desired output frequency times 2 x P, where P is from 2 to 31. The Fcco frequency must also be a multiple of the PLL reference frequency, which is either the PLL input, or the PLL input divided by N, where N is from 2 to 255.
4. There may be several ways to obtain the same PLL output frequency. PLL power depends on Fcco (a lower frequency uses less power) and the divider used. Bypassing the input and/or output divider saves power.
5. Check that the selected settings meet all of the PLL requirements:
   - Fin is in the range of 32 kHz to 100 MHz.
   - Fcco is in the range of 275 MHz to 550 MHz.
   - Fout is in the range of 1.2 MHz to 100 MHz.
   - The pre-divider is either bypassed, or N is in the range of 2 to 255.
   - The post-divider is either bypassed, or P is in the range of 2 to 31.
   - M is in the range of 3 to 65,535.

Also note that PLL startup time becomes longer as Fref drops below 500 kHz. At 500 kHz and above, startup time is up to 500 microseconds. Below 500 kHz, startup time can be estimated as 200 / Fref, or up to 6.1 milliseconds for Fref = 32 kHz. PLL accuracy and jitter is better with higher values of Fref.

8.3.6.5.2 PLL setup sequence

The following sequence should be followed to initialize and connect the PLL:

1. Make sure that the PLL output is disconnected from any downstream functions. If the PLL was previously being used to clock the CPU, and the CPU Clock Divider is being used, it may be set to speed up operation while the PLL is disconnected.
2. Select a PLL input clock source. See PLL0CLKSEL register.
3. Set up the PLL dividers and mode settings. See PLL1 and PLL0 Registers.
4. Wait for the PLL output to stabilize. The start-up time is 500 µs + 300 / Fref seconds.
5. If the PLL will be used to clock the CPU, change the CPU Clock Divider setting for the operation with the PLL, if needed. This must be done before connecting the PLL.
6. Connect the PLL to whichever downstream function with which it is being used. The structure of the clock dividers may be seen on the right in clock generation chapter in SYSCON chip-specific information.

8.4 Signals

Table 29. SYSCON pin description

Function	Type	Pin	Description
CLKOUT	O	PIO0_16, PIO0_26	CLKOUT clock output.

8.5 Memory map and register definition

This section includes the SYSCON module memory map and detailed descriptions of all registers.

8.5.1 SYSCON register descriptions

8.5.1.1 SYSCON memory map

SYSCON0 base address: 4000_0000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Memory Remap Control (MEMORYREMAP)	32	See section	See section
10	AHB Matrix priority control (AHBMATPRIO)	32	See section	See section
14	AHB Matrix priority control (AHBMATPRIO1)	32	See section	0000_0000
3C	System tick calibration for non-secure part of CPU0 (CPU0NSTCKCAL)	32	See section	See section
48	NMI Source Select (NMISRC)	32	See section	See section
100	Peripheral reset control 0 (PRESETCTRL0)	32	See section	See section
104	Peripheral reset control 1 (PRESETCTRL1)	32	See section	See section
108	Peripheral reset control 2 (PRESETCTRL2)	32	See section	See section
10C	Peripheral reset control 3 (PRESETCTRL3)	32	See section	0000_0000
120 - 124	Peripheral reset control set n (PRESETCTRLSET0 - PRESETCTRLSET1)	32	WO	0000_0000
128	Peripheral reset control set n (PRESETCTRLSET2)	32	See section	0000_0000
12C	Peripheral reset control set n (PRESETCTRLSET3)	32	WO	0000_0000
140 - 144	Peripheral reset control clear n (PRESETCTRLCLR0 - PRESETCTRLCLR1)	32	WO	0000_0000
148	Peripheral reset control clear n (PRESETCTRLCLR2)	32	See section	0000_0000
14C	Peripheral reset control clear n (PRESETCTRLCLR3)	32	WO	0000_0000
160	Software Reset (SWR_RESET)	32	WO	0000_0000
200	AHB Clock control 0 (AHBCLKCTRL0)	32	See section	See section
204	AHB Clock control 1 (AHBCLKCTRL1)	32	See section	See section
208	AHB Clock control 2 (AHBCLKCTRL2)	32	See section	See section

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
20C	AHB Clock Control 3 (AHBCLKCTRL3)	32	See section	0000_0000
220 - 22C	AHB Clock Control Set (AHBCLKCTRLSET0 - AHBCLKCTRLSET3)	32	RW	0000_0000
240 - 24C	AHB Clock Control Clear (AHBCLKCTRLCLR0 - AHBCLKCTRLCLR3)	32	RW	0000_0000
260	System Tick Timer for CPU0 source select (SYSTICKCLKSEL0)	32	See section	See section
268	Trace clock source select (TRACECLKSEL)	32	See section	See section
26C	CTimer 0 clock source select (CTIMERCLKSEL0)	32	See section	See section
270	CTimer 1 clock source select (CTIMERCLKSEL1)	32	See section	See section
274	CTimer 2 clock source select (CTIMERCLKSEL2)	32	See section	See section
278	CTimer 3 clock source select (CTIMERCLKSEL3)	32	See section	See section
27C	CTimer 4 clock source select (CTIMERCLKSEL4)	32	See section	See section
280	Main clock source select A (MAINCLKSELA)	32	See section	See section
284	Main clock source select B (MAINCLKSELB)	32	See section	See section
288	CLKOUT clock source select (CLKOUTSEL)	32	See section	See section
290	PLL0 clock source select (PLL0CLKSEL)	32	See section	See section
294	PLL1 clock source select (PLL1CLKSEL)	32	See section	See section
2A0	CAN clock source select (CANCLKSEL)	32	See section	See section
2A4	ADC0 clock source select (ADC0CLKSEL)	32	See section	See section
2A8	FS USB clock source select (USB0CLKSEL)	32	See section	See section
2B0	Flexcomm 0 clock source select for Fractional Rate Divider (FCCLKSEL0)	32	See section	See section

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
2B4	Flexcomm 1 clock source select for Fractional Rate Divider (FCCLKSEL1)	32	See section	See section
2B8	Flexcomm 2 clock source select for Fractional Rate Divider (FCCLKSEL2)	32	See section	See section
2BC	Flexcomm 3 clock source select for Fractional Rate Divider (FCCLKSEL3)	32	See section	See section
2C0	Flexcomm 4 clock source select for Fractional Rate Divider (FCCLKSEL4)	32	See section	See section
2C4	Flexcomm 5 clock source select for Fractional Rate Divider (FCCLKSEL5)	32	See section	See section
2C8	Flexcomm 6 clock source select for Fractional Rate Divider (FCCLKSEL6)	32	See section	See section
2CC	Flexcomm 7 clock source select for Fractional Rate Divider (FCCLKSEL7)	32	See section	See section
2D0	HS SPI clock source select (HSSPICLKSEL)	32	See section	See section
2E0	MCLK clock source select (MCLKCLKSEL)	32	See section	See section
2F0	SCTimer/PWM clock source select (SCTCLKSEL)	32	See section	See section
300	System Tick Timer divider for CPU0 (SYSTICKCLKDIV0)	32	See section	4000_0000
308	TRACE clock divider (TRACECLKDIV)	32	See section	4000_0000
30C	CAN clock divider (CANCLKDIV)	32	See section	4000_0000
320 - 33C	Fractional rate divider for flexcomm n (FRGCTRL0 - FRGCTRL7)	32	See section	See section
380	System clock divider (AHBCLKDIV)	32	See section	0000_0000
384	CLKOUT clock divider (CLKOUTDIV)	32	See section	4000_0000
388	FRO_HF (96MHz) clock divider (FROHFDIV)	32	See section	4000_0000
38C	WDT clock divider (WDTCLKDIV)	32	See section	See section

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
394	ADC0 clock divider (ADC0CLKDIV)	32	See section	See section
398	USB0-FS Clock divider (USB0CLKDIV)	32	See section	4000_0000
3AC	I2S MCLK clock divider (MCLKDIV)	32	See section	4000_0000
3B4	SCT/PWM clock divider (SCTCLKDIV)	32	See section	4000_0000
3C4	PLL clock divider (PLLCLKDIV)	32	See section	4000_0000
3D0 - 3E0	CTimer a clock divider (CTIMER0CLKDIV - CTIMER4CLKDIV)	32	See section	4000_0000
3FC	Clock configuration unlock (CLKUNLOCK)	32	RW	0000_0000
400	FMC configuration (FMCCR)	32	See section	0000_2000
404	ROM wait state (ROMCR)	32	See section	0000_0000
40C	USB0-FS need clock control (USB0NEEDCLKCTRL)	32	See section	See section
410	USB0-FS need clock status (USB0NEEDCLKSTAT)	32	See section	See section
41C	FMC flush control (FMCFLUSH)	32	WO	0000_0000
420	MCLK control (MCLKIO)	32	See section	See section
464	ADC1 clock source select (ADC1CLKSEL)	32	See section	See section
468	ADC1 clock divider (ADC1CLKDIV)	32	See section	See section
470	Control RAM interleave integration (RAM_INTERLEAVE)	32	See section	0000_0000
490	DAC0 functional clock selection (DAC0CLKSEL)	32	See section	0000_0007
494	DAC0 functional clock divider (DAC0CLKDIV)	32	See section	4000_0000
498	DAC1 functional clock selection (DAC1CLKSEL)	32	See section	0000_0007

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
49C	DAC1 functional clock divider (DAC1CLKDIV)	32	See section	4000_0000
4A0	DAC2 functional clock selection (DAC2CLKSEL)	32	See section	0000_0007
4A4	DAC2 functional clock divider (DAC2CLKDIV)	32	See section	4000_0000
4A8	FLEXSPI clock selection (FLEXSPICLKSEL)	32	See section	0000_000F
4AC	FLEXSPI clock divider (FLEXSPICLKDIV)	32	See section	4000_0000
4B0	Enable protection (CDPA_ENABLE)	32	See section	0000_0000
4B4	Enable protection duplicate (CDPA_ENABLE_DP)	32	See section	0000_0000
4B8	CDPA base address (CDPA_CONFIG)	32	See section	0000_0000
4D0	Flash hiding lockout address (FLASH_HIDING_LOCKOUT_ADDR)	32	RW	3CC3_5AA5
4D4	Flash hiding base address (FLASH_HIDING_BASE_ADDR)	32	See section	0003_FFFF
4D8	Flash hiding base DP address (FLASH_HIDING_BASE_DP_ADDR)	32	See section	0003_FFFF
4DC	Hiding size address (FLASH_HIDING_SIZE_ADDR)	32	See section	0000_0000
4E0	Hiding size DP address (FLASH_HIDING_SIZE_DP_ADDR)	32	See section	0000_0000
52C	PLL clock divider clock selection (PLLCLKDIVSEL)	32	See section	0000_0007
530	I3C functional clock selection (I3CFCLKSEL)	32	See section	0000_0007
534	I3C FCLK_STC clock selection (I3CFCLKSTCSEL)	32	See section	0000_0007
538	I3C FCLK_STC clock divider (I3CFCLKSTCDIV)	32	See section	4000_0000
53C	I3C FCLKS clock divider (I3CFCLKSDIV)	32	See section	4000_0000
540	I3C FCLK divider (I3CFCLKDIV)	32	See section	4000_0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
544	I3C FCLK_S selection (I3CFCLKSSEL)	32	See section	0000_0000
548	DMIC clock selection (DMICFCLKSEL)	32	See section	0000_0007
54C	DMIC clock division (DMICFCLKDIV)	32	See section	4000_0000
560	PLL1 550m control (PLL1CTRL)	32	See section	See section
564	PLL1 550m status (PLL1STAT)	32	See section	See section
568	PLL1 550m N divider (PLL1NDEC)	32	See section	See section
56C	PLL1 550m M divider (PLL1MDEC)	32	See section	See section
570	PLL1 550m P divider (PLL1PDEC)	32	See section	See section
580	PLL0 550m control (PLL0CTRL)	32	See section	See section
584	PLL0 550m status (PLL0STAT)	32	See section	See section
588	PLL0 550m N divider (PLL0NDEC)	32	See section	See section
58C	PLL0 550m P divider (PLL0PDEC)	32	See section	See section
590	PLL0 Spread Spectrum control 0 (PLL0SSCG0)	32	RW	0000_0000
594	PLL0 Spread Spectrum control 1 (PLL0SSCG1)	32	See section	See section
5D0	DAC Isolation Control (DAC_ISO_CTRL)	32	See section	0000_0000
680	Start logic wake-up enable (STARTER0)	32	See section	See section
684	Start logic wake-up enable (STARTER1)	32	See section	See section
688	Start logic wake-up enable (STARTER2)	32	See section	0000_0000
68C	Start logic wake-up enable (STARTER3)	32	See section	0000_0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
6A0	Set bits in STARTER (STARTERSET0)	32	See section	See section
6A4	Set bits in STARTER (STARTERSET1)	32	See section	See section
6A8	Set bits in STARTER (STARTERSET2)	32	See section	0000_0000
6AC	Set bits in STARTER (STARTERSET3)	32	See section	0000_0000
6C0	Clear bits in STARTER (STARTERCLR0)	32	See section	See section
6C4	Clear bits in STARTER (STARTERCLR1)	32	See section	See section
704	Functional retention control (FUNCRETENTIONCTRL)	32	See section	See section
780	Hardware Sleep control (HARDWARESLEEP)	32	See section	See section
80C	CPU Status (CPUSTAT)	32	See section	See section
824	LPCAC control (LPCAC_CTRL)	32	See section	0000_0001
82C	Flexcomm 32K clock select (FC32KCLKSEL)	32	See section	0000_0000
830 - 84C	FRG Clock Source Select (FRGCLKSEL0 - FRGCLKSEL7)	32	See section	0000_0007
850 - 86C	Flexcomm clock divider (FLEXCOMM0CLKDIV - FLEXCOMM7CLKDIV)	32	See section	4000_0000
A18	Clock Control (CLOCK_CTRL)	32	See section	See section
B10	Comparator Interrupt control (COMP_INT_CTRL)	32	See section	See section
B14	Comparator Interrupt status (COMP_INT_STATUS)	32	See section	See section
E04	Control automatic clock gating (AUTOCLKGATEOVERRIDE)	32	See section	0000_FFFF
E08	GPIO Synchronization (GPIOPSYNC)	32	See section	See section

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
E24	Control automatic clock gating (AUTOCLKGATEOVERRIDE1)	32	See section	0000_0000
E30	Memory parity ECC enable (ENABLE_MEM_PARITY_ECC_CHECK)	32	See section	0000_0000
E34	Memory parity ECC error flag (MEM_PARITY_ECC_ERROR_FLAG)	32	See section	0000_0000
E38	PWM0 submodule control (PWM0SUBCTL)	32	See section	0000_0000
E3C	PWM1 submodule control (PWM1SUBCTL)	32	See section	0000_0000
E40	CTIMER global start enable (CTIMERGLOBALSTARTEN)	32	See section	0000_0000
FA0	Control write access to security (DEBUG_LOCK_EN)	32	See section	See section
FA4	Cortex debug features control (DEBUG_FEATURES)	32	See section	See section
FA8	Cortex debug features control (duplicate) (DEBUG_FEATURES_DP)	32	See section	See section
FB4	CPU0 Software Debug Access (SWD_ACCESS_CPU0)	32	RW	0000_0000
FC4	DSP Software Debug Access (SWD_ACCESS_DSP)	32	RW	0000_0000
FF8	Device ID (DEVICE_ID0)	32	RO	0000_0000
FFC	Chip revision ID and Number (DIEID)	32	RO	See section

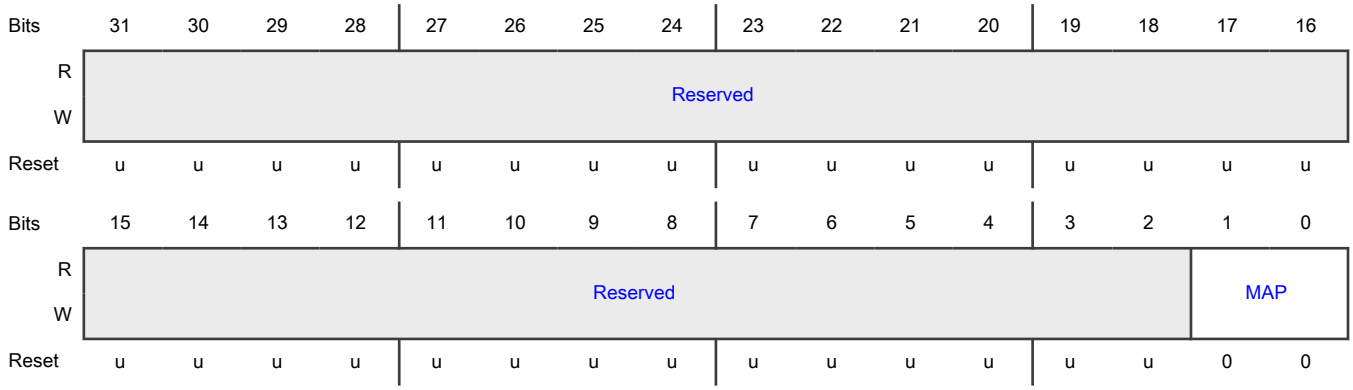
8.5.1.1.1 Memory Remap Control (MEMORYREMAP)

The memory remap control selects the memory location of the vector table.

Offset

Register	Offset
MEMORYREMAP	0h

Diagram



Fields

Field	Description
31-2 —	Reserved
1-0 MAP	Select the location of the vector table: 00 - Vector Table in ROM. 01 - Vector Table in RAM. 1x - Vector Table in Flash.

8.5.1.1.2 AHB Matrix priority control (AHBMATPRIO)

The Multilayer AHB Matrix arbitrates between masters, when they attempt to access the same matrix slave port at the same time. Priority values are 3 = highest, 0 = lowest. When the priority is the same, the master with the lower master number is given priority.

NOTE

Care should be taken if the value in this register is changed, improper settings can seriously degrade performance.

Offset

Register	Offset
AHBMATPRIO	10h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				PRI_MCAN		PRI_SDMA1		PRI_USB_FS		Reserved		PRI_PQ		Reserved	
W			Reserved													
Reset	u	u	u	u	u	u	0	0	0	0	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				PRI_SDMA0		PRI_USB_FSD		Reserved		Reserved		PRI_CPU0_SBUS		PRI_CPU0_CBUS	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-30 —	Reserved
29-28 —	Reserved Read value is undefined, only zero should be written.
27-26 PRI_MCAN	MCAN
25-24 PRI_SDMA1	DMA1 controller priority.
23-22 PRI_USB_FS	USB-FS host
21-20 —	Reserved
19-18 PRI_PQ	PQ (HW Accelerator).
17-16 —	Reserved
15-12 —	Reserved
11-10 PRI_SDMA0	DMA0 controller priority.

Table continues on the next page...

Table continued from the previous page...

Field	Description
9-8 PRI_USB_FSD	USB0-FS Device.(USB0)
7-6 —	Reserved
5-4 —	Reserved
3-2 PRI_CPU0_SBUS	CPU0 S-AHB bus.
1-0 PRI_CPU0_CBUS	CPU0 C-AHB bus.

8.5.1.1.3 AHB Matrix priority control (AHBMATPRIO1)

Priority values are:

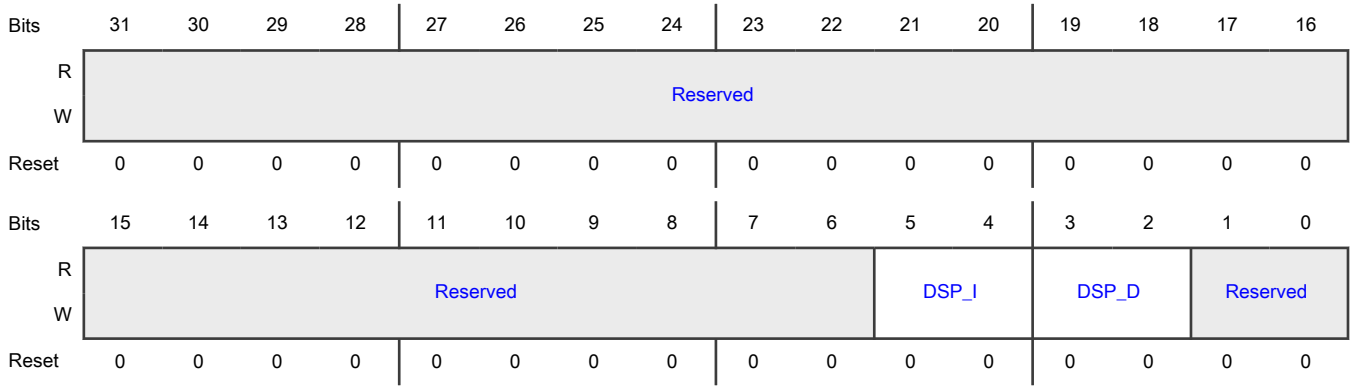
3 = highest,

0 = lowest

Offset

Register	Offset
AHBMATPRIO1	14h

Diagram



Fields

Field	Description
31-6 —	Reserved
5-4 DSP_I	DSP I bus
3-2 DSP_D	DSP D bus
1-0 —	Reserved

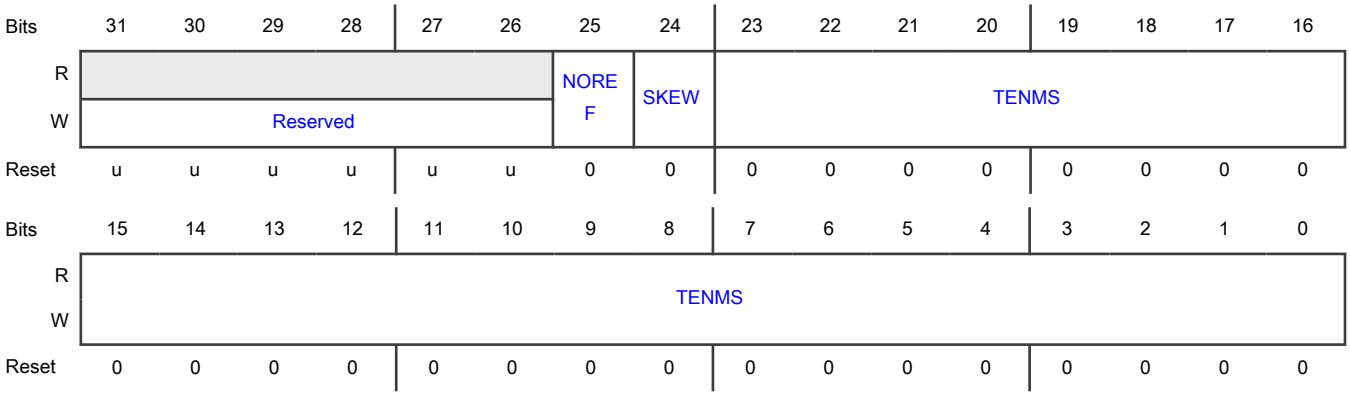
8.5.1.1.4 System tick calibration for non-secure part of CPU0 (CPU0NSTCKCAL)

CPU0NSTCKCAL register allows software to set up a default value for the SYST_CALIB register in the System Tick Timer of non-secure part of the CPU0.

Offset

Register	Offset
CPU0NSTCKCAL	3Ch

Diagram



Fields

Field	Description
31-26 —	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
25 NOREF	Indicates whether the device provides a reference clock to the processor: 0 - Reference clock provided 1 - No reference clock provided
24 SKEW	Indicates whether the TENMS value is exact: 0 - TENMS value is exact 1 - TENMS value is inexact, or not given
23-0 TENMS	Reload value for 10 ms (100 Hz) timing, subject to system clock skew errors. If the value reads as zero, the calibration value is not known.

8.5.1.1.5 NMI Source Select (NMISRC)

Offset

Register	Offset
NMISRC	48h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NMIENCP...	Reserved														
W			Reserved													
Reset	0	0	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																
W	Reserved		Reserved						Reserved		IRQCPU0					
Reset	u	u	0	0	0	0	0	0	u	u	0	0	0	0	0	0

Fields

Field	Description
31 NMIENCPU0	Write a 1 to this bit to enable the Non-Maskable Interrupt (NMI) source selected by IRQCPU0.
30 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
29-14 —	Reserved Read value is undefined, only zero should be written.
13-8 —	Reserved
7-6 —	Reserved Read value is undefined, only zero should be written.
5-0 IRQCPU0	The IRQ number of the interrupt that acts as the Non-Maskable Interrupt (NMI) for the CPU0, if enabled by NMIENCPU0.

8.5.1.1.6 Peripheral reset control 0 (PRESETCTRL0)

The PRESETCTRL0 register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

NOTE

When modifying the PRESETCTRL registers, use the related PRESETCTRLSET and PRESETCTRLCLR registers to avoid setting or clearing bits unintentionally.

Offset

Register	Offset
PRESETCTRL0	100h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		DAC0_RST	ADC1_RST	ADC0_RST	MAILBOX...	Reserved		RTC_RST	WWDT_RST	CRCGEN_...	DMA0_RST	GINT_RST	PINT_RST	GPIO3_R...	GPIO2_R...
Reset	u	u	u	u	0	0	u	u	0	0	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	GPIO1_R...	GPIO0_R...	IOCONN_R...	Reserved	Reserved	FLEXSPI...	Reserved	FMC_RST	FLASH_R...	SRAM_CT...	SRAM_CT...	SRAM_CT...	SRAM_CT...	Reserved	Reserved	Reserved
Reset	0	0	0	u	0	u	u	0	0	0	0	0	0	u	0	u

Fields

Field	Description
31-30 —	Reserved
29 DAC0_RST	DAC0 reset control. 0 - Block is not reset. 1 - Block is reset.
28 ADC1_RST	ADC1 reset control. 0 - Block is not reset. 1 - Block is reset.
27 ADC0_RST	ADC0 reset control. 0 - Block is not reset. 1 - Block is reset.
26 MAILBOX_RST	Inter CPU communication Mailbox reset control. 0 - Block is not reset. 1 - Block is reset.
25-24 —	Reserved Read value is undefined, only zero should be written.
23 RTC_RST	Real Time Clock (RTC) reset control. 0 - Block is not reset. 1 - Block is reset.
22 WWDT_RST	Watchdog Timer reset control. 0 - Block is not reset. 1 - Block is reset.
21 CRCGEN_RST	CRCGEN reset control. 0 - Block is not reset. 1 - Block is reset.
20 DMA0_RST	DMA0 reset control. 0 - Block is not reset. 1 - Block is reset.
19 GINT_RST	Group interrupt (GINT) reset control. 0 - Block is not reset. 1 - Block is reset.

Table continues on the next page...

Table continued from the previous page...

Field	Description
18 PINT_RST	Pin interrupt (PINT) reset control. 0 - Block is not reset. 1 - Block is reset.
17 GPIO3_RST	GPIO3 reset control. 0 - Block is not reset. 1 - Block is reset.
16 GPIO2_RST	GPIO2 reset control. 0 - Block is not reset. 1 - Block is reset.
15 GPIO1_RST	GPIO1 reset control. 0 - Block is not reset. 1 - Block is reset.
14 GPIO0_RST	GPIO0 reset control. 0 - Block is not reset. 1 - Block is reset.
13 IOCON_RST	I/O controller reset control. 0 - Block is not reset. 1 - Block is reset.
12 —	Reserved Read value is undefined, only zero should be written.
11 —	Reserved
10 FLEXSPI_RST	FLEXSPI reset control 0 - Block is not reset. 1 - Block is reset.
9 —	Reserved
8 FMC_RST	FMC controller reset control. 0 - Block is not reset. 1 - Block is reset.
7	Flash controller reset control.

Table continues on the next page...

Table continued from the previous page...

Field	Description
FLASH_RST	0 - Block is not reset. 1 - Block is reset.
6 SRAM_CTRL4_RST	SRAM Controller 4 reset control. 0 - Block is not reset. 1 - Block is reset.
5 SRAM_CTRL3_RST	SRAM Controller 3 reset control. 0 - Block is not reset. 1 - Block is reset.
4 SRAM_CTRL2_RST	SRAM Controller 2 reset control. 0 - Block is not reset. 1 - Block is reset.
3 SRAM_CTRL1_RST	SRAM Controller 1 reset control. 0 - Block is not reset. 1 - Block is reset.
2 —	Reserved Read value is undefined, only zero should be written.
1 —	Reserved
0 —	Reserved Read value is undefined, only zero should be written.

8.5.1.1.7 Peripheral reset control 1 (PRESETCTRL1)

The PRESETCTRL1 register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

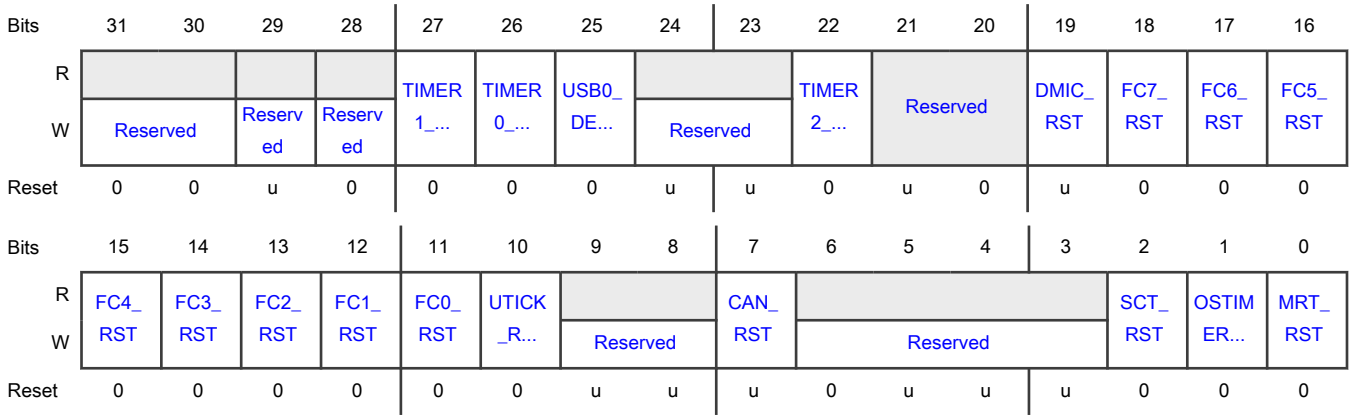
NOTE

When modifying the PRESETCTRL registers, use the related PRESETCTRLSET and PRESETCTRLCLR registers to avoid setting or clearing bits unintentionally.

Offset

Register	Offset
PRESETCTRL1	104h

Diagram



Fields

Field	Description
31-30 —	Reserved Read value is undefined, only zero should be written.
29 —	Reserved Read value is undefined, only zero should be written.
28 —	Reserved Read value is undefined, only zero should be written.
27 TIMER1_RST	Timer 1 reset control. 0 - Block is not reset. 1 - Block is reset.
26 TIMER0_RST	Timer 0 reset control. 0 - Block is not reset. 1 - Block is reset.
25 USB0_DEV_RST	USB0-FS DEV reset control. 0 - Block is not reset. 1 - Block is reset.
24-23 —	Reserved Read value is undefined, only zero should be written.
22 TIMER2_RST	Timer 2 reset control. 0 - Block is not reset. 1 - Block is reset.
21-20	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
19 DMIC_RST	DMIC reset control. 0 - Block is not reset. 1 - Block is reset.
18 FC7_RST	FC7 reset control. 0 - Block is not reset. 1 - Block is reset.
17 FC6_RST	FC6 reset control. 0 - Block is not reset. 1 - Block is reset.
16 FC5_RST	FC5 reset control. 0 - Block is not reset. 1 - Block is reset.
15 FC4_RST	FC4 reset control. 0 - Block is not reset. 1 - Block is reset.
14 FC3_RST	FC3 reset control. 0 - Block is not reset. 1 - Block is reset.
13 FC2_RST	FC2 reset control. 0 - Block is not reset. 1 - Block is reset.
12 FC1_RST	FC1 reset control. 0 - Block is not reset. 1 - Block is reset.
11 FC0_RST	FC0 reset control. 0 - Block is not reset. 1 - Block is reset.
10 UTICK_RST	UTICK reset control. 0 - Block is not reset. 1 - Block is reset.

Table continues on the next page...

Table continued from the previous page...

Field	Description
9-8 —	Reserved Read value is undefined, only zero should be written.
7 CAN_RST	CAN reset control. 0 - Block is not reset. 1 - Block is reset.
6-3 —	Reserved Read value is undefined, only zero should be written.
2 SCT_RST	SCT reset control. 0 - Block is not reset. 1 - Block is reset.
1 OSTIMER_RST	OS Event Timer reset control. 0 - Block is not reset. 1 - Block is reset.
0 MRT_RST	MRT reset control. 0 - Block is not reset. 1 - Block is reset.

8.5.1.1.8 Peripheral reset control 2 (PRESETCTRL2)

The PRESETCTRL2 register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

NOTE

When modifying the PRESETCTRL registers, use the related PRESETCTRLSET and PRESETCTRLCLR registers to avoid setting or clearing bits unintentionally.

Offset

Register	Offset
PRESETCTRL2	108h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		0		HS_SPI_...	ANACTRL...			Reserved	Reserved	TIMER4_...	TIMER3_...	Reserved	PQ_RST	Reserved	USB0_HO...	USB0_HO...
W	Reserved					Reserved	Reserved									
Reset	u	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SYSCTL_...		0		Reserved			FREQME_...	Reserved				Reserved	COMP_RST	DMA1_RST	
W		Reserved		Reserved		Reserved										Reserved
Reset	u	0	0	0	u	0	0	0	0	0	0	0	0	0	0	u

Fields

Field	Description
31 —	Reserved Read value is undefined, only zero should be written.
30-29 —	Reserved
28 HS_SPI_RST	HS SPI reset control. 0 - Block is not reset. 1 - Block is reset.
27 ANACTRL_RST	Analog control reset control. 0 - Block is not reset. 1 - Block is reset.
26 —	Reserved Read value is undefined, only zero should be written.
25 —	Reserved Read value is undefined, only zero should be written.
24 —	Reserved
23 —	Reserved
22 TIMER4_RST	Timer 4 reset control. 0 - Block is not reset.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Block is reset.
21 TIMER3_RST	Timer 3 reset control. 0 - Block is not reset. 1 - Block is reset.
20 —	Reserved
19 PQ_RST	Power Quad reset control. 0 - Block is not reset. 1 - Block is reset.
18 —	Reserved
17 USB0_HOSTS_RST	USB0-FS Host Slave reset control. 0 - Block is not reset. 1 - Block is reset.
16 USB0_HOSTM_RST	USB0-FS Host Master reset control. 0 - Block is not reset. 1 - Block is reset.
15 SYSCTL_RST	SYSCTL Block reset. 0 - Block is not reset. 1 - Block is reset.
14 —	Reserved Read value is undefined, only zero should be written.
13 —	Reserved
12 —	Reserved Read value is undefined, only zero should be written.
11 —	Reserved
10-9 —	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
8 FREQME_RST	Frequency meter reset control. 0 - Block is not reset. 1 - Block is reset.
7-4 —	Reserved
3 —	Reserved
2 COMP_RST	Analog comparator reset control. 0 - Block is not reset. 1 - Block is reset.
1 DMA1_RST	DMA1 reset control. 0 - Block is not reset. 1 - Block is reset.
0 —	Reserved Read value is undefined, only zero should be written.

8.5.1.1.9 Peripheral reset control 3 (PRESETCTRL3)

The PRESETCTRL3 register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

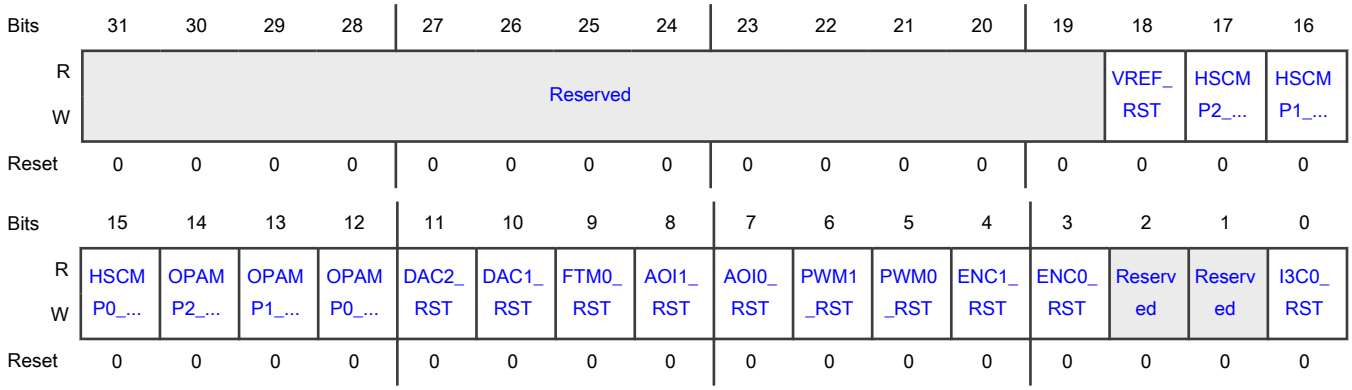
NOTE

When modifying the PRESETCTRL registers, use the related PRESETCTRLSET and PRESETCTRLCLR registers to avoid setting or clearing bits unintentionally.

Offset

Register	Offset
PRESETCTRL3	10Ch

Diagram



Fields

Field	Description
31-19 —	Reserved
18 VREF_RST	VREF reset control 0 - Block is not reset. 1 - Block is reset.
17 HSCMP2_RST	HSCMP2 reset control 0 - Block is not reset. 1 - Block is reset.
16 HSCMP1_RST	HSCMP1 reset control 0 - Block is not reset. 1 - Block is reset.
15 HSCMP0_RST	HSCMP0 reset control 0 - Block is not reset. 1 - Block is reset.
14 OPAMP2_RST	OPAMP2 reset control 0 - Block is not reset. 1 - Block is reset.
13 OPAMP1_RST	OPAMP1 reset control 0 - Block is not reset. 1 - Block is reset.
12 OPAMP0_RST	OPAMP0 reset control 0 - Block is not reset.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Block is reset.
11 DAC2_RST	DAC2 reset control 0 - Block is not reset. 1 - Block is reset.
10 DAC1_RST	DAC1 reset control 0 - Block is not reset. 1 - Block is reset.
9 FTM0_RST	FTM0 reset control 0 - Block is not reset. 1 - Block is reset.
8 AOI1_RST	AOI1 reset control 0 - Block is not reset. 1 - Block is reset.
7 AOI0_RST	AOI0 reset control 0 - Block is not reset. 1 - Block is reset.
6 PWM1_RST	PWM1 reset control 0 - Block is not reset. 1 - Block is reset.
5 PWM0_RST	PWM0 reset control 0 - Block is not reset. 1 - Block is reset.
4 ENC1_RST	ENC1 reset control 0 - Block is not reset. 1 - Block is reset.
3 ENC0_RST	ENC0 reset control 0 - Block is not reset. 1 - Block is reset.
2 —	Reserved
1	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
0 I3C0_RST	I3C reset control 0 - Block is not reset. 1 - Block is reset.

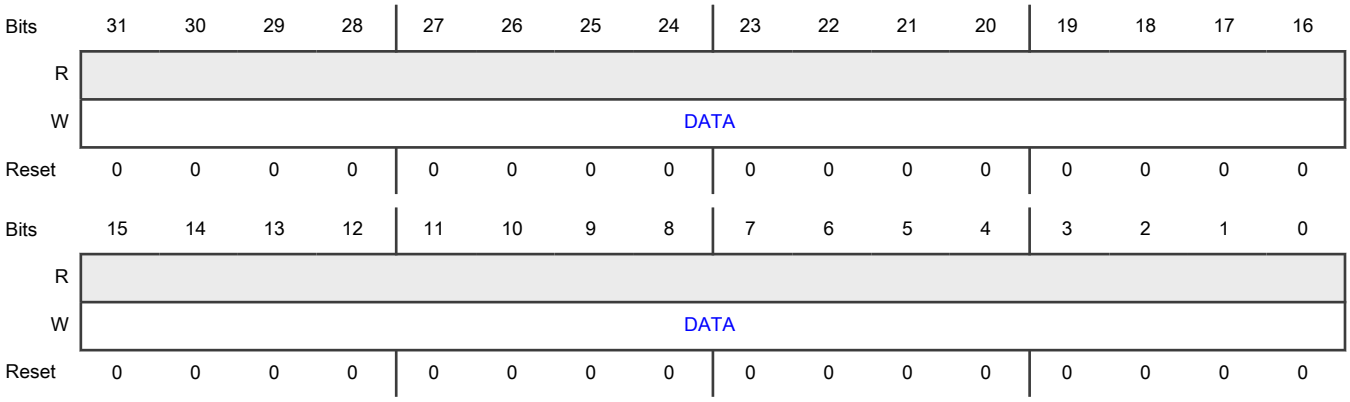
8.5.1.1.10 Peripheral reset control set n (PRESETCTRLSET0 - PRESETCTRLSET1)

Writing a 1 to a bit position in a write-only PRESETCTRLSETn register sets the corresponding position in PRESETCTRLn.

Offset

Register	Offset
PRESETCTRLSET0	120h
PRESETCTRLSET1	124h

Diagram



Fields

Field	Description
31-0 DATA	Data array value

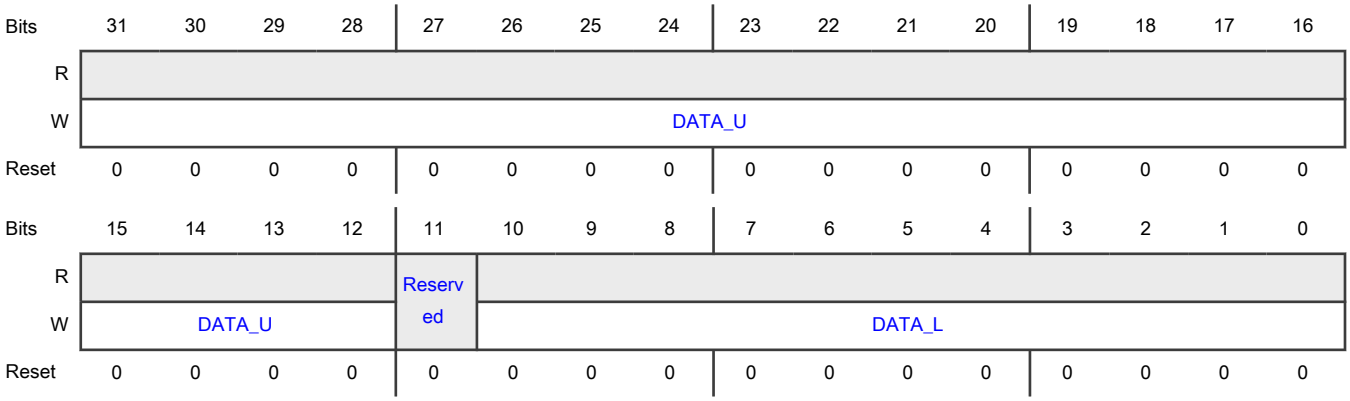
8.5.1.1.11 Peripheral reset control set n (PRESETCTRLSET2)

Writing a 1 to a bit position in a write-only PRESETCTRLSETn register sets the corresponding position in PRESETCTRLn.

Offset

Register	Offset
PRESETCTRLSET2	128h

Diagram



Fields

Field	Description
31-12 DATA_U	Data array value
11 —	Reserved
10-0 DATA_L	Data array value

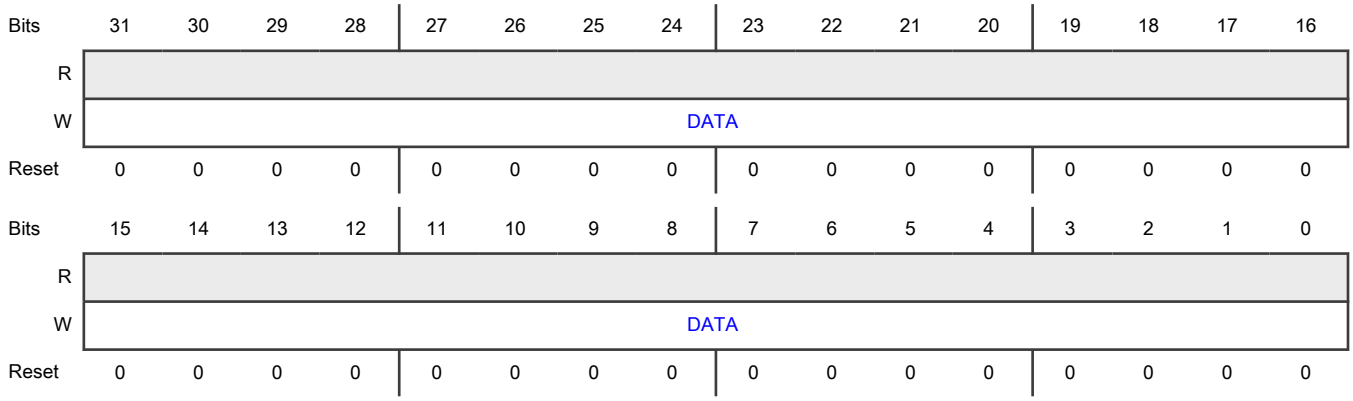
8.5.1.1.12 Peripheral reset control set n (PRESETCTRLSET3)

Writing a 1 to a bit position in a write-only PRESETCTRLSETn register sets the corresponding position in PRESETCTRLn.

Offset

Register	Offset
PRESETCTRLSET3	12Ch

Diagram



Fields

Field	Description
31-0 DATA	Data array value

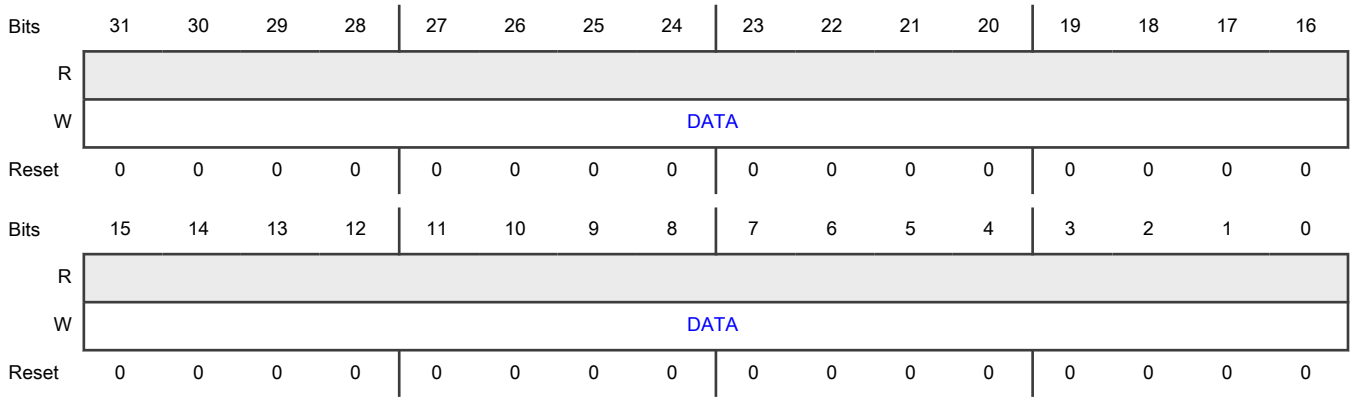
8.5.1.1.13 Peripheral reset control clear n (PRESETCTRLCLR0 - PRESETCTRLCLR1)

Writing a 1 to a bit position in a write-only PRESETCTRLCLRn register clears the corresponding position in PRESETCTRLn.

Offset

Register	Offset
PRESETCTRLCLR0	140h
PRESETCTRLCLR1	144h

Diagram



Fields

Field	Description
31-0 DATA	Data array value

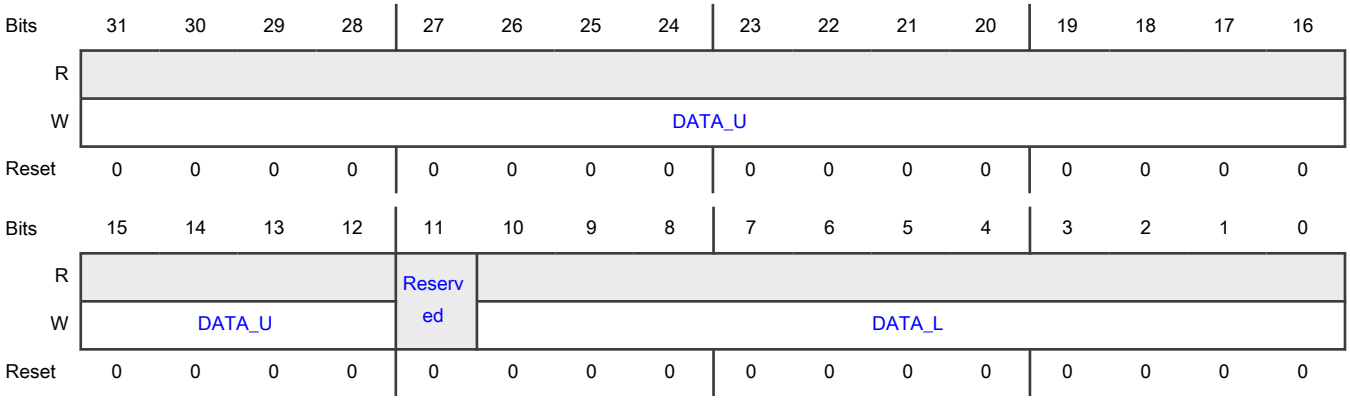
8.5.1.1.14 Peripheral reset control clear n (PRESETCTRLCLR2)

Writing a 1 to a bit position in a write-only PRESETCTRLCLRn register clears the corresponding position in PRESETCTRLn.

Offset

Register	Offset
PRESETCTRLCLR2	148h

Diagram



Fields

Field	Description
31-12 DATA_U	Data array value
11 —	Reserved
10-0 DATA_L	Data array value

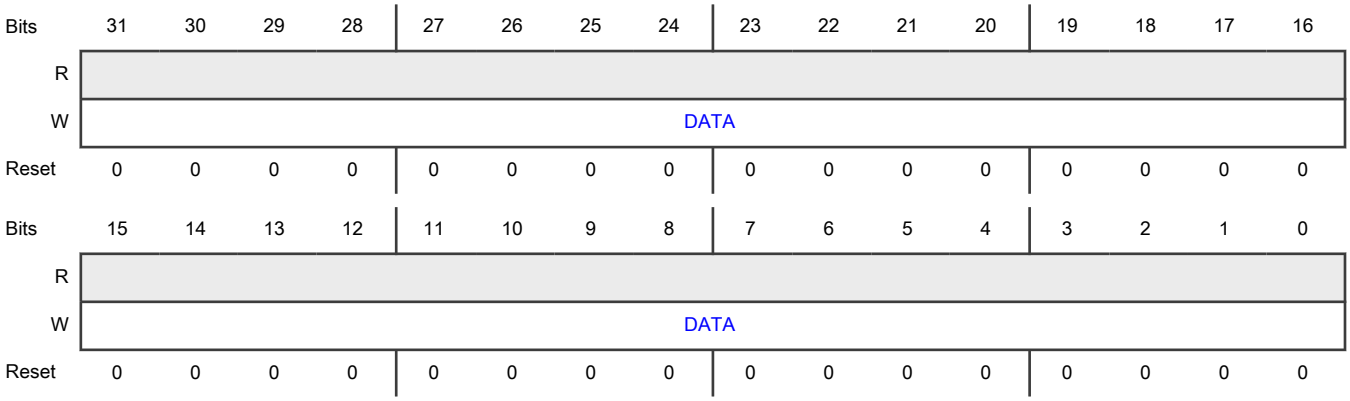
8.5.1.1.15 Peripheral reset control clear n (PRESETCTRLCLR3)

Writing a 1 to a bit position in a write-only PRESETCTRLCLRn register clears the corresponding position in PRESETCTRLn.

Offset

Register	Offset
PRESETCTRLCLR3	14Ch

Diagram



Fields

Field	Description
31-0 DATA	Data array value

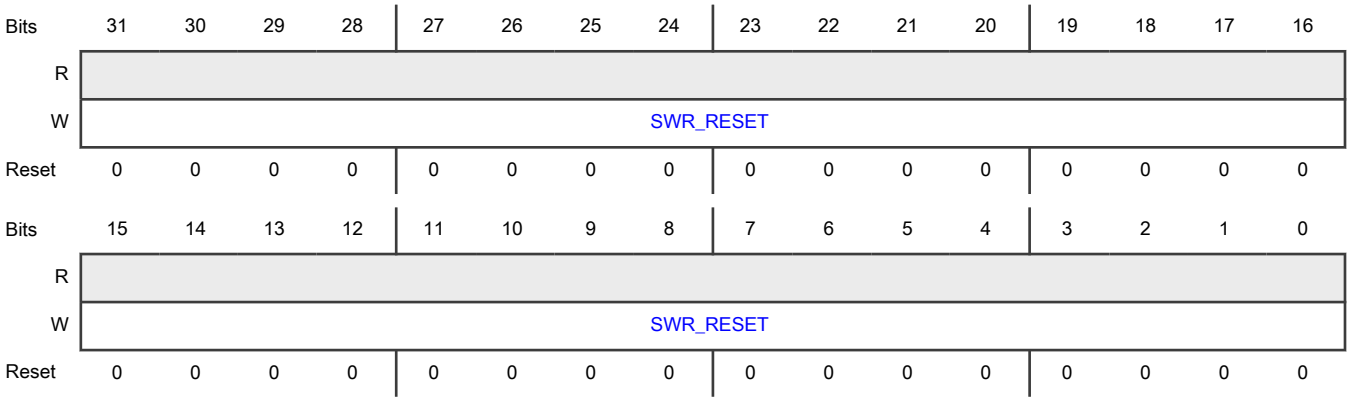
8.5.1.1.16 Software Reset (SWR_RESET)

Write 0x5A00_0001 to generate a software_reset

Offset

Register	Offset
SWR_RESET	160h

Diagram



Fields

Field	Description
31-0 SWR_RESET	Write 0x5A00_0001 to generate a software_reset.

8.5.1.1.17 AHB Clock control 0 (AHBCLKCTRL0)

The AHBCLKCTRLn registers enable the clocks to individual modules.

NOTE

When modifying the AHBCLKCTRL registers, use the related AHBCLKCTRLSET and AHBCLKCTRLCLR registers to avoid setting or clearing bits unintentionally.

Offset

Register	Offset
AHBCLKCTRL0	200h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		DAC0	ADC1	ADC0	MAILBOX	Reserved		RTC	WWDT	CRCGEN	DMA0	GINT	PINT	GPIO3	GPIO2
Reset	u	u	u	u	0	0	u	u	0	0	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	GPIO1	GPIO0	IOCON	Reserved	MUX	FLEXSPI	Reserved	FMC	FLASH	SRAM_CT...	SRAM_CT...	SRAM_CT...	SRAM_CT...	Reserved	ROM	Reserved
Reset	0	0	0	u	0	u	u	1	1	0	0	0	0	u	0	u

Fields

Field	Description
31-30 —	Reserved
29 DAC0	Enables the clock for DAC0. 0 - Disable Clock. 1 - Enable Clock.
28	Enables the clock for ADC1.

Table continues on the next page...

Table continued from the previous page...

Field	Description
ADC1	0 - Disable Clock. 1 - Enable Clock.
27 ADC0	Enables the clock for ADC0. 0 - Disable Clock. 1 - Enable Clock.
26 MAILBOX	Enables the clock for the Inter CPU communication Mailbox. 0 - Disable Clock. 1 - Enable Clock.
25-24 —	Reserved Read value is undefined, only zero should be written.
23 RTC	Enables the clock for the Real Time Clock (RTC). 0 - Disable Clock. 1 - Enable Clock.
22 WWDT	Enables the clock for the Watchdog Timer. 0 - Disable Clock. 1 - Enable Clock.
21 CRCGEN	Enables the clock for the CRCGEN. 0 - Disable Clock. 1 - Enable Clock.
20 DMA0	Enables the clock for the DMA0. 0 - Disable Clock. 1 - Enable Clock.
19 GINT	Enables the clock for the Group interrupt (GINT). 0 - Disable Clock. 1 - Enable Clock.
18 PINT	Enables the clock for the Pin interrupt (PINT). 0 - Disable Clock. 1 - Enable Clock.
17 GPIO3	Enables the clock for the GPIO3. 0 - Disable Clock. 1 - Enable Clock.

Table continues on the next page...

Table continued from the previous page...

Field	Description
16 GPIO2	Enables the clock for the GPIO2. 0 - Disable Clock. 1 - Enable Clock.
15 GPIO1	Enables the clock for the GPIO1. 0 - Disable Clock. 1 - Enable Clock.
14 GPIO0	Enables the clock for the GPIO0. 0 - Disable Clock. 1 - Enable Clock.
13 IOCON	Enables the clock for the I/O controller. 0 - Disable Clock. 1 - Enable Clock.
12 —	Reserved Read value is undefined, only zero should be written.
11 MUX	Enables the clock for the Input Mux. 0 - Disable Clock. 1 - Enable Clock.
10 FLEXSPI	Enables the clock for the Flexspi. 0 - Disable Clock. 1 - Enable Clock.
9 —	Reserved
8 FMC	Enables the clock for the FMC controller. 0 - Disable Clock. 1 - Enable Clock.
7 FLASH	Enables the clock for the Flash controller. 0 - Disable Clock. 1 - Enable Clock.
6 SRAM_CTRL4	Enables the clock for the SRAM Controller 4. 0 - Disable Clock. 1 - Enable Clock.

Table continues on the next page...

Table continued from the previous page...

Field	Description
5 SRAM_CTRL3	Enables the clock for the SRAM Controller 3. 0 - Disable Clock. 1 - Enable Clock.
4 SRAM_CTRL2	Enables the clock for the SRAM Controller 2. 0 - Disable Clock. 1 - Enable Clock.
3 SRAM_CTRL1	Enables the clock for the SRAM Controller 1. 0 - Disable Clock. 1 - Enable Clock.
2 —	Reserved Read value is undefined, only zero should be written.
1 ROM	Enables the clock for the ROM. 0 - Disable Clock. 1 - Enable Clock.
0 —	Reserved Read value is undefined, only zero should be written.

8.5.1.1.18 AHB Clock control 1 (AHBCLKCTRL1)

The AHBCLKCTRLn registers enable the clocks to individual modules.

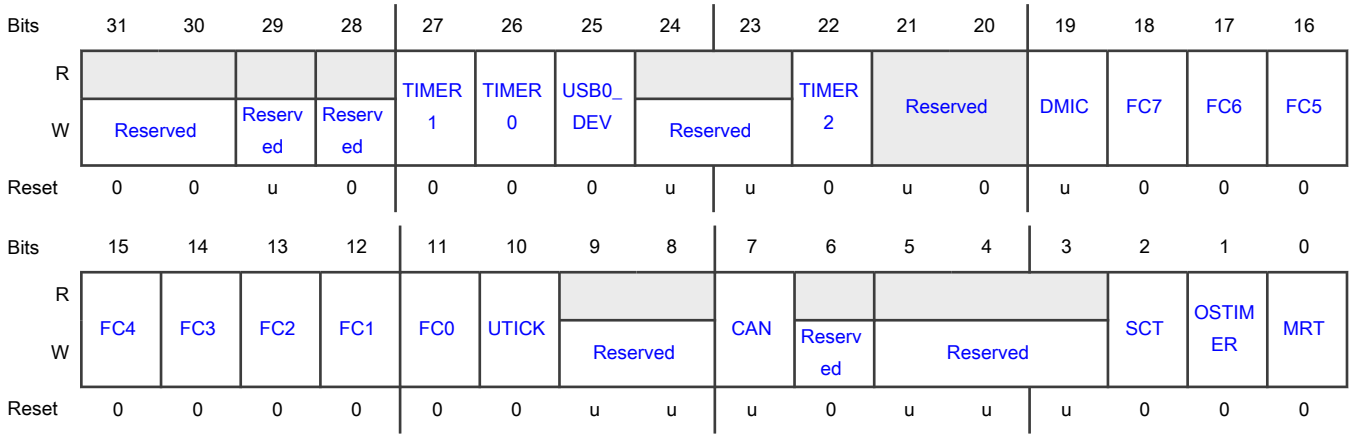
NOTE

When modifying the AHBCLKCTRL registers, use the related AHBCLKCTRLSET and AHBCLKCTRLCLR registers to avoid setting or clearing bits unintentionally.

Offset

Register	Offset
AHBCLKCTRL1	204h

Diagram



Fields

Field	Description
31-30 —	Reserved Read value is undefined, only zero should be written.
29 —	Reserved Read value is undefined, only zero should be written.
28 —	Reserved Read value is undefined, only zero should be written.
27 TIMER1	Enables the clock for the Timer 1. 0 - Disable Clock. 1 - Enable Clock.
26 TIMER0	Enables the clock for the Timer 0. 0 - Disable Clock. 1 - Enable Clock.
25 USB0_DEV	Enables the clock for the USB0-FS device. 0 - Disable Clock. 1 - Enable Clock.
24-23 —	Reserved Read value is undefined, only zero should be written.
22 TIMER2	Enables the clock for the Timer 2. 0 - Disable Clock. 1 - Enable Clock.

Table continues on the next page...

Table continued from the previous page...

Field	Description
21-20 —	Reserved
19 DMIC	Enables the clock for DMIC. 0 - Disable Clock. 1 - Enable Clock.
18 FC7	Enables the clock for the FC7. 0 - Disable Clock. 1 - Enable Clock.
17 FC6	Enables the clock for the FC6. 0 - Disable Clock. 1 - Enable Clock.
16 FC5	Enables the clock for the FC5. 0 - Disable Clock. 1 - Enable Clock.
15 FC4	Enables the clock for the FC4. 0 - Disable Clock. 1 - Enable Clock.
14 FC3	Enables the clock for the FC3. 0 - Disable Clock. 1 - Enable Clock.
13 FC2	Enables the clock for the FC2. 0 - Disable Clock. 1 - Enable Clock.
12 FC1	Enables the clock for the FC1. 0 - Disable Clock. 1 - Enable Clock.
11 FC0	Enables the clock for the FC0. 0 - Disable Clock. 1 - Enable Clock.
10 UTICK	Enables the clock for the UTICK. 0 - Disable Clock. 1 - Enable Clock.

Table continues on the next page...

Table continued from the previous page...

Field	Description
9-8 —	Reserved Read value is undefined, only zero should be written.
7 CAN	Enables the clock for the CAN. 0 - Disable Clock. 1 - Enable Clock.
6 —	Reserved Read value is undefined, only zero should be written.
5-3 —	Reserved Read value is undefined, only zero should be written.
2 SCT	Enables the clock for the SCT. 0 - Disable Clock. 1 - Enable Clock.
1 OSTIMER	Enables the clock for the OS Event Timer. 0 - Disable Clock. 1 - Enable Clock.
0 MRT	Enables the clock for the MRT. 0 - Disable Clock. 1 - Enable Clock.

8.5.1.1.19 AHB Clock control 2 (AHBCLKCTRL2)

The AHBCLKCTRLn registers enable the clocks to individual modules.

NOTE

When modifying the AHBCLKCTRL registers, use the related AHBCLKCTRLSET and AHBCLKCTRLCLR registers to avoid setting or clearing bits unintentionally.

Offset

Register	Offset
AHBCLKCTRL2	208h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		0		HS_SPI	ANALOG_...			Reserved	Reserved	TIMER4	TIMER3	Reserved	PQ	Reserved	USB0_HO...	USB0_HO...
W	Reserved					Reserved	Reserved									
Reset	u	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SYSCTL	PMUX1			Reserved			FREQME	Reserved				Reserved	COMP	DMA1	
W			Reserved	Reserved		Reserved										Reserved
Reset	u	0	0	0	u	0	0	0	0	0	0	0	0	0	0	u

Fields

Field	Description
31 —	Reserved Read value is undefined, only zero should be written.
30-29 —	Reserved
28 HS_SPI	Enables the clock for the HS SPI. 0 - Disable Clock. 1 - Enable Clock.
27 ANALOG_CTRL	Enables the clock for the Analog Controller block. 0 - Disable Clock. 1 - Enable Clock.
26 —	Reserved Read value is undefined, only zero should be written.
25 —	Reserved Read value is undefined, only zero should be written.
24 —	Reserved
23 —	Reserved
22 TIMER4	Enables the clock for the Timer 4. 0 - Disable Clock.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Enable Clock.
21 TIMER3	Enables the clock for the Timer 3. 0 - Disable Clock. 1 - Enable Clock.
20 —	Reserved
19 PQ	Enables the clock for the Power Quad. 0 - Disable Clock. 1 - Enable Clock.
18 —	Reserved
17 USB0_HOSTS	Enables the clock for the USB0-FS Host Slave. 0 - Disable Clock. 1 - Enable Clock.
16 USB0_HOSTM	Enables the clock for the USB0-FS Host Master. 0 - Disable Clock. 1 - Enable Clock.
15 SYSCTL	SYSCTL block clock. 0 - Disable Clock. 1 - Enable Clock.
14 PMUX1	Enables the clock for Peripheral Input Mux 1. 0 - Disable Clock. 1 - Enable Clock.
13 —	Reserved
12 —	Reserved Read value is undefined, only zero should be written.
11 —	Reserved
10-9	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
8 FREQME	Enables the clock for the Frequency meter. 0 - Disable Clock. 1 - Enable Clock.
7-4 —	Reserved
3 —	Reserved
2 COMP	Enables the clock for the Analog comparator. 0 - Disable Clock. 1 - Enable Clock.
1 DMA1	Enables the clock for the DMA1. 0 - Disable Clock. 1 - Enable Clock.
0 —	Reserved Read value is undefined, only zero should be written.

8.5.1.1.20 AHB Clock Control 3 (AHBCLKCTRL3)

The AHBCLKCTRLn registers enable the clocks to individual modules.

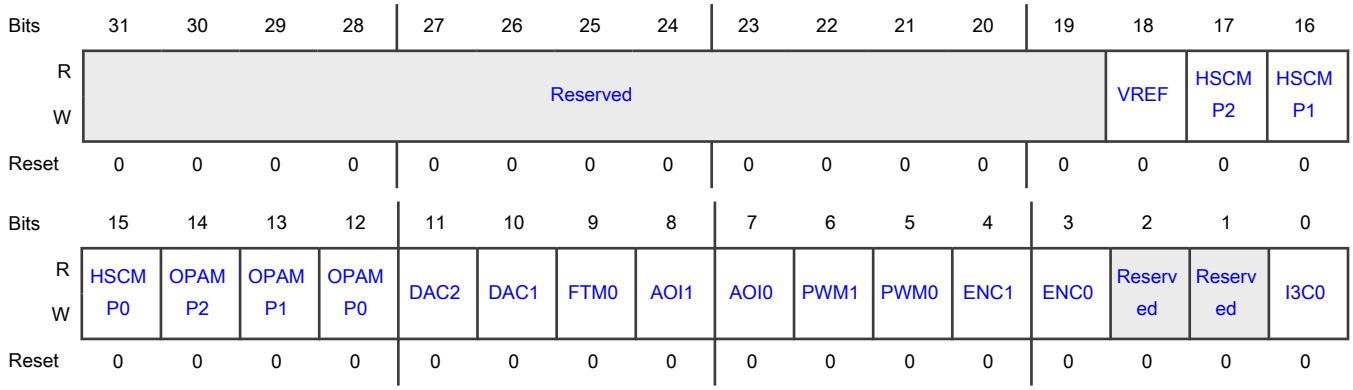
NOTE

When modifying the AHBCLKCTRL registers, use the related AHBCLKCTRLSET and AHBCLKCTRLCLR registers to avoid setting or clearing bits unintentionally.

Offset

Register	Offset
AHBCLKCTRL3	20Ch

Diagram



Fields

Field	Description
31-19 —	Reserved
18 VREF	Enables the clock for VREF. 0 - Disable Clock. 1 - Enable Clock.
17 HSCMP2	Enables the clock for HSCMP2. 0 - Disable Clock. 1 - Enable Clock.
16 HSCMP1	Enables the clock for HSCMP1. 0 - Disable Clock. 1 - Enable Clock.
15 HSCMP0	Enables the clock for HSCMP0. 0 - Disable Clock. 1 - Enable Clock.
14 OPAMP2	Enables the clock for OPAMP2. 0 - Disable Clock. 1 - Enable Clock.
13 OPAMP1	Enables the clock for OPAMP1. 0 - Disable Clock. 1 - Enable Clock.
12 OPAMP0	Enables the clock for OPAMP0. 0 - Disable Clock.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Enable Clock.
11 DAC2	Enables the clock for DAC2. 0 - Disable Clock. 1 - Enable Clock.
10 DAC1	Enables the clock for DAC1. 0 - Disable Clock. 1 - Enable Clock.
9 FTM0	Enables the clock for FTM0. 0 - Disable Clock. 1 - Enable Clock.
8 AOI1	Enables the clock for AOI1. 0 - Disable Clock. 1 - Enable Clock.
7 AOI0	Enables the clock for AOI0. 0 - Disable Clock. 1 - Enable Clock.
6 PWM1	Enables the clock for PWM1. 0 - Disable Clock. 1 - Enable Clock.
5 PWM0	Enables the clock for PWM0. 0 - Disable Clock. 1 - Enable Clock.
4 ENC1	Enables the clock for ENC1. 0 - Disable Clock. 1 - Enable Clock.
3 ENC0	Enables the clock for ENC0. 0 - Disable Clock. 1 - Enable Clock.
2 —	Reserved
1	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
0 I3C0	Enables the clock for I3C0. 0 - Disable Clock. 1 - Enable Clock.

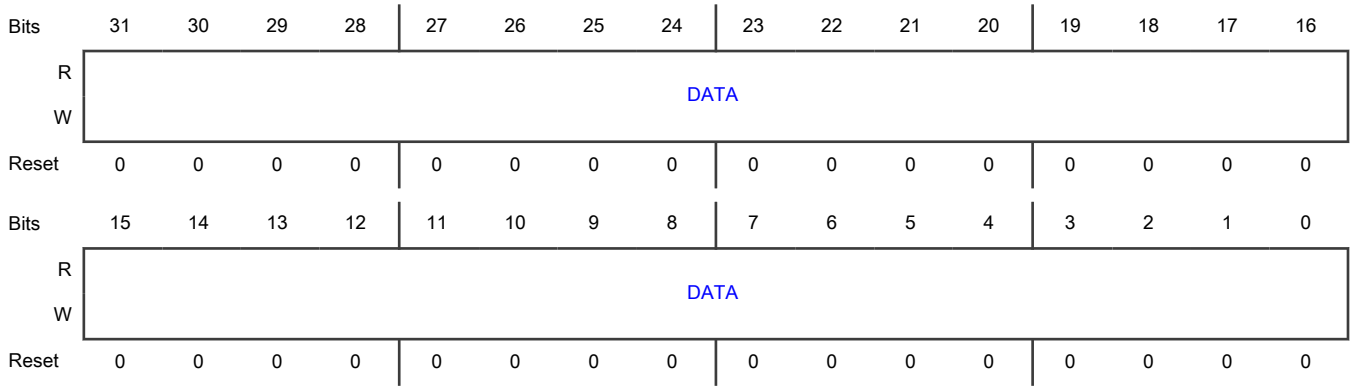
8.5.1.1.21 AHB Clock Control Set (AHBCLKCTRLSET0 - AHBCLKCTRLSET3)

Writing a 1 to a bit position in a write-only AHBCLKCTRLSETn register sets the corresponding position in AHBCLKCTRLn.

Offset

Register	Offset
AHBCLKCTRLSET0	220h
AHBCLKCTRLSET1	224h
AHBCLKCTRLSET2	228h
AHBCLKCTRLSET3	22Ch

Diagram



Fields

Field	Description
31-0 DATA	Data array value

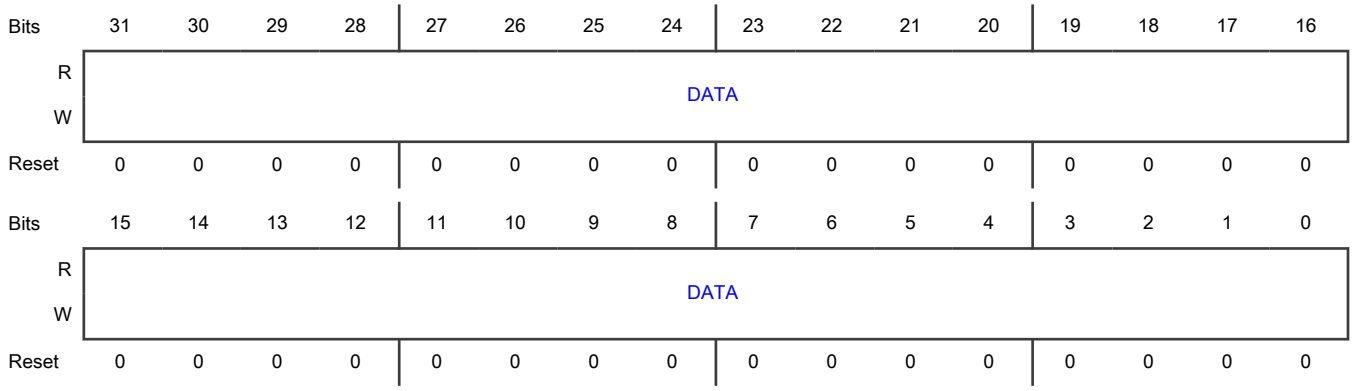
8.5.1.1.22 AHB Clock Control Clear (AHBCLKCTRLCLR0 - AHBCLKCTRLCLR3)

Writing a 1 to a bit position in a write-only AHBCLKCTRLCLRn register clears the corresponding position in AHBCLKCTRLn.

Offset

Register	Offset
AHBCLKCTRLCLR0	240h
AHBCLKCTRLCLR1	244h
AHBCLKCTRLCLR2	248h
AHBCLKCTRLCLR3	24Ch

Diagram



Fields

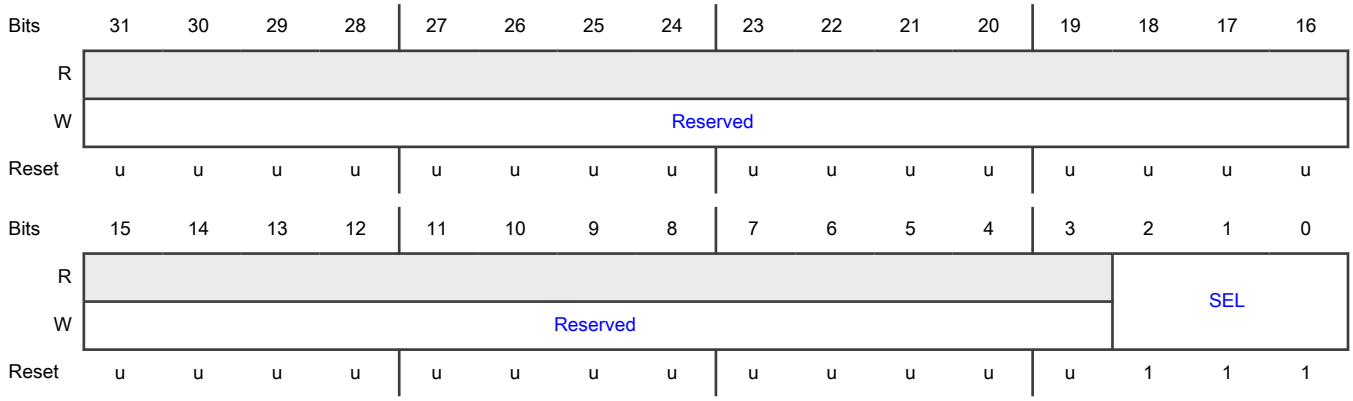
Field	Description
31-0 DATA	Data array value

8.5.1.1.23 System Tick Timer for CPU0 source select (SYSTICKCLKSEL0)

Offset

Register	Offset
SYSTICKCLKSEL0	260h

Diagram



Fields

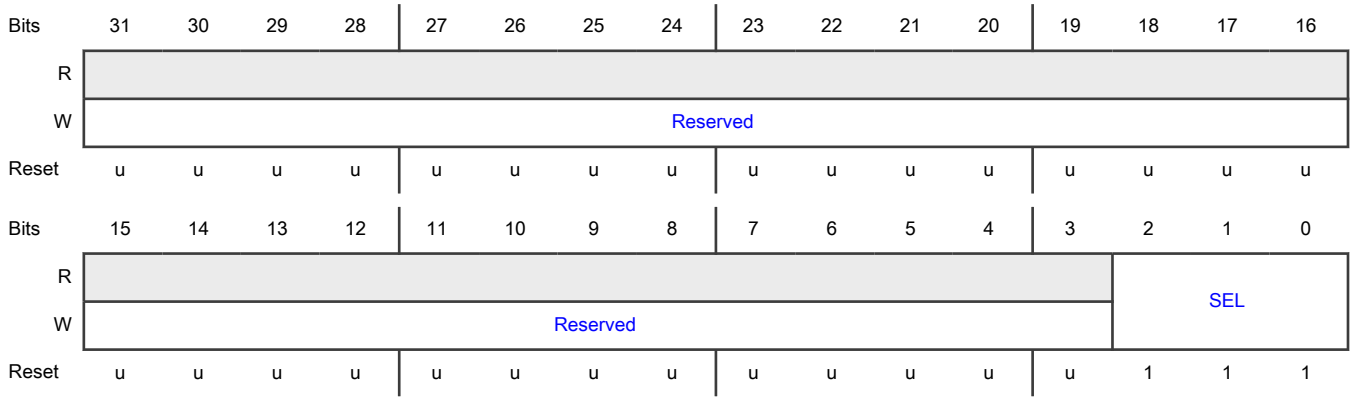
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	System Tick Timer for CPU0 source select. 000 - System Tick 0 divided clock. 001 - FRO 1MHz clock. 010 - Oscillator 32 kHz clock. 011 - No clock. 100 - No clock. 101 - No clock. 110 - No clock. 111 - No clock.

8.5.1.1.24 Trace clock source select (TRACECLKSEL)

Offset

Register	Offset
TRACECLKSEL	268h

Diagram



Fields

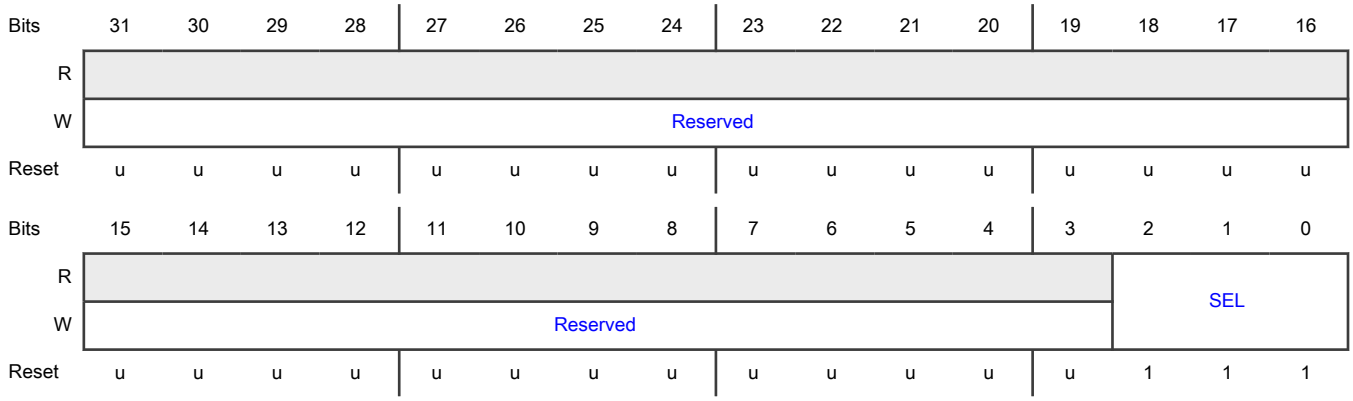
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	Trace clock source select. 000 - Trace divided clock. 001 - FRO 1MHz clock. 010 - Oscillator 32 kHz clock. 011 - No clock. 100 - No clock. 101 - No clock. 110 - No clock. 111 - No clock.

8.5.1.1.25 CTimer 0 clock source select (CTIMERCLKSEL0)

Offset

Register	Offset
CTIMERCLKSEL0	26Ch

Diagram



Fields

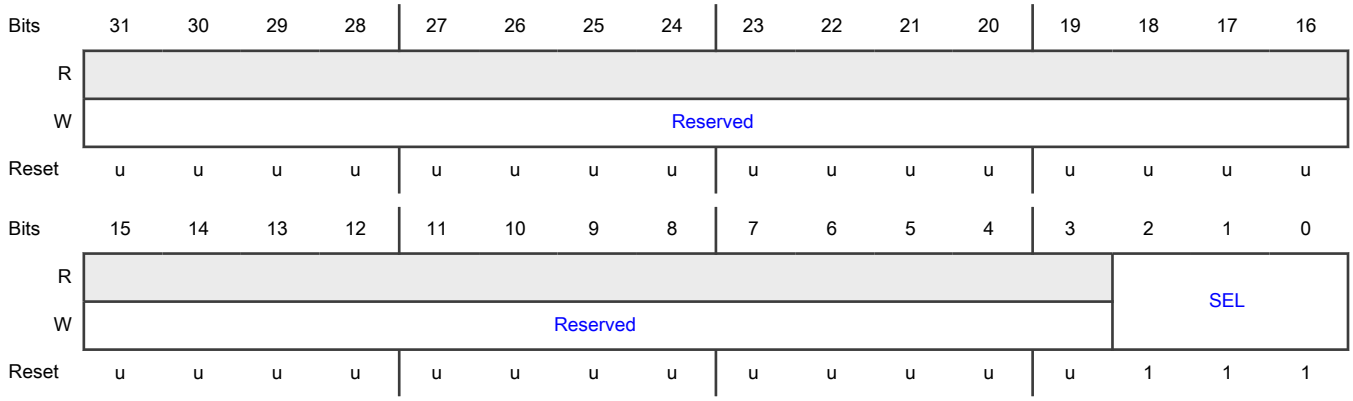
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	CTimer 0 clock source select. 000 - Main clock. 001 - PLL0 clock. 010 - PLL1 clock. 011 - FRO 96 MHz clock. 100 - FRO 1MHz clock. 101 - MCLK clock. 110 - Oscillator 32kHz clock. 111 - No clock.

8.5.1.1.26 CTimer 1 clock source select (CTIMERCLKSEL1)

Offset

Register	Offset
CTIMERCLKSEL1	270h

Diagram



Fields

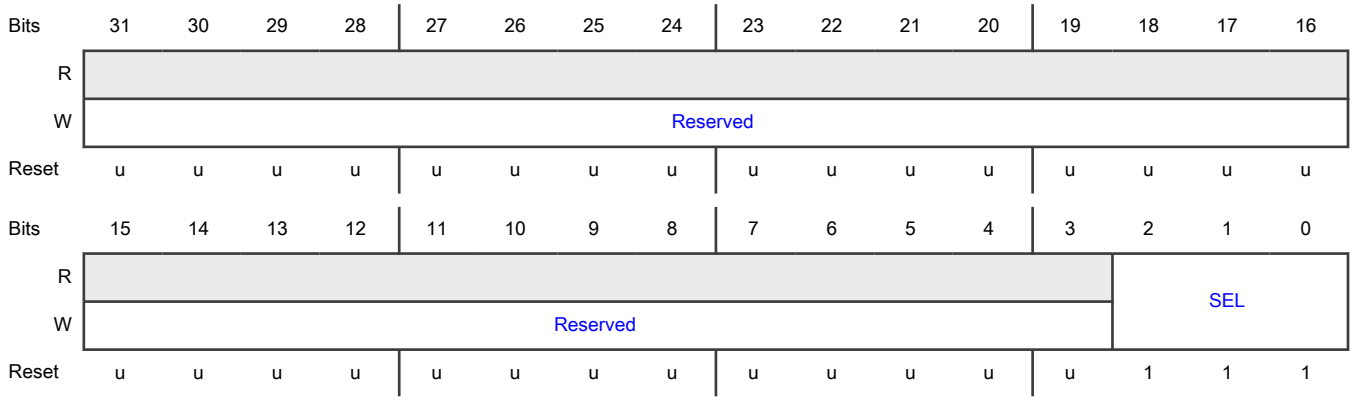
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	CTimer 1 clock source select. 000 - Main clock. 001 - PLL0 clock. 010 - PLL1 clock. 011 - FRO 96 MHz clock. 100 - FRO 1MHz clock. 101 - MCLK clock. 110 - Oscillator 32kHz clock. 111 - No clock.

8.5.1.1.27 CTimer 2 clock source select (CTIMERCLKSEL2)

Offset

Register	Offset
CTIMERCLKSEL2	274h

Diagram



Fields

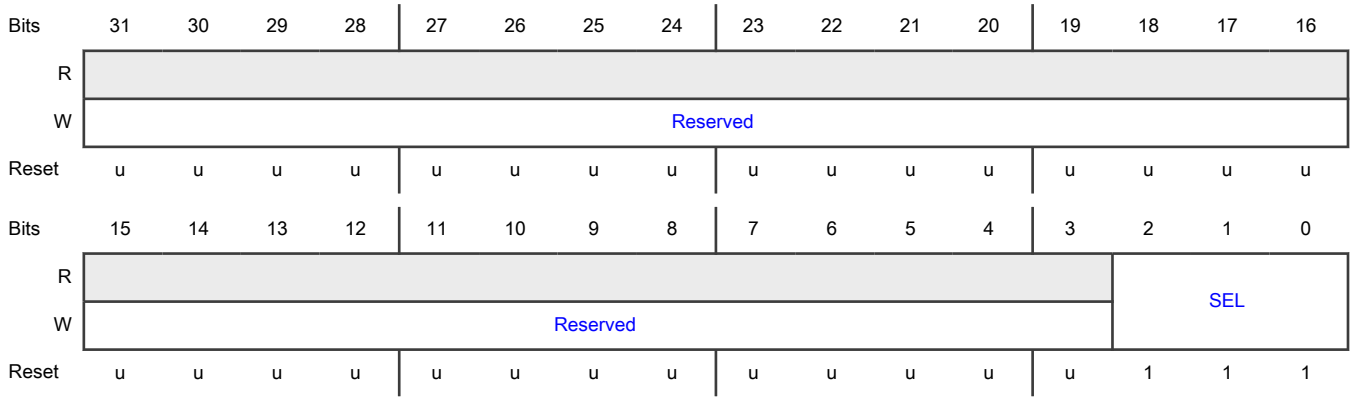
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	CTimer 2 clock source select. 000 - Main clock. 001 - PLL0 clock. 010 - PLL1 clock. 011 - FRO 96 MHz clock. 100 - FRO 1MHz clock. 101 - MCLK clock. 110 - Oscillator 32kHz clock. 111 - No clock.

8.5.1.1.28 CTimer 3 clock source select (CTIMERCLKSEL3)

Offset

Register	Offset
CTIMERCLKSEL3	278h

Diagram



Fields

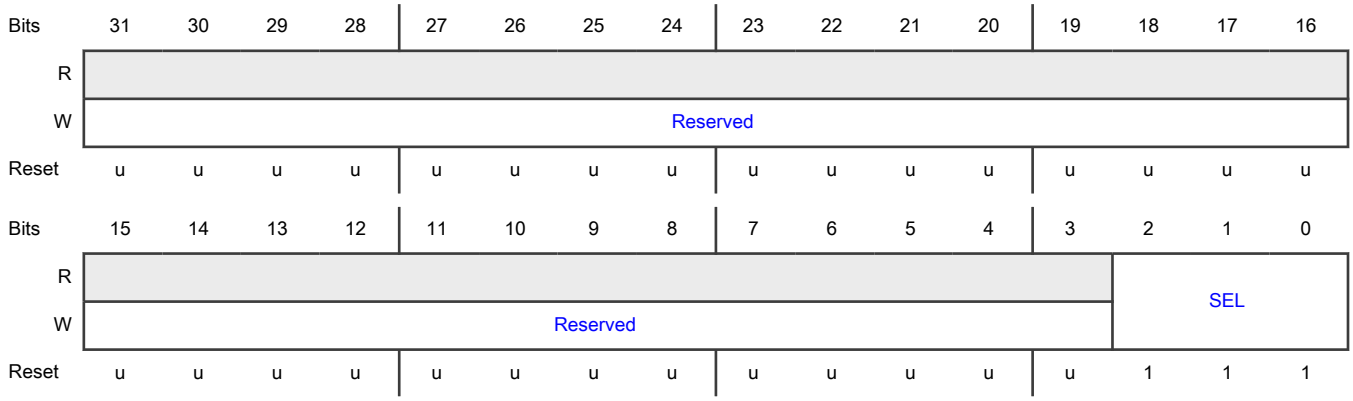
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	CTimer 3 clock source select. 000 - Main clock. 001 - PLL0 clock. 010 - PLL1 clock. 011 - FRO 96 MHz clock. 100 - FRO 1MHz clock. 101 - MCLK clock. 110 - Oscillator 32kHz clock. 111 - No clock.

8.5.1.1.29 CTimer 4 clock source select (CTIMERCLKSEL4)

Offset

Register	Offset
CTIMERCLKSEL4	27Ch

Diagram



Fields

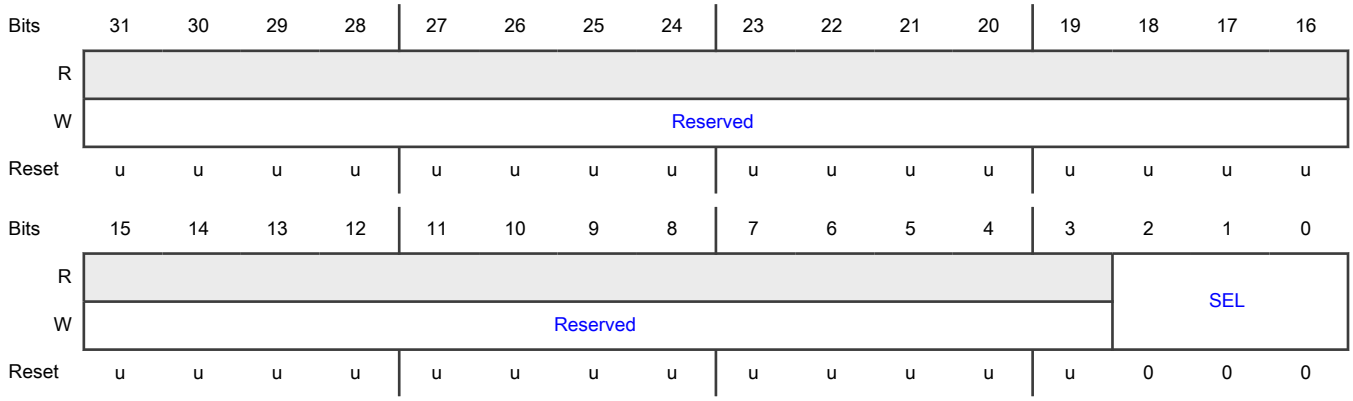
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	CTimer 4 clock source select. 000 - Main clock. 001 - PLL0 clock. 010 - PLL1 clock. 011 - FRO 96 MHz clock. 100 - FRO 1MHz clock. 101 - MCLK clock. 110 - Oscillator 32kHz clock. 111 - No clock.

8.5.1.1.30 Main clock source select A (MAINCLKSELA)

Offset

Register	Offset
MAINCLKSELA	280h

Diagram



Fields

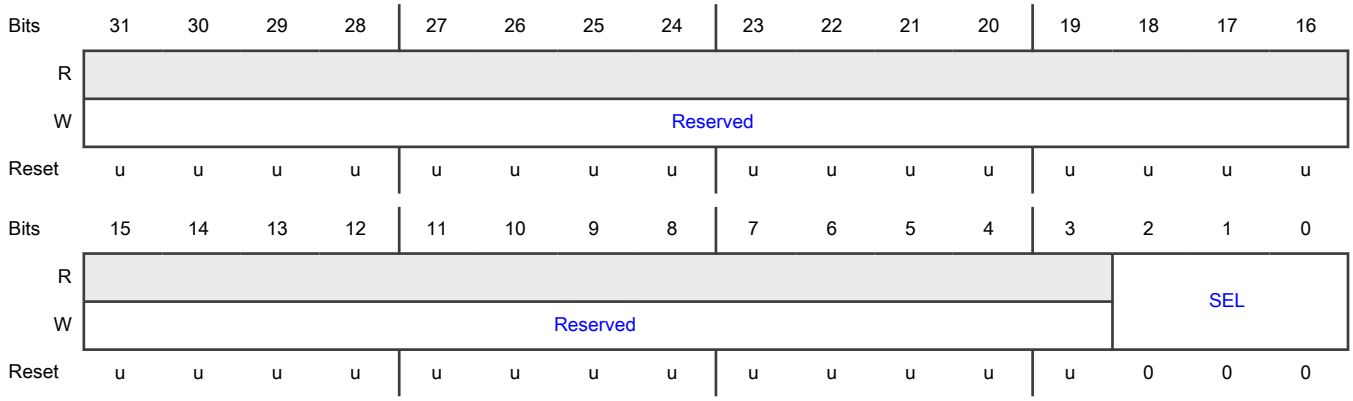
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	Main clock source select A 000 - FRO 12 MHz clock. 001 - CLKIN clock. 010 - FRO 1MHz clock. 011 - FRO 96 MHz clock. 100 - Reserved. 101 - Reserved. 110 - Reserved. 111 - Reserved.

8.5.1.1.31 Main clock source select B (MAINCLKSELB)

Offset

Register	Offset
MAINCLKSELB	284h

Diagram



Fields

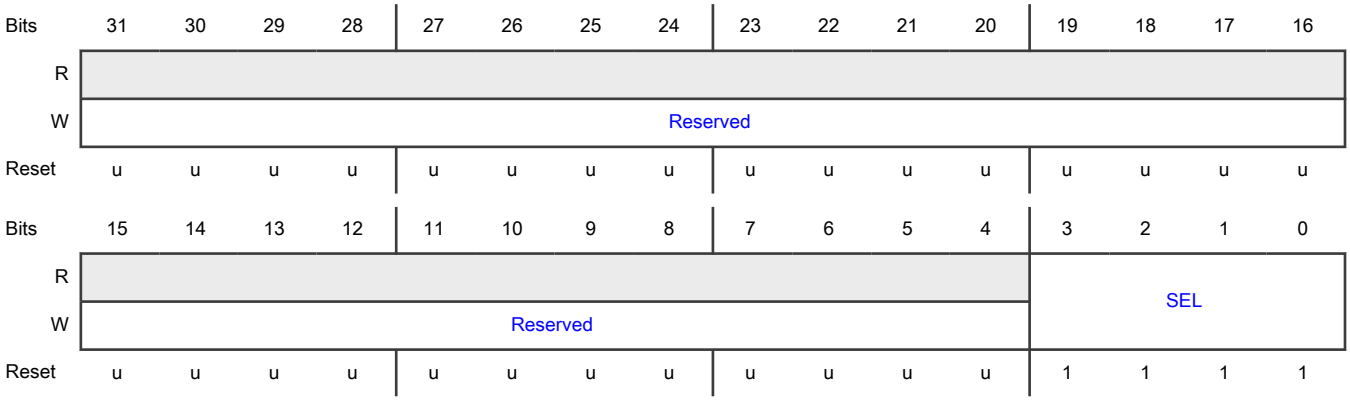
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	Main clock source select B 000 - Use the source selected in MAINCLKSELA. 001 - PLL0 clock. 010 - PLL1 clock. 011 - Oscillator 32 kHz clock. 100 - Reserved. 101 - Reserved. 110 - Reserved. 111 - Reserved.

8.5.1.1.32 CLKOUT clock source select (CLKOUTSEL)

Offset

Register	Offset
CLKOUTSEL	288h

Diagram



Fields

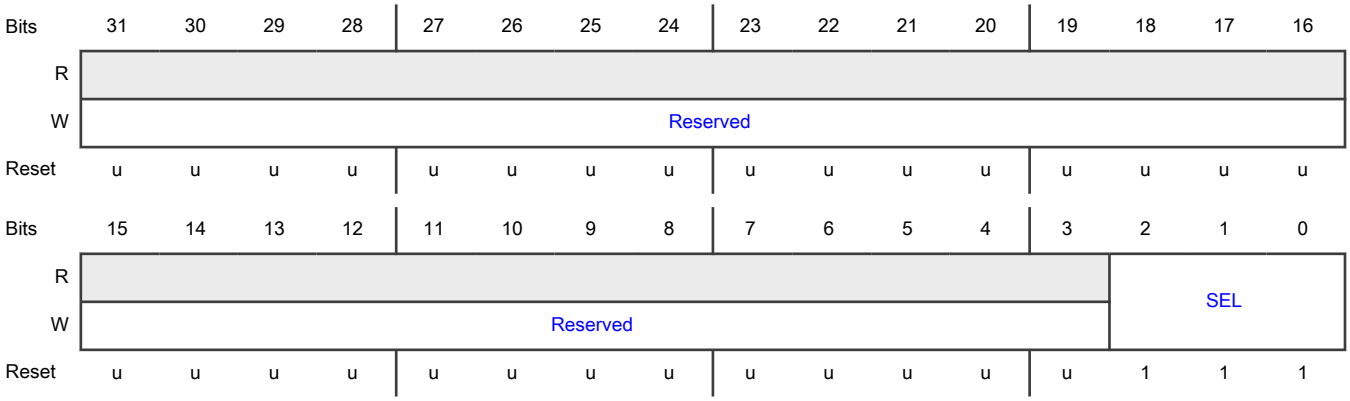
Field	Description
31-4 —	Reserved Read value is undefined, only zero should be written.
3-0 SEL	CLKOUT clock source select. 0000 - Main clock. 0001 - PLL0 clock. 0010 - CLKIN clock. 0011 - FRO 96 MHz clock. 0100 - FRO 1MHz clock. 0101 - PLL1 clock. 0110 - Oscillator 32kHz clock. 0111 - No clock. 1000 - Reserved. 1001 - Reserved. 1010 - Reserved. 1011 - Reserved. 1100 - No clock. 1101 - No clock. 1110 - No clock. 1111 - No clock.

8.5.1.1.33 PLL0 clock source select (PLL0CLKSEL)

Offset

Register	Offset
PLL0CLKSEL	290h

Diagram



Fields

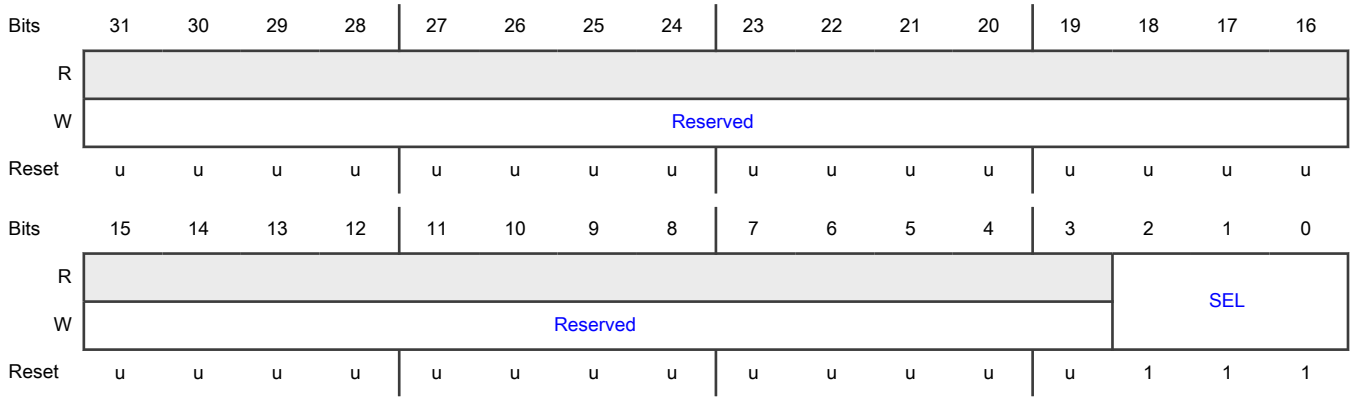
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	PLL0 clock source select. 000 - FRO 12 MHz clock. 001 - CLKIN clock. 010 - FRO 1MHz clock. 011 - Oscillator 32kHz clock. 100 - No clock. 101 - No clock. 110 - No clock. 111 - No clock.

8.5.1.1.34 PLL1 clock source select (PLL1CLKSEL)

Offset

Register	Offset
PLL1CLKSEL	294h

Diagram



Fields

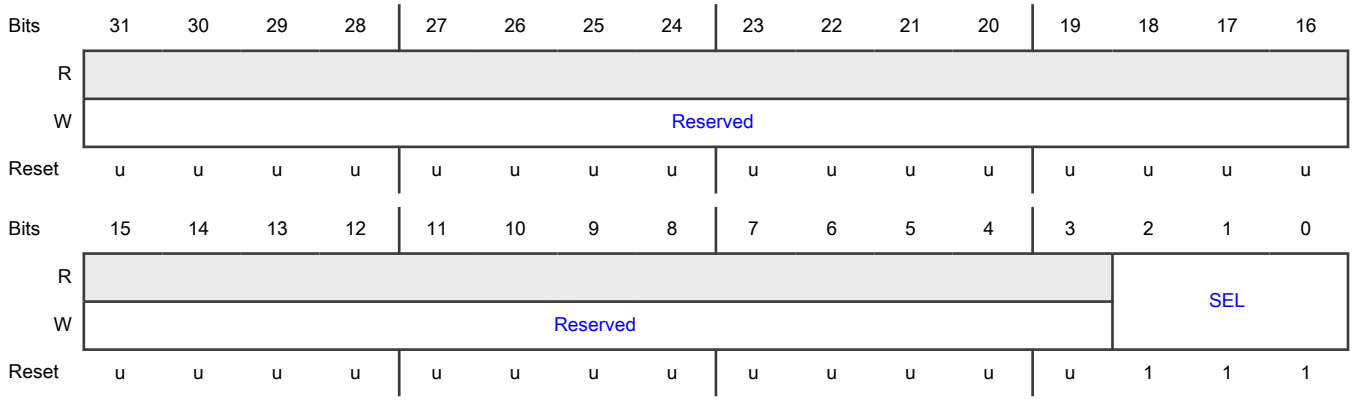
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	PLL1 clock source select. 000 - FRO 12 MHz clock. 001 - CLKIN clock. 010 - FRO 1MHz clock. 011 - Oscillator 32kHz clock. 100 - No clock. 101 - No clock. 110 - No clock. 111 - No clock.

8.5.1.1.35 CAN clock source select (CANCLKSEL)

Offset

Register	Offset
CANCLKSEL	2A0h

Diagram



Fields

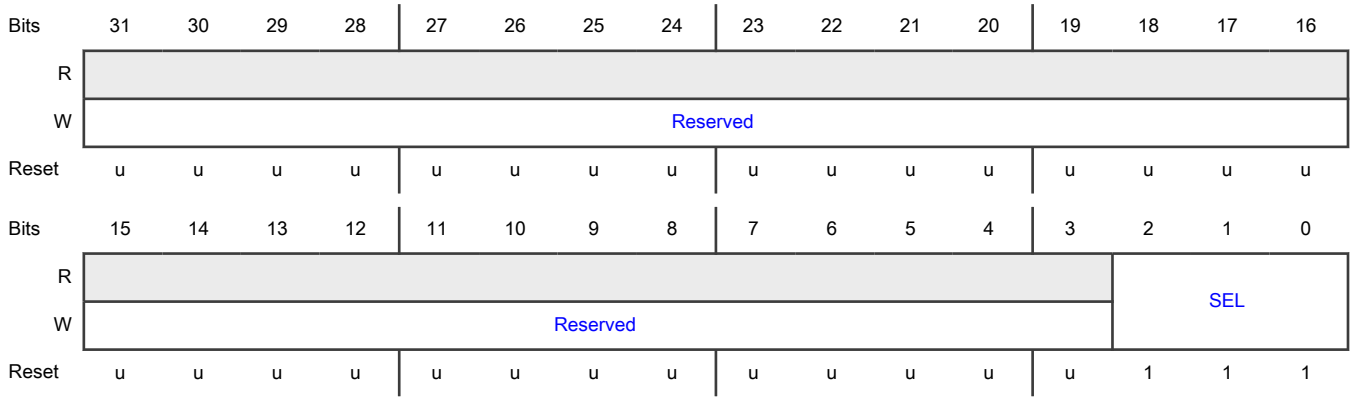
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	CAN clock source select. 000 - CAN divided clock. 001 - FRO 1MHz clock. 010 - Oscillator 32 kHz clock. 011 - No clock. 100 - No clock. 101 - No clock. 110 - No clock. 111 - No clock.

8.5.1.1.36 ADC0 clock source select (ADC0CLKSEL)

Offset

Register	Offset
ADC0CLKSEL	2A4h

Diagram



Fields

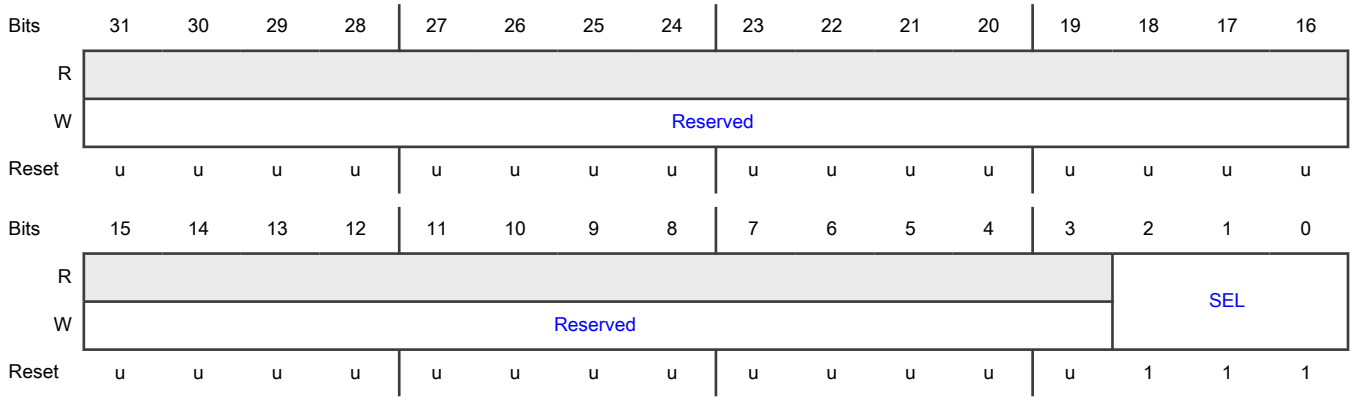
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	ADC clock source select. 000 - Main clock. 001 - PLL0 clock. 010 - FRO 96 MHz clock. 011 - Reserved. 100 - XO to ADC Clock. 101 - No clock. 110 - No clock. 111 - No clock.

8.5.1.1.37 FS USB clock source select (USB0CLKSEL)

Offset

Register	Offset
USB0CLKSEL	2A8h

Diagram



Fields

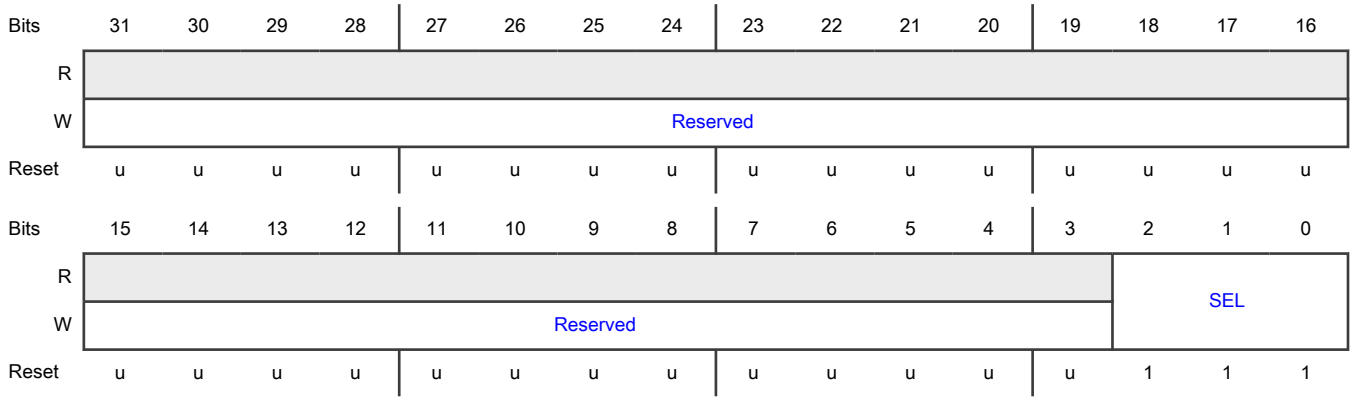
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	FS USB clock source select. 000 - Main clock. 001 - PLL0 clock. 010 - No clock. 011 - FRO 96 MHz clock. 100 - No clock. 101 - PLL1 clock. 110 - No clock. 111 - No clock.

8.5.1.1.38 Flexcomm 0 clock source select for Fractional Rate Divider (FCCLKSEL0)

Offset

Register	Offset
FCCLKSEL0	2B0h

Diagram



Fields

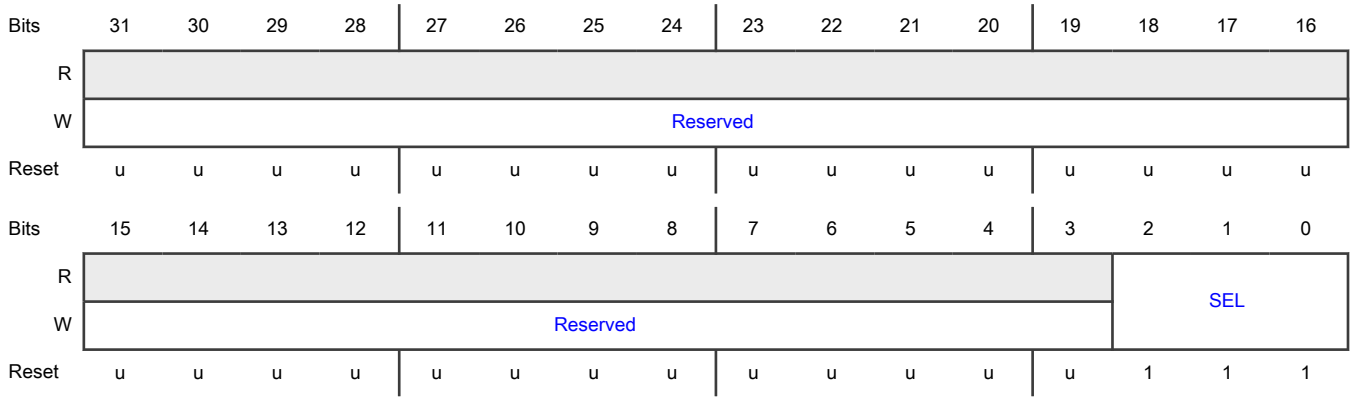
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	Flexcomm 0 clock source select for Fractional Rate Divider. 000 - Main clock. 001 - system PLL divided clock. 010 - FRO 12 MHz clock. 011 - FRO 96 MHz clock. 100 - FRO 1MHz clock. 101 - MCLK clock. 110 - Oscillator 32 kHz clock. 111 - No clock.

8.5.1.1.39 Flexcomm 1 clock source select for Fractional Rate Divider (FCCLKSEL1)

Offset

Register	Offset
FCCLKSEL1	2B4h

Diagram



Fields

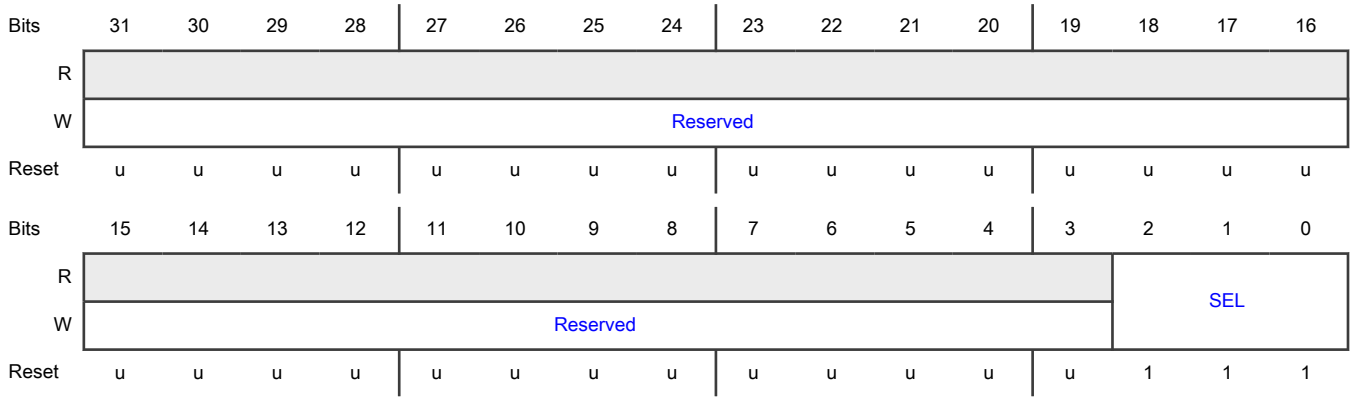
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	Flexcomm 1 clock source select for Fractional Rate Divider. 000 - Main clock. 001 - system PLL divided clock. 010 - FRO 12 MHz clock. 011 - FRO 96 MHz clock. 100 - FRO 1MHz clock. 101 - MCLK clock. 110 - Oscillator 32 kHz clock. 111 - No clock.

8.5.1.1.40 Flexcomm 2 clock source select for Fractional Rate Divider (FCCLKSEL2)

Offset

Register	Offset
FCCLKSEL2	2B8h

Diagram



Fields

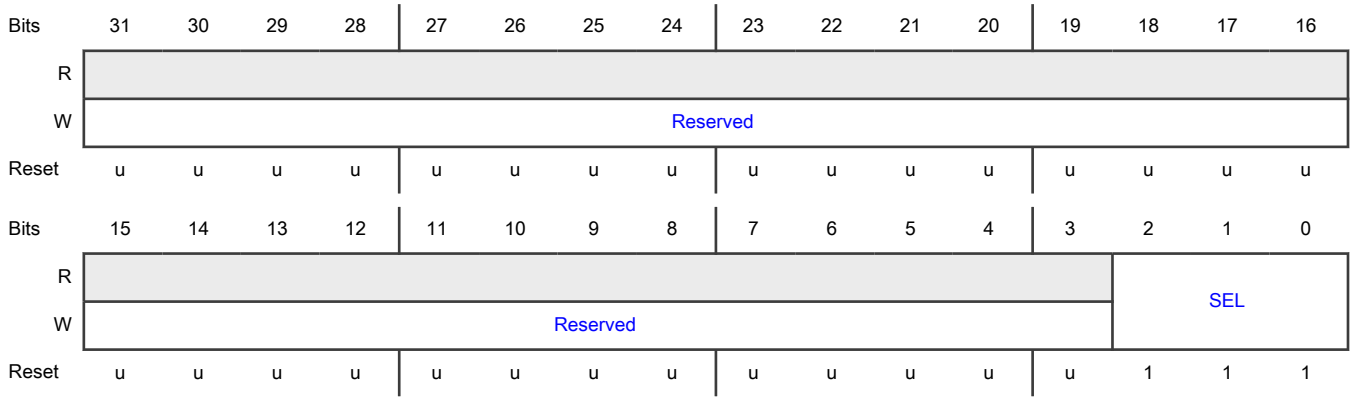
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	Flexcomm 2 clock source select for Fractional Rate Divider. 000 - Main clock. 001 - system PLL divided clock. 010 - FRO 12 MHz clock. 011 - FRO 96 MHz clock. 100 - FRO 1MHz clock. 101 - MCLK clock. 110 - Oscillator 32 kHz clock. 111 - No clock.

8.5.1.1.41 Flexcomm 3 clock source select for Fractional Rate Divider (FCCLKSEL3)

Offset

Register	Offset
FCCLKSEL3	2BCh

Diagram



Fields

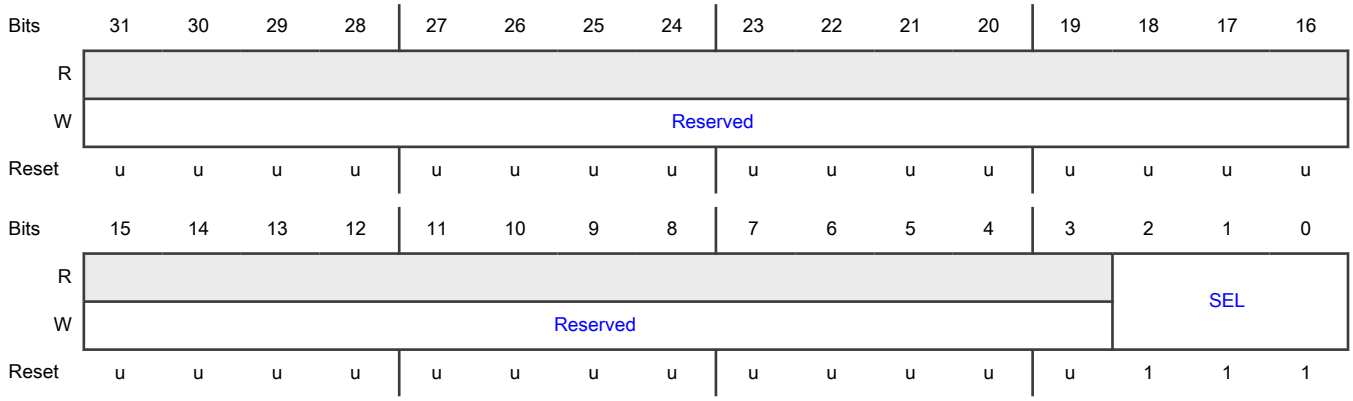
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	Flexcomm 3 clock source select for Fractional Rate Divider. 000 - Main clock. 001 - system PLL divided clock. 010 - FRO 12 MHz clock. 011 - FRO 96 MHz clock. 100 - FRO 1MHz clock. 101 - MCLK clock. 110 - Oscillator 32 kHz clock. 111 - No clock.

8.5.1.1.42 Flexcomm 4 clock source select for Fractional Rate Divider (FCCLKSEL4)

Offset

Register	Offset
FCCLKSEL4	2C0h

Diagram



Fields

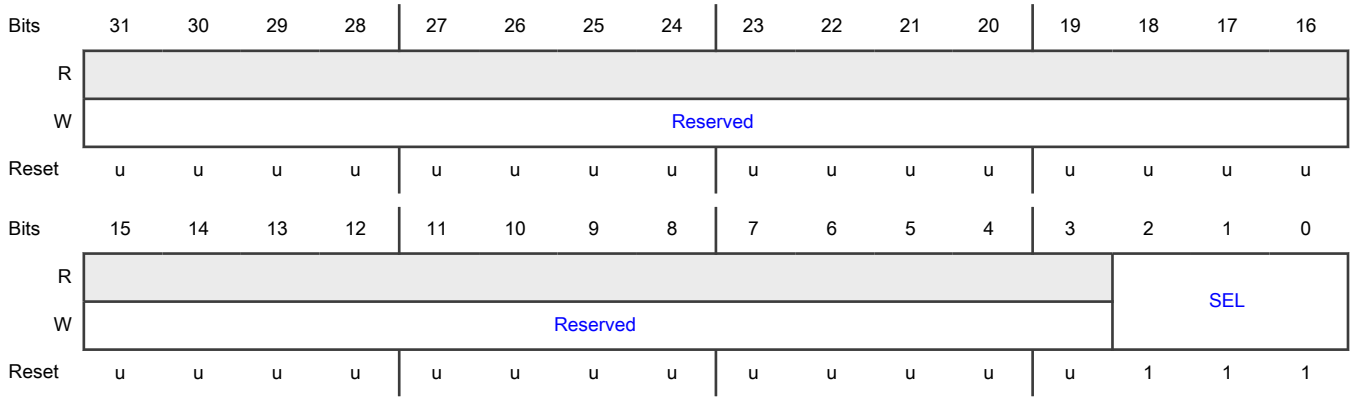
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	Flexcomm 4 clock source select for Fractional Rate Divider. 000 - Main clock. 001 - system PLL divided clock. 010 - FRO 12 MHz clock. 011 - FRO 96 MHz clock. 100 - FRO 1MHz clock. 101 - MCLK clock. 110 - Oscillator 32 kHz clock. 111 - No clock.

8.5.1.1.43 Flexcomm 5 clock source select for Fractional Rate Divider (FCCLKSEL5)

Offset

Register	Offset
FCCLKSEL5	2C4h

Diagram



Fields

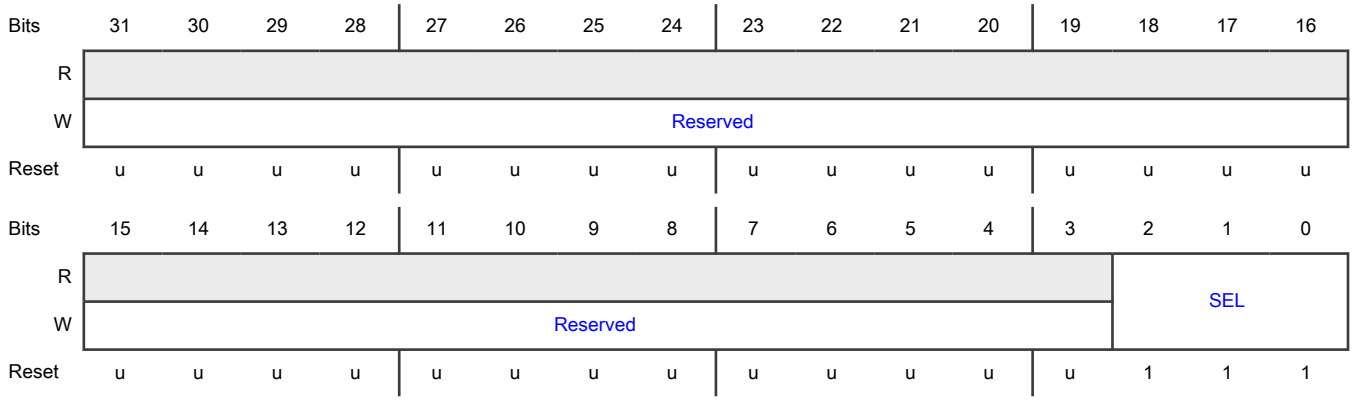
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	Flexcomm 5 clock source select for Fractional Rate Divider. 000 - Main clock. 001 - system PLL divided clock. 010 - FRO 12 MHz clock. 011 - FRO 96 MHz clock. 100 - FRO 1MHz clock. 101 - MCLK clock. 110 - Oscillator 32 kHz clock. 111 - No clock.

8.5.1.1.44 Flexcomm 6 clock source select for Fractional Rate Divider (FCCLKSEL6)

Offset

Register	Offset
FCCLKSEL6	2C8h

Diagram



Fields

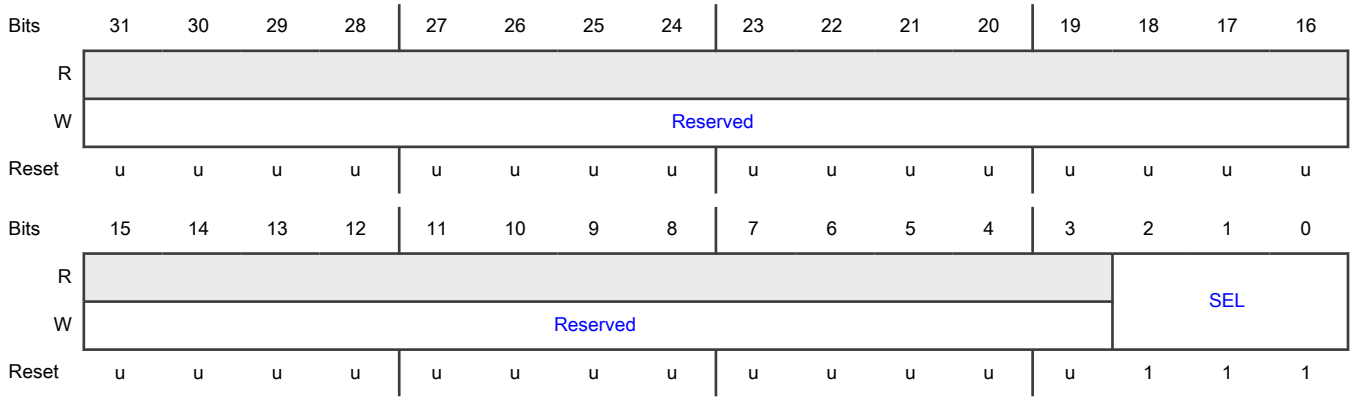
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	Flexcomm 6 clock source select for Fractional Rate Divider. 000 - Main clock. 001 - system PLL divided clock. 010 - FRO 12 MHz clock. 011 - FRO 96 MHz clock. 100 - FRO 1MHz clock. 101 - MCLK clock. 110 - Oscillator 32 kHz clock. 111 - No clock.

8.5.1.1.45 Flexcomm 7 clock source select for Fractional Rate Divider (FCCLKSEL7)

Offset

Register	Offset
FCCLKSEL7	2CCh

Diagram



Fields

Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	Flexcomm 7 clock source select for Fractional Rate Divider. 000 - Main clock. 001 - system PLL divided clock. 010 - FRO 12 MHz clock. 011 - FRO 96 MHz clock. 100 - FRO 1MHz clock. 101 - MCLK clock. 110 - Oscillator 32 kHz clock. 111 - No clock.

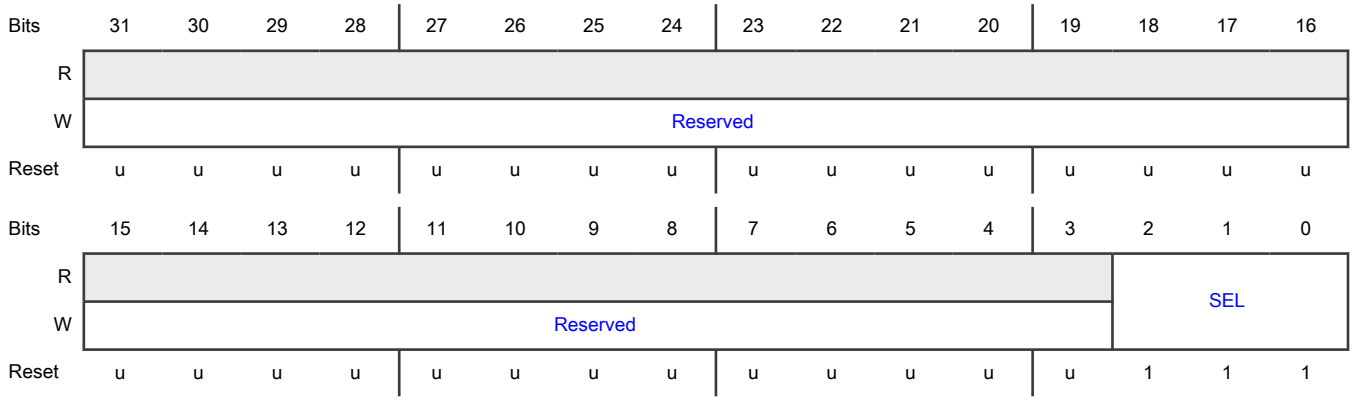
8.5.1.1.46 HS SPI clock source select (HSSPICLKSEL)

HSSPICLKSEL register select the clock source for High-Speed SPI interface.

Offset

Register	Offset
HSSPICLKSEL	2D0h

Diagram



Fields

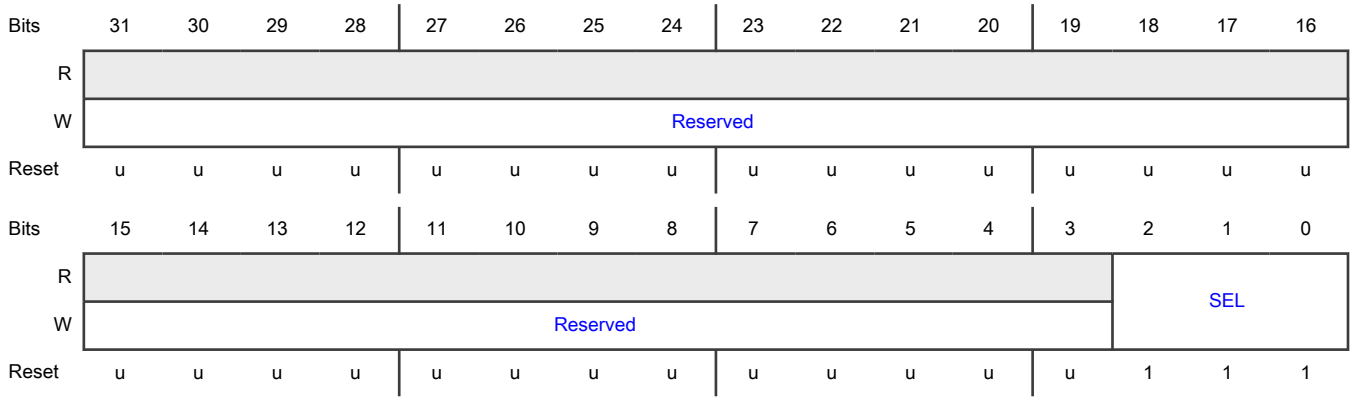
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	HS SPI clock source select. 000 - Main clock. 001 - system PLL divided clock. 010 - FRO 12 MHz clock. 011 - FRO 96 MHz clock. 100 - FRO 1MHz clock. 101 - No clock. 110 - Oscillator 32 kHz clock. 111 - No clock.

8.5.1.1.47 MCLK clock source select (MCLKCLKSEL)

Offset

Register	Offset
MCLKCLKSEL	2E0h

Diagram



Fields

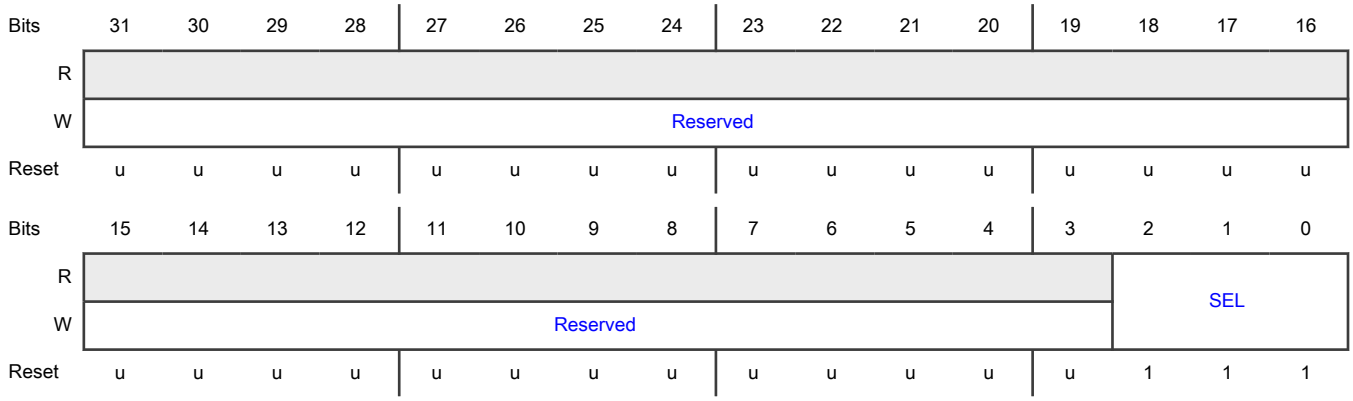
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	MCLK clock source select. 000 - FRO 96 MHz clock. 001 - PLL0 clock. 010 - Reserved. 011 - Reserved. 100 - No clock. 101 - No clock. 110 - No clock. 111 - No clock.

8.5.1.1.48 SCTimer/PWM clock source select (SCTCLKSEL)

Offset

Register	Offset
SCTCLKSEL	2F0h

Diagram



Fields

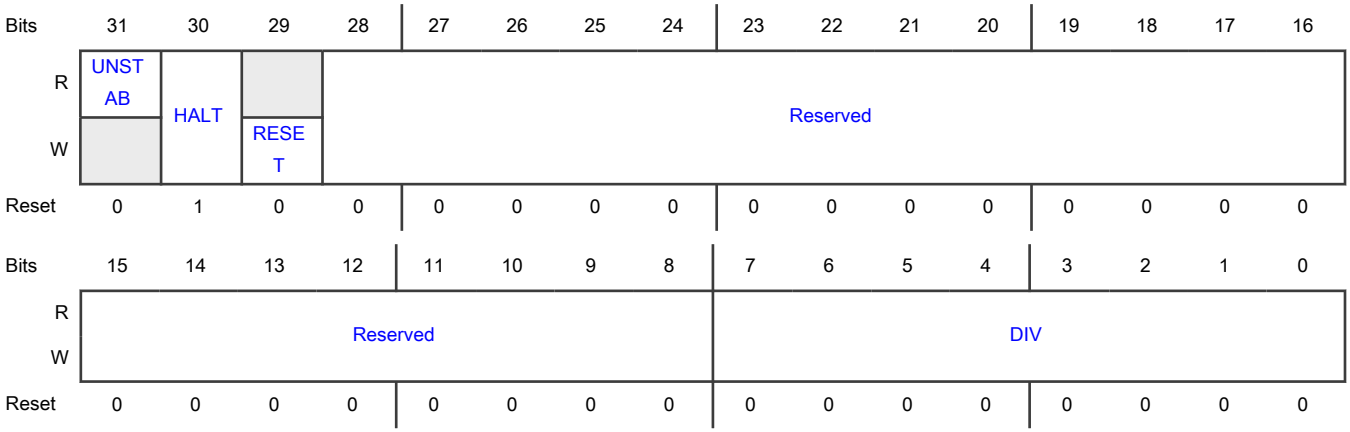
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	SCTimer/PWM clock source select. 000 - Main clock. 001 - PLL0 clock. 010 - CLKIN clock. 011 - FRO 96 MHz clock. 100 - PLL1 clock. 101 - MCLK clock. 110 - No clock. 111 - No clock.

8.5.1.1.49 System Tick Timer divider for CPU0 (SYSTICKCLKDIV0)

Offset

Register	Offset
SYSTICKCLKDIV0	300h

Diagram



Fields

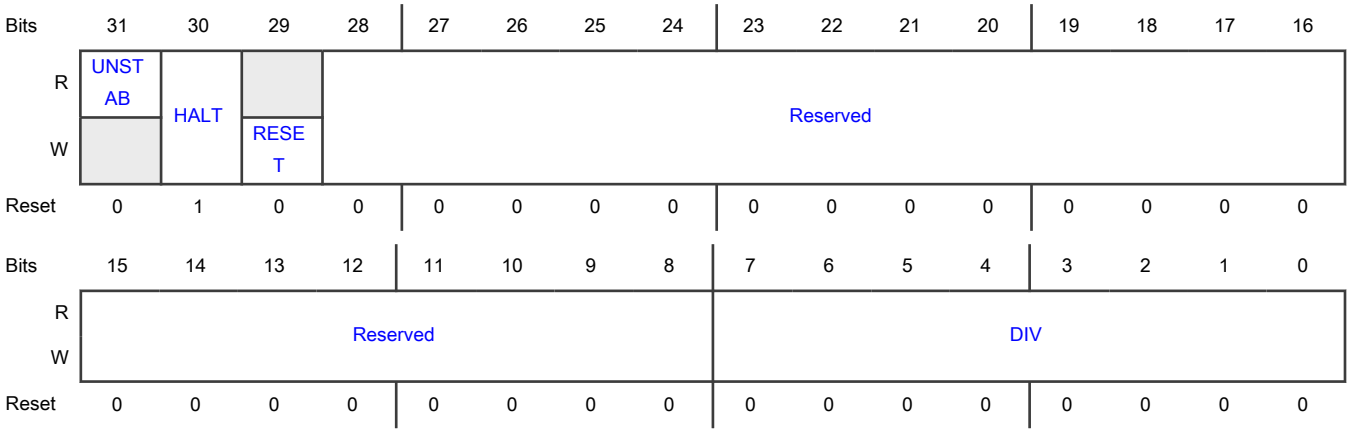
Field	Description
31 UNSTAB	Divider status flag. 0 - Divider clock is stable. 1 - Clock frequency is not stable.
30 HALT	Halts the divider counter. 0 - Divider clock is running. 1 - Divider clock is stopped.
29 RESET	Resets the divider counter. 0 - Divider is not reset. 1 - Divider is reset.
28-8 —	Reserved Read value is undefined, only zero should be written.
7-0 DIV	Clock divider value. The divider value = (DIV + 1).

8.5.1.1.50 TRACE clock divider (TRACECLKDIV)

Offset

Register	Offset
TRACECLKDIV	308h

Diagram



Fields

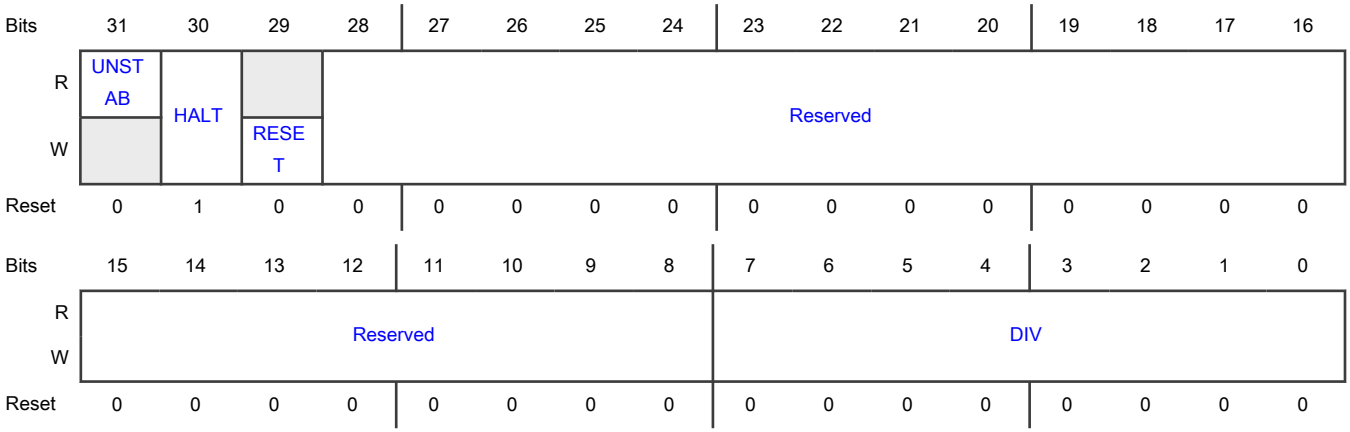
Field	Description
31 UNSTAB	Divider status flag. 0 - Divider clock is stable. 1 - Clock frequency is not stable.
30 HALT	Halts the divider counter. 0 - Divider clock is running. 1 - Divider clock is stopped.
29 RESET	Resets the divider counter. 0 - Divider is not reset. 1 - Divider is reset.
28-8 —	Reserved Read value is undefined, only zero should be written.
7-0 DIV	Clock divider value. The divider value = (DIV + 1).

8.5.1.1.51 CAN clock divider (CANCLKDIV)

Offset

Register	Offset
CANCLKDIV	30Ch

Diagram



Fields

Field	Description
31 UNSTAB	Divider status flag. 0 - Divider clock is stable. 1 - Clock frequency is not stable.
30 HALT	Halts the divider counter. 0 - Divider clock is running. 1 - Divider clock is stopped.
29 RESET	Resets the divider counter. 0 - Divider is not reset. 1 - Divider is reset.
28-8 —	Reserved Read value is undefined, only zero should be written.
7-0 DIV	Clock divider value. The divider value = (DIV + 1).

8.5.1.1.52 Fractional rate divider for flexcomm n (FRGCTRL0 - FRGCTRL7)

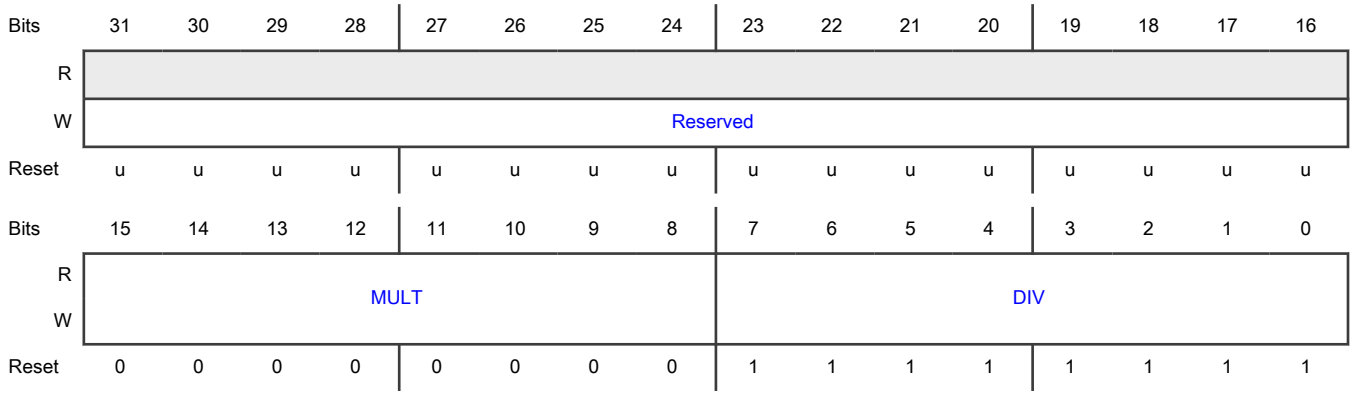
Frequency output = (Frequency of clock selected via FCCLKSEL0) / (1 + MULT / DIV)

Offset

For n = 0 to 7:

Register	Offset
FRGCTRLn	320h + (n × 4h)

Diagram



Fields

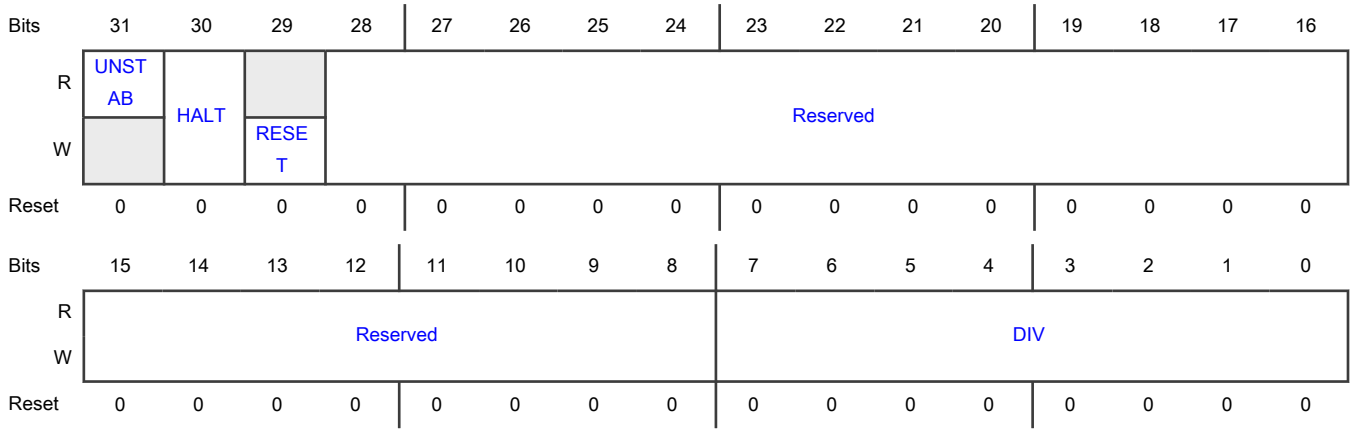
Field	Description
31-16 —	Reserved Read value is undefined, only zero should be written.
15-8 MULT	Numerator of the fractional rate divider.
7-0 DIV	Denominator of the fractional rate divider. The divider value = (DIV + 1).

8.5.1.1.53 System clock divider (AHBCLKDIV)

Offset

Register	Offset
AHBCLKDIV	380h

Diagram



Fields

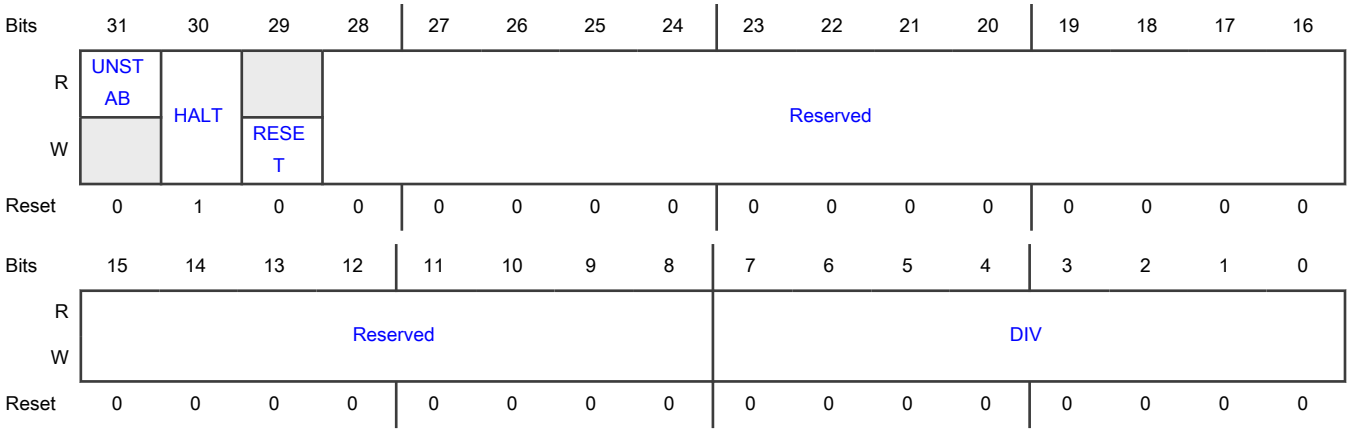
Field	Description
31 UNSTAB	Divider status flag. 0 - Divider clock is stable. 1 - Clock frequency is not stable.
30 HALT	Halts the divider counter. 0 - Divider clock is running. 1 - Divider clock is stopped.
29 RESET	Resets the divider counter. 0 - Divider is not reset. 1 - Divider is reset.
28-8 —	Reserved Read value is undefined, only zero should be written.
7-0 DIV	Clock divider value. The divider value = (DIV + 1).

8.5.1.1.54 CLKOUT clock divider (CLKOUTDIV)

Offset

Register	Offset
CLKOUTDIV	384h

Diagram



Fields

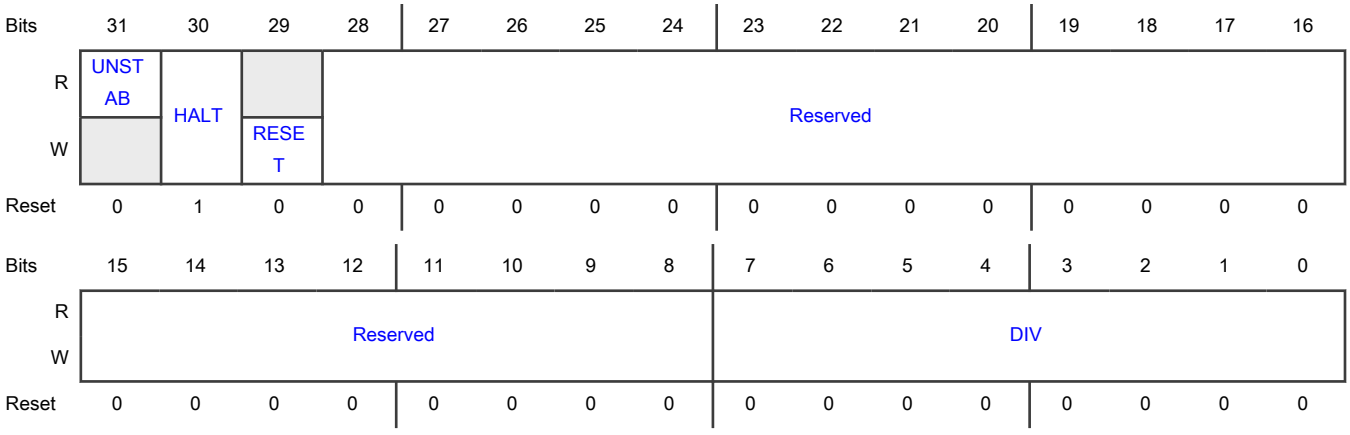
Field	Description
31 UNSTAB	Divider status flag. 0 - Divider clock is stable. 1 - Clock frequency is not stable.
30 HALT	Halts the divider counter. 0 - Divider clock is running. 1 - Divider clock is stopped.
29 RESET	Resets the divider counter. 0 - Divider is not reset. 1 - Divider is reset.
28-8 —	Reserved Read value is undefined, only zero should be written.
7-0 DIV	Clock divider value. The divider value = (DIV + 1).

8.5.1.1.55 FRO_HF (96MHz) clock divider (FROHFDIV)

Offset

Register	Offset
FROHFDIV	388h

Diagram



Fields

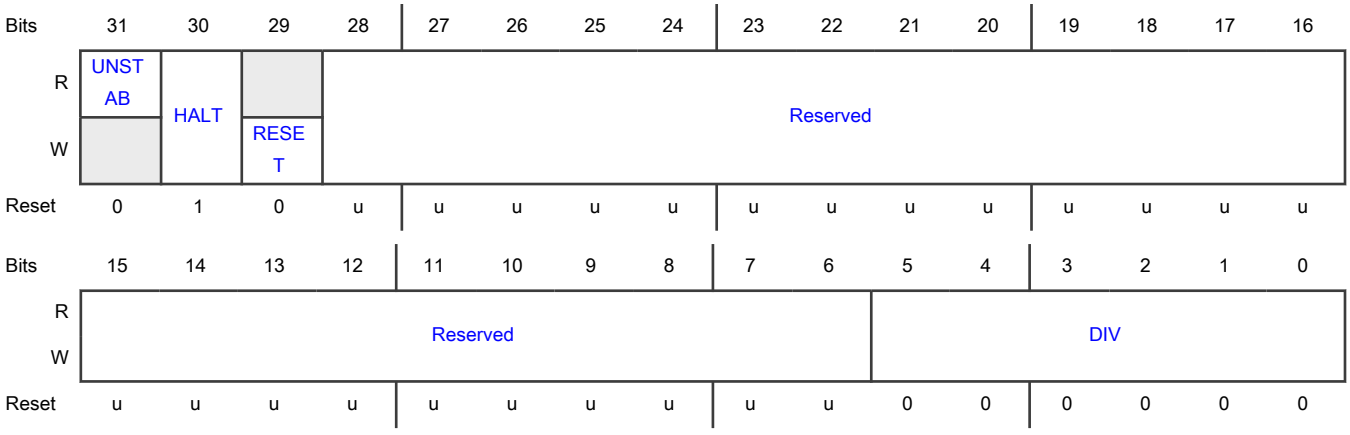
Field	Description
31 UNSTAB	Divider status flag. 0 - Divider clock is stable. 1 - Clock frequency is not stable.
30 HALT	Halts the divider counter. 0 - Divider clock is running. 1 - Divider clock is stopped.
29 RESET	Resets the divider counter. 0 - Divider is not reset. 1 - Divider is reset.
28-8 —	Reserved Read value is undefined, only zero should be written.
7-0 DIV	Clock divider value. The divider value = (DIV + 1).

8.5.1.1.56 WDT clock divider (WDTCLKDIV)

Offset

Register	Offset
WDTCLKDIV	38Ch

Diagram



Fields

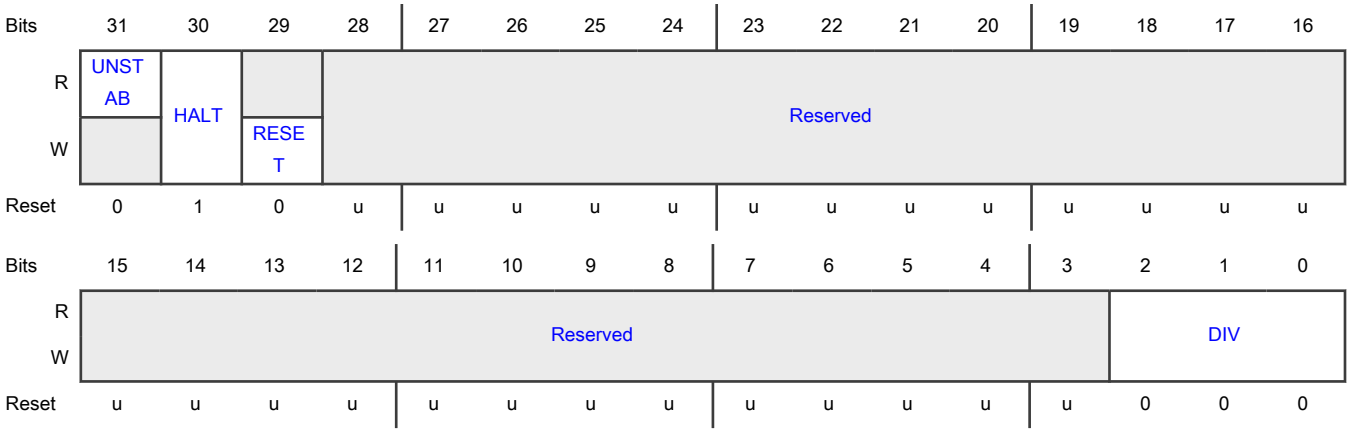
Field	Description
31 UNSTAB	Divider status flag. 0 - Divider clock is stable. 1 - Clock frequency is not stable.
30 HALT	Halts the divider counter. 0 - Divider clock is running. 1 - Divider clock is stopped.
29 RESET	Resets the divider counter. 0 - Divider is not reset. 1 - Divider is reset.
28-6 —	Reserved Read value is undefined, only zero should be written.
5-0 DIV	Clock divider value. The divider value = (DIV + 1).

8.5.1.1.57 ADC0 clock divider (ADC0CLKDIV)

Offset

Register	Offset
ADC0CLKDIV	394h

Diagram



Fields

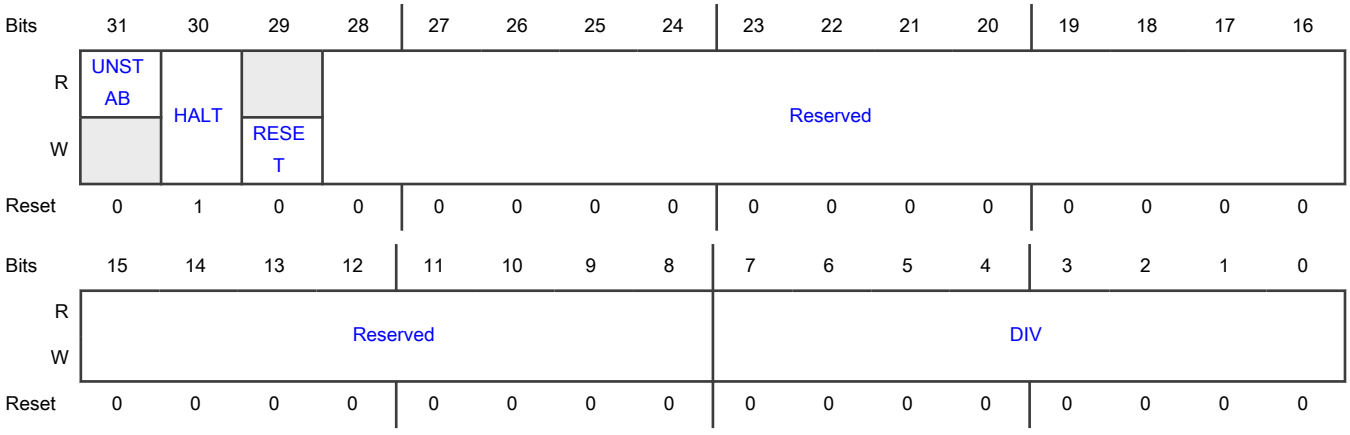
Field	Description
31 UNSTAB	Divider status flag. 0 - Divider clock is stable. 1 - Clock frequency is not stable.
30 HALT	Halts the divider counter. 0 - Divider clock is running. 1 - Divider clock is stopped.
29 RESET	Resets the divider counter. 0 - Divider is not reset. 1 - Divider is reset.
28-3 —	Reserved Read value is undefined, only zero should be written.
2-0 DIV	Clock divider value. The divider value = (DIV + 1).

8.5.1.1.58 USB0-FS Clock divider (USB0CLKDIV)

Offset

Register	Offset
USB0CLKDIV	398h

Diagram



Fields

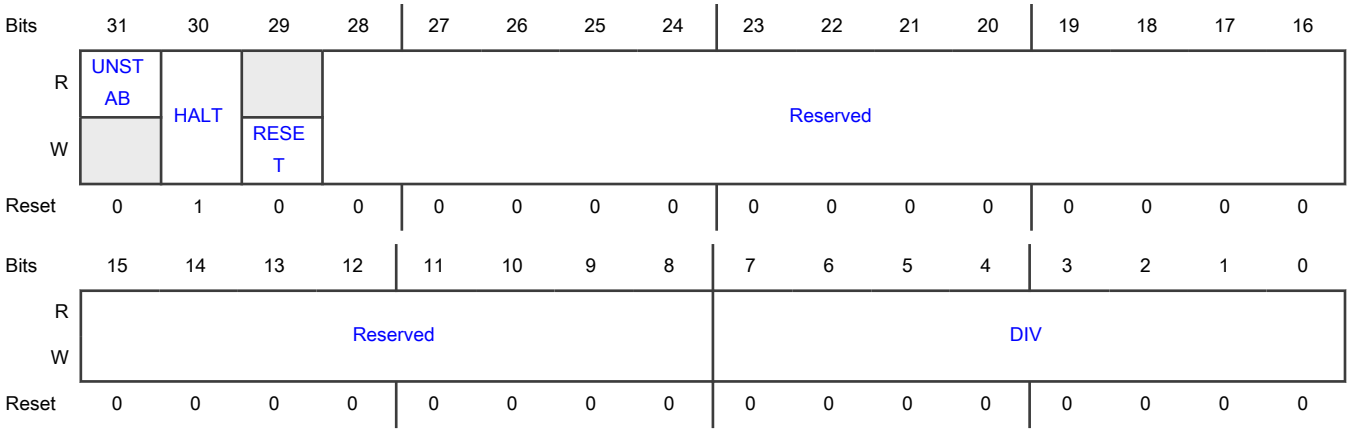
Field	Description
31 UNSTAB	Divider status flag. 0 - Divider clock is stable. 1 - Clock frequency is not stable.
30 HALT	Halts the divider counter. 0 - Divider clock is running. 1 - Divider clock is stopped.
29 RESET	Resets the divider counter. 0 - Divider is not reset. 1 - Divider is reset.
28-8 —	Reserved Read value is undefined, only zero should be written.
7-0 DIV	Clock divider value. The divider value = (DIV + 1).

8.5.1.1.59 I2S MCLK clock divider (MCLKDIV)

Offset

Register	Offset
MCLKDIV	3ACh

Diagram



Fields

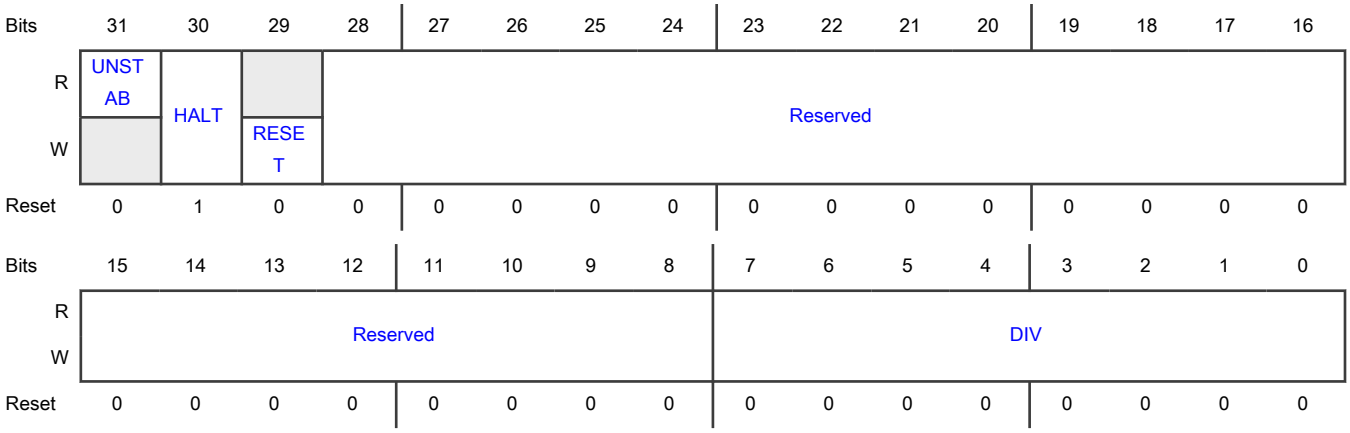
Field	Description
31 UNSTAB	Divider status flag. 0 - Divider clock is stable. 1 - Clock frequency is not stable.
30 HALT	Halts the divider counter. 0 - Divider clock is running. 1 - Divider clock is stopped.
29 RESET	Resets the divider counter. 0 - Divider is not reset. 1 - Divider is reset.
28-8 —	Reserved Read value is undefined, only zero should be written.
7-0 DIV	Clock divider value. The divider value = (DIV + 1).

8.5.1.1.60 SCT/PWM clock divider (SCTCLKDIV)

Offset

Register	Offset
SCTCLKDIV	3B4h

Diagram



Fields

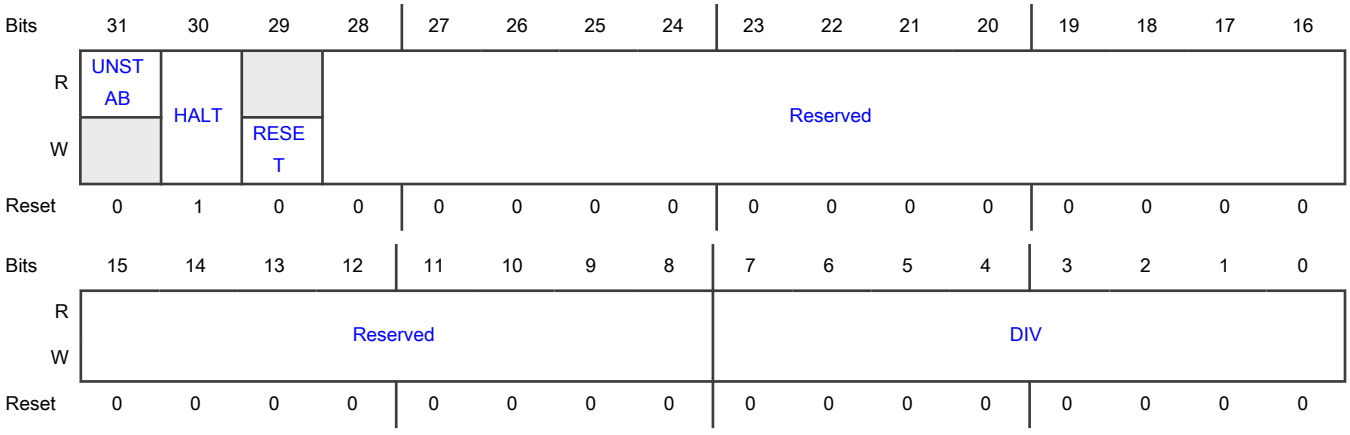
Field	Description
31 UNSTAB	Divider status flag. 0 - Divider clock is stable. 1 - Clock frequency is not stable.
30 HALT	Halts the divider counter. 0 - Divider clock is running. 1 - Divider clock is stopped.
29 RESET	Resets the divider counter. 0 - Divider is not reset. 1 - Divider is reset.
28-8 —	Reserved Read value is undefined, only zero should be written.
7-0 DIV	Clock divider value. The divider value = (DIV + 1).

8.5.1.1.61 PLL clock divider (PLLCLKDIV)

Offset

Register	Offset
PLLCLKDIV	3C4h

Diagram



Fields

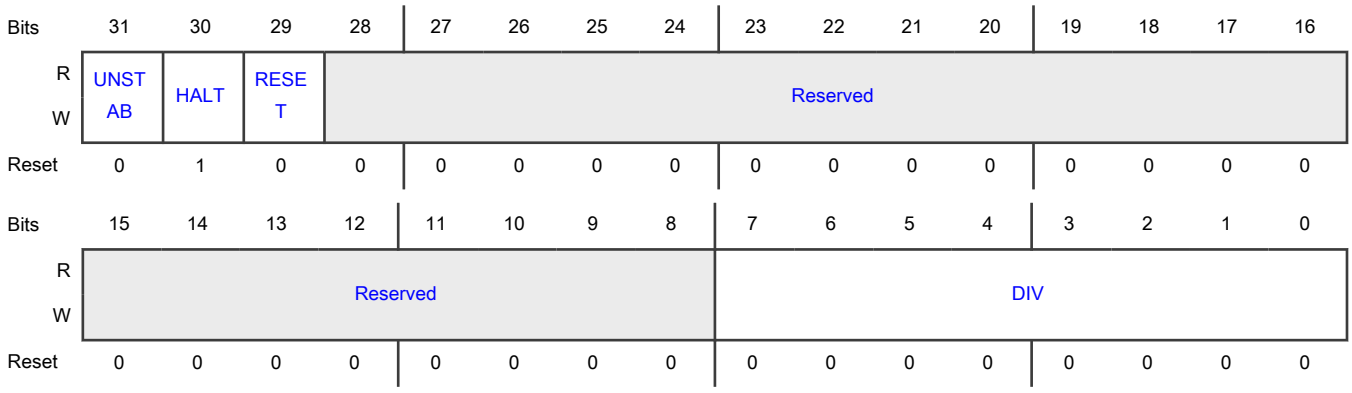
Field	Description
31 UNSTAB	Divider status flag. 0 - Divider clock is stable. 1 - Clock frequency is not stable.
30 HALT	Halts the divider counter. 0 - Divider clock is running. 1 - Divider clock is stopped.
29 RESET	Resets the divider counter. 0 - Divider is not reset. 1 - Divider is reset.
28-8 —	Reserved Read value is undefined, only zero should be written.
7-0 DIV	Clock divider value. The divider value = (DIV + 1).

8.5.1.1.62 CTimer a clock divider (CTIMER0CLKDIV - CTIMER4CLKDIV)

Offset

Register	Offset
CTIMER0CLKDIV	3D0h
CTIMER1CLKDIV	3D4h
CTIMER2CLKDIV	3D8h
CTIMER3CLKDIV	3DCh
CTIMER4CLKDIV	3E0h

Diagram



Fields

Field	Description
31 UNSTAB	Divider status flag 0 - Stable divider clock. 1 - Unstable clock frequency.
30 HALT	Halts the divider counter 0 - Divider clock is running. 1 - Divider clock has stopped.
29 RESET	Resets the divider counter 0 - Divider is not reset 1 - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value The divider value = (DIV + 1)

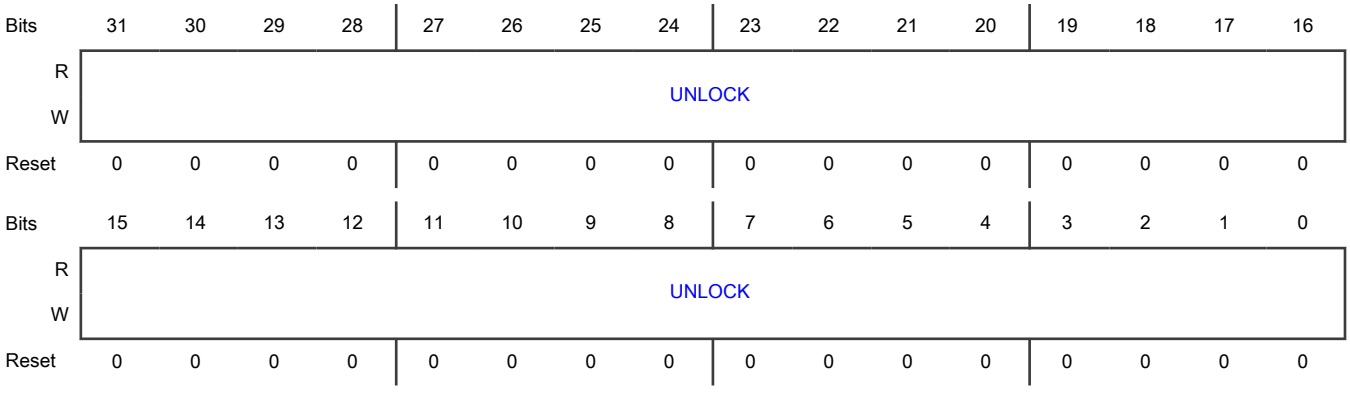
8.5.1.1.63 Clock configuration unlock (CLKUNLOCK)

This register controls access to clock select and divider configuration registers.

Offset

Register	Offset
CLKUNLOCK	3FCh

Diagram



Fields

Field	Description
31-0 UNLOCK	Control clock configuration registers access (for example, xxxDIV, xxxSEL). 0000_0000_0000_0000_0000_0000_0000_0000 - All hardware clock configruration are freeze. 0000_0000_0000_0000_0000_0000_0000_0001 - Update all clock configuration.

8.5.1.1.64 FMC configuration (FMCCR)

This register controls FMC configuration. Depending on the system clock frequency, access to the flash memory can be configured with various access times by writing to the FMCCR register.

It is recommended to use the power API to configure device operation in order to achieve lower power operation.

Enabling buffering, acceleration, and prefetch will substantially improve performance. Buffering saves power by allowing previously accessed information to be reused without a flash read. Acceleration saves power by reducing CPU stalls. Prefetch typically has a small power cost due to some flash reads being performed that ultimately are not needed.

Remark: Improper setting of this register may result in incorrect operation of the flash memory.

Offset

Register	Offset
FMCCR	400h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved				CLKDIV		Reserved				ECCABORTEN		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 W	FLASHTIM				Reserved					PREFOVR	PREFEN	ACCEL	DATACFG		FETCHCFG	
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-28 —	Reserved
27-26 CLKDIV	CLKDIV; default value is 00. 00 - 1 division

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01 - 2 division 10 - 3 division 11 - 4 division
25-22 —	Reserved
21-20 ECCABORTEN	ECC error abort enable 10 - disable others - enable; default value is 1
19-16 —	Reserved
15-12 FLASHTIM	Flash memory access time. The number of system clocks used for flash accesses is equal to FLASHTIM +1. 0000 - 1 system clock flash access time (for system clock rates up to 11 MHz). 0001 - 2 system clocks flash access time (for system clock rates up to 22 MHz). 0010 - 3 system clocks flash access time (for system clock rates up to 33 MHz). 0011 - 4 system clocks flash access time (for system clock rates up to 44 MHz). 0100 - 5 system clocks flash access time (for system clock rates up to 55 MHz). 0101 - 6 system clocks flash access time (for system clock rates up to 66 MHz). 0110 - 7 system clocks flash access time (for system clock rates up to 77 MHz). 0111 - 8 system clocks flash access time (for system clock rates up to 88 MHz). 1000 - 9 system clocks flash access time (for system clock rates up to 100 MHz). 1001 - 10 system clocks flash access time (for system clock rates up to 115 MHz). 1010 - 11 system clocks flash access time (for system clock rates up to 130 MHz). 1011 - 12 system clocks flash access time (for system clock rates up to 150 MHz).
11-7 —	Reserved
6 PREFOVR	Prefetch override. This bit only applies when PREFEN = 1 and a buffered instruction is completing for which the next flash line is not already buffered or being prefetched. 0 - Any previously initiated prefetch will be completed. 1 - Any previously initiated prefetch will be aborted, and the next flash line following the current execution address will be prefetched if not already buffered.

Table continues on the next page...

Table continued from the previous page...

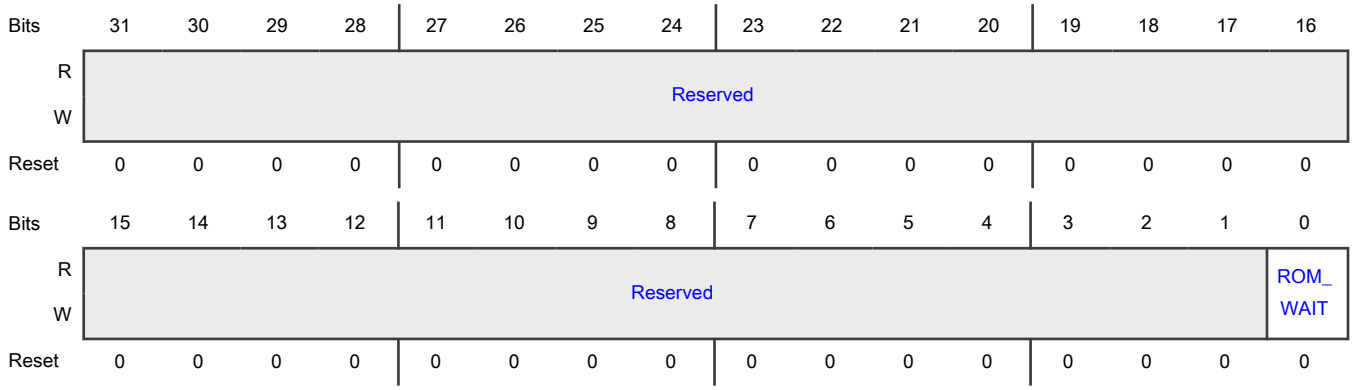
Field	Description
5 PREFEN	Prefetch enable. The prefetch bit must be disabled before executing any flash programming/erasing and flash controller commands 0 - No instruction prefetch is performed. 1 - Instruction prefetch is enabled. If the FETCHCFG field is not 0, the next flash line following the current execution address is automatically prefetched if it is not already buffered.
4 ACCEL	Acceleration enable. 0 - Flash acceleration is disabled. Every flash read (including those fulfilled from a buffer) takes FLASHTIM + 1 system clocks. This allows more determinism at a cost of performance. 1 - Flash acceleration is enabled. Performance is enhanced, dependent on other FMCCR settings.
3-2 DATACFG	Data read configuration. This field determines how flash accelerator buffers are used for data accesses. 00 - Data accesses from flash are not buffered. Every data access from the CPU results in a read of the flash memory. 01 - One buffer is used for all data accesses. 10 - All buffers can be used for data accesses.
1-0 FETCHCFG	Instruction fetch configuration. This field determines how flash accelerator buffers are used for instruction fetches. 00 - Instruction fetches from flash are not buffered. Every fetch request from the CPU results in a read of the flash memory. This setting may use significantly more power than when buffering is enabled. 01 - One buffer is used for all instruction fetches. 10 - All buffers may be used for instruction fetches.

8.5.1.1.65 ROM wait state (ROMCR)

Offset

Register	Offset
ROMCR	404h

Diagram



Fields

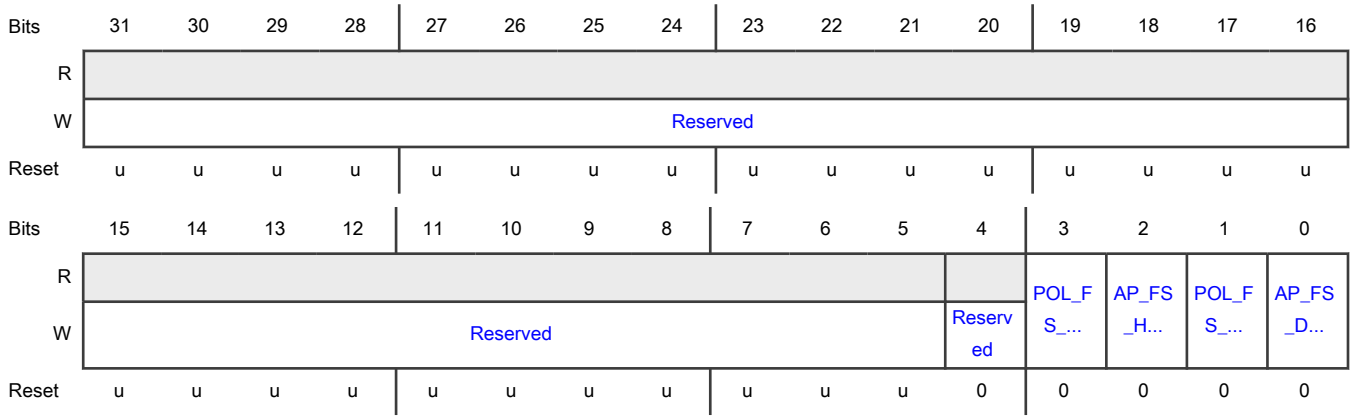
Field	Description
31-1 —	Reserved
0 ROM_WAIT	ROM waiting Arm core and other masters.

8.5.1.1.66 USB0-FS need clock control (USB0NEEDCLKCTRL)

Offset

Register	Offset
USB0NEEDCLKCTRL	40Ch

Diagram



Fields

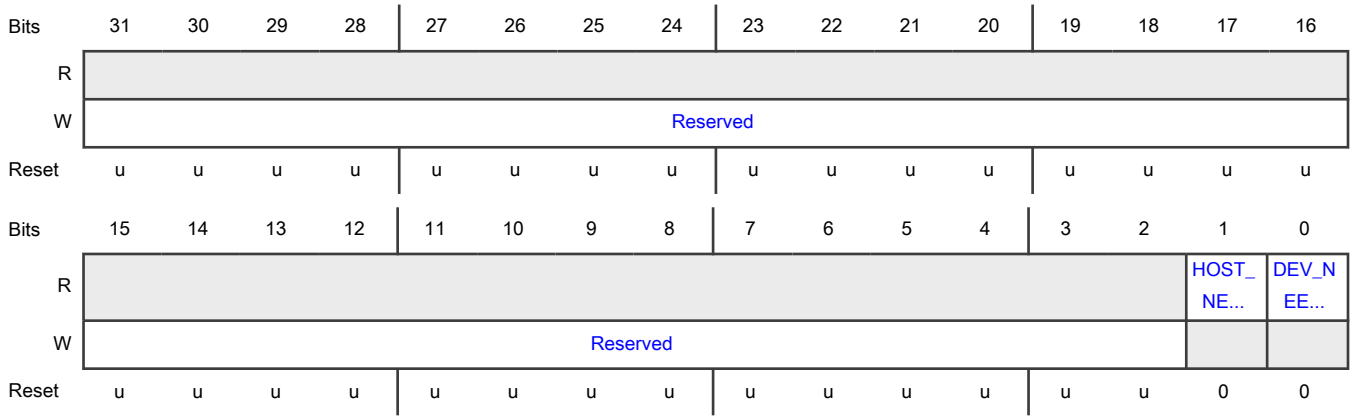
Field	Description
31-5 —	Reserved Read value is undefined, only zero should be written.
4 —	Reserved Read value is undefined, only zero should be written.
3 POL_FS_HOST_NEEDCLK	USB0-FS Host USB0_NEEDCLK polarity for triggering the USB0-FS wake-up interrupt:. 0 - Falling edge of device USB0_NEEDCLK triggers wake-up. 1 - Rising edge of device USB0_NEEDCLK triggers wake-up.
2 AP_FS_HOST_NEEDCLK	USB0-FS Host USB0_NEEDCLK signal control:. 0 - Under hardware control. 1 - Forced high.
1 POL_FS_DEV_NEEDCLK	USB0-FS Device USB0_NEEDCLK polarity for triggering the USB0-FS wake-up interrupt:. 0 - Falling edge of device USB0_NEEDCLK triggers wake-up. 1 - Rising edge of device USB0_NEEDCLK triggers wake-up.
0 AP_FS_DEV_NEEDCLK	USB0-FS Device USB0_NEEDCLK signal control:. 0 - Under hardware control. 1 - Forced high.

8.5.1.1.67 USB0-FS need clock status (USB0NEEDCLKSTAT)

Offset

Register	Offset
USB0NEEDCLKSTAT	410h

Diagram



Fields

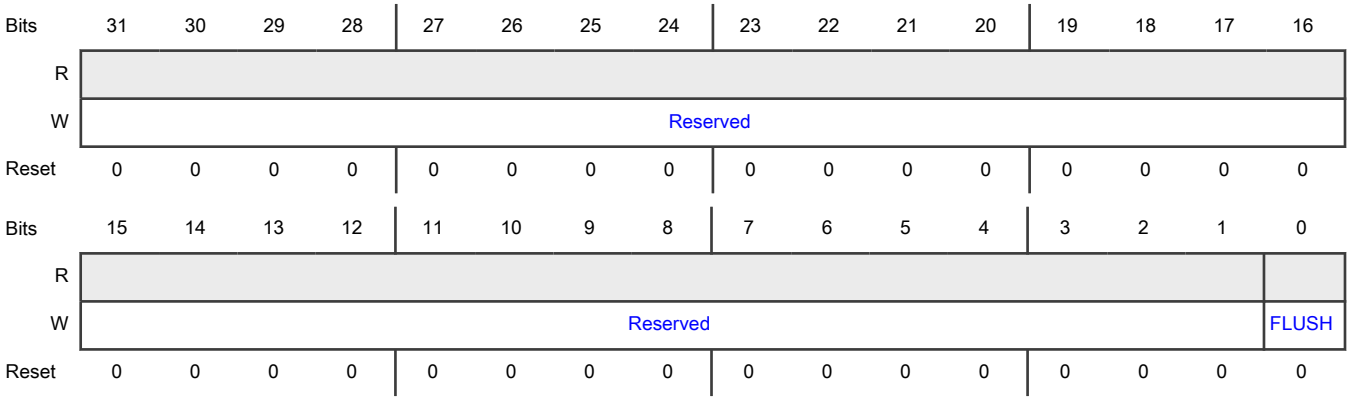
Field	Description
31-2 —	Reserved Read value is undefined, only zero should be written.
1 HOST_NEEDCLK	USB0-FS Host USB0_NEEDCLK signal status:. 0 - USB0-FS Host clock is low. 1 - USB0-FS Host clock is high.
0 DEV_NEEDCLK	USB0-FS Device USB0_NEEDCLK signal status:. 0 - USB0-FS Device clock is low. 1 - USB0-FS Device clock is high.

8.5.1.1.68 FMC flush control (FMCFLUSH)

Offset

Register	Offset
FMCFLUSH	41Ch

Diagram



Fields

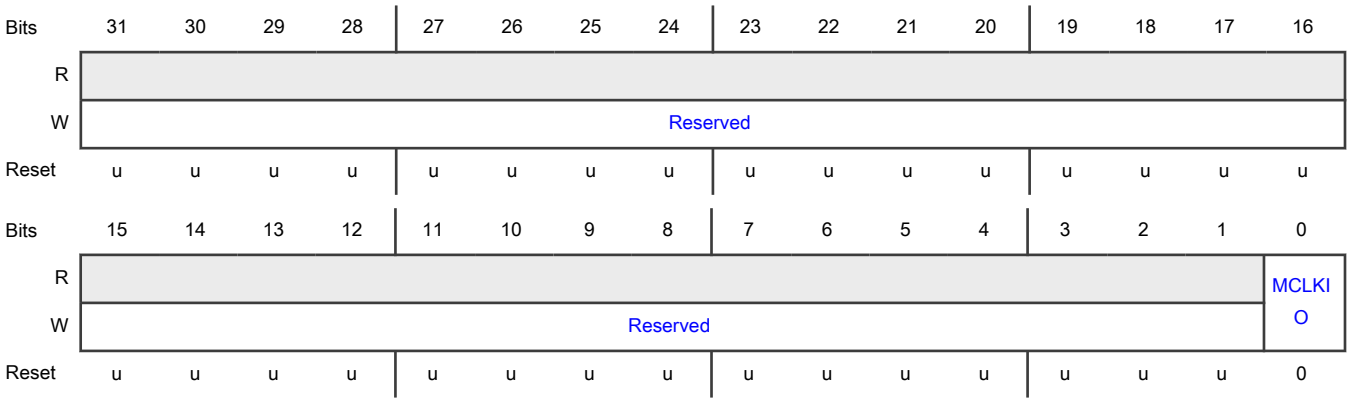
Field	Description
31-1 —	Reserved
0 FLUSH	Flush control 0 - No action. 1 - Flush the FMC buffer contents.

8.5.1.1.69 MCLK control (MCLKIO)

Offset

Register	Offset
MCLKIO	420h

Diagram



Fields

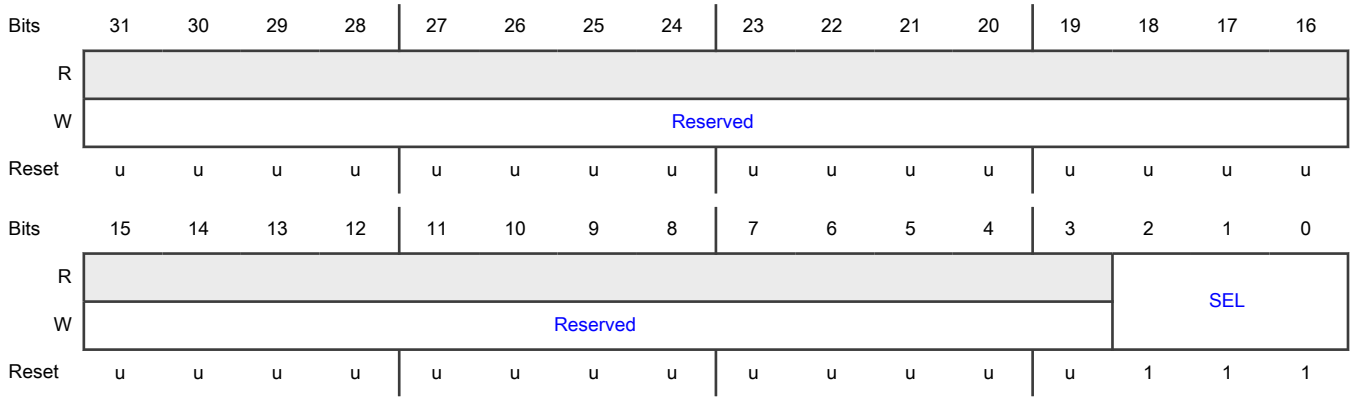
Field	Description
31-1 —	Reserved
0 MCLKIO	MCLK control. 0 - input mode. 1 - output mode.

8.5.1.1.70 ADC1 clock source select (ADC1CLKSEL)

Offset

Register	Offset
ADC1CLKSEL	464h

Diagram



Fields

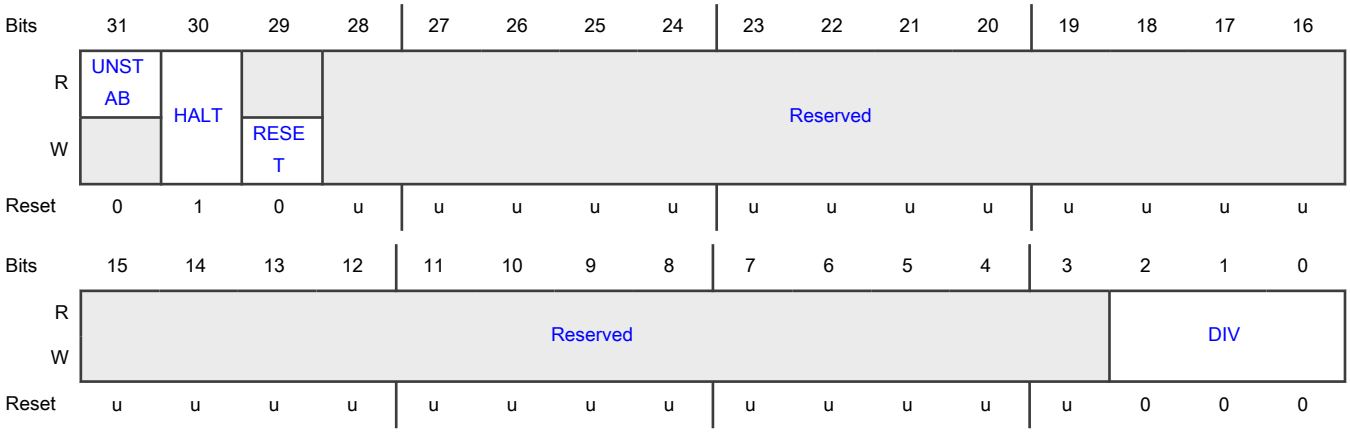
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	ADC clock source select. 000 - Main clock. 001 - PLL0 clock. 010 - FRO 96 MHz clock. 011 - Reserved. 100 - XO to ADC clock. 101 - No clock. 110 - No clock. 111 - No clock.

8.5.1.1.71 ADC1 clock divider (ADC1CLKDIV)

Offset

Register	Offset
ADC1CLKDIV	468h

Diagram



Fields

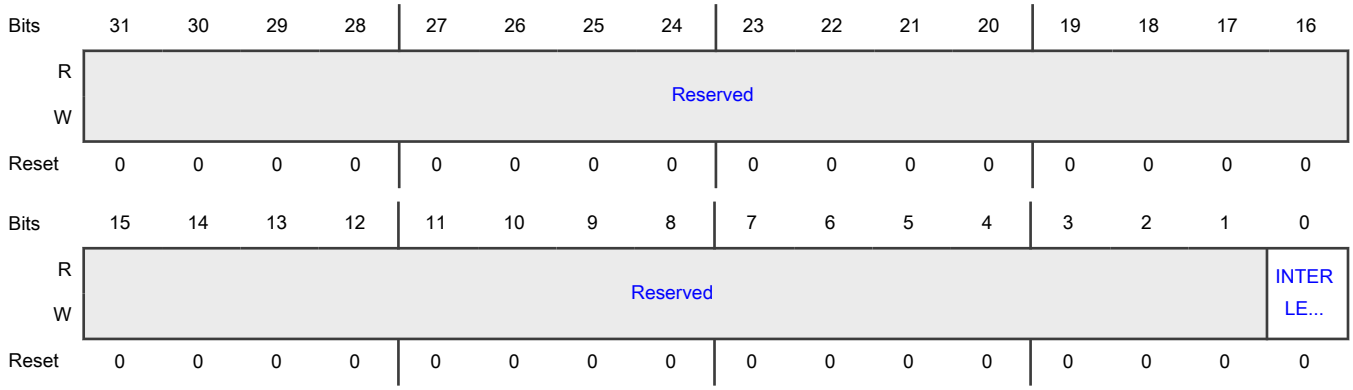
Field	Description
31 UNSTAB	Divider status flag. 0 - Divider clock is stable. 1 - Clock frequency is not stable.
30 HALT	Halts the divider counter. 0 - Divider clock is running. 1 - Divider clock is stopped.
29 RESET	Resets the divider counter. 0 - Divider is not reset. 1 - Divider is reset.
28-3 —	Reserved Read value is undefined, only zero should be written.
2-0 DIV	Clock divider value.

8.5.1.1.72 Control RAM interleave integration (RAM_INTERLEAVE)

Offset

Register	Offset
RAM_INTERLEAVE	470h

Diagram



Fields

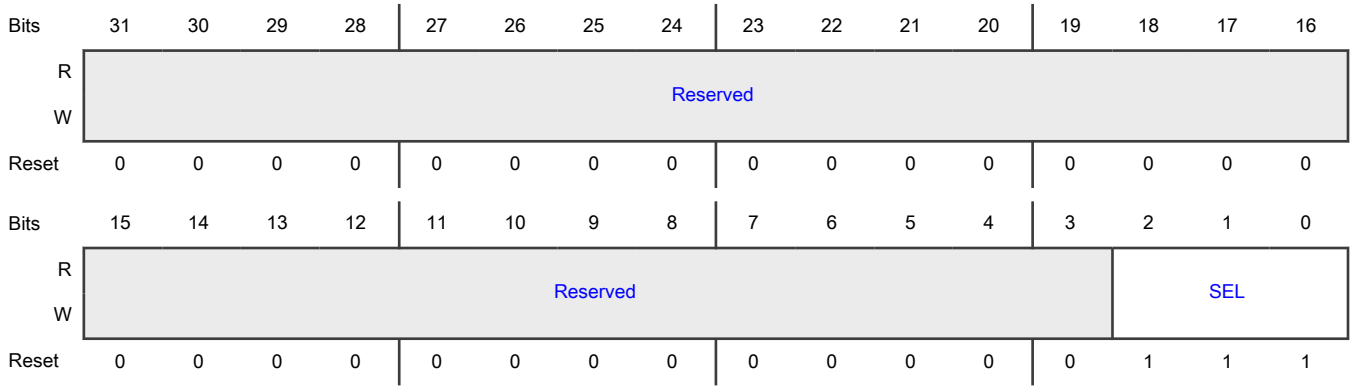
Field	Description
31-1 —	Reserved
0 INTERLEAVE	Control RAM access for RAM_02 and RAM_03. 0 - RAM access to RAM_02 and RAM_03 is consecutive. 1 - RAM access to RAM_02 and RAM_03 is interleaved.

8.5.1.1.73 DACn functional clock selection (DAC0CLKSEL - DAC2CLKSEL)

Offset

Register	Offset
DAC0CLKSEL	490h
DAC1CLKSEL	498h
DAC2CLKSEL	4A0h

Diagram



Fields

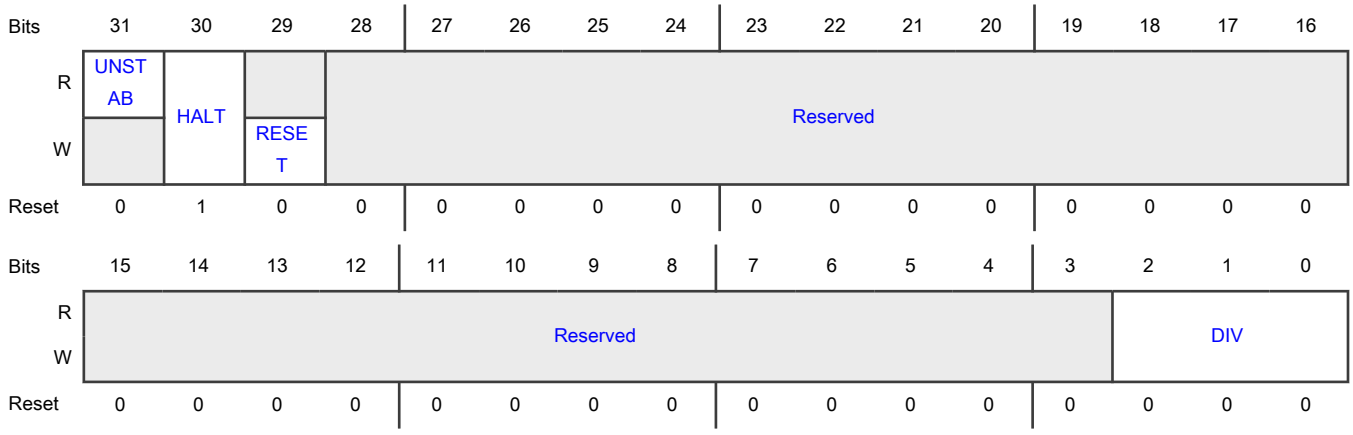
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2-0 SEL	DAC clock source select. 000 - Main clock. 001 - PLL0 clock. 010 - No clock. 011 - FRO_HF. 100 - FRO_12M. 101 - PLL1 clock. 110 - FRO_1M. 111 - No clock.

8.5.1.1.74 DACn functional clock divider (DAC0CLKDIV - DAC2CLKDIV)

Offset

Register	Offset
DAC0CLKDIV	494h
DAC1CLKDIV	49Ch
DAC2CLKDIV	4A4h

Diagram



Fields

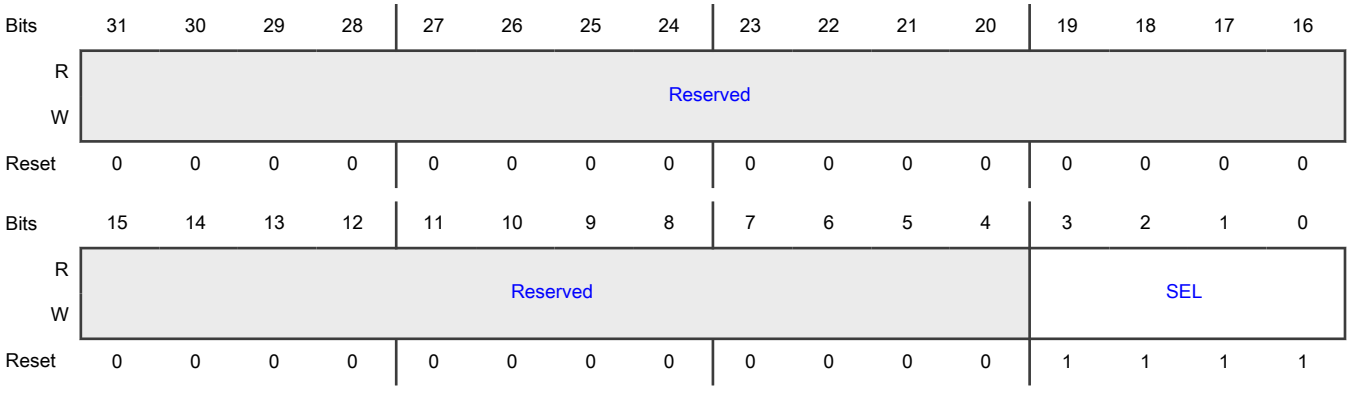
Field	Description
31 UNSTAB	Divider status flag. 0 - Divider clock is stable. 1 - Clock frequency is not stable.
30 HALT	Halts the divider counter. 0 - Divider clock is running. 1 - Divider clock is stopped.
29 RESET	Resets the divider counter. 0 - Divider is not reset. 1 - Divider is reset.
28-3 —	Reserved Read value is undefined, only zero should be written.
2-0 DIV	Clock divider value. The divider value = (DIV + 1).

8.5.1.1.75 FLEXSPI clock selection (FLEXSPICLKSEL)

Offset

Register	Offset
FLEXSPICLKSEL	4A8h

Diagram



Fields

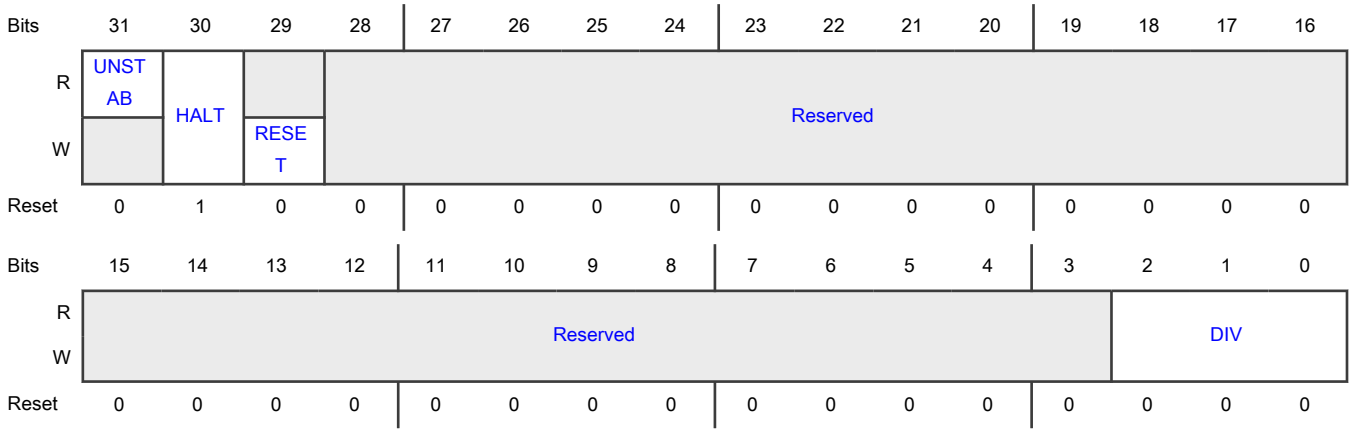
Field	Description
31-4 —	Reserved
3-0 SEL	Flexspi clock select 0000 - Main clock 0001 - PLL0 clock 0010 - No clock 0011 - FRO_HF 0100 - No clock 0101 - PLL1 clock 0110 - No clock 0111 - No clock 1000 - No clock 1001 - No clock 1010 - No clock 1011 - No clock 1100 - No clock 1101 - No clock 1110 - No clock 1111 - No clock

8.5.1.1.76 FLEXSPI clock divider (FLEXSPICLKDIV)

Offset

Register	Offset
FLEXSPICLKDIV	4ACh

Diagram



Fields

Field	Description
31 UNSTAB	Divider status flag. 0 - Divider clock is stable. 1 - Clock frequency is not stable.
30 HALT	Halts the divider counter. 0 - Divider clock is running. 1 - Divider clock is stopped.
29 RESET	Resets the divider counter. 0 - Divider is not reset. 1 - Divider is reset.
28-3 —	Reserved Read value is undefined, only zero should be written.
2-0 DIV	Clock divider value. The divider value = (DIV + 1).

8.5.1.1.77 Enable protection (CDPA_ENABLE)

This register along with CDPA_ENABLE_DP provides the enable and register write lock control as defined in the table below.

This register is writable only when CDPA_WRITE_EN is true. This register provides protection from intentional or unintentional modifications once the device is in field. It can be used to store data or can also be used as a second bootloader. To implement this feature at a certain location in the flash memory, the user can manipulate the CDPA_CONFIG bit fields, located at location 0x3E28C, in the CMPA region. The values provided in these bit fields are copied to SYSCON registers (CDPA_ENABLE, CDPA_ENABLE_DP, and CDPA_CONFIG) by the ROM.

CDPA_ENABLE is a part of the CDPA feature which allows the user to define an area between 0x0 and 0x3dc00 in the internal flash memory to protect it from erase and write access.

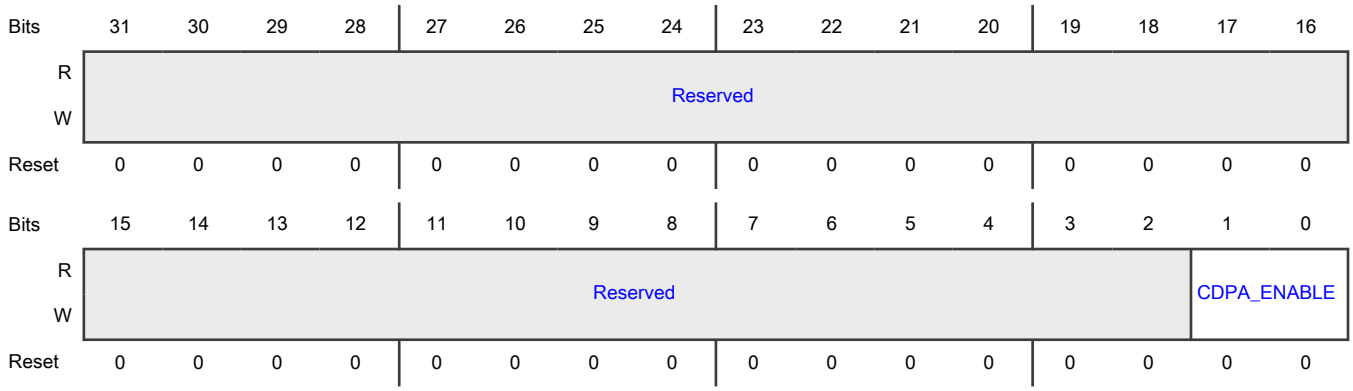
The area 0x0 and 0x3dc00 can be defined by user in internal flash memory to protect from erase and write access.

CDPA_ENABLE (1:0)	CDPA_ENABLE_DP (1:0)	CDPA Protection Status	CDPA_ENABLE Reg Writability
10	10	CDPA unprotected (CDPA Erase/Program/Read functions available)	Writable
01	10	CDPA protected (CDPA Read only function)	Writable
don't care	!="10"	CDPA protected (CDPA Read only function)	Write locked

Offset

Register	Offset
CDPA_ENABLE	4B0h

Diagram



Fields

Field	Description
31-2 —	Reserved
1-0 CDPA_ENABLE	Enable control The default values of this field is set to 0x2 after reset.

8.5.1.1.78 Enable protection duplicate (CDPA_ENABLE_DP)

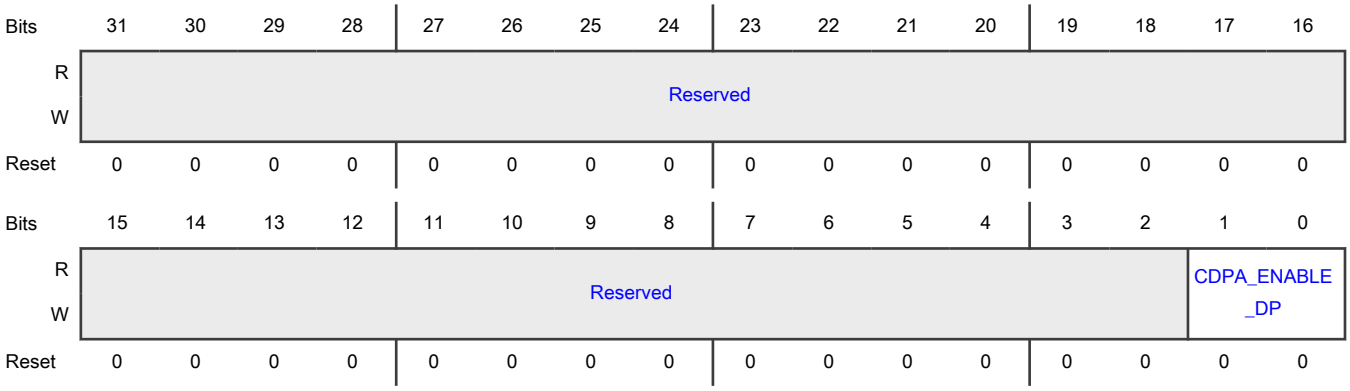
This register along with CDPA_ENABLE_DP provides the enable and register write lock control.

This register is writable only when CDPA_WRITE_EN is true.

Offset

Register	Offset
CDPA_ENABLE_DP	4B4h

Diagram



Fields

Field	Description
31-2 —	Reserved
1-0 CDPA_ENABLE_DP	Enable control The default values of this field is set to 0x2 after reset.

8.5.1.1.79 CDPA base address (CDPA_CONFIG)

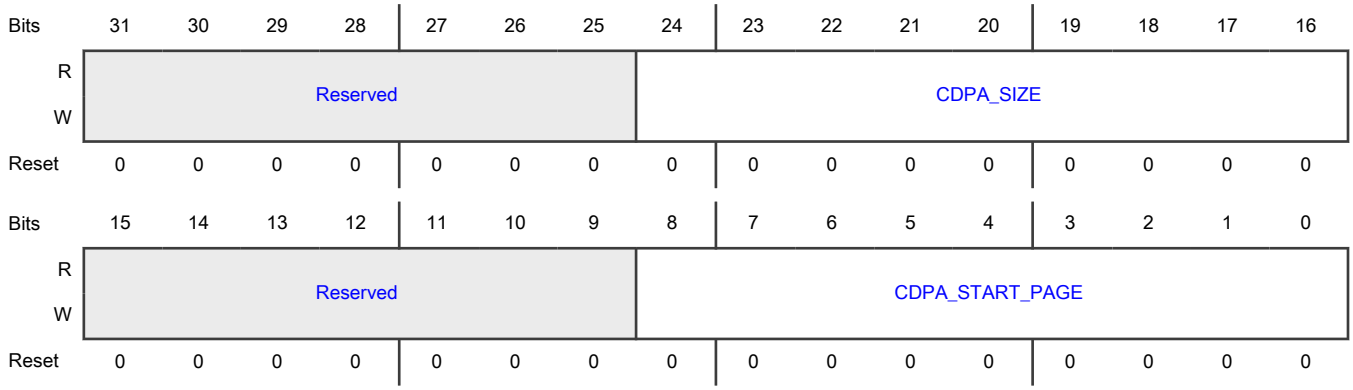
This register is writable only when CDPA_WRITE_EN is true.

This register sets the CDPA start page address and the size (number of pages).

Offset

Register	Offset
CDPA_CONFIG	4B8h

Diagram



Fields

Field	Description
31-25 —	Reserved
24-16 CDPA_SIZE	Specifies the size of CDPA in number of pages. The default values of this field is set 0x00 after reset. The flash page size for LPC553x devices is set to 512 bytes.
15-9 —	Reserved
8-0 CDPA_START_PAGE	Specifies the starting page number of CDPA. The default values of this field is set 0x00 after reset.

8.5.1.1.80 Flash hiding lockout address (FLASH_HIDING_LOCKOUT_ADDR)

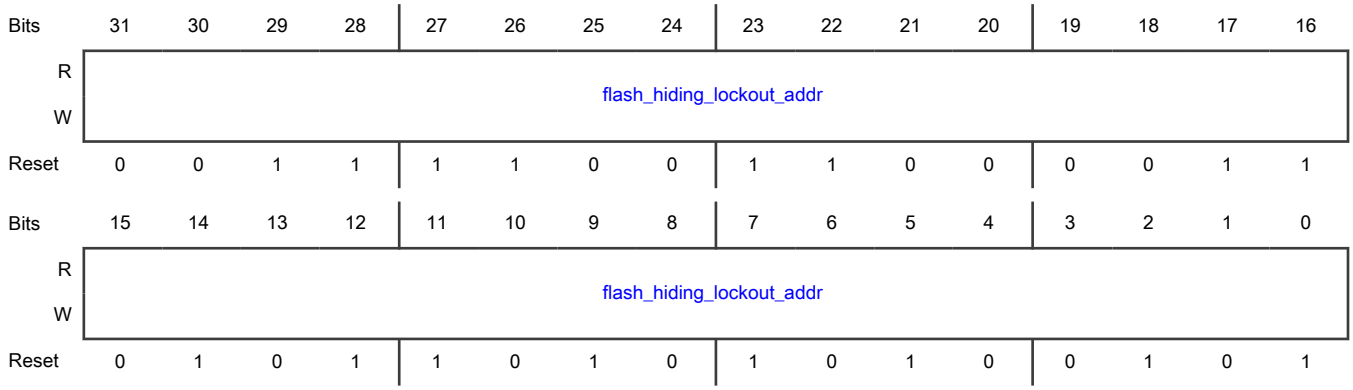
The register hides a flash area by disallowing the read operation on that area or on an area that includes a part of the flash hiding area.

User can use this flash hiding feature to hide data. The flash area where this feature would be implemented should not accommodate the user application, otherwise the application will run into a hard fault because of the inability to read that area. The enabling and disabling of the flash hiding feature can be done by manipulating the FLASH_HIDING_LOCKOUT_ADDR register of the SYSCON registers.

Offset

Register	Offset
FLASH_HIDING_LOCKOUT_ADDR	4D0h

Diagram



Fields

Field	Description
31-0 flash_hiding_lockout_addr	while flash hiding is disabled, register write is locked.

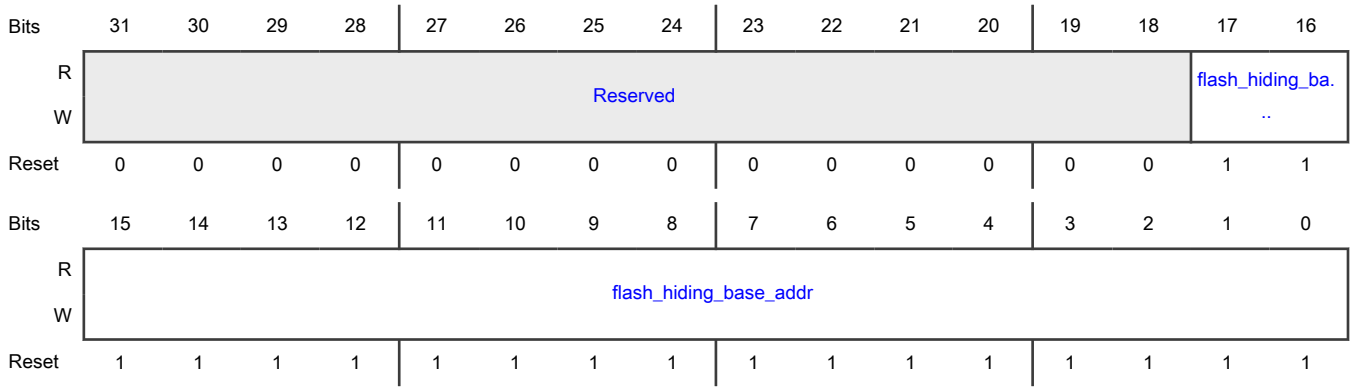
8.5.1.1.81 Flash hiding base address (FLASH_HIDING_BASE_ADDR)

The register allows setting of the base address for the region where the user wants to implement this feature.

Offset

Register	Offset
FLASH_HIDING_BASE_ADDR	4D4h

Diagram



Fields

Field	Description
31-18 —	Reserved
17-0 flash_hiding_base_addr	Base address for flash hiding For AHB port it is byte alignment and for APB port (flash IP read command), it is flash-word alignment (128b/16B).The 4 LSB bit ignored. **NOTE** Always program the FLASH_HIDING_BASE_ADDR register to flash word boundaries. Otherwise, there is a risk of exposing data on the APB bus.

8.5.1.1.82 Flash hiding base DP address (FLASH_HIDING_BASE_DP_ADDR)

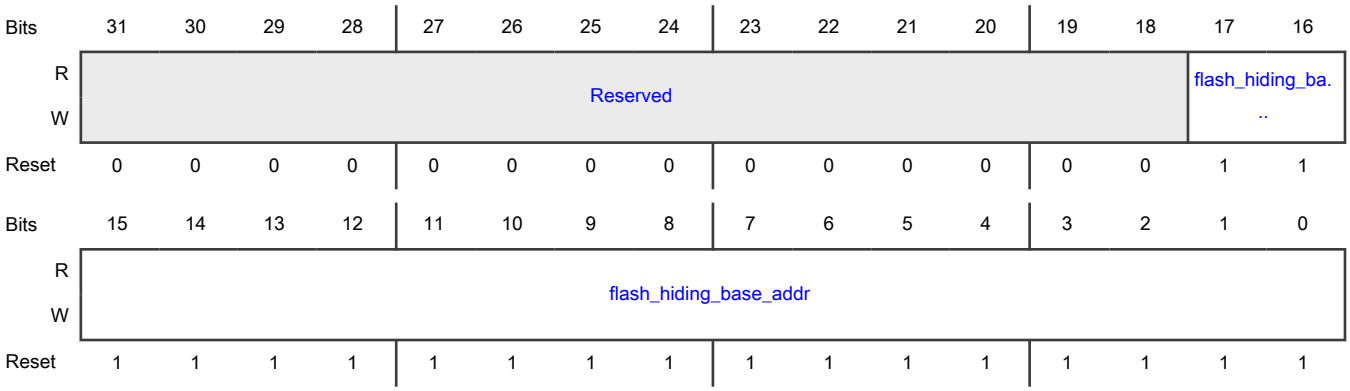
Both registers, FLASH_HIDING_BASE_ADDR and FLASH_HIDING_BASE_DP_ADDR needs to be filled with same value.

They provide added protection against intentional tampering of this register or glitches, since both of them needs to have same value in order for flash hiding function properly.

Offset

Register	Offset
FLASH_HIDING_BASE_DP_ADDR	4D8h

Diagram



Fields

Field	Description
31-18 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
17-0 flash_hiding_base_addr	Base address for flash hiding For AHB port it is byte alignment and for APB port (flash IP read command), it is flash-word alignment (128b/16B).The 4 LSB bit ignored. **NOTE** Always program the FLASH_HIDING_BASE_DP_ADDR register to flash word boundaries. Otherwise, there is a risk of exposing data on the APB bus.

8.5.1.1.83 Hiding size address (FLASH_HIDING_SIZE_ADDR)

This register allows setting of the size address for the region where the user wants to implement flash hiding. This value is the offset size added to the base register.

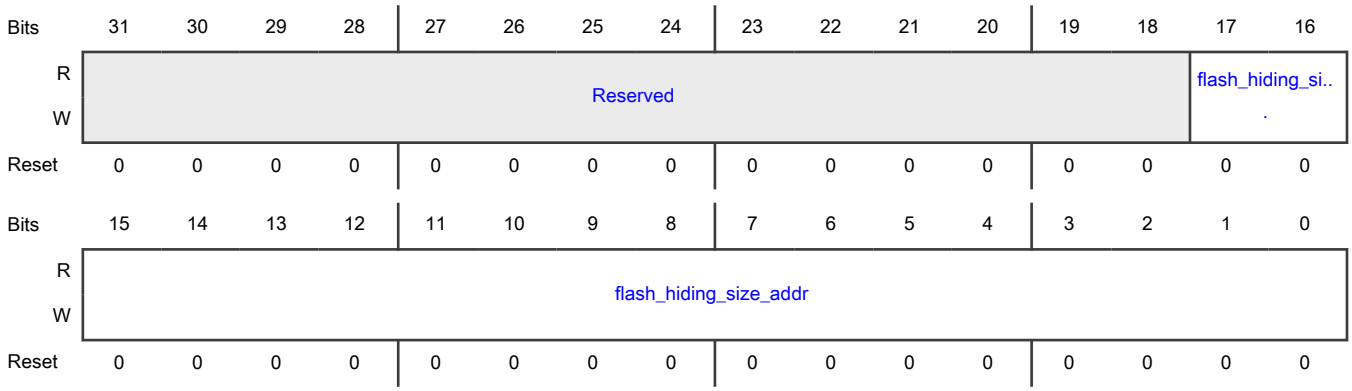
NOTE

For example, if the base address register is set to 0x20000 and the size address register is set to 0x800. The flash hiding area will be until 0x20800 (0x20000 + 0x800).

Offset

Register	Offset
FLASH_HIDING_SIZE_ADDR	4DCh

Diagram



Fields

Field	Description
31-18 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
17-0 flash_hiding_size_addr	Size address for flash hiding For AHB port it is byte alignment and for APB port (flash IP read command), it is flash-word alignment (128b/16B).The 4 LSB bit ignored. **NOTE** Always program the FLASH_HIDING_SIZE_ADDR register to flash word boundaries. Otherwise, there is a risk of exposing data on the APB bus.

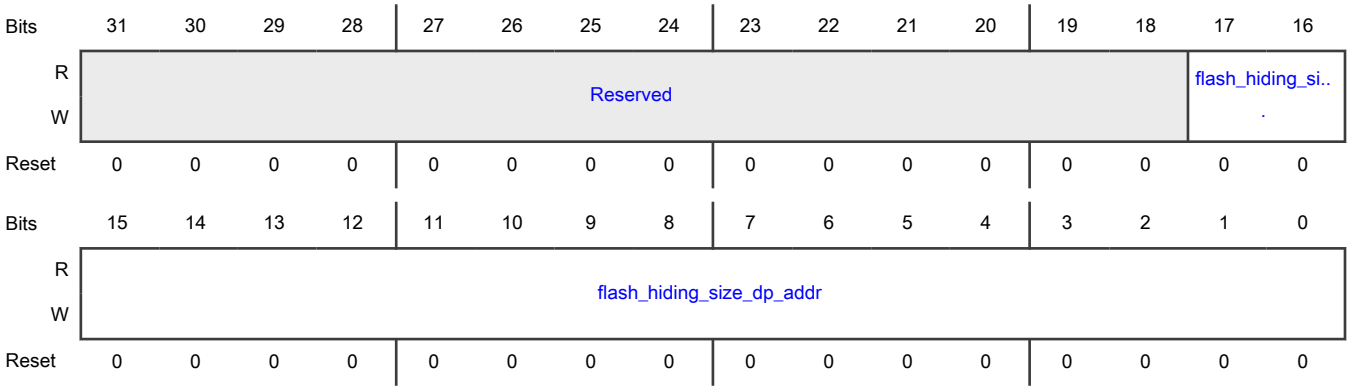
8.5.1.1.84 Hiding size DP address (FLASH_HIDING_SIZE_DP_ADDR)

Both the registers, FLASH_HIDING_SIZE_ADDR and FLASH_HIDING_SIZE_DP_ADDR needs to be filled with same value. They provide added protection against intentional tampering of this register or glitches, since both of them needs to have same value in order for flash hiding function properly.

Offset

Register	Offset
FLASH_HIDING_SIZE_DP_ADDR	4E0h

Diagram



Fields

Field	Description
31-18 —	Reserved
17-0	Size address for flash hiding For AHB port it is byte alignment and for APB port (flash IP read command), it is flash-word alignment (128b/16B).The 4 LSB bit ignored.

Table continues on the next page...

Table continued from the previous page...

Field	Description
flash_hiding_size_dp_addr	**NOTE** Always program the FLASH_HIDING_SIZE_DP_ADDR register to flash word boundaries. Otherwise, there is a risk of exposing data on the APB bus.

8.5.1.1.85 PLL clock divider clock selection (PLLCLKDIVSEL)

Offset

Register	Offset
PLLCLKDIVSEL	52Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	Reserved													SEL		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Fields

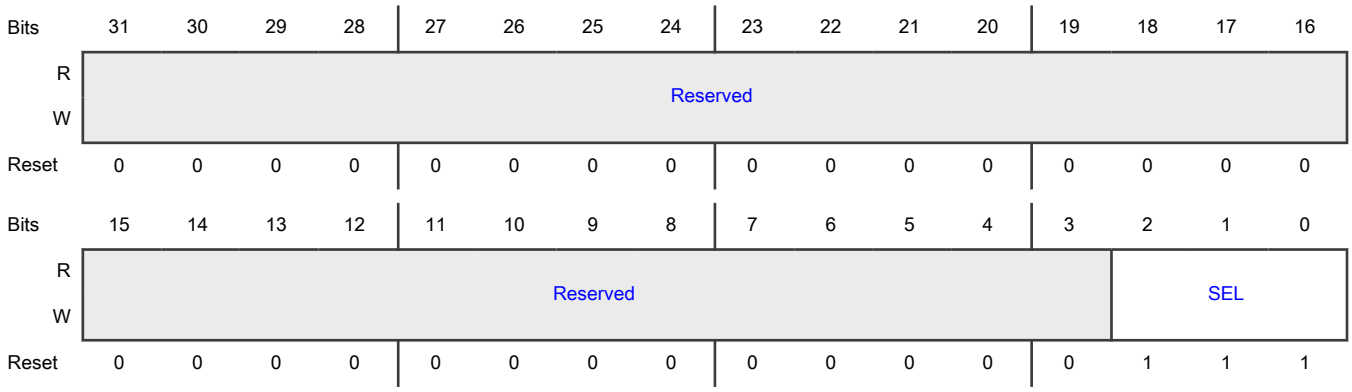
Field	Description
31-3 —	Reserved
2-0 SEL	Flexspi clock select 000 - PLL0 clock 001 - PLL1 clock 010 - No clock 011 - No clock 100 - No clock 101 - No clock 110 - No clock 111 - No clock

8.5.1.1.86 I3C functional clock selection (I3CFCLKSEL)

Offset

Register	Offset
I3CFCLKSEL	530h

Diagram



Fields

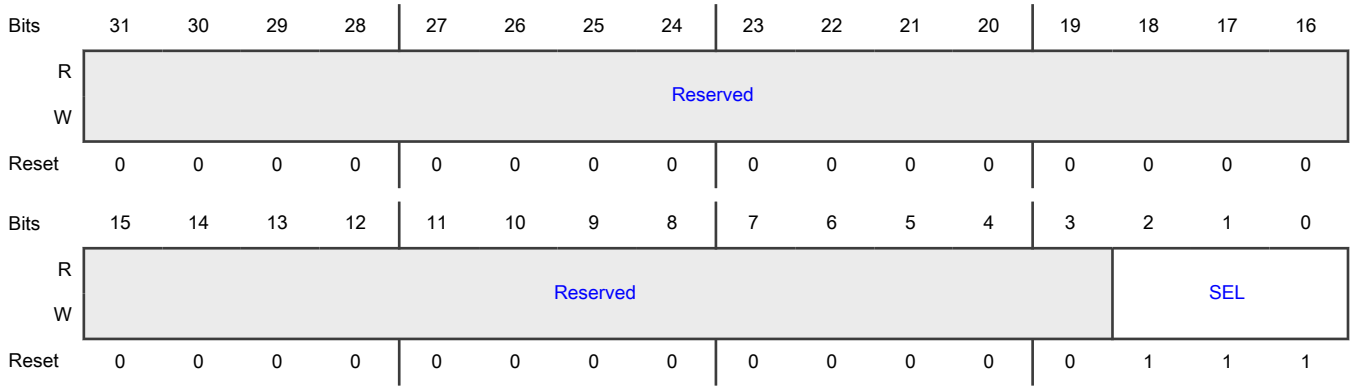
Field	Description
31-3 —	Reserved
2-0 SEL	I3C clock select 000 - Main clock 001 - FRO_HF_DIV 010 - No clock 011 - No clock 100 - No clock 101 - No clock 110 - No clock 111 - No clock

8.5.1.1.87 I3C FCLK_STC clock selection (I3CFCLKSTCSEL)

Offset

Register	Offset
I3CFCLKSTCSEL	534h

Diagram



Fields

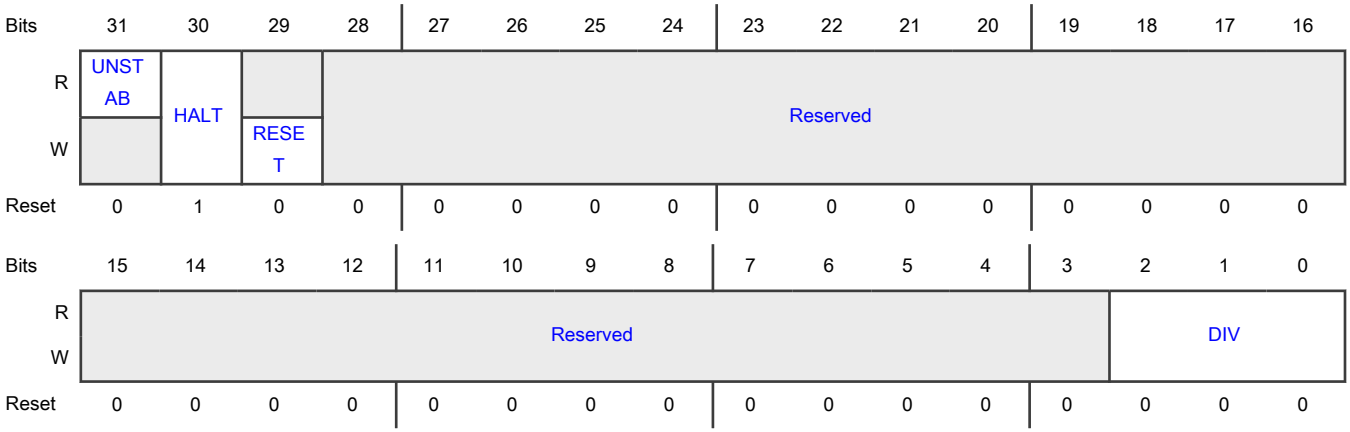
Field	Description
31-3 —	Reserved
2-0 SEL	I3C FCLK_STC clock select 000 - I3CFCLK 001 - FRO_1M 010 - No clock 011 - No clock 100 - No clock 101 - No clock 110 - No clock 111 - No clock

8.5.1.1.88 I3C FCLK_STC clock divider (I3CFCLKSTCDIV)

Offset

Register	Offset
I3CFCLKSTCDIV	538h

Diagram



Fields

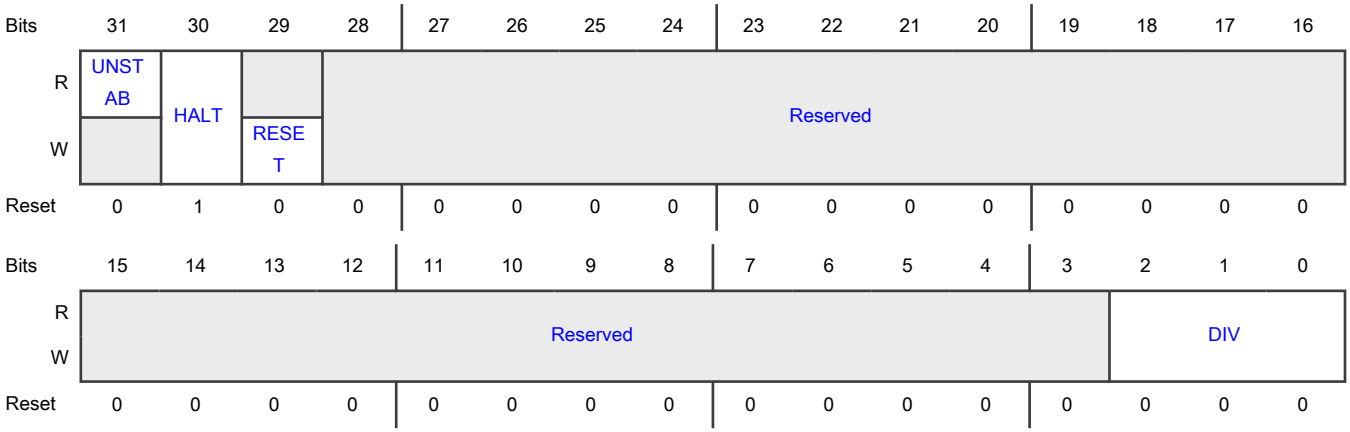
Field	Description
31 UNSTAB	Divider status flag. 0 - Divider clock is stable. 1 - Clock frequency is not stable.
30 HALT	Halts the divider counter. 0 - Divider clock is running. 1 - Divider clock is stopped.
29 RESET	Resets the divider counter. 0 - Divider is not reset. 1 - Divider is reset.
28-3 —	Reserved Read value is undefined, only zero should be written.
2-0 DIV	Clock divider value. The divider value = (DIV + 1).

8.5.1.1.89 I3C FCLKS clock divider (I3CFCLKSDIV)

Offset

Register	Offset
I3CFCLKSDIV	53Ch

Diagram



Fields

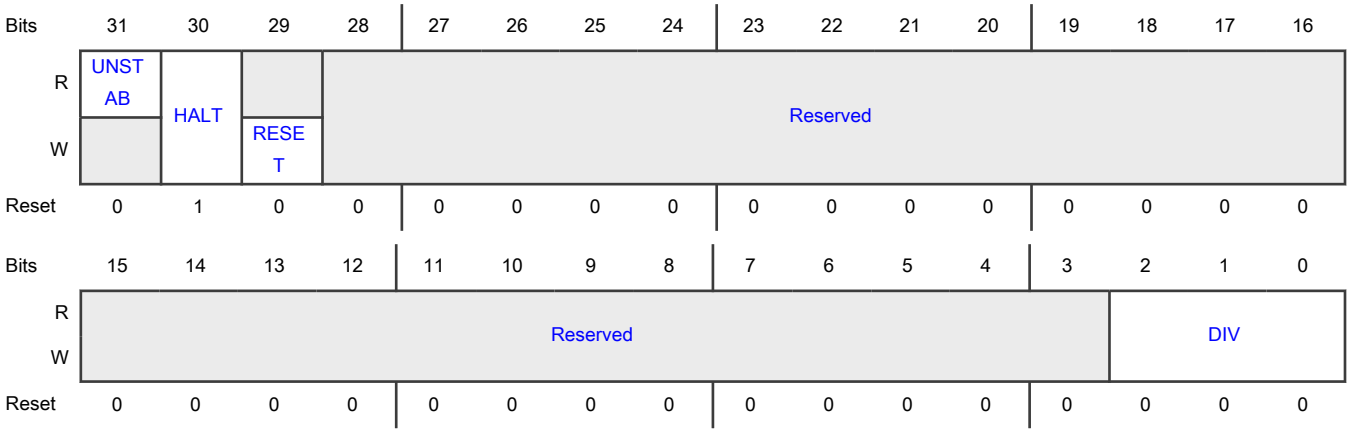
Field	Description
31 UNSTAB	Divider status flag. 0 - Divider clock is stable. 1 - Clock frequency is not stable.
30 HALT	Halts the divider counter. 0 - Divider clock is running. 1 - Divider clock is stopped.
29 RESET	Resets the divider counter. 0 - Divider is not reset. 1 - Divider is reset.
28-3 —	Reserved Read value is undefined, only zero should be written.
2-0 DIV	Clock divider value. The divider value = (DIV + 1).

8.5.1.1.90 I3C FCLK divider (I3CFCLKDIV)

Offset

Register	Offset
I3CFCLKDIV	540h

Diagram



Fields

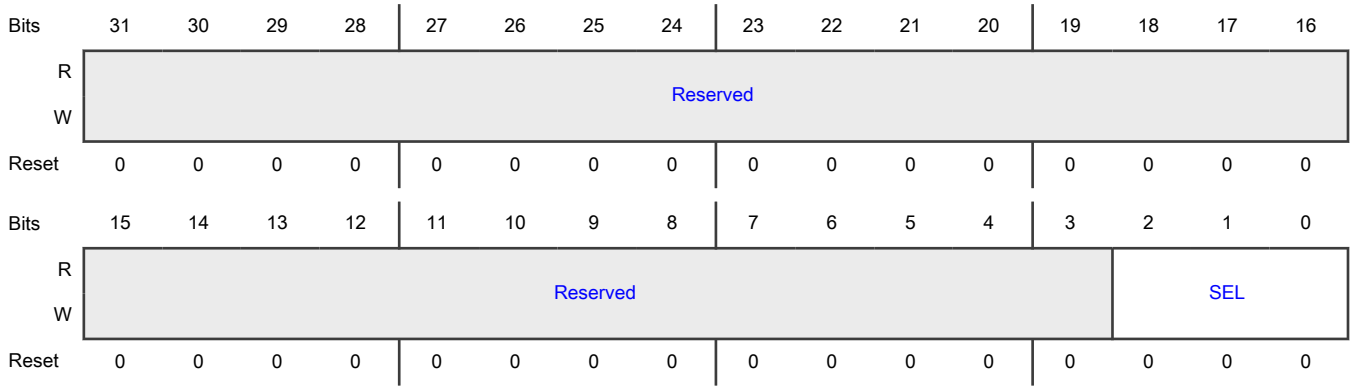
Field	Description
31 UNSTAB	Divider status flag. 0 - Divider clock is stable. 1 - Clock frequency is not stable.
30 HALT	Halts the divider counter. 0 - Divider clock is running. 1 - Divider clock is stopped.
29 RESET	Resets the divider counter. 0 - Divider is not reset. 1 - Divider is reset.
28-3 —	Reserved Read value is undefined, only zero should be written.
2-0 DIV	Clock divider value. The divider value = (DIV + 1).

8.5.1.1.91 I3C FCLK_S selection (I3CFCLKSSEL)

Offset

Register	Offset
I3CFCLKSSEL	544h

Diagram



Fields

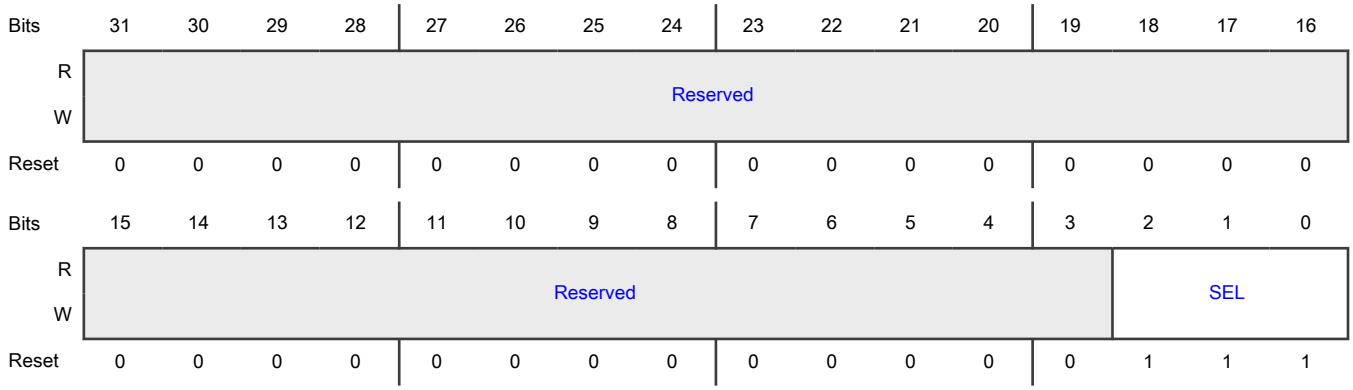
Field	Description
31-3 —	Reserved
2-0 SEL	I3C FCLK_S clock select 000 - FRO_1M 001 - No clock 010 - No clock 011 - No clock 100 - No clock 101 - No clock 110 - No clock 111 - No clock

8.5.1.1.92 DMIC clock selection (DMICFCLKSEL)

Offset

Register	Offset
DMICFCLKSEL	548h

Diagram



Fields

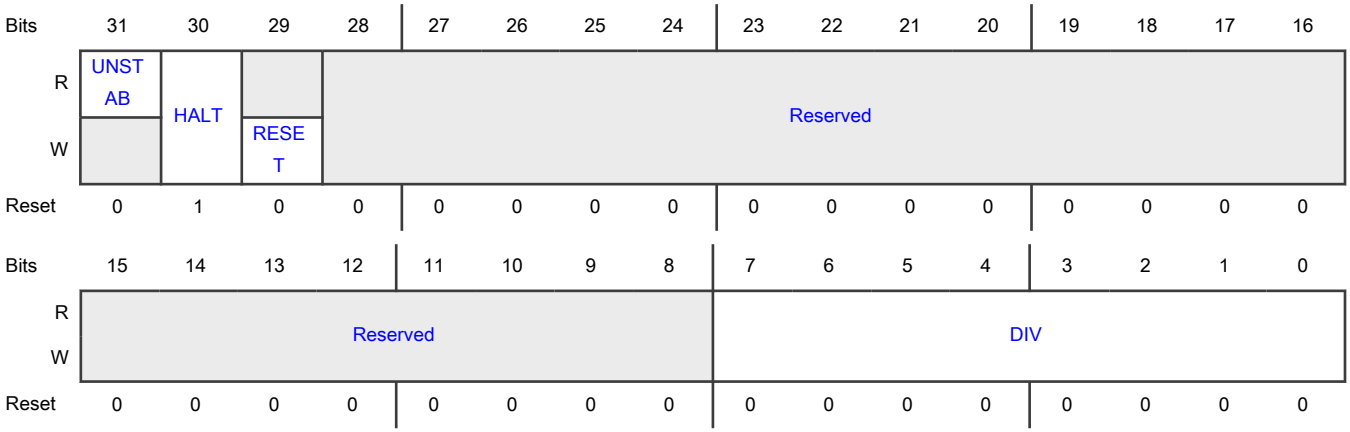
Field	Description
31-3 —	Reserved
2-0 SEL	DMIC clock select 000 - Main clock 001 - PLL0 clock 010 - Clock in 011 - FRO_HF 100 - PLL1 clock 101 - MCLK in 110 - No clock 111 - No clock

8.5.1.1.93 DMIC clock division (DMICFCLKDIV)

Offset

Register	Offset
DMICFCLKDIV	54Ch

Diagram



Fields

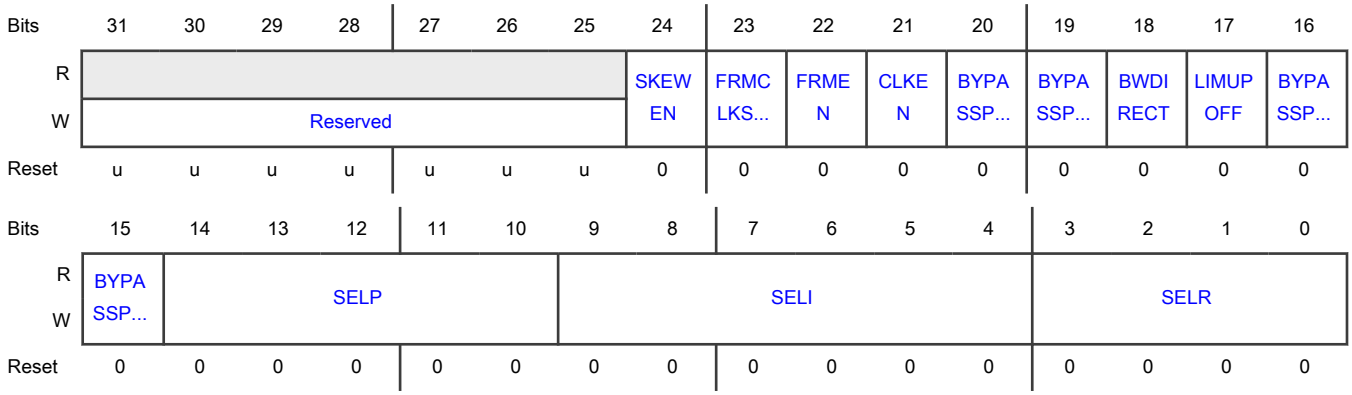
Field	Description
31 UNSTAB	Divider status flag. 0 - Divider clock is stable. 1 - Clock frequency is not stable.
30 HALT	Halts the divider counter. 0 - Divider clock is running. 1 - Divider clock is stopped.
29 RESET	Resets the divider counter. 0 - Divider is not reset. 1 - Divider is reset.
28-8 —	Reserved Read value is undefined, only zero should be written.
7-0 DIV	Clock divider value. The divider value = (DIV + 1).

8.5.1.1.94 PLL1 550m control (PLL1CTRL)

Offset

Register	Offset
PLL1CTRL	560h

Diagram



Fields

Field	Description
31-25 —	Reserved Read value is undefined, only zero should be written.
24 SKEWEN	Skew mode. 0 - skewmode is disable. 1 - skewmode is enable.
23 FRMCLKSTABLE	free running mode clockstable: Warning: Only make frm_clockstable = 1 after the PLL output frequency is stable.
22 FRMEN	1: free running mode.
21 CLKEN	enable the output clock. 0 - Disable the output clock. 1 - Enable the output clock.
20 BYPASSPOSTDIV	bypass of the post-divider. 0 - use the post-divider. 1 - bypass of the post-divider.
19 BYPASSPREDIV	bypass of the pre-divider. 0 - use the pre-divider. 1 - bypass of the pre-divider.
18 BWDIRECT	control of the bandwidth of the PLL. 0 - the bandwidth is changed synchronously with the feedback-divider. 1 - modify the bandwidth of the PLL directly.

Table continues on the next page...

Table continued from the previous page...

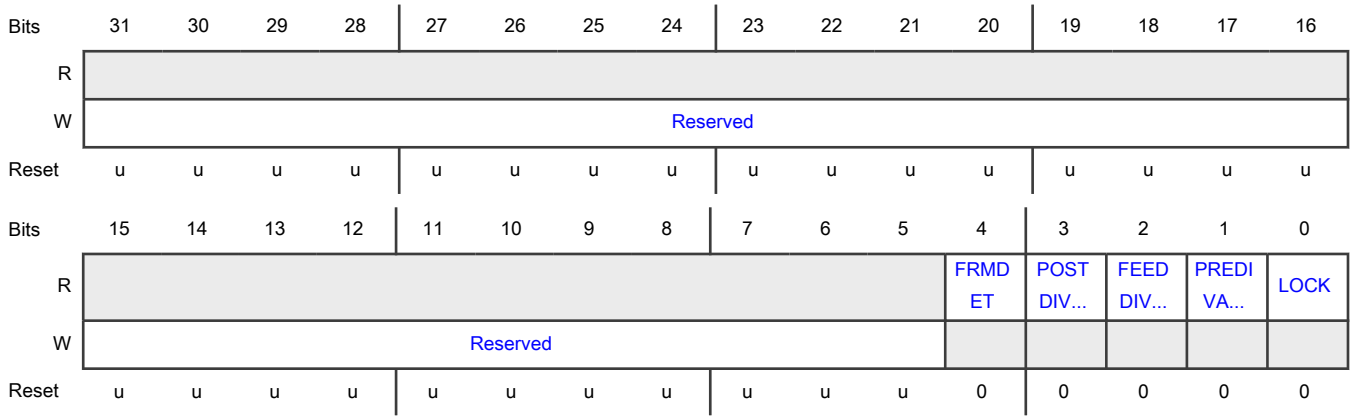
Field	Description
17 LIMUPOFF	limup_off = 1 in spread spectrum and fractional PLL applications.
16 BYPASSPOSTDIV2	bypass of the divide-by-2 divider in the post-divider. 0 - use the divide-by-2 divider in the post-divider. 1 - bypass of the divide-by-2 divider in the post-divider.
15 BYPASSPLL	Bypass PLL input clock is sent directly to the PLL output (default). 0 - use PLL. 1 - PLL input clock is sent directly to the PLL output.
14-10 SELP	Bandwidth select P value.
9-4 SELI	Bandwidth select I value.
3-0 SELR	Bandwidth select R value.

8.5.1.1.95 PLL1 550m status (PLL1STAT)

Offset

Register	Offset
PLL1STAT	564h

Diagram



Fields

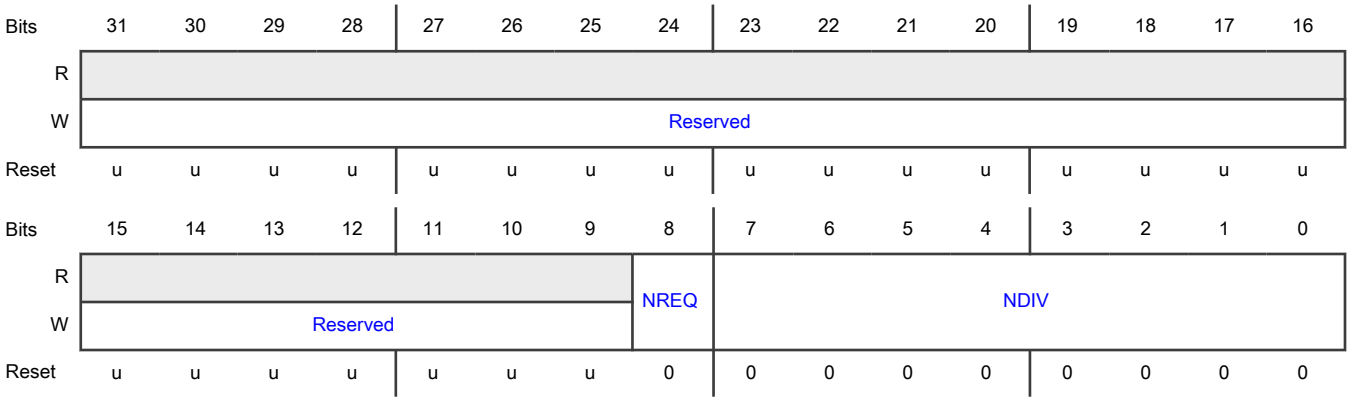
Field	Description
31-5 —	Reserved Read value is undefined, only zero should be written.
4 FRMDET	free running detector output (active high).
3 POSTDIVACK	post-divider ratio change acknowledge.
2 FEEDDIVACK	feedback divider ratio change acknowledge.
1 PREDIVACK	pre-divider ratio change acknowledge.
0 LOCK	lock detector output (active high) Warning: The lock signal is only reliable between fref[2] :100 kHz to 20 MHz.

8.5.1.1.96 PLL1 550m N divider (PLL1NDEC)

Offset

Register	Offset
PLL1NDEC	568h

Diagram



Fields

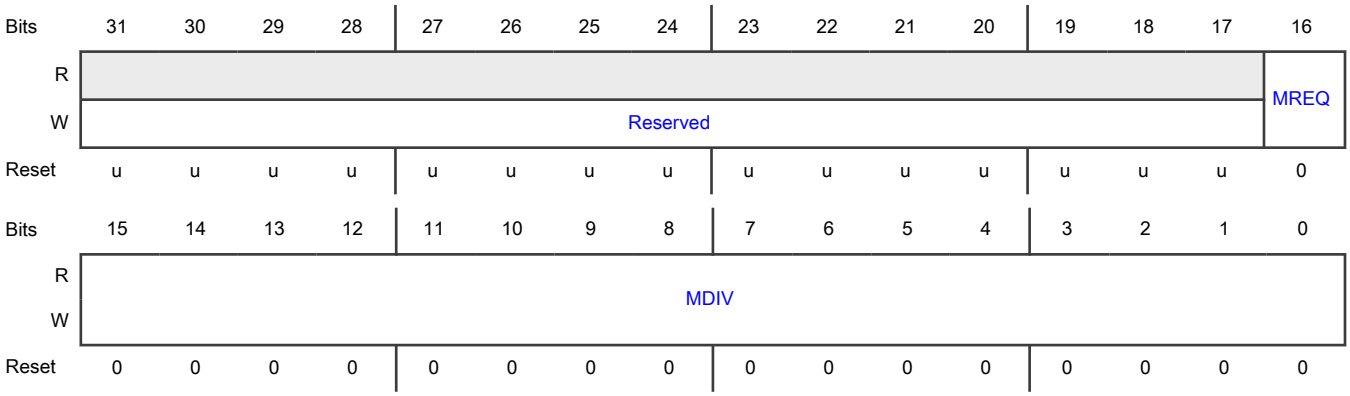
Field	Description
31-9 —	Reserved Read value is undefined, only zero should be written.
8 NREQ	pre-divider ratio change request.
7-0 NDIV	pre-divider divider ratio (N-divider).

8.5.1.1.97 PLL1 550m M divider (PLL1MDEC)

Offset

Register	Offset
PLL1MDEC	56Ch

Diagram



Fields

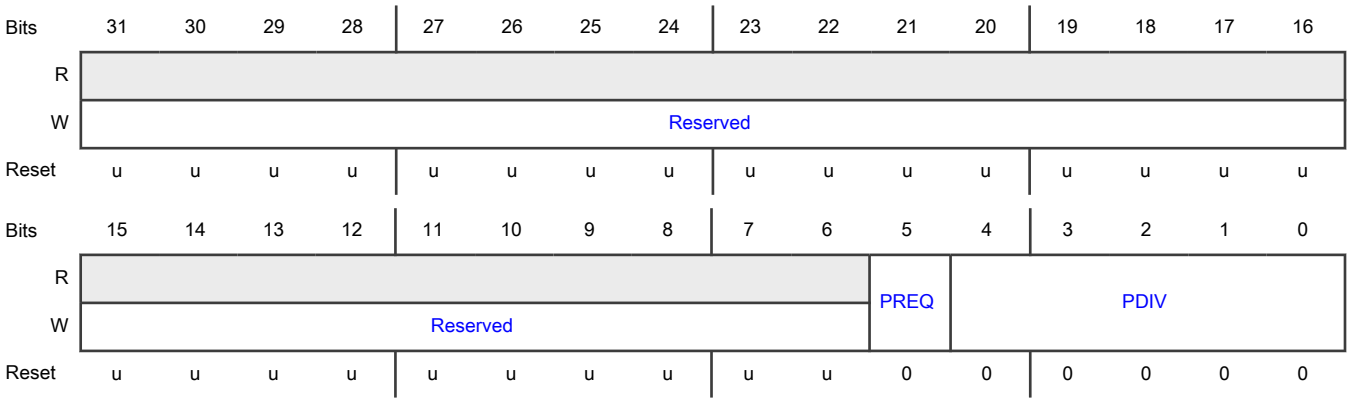
Field	Description
31-17 —	Reserved Read value is undefined, only zero should be written.
16 MREQ	feedback ratio change request.
15-0 MDIV	feedback divider divider ratio (M-divider).

8.5.1.1.98 PLL1 550m P divider (PLL1PDEC)

Offset

Register	Offset
PLL1PDEC	570h

Diagram



Fields

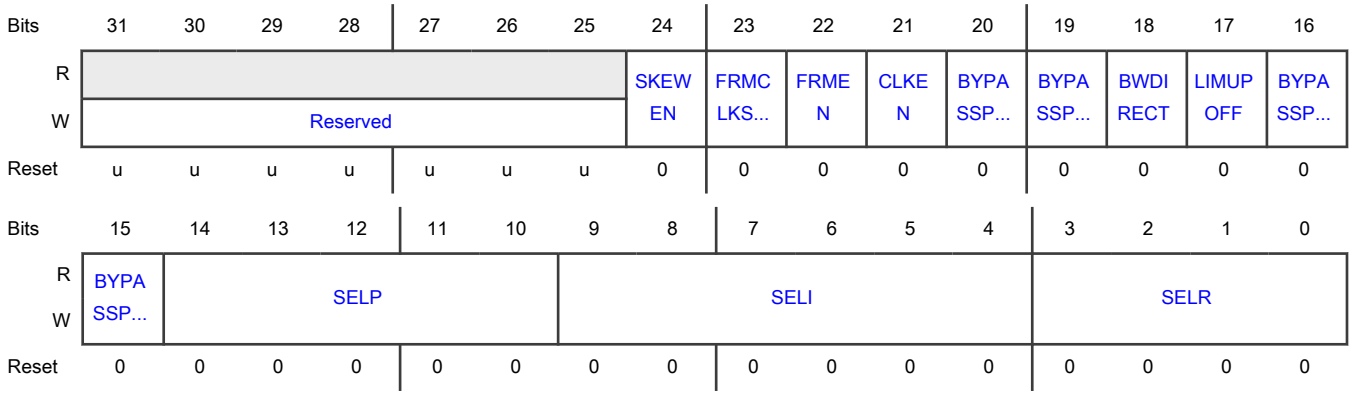
Field	Description
31-6 —	Reserved Read value is undefined, only zero should be written.
5 PREQ	feedback ratio change request.
4-0 PDIV	post-divider divider ratio (P-divider)

8.5.1.1.99 PLL0 550m control (PLL0CTRL)

Offset

Register	Offset
PLL0CTRL	580h

Diagram



Fields

Field	Description
31-25 —	Reserved Read value is undefined, only zero should be written.
24 SKEWEN	skew mode. 0 - skew mode is disable. 1 - skew mode is enable.
23 FRMCLKSTABLE	free running mode clockstable: Warning: Only make frm_clockstable =1 after the PLL output frequency is stable.
22 FRMEN	free running mode. 0 - free running mode is disable. 1 - free running mode is enable.
21 CLKEN	enable the output clock. 0 - disable the output clock. 1 - enable the output clock.
20 BYPASSPOSTDIV	bypass of the post-divider. 0 - use the post-divider. 1 - bypass of the post-divider.
19 BYPASSPREDIV	bypass of the pre-divider. 0 - use the pre-divider. 1 - bypass of the pre-divider.
18 BWDIRECT	Control of the bandwidth of the PLL. 0 - the bandwidth is changed synchronously with the feedback-divider.

Table continues on the next page...

Table continued from the previous page...

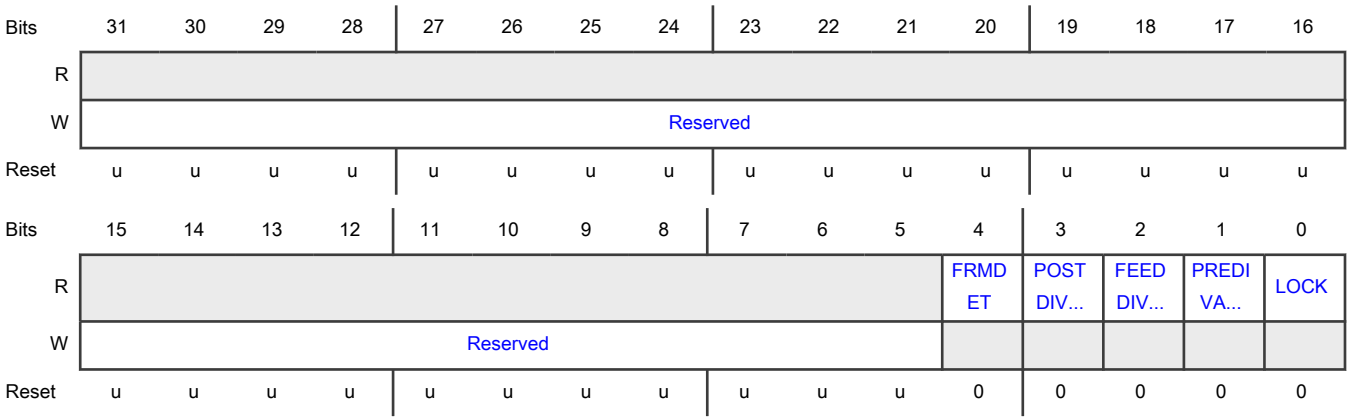
Field	Description
	1 - modify the bandwidth of the PLL directly.
17 LIMUPOFF	limup_off = 1 in spread spectrum and fractional PLL applications.
16 BYPASSPOSTDIV2	bypass of the divide-by-2 divider in the post-divider. 0 - use the divide-by-2 divider in the post-divider. 1 - bypass of the divide-by-2 divider in the post-divider.
15 BYPASSPLL	Bypass PLL input clock is sent directly to the PLL output (default). 0 - use PLL. 1 - Bypass PLL input clock is sent directly to the PLL output.
14-10 SELP	Bandwidth select P value.
9-4 SELI	Bandwidth select I value.
3-0 SELR	Bandwidth select R value.

8.5.1.1.100 PLL0 550m status (PLL0STAT)

Offset

Register	Offset
PLL0STAT	584h

Diagram



Fields

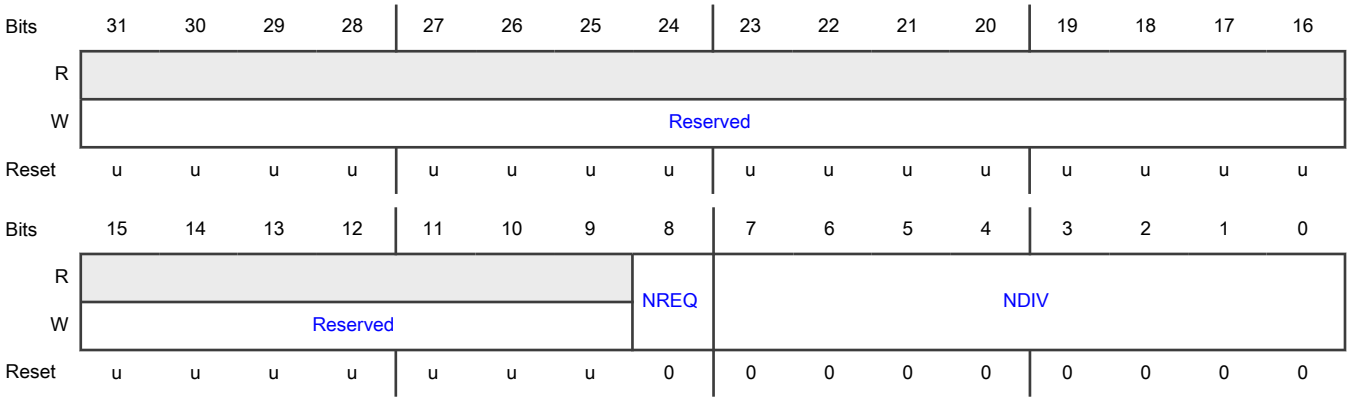
Field	Description
31-5 —	Reserved Read value is undefined, only zero should be written.
4 FRMDET	free running detector output (active high).
3 POSTDIVACK	post-divider ratio change acknowledge.
2 FEEDDIVACK	feedback divider ratio change acknowledge.
1 PREDIVACK	pre-divider ratio change acknowledge.
0 LOCK	lock detector output (active high) Warning: The lock signal is only reliable between fref[2] :100 kHz to 20 MHz.

8.5.1.1.101 PLL0 550m N divider (PLL0NDEC)

Offset

Register	Offset
PLL0NDEC	588h

Diagram



Fields

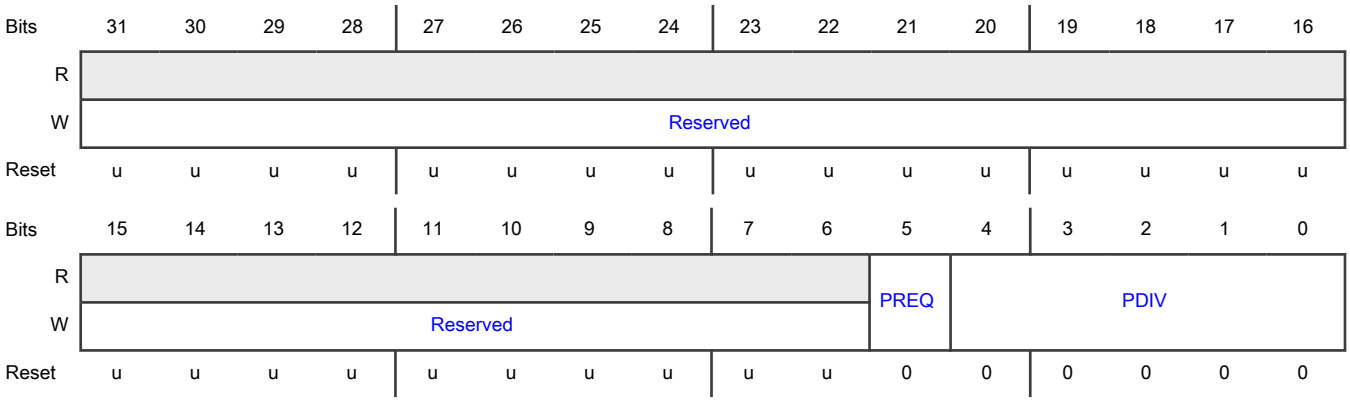
Field	Description
31-9 —	Reserved Read value is undefined, only zero should be written.
8 NREQ	pre-divider ratio change request.
7-0 NDIV	pre-divider divider ratio (N-divider).

8.5.1.1.102 PLL0 550m P divider (PLL0PDEC)

Offset

Register	Offset
PLL0PDEC	58Ch

Diagram



Fields

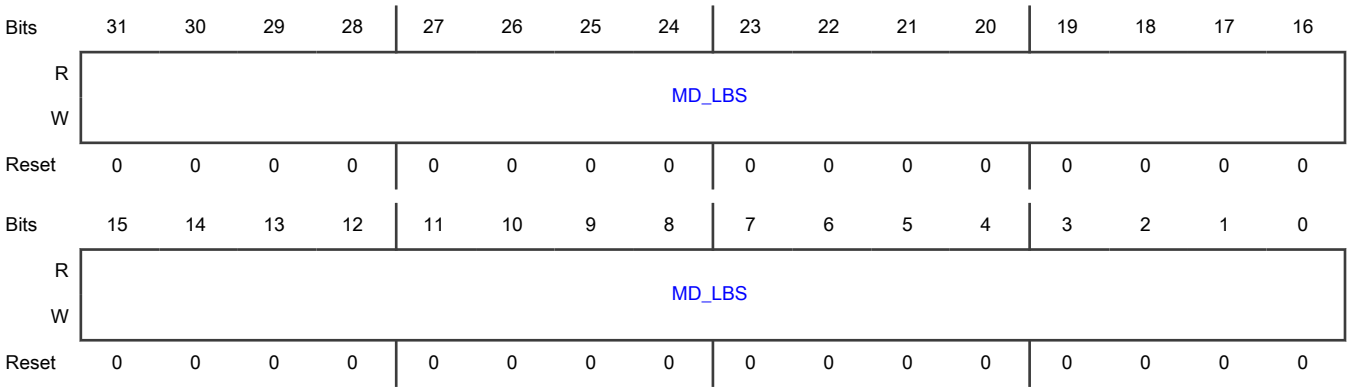
Field	Description
31-6 —	Reserved Read value is undefined, only zero should be written.
5 PREQ	feedback ratio change request.
4-0 PDIV	post-divider divider ratio (P-divider)

8.5.1.1.103 PLL0 Spread Spectrum control 0 (PLL0SSCG0)

Offset

Register	Offset
PLL0SSCG0	590h

Diagram



Fields

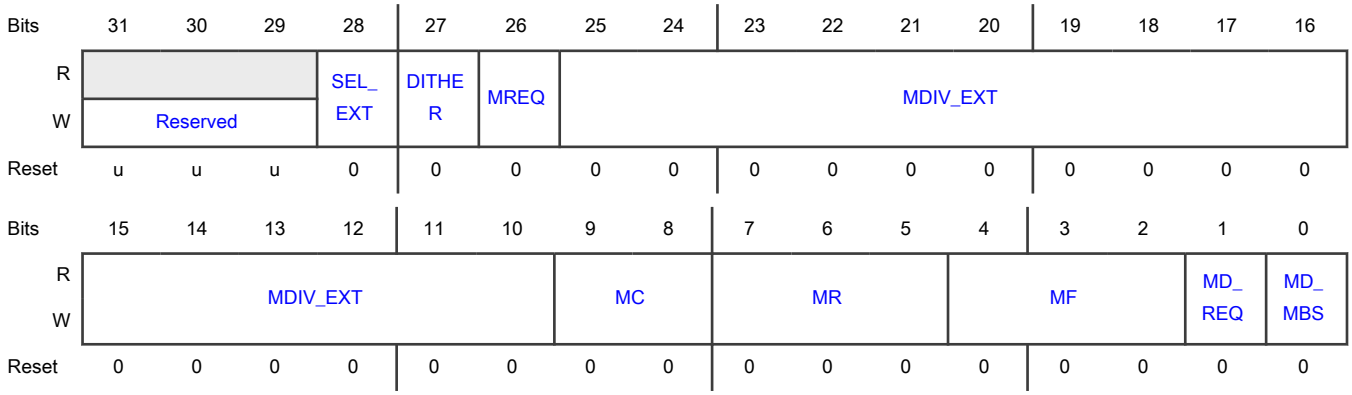
Field	Description
31-0 MD_LBS	input word of the wrapper bit 31 to 0.

8.5.1.1.104 PLL0 Spread Spectrum control 1 (PLL0SSCG1)

Offset

Register	Offset
PLL0SSCG1	594h

Diagram



Fields

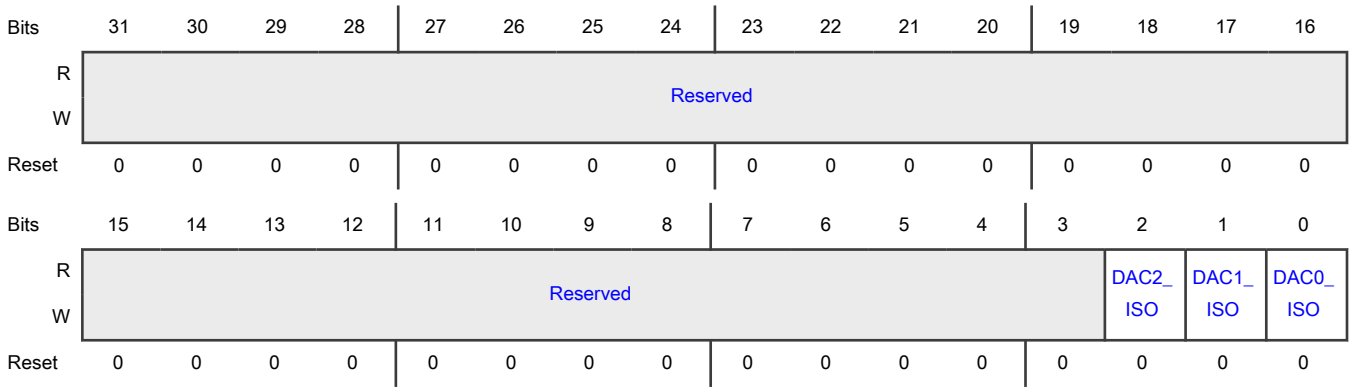
Field	Description
31-29 —	Reserved Read value is undefined, only zero should be written.
28 SEL_EXT	to select mdiv_ext and mreq_ext sel_ext = 0: mdiv ~ md[32:0], mreq = 1 sel_ext = 1 : mdiv = mdiv_ext, mreq = mreq_ext.
27 DITHER	dithering between two modulation frequencies in a random way or in a pseudo random way (white noise), in order to decrease the probability that the modulated waveform will occur with the same phase on a particular point on the screen.
26 MREQ	to select an external mreq value.
25-10 MDIV_EXT	to select an external mdiv value.
9-8 MC	modulation waveform control Compensation for low pass filtering of the PLL to get a triangular modulation at the output of the PLL, giving a flat frequency spectrum.
7-5 MR	programmable frequency modulation depth Dfmodpk-pk = Fref*kss/Fcco = kss/(2*md[32:25]dec) mr[2:0] = 000 => kss = 0 (no spread spectrum) mr[2:0] = 001 => kss ~ 1 mr[2:0] = 010 => kss ~ 1.
4-2 MF	programmable modulation frequency fm = Fref/Nss mf[2:0] = 000 => Nss=512 (fm ~ 3.
1 MD_REQ	md change request.
0 MD_MBS	input word of the wrapper bit 32.

8.5.1.1.105 DAC Isolation Control (DAC_ISO_CTRL)

Offset

Register	Offset
DAC_ISO_CTRL	5D0h

Diagram



Fields

Field	Description
31-3 —	Reserved
2 DAC2_ISO	DAC2 Isolation 0 - DAC2 isolation disabled 1 - DAC2 isolation enabled
1 DAC1_ISO	DAC1 Isolation 0 - DAC1 isolation disabled 1 - DAC1 isolation enabled
0 DAC0_ISO	DAC0 Isolation 0 - DAC0 isolation disabled 1 - DAC0 isolation enabled

8.5.1.1.106 Start logic wake-up enable (STARTER0)

Offset

Register	Offset
STARTER0	680h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	WAKEUP_...	Reserved	RTC_LIT...	USB0	USB0_NE...	Reserved	DMIC	ACMP_OVR	ADC1	ADC0	FLEXINT7	FLEXINT6	FLEXINT5	FLEXINT4	FLEXINT3	FLEXINT2
W																
Reset	0	0	0	0	0	u	u	0	u	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FLEXINT1	FLEXINT0	CTIMER3	SCT0	CTIMER1	CTIMER0	MRT0	UTICK0	PIO_INT3	PIO_INT2	PIO_INT1	PIO_INT0	GINT1	GINT0	SDMA0	SYS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31 WAKEUP_MAILBOX0	WAKEUP_MAILBOX0 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
30 —	Reserved
29 RTC_LITE0	RTC_LITE0 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
28 USB0	USB0-FS interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
27 USB0_NEEDCLK	USB0_NEEDCLK interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
26 —	Reserved Read value is undefined, only zero should be written.
25 DMIC	DMIC interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
24 ACMP_OVR	ACMP_OVR interrupt wake-up. 0 - Wake-up disabled.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Wake-up enabled.
23 ADC1	ADC1 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
22 ADC0	ADC0 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
21 FLEXINT7	FLEXINT7 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
20 FLEXINT6	FLEXINT6 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
19 FLEXINT5	FLEXINT5 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
18 FLEXINT4	FLEXINT4 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
17 FLEXINT3	FLEXINT3 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
16 FLEXINT2	FLEXINT2 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
15 FLEXINT1	FLEXINT1 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
14 FLEXINT0	FLEXINT0 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Description
13 CTIMER3	CTIMER3 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
12 SCT0	SCT0 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
11 CTIMER1	CTIMER1 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
10 CTIMER0	CTIMER0 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
9 MRT0	MRT0 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
8 UTICK0	UTICK0 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
7 PIO_INT3	PIO_INT3 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
6 PIO_INT2	PIO_INT2 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
5 PIO_INT1	PIO_INT1 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
4 PIO_INT0	PIO_INT0 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
3 GINT1	GINT1 interrupt wake-up. 0 - Wake-up disabled.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Wake-up enabled.
2 GINT0	GINT0 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
1 SDMA0	SDMA0 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
0 SYS	SYS interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.

8.5.1.1.107 Start logic wake-up enable (STARTER1)

Offset

Register	Offset
STARTER1	684h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved	I3C0	Reserved	Reserved	SPI_HS	SDMA1	PQ	Reserved	Reserved	Reserved		FREQ_ME...	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 W	Reserved	Reserved	Reserved	CAN0_IR...	CAN0_IR...	Reserved	Reserved		FLEXSPI	OS_EVENT	CTIMER4	CTIMER2	GPIO_IN...	GPIO_IN...	GPIO_IN...	GPIO_IN...
Reset	0	u	u	u	u	0	u	u	u	0	0	0	0	0	0	0

Fields

Field	Description
31 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
30 I3C0	I3C0 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
29 —	Reserved
28 —	Reserved
27 SPI_HS	SPI_HS interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
26 SDMA1	SDMA1 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
25 PQ	PQ interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
24 —	Reserved
23 —	Reserved
22-21 —	Reserved
20 FREQ_ME_PLUS	FREQME interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
19-17 —	Reserved
16-15 —	Reserved
14	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
13 —	Reserved
12 CAN0_IRQ1	CAN0_IRQ0 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
11 CAN0_IRQ0	CAN0_IRQ0 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
10 —	Reserved
9-8 —	Reserved Read value is undefined, only zero should be written.
7 FLEXSPI	FLEXSPI interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
6 OS_EVENT	OS_EVENT interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
5 CTIMER4	CTIMER4 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
4 CTIMER2	CTIMER2 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
3 GPIO_INT07	GPIO_INT07 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
2 GPIO_INT06	GPIO_INT06 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Description
1 GPIO_INT05	GPIO_INT05 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
0 GPIO_INT04	GPIO_INT04 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.

8.5.1.1.108 Start logic wake-up enable (STARTER2)

Offset

Register	Offset
STARTER2	688h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...
Reset	0	0	0	0	0	0	0	0	0	0	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	HS_CMP2...	HS_CMP1...	HS_CMP0...	DAC2_IRQ	DAC1_IRQ	DAC0_IRQ	Reserved						Reserved	Reserved	Reserved	Reserved
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31 FLEXPWM1_RELOAD0_IRQ	FlexPWM1 reload interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
30 FLEXPWM1_COMPARE0_IRQ	FlexPWM1 compare interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
29	FlexPWM1 reload error interrupt wake-up.

Table continues on the next page...

Table continued from the previous page...

Field	Description
FLEXPWM1_RELOAD_ERR_IRQ	0 - Wake-up disabled. 1 - Wake-up enabled.
28 FLEXPWM1_FAULT_IRQ	FlexPWM1 fault interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
27 FLEXPWM1_CAPTURE_IRQ	FlexPWM1 capture interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
26 FLEXPWM0_RELOAD3_IRQ	FlexPWM0 reload interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
25 FLEXPWM0_COMPARE3_IRQ	FlexPWM0 compare interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
24 FLEXPWM0_RELOAD2_IRQ	FlexPWM0 reload interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
23 FLEXPWM0_COMPARE2_IRQ	FlexPWM0 compare interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
22 FLEXPWM0_RELOAD1_IRQ	FlexPWM0 reload interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
21 FLEXPWM0_COMPARE1_IRQ	FlexPWM0 compare interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
20 FLEXPWM0_RELOAD0_IRQ	FlexPWM0 reload interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
19	FlexPWM0 compare interrupt wake-up. 0 - Wake-up disabled.

Table continues on the next page...

Table continued from the previous page...

Field	Description
FLEXPWM0_COMPARE0_IRQ	1 - Wake-up enabled.
18 FLEXPWM0_RELOAD_ERR_IRQ	FlexPWM0 reload error interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
17 FLEXPWM0_FAULT_IRQ	FlexPWM0 fault interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
16 FLEXPWM0_CAPTURE_IRQ	FlexPWM0 capture interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
15 HS_CMP2_IRQ	HS_CMP2 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
14 HS_CMP1_IRQ	HS_CMP1 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
13 HS_CMP0_IRQ	HS_CMP0 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
12 DAC2_IRQ	DAC2 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
11 DAC1_IRQ	DAC1 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
10 DAC0_IRQ	DAC0 interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
9-4 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
3 —	Reserved
2 —	Reserved
1 —	Reserved
0 —	Reserved

8.5.1.1.109 Start logic wake-up enable (STARTER3)

Offset

Register	Offset
STARTER3	68Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved										PVTVF1_...	PVTVF1_...	PVTVF0_...	PVTVF0_...	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 W	Reserved	ITRC_IRQ	ENC1_ID...	ENC1_WD...	ENC1_HO...	ENC1_CO...	ENCO_ID...	ENCO_WD...	ENCO_HO...	ENCO_CO...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-22 —	Reserved
21	PVTVF1 red interrupt wake-up. 0 - Wake-up disabled.

Table continues on the next page...

Table continued from the previous page...

Field	Description
PVTVF1_RED_IRQ	1 - Wake-up enabled.
20 PVTVF1_AMBER_IRQ	PVTVF1 amber interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
19 PVTVF0_RED_IRQ	PVTVF0 red interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
18 PVTVF0_AMBER_IRQ	PVTVF0 amber interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
17 —	Reserved
16 —	Reserved
15 —	Reserved
14 ITRC_IRQ	ITRC interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
13 ENC1_IDX_IRQ	ENC1 IDX interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
12 ENC1_WDG_IRQ	ENC1 WDOG interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
11 ENC1_HOME_IRQ	ENC1 home interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
10	ENC1 compare interrupt wake-up. 0 - Wake-up disabled.

Table continues on the next page...

Table continued from the previous page...

Field	Description
ENC1_COMPARE_IRQ	1 - Wake-up enabled.
9 ENCO_IDX_IRQ	ENC0 IDX interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
8 ENCO_WDG_IRQ	ENC0 WDOG interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
7 ENCO_HOME_IRQ	ENC0 home interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
6 ENCO_COMPARE_IRQ	ENC0 compare interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
5 FLEXPWM1_RELOAD3_IRQ	FlexPWM1 reload interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
4 FLEXPWM1_COMPARE3_IRQ	FlexPWM1 compare interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
3 FLEXPWM1_RELOAD2_IRQ	FlexPWM1 reload interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
2 FLEXPWM1_COMPARE2_IRQ	FlexPWM1 compare interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
1 FLEXPWM1_RELOAD1_IRQ	FlexPWM1 reload interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.
0 FLEXPWM1_COMPARE1_IRQ	FlexPWM1 compare interrupt wake-up. 0 - Wake-up disabled. 1 - Wake-up enabled.

8.5.1.1.110 Set bits in STARTER (STARTERSET0)

Offset

Register	Offset
STARTERSET0	6A0h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		Reserved														
W	WAKEUP_...		RTC_LIT...	USB0_SET	USB0_NE...	Reserved		ADC0_TH...	Reserved	ADC0_SET	FLEXINT...	FLEXINT...	FLEXINT...	FLEXINT...	FLEXINT...	FLEXINT...
Reset	0	0	0	0	0	u	u	0	u	0	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	FLEXINT...	FLEXINT...	CTIMER3...	SCT0_SET	CTIMER1...	CTIMER0...	MRT0_SET	UTICK0_...	GPIO_IN...	GPIO_IN...	GPIO_IN...	GPIO_IN...	GPIO_GL...	GPIO_GL...	SDMA0_S...	SYS_SET
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31 WAKEUP_MAILBOX0_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
30 —	Reserved
29 RTC_LITE0_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
28 USB0_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
27 USB0_NEEDCLK_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
26-25 —	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
24 ADC0_THCMP_OVR_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
23 —	Reserved Read value is undefined, only zero should be written.
22 ADC0_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
21 FLEXINT7_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
20 FLEXINT6_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
19 FLEXINT5_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
18 FLEXINT4_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
17 FLEXINT3_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
16 FLEXINT2_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
15 FLEXINT1_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
14 FLEXINT0_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
13 CTIMER3_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
12 SCT0_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
11 CTIMER1_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.

Table continues on the next page...

Table continued from the previous page...

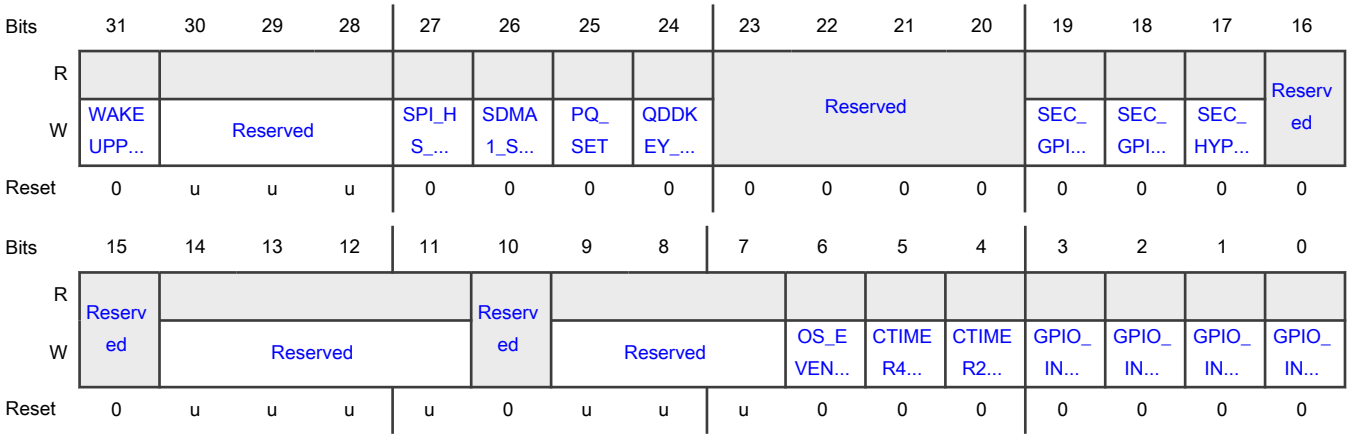
Field	Description
10 CTIMER0_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
9 MRT0_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
8 UTICK0_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
7 GPIO_INT03_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
6 GPIO_INT02_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
5 GPIO_INT01_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
4 GPIO_INT00_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
3 GPIO_GLOBALINT1_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
2 GPIO_GLOBALINT0_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
1 SDMA0_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.
0 SYS_SET	Writing ones to this register sets the corresponding bit in the STARTER0 register.

8.5.1.1.111 Set bits in STARTER (STARTERSET1)

Offset

Register	Offset
STARTERSET1	6A4h

Diagram



Fields

Field	Description
31 WAKEUPPADS_SET	Writing ones to this register sets the corresponding bit in the STARTER1 register.
30-28 —	Reserved Read value is undefined, only zero should be written.
27 SPI_HS_SET	Writing ones to this register sets the corresponding bit in the STARTER1 register.
26 SDMA1_SET	Writing ones to this register sets the corresponding bit in the STARTER1 register.
25 PQ_SET	Writing ones to this register sets the corresponding bit in the STARTER1 register.
24 QDDKEY_SET	Writing ones to this register sets the corresponding bit in the STARTER1 register.
23-20 —	Reserved
19	Writing ones to this register sets the corresponding bit in the STARTER1 register.

Table continues on the next page...

Table continued from the previous page...

Field	Description
SEC_GPIO_INT01_SET	
18 SEC_GPIO_INT00_SET	Writing ones to this register sets the corresponding bit in the STARTER1 register.
17 SEC_HYPERVISOR_CALL_SET	Writing ones to this register sets the corresponding bit in the STARTER1 register.
16-15 —	Reserved
14-11 —	Reserved Read value is undefined, only zero should be written.
10 —	Reserved
9-7 —	Reserved Read value is undefined, only zero should be written.
6 OS_EVENT_SET	Writing ones to this register sets the corresponding bit in the STARTER1 register.
5 CTIMER4_SET	Writing ones to this register sets the corresponding bit in the STARTER1 register.
4 CTIMER2_SET	Writing ones to this register sets the corresponding bit in the STARTER1 register.
3 GPIO_INT07_SET	Writing ones to this register sets the corresponding bit in the STARTER1 register.
2 GPIO_INT06_SET	Writing ones to this register sets the corresponding bit in the STARTER1 register.
1	Writing ones to this register sets the corresponding bit in the STARTER1 register.

Table continues on the next page...

Table continued from the previous page...

Field	Description
GPIO_INT05_SET	
0 GPIO_INT04_SET	Writing ones to this register sets the corresponding bit in the STARTER1 register.

8.5.1.1.112 Set bits in STARTER (STARTERSET2)

Offset

Register	Offset
STARTERSET2	6A8h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...
Reset	0	0	0	0	0	0	0	0	0	0	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	HS_CMP2...	HS_CMP1...	HS_CMP0...	DAC2_SET	DAC1_SET	DAC0_SET	Reserved						Reserved	TAMPER_...	ELS_SET	Reserved
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31 FLEXPWM1_RELOAD0_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
30 FLEXPWM1_COMPARE0_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
29	Writing ones to this register sets the corresponding bit in the STARTER2 register.

Table continues on the next page...

Table continued from the previous page...

Field	Description
FLEXPWM1_RELOAD_ERROR_SET	
28 FLEXPWM1_FAULT_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
27 FLEXPWM1_CAPTURE_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
26 FLEXPWM0_RELOAD3_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
25 FLEXPWM0_COMPARE3_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
24 FLEXPWM0_RELOAD2_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
23 FLEXPWM0_COMPARE2_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
22 FLEXPWM0_RELOAD1_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
21 FLEXPWM0_COMPARE1_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
20 FLEXPWM0_RELOAD0_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
19	Writing ones to this register sets the corresponding bit in the STARTER2 register.

Table continues on the next page...

Table continued from the previous page...

Field	Description
FLEXPWM0_COMPARE0_SET	
18 FLEXPWM0_RELOAD_ERROR_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
17 FLEXPWM0_FAULT_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
16 FLEXPWM0_CAPTURE_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
15 HS_CMP2_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
14 HS_CMP1_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
13 HS_CMP0_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
12 DAC2_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
11 DAC1_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
10 DAC0_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
9-4 —	Reserved
3 —	Reserved
2 TAMPER_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.

Table continues on the next page...

Table continued from the previous page...

Field	Description
1 ELS_SET	Writing ones to this register sets the corresponding bit in the STARTER2 register.
0 —	Reserved

8.5.1.1.113 Set bits in STARTER (STARTERSET3)

Offset

Register	Offset
STARTERSET3	6ACh

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved										PVTVF1_...	PVTVF1_...	PVTVF0_...	PVTVF0_...	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 W	Reserved	ITRC_SET	ENC1_ID...	ENC1_WD...	ENC1_HO...	ENC1_CO...	ENC0_ID...	ENC0_WD...	ENC0_HO...	ENC0_CO...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...	FLEXPWM...
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-22 —	Reserved
21 PVTVF1_RED_SET	Writing ones to this register sets the corresponding bit in the STARTER3 register.
20 PVTVF1_AMBER_SET	Writing ones to this register sets the corresponding bit in the STARTER3 register.

Table continues on the next page...

Table continued from the previous page...

Field	Description
19 PVTVF0_RED_SET	Writing ones to this register sets the corresponding bit in the STARTER3 register.
18 PVTVF0_AMBER_SET	Writing ones to this register sets the corresponding bit in the STARTER3 register.
17 —	Reserved
16 —	Reserved
15 —	Reserved
14 ITRC_SET	Writing ones to this register sets the corresponding bit in the STARTER3 register.
13 ENC1_IDX_SET	Writing ones to this register sets the corresponding bit in the STARTER3 register.
12 ENC1_WDG_SET	Writing ones to this register sets the corresponding bit in the STARTER3 register.
11 ENC1_HOME_SET	Writing ones to this register sets the corresponding bit in the STARTER3 register.
10 ENC1_COMPARE_SET	Writing ones to this register sets the corresponding bit in the STARTER3 register.
9 ENC0_IDX_SET	Writing ones to this register sets the corresponding bit in the STARTER3 register.
8 ENC0_WDG_SET	Writing ones to this register sets the corresponding bit in the STARTER3 register.

Table continues on the next page...

Table continued from the previous page...

Field	Description
7 ENC0_HOME_SET	Writing ones to this register sets the corresponding bit in the STARTER3 register.
6 ENC0_COMPARE_SET	Writing ones to this register sets the corresponding bit in the STARTER3 register.
5 FLEXPWM1_RELOAD3_SET	Writing ones to this register sets the corresponding bit in the STARTER3 register.
4 FLEXPWM1_COMPARE3_SET	Writing ones to this register sets the corresponding bit in the STARTER3 register.
3 FLEXPWM1_RELOAD2_SET	Writing ones to this register sets the corresponding bit in the STARTER3 register.
2 FLEXPWM1_COMPARE2_SET	Writing ones to this register sets the corresponding bit in the STARTER3 register.
1 FLEXPWM1_RELOAD1_SET	Writing ones to this register sets the corresponding bit in the STARTER3 register.
0 FLEXPWM1_COMPARE1_SET	Writing ones to this register sets the corresponding bit in the STARTER3 register.

8.5.1.1.114 Clear bits in STARTER (STARTERCLR0)

Offset

Register	Offset
STARTERCLR0	6C0h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R		Reserved														
W	WAKEUP_...		RTC_LIT...	USB0_CLR	USB0_NE...	Reserved		ADC0_TH...	Reserved	ADC0_CLR	FLEXINT...	FLEXINT...	FLEXINT...	FLEXINT...	FLEXINT...	FLEXINT...
Reset	0	0	0	0	0	u	u	0	u	0	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	FLEXINT...	FLEXINT...	CTIMER3...	SCT0_CLR	CTIMER1...	CTIMER0...	MRT0_CLR	UTICK0_...	GPIO_IN...	GPIO_IN...	GPIO_IN...	GPIO_IN...	GPIO_GL...	GPIO_GL...	SDMA0_C...	SYS_CLR
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31 WAKEUP_MAILBOX0_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
30 —	Reserved
29 RTC_LITE0_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
28 USB0_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
27 USB0_NEEDCLK_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
26-25 —	Reserved Read value is undefined, only zero should be written.
24 ADC0_THCMP_OVR_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
23 —	Reserved Read value is undefined, only zero should be written.
22	Writing ones to this register clears the corresponding bit in the STARTER0 register.

Table continues on the next page...

Table continued from the previous page...

Field	Description
ADC0_CLR	
21 FLEXINT7_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
20 FLEXINT6_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
19 FLEXINT5_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
18 FLEXINT4_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
17 FLEXINT3_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
16 FLEXINT2_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
15 FLEXINT1_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
14 FLEXINT0_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
13 CTIMER3_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
12 SCT0_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
11 CTIMER1_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
10 CTIMER0_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
9 MRT0_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
8 UTICK0_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.

Table continues on the next page...

Table continued from the previous page...

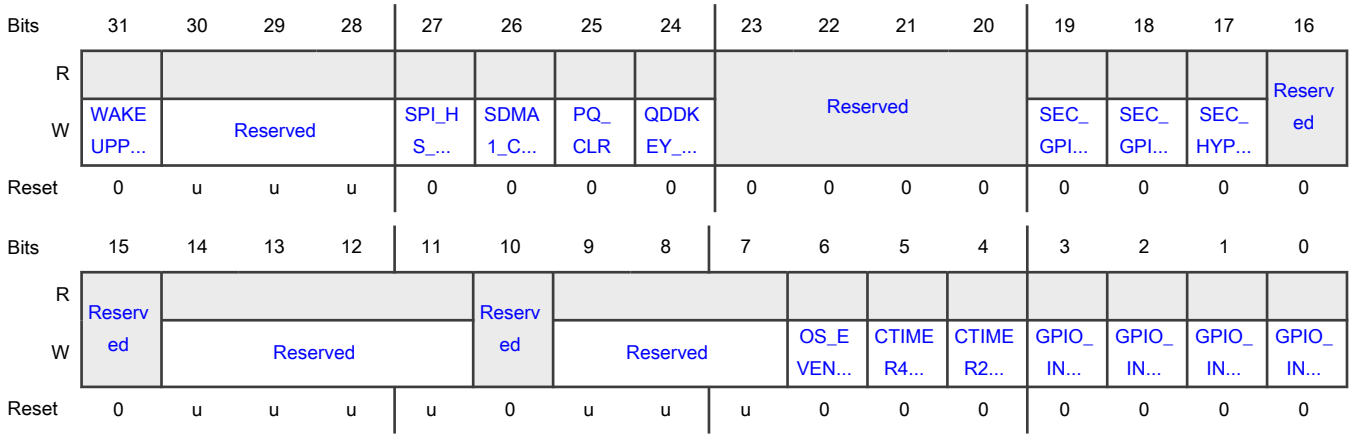
Field	Description
7 GPIO_INT03_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
6 GPIO_INT02_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
5 GPIO_INT01_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
4 GPIO_INT00_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
3 GPIO_GLOBALINT1_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
2 GPIO_GLOBALINT0_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
1 SDMA0_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.
0 SYS_CLR	Writing ones to this register clears the corresponding bit in the STARTER0 register.

8.5.1.1.115 Clear bits in STARTER (STARTERCLR1)

Offset

Register	Offset
STARTERCLR1	6C4h

Diagram



Fields

Field	Description
31 WAKEUPPADS_CLR	Writing ones to this register clears the corresponding bit in the STARTER1 register.
30-28 —	Reserved Read value is undefined, only zero should be written.
27 SPI_HS_CLR	Writing ones to this register clears the corresponding bit in the STARTER1 register.
26 SDMA1_CLR	Writing ones to this register clears the corresponding bit in the STARTER1 register.
25 PQ_CLR	Writing ones to this register clears the corresponding bit in the STARTER1 register.
24 QDDKEY_CLR	Writing ones to this register clears the corresponding bit in the STARTER1 register.
23-20 —	Reserved
19 SEC_GPIO_INT01_CLR	Writing ones to this register clears the corresponding bit in the STARTER1 register.
18 SEC_GPIO_INT00_CLR	Writing ones to this register clears the corresponding bit in the STARTER1 register.

Table continues on the next page...

Table continued from the previous page...

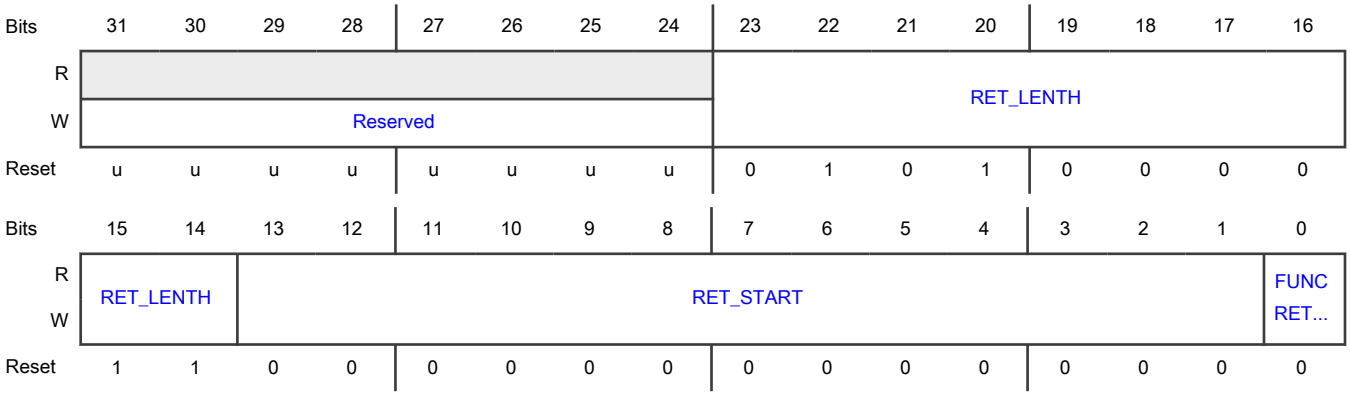
Field	Description
17 SEC_HYPERVISOR_CALL_CLR	Writing ones to this register clears the corresponding bit in the STARTER1 register.
16-15 —	Reserved
14-11 —	Reserved Read value is undefined, only zero should be written.
10 —	Reserved
9-7 —	Reserved Read value is undefined, only zero should be written.
6 OS_EVENT_CLR	Writing ones to this register clears the corresponding bit in the STARTER1 register.
5 CTIMER4_CLR	Writing ones to this register clears the corresponding bit in the STARTER1 register.
4 CTIMER2_CLR	Writing ones to this register clears the corresponding bit in the STARTER1 register.
3 GPIO_INT07_CLR	Writing ones to this register clears the corresponding bit in the STARTER1 register.
2 GPIO_INT06_CLR	Writing ones to this register clears the corresponding bit in the STARTER1 register.
1 GPIO_INT05_CLR	Writing ones to this register clears the corresponding bit in the STARTER1 register.
0 GPIO_INT04_CLR	Writing ones to this register clears the corresponding bit in the STARTER1 register.

8.5.1.1.116 Functional retention control (FUNCRETENTIONCTRL)

Offset

Register	Offset
FUNCRETENTIONCTRL	704h

Diagram



Fields

Field	Description
31-24 —	Reserved Read value is undefined, only zero should be written.
23-14 RET_LENTH	lenth of Scan chains to save.
13-1 RET_START	Start address divided by 4 inside SRAMX bank.
0 FUNCRETENA	functional retention in power down only. 0 - disable functional retention. 1 - enable functional retention.

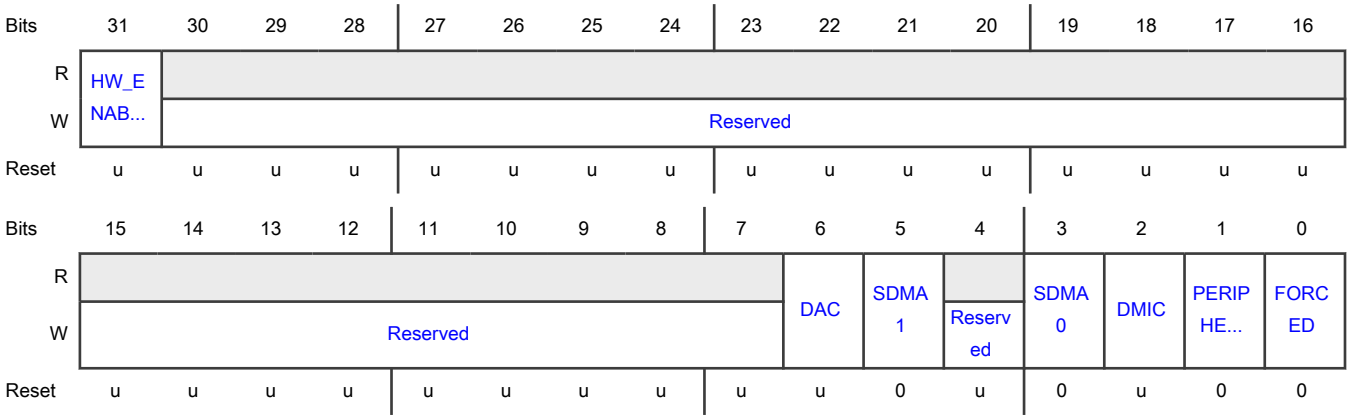
8.5.1.1.117 Hardware Sleep control (HARDWARESLEEP)

used to postpone power down modes in case an interrupt is pending when the processor request deepsleep

Offset

Register	Offset
HARDWARESLEEP	780h

Diagram



Fields

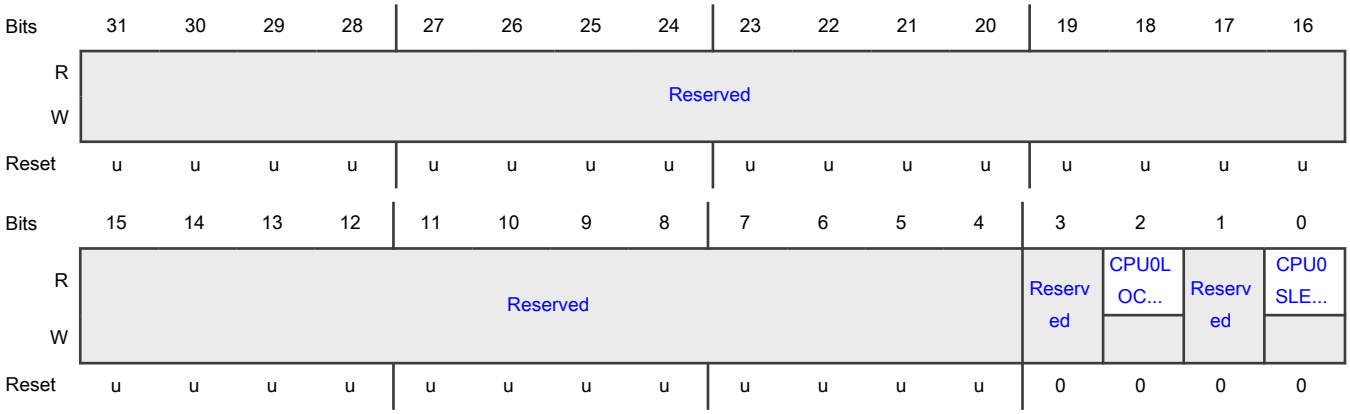
Field	Description
31 HW_ENABLE_FRO192M	Set this bit if FRO192M is diabled.
30-7 —	Reserved Read value is undefined, only zero should be written.
6 DAC	Wake for DAC0/1/2.
5 SDMA1	Wake for DMA1.
4 —	Reserved Read value is undefined, only zero should be written.
3 SDMA0	Wake for DMA0.
2 DMIC	Wake for DMIC.
1 PERIPHERALS	Wake for Flexcomms.
0 FORCED	Force peripheral clocking to stay on during Deep Sleep and Power-down modes.

8.5.1.1.118 CPU Status (CPUSTAT)

Offset

Register	Offset
CPUSTAT	80Ch

Diagram



Fields

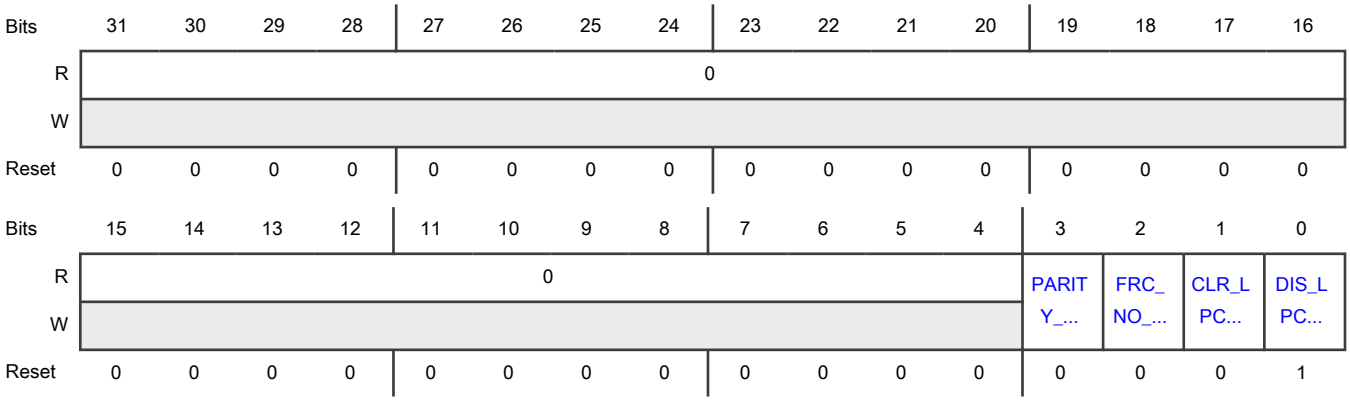
Field	Description
31-4 —	Reserved
3 —	Reserved
2 CPU0LOCKUP	The CPU0 lockup state. 0 - the CPU is not in lockup. 1 - the CPU is in lockup.
1 —	Reserved
0 CPU0SLEEPING	The CPU0 sleeping state. 0 - the CPU is not sleeping. 1 - the CPU is sleeping.

8.5.1.1.119 LPCAC control (LPCAC_CTRL)

Offset

Register	Offset
LPCAC_CTRL	824h

Diagram



Fields

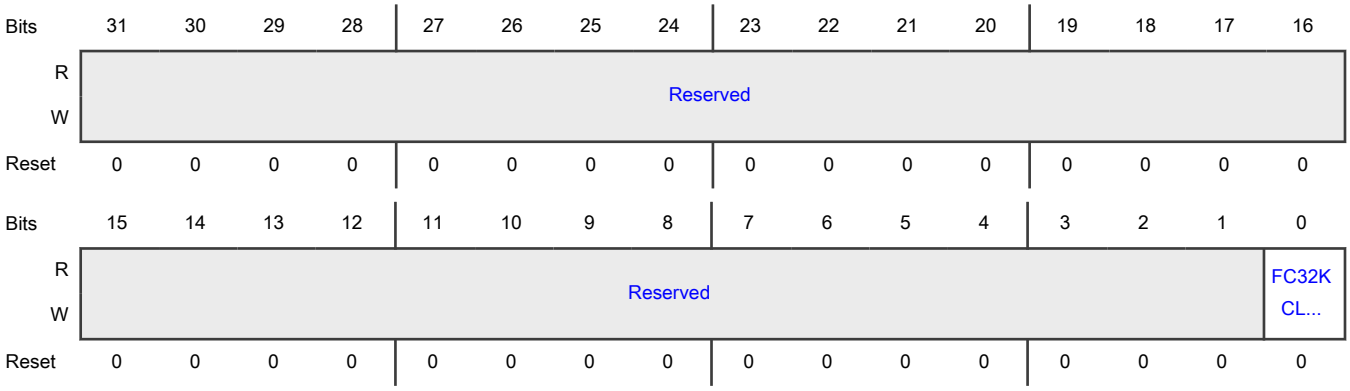
Field	Description
31-4 —	Reserved
3 PARITY_MISS_EN	Enable parity miss. Default value is 0. 0 - Disable 1 - Enable parity, miss on parity error
2 FRC_NO_ALLOC	Force no allocation. Default value is 0. 0 - Force allocation 1 - Force no allocation
1 CLR_LPCAC	Clear cache function. Default value is 0. 0 - Unclear cache 1 - Clear cache
0 DIS_LPCAC	Disable/enable cache function. Default value is 1. 0 - Enable 1 - Disable

8.5.1.1.120 Flexcomm 32K clock select (FC32KCLKSEL)

Offset

Register	Offset
FC32KCLKSEL	82Ch

Diagram



Fields

Field	Description
31-1 —	Reserved
0 FC32KCLKSEL	Flexcomm 32K clock select 0 - FRO32K 1 - XTAL 32K

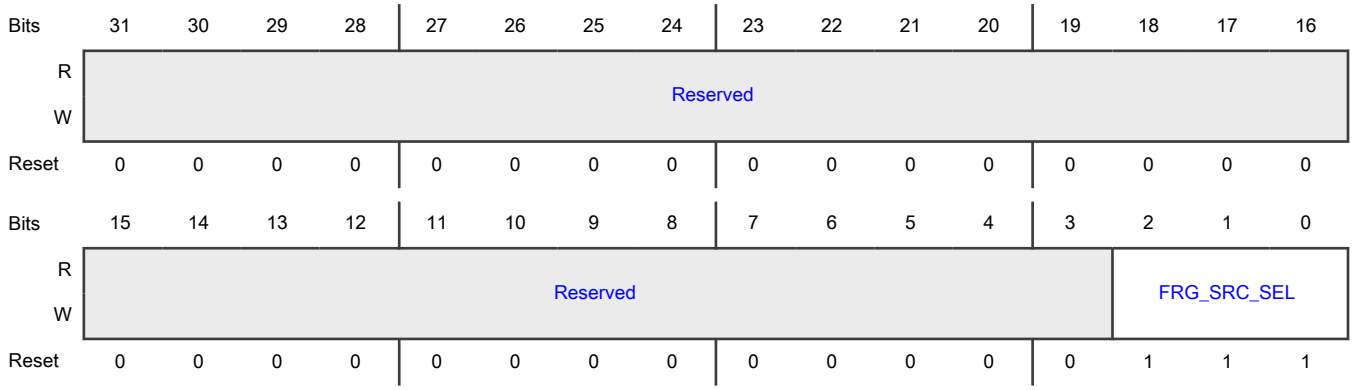
8.5.1.1.121 FRG Clock Source Select (FRGCLKSEL0 - FRGCLKSEL7)

Offset

For n = 0 to 7:

Register	Offset
FRGCLKSELn	830h + (n × 4h)

Diagram



Fields

Field	Description
31-3 —	Reserved
2-0 FRG_SRC_SEL	FRG clock source select 000 - main clock 001 - PLL clock 010 - fro_div_hf 011 - None 100 - None 101 - None 110 - None 111 - None

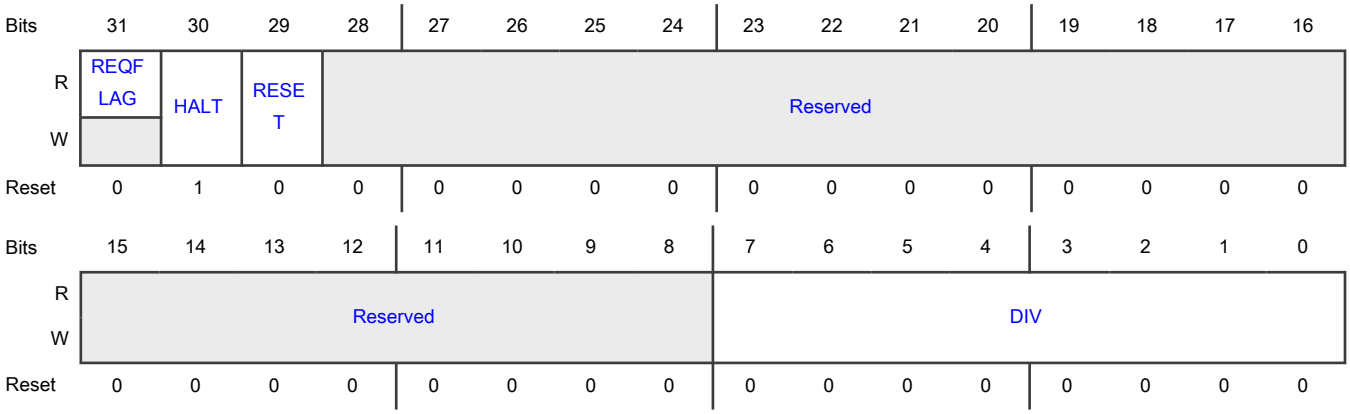
8.5.1.1.122 Flexcomm clock divider (FLEXCOMM0CLKDIV - FLEXCOMM7CLKDIV)

Offset

For n = 0 to 7:

Register	Offset
FLEXCOMMnCLKDIV	850h + (n × 4h)

Diagram



Fields

Field	Description
31 REQFLAG	Reset 0 - Divider clock is stable 1 - Clock frequency is not stable
30 HALT	Reset 0 - Divider clock is running 1 - Divider clock has stopped
29 RESET	Reset 0 - Divider is not reset 1 - Divider is reset
28-8 —	Reserved
7-0 DIV	Clock divider value: • 0 - Divide by 1 • 1 - Divide by 2 • ... • 255 - Divide by 256

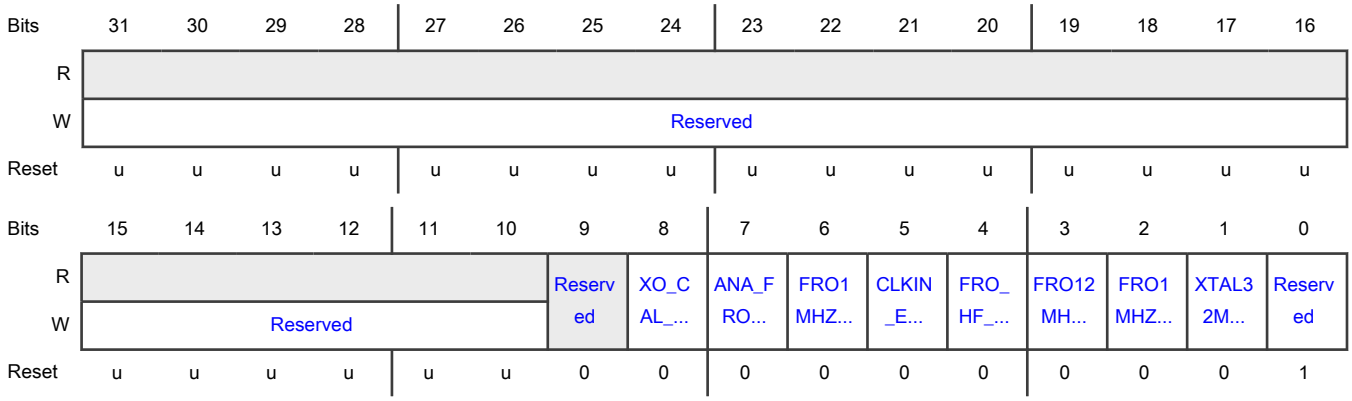
8.5.1.1.123 Clock Control (CLOCK_CTRL)

Various system clock controls : Flash clock (48 MHz) control, clocks to Frequency Measures

Offset

Register	Offset
CLOCK_CTRL	A18h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 —	Reserved
8 XO_CAL_CLK_ENA	Enable clock for crystal oscillator calibration 0 - The clock is not enabled. 1 - The clock is enabled.
7 ANA_FRO12M_CLK_ENA	Enable FRO 12MHz clock for analog control of the FRO 192MHz. 0 - The clock is not enabled. 1 - The clock is enabled.
6 FRO1MHZ_CLK_ENA	Enable FRO 1MHz clock for clock muxing in clock gen. 0 - The clock is not enabled. 1 - The clock is enabled.
5 CLKIN_ENA	Enable clock_in clock for clock module. 0 - The clock is not enabled. 1 - The clock is enabled.
4 FRO_HF_FREQM_ENA	Enable FRO 96MHz clock for Frequency Measure module. 0 - The clock is not enabled. 1 - The clock is enabled.
3 FRO12MHZ_FREQM_ENA	Enable FRO 12MHz clock for Frequency Measure module. 0 - The clock is not enabled. 1 - The clock is enabled.

Table continues on the next page...

Table continued from the previous page...

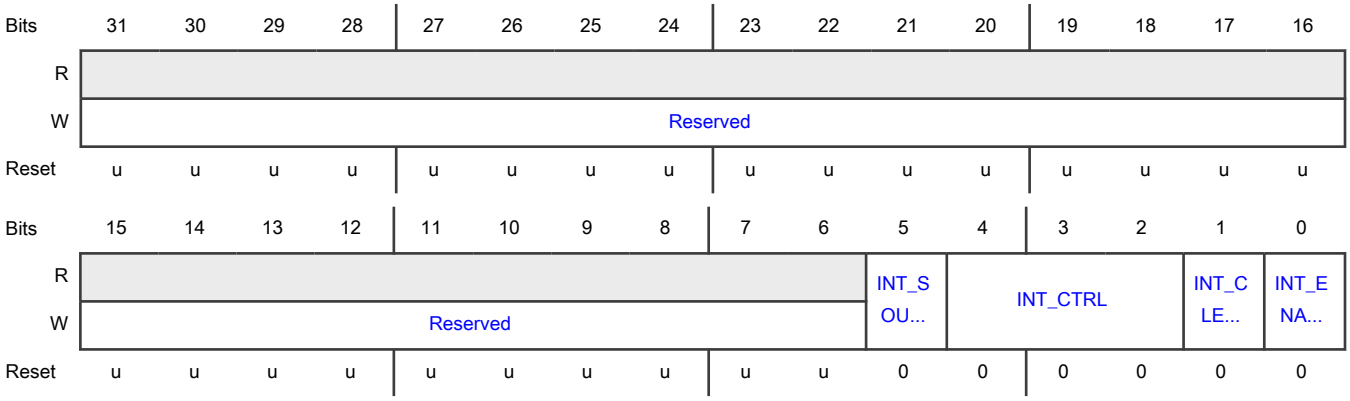
Field	Description
2 FRO1MHZ_UTICK_ENA	Enable FRO 1MHz clock for Frequency Measure module and for UTICK. 0 - The clock is not enabled. 1 - The clock is enabled.
1 XTAL32MHZ_FREQM_ENA	Enable XTAL32MHz clock for Frequency Measure module. 0 - The clock is not enabled. 1 - The clock is enabled.
0 —	Reserved need to write '1'.

8.5.1.1.124 Comparator Interrupt control (COMP_INT_CTRL)

Offset

Register	Offset
COMP_INT_CTRL	B10h

Diagram



Fields

Field	Description
31-6 —	Reserved Read value is undefined, only zero should be written.
5 INT_SOURCE	Select which Analog comparator output (filtered our un-filtered) is used for interrupt detection. 0 - Select Analog Comparator filtered output as input for interrupt detection.

Table continues on the next page...

Table continued from the previous page...

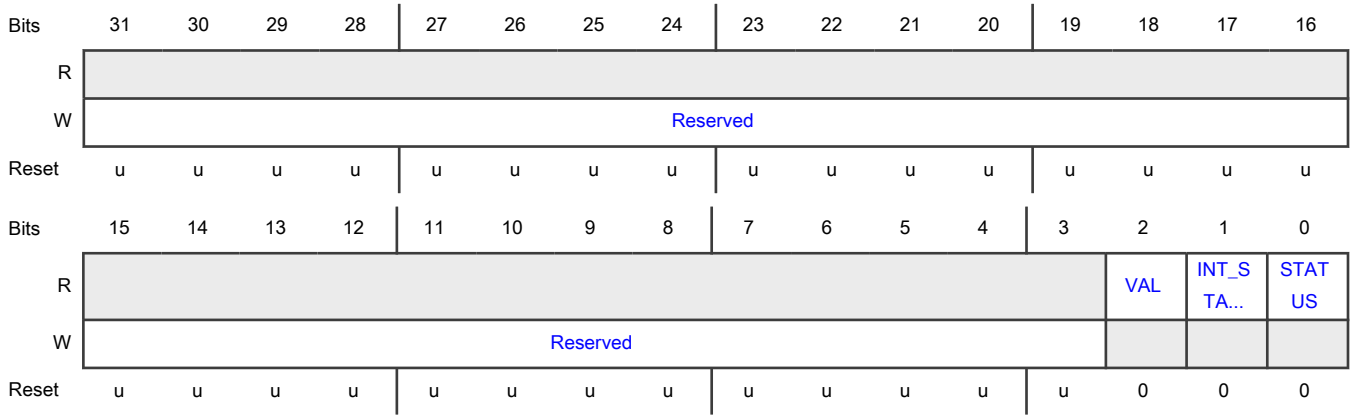
Field	Description
	1 - Select Analog Comparator raw output (unfiltered) as input for interrupt detection. Must be used when Analog comparator is used as wake up source in Power down mode.
4-2 INT_CTRL	Comparator interrupt type selector:. 000 - The analog comparator interrupt edge sensitive is disabled. 001 - The analog comparator interrupt level sensitive is disabled. 010 - analog comparator interrupt is rising edge sensitive. 011 - Analog Comparator interrupt is high level sensitive. 100 - analog comparator interrupt is falling edge sensitive. 101 - Analog Comparator interrupt is low level sensitive. 110 - analog comparator interrupt is rising and falling edge sensitive. 111 - The analog comparator interrupt level sensitive is disabled.
1 INT_CLEAR	Analog Comparator interrupt clear. 0 - No effect. 1 - Clear the interrupt. Self-cleared bit.
0 INT_ENABLE	Analog Comparator interrupt enable control:. 0 - interrupt disable. 1 - interrupt enable.

8.5.1.1.125 Comparator Interrupt status (COMP_INT_STATUS)

Offset

Register	Offset
COMP_INT_STATUS	B14h

Diagram



Fields

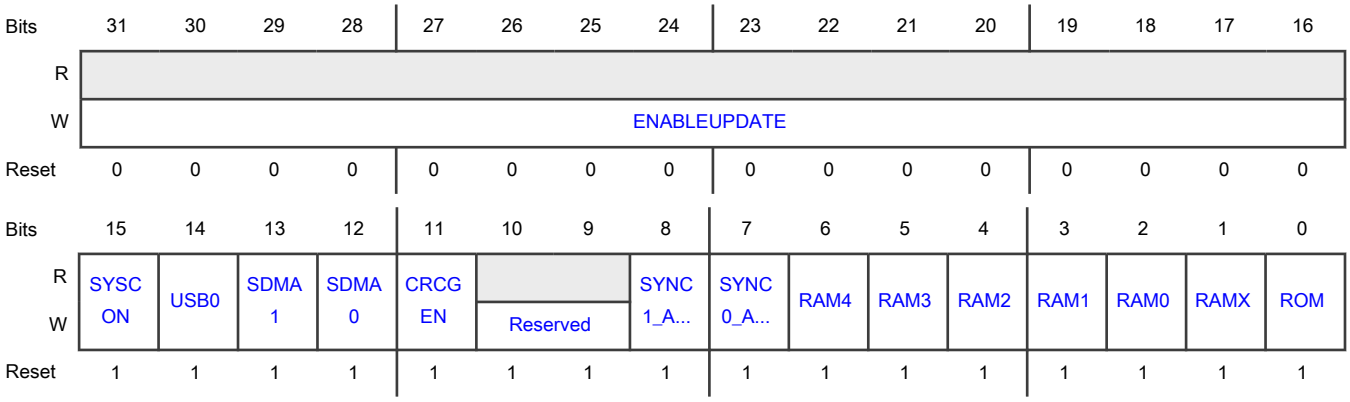
Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2 VAL	comparator analog output. 0 - P+ is smaller than P-. 1 - P+ is greater than P-.
1 INT_STATUS	Interrupt status AFTER Interrupt Enable. 0 - no interrupt pending. 1 - interrupt pending.
0 STATUS	Interrupt status BEFORE Interrupt Enable. 0 - No interrupt pending. 1 - Interrupt pending.

8.5.1.1.126 Control automatic clock gating (AUTOCLKGATEOVERRIDE)

Offset

Register	Offset
AUTOCLKGATEOVERRIDE	E04h

Diagram



Fields

Field	Description
31-16 ENABLEUPDATE	The value 0xC0DE must be written for AUTOCLKGATEOVERRIDE registers fields updates to have effect. 0000_0000_0000_0000 - Bit Fields 0 - 15 of this register are not updated 1100_0000_1101_1110 - Bit Fields 0 - 15 of this register are updated
15 SYSCON	Control automatic clock gating of synchronous system controller registers bank. 0 - Automatic clock gating is not overridden. 1 - Automatic clock gating is overridden (Clock gating is disabled).
14 USB0	Control automatic clock gating of USB controller. 0 - Automatic clock gating is not overridden. 1 - Automatic clock gating is overridden (Clock gating is disabled).
13 SDMA1	Control automatic clock gating of DMA1 controller. 0 - Automatic clock gating is not overridden. 1 - Automatic clock gating is overridden (Clock gating is disabled).
12 SDMA0	Control automatic clock gating of DMA0 controller. 0 - Automatic clock gating is not overridden. 1 - Automatic clock gating is overridden (Clock gating is disabled).
11 CRCGEN	Control automatic clock gating of CRCGEN controller. 0 - Automatic clock gating is not overridden. 1 - Automatic clock gating is overridden (Clock gating is disabled).
10-9 —	Reserved Read value is undefined, only one should be written.
8 SYNC1_APB	Control automatic clock gating of synchronous bridge controller 1. 0 - Automatic clock gating is not overridden. 1 - Automatic clock gating is overridden (Clock gating is disabled).
7 SYNC0_APB	Control automatic clock gating of synchronous bridge controller 0. 0 - Automatic clock gating is not overridden. 1 - Automatic clock gating is overridden (Clock gating is disabled).
6 RAM4	Control automatic clock gating of RAM4 controller. 0 - Automatic clock gating is not overridden. 1 - Automatic clock gating is overridden (Clock gating is disabled).
5 RAM3	Control automatic clock gating of RAM3 controller. 0 - Automatic clock gating is not overridden.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Automatic clock gating is overridden (Clock gating is disabled).
4 RAM2	Control automatic clock gating of RAM2 controller. 0 - Automatic clock gating is not overridden. 1 - Automatic clock gating is overridden (Clock gating is disabled).
3 RAM1	Control automatic clock gating of RAM1 controller. 0 - Automatic clock gating is not overridden. 1 - Automatic clock gating is overridden (Clock gating is disabled).
2 RAM0	Control automatic clock gating of RAM0 controller. 0 - Automatic clock gating is not overridden. 1 - Automatic clock gating is overridden (Clock gating is disabled).
1 RAMX	Control automatic clock gating of RAMX controller. 0 - Automatic clock gating is not overridden. 1 - Automatic clock gating is overridden (Clock gating is disabled).
0 ROM	Control automatic clock gating of ROM controller. 0 - Automatic clock gating is not overridden. 1 - Automatic clock gating is overridden (Clock gating is disabled).

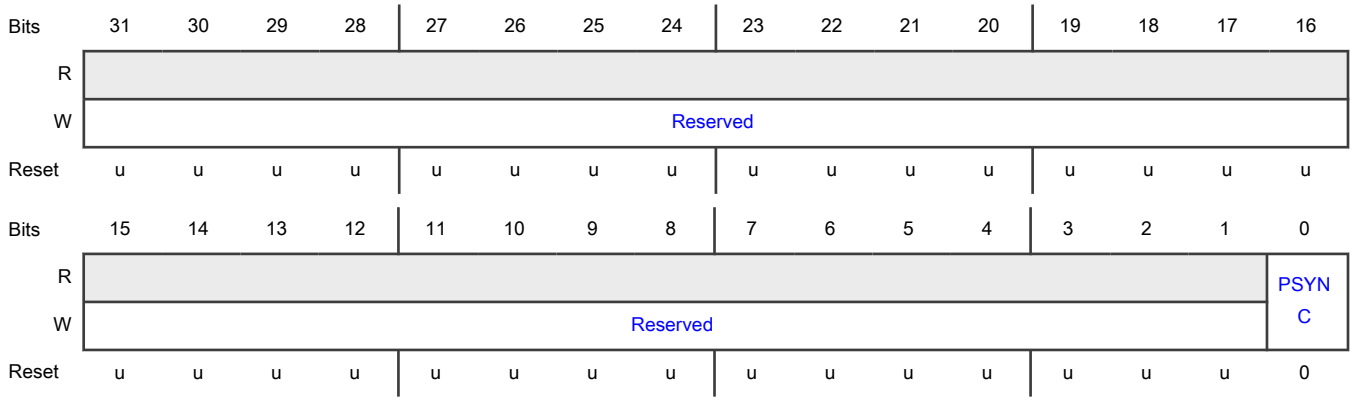
8.5.1.1.127 GPIO Synchronization (GPIOPSYNC)

Enable bypass of the first stage of synchronization inside GPIO_INT module

Offset

Register	Offset
GPIOPSYNC	E08h

Diagram



Fields

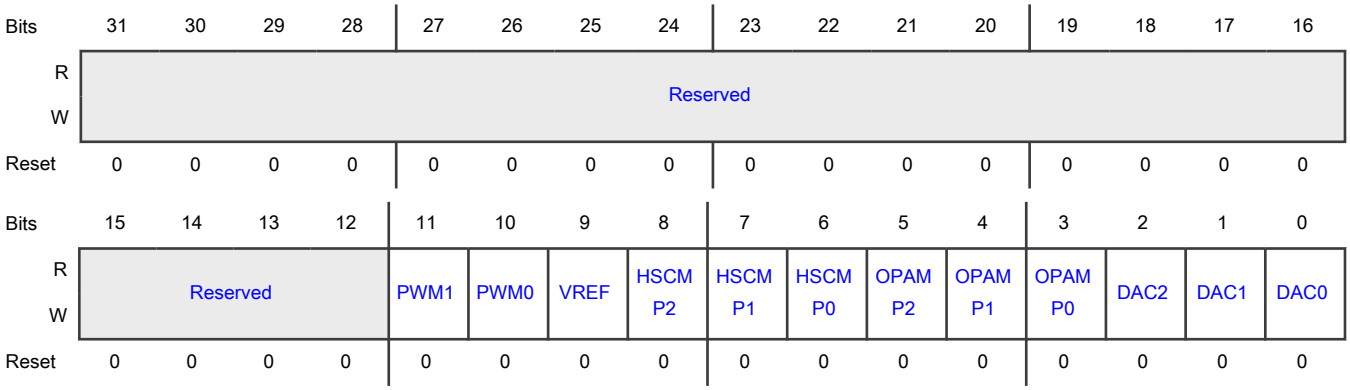
Field	Description
31-1 —	Reserved Read value is undefined, only zero should be written.
0 PSYNC	Enable bypass of the first stage of synchronization inside GPIO_INT module. 0 - Use the first stage of synchronization inside GPIO_INT module. 1 - Bypass of the first stage of synchronization inside GPIO_INT module.

8.5.1.1.128 Control automatic clock gating (AUTOCLKGATEOVERRIDE1)

Offset

Register	Offset
AUTOCLKGATEOVERRIDE1	E24h

Diagram



Fields

Field	Description
31-12 —	Reserved
11 PWM1	PWM1
10 PWM0	PWM0

Table continues on the next page...

Table continued from the previous page...

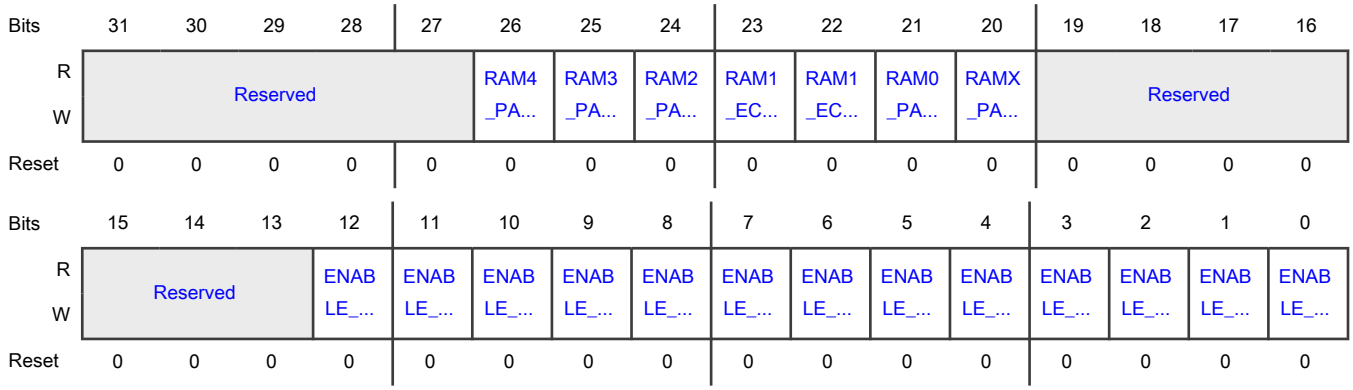
Field	Description
9 VREF	VREF
8 HSCMP2	HSCMP2
7 HSCMP1	HSCMP0
6 HSCMP0	HSCMP0
5 OPAMP2	OPAMP2
4 OPAMP1	OPAMP1
3 OPAMP0	OPAMP0
2 DAC2	DAC2
1 DAC1	DAC1
0 DAC0	DAC0

8.5.1.1.129 Memory parity ECC enable (ENABLE_MEM_PARITY_ECC_CHECK)

Offset

Register	Offset
ENABLE_MEM_PARITY_ECC_CHECK	E30h

Diagram



Fields

Field	Description
31-27 —	Reserved
26 RAM4_PARITY_ERROR_INTEN	Interrupt enable for RAM4 parity error 0 - Disable. 1 - Enable RAM error interrupt when RAM4 parity error status flag is set.
25 RAM3_PARITY_ERROR_INTEN	Interrupt enable for RAM3 parity error 0 - Disable. 1 - Enable RAM error interrupt when RAM3 parity error status flag is set.
24 RAM2_PARITY_ERROR_INTEN	Interrupt enable for RAM2 parity error 0 - Disable. 1 - Enable RAM error interrupt when RAM2 parity error status flag is set.
23 RAM1_ECC_SBIT_ERROR_INTEN	Interrupt enable for RAM1 ECC sbit_err 0 - Disable. 1 - Enable RAM error interrupt when RAM1 ECC sbit_err status flag is set.
22 RAM1_ECC_MBIT_ERROR_INTEN	Interrupt enable for RAM1 ECC mbit_err 0 - Disable. 1 - Enable RAM error interrupt when RAM1 ECC mbit_err status flag is set.
21	Interrupt enable for RAM0 parity error 0 - Disable. 1 - Enable RAM error interrupt when RAM0 parity error status flag is set.

Table continues on the next page...

Table continued from the previous page...

Field	Description
RAM0_PARITY_ERROR_INTEN	
20 RAMX_PARITY_ERROR_INTEN	Interrupt enable for RAMX parity error 0 - Disable. 1 - Enable RAM error interrupt when RAMX parity error status flag is set.
19-13 —	Reserved
12 ENABLE_RAM43	Enable RAM43 parity error check 0 - Disabled 1 - Enabled
11 ENABLE_RAM42	Enable RAM42 parity error check 0 - Disabled 1 - Enabled
10 ENABLE_RAM41	Enable RAM41 parity error check 0 - Disabled 1 - Enabled
9 ENABLE_RAM40	Enable RAM40 parity error check 0 - Disabled 1 - Enabled
8 ENABLE_RAM3	Enable RAM3 parity error check 0 - Disabled 1 - Enabled
7 ENABLE_RAM2	Enable RAM2 parity error check 0 - Disabled 1 - Enabled
6 ENABLE_RAM1_SBIT	Enable RAM1 ECC sbit error check 0 - Disabled 1 - Enabled
5 ENABLE_RAM1_MBIT	Enable RAM1 ECC mbit error check 0 - Disabled 1 - Enabled

Table continues on the next page...

Table continued from the previous page...

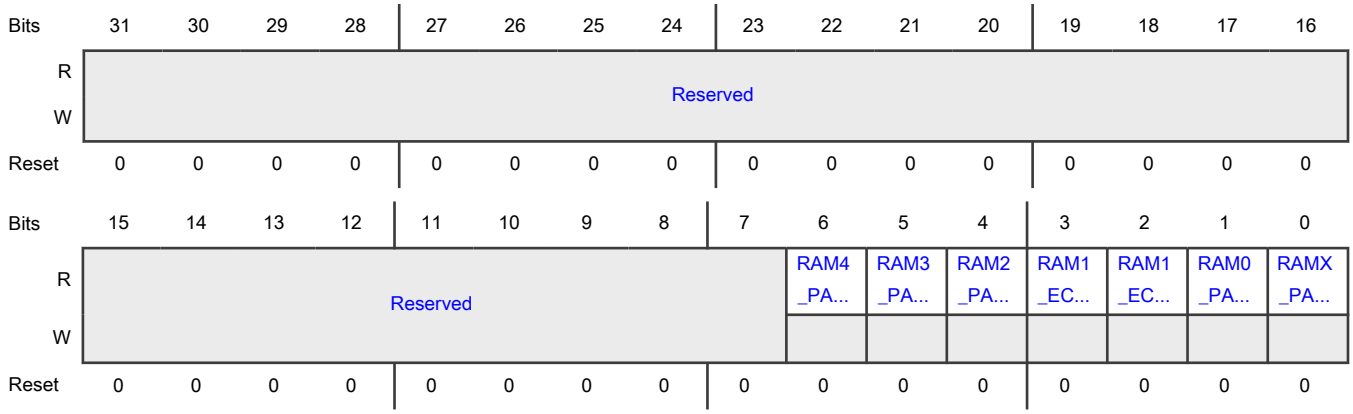
Field	Description
4 ENABLE_RAM03	Enable RAM03 parity error check 0 - Disabled 1 - Enabled
3 ENABLE_RAMx02	Enable RAMx02 parity error check 0 - Disabled 1 - Enabled
2 ENABLE_RAM01	Enable RAM01 parity error check 0 - Disabled 1 - Enabled
1 ENABLE_RAM00	Enable RAM00 parity error check 0 - Disabled 1 - Enabled
0 ENABLE_RAMx	Enable RAMx parity error check 0 - Disabled 1 - Enabled

8.5.1.1.130 Memory parity ECC error flag (MEM_PARITY_ECC_ERROR_FLAG)

Offset

Register	Offset
MEM_PARITY_ECC_ERROR_FLAG	E34h

Diagram



Fields

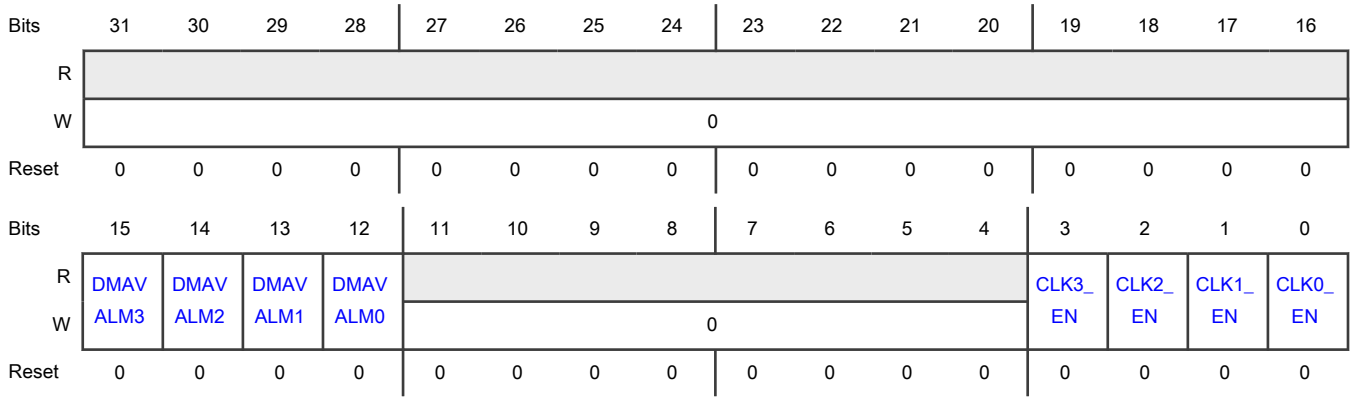
Field	Description
31-7 —	Reserved
6 RAM4_PARITY_ERROR	RAM4 parity error detected 0 - No error detected 1 - Error detected
5 RAM3_PARITY_ERROR	RAM3 parity error detected 0 - No error detected 1 - Error detected
4 RAM2_PARITY_ERROR	RAM2 parity error detected 0 - No error detected 1 - Error detected
3 RAM1_ECC_SBIT_ERROR	RAM1 ECC sbit error detected 0 - No error detected 1 - Error detected
2 RAM1_ECC_MBIT_ERROR	RAM1 ECC mbit error detected 0 - No error detected 1 - Error detected
1 RAM0_PARITY_ERROR	RAM0 parity error detected 0 - No error detected 1 - Error detected
0 RAMX_PARITY_ERROR	RAMx parity error detected 0 - No error detected 1 - Error detected

8.5.1.1.131 PWM0 submodule control (PWM0SUBCTL)

Offset

Register	Offset
PWM0SUBCTL	E38h

Diagram



Fields

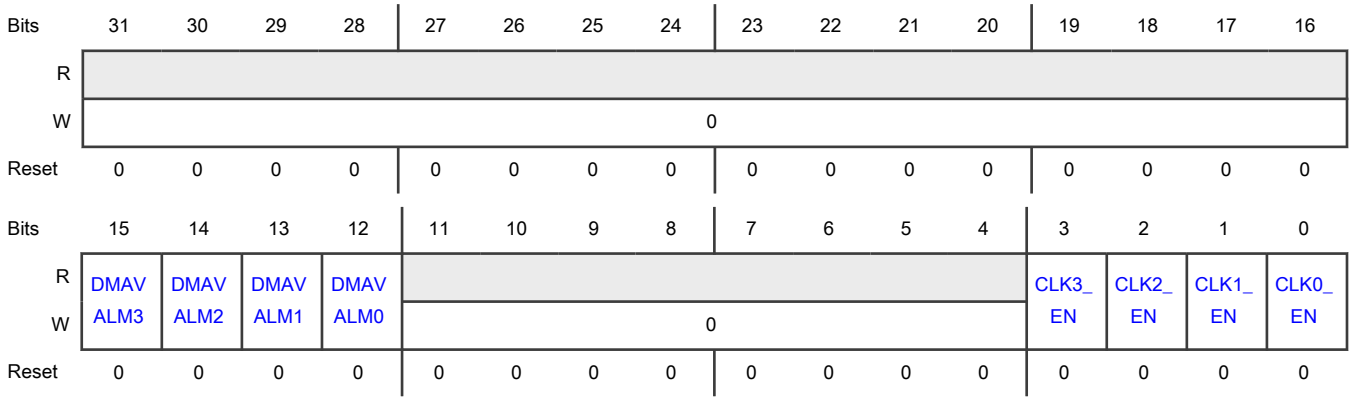
Field	Description
31-16 —	Reserved
15-12 DMAVALMn	PWM0 submodule n DMA Compare Value Done Mask When the mask is used, DMA should write the PWM's MCTRL register in the end, to set the LDOK bits of relevant submodules at the same time. Since the DMA done signal is blocked, t he DMA request will not be cleared automatically by hardware. After the DMA transfer is completed for the current DMA request, the DMA request needs to be cleared manually by clearing SMxDMAEN[VALDE] register bit followed by clearing SMxSTS[RF] bit with software.
11-4 —	Reserved
3-0 CLKn_EN	PWM0 SUB Clockn enable

8.5.1.1.132 PWM1 submodule control (PWM1SUBCTL)

Offset

Register	Offset
PWM1SUBCTL	E3Ch

Diagram



Fields

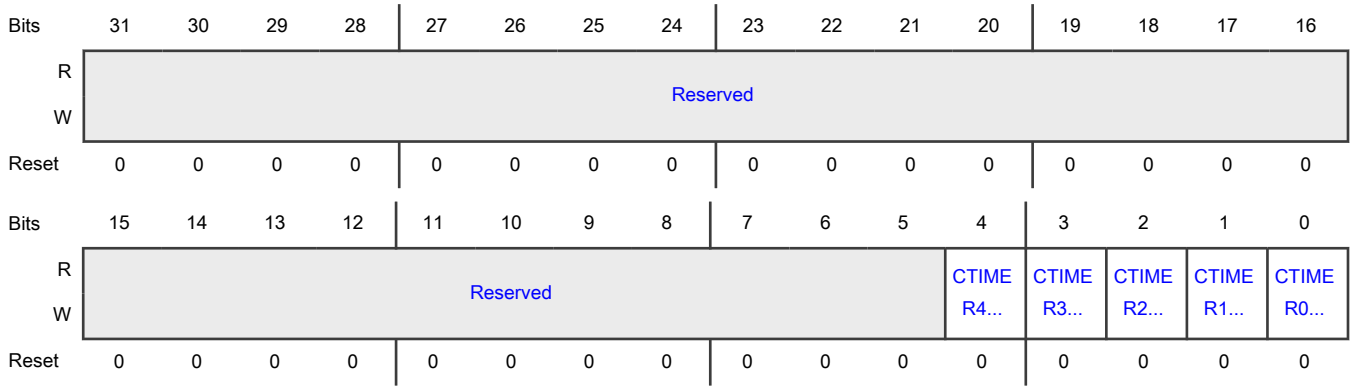
Field	Description
31-16 —	Reserved
15-12 DMAVALMn	PWM1 submodule n DMA Compare Value Done Mask When the mask is used, DMA should write the PWM's MCTRL register in the end, to set the LDOK bits of relevant submodules at the same time. Since the DMA done signal is blocked, t he DMA request will not be cleared automatically by hardware. After the DMA transfer is completed for the current DMA request, the DMA request needs to be cleared manually by clearing SMxDMAEN[VALDE] register bit followed by clearing SMxSTS[RF] bit with software.
11-4 —	Reserved
3-0 CLKn_EN	PWM1 SUB Clockn enable

8.5.1.1.133 CTIMER global start enable (CTIMERGLOBALSTARTEN)

Offset

Register	Offset
CTIMERGLOBALSTARTEN	E40h

Diagram



Fields

Field	Description
31-5 —	Reserved
4 CTIMER4_CLK_EN	CTIMER4 function clock enable
3 CTIMER3_CLK_EN	CTIMER3 function clock enable
2 CTIMER2_CLK_EN	CTIMER2 function clock enable
1 CTIMER1_CLK_EN	CTIMER1 function clock enable
0 CTIMER0_CLK_EN	CTIMER0 function clock enable

8.5.1.1.134 Control write access to security (DEBUG_LOCK_EN)

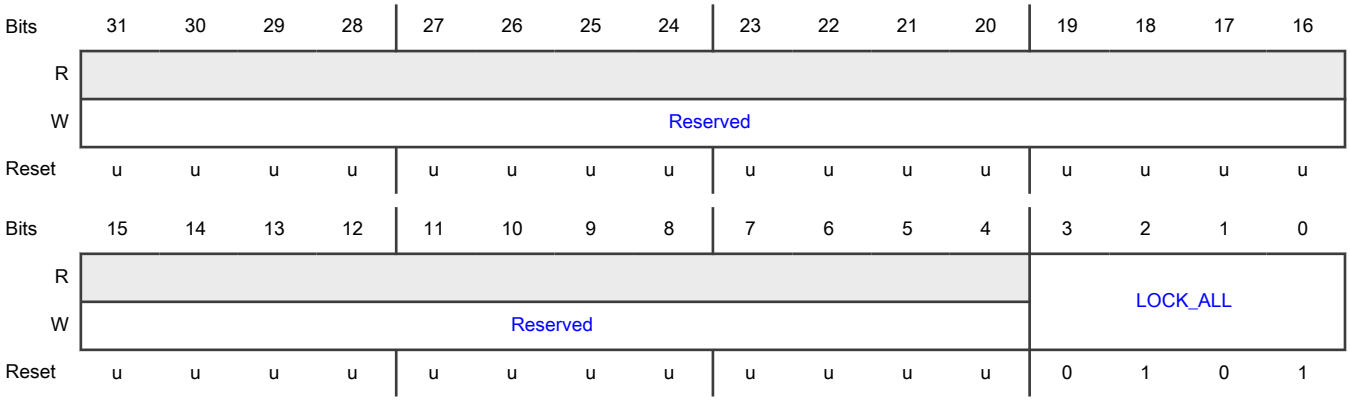
Control write access to DEBUG_FEATURES.

Control write access to SWD_ACCESS_CPU0, SWD_ACCESS_DSP, and DEBUG_AUTH_BEACON, DEBUG_FEATURES registers

Offset

Register	Offset
DEBUG_LOCK_EN	FA0h

Diagram



Fields

Field	Description
31-4 —	Reserved Read value is undefined, only zero should be written.
3-0 LOCK_ALL	Control write access to security registers. Control write access to , DEBUG_FEATURES register. 0000 - Any other value than b1010: disable write access to all registers. 1010 - 1010: Enable write access to all registers.

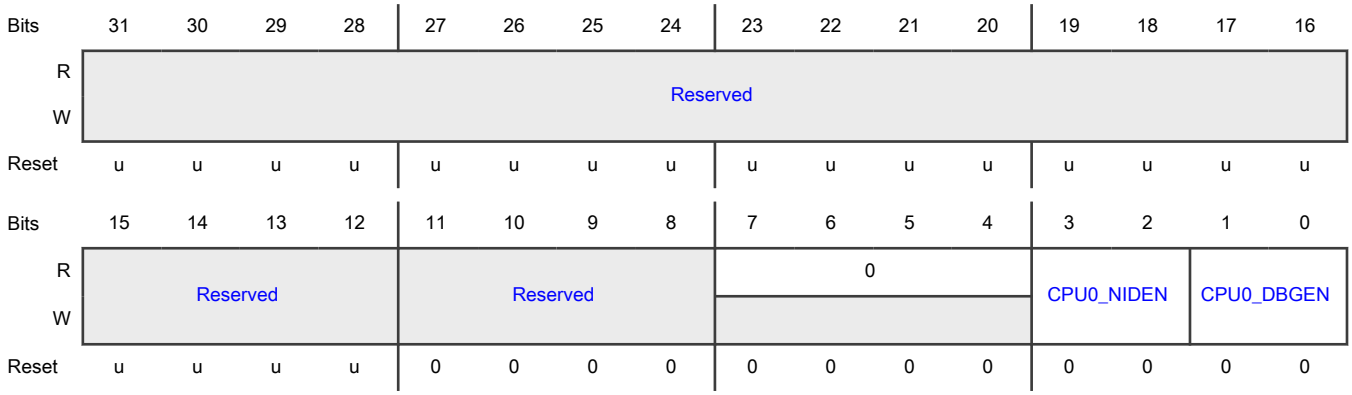
8.5.1.1.135 Cortex debug features control (DEBUG_FEATURES)

Cortex M33 (CPU0) debug features control.

Offset

Register	Offset
DEBUG_FEATURES	FA4h

Diagram



Fields

Field	Description
31-12 —	Reserved Read value is undefined, only zero should be written.
11-8 —	Reserved Read value is undefined, only zero should be written.
7-4 —	Reserved Read value is undefined, only zero should be written.
3-2 CPU0_NIDEN	CPU0 Non Invasive Debug Control Enables non-invasive debug for CPU0. (Values not listed are reserved.) 01 - Disable debug 10 - Enable debug
1-0 CPU0_DBGEN	CPU0 Invasive Debug Control Enables invasive debug for CPU0. (Values not listed are reserved.) 01 - Disable debug 10 - Enable debug

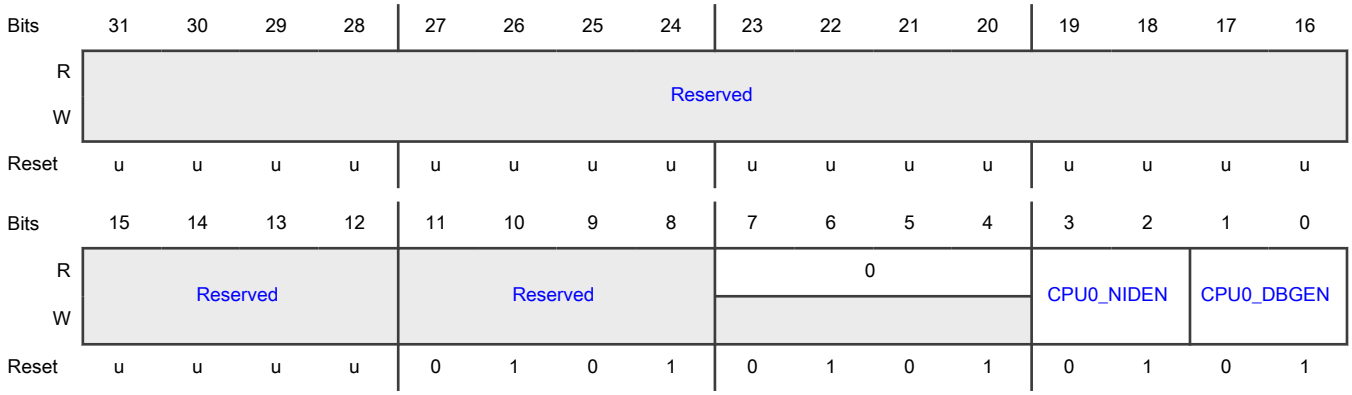
8.5.1.1.136 Cortex debug features control (duplicate) (DEBUG_FEATURES_DP)

Cortex M33 (CPU0) debug features control.

Offset

Register	Offset
DEBUG_FEATURES_DP	FA8h

Diagram



Fields

Field	Description
31-12 —	Reserved Read value is undefined, only zero should be written.
11-8 —	Reserved
7-4 —	Reserved Read value is undefined, only zero should be written.
3-2 CPU0_NIDEN	CPU0 Non Invasive Debug Control Enables non-invasive debug for CPU0. (Values not listed are reserved.) 01 - Disable debug 10 - Enable debug
1-0 CPU0_DBGEN	CPU0 Invasive Debug Control Enables invasive debug for CPU0. (Values not listed are reserved.) 01 - Disable debug 10 - Enable debug

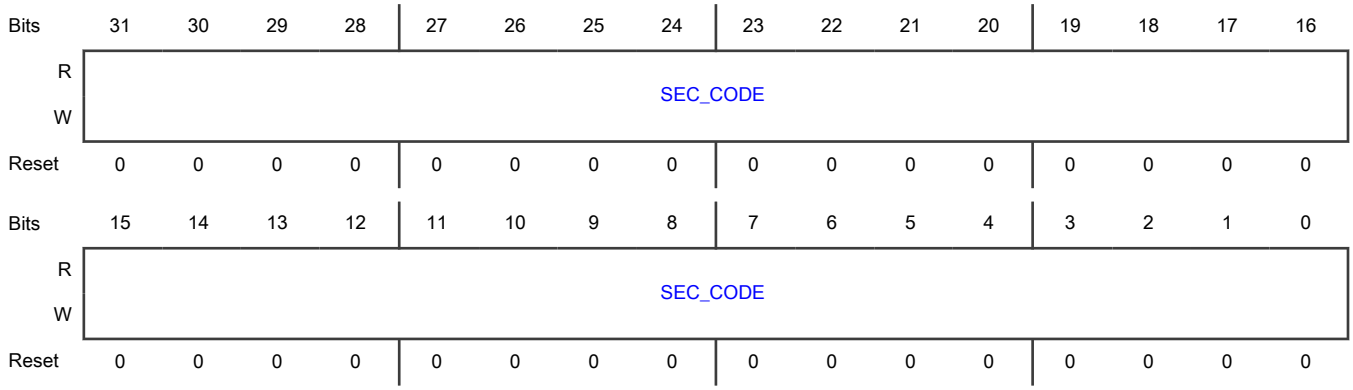
8.5.1.1.137 CPU0 Software Debug Access (SWD_ACCESS_CPU0)

This register is used by ROM during DEBUG authentication mechanism to enable debug access port for CPU0.

Offset

Register	Offset
SWD_ACCESS_CPU0	FB4h

Diagram



Fields

Field	Description
31-0 SEC_CODE	CPU0 SWD-AP: 0x12345678. 0000_0000_0000_0000_0000_0000_0000_0000 - CPU0 DAP is not allowed. Reading back register will be read as 0x5. 0001_0010_0011_0100_0101_0110_0111_1000 - Value to write to enable CPU0 SWD access. Reading back register will be read as 0xA.

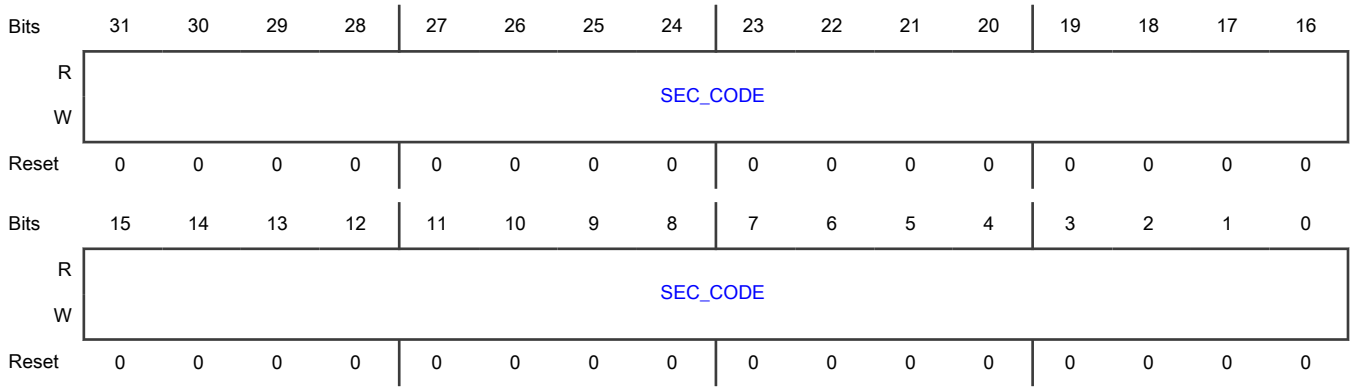
8.5.1.1.138 DSP Software Debug Access (SWD_ACCESS_DSP)

This register is used by ROM during DEBUG authentication mechanism to enable debug access port for DSP.

Offset

Register	Offset
SWD_ACCESS_DSP	FC4h

Diagram



Fields

Field	Description
31-0 SEC_CODE	DSP SWD-AP: 0x12345678. 0000_0000_0000_0000_0000_0000_0000_0000 - DSP DAP is not allowed. Reading back register will be read as 0x5. 0001_0010_0011_0100_0101_0110_0111_1000 - Value to write to enable DSP SWD access. Reading back register will be read as 0xA.

8.5.1.1.139 Device ID (DEVICE_ID0)

This register contains the device ID.

The following table shows the DEVICE_ID for the LPC553x family:

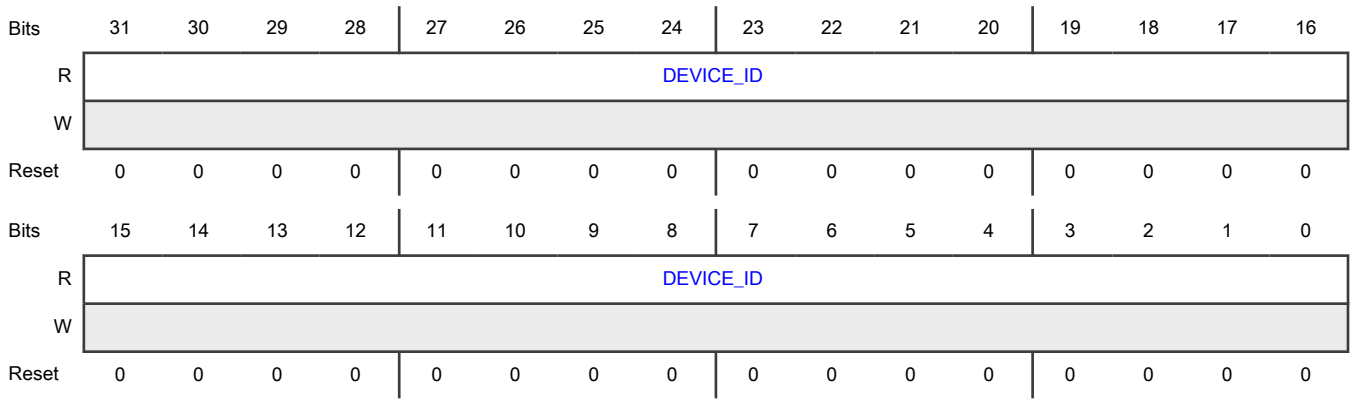
Table 30. DEVICE_ID

Part Number (Rev 0A)	LPC5536JBD100	LPC5536JBD64	LPC5536JHI48	LPC5534JBD100	LPC5534JBD64	LPC5534JHI48
DEVICE_ID0 Value	0x505a11a0	0x505a11a0	0x505a11a0	0x505a0920	0x505a0920	0x505a0920

Offset

Register	Offset
DEVICE_ID0	FF8h

Diagram



Fields

Field	Description
31-0 DEVICE_ID	Device ID.

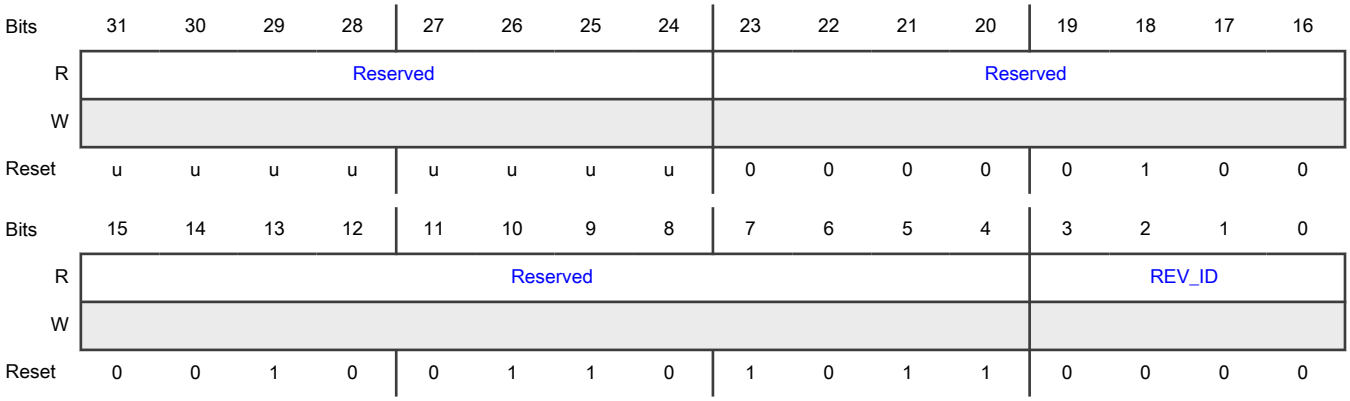
8.5.1.1.140 Chip revision ID and Number (DIEID)

This register contains the Chip Number and Revision.

Offset

Register	Offset
DIEID	FFCh

Diagram



Fields

Field	Description
31-24 —	Reserved Read value is undefined, only zero should be written.
23-4 —	Reserved Read value is undefined, only zero should be written.
3-0 REV_ID	Device Revision. Value read as 0x0 applies to device revision 0A and 1 applies to device revision 1B.

Chapter 9
Analog Controller (ANACTRL)

9.1 Chip-specific ANACTRL information

Table 31. Reference links to related information

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

9.1.1 Configuration

Configuration setting for ANACTRL module is listed below:

- Set the ANALOG_CTRL bit in the AHBCLKCTRL2 register to enable the clock to the analog controller module.

NOTE

The clock to analog controller module is enabled during boot time by the boot loader and should always remain enabled.

- The 32 kHz Free Running Oscillator will be automatically enabled when: the RTC_OSP_PD bit in the RTC control register = 0 and the SEL bit in the RTCOSC32K register = 0, or when the OSC32KPD bit in OSTIMER register = 0 and the SEL bit in the RTCOSC32K = 0, or when the PDEN_FRO32K bit in PDRUNCFG0 = 0.

9.1.1 Module instances

This device has one instance of the ANACTRL module, ANACTRL0.

9.2 Overview

This section provides an introduction to the Analog Controller module.

9.2.1 Features

- Internal Free Running Oscillator (FRO). This oscillator provides a selectable 96 MHz output, and a 12 MHz output (divided down from the selected higher frequency) that can be used as a system clock.
- Internal 32 KHz FRO.
- High-speed Crystal oscillator module control and status (from 16 MHz to 32 MHz).
- All Brown out Detectors (BoD) and DCDC converter interrupt control and status.
- Ring oscillators (True Random Number Generator clock sources) function control.
- All Crystal oscillators (both 32 kHz and high-speed 12 MHz to 32 MHz) capacitive banks calibration functions control.
- Some USB high-speed physical interface parameters control.

9.3 Memory map and register definition

This section includes the Analog Controller module memory map and detailed descriptions of all registers.

9.3.1 Analog Controller register descriptions

9.3.1.1 ANACTRL memory map

ANACTRL0 base address: 4001_3000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Various Analog blocks configuration (like FRO 192MHz trimmings source …) (ANALOG_CTRL_CFG)	32	See section	See section
4	Analog Control and Status (ANALOG_CTRL_STATUS)	32	See section	See section
10	192MHz Free Running Oscillator (FRO) Control (FRO192M_CTRL)	32	See section	See section
14	192MHz Free Running Oscillator (FRO) Status (FRO192M_STATUS)	32	See section	See section
18	General Purpose ADC VBAT Divider branch control (ADC_CTRL)	32	See section	See section
20	High speed Crystal Oscillator Control register (XO32M_CTRL)	32	See section	See section
24	High speed Crystal Oscillator Status (XO32M_STATUS)	32	See section	See section
30	Brown Out Detectors & DCDC interrupt control (BOD_DCDC_INT_CTRL)	32	See section	See section
34	BoDs & DCDC interrupt status (BOD_DCDC_INT_STATUS)	32	See section	See section
B0	High Speed Crystal Oscillator (16 MHz - 32 MHz) Voltage Source Supply Control register (LDO_XO32M)	32	See section	See section
B4	AUX_BIAS (AUX_BIAS)	32	See section	See section
F0	Oscillators Analog Macrobloc ACBUS and DCBUS control (OSC_TESTBUS)	32	See section	See section
F8	Dummy Control bus to analog modules (DUMMY_CTRL)	32	See section	See section

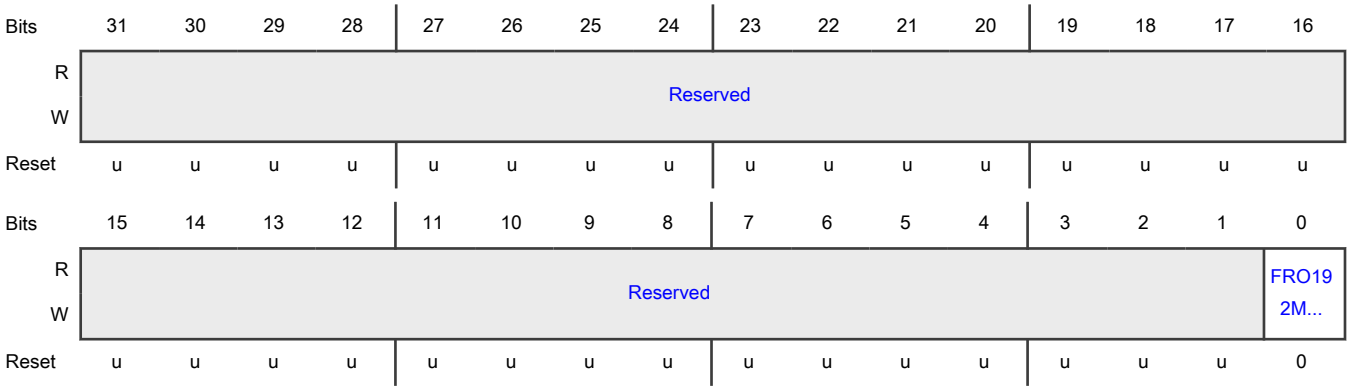
9.3.1.1.1 Various Analog blocks configuration (like FRO 192MHz trimmings source …) (ANALOG_CTRL_CFG)

The Analog Control Configuration register gathers miscellaneous general configurations related to various analog modules.

Offset

Register	Offset
ANALOG_CTRL_CFG	0h

Diagram



Fields

Field	Description
31-1 —	Reserved
0 FRO192M_TRIM_SRC	FRO192M trimming and 'Enable' source. 0 - FRO192M trimming and 'Enable' comes from OTP-eFUSE. 1 - FRO192M trimming and 'Enable' comes from FRO192M_CTRL registers.

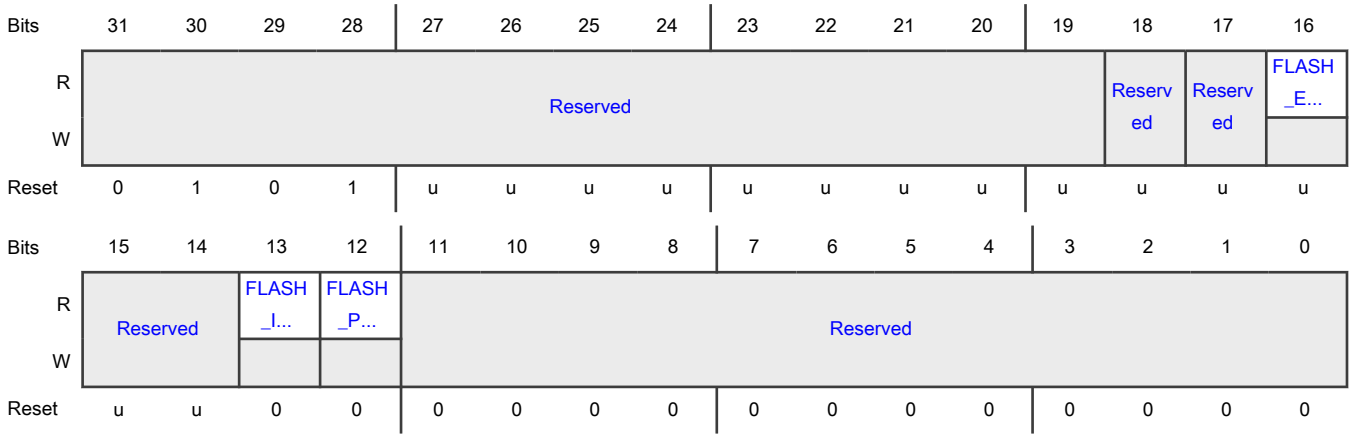
9.3.1.1.2 Analog Control and Status (ANALOG_CTRL_STATUS)

The Analog Control Status register (ANALOG_CTRL_STATUS) contains information related to the Flash status.

Offset

Register	Offset
ANALOG_CTRL_STATUS	4h

Diagram



Fields

Field	Description
31-19 —	Reserved
18 —	Reserved
17 —	Reserved
16 FLASH_ECC_ERROR_FLAG	Flash ECC Error Flag
15-14 —	Reserved
13 FLASH_INIT_ERROR	Flash initialization error status Indicates whether an error has occurred during flash initialization. 0 - No error 1 - At least one error occurred
12 FLASH_PWRDWN	Flash Power Down status Indicates whether the flash is in power down mode. 0 - Not in power down mode. 1 - In power down mode.
11-0 —	Reserved

9.3.1.1.3 192MHz Free Running Oscillator (FRO) Control (FRO192M_CTRL)

The 192MHz Free Running Oscillator (FRO) Control register (FRO192M_CTRL) configures the on-chip high speed Free Running Oscillator (FRO192 MHz) and the automatic USB rate adjustment mode (USBCLKADJ).

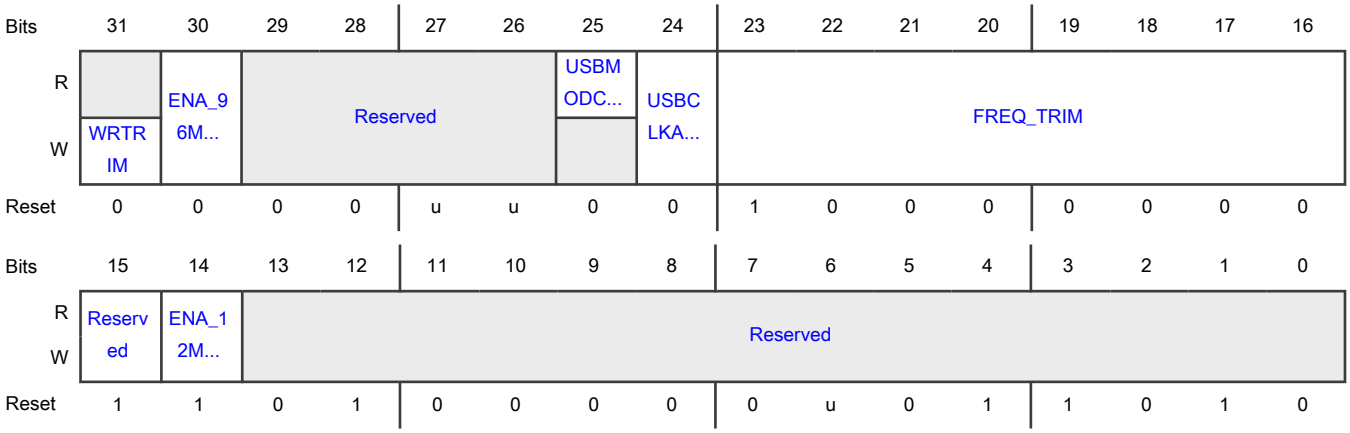
Notes on using USBCLKADJ:

- When turning on USBCLKADJ, the current FREQTRIM value will be used as the starting value. From then on, the adjusted value will be used as long as enabled (whether USB is active or not).
- When USBCLKADJ is turned off, the application may take one of the following actions:
  - Read the register to pick up the adjusted FREQTRIM and then write back with the USBADJ cleared. The FRO will continue to use the adjusted value.
  - If software saved the original factory trimmed value of FREQTRIM, it can be written back as above.

Offset

Register	Offset
FRO192M_CTRL	10h

Diagram



Fields

Field	Description
31 WRTRIM	This must be written to 1 to modify the BIAS_TRIM and TEMP_TRIM fields.
30 ENA_96MHZCLK	96 MHz clock control 0 - Disable the 96 MHz clock. 1 - Enable the 96 MHz clock.
29-26 —	Reserved

Table continues on the next page...

Table continued from the previous page...

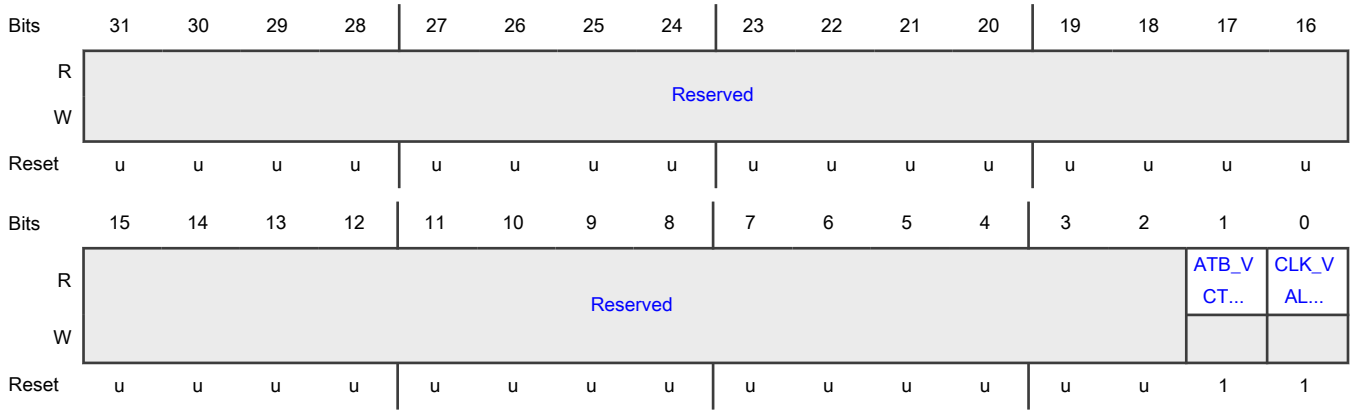
Field	Description
25 USBMODCHG	USBCLKADJ mode trim change Applies in automatic USB rate adjustment mode (USBCLKADJ=1) only. When USBMODCHG reads as 1, the FREQTRIM value is not stable. Re-read the FREQTRIM field until USBMODCHG = 0.
24 USBCLKADJ	If USBCLKADJ bit is set and the USB peripheral is enabled for full speed device mode, the USB block will provide FRO clock adjustments to synchronize the frequency to the host clock using the SOF packets.
23-16 FREQ_TRIM	Frequency trim. Boot code configures this to a device-specific factory trim value for the FRO. When USBCLKADJ = 1, this field is read-only and provides the value resulting from USB rate adjustment. See the USBMODCFG flag regarding reading this field. When USBCLKADJ = 0, application code may adjust this field. A single step of DAC_TRIM is roughly equivalent to 0.1% of the selected FRO frequency.
15 —	Reserved Only 1 should be written. Writing zero prevents the Flash from working.
14 ENA_12MHZCLK	12 MHz clock control. 0 - Disable the 12 MHz clock. 1 - Enable the 12 MHz clock.
13-0 —	Reserved

9.3.1.1.4 192MHz Free Running Oscillator (FRO) Status (FRO192M_STATUS)

Offset

Register	Offset
FRO192M_STATUS	14h

Diagram



Fields

Field	Description
31-2 —	Reserved
1 ATB_VCTRL	CCO threshold voltage detector output (signal vcco_ok).
0 CLK_VALID	Output clock valid. Indicates that the CCO clock has settled. When CLK_VALID = 1, the output clocks (12 MHz, 48 MHz or 96 MHz) can be output if they are enabled in the FRO192M_CTRL register. 0 - No output clock available 1 - Output clock is available

9.3.1.1.5 General Purpose ADC VBAT Divider branch control (ADC_CTRL)

Offset

Register	Offset
ADC_CTRL	18h

Diagram

Fields

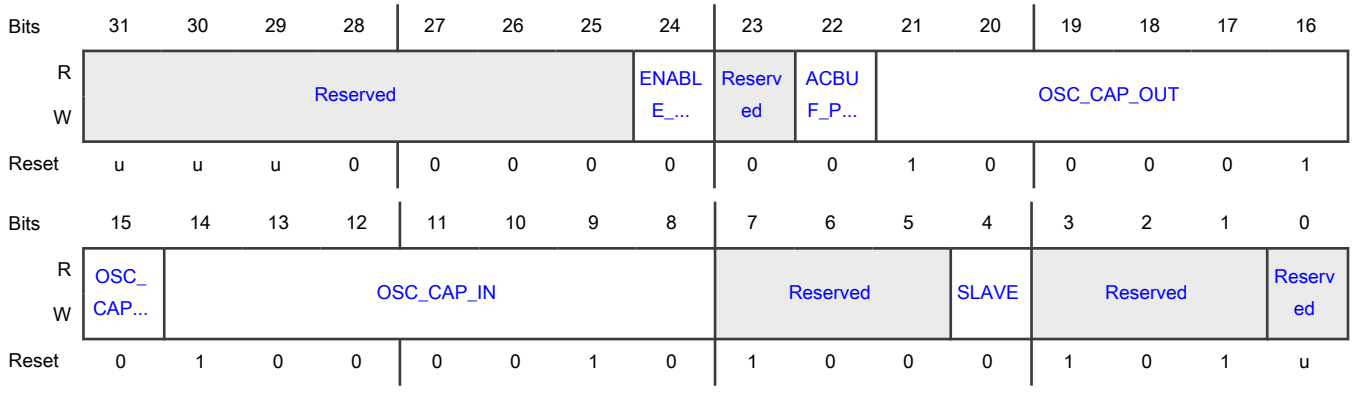
Field	Description
31-1 —	Reserved
0 VBATDIVENABLE	Switch On/Off VDDMAIN divider branch. 0 - VDDMAIN divider branch is disabled. 1 - VDDMAIN divider branch is enabled.

9.3.1.1.6 High speed Crystal Oscillator Control register (XO32M_CTRL)

Offset

Register	Offset
XO32M_CTRL	20h

Diagram



Fields

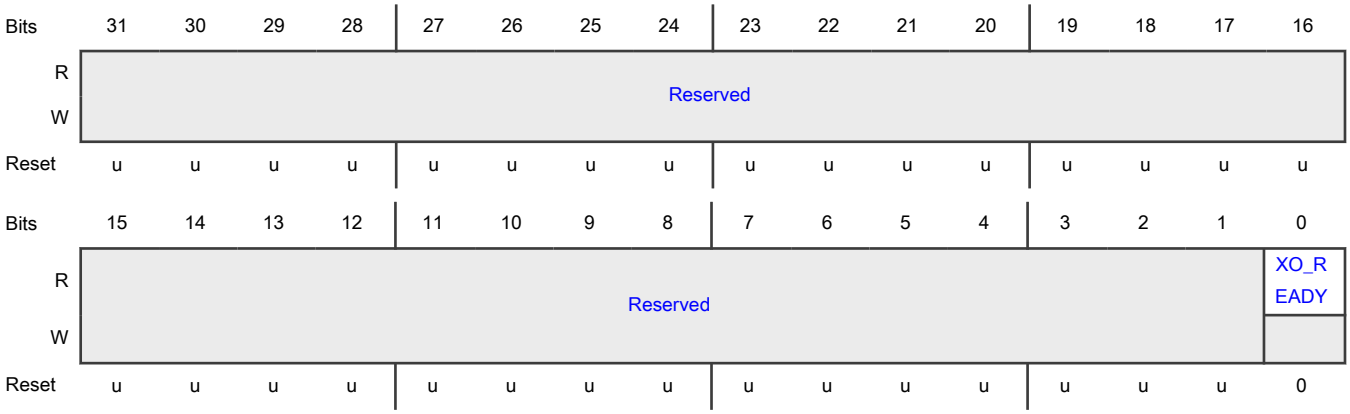
Field	Description
31-25 —	Reserved
24 ENABLE_SYSTEM_CLK_OUT	Enable High speed Crystal oscillator output to CPU system. 0 - Disable the oscillator. 1 - Enable the oscillator.
23 —	Reserved
22 ACBUF_PASS_ENABLE	Allows XO32M to be configured in bypass mode. 0 - XO bypass is disabled. 1 - XO bypass is enabled.
21-15 OSC_CAP_OUT	Tune capa banks of High speed Crystal Oscillator output pin **NOTE** Do not modify these bit values. These registers are configured by the SDK power library. Changes to the bits can cause application failure. NXP is not responsible for any change to these bits and is not obligated to provide support.
14-8 OSC_CAP_IN	Tune capa banks of High speed Crystal Oscillator input pin **NOTE** Do not modify these bit values. These registers are configured by the SDK power library. Changes to the bits can cause application failure. NXP is not responsible for any change to these bits and is not obligated to provide support.
7-5 —	Reserved
4 SLAVE	XO in slave mode. **NOTE** Do not modify these bit values. These registers are configured by the SDK power library. Changes to the bits can cause application failure. NXP is not responsible for any change to these bits and is not obligated to provide support.
3-1 —	Reserved
0 —	Reserved

9.3.1.1.7 High speed Crystal Oscillator Status (XO32M_STATUS)

Offset

Register	Offset
XO32M_STATUS	24h

Diagram



Fields

Field	Description
31-1 —	Reserved
0 XO_READY	Crystal Oscillator Ready Indicates XO out frequency stability. 0 - Frequency is not yet stable. 1 - Frequency is stable.

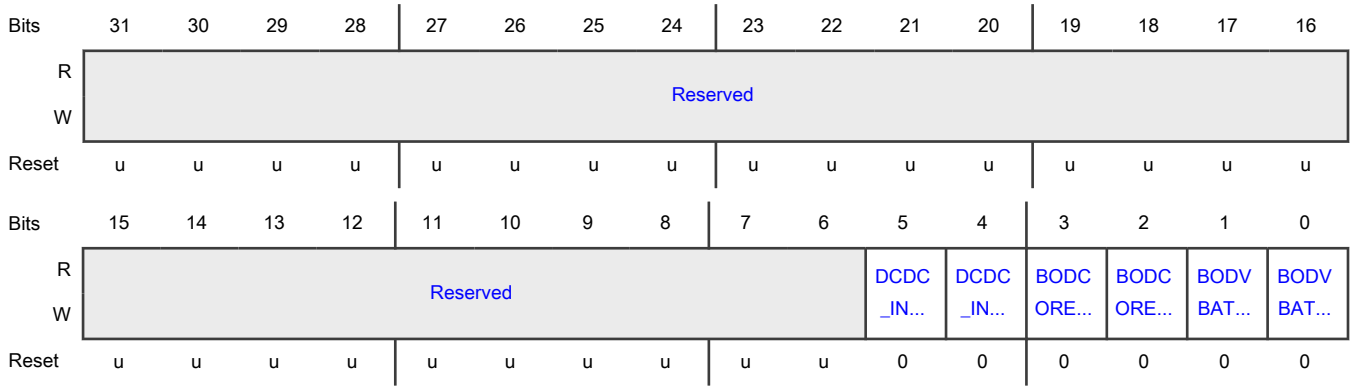
9.3.1.1.8 Brown Out Detectors & DCDC interrupt control (BOD_DCDC_INT_CTRL)

Use the Brown Out Detectors & DCDC Interrupt Control register (BOD_DCDC_INT_CTRL) to manage interrupts coming from the DCDC and the BoDs of the VDDMAIN and CORE voltage domains.

Offset

Register	Offset
BOD_DCDC_INT_CTRL	30h

Diagram



Fields

Field	Description
31-6 —	Reserved
5 DCDC_INT_CLEAR	DCDC interrupt clear.1: Clear the interrupt. Self-cleared bit.
4 DCDC_INT_ENABLE	DCDC interrupt control. 0 - Disable the interrupt. 1 - Enable the interrupt.
3 BODCORE_INT_CLEAR	BOD CORE interrupt clear.1: Clear the interrupt. Self-cleared bit.
2 BODCORE_INT_ENABLE	BOD CORE interrupt control. 0 - Disable the interrupt. 1 - Enable the interrupt.
1 BODVBAT_INT_CLEAR	BOD VDDMAIN interrupt clear.1: Clear the interrupt. Self-cleared bit.
0 BODVBAT_INT_ENABLE	BOD VDDMAIN interrupt control. 0 - Disable the interrupt. 1 - Enable the interrupt.

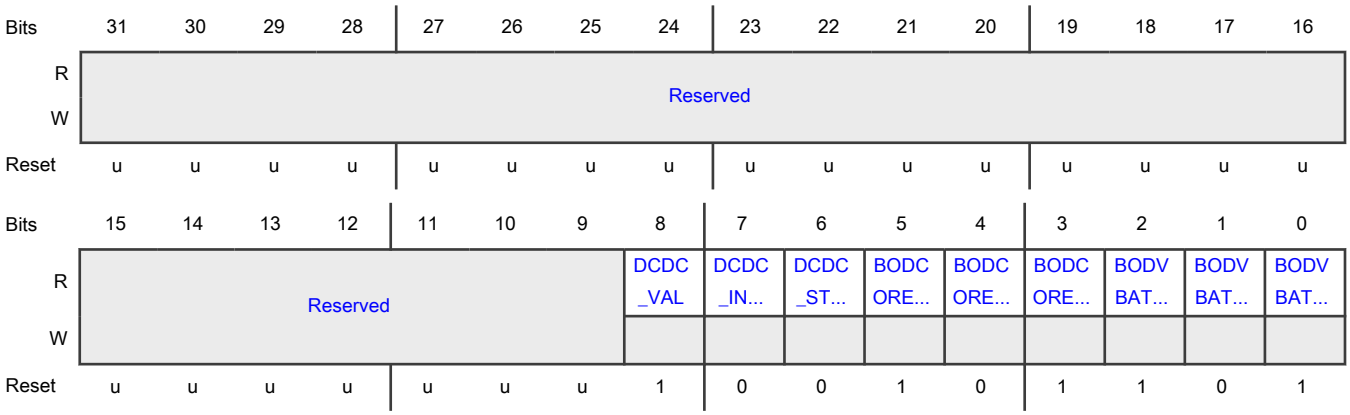
9.3.1.1.9 BoDs & DCDC interrupt status (BOD_DCDC_INT_STATUS)

The Brown Out Detectors & DCDC Interrupt Status register (BOD_DCDC_INT_STAT) provides the output status and the interrupt status of the BoD VDDMAIN, BoD CORE, and the DCDC.

Offset

Register	Offset
BOD_DCDC_INT_STATUS	34h

Diagram



Fields

Field	Description
31-9 —	Reserved
8 DCDC_VAL	DCDC power status Indicates the current value of the DCDC output voltage relative to the targeted regulation level. 0 - Below the target. 1 - Above the target.
7 DCDC_INT_STATUS	DCDC Interrupt status after Interrupt Enable. 0 - No interrupt pending. 1 - Interrupt pending.
6 DCDC_STATUS	DCDC Interrupt status before Interrupt Enable. 0 - No interrupt pending. 1 - Interrupt pending.
5 BODCORE_VAL	BOD CORE power status Indicates the current value of the CORE voltage level relative to the BoD threshold. 0 - Below the threshold. 1 - Above the threshold.

Table continues on the next page...

Table continued from the previous page...

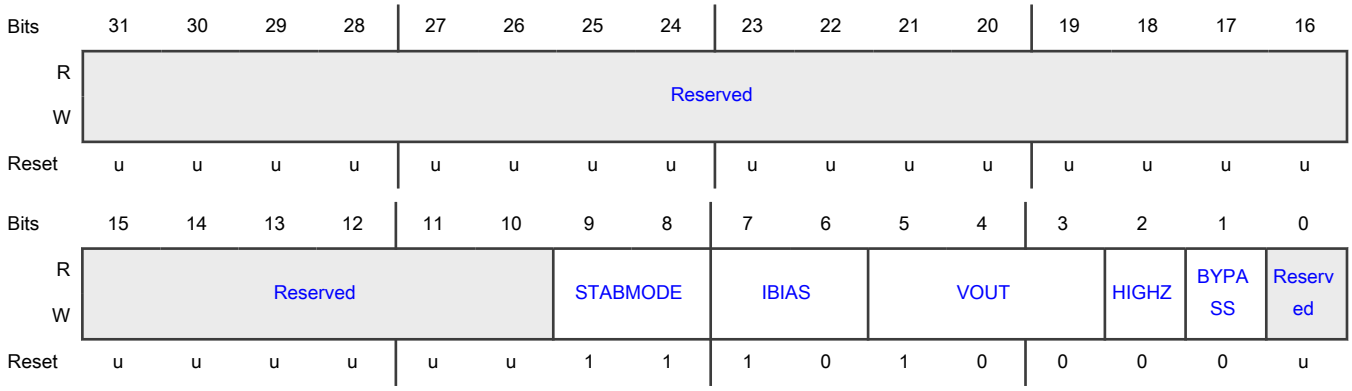
Field	Description
4 BODCORE_INT_STATUS	BOD CORE Interrupt status after Interrupt Enable. 0 - No interrupt pending. 1 - Interrupt pending.
3 BODCORE_STATUS	BOD CORE Interrupt status before Interrupt Enable. 0 - No interrupt pending. 1 - Interrupt pending.
2 BODVBAT_VAL	BOD VDDMAIN power status Indicates the current value of the VDDMAIN voltage level relative to the BoD threshold. 0 - Below the threshold. 1 - Above the threshold.
1 BODVBAT_INT_STATUS	BOD VDDMAIN Interrupt status after Interrupt Enable. 0 - No interrupt pending. 1 - Interrupt pending.
0 BODVBAT_STATUS	BOD VDDMAIN Interrupt status before Interrupt Enable. 0 - No interrupt pending. 1 - Interrupt pending.

9.3.1.1.10 High Speed Crystal Oscillator (16 MHz - 32 MHz) Voltage Source Supply Control register (LDO_XO32M)

Offset

Register	Offset
LDO_XO32M	B0h

Diagram



Fields

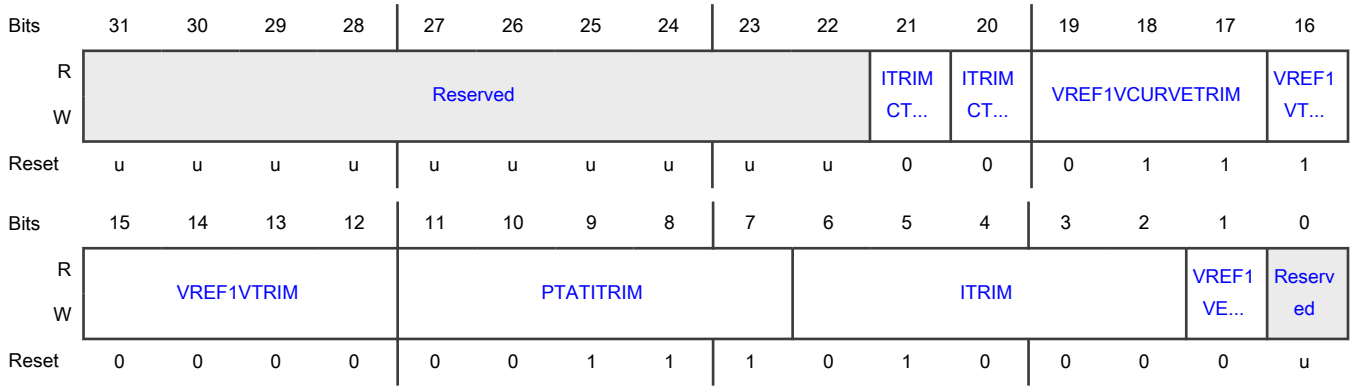
Field	Description
31-10 —	Reserved
9-8 STABMODE	Stability configuration.
7-6 IBIAS	Adjust the biasing current.
5-3 VOUT	Sets the LDO output level. 000 - 0.750 V. 001 - 0.775 V. 010 - 0.800 V. 011 - 0.825 V. 100 - 0.850 V. 101 - 0.875 V. 110 - 0.900 V. 111 - 0.925 V.
2 HIGHZ	. 0 - Output in High normal state. 1 - Output in High Impedance state.
1 BYPASS	Activate LDO bypass. 0 - Disable bypass mode (for normal operations). 1 - Activate LDO bypass.
0 —	Reserved

9.3.1.1.11 AUX_BIAS (AUX_BIAS)

Offset

Register	Offset
AUX_BIAS	B4h

Diagram



Fields

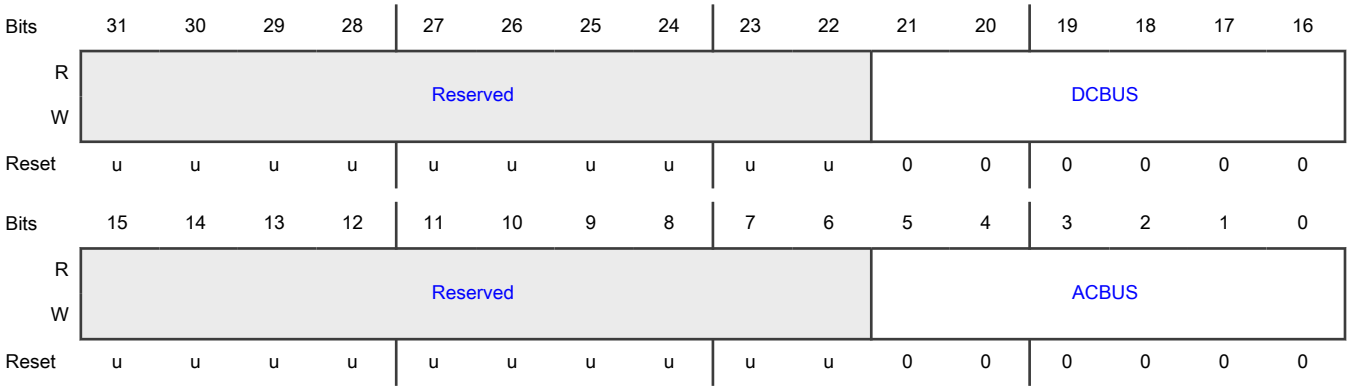
Field	Description
31-22 —	Reserved
21 ITRIMCTRL1	Control bit to configure trimming state of mirror.
20 ITRIMCTRL0	Control bit to configure trimming state of mirror.
19-17 VREF1VCURVETRIM	Control bit to configure trimming state of mirror.
16-12 VREF1VTRIM	voltage trimming control word.
11-7 PTATITRIM	current trimming control word for ptat current.
6-2 ITRIM	current trimming control word.
1 VREF1VENABLE	Control output of 1V reference voltage. 0 - Output of 1V reference voltage buffer is bypassed. 1 - Output of 1V reference voltage is enabled.
0 —	Reserved

9.3.1.1.12 Oscillators Analog Macrobloc ACBUS and DCBUS control (OSC_TESTBUS)

Offset

Register	Offset
OSC_TESTBUS	F0h

Diagram



Fields

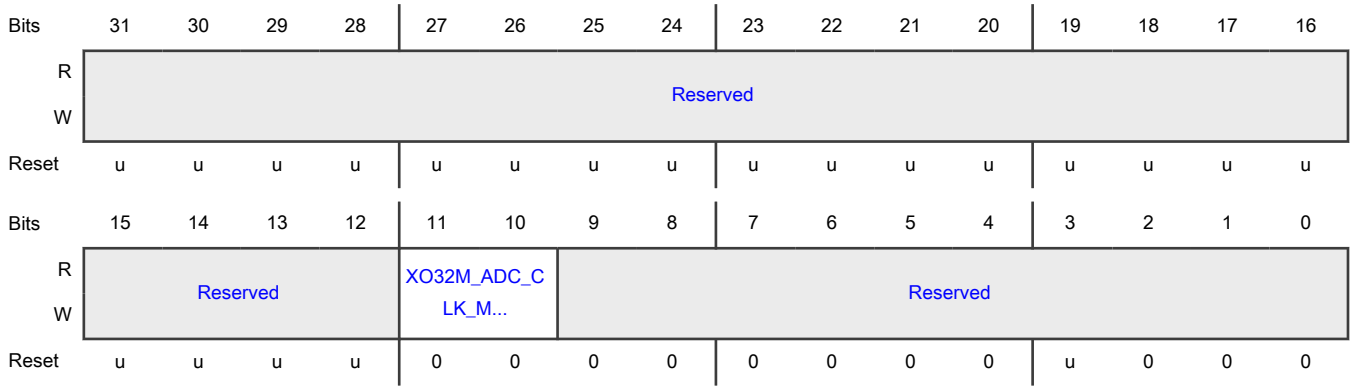
Field	Description
31-22 —	Reserved
21-16 DCBUS	Direct current BUS
15-6 —	Reserved
5-0 ACBUS	Alternate current BUS

9.3.1.1.13 Dummy Control bus to analog modules (DUMMY_CTRL)

Offset

Register	Offset
DUMMY_CTRL	F8h

Diagram



Fields

Field	Description
31-12 —	Reserved
11-10 XO32M_ADC_CLK_MODE	Control High speed Crystal oscillator mode of the ADC clock. 2b00: High speed Crystal oscillator output to ADC is disabled. 2b01: High speed Crystal oscillator output to ADC is enabled. 00 - High speed Crystal oscillator output to ADC is disabled. 01 - High speed Crystal oscillator output to ADC is enable.
9-0 —	Reserved

Chapter 10
Cap Bank API

10.1 Overview

There are two crystal oscillators in this device: A High Frequency crystal oscillator (also referred as High Speed crystal oscillator) and a 32 kHz crystal oscillator (also referred as Low Speed crystal oscillator).

Each crystal oscillator has one embedded capacitor bank, where each can be used as an integrated load capacitor for the crystal oscillators. The capacitor banks on each crystal pin can tune the frequency for crystals with a Capacitive Load (CL) between 6 to 10pF (IEC equivalent).

10.1.1 Features

- Conserves space on the PCB.
- Reduces the overall Bill of Materials (BOM).
- Accommodates a Capacitive Load (CL) between 6 - 10 pF (IEC equivalent).
- Allows simple APIs to configure the Capacitor Banks based on the crystal Capacitive Load (CL) and measured PCB parasitic capacitances on XIN and XOUT pins.

10.2 Crystal Oscillator Capacitor Banks API description

Table 32. Low power API calls

Function prototype	API description	Section
void CLOCK_XtalHfCapabankTrim (int32_t pi32_hfXtalIecLoadpF_x100, int32_t pi32_hfXtalPPcbParCappF_x100, int32_t pi32_hfXtalNPcbParCappF_x100);	This API configures the Capacitor Bank of the High Frequency crystal oscillator.	
void CLOCK_Xtal32khzCapabankTrim (int32_t pi32_32kfXtalIecLoadpF_x100, int32_t pi32_32kfXtalPPcbParCappF_x100, int32_t pi32_32kfXtalNPcbParCappF_x100) ;	This API configures the Capacitor Bank of the 32 kHz crystal oscillator.	

10.2.1 CLOCK_XtalHfCapabankTrim

This API function configures the High Frequency crystal oscillator Capacitor Bank.

Table 33. CLOCK_XtalHfCapabankTrim API routine

Routine	CLOCK_XtalHfCapabankTrim
SKD prototype	void CLOCK_XtalHfCapabankTrim (int32_t pi32_hfXtalIecLoadpF_x100, int32_t pi32_hfXtalPPcbParCappF_x100, int32_t pi32_hfXtalNPcbParCappF_x100)
Input parameter	**Param0:** pi32_hfXtalIecLoadpF_x100 **Param1:** pi32_hfXtalPPcbParCappF_x100 **Param2:** pi32_hfXtalNPcbParCappF_x100
Result	None
Description	Enables the High Frequency crystal oscillator LDO (voltage regulator) then sets up the Capacitors Banks according to the parameters provided by the user.

NOTE

This API does not enable the High Frequency crystal oscillator.

10.2.1.1 Param0: pi32_hfXtalIecLoadpF_x100

The Crystal Oscillator IEC Load capacitance, in pF x 100. For example:

- 6pF IEC equivalent Load Capacitance (which means 12pF on pin XIN and 12pF on pin XOUT) becomes 600.
- 10.2pF IEC Load Capacitance (which means 20.4pF on pin XIN and 20.4pF on pin XOUT) becomes 1020.

10.2.1.2 Param1: pi32_hfXtalPPcbParCappF_x100

PCB parasitic capacitance on pin XIN, in pF x 100. For example:

- 2pF parasitic capacitance becomes 200.
- 0.2pF parasitic capacitance becomes 20.

10.2.1.3 Param2: pi32_hfXtalNPcbParCappF_x100

PCB parasitic capacitance on pin XOUT, in pF x 100. For example:

- 2pF parasitic capacitance becomes 200.
- 0.2pF parasitic capacitance becomes 20.

10.2.2 CLOCK_Xtal32khzCapabankTrim

This API function configures the 32 kHz Crystal Oscillator Capacitor Bank.

Table 34. CLOCK_Xtal32khzCapabankTrim API routine

Routine	CLOCK_Xtal32khzCapabankTrim
SKD prototype	void CLOCK_Xtal32khzCapabankTrim (int32_t pi32_32kfXtalIecLoadpF_x100, int32_t pi32_32kfXtalPPcbParCappF_x100, int32_t pi32_32kfXtalNPcbParCappF_x100)
Input parameter	**Param0:** pi32_32kfXtalIecLoadpF_x100 **Param1:** pi32_32kfXtalPPcbParCappF_x100

Table continues on the next page...

Table 34. CLOCK_Xtal32khzCapabankTrim API routine (continued)

Routine	CLOCK_Xtal32khzCapabankTrim
	Param2: pi32_32kfXtalNPcbParCappF_x100
Result	None
Description	Sets up the Capacitors Banks according to the parameters provided by the user.

Remark: This API does not enable the 32 kHz Crystal Oscillator.

10.2.2.1 Param0: pi32_32kfXtallecLoadF_x100

The Crystal Oscillator IEC Load capacitance, in pF x 100. For example:

- 6pF IEC equivalent Load Capacitance (which means 12pF on pin XIN and 12pF on pin XOUT) becomes 600.
- 10.2pF IEC Load Capacitance (which means 5.1pF on pin XIN and 5.1pF on pin XOUT) becomes 1020.

10.2.2.2 Param1: pi32_32kfXtalPPcbParCappF_x100

PCB parasitic capacitance on pin XIN, in pF x 100. For example:

- 2pF parasitic capacitance becomes 200.
- 0.2pF parasitic capacitance becomes 20.

10.2.2.3 Param2: pi32_32kfXtalNPcbParCappF_x100

PCB parasitic capacitance on pin XOUT, in pF x 100. For example:

- 2pF parasitic capacitance becomes 200.
- 0.2pF parasitic capacitance becomes 20.

10.3 Programming examples

10.3.1 High Frequency Crystal Oscillator

The three variables below are used in all subsequent examples:

int32_t i32_iec_cl_pf; /* IEC equivalent Capacitance Load, in pF */

int32_i i32_xin_pcb_para_pf; /* PCB parasitic capacitance on XIN pin, in pF */

int32_i i32_xout_pcb_para_pf; /* PCB parasitic capacitance on XOUT pin, in pF */

10.3.1.1 Example 1: 8pF IEC Capacitance Load, 2pF PCB parasitic capacitance on XIN pin, 3pF PCB parasitic capacitance on XOUT pin.

i32_iec_cl_pf = 8; /* IEC equivalent Capacitance Load, in pF, which means 16 pF on XIN pin and 16 pF on XOUT pin */

i32_xin_pcb_para_pf = 2; /* PCB parasitic capacitance on XIN pin, in pF */

i32_xout_pcb_para_pf = 3; /* PCB parasitic capacitance on XOUT pin, in pF */

Computation of the required Capacitance Load:

MAXIMUM(2*i32_iec_cl_pf - i32_xin_pcb_para_pf, 2*i32_iec_cl_pf - i32_xout_pcb_para_pf) = MAXIMUM(2*8 - 2, 2*8 - 3) = MAXIMUM(14,13) = 14 pF

14 pF is below 20 pF (10 pF equivalent IEC); therefore, there is no need to add some capacitance on PCB.

Configuration of the internal capa banks:

/*

* - Setup High Frequency Crystal Oscillator Capacitor Bank and enable High Frequency Crystal Oscillator LDO (Voltage regulator).

*/

CLOCK_XtalHfCapabankTrim(**8** * 100, **2** * 100, **3** * 100);

/*

* - Enable High Frequency Crystal Oscillator

*/

PMC->PDRUNCFGCLR0 = **PMC_PDRUNCFG0_PDEN_XTAL32M_MASK**;

/*

* - Enable High Frequency Crystal Oscillator output towards USB High Speed PLL

*/

ANACTRL->XO32M_CTRL = ANACTRL->XO32M_CTRL | **ANACTRL_XO32M_CTRL_ENABLE_PLL_USB_OUT_MASK**;

/*

* - (If required) Enable High Frequency Crystal Oscillator output for use as System Clock.

*/

ANACTRL->XO32M_CTRL = ANACTRL->XO32M_CTRL | **ANACTRL_XO32M_CTRL_ENABLE_SYSTEM_CLK_OUT_MASK**;

10.3.1.2 Example 2: 15pF IEC Capacitance Load, 2pF PCB parasitic capacitance on XIN pin, 2pF PCB parasitic capacitance on XOUT pin.

i32_iec_cl_pf = 15; /* IEC equivalent Capacitance Load, in pF, which means 30 pF on XIN pin and 30 pF on XOUT pin */

i32_xin_pcb_para_pf = 2; /* PCB parasitic capacitance on XIN pin, in pF */

i32_xout_pcb_para_pf = 2; /* PCB parasitic capacitance on XOUT pin, in pF */

Computation of the required Capacitance Load:

MAXIMUM(2*i32_iec_cl_pf - i32_xin_pcb_para_pf, 2*i32_iec_cl_pf - i32_xout_pcb_para_pf) = MAXIMUM(2*15 - 2, 2*15 - 2) = MAXIMUM(28,28) = 28 pF

28 pF is above 20 pF (10 pF equivalent IEC); therefore, some extra capacitance on PCB are required.

Because some extra capacitance is required on PCB, it is recommended to configure the internal capa bank *as if an **8pF** Load Capacitance IEC equivalent (16pF on both XIN and XOUT pins) was required*, which means:

2*i32_iec_cl_pf - i32_xin_pcb_para_pf must be equal to 16pF.

=> 2*i32_iec_cl_pf = 16 + i32_xin_pcb_para_pf = 16 + 2 = 18 pF (9pF Load Capacitance load IEC equivalent)

=> i32_iec_cl_pf = 18 / 2

=> i32_iec_cl_pf = 9.

Therefore, only 30 pF – 18 pF = 12 pF Load Capacitance is required on the PCB for Xin and XOUT pins.

Configuration of the internal capa banks:

/*

* - Setup High Frequency Crystal Oscillator Capacitor Bank and enable High Frequency Crystal Oscillator LDO (voltage regulator).

*/

```
CLOCK_XtalHfCapabankTrim(9 * 100, 2 * 100, 2 * 100);
/*
* - Enable High Frequency Crystal Oscillator
*/
PMC->PDRUNCFGCLR0 = PMC_PDRUNCFG0_PDEN_XTAL32M_MASK;
/*
* - Enable High Frequency Crystal Oscillator output towards USB High Speed PLL
*/
ANACTRL->XO32M_CTRL = ANACTRL->XO32M_CTRL | ANACTRL_XO32M_CTRL_ENABLE_PLL_USB_OUT_MASK;
/*
* - (If required) Enable High Frequency Crystal Oscillator output for use as System Clock.
*/
ANACTRL->XO32M_CTRL = ANACTRL->XO32M_CTRL | ANACTRL_XO32M_CTRL_ENABLE_SYSTEM_CLK_OUT_MASK;
```

10.3.2 32 kHz Crystal Oscillator

The three variables below are used in all subsequent examples:

```
int32_t i32_iec_cl_pf; /* IEC equivalent Capacitance Load, in pF */
int32_i i32_xin_pcb_para_pf; /* PCB parasitic capacitance on XIN pin, in pF */
int32_i i32_xout_pcb_para_pf; /* PCB parasitic capacitance on XOUT pin, in pF */
```

10.3.2.1 Example 1: 8pF IEC Capacitance Load, 2pF PCB parasitic capacitance on XIN pin, 3pF PCB parasitic capacitance on XOUT pin.

```
i32_iec_cl_pf = 8; /* IEC equivalent Capacitance Load, in pF, which means 16 pF on XIN pin and 16 pF on XOUT pin */
i32_xin_pcb_para_pf = 2; /* PCB parasitic capacitance on XIN pin, in pF */
i32_xout_pcb_para_pf = 3; /* PCB parasitic capacitance on XOUT pin, in pF */
```

Computation of the required Capacitance Load:

MAXIMUM(2*i32_iec_cl_pf - i32_xin_pcb_para_pf, 2*i32_iec_cl_pf - i32_xout_pcb_para_pf) = MAXIMUM(2*8 - 2, 2*8 - 3) = MAXIMUM(14,13) = 14 pF

14 pF is below 20 pF (10 pF equivalent IEC); therefore, there is no need to add some capacitance on PCB.

Configuration of the internal capa banks:

```
/*
* - Setup 32 kHz Crystal Oscillator Capacitor Bank
*/
CLOCK_Xtal32khzCapabankTrim(8 * 100, 2 * 100, 3 * 100);
/*
* - Enable 32 kHz Crystal Oscillator
*/
PMC->PDRUNCFGCLR0 = PMC_PDRUNCFG0_PDEN_XTAL32K_MASK;
/*
* - Select 32 kHz Crystal Oscillator (instead of 32 kHz Free Running Oscillator)
```

*/

PMC->RTCOSC32K = PMC->RTCOSC32K | PMC_RTCOSC32K_SEL_MASK;

10.3.2.2 Example 2: 15pF IEC Capacitance Load, 2pF PCB parasitic capacitance on XIN pin, 2pF PCB parasitic capacitance on XOUT pin.

i32_iec_cl_pf = 15; /* IEC equivalent Capacitance Load, in pF, which means 30 pF on XIN pin and 30 pF on XOUT pin */

i32_xin_pcb_para_pf = 2; /* PCB parasitic capacitance on XIN pin, in pF */

i32_xout_pcb_para_pf = 2; /* PCB parasitic capacitance on XOUT pin, in pF */

Computation of the required Capacitance Load:

MAXIMUM(2*i32_iec_cl_pf - i32_xin_pcb_para_pf, 2*i32_iec_cl_pf - i32_xout_pcb_para_pf) = MAXIMUM(2*15 - 2, 2*15 - 2) = MAXIMUM(28,28) = 28 pF

28 pF is above 20 pF (10 pF equivalent IEC); therefore, some extra capacitance on PCB are required.

Because some extra capacitance is required on PCB, it is recommended to configure the internal capa bank *as if an 8pF Load Capacitance IEC equivalent (16pF on both XIN and XOUT pins) was required*, which means:

2*i32_iec_cl_pf - i32_xin_pcb_para_pf must be equal to 16pF.

=> 2*i32_iec_cl_pf = 16 + i32_xin_pcb_para_pf = 16 + 2 = 18 pF (9pF Load Capacitance load IEC equivalent)

=> i32_iec_cl_pf = 18 / 2

=> i32_iec_cl_pf = 9.

Therefore, only 30 pF – 18 pF = 12 pF Load Capacitance is required on the PCB for Xin and XOUT pins.

Configuration of the internal capa banks:

/*

* - Setup 32 kHz Crystal Oscillator Capacitor Bank

*/

CLOCK_Xtal32khzCapabankTrim(9 * 100, 2 * 100, 2 * 100);

/*

* - Enable 32 kHz Crystal Oscillator

*/

PMC->PDRUNCFGCLR0 = PMC_PDRUNCFG0_PDEN_XTAL32K_MASK;

/*

* - Select 32 kHz Crystal Oscillator (instead of 32 kHz Free Running Oscillator)

*/

PMC->RTCOSC32K = PMC->RTCOSC32K | PMC_RTCOSC32K_SEL_MASK;

Chapter 11
Independent Real Time Clock (RTC)

11.1 Chip-specific RTC information

Table 35. Reference links to related information

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

11.1.1 Module instances

This device has one instance of the RTC module, RTC. The Brown-Out Detect (BOD) monitors voltage levels of the supply rails, and triggers isolation for this RTC.

11.1.2 Chip-specific RTC block diagram

This RTC includes these features:

- A 15-bit, 32 kHz sub-second counter
- One 16-bit general purpose register, GP_DATA_REG. The contents of this register are powered and retained by the VBAT domain.

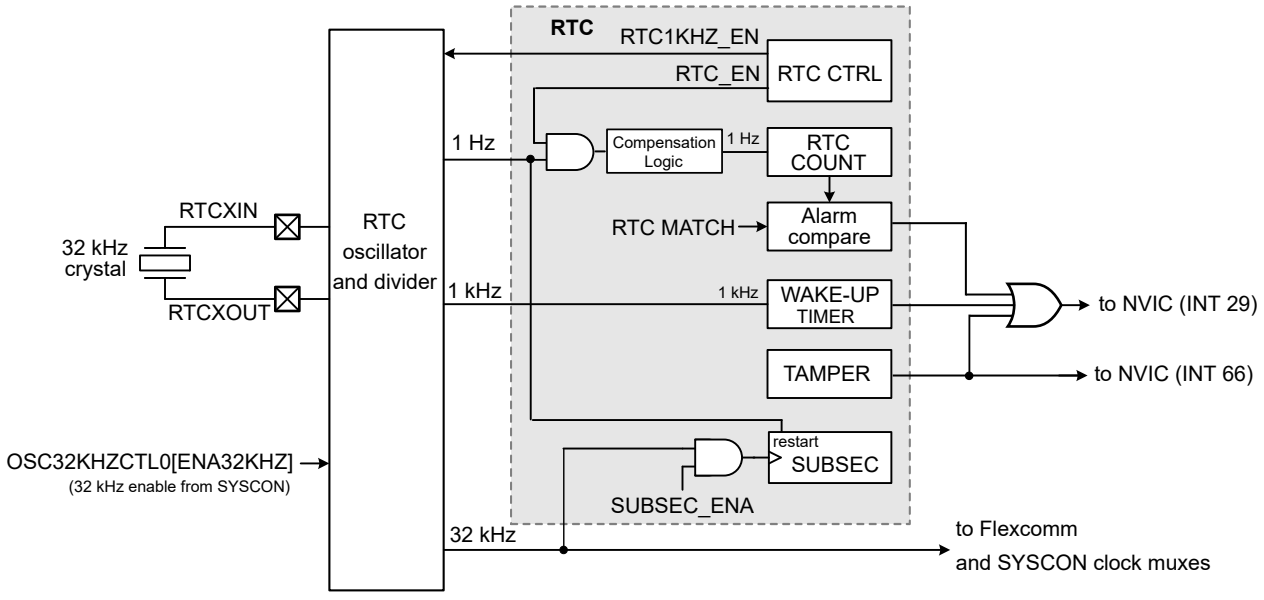


Figure 15. Chip-specific RTC block diagram

11.1.3 Chip-specific RTC implementation

- All bits in the 16-bit register GP_DATA_REG are available for the application. None of these bits are used by the MCU. These bits are powered and retained by the VBAT domain.
- TAMPER_SCR[8] resets to 1 to indicate battery removal when MCU is powered OFF. After this bit is cleared by software, it will also indicate the status of the TAMPER0 external pin.
- This RTC implements 10 sampling timer interrupts, ranging from 1 Hz to 512 Hz.
- The Tamper FIFO read with TAMPER_QUEUE register has a depth of 8.

11.1.4 Chip-specific RTC initialization

- Enable the clock to the RTC register interface and peripheral clock with the AHBCLKCTRL0 register in SYSCON.
- Enable the RTC software reset in the RTC CTRL register. The RTC is reset only by initial power-up of the device or when an RTC software reset is applied, RTC is not initialized by other system resets.
- Enable the interrupts to the NVIC for the RTC_WAKE and RTC_ALARM functions. See the Nested Vectored Interrupt Controller (NVIC).
- Enable the RTC interrupts for waking up the device from Deep-sleep mode by enabling the interrupts in the STARTEN1 register of SYSCTL and also in the NVIC.
- Enable the RTC interrupts for waking up from Deep Power-down mode using the RTC CTRL register.

If enabled, the RTC and RTC oscillator continue running in all low power modes as long as power is supplied to the device. Therefore, the 32 kHz output is always available to be enabled for SYSCTL clock generation. Once enabled, the 32 kHz clock can be selected for the system clock or be observed through the CLKOUT pin. The 1 Hz output is enabled in the RTC CTRL[RTC_EN] bit. Once the 1 Hz output is enabled, the 1 kHz output for the high-resolution wake-up timer can be enabled in the RTC CTRL[RTC1KHZ_EN] bit.

If the 32 kHz output of the RTC is used by another part of the system, enable it via the RTC CTRL[RTC_EN].

NOTE

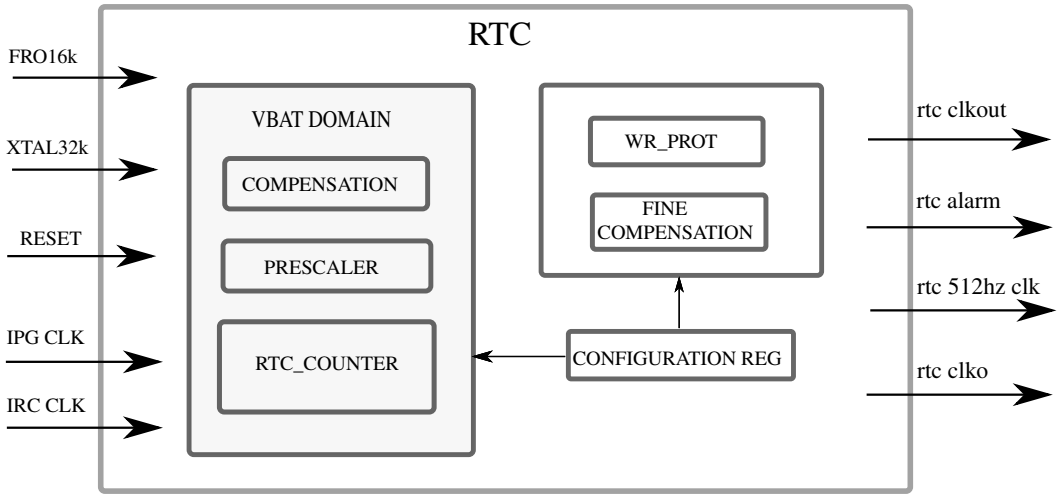
By default, the functional clock FRO32K is not enabled initially and the RTC is held in reset and cannot be accessed via APB.

11.2 Overview

This block is a low power module that provides time keeping and calendaring functions and additionally provides protection against tampering (external or internal tamper events), protection against spurious memory/register updates and battery operation. It can additionally compensate the 1 Hz clock against variations in 32kHz clock in oscillator due to crystal or temperature. A Standby RAM is provided if the CPU wants to store any data that has to be retained when in battery operation mode.

11.2.1 Block diagram

RTC block diagram



11.2.2 Features

This block supports the following features:

- Designed for low power
  - Time and Date counters are rippled with respect to each other to prevent simultaneous toggling
- Basic Clock functions
  - Separate counters for Days, Hour, Minutes and Seconds
  - Calendaring support – Separate counters for Months, Year and Day of the Week
    - Automatic adjustment for Day Light Saving with user defined parameters
    - Automatic month and leap year adjustment
  - External clock support to run the counters in the case user wishes to provide externally compensated 1Hz clock.
- RTC utilizes 'local time' which implicitly contains the time zone offset
- Programmable alarm with interrupt. Alarm is output from RTC in case MCU wants to use it as a wake up event
- Periodic interrupts (Sampling Timer Interrupts)
- Hardware Compensation to compensate 1 Hz clock (to the counters) against frequency variations in oscillator clock due to temperature or crystal characteristics. Correction factor calculated by firmware. (Programmable correction factor).
- 16-bit CPU register programming interface with protection against run-away code
- Reset to the RTC block is generated when battery supply is removed and powered up later.
- Battery operation (Standby Mode) ensures seamless RTC operation when CPU power is removed
- Option to output the buffered 32.768 kHz clock or the compensated 1 Hz clock.
- 32 Bytes of Battery Backed Up RAM
- Enhanced Tamper Detection
  - Tamper Detection to detect illegal access into the system.

- Tamper sources can be individually enabled/disabled and polarity can be controlled
- External tamper sources are filtered for noise & glitches by the RTC. Internal tamper events are not filtered.
- A tamper queue is implemented in the design to store the tamper type and time stamp (excluding the year value) of the tampers.
- Configurable width of noise or glitch pulses that can be filtered out
  - Minimum 488 µs to 2 s depending on the tamper filter clock that is chosen.

11.3 Functional Description

11.3.1 Modes of Operation

This section describes the RTC modes of operation.

11.3.1.1 Wait Mode

In Wait mode, the RTC is fully operational. Since the RTC runs off a 32.768 kHz clock, which is not gated in Wait mode; RTC's time keeping functions and other functions that depend on the 32.768 kHz clock, continue to operate independent of CPU. Only the register block's clock is gated. No register contents are lost.

11.3.1.2 Stop Mode

In Stop mode, the RTC is fully operational and behaves in the same way as in Wait mode.

11.3.2 Design Overview

The RTC block provides basic time keeping functions through seconds, minutes, hours counters, and calendaring functions via date, day-of-week, month and year counters; along with automatic adjustments for leap year and day light saving. Reading these counters indicates the current date and time and writing to these registers sets the date and time as provided by the user.

The alarm is set for specific hour, minute and second. When the time counters match the configured alarm hour, minute and second settings, the alarm is set and an interrupt to CPU is generated, if the alarm interrupt is enabled. The alarm can additionally be configured to match days, months and year to generate the alarm interrupt. The alarm signal has been brought out on an MCU pin to allow wake up or control of external devices on board.

RTC module also provides sampling timer interrupts.

A frequency compensation module is integrated into the RTC to correct any error in the 1 Hz clock due to variations in the 32.768 kHz clock which could be caused by crystal inaccuracy, board variations or change in temperature. The combined compensation value for both crystal and temperature variation is determined by software and correction is done in hardware.

The registers in the RTC module are configured via the CPU register programming interface. A protection mechanism is built in to the RTC to protect its registers against spurious writes by any run-away code. The protection mechanism requires the CPU to write a specific sequence of codes to the STATUS[7:6] bits to allows write access to the registers .On completing the update of registers the CPU should write "10" to STATUS[7:6] bits to enable the write protection. After unlocking the registers, the CPU has a window of 2 seconds for updating the register space. On power on reset a window of 15 seconds is allowed for the CPU to configure the RTC after which the registers are locked. Any updates beyond the unlock window would require the CPU to unlock the registers, again.

The RTC battery supply maintains normal RTC functionality when the MCU power (VDD) is removed. The battery supply allows RTC to keep functioning in case CPU is completely turned off. Reset to the RTC block is generated only when the battery supply is removed and powered up again.

RTC can detect intrusion via its tamper detection and logging mechanism. RTC supports "4 tamper events all individually configurable and any change of state will indicate a tamper that will get stored in the RTC tamper status and control register. Removal of battery after power up is also considered as a tamper event. The external tamper inputs pass through a low pass filter (digital) to prevent accidental assertion by noise or glitch pulses. The duration of pulses to be filtered out is configurable. Any tamper detected will cause an interrupt to the CPU. RTC has a FIFO based queue structure implemented in the design to support

the logging of tamper time stamp entries. Tamper detection logic and its interrupt are enabled on reset. Once a tamper is detected and event stored, a new event for the same tamper will not be stored until the CPU acknowledges the tamper status.

For detailed description on the complete functionality of RTC, refer to Functional Description.

11.3.3 Clocking

This section describes clocks of the RTC module.

The RTC has a 32kHz clock input from the oscillator and a 16KHz clock input from the Fro, which clock to use is selected by CTRL[regsp_clk_sel]. The selected clock is output to the peripheral, and be divided to generate a 512Hz clock and a 1Hz clock. The 512Hz clock is output to the peripheral, and the 1Hz clock is used as the input of the counter.

There is a input clk named rtc_irc_clk, which is Mhz clock used to fine compensate the 1 Hz clock against variations in 32 kHz clock in oscillator due to crystal or temperature.

RTC has one system bus clock used to synchronize the oscillator 32KHz/16KHz clock to the ipg_clk domain.

Register bit CTRL[CLKOUT] is used to configure either the selected clock(32KHz/16KHz) or 1 Hz clock to be output from the SOC for use outside RTC. Register bit CTRL[CLKO] s used to configure whether the selected clock is output to other peripherals.

11.3.4 Interrupts

This section describes all the interrupts that the RTC module generates.

If IER[ALM_IE]=1, alarm interrupt be enabled. Alarm interrupt is asserted when the alarm value programmed matches the counter values, and the alarm Interrupt Status bit ALM_IS will be set. The status bit is cleared by writing a value of 1, which also clears the interrupt. The status register is also cleared on software reset except for the tamper status which remains unaffected . The counters matched for the alarm interrupt are selected based on the Alarm Type set in CTRL[3:2] bits.

There are 8 periodic interrupts (Sampling Timer Interrupts). When the clock generation counter counts to the corresponding value, the corresponding bit in the status register will be set. The status bits are cleared by writing a value of 1,and can also be cleared by software reset.

11.3.5 Time and Calendaring Functions

RTC performs and controls all chronological functions as mentioned below:

- Implements all counters for date and time and their related control logic
- Leap Year calculation and adjusting the day count accordingly
- Increment/Decrement of counters for adjustment of leap seconds
- Tracking the number of days in a month
- Automatic Daylight adjustment for time
- Alarm generation with selectable matching of different counters

Dynamic modifications to the date counters are done based on a) Leap Year b) Month and c) Daylight Saving. Additionally, the user software can add or subtract a second to take care of Leap Seconds Adjustment. All changes are hardware controlled and triggered by software. A leap year is defined as a year in which the year value is divisible by 4 and 400 and will have an extra day in February. Daylight savings are done in different regions of the world to shift the local time according to summer or winters. Time is shifted at a pre-defined time decided by the regional conditions and programmed by the user software into the RTC and RTC automatically adjusts the time using hardware alarms for daylight saving. Day counter is also adjusted for months having 28/29/30/31 days.

Alarm generation is done by the RTC block. The matching of counters is controlled by ALM_MATCH bits inside the control register to give various alarm options of daily, monthly, yearly or one-time alarms.

NOTE

Setting an hour alarm value that coincides with the Daylight Saving Start Hour value will not generate an alarm as when Daylight Saving comes into effect, the hour counter is incremented by 2 instead or normal 1. For example, setting ALM_HOURMIN = 0x1500 (hour alarm value = 15 or 3 PM) and DST_HOUR = 0x1411 (DST Start Hour value = 14 or 2 PM), will cause the hour counter to go from 14 to 16. The hour counter will not be 15 and hence the alarm which was set for 15 (or 3 PM) will not trigger. User software must take into account Daylight Saving changes when setting alarm values.

NOTE

The user software must program an alarm time equal to the daylight saving end time that is fallback time and when the alarm interrupt comes, the user software must clear the DSTEN bit in control register, else correct operation might not occur.

11.3.6 RTC Compensation Logic

The Compensation Logic provides an accurate and wide range of compensation which is suitable for many crystals, and can correct a wide range crystal offsets with offsets being as low as 0.119 PPM. The same hardware logic supports both temperature compensation and frequency compensation.

There are two important components in the compensation logic: Coarse Compensation and Fine Compensation. The coarse compensation provides accurate clock to RTC's internal time and date counters while fine compensation generate an accurate 1 Hz clock output (via MCU pin) with high resolution clock edge placement (up to 0.88 ppm) and near 50% duty cycle.

Enabling the required compensation logic

Coarse compensation is enabled by writing to CTRL[COMP_EN] and Fine compensation is enabled by writing to CTRL[FINEEN]. However, when FINEEN = 1, COMP_EN must be equal to 1. Setting both bits to 0 disables all compensation and this is the default state out of reset.

In addition, compensation parameters need to be provided to allow required compensation logic to run. The parameters are provided by writing to the COMPEN register.

NOTE

If CTRL[FINEEN] = 0, then before changing COMPEN for the first time, disable the compensation using CTRL[COMP_EN]. Successive changes can be done without disabling compensation. If FINEEN = 1, then there is an option to either clear previously accumulated fractional compensation value by disabling compensation from the CTRL register, program new values, and re-enable compensation and start a fresh or to overwrite the current values which leads to adding of previously accumulated fractional compensation value to programmed fractional compensation value. The integer part is always overwritten.

Compensation Parameters

Compensation logic requires the user software to provide compensation parameters in the COMPEN register in order to generate accurate 1 Hz clock. These parameters are defined depending on the type of compensation enabled.

The compensation parameters (when FINEEN = 0) are defined as:

- **Compensation/Correction Value:** Compensation/Correction Value is a 2's complement value by which the 1 Hz Clock is modified (during its generation) by either adding or removing RTC Oscillator clock cycles.
- **Compensation Interval:** Compensation Interval is the duration in seconds over which the correction is applied. This is the time in which the addition or removal of 32.768 kHz clock cycles is done thereby ensuring that the compensation interval is close to the interval obtained with an ideal 1 Hz clock.

The compensation parameters (when FINEEN = 1) are defined as:

- **Integral Compensation Value:** This is a 2's complement value of the integer part of correction or compensation value that has to be adjusted in every 1 second period. This value is expressed in terms of number of clock cycles of the RTC Oscillator clock.

- **Fraction Compensation Value:** This is the fractional part of the correction or compensation value that has to be adjusted. This value is expressed as number of clock cycles of a fixed 4.194304 MHz clock that have to be added. This value is always a positive number.

NOTE

When FINEEN = 1, the compensation interval is by default set to "1" indicating that the compensation will be done at every second.

Coarse Compensation Logic

The coarse compensation logic provides the accurate 1 Hz clock pulses to the time and date counters and the coarse 1 Hz clock to the fine compensation logic.

When FINEEN = 0: RTC compensates the clock over the provided compensation interval. The addition or removal of clocks is done in a single 1 Hz period, leaving the other 1 Hz periods of the compensation interval same. This addition or removal of RTC Oscillator clock cycles adjusts the 1 Hz clock in a way to keep the overall compensation interval time close to the same interval being measured with an ideal clock.

When FINEN = 1: The integral part of compensation value (or correction value) is added or removed every 1 Hz period while the fraction part is accumulated and adjusted when accumulated value equals 1 RTC Oscillator clock cycle period.

To perform crystal offset compensation, the user software computes the compensation parameters external to RTC and using the details about crystal characteristics (ageing, drift, etc), computes the correction factor and programs it in the two's complement format in the Compensation Register (COMPEN). Based on the values written in the COMPEN register, the compensation is performed.

To perform temperature compensation the user software can maintain a look-up-table in its memory which lists the change in frequency of 32.768 kHz crystal clock for each degree change in temperature. The CPU wakes up periodically to measure the external temperature via a temperature sensor connected to an A/D converter. The user software uses the look-up-table to determine the compensation factor and writes the value to be compensated (in terms of number of 32.768 kHz clock cycles in 2's complement format) in the Compensation Register (COMPEN). Based on the new values written the compensation logic adjusts the clock from the next compensation interval.

The user software must compute a common compensation factor if it detects a variation in 32.768 kHz clock due to both temperature and crystal characteristics.

Vital Statistics

- Range of Compensation Interval: 1 second to 255 seconds. If FINEEN = 0, programming a 0 disables compensation. If FINEEN = 1, this interval is always 1 second.
- Range of Compensation: -128 to +127 (number of 32.768 kHz clock cycles)
- Selection Criteria: Compensation is done only when enabled by user software. User software can disable compensation by programming a zero compensation interval or setting CTRL[COMPEN] = 0.

Fine Compensation Logic

The fine compensation logic takes the coarse 1 Hz clock (from coarse compensation block) and generates accurate 1 Hz clock output with accurate clock edge placement and near 50% duty cycle.

The fine compensation runs when FINEEN = 1 and uses the MCU's IRC clock to adjust the fractional part of the compensation value every 1 Hz period. This provides an accurate edge placement on 1 Hz clock.

The fine compensation module automatically adjusts the fractional compensation value for any variation in the IRC clock.

NOTE

Since the IRC clock is generated on MCU, this clock will get disabled when MCU power is OFF or in certain low power modes. In this case, the coarse 1 Hz clock is output from the MCU. The coarse 1 Hz clock is not a 50% duty cycle clock.

11.3.7 Write Protection Mechanism

This logic protects the RTC Registers from any spurious updates that can happen from a run-away code. The logic monitors the values written to STATUS[WE]. By default, unconditional write access is allowed to these bits only. For writing the locking and unlocking sequence, 8-bit access should be given on STATUS[7:0].

To enable write protection, "10" is written on these bits. To disable write protection, the sequence "00 – 01 – 11 – 10" is written onto these bits.

After a power on reset, the write-protect mechanism is disabled, allowing the user software to configure the RTC block. Once configuration is complete, the user code should enable the write protection mode. If this is not done by the user software, the registers are put into write protect mode 15 seconds after power on. In case the write protect mode is unlocked to update registers, then the write protect mode is enabled automatically 2 seconds after unlock, if not locked by user software.

Any write access made to the registers when write protection is enabled (i.e. registers in locked mode) will cause the transfer error signal to be asserted. Reads are allowed at all times.

NOTE

1. Always check STATUS[WRITE_PROT_EN] after running unlocking and locking sequence and re-run the sequence if STATUS[WRITE_ PROT_EN] is not in the required state.
2. Two consecutive unlocking sequences will lock the registers. Once unlocked, running an unlock sequence before the timeout will lock RTC registers.

11.3.8 Tamper Detection and Logging

A mechanism has been provided in RTC to detect and log any intrusion made to the system. The tamper logic supports up to 4 tamper signal inputs which can be used for detecting either internal tamper events (i.e. battery removal, etc) or external tamper events (i.e. off chip tamper switches). RTC stores these individual and unique tamper events in the RTC tamper queue along with the time stamp of the tamper event and generates a common interrupt when any tamper event occurs.

The tamper queue does not store the year field as this value can be read from YEARMON. It is expected that the queue will be read at least once in a year.

Internal Tamper Events: Battery removal when MCU is powered OFF

The detection of battery removal during system power off is done using a flop that is asserted on power-on reset. Since there is no difference between a proper shutdown and this tamper condition, this is considered as a tamper always. The firmware needs to differentiate between the tamper condition and normal power up. One way is to ignore this tamper interrupt when the SoC or application is in Service mode and simply reset the tamper interrupt. For other conditions it will be taken as a tamper.

Internal Tamper Events: Battery removal when MCU is powered ON

The analog circuitry monitoring the battery voltage indicates when the battery voltage is removed. This signal is used by the tamper circuit to indicate a tamper provided MCU power is ON. A flop will be present that will be cleared by CPU during calibration. Any change of state on this signal will be detected as a tamper.

Internal Tamper Events: SoC internal security violations

There are several events that a SoC detects as a security violation depending on the security architecture of the SoC. The RTC tamper logic can enhance the SoC security by providing a logging mechanism for these events. The RTC stores each tamper in a status bit along with the time stamp of latest tamper event.

External Tamper Events: Off-Chip Tamper Indication

An external off-chip tamper switch is also used to monitor tampering external to the SoC. For example, a signal can be used to indicate that the case housing the SoC/board is opened or not. Since these events are detected off chip, these tamper inputs are pre-condition in the SoC logic (i.e. conversion to digital or level shifting, etc) before being input to RTC.

External tamper switches are prone to noise and can cause false tamper indication if not filtered. Noise filtering is present in RTC for all external tamper inputs. The duration for which a tamper signal should be asserted to be indicated as tamper is programmable by user software in the register space. This duration is programmable to support a variety of tamper switches.

The filtered signals are then used to generate the tamper status and interrupt signals. The polarity of the tamper inputs can be configured to be active high or active low.

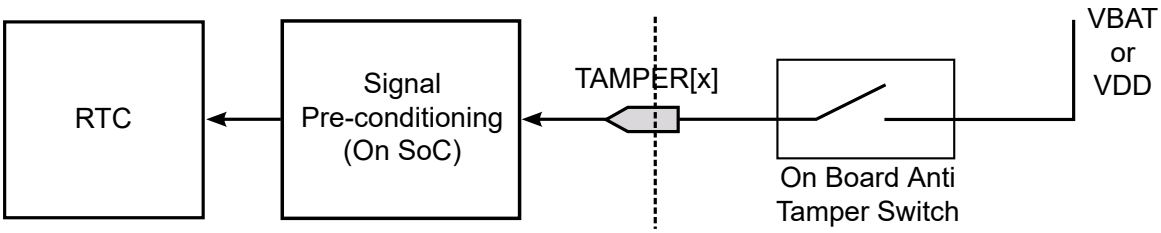


Figure 16. External Tamper Detection

Time Stamping:

RTC can record the date (day and month) and time (hour, minute, second) in the queue if the tamper is detected by RTC.

11.3.8.1 Tamper Detection Flow

A tamper queue is present in RTC to store tamper events (no priority to any tamper), every tamper entry stores the following information:-

- The tamper time-stamp i.e. month, day, hour, minute, and seconds.
- The tamper type or the tamper pin number.

The tamper data stored in the tamper queue has to be read via two read accesses.

- the first access gives the tamper_month, the tamper_day and the tamper_hour value.
- the second access gives the information about the tamper_type, tamper_minute, and tamper_second value.

The tampering logic stores only the first occurrence of tamper from a pin till that tamper status is acknowledged by the CPU. As soon as the CPU reads the queue entry it has to clear the status of the corresponding pin to enable logging of subsequent tampers from that pin. It may choose to disable that tamper bit if it does not want further tamper logging from that tamper pin.

The tamper queue also generates a full indication (both as a status bit and a maskable interrupt to the core) to the software, indicating the software that the maximum number of tampers that can be stored in the FIFO has been reached. The user has the capability to clear the tamper queue at one go by asserting TAMPER_QSCR[Q_CLEAR].

Following steps describe how tamper is logged in the registers and tamper queue, and how the software acknowledges the tamper status indication:

- Tamper Interrupt Enable bit is also asserted on POR and an interrupt indication to CPU.
- When CPU acknowledges the tamper interrupt by writing 1'b1 to the tamper interrupt status bit (TAMPER_SCR[8]), the tamper status bits are cleared.
- Internal tamper event (detected by SoC logic) are simply stored in their corresponding status bits and time stamp is captured.
- All tamper signals asserted externally are filtered inside the tamper block.
- Tamper status bit is asserted when the filter counter matches the programmed filter duration.
- Tamper status bit is asserted in the tamper status and control register irrespective the tamper control bit is enabled or not. Time stamp is also logged for the first occurrence of the tamper event. The first occurrence of each tamper along with time stamp get pushed into the tamper queue. Until current tamper status bit is acknowledged by CPU, next status and time stamp for that tamper is ignored.
- The tamper interrupt is asserted for the tamper status bits that have the corresponding interrupt enable bits set.
- Tamper interrupt status bit (in ISR) is cleared when all tamper status bits (in TAMPER_SCR) are cleared by writing 1 to them.
- The software must read the tamper queue entry first and then clear the corresponding status bit from the TAMPER_SCR register.

NOTE

When programing the tamper parameters such as filter duration, clock selection, polarity, etc, the user software must gate the 32k clock to RTC, program these parameters and then enable clock. While programming the parameters, the clock selection must be changed at the last. It is assumed that time and date will be set after this and tamper parameters will not be required to be changed again later.

11.3.9 RTC Isolation

To prevent leakage and erratic behavior, RTC isolates its CPU register programming interface and other control signals. There are two levels of isolation provided:

- CPU is in certain low power modes where the voltage monitor is disabled.
- When VDD < the voltage monitoring threshold, both read and writes to all registers are blocked. Isolation is enabled to prevent leakage from signals coming from the powered off domain of SoC.

The voltage threshold detection is done by an analog block (outside digital RTC) which monitors the voltage levels. Refer to the chip-specific section for more details on this voltage monitor.

11.4 External Signals

Table 36. External signals

Signal	Description	Direction
EXTAL32K, XTAL32K	Connections to an external 32.768 kHz crystal	Input
TAMPER*n*	External tamper pins act as an input or output depending upon whether passive or active	Input or Output
VBAT	External battery or standby power supply	Input
RTC_ALARMOUT	Alarm signal to allow wake up or control of external devices on board	Output

11.5 Memory Map and Registers

11.5.1 RTC register descriptions

The address of a register is the sum of a base address and an address offset. The base address is defined at the chip level. The address offset is defined at the module level.

NOTE

Initially, a 32 kHz clock is needed to initialize the RTC. This clock can later be gated while programming the RTC registers.

NOTE

The use of the fields of the General Purpose Data Register (GP_DATA_REG) is specific to the MCU. See the Chip-specific information section for a description of this register. Software reset has no effect on the contents of this register.

11.5.1.1 RTC memory map

RTC_TOP base address: 4002_C000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Year and Month Counters (YEARMON)	16	See section	0001
2	Days and Day-of-Week Counters (DAYS)	16	See section	0001
4	Hours and Minutes Counters (HOURMIN)	16	See section	0000
6	Seconds Counters (SECONDS)	16	See section	0000
8	Year and Months Alarm (ALM_YEARMON)	16	See section	0000
A	Days Alarm (ALM_DAYS)	16	See section	0000
C	Hours and Minutes Alarm (ALM_HOURMIN)	16	See section	0000
E	Seconds Alarm (ALM_SECONDS)	16	See section	0000
10	Control (CTRL)	16	See section	0000
12	Status (STATUS)	16	See section	0008
14	Interrupt Status (ISR)	16	See section	0001
16	Interrupt Enable (IER)	16	See section	0001
20	General Purpose Data (GP_DATA_REG)	16	RW	0000
22	Daylight Saving Hour (DST_HOUR)	16	See section	0000
24	Daylight Saving Month (DST_MONTH)	16	See section	0000
26	Daylight Saving Day (DST_DAY)	16	See section	0000
28	Compensation (COMPEN)	16	RW	0000
2E	Tamper Queue Status and Control (TAMPER_QSCR)	16	See section	0002
32	Tamper Status and Control (TAMPER_SCR)	16	See section	010F
34	Tamper 01 Filter Configuration (FILTER01_CFG)	16	RW	0000
36	Tamper 23 Filter Configuration (FILTER23_CFG)	16	RW	0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
40	Tamper Queue (TAMPER_QUEUE)	16	RO	0000
42	Control 2 (CTRL2)	16	See section	0000

11.5.1.1.1 Year and Month Counters (YEARMON)

This register stores the value of the month and year counters. The year field does not store the actual year value but the offset in years from a hardcoded BASE YEAR taken as the year 2112. This is a signed value and the ranges from -128 to +127. The software should program the offset from that base year into this register. For example, if the current year is 2007, then this will be represented in this register as 105 or 0x97. The actual year value can be found by adding the BASE YEAR and the offset in the YEARMON[15:8] register. Hence the range of year supported will be 1984 (2112 - 128) to 2239 (2112 + 127).

Hence for year calculation:

Actual Year = Base Year (for example, 2112) + Offset Year

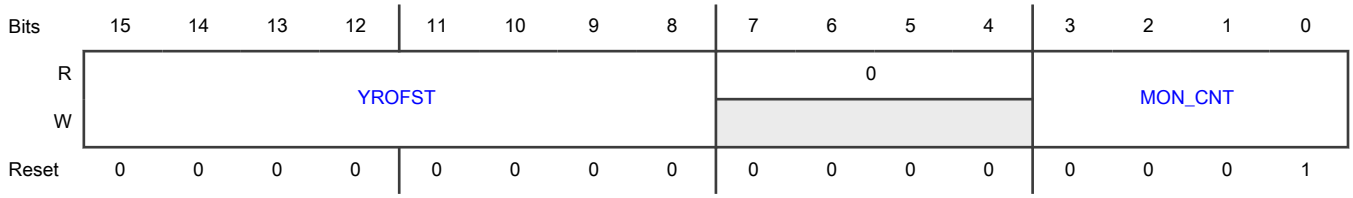
The month register stores the count value of the months register. Writing to this register loads the months counter with this new value. The valid values are mentioned in table below. Both month and year are unaffected on software reset.

User software should first determine the state of STATUS[INVAL_BIT] to determine that the counters are stable before their value can be read or changed. The assertion of STATUS[INVAL_BIT] ensures that no operation is done at the boundary of a second when counters change value.

Offset

Register	Offset
YEARMON	0h

Diagram



Fields

Field	Description
15-8 YROFST	Year Offset Count Value These bits indicate the offset in years from the base year (hard coded as 2112) and do not show the actual year value. This is a signed value. Valid values are -128 to 127.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	With the Base Year as 2112 and if the value of YEAR field is 0x10, the actual year will be 2112 + 16 = 2128.
7-4 —	Reserved
3-0 MON_CNT	Month Counter These bits give the value of the Months Counter . Valid Values are: 0000,1101,1110,1111 - Illegal Value 0001 - January 0010 - February 0011 - March 0100 - April 0101 - May 0110 - June 0111 - July 1000 - August 1001 - September 1010 - October 1011 - November 1100 - December

11.5.1.1.2 Days and Day-of-Week Counters (DAYS)

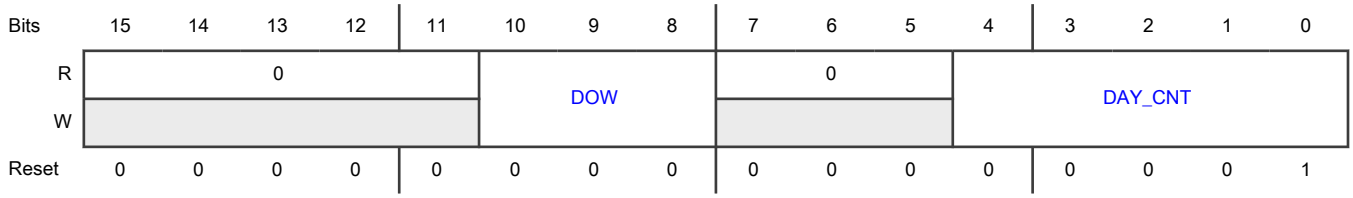
This read/write register shows the current value of the day-of-week counter and days counter. Reading this register returns the latest value of the counters. Writing to this register loads the value to the day-of-week and days counters and the counters continue to count from this new value. The day-of-week is not calculated automatically and should be written by CPU. This register is unaffected by software reset.

User software should first determine the state of STATUS[INVAL_BIT] to determine that the counters are stable before their value can be read or changed. The assertion of STATUS[INVAL_BIT] ensures that no operation is done at the boundary of a second when counters change value.

Offset

Register	Offset
DAYS	2h

Diagram



Fields

Field	Description
15-11 —	Reserved
10-8 DOW	Day of Week Counter Value 000 - Sunday 001 - Monday 010 - Tuesday 011 - Wednesday 100 - Thursday 101 - Friday 110 - Saturday 111 - Reserved
7-5 —	Reserved
4-0 DAY_CNT	Days Counter Value Valid values are 1 to 31.

11.5.1.1.3 Hours and Minutes Counters (HOURMIN)

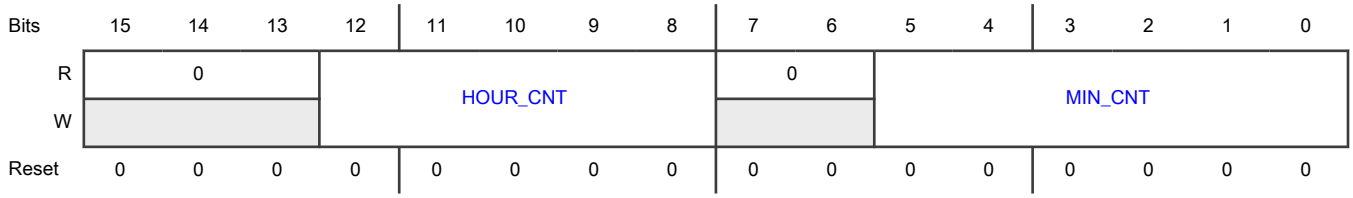
This register is used to program the hours and minutes counter. It can be read anytime to get the current value of the counters. Only power-on reset can reset this register. Hours counter can be set anything between 0 and 23. Minutes counter can be set anything between 0 and 59. This register is unaffected by software reset.

User software should first determine the state of the STATUS[INVAL_BIT] to determine that the counters are stable before their value can be read or changed. The assertion of STATUS[INVAL_BIT] ensures that no operation is done at the boundary of a second when counters change value.

Offset

Register	Offset
HOURMIN	4h

Diagram



Fields

Field	Description
15-13 —	Reserved
12-8 HOUR_CNT	Hours Counter Value Valid count values are 0 to 23.
7-6 —	Reserved
5-0 MIN_CNT	Minutes Counter Value Valid count values are 0 to 59.

11.5.1.1.4 Seconds Counters (SECONDS)

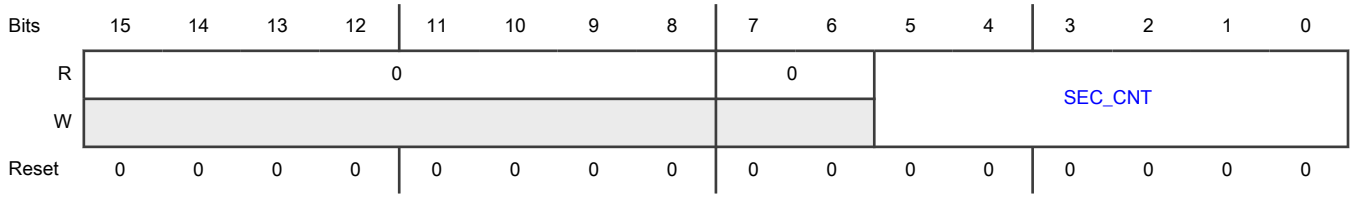
This register is used to program the seconds counter. It can be read anytime to get the current value of the counter. Only power-on-reset can reset this register. Seconds counter can be set anything between 0 and 59 both included. This register is unaffected by software reset.

User software should first determine the state of the STATUS[INVAL_BIT] to determine that the counters are stable before their value can be read or changed. The assertion of STATUS[INVAL_BIT] ensures that no operation is done at the boundary of a second when counters change value.

Offset

Register	Offset
SECONDS	6h

Diagram



Fields

Field	Description
15-8 —	Reserved
7-6 —	Reserved
5-0 SEC_CNT	Seconds Counter Value Valid count values are 0 to 59.

11.5.1.1.5 Year and Months Alarm (ALM_YEARMON)

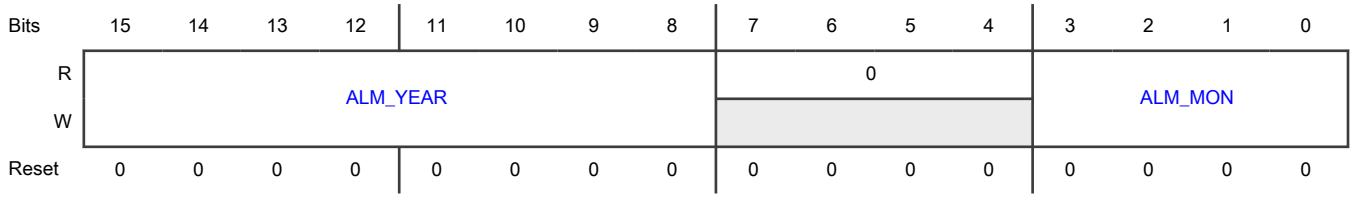
This register is used to configure the months and year setting of the alarm. The alarm setting can be read or written anytime. This register is reset to its default state on software reset. Alarm interrupt bit is set when all values of alarm seconds, minutes, hours, days, month, and year match their respective counter values.

User software can configure the type of alarm using the CTRL[ALM_MATCH].

Offset

Register	Offset
ALM_YEARMON	8h

Diagram



Fields

Field	Description
15-8	Year Value for Alarm

Table continues on the next page...

Table continued from the previous page...

Field	Description
ALM_YEAR	Same as Years Offset Value in Year and Month Counters (YEARMON) .
7-4 —	Reserved
3-0 ALM_MON	Months Value for Alarm Same as Months Counter Value in Year and Month Counters (YEARMON) .

11.5.1.1.6 Days Alarm (ALM_DAYS)

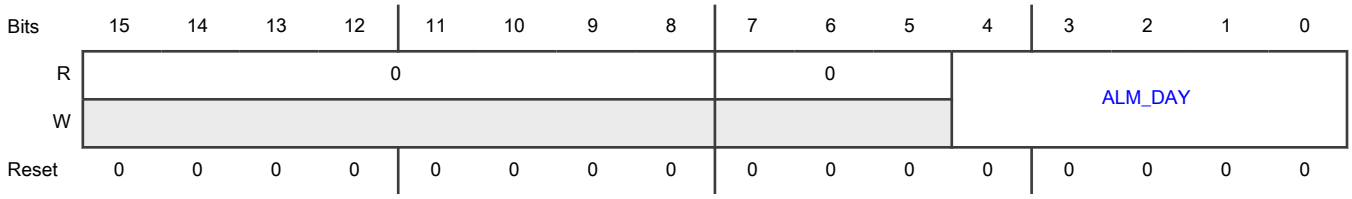
The days alarm register is used to configure the day setting of the alarm. The alarm setting can be read or written anytime. This register is reset to its default state on software reset. Alarm interrupt bit is set when all values of alarm seconds, minutes, hours, days, month, and year match their respective counter values.

User software can configure the type of alarm using CTRL[ALM_MATCH].

Offset

Register	Offset
ALM_DAYS	Ah

Diagram



Fields

Field	Description
15-8 —	Reserved
7-5 —	Reserved
4-0 ALM_DAY	Days Value for Alarm Same as Days Counter Value in Days and Day-of-Week Counters (DAYS) .

11.5.1.1.7 Hours and Minutes Alarm (ALM_HOURMIN)

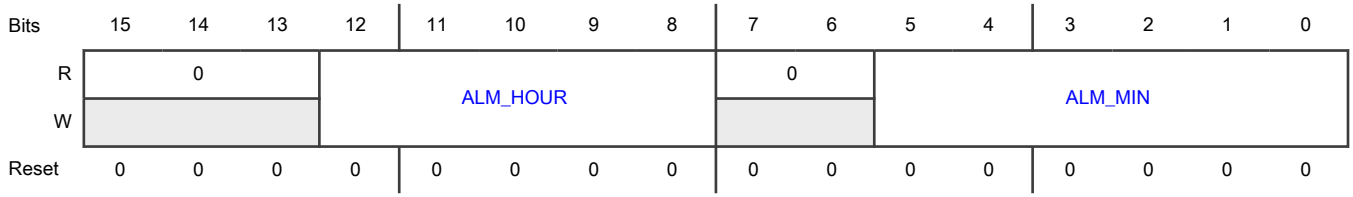
The hours and minutes alarm register is used to configure the hour and minute setting of the alarm. The alarm setting can be read or written anytime. This register is reset to default state on software reset.

User software can configure the type of alarm using CTRL[ALM_MATCH].

Offset

Register	Offset
ALM_HOURMIN	Ch

Diagram



Fields

Field	Description
15-13 —	Reserved
12-8 ALM_HOUR	Hours Value for Alarm Same as Hours Counter Value in Hours and Minutes Counters (HOURMIN) .
7-6 —	Reserved
5-0 ALM_MIN	Minutes Value for Alarm Same as Minutes Counter Value in Hours and Minutes Counters (HOURMIN) .

11.5.1.1.8 Seconds Alarm (ALM_SECONDS)

The seconds alarm register is used to configure the seconds setting of the alarm. The alarm setting can be read or written anytime. This register is reset to default value on software reset.

Bits 9:8 provide option to the user software to perform correction on seconds counter to compensate for the leap seconds. Write to these bits adds or subtracts 1 from the seconds counter and read returns zeros.

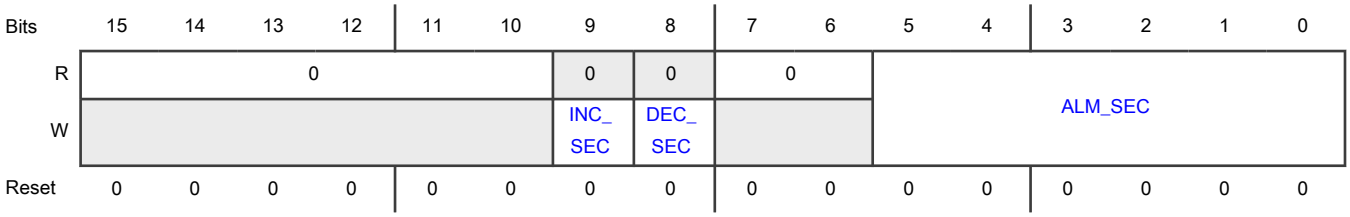
User software should first determine the state of STATUS[INVAL_BIT] to determine that the counters are stable before they can be incremented or decremented. The assertion of STATUS[INVAL_BIT] ensures that no operation is done at the boundary of a second when counters change value.

User software can configure the type of alarm using CTRL[ALM_MATCH].

Offset

Register	Offset
ALM_SECONDS	Eh

Diagram



Fields

Field	Description
15-10 —	Reserved
9 INC_SEC	Increment Seconds Counter by 1. This bit controls the increment of seconds counter in case the software wants to make corrections to the seconds counter to compensate for the leap seconds or to perform fine trimming of time when needed. Write to this bit increments the seconds counter and then the bit gets cleared on next posedge.
8 DEC_SEC	Decrement Seconds Counter by 1. This bit controls the decrement of seconds counter in case the software wants to make corrections to the seconds counter to compensate for the leap seconds or to perform fine trimming of time when needed. Write to this bit has decrements the seconds counter and then the bit gets cleared on next posedge of bus clock.
7-6 —	Reserved
5-0 ALM_SEC	Seconds Alarm Value Seconds Value for Alarm. Same as Seconds Counter Value in Seconds Counters (SECONDS) .

11.5.1.1.9 Control (CTRL)

This is the control register and governs all operations being done inside the RTC. This register is used to specify the software reset, daylight controls, and the type of alarm function needed.

Offset

Register	Offset
CTRL	10h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	CLKOUT		0		0	0	0	0	DST_EN	0	TIMER_S...	ALM_MATCH		COMP_EN	FINEEN
W								SWR								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
15 —	Reserved
14-13 CLKOUT	RTC Clock Output Selection Selects which clock to output from SoC for use outside RTC. 00 - No Output Clock 01 - Fine 1 Hz Clock with both precise edges 10 - 32.768 kHz Clock 11 - Coarse 1 Hz Clock with both precise edges
12-11 —	Reserved
10 —	Reserved
9 —	Reserved
8 SWR	Software Reset Self clearing bit. Asserting this field clears the contents of alarm, interrupt (status and enable except tamper interrupt enable bit) registers, STATUS[CMP_DONE], and STATUS[BUS_ERR] and has no effect on DST, calendaring, Standby time and tamper detect registers. 0 - Software Reset cleared 1 - Software Reset asserted
7 —	Reserved
6 DST_EN	Daylight Saving Enable The date and time for daylight saving changes are stored in the Daylight Saving Registers. These registers can be changed when this bit is 0. Once this bit is set, those registers cannot be changed and when time and date match the values in those registers, daylight adjustment will happen. To disable Daylight Saving function, this bit should be set to 0.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - Disabled. Daylight saving changes are not applied. Daylight saving registers can be modified. 1 - Enabled. Daylight saving changes are applied.
5 —	Reserved
4 TIMER_STB_MASK	Sampling Timer Clocks Mask 0 - Sampling clocks are not gated when in standby mode 1 - Sampling clocks are gated in Standby mode
3-2 ALM_MATCH	Alarm Match These bits define the type of alarm function. These bits select which time and calendar counters will be used for matching and generate an alarm. 00 - Only Seconds, Minutes, and Hours matched. 01 - Only Seconds, Minutes, Hours, and Days matched. 10 - Only Seconds, Minutes, Hours, Days, and Months matched. 11 - Only Seconds, Minutes, Hours, Days, Months, and Year (offset) matched.
1 COMP_EN	Compensation Enable **NOTE** If both the FINEEN and COMP_EN bits are meant to be set "1'b1", hardware will not let the COMP_EN bit to be written 1. 0 - Coarse Compensation is disabled. 1 - Coarse Compensation is enabled.
0 FINEEN	Fine Compensation Enable 0 - Fine compensation is disabled 1 - Fine compensation is enabled.

11.5.1.1.10 Status (STATUS)

This register indicates the status of various processes going inside the RTC. This register also helps the user software to read time or date register when their values are stable and not changing. Compensation Done bit and Bus Error bit get cleared by writing 1 to them. Software Reset resets the whole register (except the RST_SRC bit) to its default state.

All memory mapped registers are protected against spurious updates by the write protect mechanism. To unlock the registers , a specific pattern (as mentioned in above table) has to be written in the write enable bits (WE[1:0]) to enable or disable write protection. The WE[1:0] bits are the only bits that are freely writeable by user software. The write enable bits are self-clearing bits that always return zeros on read.

Offset

Register	Offset
STATUS	12h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				CMP_DONE	0	0	BUS_ERR	0		CMP_INT	0	RST_SRC	CPU_LOW...	WRITE_P...	INVAL_B...
W					W1C			W1C	WE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Fields

Field	Description
15-12 —	Reserved
11 CMP_DONE	Compensation Done CMP_DONE indicates that current compensation cycle is complete. The bit gets cleared by writing 1 to it. Done bit is asserted seven 32.768 kHz clock cycles before actual compensation interval completes so that back to back compensation can be enabled. 0 - Compensation busy or not enabled 1 - Compensation completed
10 —	Reserved
9 —	Reserved
8 BUS_ERR	Bus Error This bit indicates that a read or write cycle was initiated by software when the STATUS[INVAL_BIT] = 1. Write access to time /date registers gets nullified (terminate normally) and no register value gets changed. Read during STATUS[INVAL_BIT] asserted returns 0xFFFF. No Transfer Error is asserted. This bit gets cleared by writing 1 to it. 0 - Read and Write accesses are normal. 1 - Read or Write accesses occurred when STATUS[INVAL_BIT] was asserted.
7-6 WE	Write Enable These bits control the entry and exit of the write protection mode. Both registers are protected by the write protection mechanism. These are self-clearing bits. Reads will return zeros.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	NOTE When the registers are unlocked, they remain in this unlocked state for a time of 2 seconds after which they get locked automatically. After power-on-reset, the registers come out as unlocked but they get locked automatically 15 seconds after power on. 00,01,11,10 - Disable Write Protection - Registers are unlocked. 10 - Enable Write Protection - Registers are locked.
5 CMP_INT	Compensation Interval This read-only status bit is asserted for a time equal to compensation interval seconds (as configured by user). This bit is used by MCU to calculate the interrupts serviced during this interval and perform corrections in case of deviations (i.e. Calibration). This bit toggles on every start of new compensation interval and is either 0 or 1 during the entire duration. This bit will not toggle if compensation logic has been disabled by MCU.
4 —	Reserved
3 RST_SRC	Reset Source This bit indicates to the user software the cause for reset to the part and subsequent boot up of CPU. This bit is asserted only on the power-on-reset that is generated within RTC (that is Power On Reset when both VBAT and VDD are powered up) and this indicates that part has been reset. On entering Standby mode, this bit is cleared which indicates the part is booting after Standby Mode Exit, when the Standby mode is actually exited. 0 - Standby Mode Exit. Part was reset due to Standby Mode Exit (that is when VDD is powered up and VBAT was not powered down at all). 1 - Power-On Reset. Part was reset due to Power-On Reset (that is Power-On-Reset when both VBAT and VDD are powered up).
2 CPU_LOW_VOLT	CPU Low Voltage Warning Status This bit is asserted when the MCU/CPU power falls below the read-only threshold when all write cycles are terminated normally and no change is done to the registers. Registers are read only. Transfer Error is not asserted on write access. 0 - CPU in Normal Operating Voltage. 1 - CPU Voltage is below Normal Operating Voltage. RTC Registers in read-only mode.
1 WRITE_PROT_EN	Write Protect Enable Status This read-only bit indicates that registers are in locked mode and write to them is disabled. Any write access made to the register space when write protection is enabled (that is in locked mode) will cause the transfer error signal to be asserted. 0 - Registers are unlocked and can be accessed. 1 - Registers are locked and in read-only mode.

Table continues on the next page...

Table continued from the previous page...

Field	Description
0 INVAL_BIT	Invalidate CPU Read/Write Access This read-only bit indicates the time /date counters are invalid or changing and therefore should not be read/written to. This bit is asserted for 1 oscillator clock cycle before and after the 1 Hz (seconds clock) boundary/edge. Write access to time /date registers gets nullified (terminate normally) and no register value gets changed. Read during STATUS[INVAL_BIT] asserted returns 0xFFFF. No Transfer Error is asserted. 0 - Time /Date Counters can be read/written. Time /Date is valid. 1 - Time /Date Counter values are changing or Time /Date is invalid and cannot be read or written.

11.5.1.1.11 Interrupt Status (ISR)

NOTE

For the sampling timer interrupt status bits [14:6], refer to chip configuration chapter for the applicable sampling timer frequencies.

This register indicates the status of the various real-time clock interrupts. When an event of the types included in this register occurs then the bit will be set in this register regardless of its corresponding interrupt enable bit being set. The status bits are cleared by writing a value of 1, which also clears the interrupt. Interrupts may occur while the system clock is idle or in Standby mode. When the system enters the active power mode, interrupt will be indicated to the CPU. The first event of the Sampling Timer interrupts after Power-on-Reset should not used to qualify any periodic interval. However, the correct periodic interval (that is 512 Hz or 256 Hz, and so on) should be determined using two sampling timer interrupts. The time between two interrupt would always be the correct time period.

Tamper Interrupt Status is set on reset as POR is generated when battery and CPU power are unavailable and either is powered up. Removal of battery is considered as a tamper and therefore this bit is set on reset.

The status register is also cleared on software reset except for the tamper status which remains unaffected .

NOTE

Sampling interrupts from 512 Hz to 2 Hz are generated using uncompensated clock. Only 1 Hz is compensated.

Offset

Register	Offset
ISR	14h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IS_512HZ	IS_256HZ	IS_128HZ	IS_64HZ	IS_32HZ	IS_16HZ	IS_8HZ	IS_4HZ	IS_2HZ	IS_1HZ	MIN_IS	HOUR_IS	DAY_IS	ALM_IS	0	TAMPER_...
W	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	1

Fields

Field	Description
15 IS_512HZ	512 Hz Interval Interrupt Status 0 - Interrupt is de-asserted. 1 - Interrupt is asserted.
14 IS_256HZ	256 Hz Interval Interrupt Status 0 - Interrupt is de-asserted. 1 - Interrupt is asserted.
13 IS_128HZ	128 Hz Interval Interrupt Status 0 - Interrupt is de-asserted. 1 - Interrupt is asserted.
12 IS_64HZ	64 Hz Interval Interrupt Status 0 - Interrupt is de-asserted. 1 - Interrupt is asserted.
11 IS_32HZ	32 Hz Interval Interrupt Status 0 - Interrupt is de-asserted. 1 - Interrupt is asserted.
10 IS_16HZ	16 Hz Interval Interrupt Status 0 - Interrupt is de-asserted. 1 - Interrupt is asserted.
9 IS_8HZ	8 Hz Interval Interrupt Status 0 - Interrupt is de-asserted. 1 - Interrupt is asserted.
8 IS_4HZ	4 Hz Interval Interrupt Status 0 - Interrupt is de-asserted. 1 - Interrupt is asserted.
7 IS_2HZ	2 Hz Interval Interrupt Status 0 - Interrupt is de-asserted.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Interrupt is asserted.
6 IS_1HZ	1 Hz Interval Interrupt Status 0 - Interrupt is de-asserted. 1 - Interrupt is asserted.
5 MIN_IS	Minutes Interrupt Status 0 - Interrupt is de-asserted. 1 - Interrupt is asserted.
4 HOUR_IS	Hours Interrupt Status 0 - Interrupt is de-asserted. 1 - Interrupt is asserted.
3 DAY_IS	Days Interrupt Status 0 - Interrupt is de-asserted. 1 - Interrupt is asserted.
2 ALM_IS	Alarm Interrupt Status This bit indicates that the alarm value programmed matches the counter values. 0 - Interrupt is de-asserted. 1 - Interrupt is asserted.
1 —	Reserved
0 TAMPER_IS	Tamper Interrupt Status This field is cleared when TAMPER_SCR[TMPR_STS] is cleared. 0 - Interrupt is de-asserted. 1 - Interrupt is asserted (Default on reset) .

11.5.1.1.12 Interrupt Enable (IER)

For the sampling timer interrupt enable bits [14:6], refer to chip configuration chapter for the applicable sampling timer frequencies.

The real-time clock interrupt enable register (IER) is used to enable/disable the various real-time clock interrupts. De-asserting an interrupt enable bit has no effect on the assertion of its corresponding status bit.

Alarm interrupt is asserted on counters matching the alarm setting done in the memory map. The counters matched for the alarm interrupt are selected based on the Alarm Type set in CTRL[3:2] bits. The various types of alarm available are as per the following table. Only one alarm type can be used at a time.

Table 37. Alarm Match Table

ALM_MATCH[1:0] (CTRL[3:2])	Counters Matched	Alarm Type
00	Seconds, Minutes, and Hours	Daily
01	Seconds, Minutes, Hours, and Days	Monthly
10	Seconds, Minutes, Hours, Days, and Months	Yearly
11	Seconds, Minutes, Hours, Days, Months, and Year	One Time

A common interrupt is generated by the block. The user software should read the status register in the interrupt service routine to determine which interrupt has occurred.

The tamper detect interrupt enable is an exception as it is enabled after power-on-reset. All interrupts enables except tamper interrupt enable are reset to default state on software reset.

Offset

Register	Offset
IER	16h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IE_512HZ	IE_256HZ	IE_128HZ	IE_64HZ	IE_32HZ	IE_16HZ	IE_8HZ	IE_4HZ	IE_2HZ	IE_1HZ	MIN_IE	HOUR_IE	DAY_IE	ALM_IE	0	TAMPER_...
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Fields

Field	Description
15 IE_512HZ	512 Hz Interval Interrupt Enable 0 - Interrupt is disabled. 1 - Interrupt is enabled.
14 IE_256HZ	256 Hz Interval Interrupt Enable 0 - Interrupt is disabled. 1 - Interrupt is enabled.
13 IE_128HZ	128 Hz Interval Interrupt Enable 0 - Interrupt is disabled. 1 - Interrupt is enabled.
12	64 Hz Interval Interrupt Enable

Table continues on the next page...

Table continued from the previous page...

Field	Description
IE_64HZ	0 - Interrupt is disabled. 1 - Interrupt is enabled.
11 IE_32HZ	32 Hz Interval Interrupt Enable 0 - Interrupt is disabled. 1 - Interrupt is enabled.
10 IE_16HZ	16 Hz Interval Interrupt Enable 0 - Interrupt is disabled. 1 - Interrupt is enabled.
9 IE_8HZ	8 Hz Interval Interrupt Enable 0 - Interrupt is disabled. 1 - Interrupt is enabled.
8 IE_4HZ	4 Hz Interval Interrupt Enable 0 - Interrupt is disabled. 1 - Interrupt is enabled.
7 IE_2HZ	2 Hz Interval Interrupt Enable 0 - Interrupt is disabled. 1 - Interrupt is enabled.
6 IE_1HZ	1 Hz Interval Interrupt Enable 0 - Interrupt is disabled. 1 - Interrupt is enabled.
5 MIN_IE	Minutes Interrupt Enable 0 - Interrupt is disabled. 1 - Interrupt is enabled.
4 HOUR_IE	Hours Interrupt Enable 0 - Interrupt is disabled. 1 - Interrupt is enabled.
3 DAY_IE	Days Interrupt Enable 0 - Interrupt is disabled. 1 - Interrupt is enabled.
2 ALM_IE	Alarm Interrupt Enable This bit indicates that the alarm value programmed matches the counter values. 0 - Interrupt is disabled.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Interrupt is enabled.
1 —	Reserved
0 TAMPER_IE	Tamper Interrupt Enable 0 - Interrupt is disabled. 1 - Interrupt is enabled (Default on reset).

11.5.1.1.13 General Purpose Data (GP_DATA_REG)

The use of the bits in this register are specific to the MCU. See the Chip-specific section for description of this register. Software reset has no effect on the contents of this register.

Offset

Register	Offset
GP_DATA_REG	20h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CFG15	CFG14	CFG13	CFG12	CFG11	CFG10	CFG9	CFG8	CFG7	CFG6	CFG5	CFG4	CFG3	CFG2	CFG1	CFG0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
15-0 CFGn	CFGn General Purpose bit field

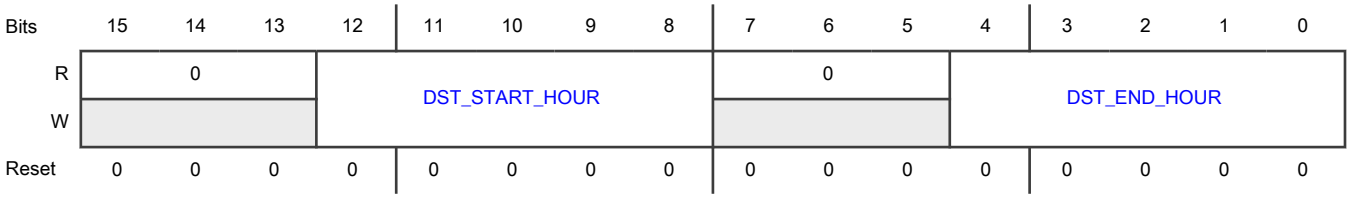
11.5.1.1.14 Daylight Saving Hour (DST_HOUR)

DST_HOUR stores the time in hours when the Daylight Saving has to be applied or reversed. This register is programmable when CTRL[DST_EN] = 0. When CTRL[DST_EN] = 1, the contents of this register cannot be changed. The user software should program the correct hour value (0 - 23) as per the regional settings. For example, if the Daylight Saving starts at 2:00 AM on March 25 and ends at 2:00 AM on October 28 in 2007 then the time at which the RTC advances or falls back is actually 1:59 AM. Hence the user software should program 1 for the hour count value (and not 2!) i.e. write 0x0101 in this register. 59 minute count is automatically checked inside RTC and therefore not required to be programmed. This register has no effect on software reset.

Offset

Register	Offset
DST_HOUR	22h

Diagram



Fields

Field	Description
15-13 —	Reserved
12-8 DST_START_HOUR	Daylight Saving Time (DST) Hours Start Value This is the hour value for the time when DST comes into effect. Same as Hours Counter Value in the Hours and Minutes Counters (HOURMIN) .
7-5 —	Reserved
4-0 DST_END_HOUR	Daylight Saving Time (DST) Hours End Value This is the hour value for the time when DST is reversed. Same as Hours Counter Value in the Hours and Minutes Counters (HOURMIN) .

11.5.1.1.15 Daylight Saving Month (DST_MONTH)

This register stores the month when the Daylight Saving has to be applied or reversed. This register is programmable when CTRL[DST_EN] = 0. When CTRL[DST_EN] = 1, the contents of this register cannot be changed. The CPU should program the correct month value (1 - 12) as per the regional settings. For example, if the Daylight Saving starts at March 25 and ends at October 28 in 2007. Hence the CPU should write 0x030A in this register. This register has no effect on software reset.

Offset

Register	Offset
DST_MONTH	24h

Diagram

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

Fields

Field	Description
15-12 —	Reserved
11-8 DST_START_MONTH	Daylight Saving Time (DST) Month Start Value This is the month value for the time when DST comes in to effect. See the Year and Month Counters (YEARMON) .
7-4 —	Reserved
3-0 DST_END_MONTH	Daylight Saving Time (DST) Month End Value This is the month value for the time when DST is reversed. See the Year and Month Counters (YEARMON) .

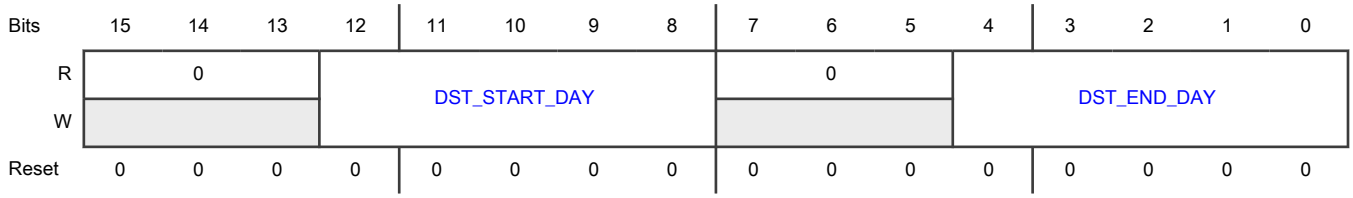
11.5.1.1.16 Daylight Saving Day (DST_DAY)

DST_DAY stores the day when the Daylight Saving has to be applied or reversed. This register is programmable when CTRL[DST_EN] = 0. When CTRL[DST_EN] = 1, the contents of this register cannot be changed. The CPU should program the correct day value (1 - 31) as per the regional settings. For example, if the Daylight Saving starts at March 25 and ends at October 28 in 2007. Hence the CPU should write 0x191C in this register. This register is unaffected by software reset.

Offset

Register	Offset
DST_DAY	26h

Diagram



Fields

Field	Description
15-13 —	Reserved
12-8 DST_START_DAY	Daylight Saving Time (DST) Day Start Value This is the day value for the time when DST comes into effect.
7-5 —	Reserved
4-0 DST_END_DAY	Daylight Saving Time (DST) Day End Value This is the day value for the time when DST is reversed.

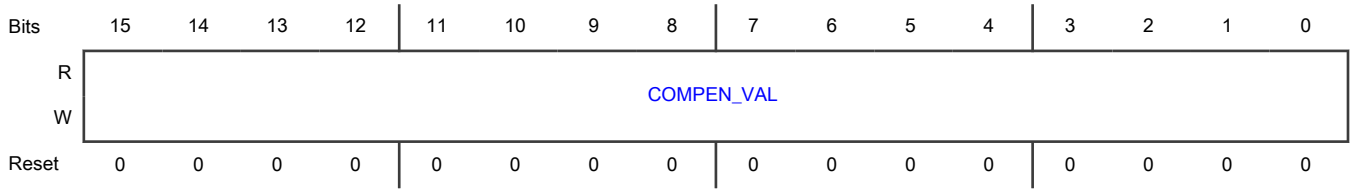
11.5.1.1.17 Compensation (COMPEN)

The compensation register stores the compensation value that will be used by the compensation block to correct the 1 Hz clock.

Offset

Register	Offset
COMPEN	28h

Diagram



Fields

Field	Description
15-0	Compensation Value

Table continues on the next page...

Field	Description
COMPEN_VAL	This register stores the compensation parameters. The definition of this field is dependent on the setting of CTRL[FINEEN]. If CTRL[FINEEN] = 0: • Compensation/Correction Value (COMPEN[7:0]) - Compensation/Correction Value is a 2's complement value by which the 1 Hz Clock is modified (during its generation) by either adding or removing RTC Oscillator clock cycles. • Compensation Interval (COMPEN[15:8]) - Compensation Interval is the duration in seconds over which the correction is applied. This is the time in which the addition or removal of 32.768 kHz clock cycles is done thereby ensuring that the compensation interval is close to the interval obtained with an ideal 1 Hz clock. If CTRL[FINEEN] = 1 • Integral Compensation Value (COMPEN[15:12])- This is a 2's complement value of the integer part of correction or compensation value that has to be adjusted in every 1 second period. This value is expressed in terms of number of clock cycles of the RTC Oscillator clock. • COMPEN[11:7] -- Should be zero • Fraction Compensation Value (COMPEN[6:0]) - This is the fractional part of the correction or compensation value that has to be adjusted. This value is expressed as number of clock cycles of a fixed 4.194304 MHz clock . This value is always a positive number.

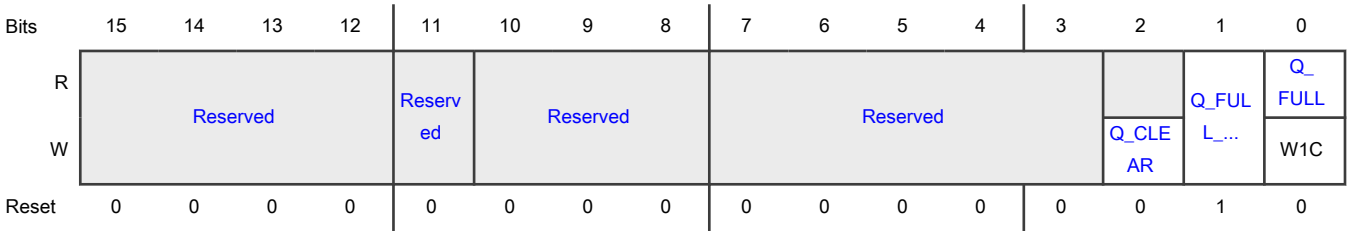
11.5.1.1.18 Tamper Queue Status and Control (TAMPER_QSCR)

The TAMPER_QSCR register contains queue status and interrupt enable fields.

Offset

Register	Offset
TAMPER_QSCR	2Eh

Diagram



Fields

Field	Description
15-12 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
11 —	Reserved
10-8 —	Reserved
7-3 —	Reserved
2 Q_CLEAR	Q_CLEAR This field when set clears the Tamper Queue. This is an auto clear bit. It gets cleared in the next cycle.
1 Q_FULL_INT_EN	Q_FULL_INT_EN Queue full interrupt enable 0 - Queue full interrupt is disabled. 1 - Queue full interrupt is enabled.
0 Q_FULL	Q_FULL Tamper Queue full Status 0 - The tamper queue is not full. 1 - The tamper queue is full.

11.5.1.1.19 Tamper Status and Control (TAMPER_SCR)

For the implementation of TAMPER_SCR in the MCU, refer to chip configuration chapter.

This register stores the tamper event and provides control to the user to enable or disable each tamper individually. No tamper is disabled automatically unless corresponding control bit is de-asserted by software. The tamper statuses are stored as active high. The tamper status bits store the tamper event irrespective of their corresponding control bit being asserted or not. The control bits gate the assertion of tamper interrupt bit in ISR. These status and controls bits combined assert the tamper interrupt in the ISR register. The tamper interrupt will be cleared when all above Status bits are cleared.

Offset

Register	Offset
TAMPER_SCR	32h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				TMPR_STS				0				TMPR_EN			
W					W1C											
Reset	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	1

Fields

Field	Description
15-12 —	Reserved
11-8 TMPR_STS	Tamper Status Indicates if a tamper event was detected or not. Writing '1' to this field clears the tamper status. 0000 - No Tamper Detected 0001 - Tamper Event Detected
7-4 —	Reserved
3-0 TMPR_EN	Tamper Control Controls the generation of tamper interrupt from corresponding tamper status bit. Enabled on reset. 0000 - Tamper Status reporting disabled. However, corresponding tamper status bit still stores the tamper event. 0001 - Tamper Status reporting enabled. Corresponding tamper status bit causes tamper interrupt when detected

11.5.1.1.20 Tamper 01 Filter Configuration (FILTER01_CFG)

This register must be configured once during the initial startup of application and must not be changed on the fly to avoid any erratic behavior. Changing the register on the fly might lead to missing of tamper events or assertion of false tamper events.

Offset

Register	Offset
FILTER01_CFG	34h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	POL0	CLK_SEL0			FIL_DUR0				POL1	CLK_SEL1			FIL_DUR1			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
15 POL0	Tamper Detect Input Bit 0 Polarity Control This bit controls the polarity of tamper detect input bit 0 (TAMPER[0]). 0 - Tamper detect input bit 0 is active high. 1 - Tamper detect input bit 0 is active low.
14-12 CLK_SEL0	Tamper Filter 0 Clock Select This bit is a write once bit and selects the clock source for tamper filter for tamper detect input bit 0. 000 - 32 kHz clock 001 - 512 Hz clock 010 - 128 Hz clock 011 - 64 Hz clock 100 - 16 Hz clock 101 - 8 Hz clock 110 - 4 Hz clock 111 - 2 Hz clock
11-8 FIL_DUR0	Tamper Detect Bit 0 Filter Duration This bit indicates the number of tamper filter clock cycles for which the TAMPER[0] signal should remains stable before being detected as a tamper. These bits are used by the tamper filtering operation. With the tamper duration set to Zero any tamper detected on the tamper pins will directly set the tamper status and interrupt bits. Caution is required when making the tamper filter duration equal to 0 as any glitches on the tamper pins will cause a tamper interrupt. 0000 - Filtering operation disabled. 0001-1111 - Number of tamper filter clock cycles to be counted when tamper is asserted.
7 POL1	Tamper Detect Input Bit 1 Polarity Control This bit controls the polarity of tamper detect input bit 1 (TAMPER[1]). 0 - Tamper detect input bit 1 is active high. 1 - Tamper detect input bit 1 is active low.
6-4	Tamper Filter 1 Clock Select

Table continues on the next page...

Table continued from the previous page...

Field	Description
CLK_SEL1	This bit is a write once bit and selects the clock source for tamper filter for tamper detect input bit 1. 000 - 32 kHz clock 001 - 512 Hz clock 010 - 128 Hz clock 011 - 64 Hz clock 100 - 16 Hz clock 101 - 8 Hz clock 110 - 4 Hz clock 111 - 2 Hz clock
3-0 FIL_DUR1	Tamper Detect Bit 1 Filter Duration This bit indicates the number of tamper filter clock cycles for which the TAMPER[1] signal should remain stable before being detected as a tamper. These bits are used by the tamper filtering operation. With the tamper duration set to Zero any tamper detected on the tamper pins will directly set the tamper status and interrupt bits. Caution is required when making the tamper filter duration equal to 0 as any glitches on the tamper pins will cause a tamper interrupt. 0000 - Filtering operation disabled. 0001-1111 - Number of tamper filter clock cycles to be counted when tamper is asserted.

11.5.1.1.21 Tamper 23 Filter Configuration (FILTER23_CFG)

This register must be configured once during the initial startup of application and must not be changed on the fly to avoid any erratic behavior. Changing the register on the fly might lead to missing of tamper events or assertion of false tamper events.

Offset

Register	Offset
FILTER23_CFG	36h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	POL2	CLK_SEL2			FIL_DUR2				POL3	CLK_SEL3			FIL_DUR3			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
15 POL2	Tamper Detect Input Bit 2 Polarity Control This bit controls the polarity of tamper detect input bit 2 (TAMPER[2]). 0 - Tamper detect input bit 2 is active high. 1 - Tamper detect input bit 2 is active low.
14-12 CLK_SEL2	Tamper Filter 2 Clock Select This bit is a write once bit and selects the clock source for tamper filter for tamper detect input bit 2. 000 - 32 kHz clock 001 - 512 Hz clock 010 - 128 Hz clock 011 - 64 Hz clock 100 - 16 Hz clock 101 - 8 Hz clock 110 - 4 Hz clock 111 - 2 Hz clock
11-8 FIL_DUR2	Tamper Detect Bit 2 Filter Duration This bit indicates the number of tamper filter clock cycles for which the TAMPER[2] signal should remain stable before being detected as a tamper. These bits are used by the tamper filtering operation. With the tamper duration set to Zero any tamper detected on the tamper pins will directly set the tamper status and interrupt bits. Caution is required when making the tamper filter duration equal to 0 as any glitches on the tamper pins will cause a tamper interrupt. 0000 - Filtering operation disabled. 0001-1111 - Number of tamper filter clock cycles to be counted when tamper is asserted.
7 POL3	Tamper Detect Input Bit 3 Polarity Control This bit controls the polarity of tamper detect input bit 3 (TAMPER[3]). 0 - Tamper detect input bit 3 is active high. 1 - Tamper detect input bit 3 is active low.
6-4	Tamper Filter 3 Clock Select

Table continues on the next page...

Table continued from the previous page...

Field	Description
CLK_SEL3	This bit is a write once bit and selects the clock source for tamper filter for tamper detect input bit 3. 000 - 32 kHz clock 001 - 512 Hz clock 010 - 128 Hz clock 011 - 64 Hz clock 100 - 16 Hz clock 101 - 8 Hz clock 110 - 4 Hz clock 111 - 2 Hz clock
3-0 FIL_DUR3	Tamper Detect Bit 3 Filter Duration This bit indicates the number of tamper filter clock cycles for which the TAMPER[3] signal should remain stable before being detected as a tamper. These bits are used by the tamper filtering operation. With the tamper duration set to Zero any tamper detected on the tamper pins will directly set the tamper status and interrupt bits. Caution is required when making the tamper filter duration equal to 0 as any glitches on the tamper pins will cause a tamper interrupt. 0000 - Filtering operation disabled. 0001-1111 - Number of tamper filter clock cycles to be counted when tamper is asserted.

11.5.1.1.22 Tamper Queue (TAMPER_QUEUE)

Tamper queue register provides the Time stamp and the Pin number on which the tamper occurred.

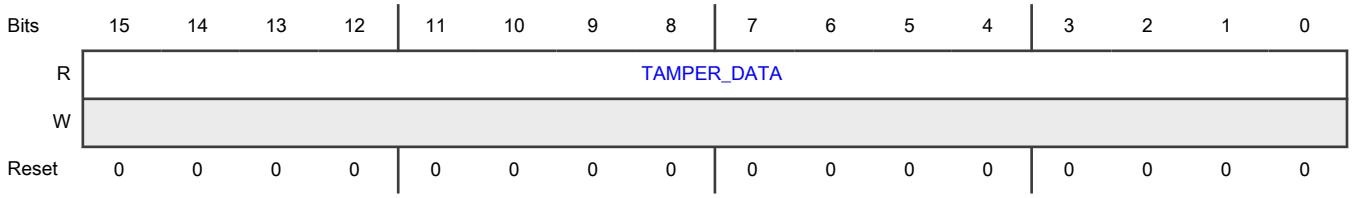
NOTE

Reading beyond the depth of FIFO when the FIFO is full or empty returns random data. See the Chip Configuration chapter for the depth of the Tamper FIFO.

Offset

Register	Offset
TAMPER_QUEUE	40h

Diagram



Fields

Field	Description
15-0 TAMPER_DATA	Tamper type stamp and pin number information register The complete tamper timestamp and status information can be obtained by making two read accesses to this register: The first read gives the following information: • TAMPER_DATA[4:0] = Hour • TAMPER_DATA[9:5] = Day • TAMPER_DATA[13:10] = Month • TAMPER_DATA[15:14] = 2'b0 The second read gives the remaining information, that is: • TAMPER_DATA[5:0] = seconds • TAMPER_DATA[11:6] = minutes • TAMPER_DATA[14:12] = Tamper Index, that is, index of tamper status bits from 0 to 3 • TAMPER_DATA[15] = 1'b0 **NOTE** The year value can be obtained by reading the year counter register

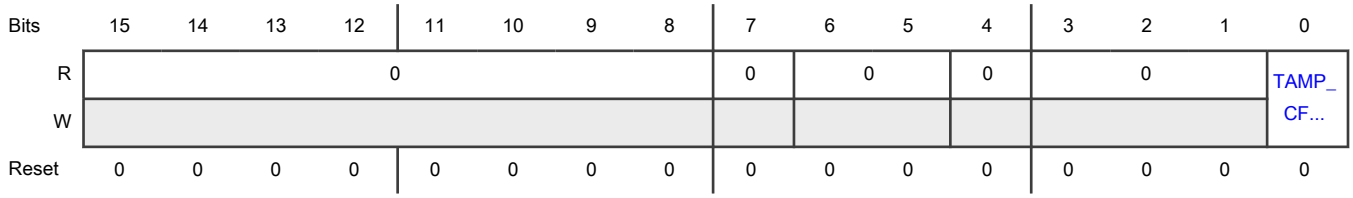
11.5.1.1.23 Control 2 (CTRL2)

The CTRL2 register defines the attributes for the configuration of tamper pins.

Offset

Register	Offset
CTRL2	42h

Diagram



Fields

Field	Description
15-8 —	Reserved
7 —	Reserved
6-5 —	Reserved
4 —	Reserved
3-1 —	Reserved
0 TAMP_CFG_OVER	Tamper Configuration Over **NOTE** Set this field to 1 only when all the tamper_cfg registers are programmed. **NOTE** To enable the tamper feature, this field should be set. 0 - Tamper filter processing disabled. 1 - Tamper filter processing enabled.

Chapter 12
RTC Subsystem

12.1 Overview

RTC subsystem has sub-second and wake timer feature. The 16-bit sub-second counter is clocked by the 32 khz clock input. The wake timer is clocked by the 1 khz clock which is the output of the 5-bit ripple counter with the 32 khz clock as the clock source.

12.1.1 Block diagram

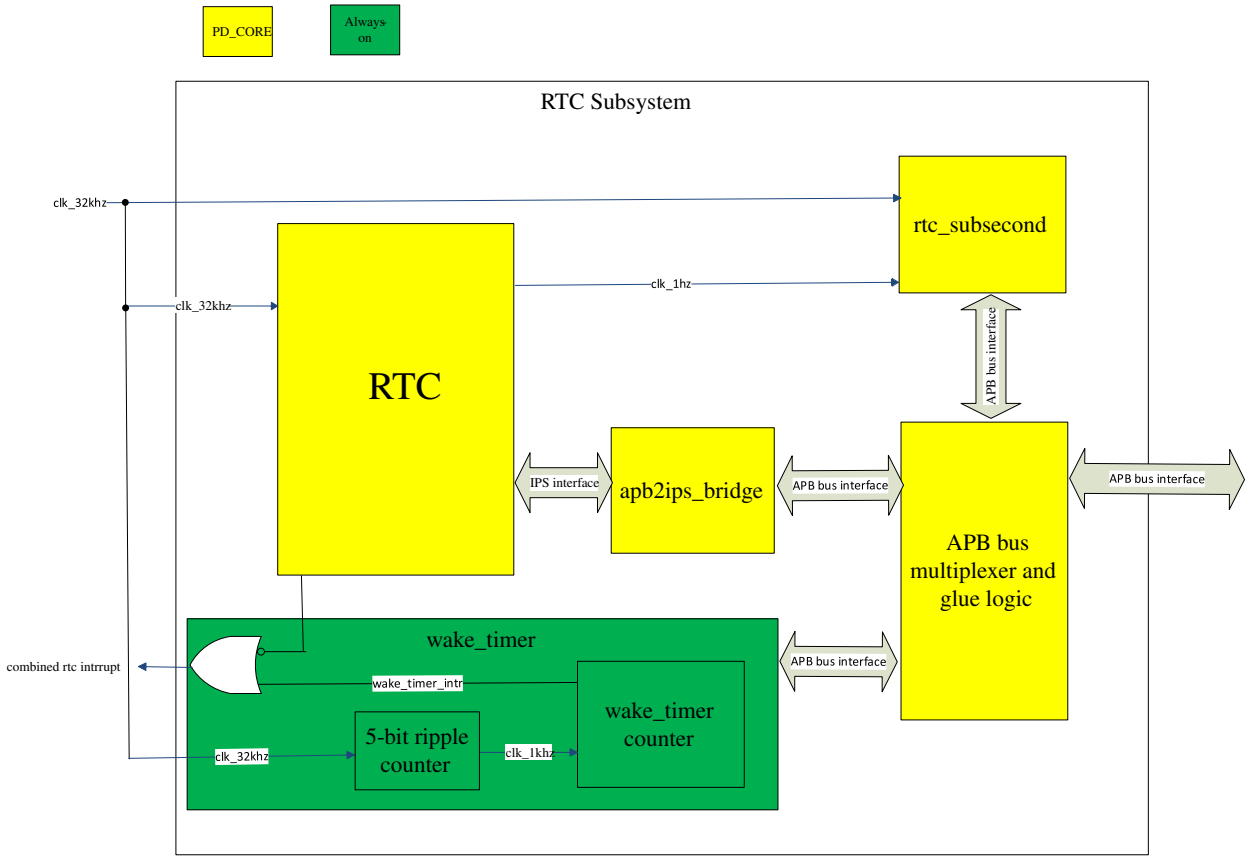


Figure 17. RTC Subsystem block diagram

12.1.2 Features

- 16-bits sub-second counter function support in real-time application.
- 5-bit ripple down counter in VBAT_SWI domain to wakeup chip from deep power down mode.

12.2 Memory map and register description

This section includes the RTC subsystem memory map and detailed descriptions of all registers.

12.2.1 RTC subsystem register descriptions

12.2.1.1 ROBUST_RTC memory map

RTC_SUBSYSTEM base address: 4002_C000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
800	Sub-second control (SUBSECOND_CTRL)	32	See section	0000_0000
804	Sub-second counter (SUBSECOND_CNT)	32	See section	0000_0000
C00	Wake timer control (WAKE_TIMER_CTRL)	32	See section	0000_0000
C0C	Wake timer counter (WAKE_TIMER_CNT)	32	RW	0000_0000

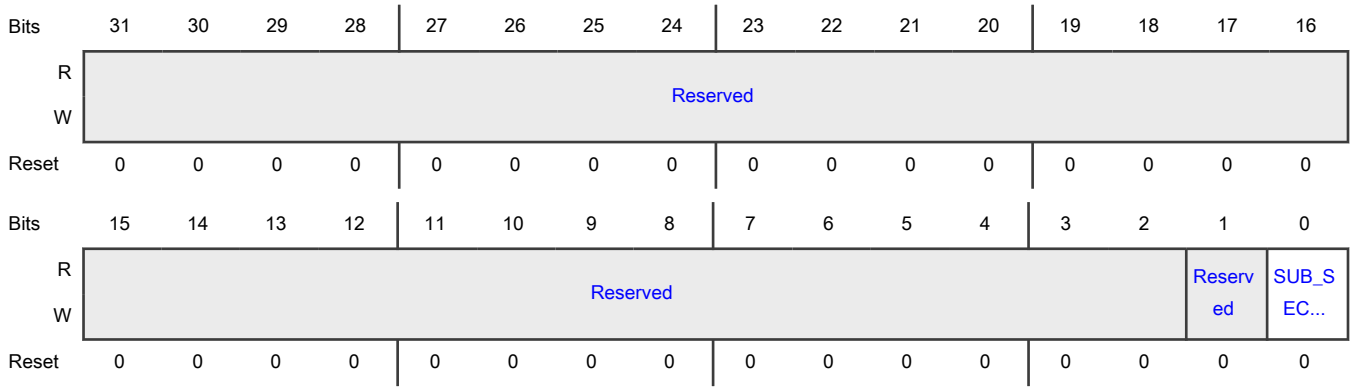
12.2.1.1.1 Sub-second control (SUBSECOND_CTRL)

This register controls the sub-second counter register.

Offset

Register	Offset
SUBSECOND_CTRL	800h

Diagram



Fields

Field	Description
31-2 —	Reserved
1 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
0 SUB_SECOND_CNT_EN	Sub-second counter enable bit **NOTE** CTRL[CLKOUT] field should be set to choose the 1 HZ clock output in order for the sub-second counter to synchronize with the RTC_SECONDS counter. 0 - Disabled. 1 - Enabled.

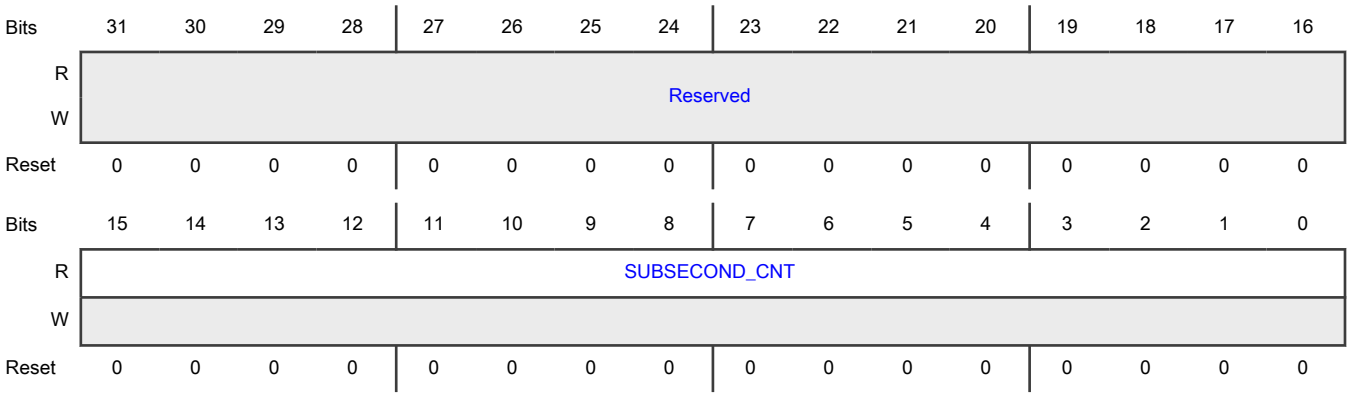
12.2.1.1.2 Sub-second counter (SUBSECOND_CNT)

The 16-bit sub-second counter is clocked by the 32khz clock input.

Offset

Register	Offset
SUBSECOND_CNT	804h

Diagram



Fields

Field	Description
31-16 —	Reserved
15-0 SUBSECOND_CNT	Current sub-second counter value Due to clock domain crossing, it is recommended to read the register twice in succession and confirm the two values are the same. If not, repeat the procedure until the two values are the same.

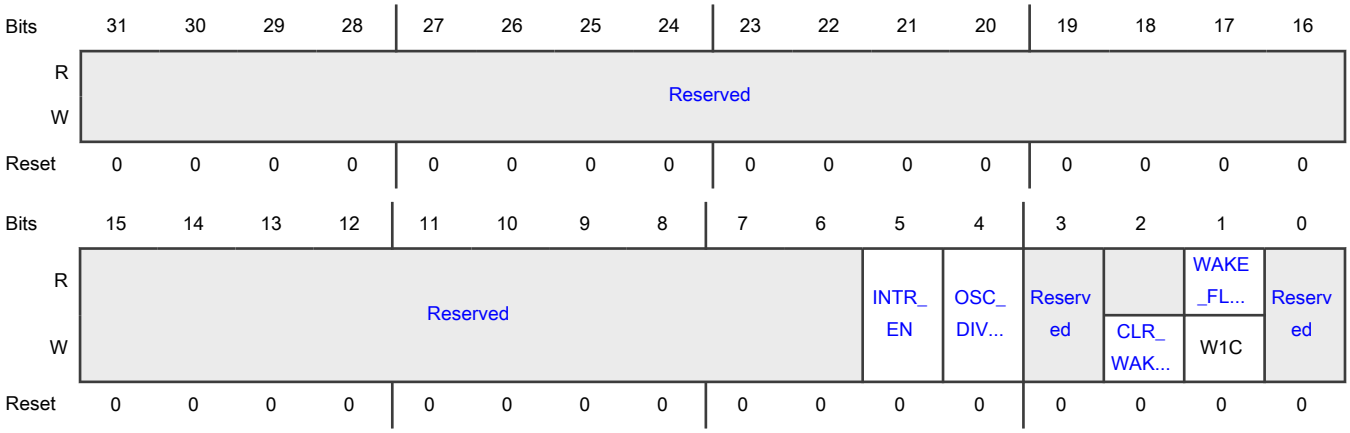
12.2.1.1.3 Wake timer control (WAKE_TIMER_CTRL)

The wake timer is clocked by the 1 kHz clock which is the output of the 5-bit ripple counter with the 32 kHz clock as the clock source. This register controls the wake timer counter register.

Offset

Register	Offset
WAKE_TIMER_CTRL	C00h

Diagram



Fields

Field	Description
31-6 —	Reserved
5 INTR_EN	Enable interrupt when WAKE_FLAG is set. 0 - Disabled. 1 - Enabled.
4 OSC_DIV_ENA	Enable the 5-bit clock divider to divide down the 32Khz input clock to generate the 1Khz clock source for the wake timer. 0 - Disabled. 1 - Enabled.
3 —	Reserved
2 CLR_WAKE_TIMER	Clear wake timer **NOTE** Reading this bit always returns 0.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - No effect. 1 - Clears the wake counter and halt operation until a new count value is loaded.
1 WAKE_FLAG	Wake timer status flag Writing a 1 clears this status bit. Writing 0 has no effect. 0 - Wake timer has not timed out. 1 - Wake timer has timed out.
0 —	Reserved

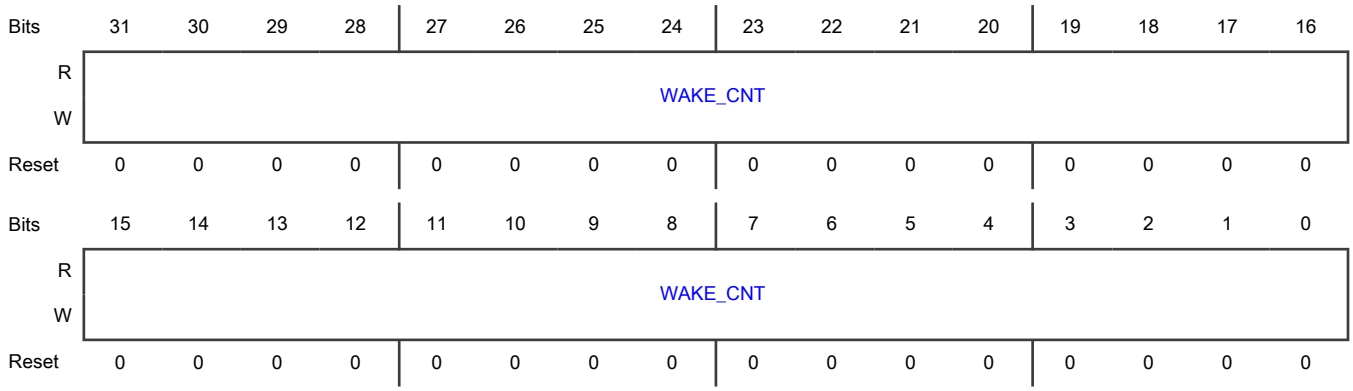
12.2.1.1.4 Wake timer counter (WAKE_TIMER_CNT)

This register should not be written-to while counting is in progress.

Offset

Register	Offset
WAKE_TIMER_CNT	C0Ch

Diagram



Fields

Field	Description
31-0 WAKE_CNT	Wake counter A read reflects the current value of the wake timer. Due to clock domain crossing, it is recommended to read the register twice in succession and confirm the two values are the same. If not, repeat the procedure until the two values are the same. A write pre-loads a start-count value into the timer and initializes a countdown sequence.

Chapter 13
System Controller (SYSCTL)

13.1 Chip-specific SYSCTL information

Table 38. Reference links to related information

Topic	Related module	Reference
Full description	SYSCTL	SYSCTL
System memory map		System memory map
Clocking		Clock generation
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

13.1.1 Module instances

This device has one instance of the SYSCTL module, SYSCTL0.

13.1.2 I^2S signal sharing

Flexcomm3 is not available for I2S signal sharing, and is excluded from the SHAREDFn registers.

Configure I^2S signal sharing as follows.

Before writing FCnCTRLSEL and SHAREDCTRLSETx registers, remove write protection inside UPDATELCKOUT register by resetting bit UPDATELCKOUT.

1. Select the appropriate functions in IOCON for the pins that will actually be connected to the outside world for I^2S operation.

   Set up shared signal sets that will be used by writing to the SHAREDCTRLSET0 and/or SHAREDCTRLSET1 registers.

2. Set up any signal sharing for each Flexcomm Interface that uses shared signals by writing to the registers FC0CTRLSEL through FC7CTRLSEL as required.

   Set up Flexcomm Interfaces using I^2S signal sharing as needed. Any Flexcomm Interface acting as master first, then slaves.

   NOTE

   Signal sharing connections are made as register values are changed, without synchronization, and so should be done prior to the start of data streams.

Also, any I^2S master that is providing SCK and WS signals for shared usage should also be configured to use the shared signal. For example, if the Flexcomm Interface 0 is providing SCK and WS to shared set 0, FC0CTRLSEL should select shared set 0 for SCK and WS.

13.2 Overview

The System Controller (SYSCTL) provides these system-level features: I2S signal sharing and gray-to-binary decoding. The I2S signal sharing feature supports applications where the requirements exceed what can be accomplished using a single I2S interface with four channel pairs.

13.2.1 Features

- I²S signal sharing allows multiple Flexcomm I²S interfaces to share a configurable combination of I²S clock, WS, and DATA signals without external board wiring.
- Gray to binary decoder allows decoding gray value coming from OS Event Timer.

13.3 Signals

The I2S signal sharing feature does not use pins directly but offers additional internal routing of existing I²S signals.

13.4 Functional description

13.4.1 I²S signal sharing

The I²S signal sharing feature allows more than one on-chip I²S interface to be internally connected to common clock, WS, and input data pins. Multiple I2S interfaces can be combined into a single TDM (time-division multiplexed) stream to reduce the need for external board wiring.

NOTE

Multiple I²S interfaces contributing output data to a single data line must still be accomplished with an external connection.

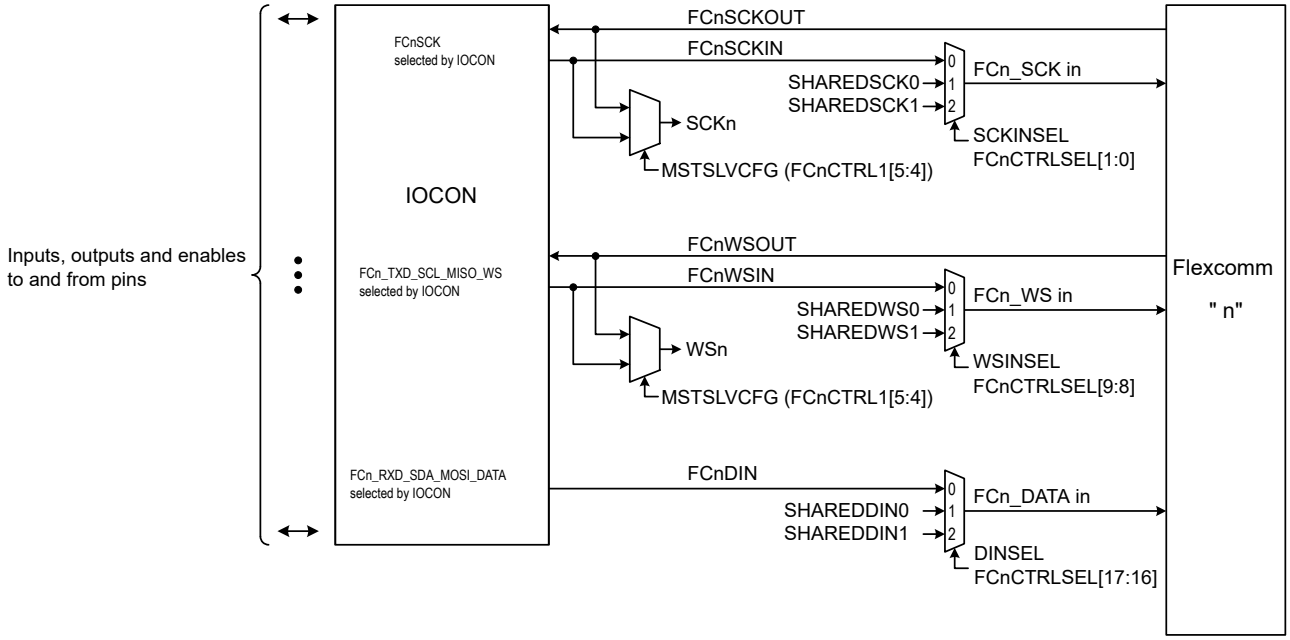
In general, each Flexcomm module configured for I²S can choose:

- Its own SCK, or a shared SCK.
- Its own WS, or a shared WS.
- Its own DATA in, or a shared DATA in.

Each Flexcomm module potentially contributes to shared signals:

- Its own SCK output (master) or SCK input (slave).
- Its own WS output (master) or WS input (slave).
- Its own DATA input.

Representative logic for the connection possibilities are shown below.



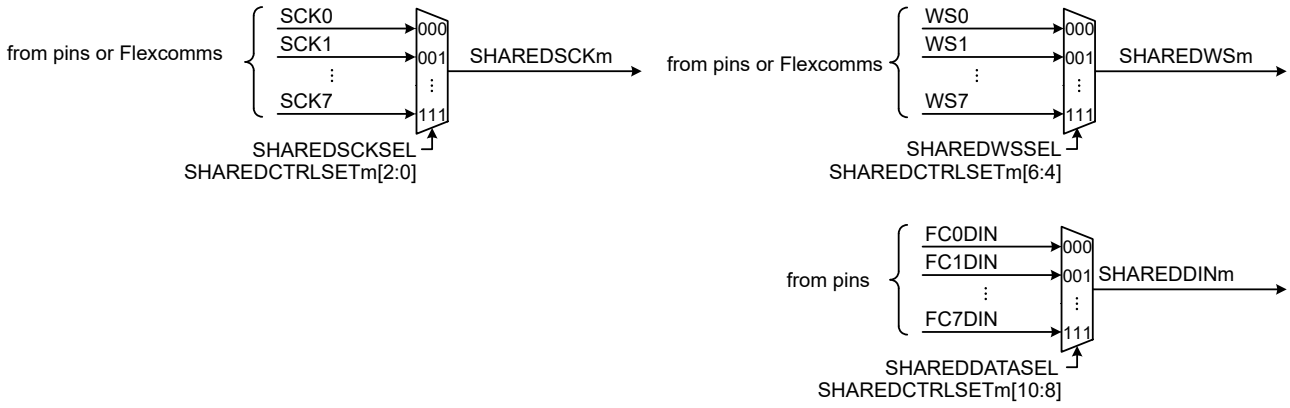
Note: These connection options are replicated for each Flexcomm "n", for all Flexcomm modules with I2S support.

Figure 18. Shared signal connections for each Flexcomm Module

Each SCK comes either from the pin selected as FCn_SCK (in IOCON) or from the related Flexcomm module if the Flexcomm is configured as an I2S master.

Each WS comes either from the pin selected as FCn_TXD_SCL_MISO_WS (in IOCON) or from the related Flexcomm module if the Flexcomm is configured as an I2S master.

Each FCnDIN comes from the pin selected as FCn_RXD_SDA_MOSI_DATA (in IOCON).



Note: This logic is replicated for each of the two sets of shared signals ("m", 0 through 1).

Figure 19. Shared signal source selection and control

13.4.1.1 Examples

Figure 20 shows an example application of an off-chip bidirectional codec connected to two different on-chip I²S interfaces: a master transmitter for the codec's input data and a slave receiver for the output data. This example uses signal sharing on a single SCK pin and on a single WS pin to reduce the number of external connections required. The data input and output signals cannot be shared on one pin because they are separate pins on the external codec.

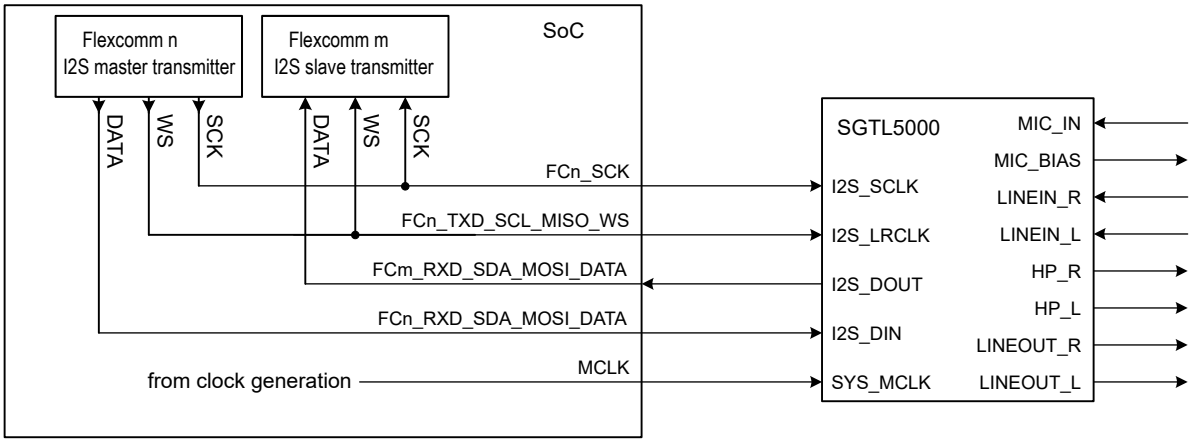


Figure 20. Example connection to an I²S bidirectional codec

Figure 21 shows multiple I²S slave receivers sharing signals for SCK, WS, and DATA. Each I²S interface receives its data from different time slots within a TDM stream.

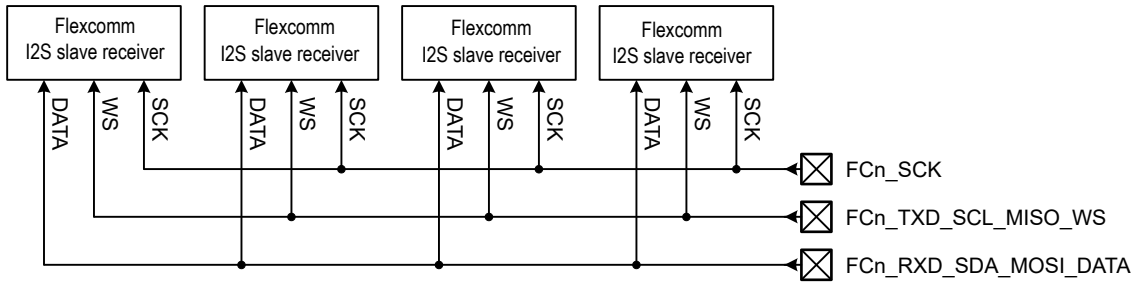


Figure 21. I²S signal sharing example, multiple slave receivers

Figure 22 shows master-to-slave operation where one I²S interface is a master going off-chip, and the other on-chip I²S interfaces are connected to it as slaves. All I²S interfaces are transmitters sharing SCK and WS. The multiple I²S interfaces supply data to their own device pins, but they contribute to a single TDM stream by wiring the data pins together. Here the I²S interfaces are transmitters; other applications could have the same connection topology but use receivers instead.

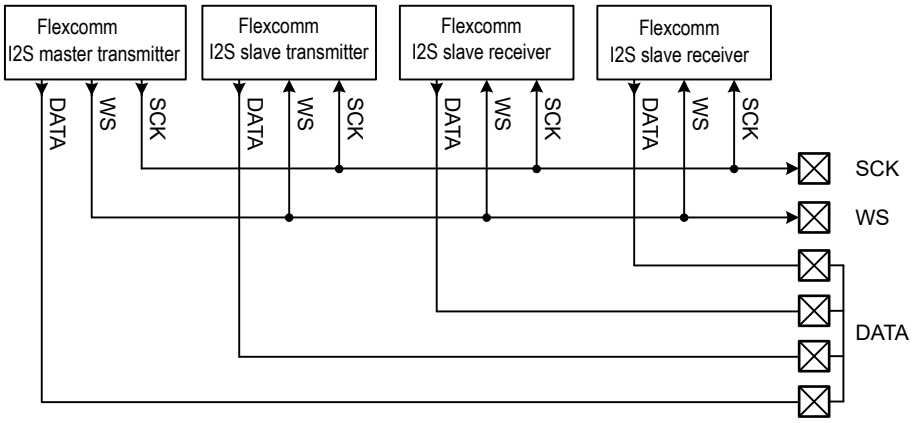


Figure 22. I²S signal sharing example: one master and multiple slave transmitters

Figure 23 shows two I²S interfaces transmitting and two receiving. All I²S interfaces share SCK and WS. One I²S interface is the master, the others are slaves. This example shows data with one I²S interface transmitting onto a shared DATA line while at least one other I²S is receiving from the same DATA line. Sharing the DATA line does not necessarily mean the transmitted data and the received data are the same because they could be delivered in different packets within a TDM frame.

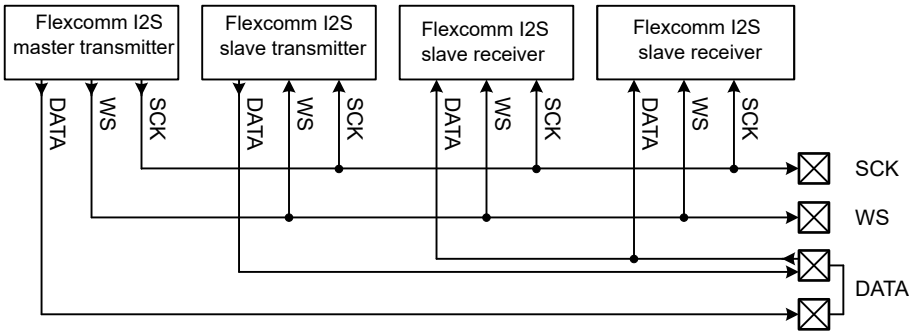


Figure 23. I²S signal sharing example: one master with mixed transmitters and receivers

13.5 Memory map and register definition

This section includes the SYSCTL module memory map and detailed descriptions of all registers.

13.5.1 System controller register descriptions

13.5.1.1 SYSCTL memory map

SYSCTL0 base address: 4002_3000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Write Lock Out (UPDATELCKOUT)	32	See section	See section

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
40	Shared Signal Select for Flexcomm 0 (FC0CTRLSEL)	32	See section	See section
44	Shared Signal Select for Flexcomm 1 (FC1CTRLSEL)	32	See section	See section
48	Shared Signal Select for Flexcomm 2 (FC2CTRLSEL)	32	See section	See section
50	Shared Signal Select for Flexcomm 4 (FC4CTRLSEL)	32	See section	See section
54	Shared Signal Select for Flexcomm 5 (FC5CTRLSEL)	32	See section	See section
58	Shared Signal Select for Flexcomm 6 (FC6CTRLSEL)	32	See section	See section
5C	Shared Signal Select for Flexcomm 7 (FC7CTRLSEL)	32	See section	See section
80 - 84	Shared Signal Set n (SHAREDCTRLSET0 - SHAREDCTRLSET1)	32	See section	See section
180	Gray Code LSB Input (CODE_GRAY_LSB)	32	RW	0000_0000
184	Gray Code MSB Input (CODE_GRAY_MSB)	32	See section	0000_0000
188	Binary Code LSB Input (CODE_BIN_LSB)	32	RO	0000_0000
18C	Binary Code MSB Input (CODE_BIN_MSB)	32	See section	0000_0000

13.5.1.1.1 Write Lock Out (UPDATELCKOUT)

The Write Lock Out register (UPDATELCKOUT) can be used to prevent write access to all SYSCTL registers except UPDATELCKOUT.

Offset

Register	Offset
UPDATELCKOUT	0h

Diagram

Fields

Field	Description
31-1 —	Reserved
0 UPDATELCKOUT	Lock Out Setting UPDATELCKOUT prevents write access to all registers other than the UPDATELCKOUT register itself. 0 - Normal Mode: Allow writes to all registers. 1 - Protected Mode: Do not allow writes to all registers except UPDATELCKOUT.

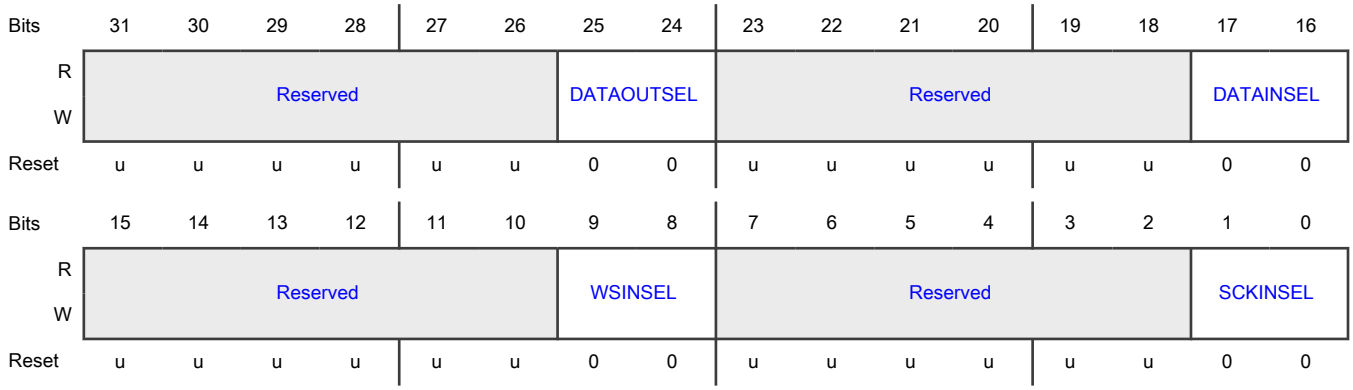
13.5.1.1.2 Shared Signal Select for Flexcomm n (FC0CTRLSEL - FC7CTRLSEL)

Each Shared Signal Select for Flexcomm *n* register (FC*n*CTRLSEL) selects the SCK, WS, DATA input, and DATA output signal sources for the corresponding Flexcomm *n*, excluding Flexcomm 3.

Offset

Register	Offset
FC0CTRLSEL	40h
FC1CTRLSEL	44h
FC2CTRLSEL	48h
FC4CTRLSEL	50h
FC5CTRLSEL	54h
FC6CTRLSEL	58h
FC7CTRLSEL	5Ch

Diagram



Fields

Field	Description
31-26 —	Reserved
25-24 DATAOUTSEL	DATA Output Select Selects the source for DATA output. 00 - Selects the dedicated FCn_RXD_SDA_MOSI_DATA output 01 - Selects from shared signal set 0 (SHAREDCTRLSET0) 10 - Selects from shared signal set 1 (SHAREDCTRLSET1) 11 - Reserved
23-18 —	Reserved
17-16 DATAINSEL	DATA Input Select Selects the source for DATA input. 00 - Selects the dedicated FCn_RXD_SDA_MOSI_DATA input 01 - Selects from shared signal set 0 (SHAREDCTRLSET0) 10 - Selects from shared signal set 1 (SHAREDCTRLSET1) 11 - Reserved
15-10 —	Reserved
9-8 WSINSEL	WS Input Select Selects the source for the WS input. 00 - Selects the dedicated FCn_TXD_SCL_MISO_WS signal 01 - Selects from shared signal set 0 (SHAREDCTRLSET0)

Table continues on the next page...

Table continued from the previous page...

Field	Description
	10 - Selects from shared signal set 1 (SHAREDCTRLSET1) 11 - Reserved
7-2 —	Reserved
1-0 SCKINSEL	SCK Input Select Selects the source for the SCK input. 00 - Selects the dedicated FCn_SCK signal 01 - Selects from shared signal set 0 (SHAREDCTRLSET0) 10 - Selects from shared signal set 1 (SHAREDCTRLSET1) 11 - Reserved

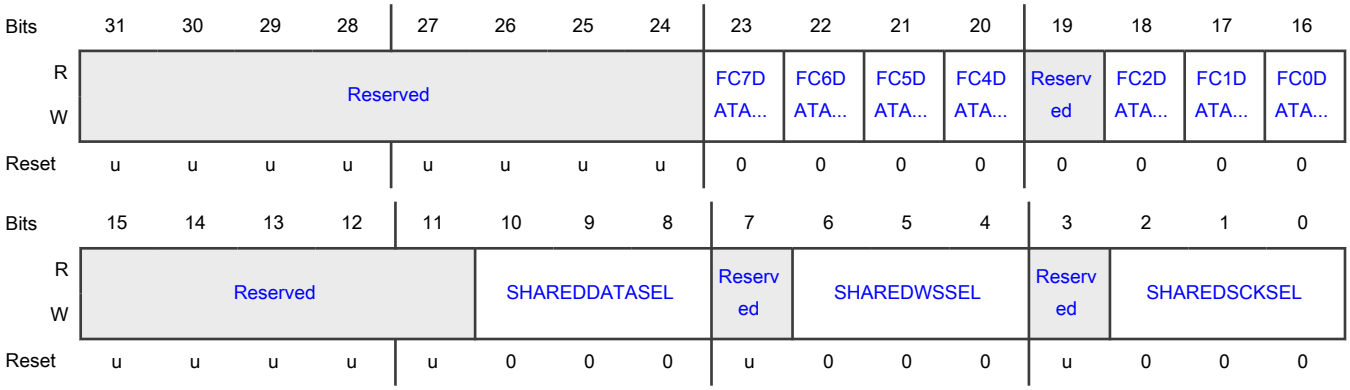
13.5.1.1.3 Shared Signal Set n (SHAREDCTRLSET0 - SHAREDCTRLSET1)

The Shared Signal Set *n* registers (SHAREDCTRLSET*n*) select the sources of SCK, WS, and DATA input for the two sets of shared signals, and selects which Flexcomm modules contribute to shared DATA outputs.

Offset

Register	Offset
SHAREDCTRLSET0	80h
SHAREDCTRLSET1	84h

Diagram



Fields

Field	Description
31-24 —	Reserved
23 FC7DATAOUTEN	DATAOUT Enable for Flexcomm 7 Controls the contribution to SHAREDDATAOUT from Flexcomm 7. 0 - Does not contribute 1 - Contributes
22 FC6DATAOUTEN	DATAOUT Enable for Flexcomm 6 Controls the contribution to SHAREDDOUT from Flexcomm 6. 0 - Does not contribute 1 - Contributes
21 FC5DATAOUTEN	DATAOUT Enable for Flexcomm 5 Controls the contribution to SHAREDDATAOUT from Flexcomm 5. 0 - Does not contribute 1 - Contributes
20 FC4DATAOUTEN	DATAOUT Enable for Flexcomm 4 Controls the contribution to SHAREDDATAOUT from Flexcomm 4. 0 - Does not contribute 1 - Contributes
19 —	Reserved
18 FC2DATAOUTEN	DATAOUT Enable for Flexcomm 2 Controls the contribution to SHAREDDATAOUT from Flexcomm 2. 0 - Does not contribute 1 - Contributes
17 FC1DATAOUTEN	DATAOUT Enable for Flexcomm 1 Controls the contribution to SHAREDDATAOUT from Flexcomm 1. 0 - Does not contribute 1 - Contributes
16 FC0DATAOUTEN	DATAOUT Enable for Flexcomm 0 Controls the contribution to SHAREDDATAOUT from Flexcomm 0. 0 - Does not contribute

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Contributes
15-11 —	Reserved
10-8 SHAREDDATASEL	DATA Input Source Select Selects which Flexcomm module provides the source for DATA input. 000 - Flexcomm 0 001 - Flexcomm 1 010 - Flexcomm 2 011 - Reserved 100 - Flexcomm 4 101 - Flexcomm 5 110 - Flexcomm 6 111 - Flexcomm 7
7 —	Reserved
6-4 SHAREDWSSEL	WS Source Select Selects which Flexcomm module provides the source for WS. 000 - Flexcomm 0 001 - Flexcomm 1 010 - Flexcomm 2 011 - Reserved 100 - Flexcomm 4 101 - Flexcomm 5 110 - Flexcomm 6 111 - Flexcomm 7
3 —	Reserved
2-0 SHAREDSCKSEL	SCK Source Select Selects which Flexcomm module provides the source for SCK. 000 - Flexcomm 0 001 - Flexcomm 1 010 - Flexcomm 2

Table continues on the next page...

Table continued from the previous page...

Field	Description
	011 - Reserved 100 - Flexcomm 4 101 - Flexcomm 5 110 - Flexcomm 6 111 - Flexcomm 7

13.5.1.1.4 Gray Code LSB Input (CODE_GRAY_LSB)

The Gray Code LSB Input register (CODE_GRAY_LSB) contains the least-significant portion of the Gray code to be converted back to binary.

Offset

Register	Offset
CODE_GRAY_LSB	180h

Diagram

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

Fields

Field	Description
31-0 CODE_GRAY_LSB	Gray code (least-significant) CODE_GRAY_LSB is the least-significant 32 bits of the 42-bit Gray code to be converted.

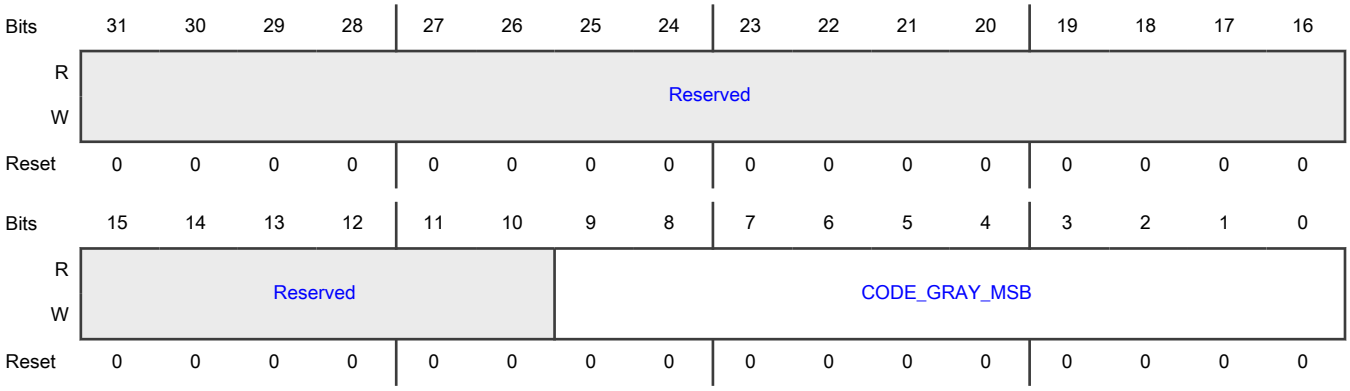
13.5.1.1.5 Gray Code MSB Input (CODE_GRAY_MSB)

The Gray Code MSB Input register (CODE_GRAY_MSB) contains the most-significant portion of the Gray code to be converted back to binary.

Offset

Register	Offset
CODE_GRAY_MSB	184h

Diagram



Fields

Field	Description
31-10 —	Reserved
9-0 CODE_GRAY_MSB	Gray code (most-significant) CODE_GRAY_MSB is the most-significant 10 bits of the 42-bit Gray code to be converted.

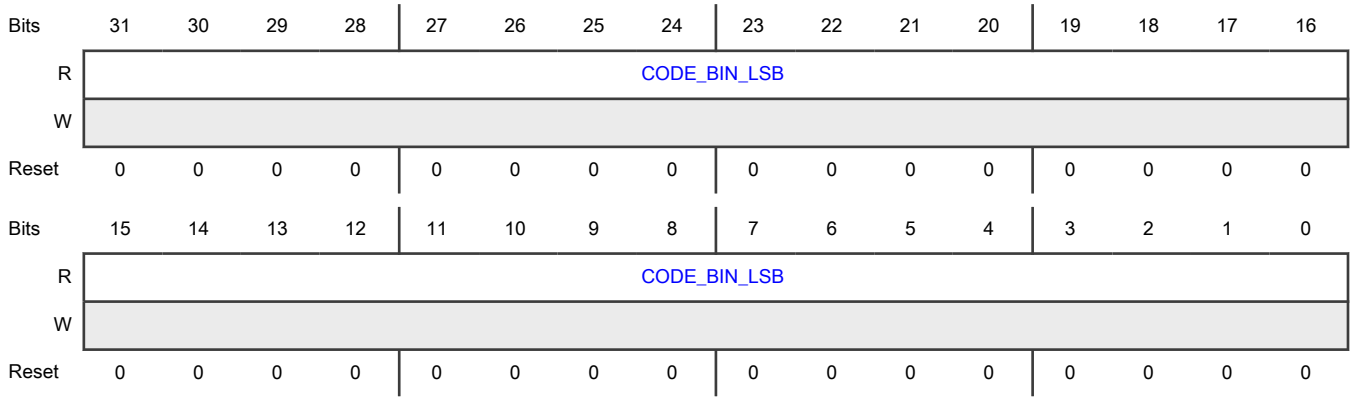
13.5.1.1.6 Binary Code LSB Input (CODE_BIN_LSB)

The Binary Code LSB Input register (CODE_BIN_LSB) contains the least-significant portion of the code converted from Gray to binary coding.

Offset

Register	Offset
CODE_BIN_LSB	188h

Diagram



Fields

Field	Description
31-0 CODE_BIN_LSB	Binary converted code (least-significant) CODE_BIN_LSB is the least-significant 32 bits of the 42-bit converted code.

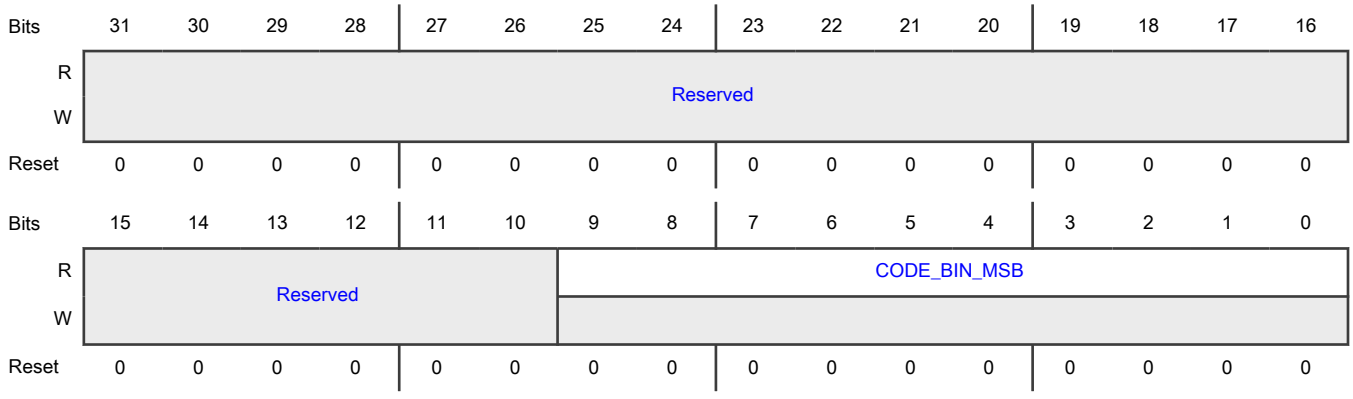
13.5.1.1.7 Binary Code MSB Input (CODE_BIN_MSB)

The Binary Code MSB Input register (CODE_BIN_MSB) contains the most-significant portion of the code converted from Gray to binary coding.

Offset

Register	Offset
CODE_BIN_MSB	18Ch

Diagram



Fields

Field	Description
31-10 —	Reserved
9-0 CODE_BIN_MSB	Binary converted code (most-significant) CODE_BIN_MSB is the most-significant 10 bits of the 42-bit converted code.

Chapter 14
Input Multiplexing (INPUTMUX)

14.1 Chip-specific INPUTMUX information

Table 39. Reference links to related information

Topic	Related module	Reference
Full description	INPUTMUX	INPUTMUX
System memory map		System memory map
Clocking		Clock generation
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

Once set up, no clocks are required for the input multiplexer to function. The system clock is needed only to write to, or read from the INPUTMUX registers. Once the input multiplexer is configured, disable the clock to the INPUTMUX module in the AHBCLKCTRL register.

14.1.1 Module instances

This device has one instance of the INPUTMUX module, INPUTMUX0.

14.2 Overview

The Input Multiplexing module (INPUTMUX) provides signal routing options for internal peripherals. Some peripheral inputs are multiplexed to multiple input sources. The sources can be external pins, interrupts, output signals of other peripherals, or other internal signals.

NOTE

Depending on the package, not all inputs from external pins may be available.

14.2.1 Block Diagram

Figure 24 shows a generic input multiplexer arrangement with n inputs.

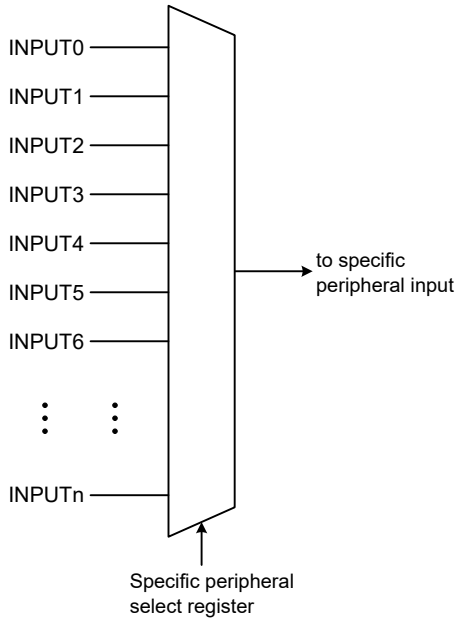


Figure 24. Generic input multiplexing

14.2.2 Features

- Configures the inputs to the SCT.
- Configures the inputs to the asynchronous CTimers.
- Configures the inputs to the pin interrupt module and pattern match engine.
- Configures the inputs to the DMA triggers.
- Enables the inputs to the DMA requests.
- Configures the inputs to CMP, ADC, DAC, ENC, PWM, PWM EXT clock, AOI, external trigger.
- Configures the inputs to the frequency measure function in the ANACTL module. This function is controlled by the FREQMECTRL register (See ANACTRL).

14.3 Functional description

Input multiplexers for most peripherals consist of one or more multiplexers that choose one of several inputs to be routed to a specific input of that peripheral. For example, each CTimer has four capture inputs, each of which has an input multiplexer that can select from among a number of pin functions and some internal signals.

Some peripherals have a more complex arrangement and have detailed figures. See SCT0 input multiplexing for SCT/PWM input multiplexing, Pin interrupt input multiplexing for Pin interrupt (PINT) multiplexing, and DMA trigger input multiplexing for DMA trigger multiplexing.

14.3.1 SCT0 input multiplexing

The input multiplexing for the SCT0 timer multiplexes between 59 internal or external sources for each of its 7 outputs. These outputs with the pll_clk are the 8 inputs of the SCTIMER. Figure 25 shows the detail of this multiplexing.

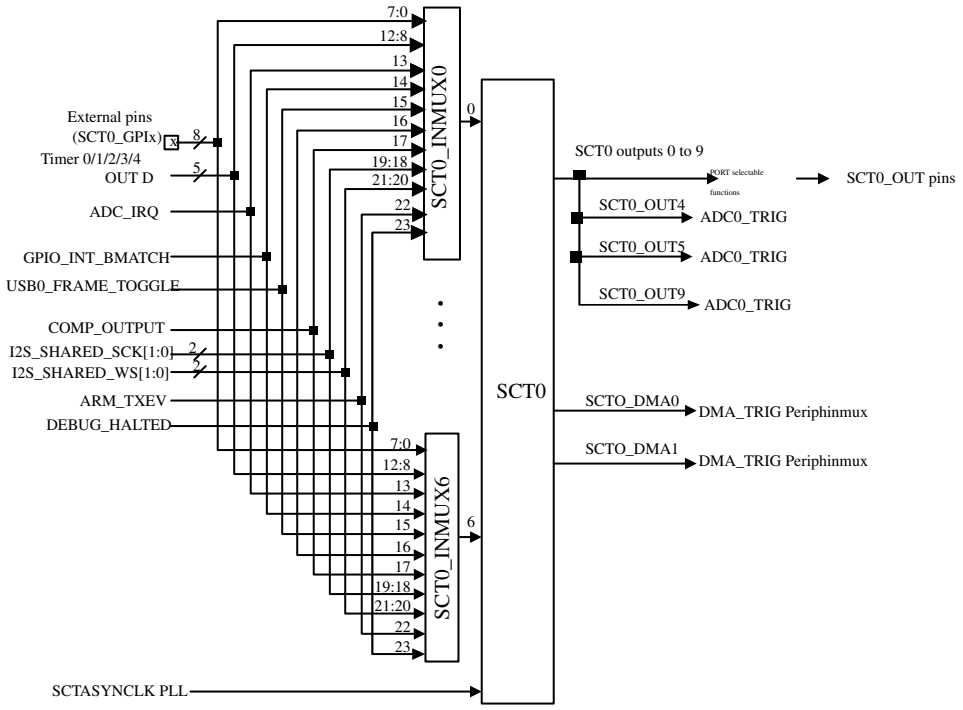


Figure 25. SCT0 input multiplexing

14.3.2 Pin interrupt input multiplexing

The input multiplexing for the pin interrupts and pattern match engine multiplexes all existing pins from ports 0 and 1.

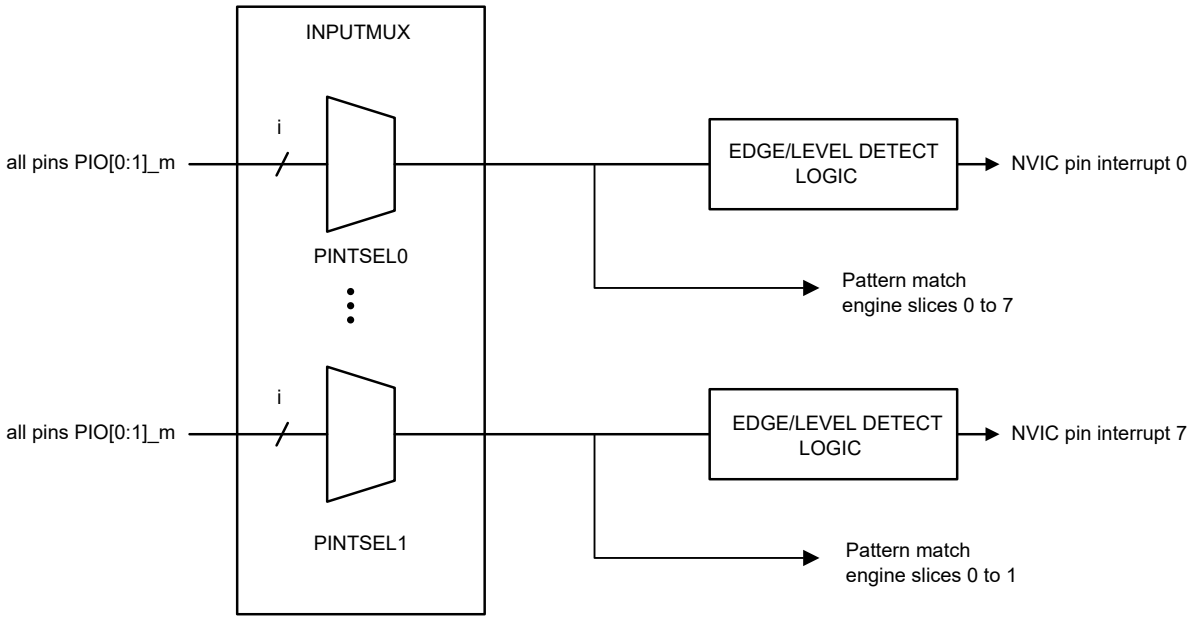


Figure 26. Pin interrupt multiplexing

14.3.3 DMA trigger input multiplexing

The trigger input multiplexing for each DMA controller is configured as shown in Figure 27. In each DMA controller, four of these input triggers are selected from the DMA trigger outputs, controlled by the DMA_OTRIG_INMUX registers. See DMA0_OTRIG_INMUX and DMA1_OTRIG_INMUX.

The potential trigger selections for DMA0 are shown inDMA0_ITRIG_INMUX. The potential trigger selections for DMA1 are shown in DMA1_ITRIG_INMUX.

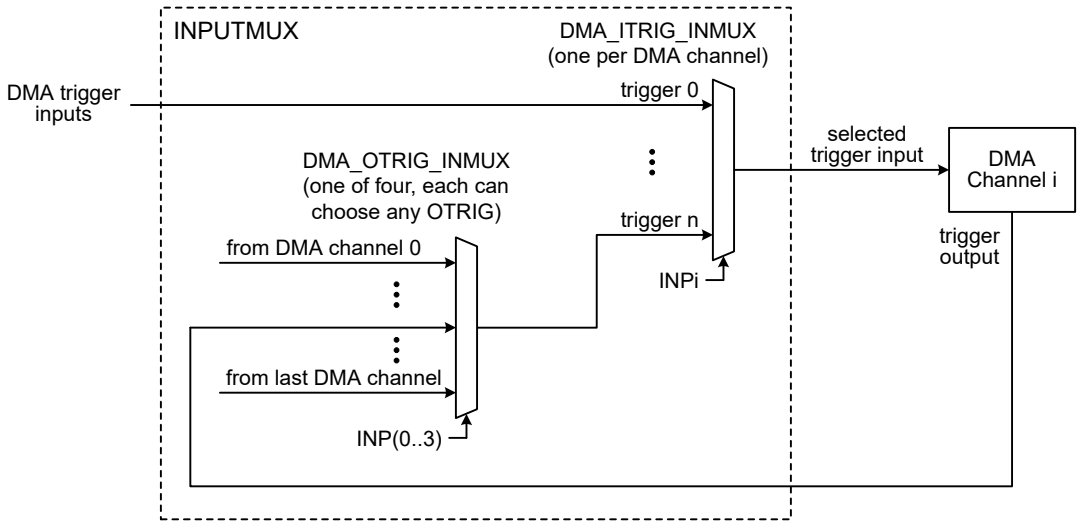


Figure 27. DMA trigger input multiplexing

The two DMA controllers, DMA0 and DMA1, each receive the same DMA requests from peripherals on the first 10 requests. DMA request enables are provided to allow controlling which DMA controller (if any) receives each request.

14.4 External Signals

The INPUTMUX has no dedicated pins. However, all digital pins of ports 0 and 1 can be selected as inputs to the pin interrupts. Multiplexer inputs from external pins work independently of any other function assigned to the pin as long as no analog function is enabled.

Table 40. INPUTMUX pin description

Pins	Peripheral	Section
Any existing pin on port 0 or 1	Pin interrupts 0 to 7	See INPUTMUX.
PIO0_11, PIO0_12, PIO1_4	Frequency measure module	See INPUTMUX.
SCT0_GPI [0:7] pin functions selected from IOCON register	SCTimer/PWM	See SCTIMER.

14.5 Memory map and register definition

This section includes the INPUTMUX module memory map and detailed descriptions of all registers.

14.5.1 Input multiplexing (INPUTMUX) register descriptions

14.5.1.1 INPUTMUX memory map

INPUTMUX0 base address: 4000_6000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0 - 18	Inputmux register for SCT0 input (SCT0_INMUX0 - SCT0_INMUX6)	32	See section	0000_003F
20 - 2C	Capture select register for TIMER0 inputs (TIMER0CAP0 - TIMER0CAP3)	32	See section	0000_003F
30	Trigger register for TIMER0 (TIMER0TRIG)	32	See section	0000_003F
40 - 4C	Capture select register for TIMER1 inputs (TIMER1CAP0 - TIMER1CAP3)	32	See section	0000_003F
50	Trigger register for TIMER1 (TIMER1TRIG)	32	See section	0000_003F
60 - 6C	Capture select register for TIMER2 inputs (TIMER2CAP0 - TIMER2CAP3)	32	See section	0000_003F
70	Trigger register for TIMER2 (TIMER2TRIG)	32	See section	0000_003F
C0 - DC	Pin interrupt select (PINTSEL0 - PINTSEL7)	32	See section	0000_007F
E0 - 15C	Trigger select for DMA0 channel (DMA0_ITRIG_INMUX0 - DMA0_ITRIG_INMUX31)	32	See section	0000_003F
160 - 178	DMA0 output trigger selection for DMA0 input trigger (DMA0_OTRIG_INMUX0 - DMA0_OTRIG_INMUX6)	32	See section	See section
180	Selection for frequency measurement reference clock (FREQMEAS_REF)	32	See section	See section
184	Selection for frequency measurement target clock (FREQMEAS_TAR)	32	See section	See section
1A0 - 1AC	Capture select register for TIMER3 inputs (TIMER3CAP0 - TIMER3CAP3)	32	See section	0000_003F
1B0	Trigger register for TIMER3 (TIMER3TRIG)	32	See section	0000_003F
1C0 - 1CC	Capture select register for TIMER4 inputs (TIMER4CAP0 - TIMER4CAP3)	32	See section	See section
1D0	Trigger register for TIMER4 (TIMER4TRIG)	32	See section	0000_003F
200 - 23C	Trigger select for DMA1 channel (DMA1_ITRIG_INMUX0 - DMA1_ITRIG_INMUX15)	32	See section	See section
240 - 24C	DMA1 output trigger selection for DMA1 input trigger (DMA1_OTRIG_INMUX0 - DMA1_OTRIG_INMUX3)	32	See section	See section

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
260	Input connections for HSCMP0 (HSCMP0_TRIG)	32	See section	0000_003F
280 - 28C	ADC0 Trigger input connections (ADC0_TRIG0 - ADC0_TRIG3)	32	See section	0000_003F
2C0 - 2CC	ADC1 Trigger input connections (ADC1_TRIG0 - ADC1_TRIG3)	32	See section	0000_003F
300	DAC0 Trigger Inputs (DAC0_TRIG)	32	See section	0000_001F
320	DAC1 Trigger Inputs (DAC1_TRIG)	32	See section	0000_001F
340	DAC2 Trigger Inputs (DAC2_TRIG)	32	See section	0000_001F
360	ENC0 Trigger Input Connections (ENC0_TRIG)	32	See section	0000_003F
364	ENC0 Input Connections (ENC0_HOME)	32	See section	0000_003F
368	ENC0 Input Connections (ENC0_INDEX)	32	See section	0000_003F
36C	ENC0 Input Connections (ENC0_PHASEB)	32	See section	0000_003F
370	ENC0 Input Connections (ENC0_PHASEA)	32	See section	0000_003F
380	ENC1 Trigger Input Connections (ENC1_TRIG)	32	See section	0000_003F
384	ENC1 Input Connections (ENC1_HOME)	32	See section	0000_003F
388	ENC1 Input Connections (ENC1_INDEX)	32	See section	0000_003F
38C	ENC1 Input Connections (ENC1_PHASEB)	32	See section	0000_003F
390	ENC1 Input Connections (ENC1_PHASEA)	32	See section	0000_003F
3A0 - 3AC	PWM0 external synchronization (PWM0_EXTSYNC0 - PWM0_EXTSYNC3)	32	See section	0000_003F
3B0 - 3BC	PWM0 input trigger connections (PWM0_EXTA0 - PWM0_EXTA3)	32	See section	0000_003F

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
3C0	PWM0 external force trigger connections (PWM0_EXTFORCE)	32	See section	0000_003F
3C4 - 3D0	PWM0 fault input trigger connections (PWM0_FAULT0 - PWM0_FAULT3)	32	See section	0000_003F
3E0 - 3EC	PWM1 external synchronization (PWM1_EXTSYNC0 - PWM1_EXTSYNC3)	32	See section	0000_003F
3F0 - 3FC	PWM1 input trigger connections (PWM1_EXTA0 - PWM1_EXTA3)	32	See section	0000_003F
400	PWM1 external force trigger connections (PWM1_EXTFORCE)	32	See section	0000_003F
404 - 410	PWM1 fault input trigger connections (PWM1_FAULT0 - PWM1_FAULT3)	32	See section	0000_003F
420	PWM0 external clock trigger connections (PWM0_EXTCLK)	32	See section	0000_0007
424	PWM1 external clock trigger connections (PWM1_EXTCLK)	32	See section	0000_0007
440 - 47C	AOI0 trigger inputs (AOI0_IN0 - AOI0_IN15)	32	See section	0000_003F
480 - 4BC	AOI1 trigger inputs (AOI1_IN0 - AOI1_IN15)	32	See section	0000_003F
4C0 - 4DC	AOI External Trigger Inputs (AOI_EXT_TRIG0 - AOI_EXT_TRIG7)	32	See section	0000_001F
4E0	Input connections for HSCMP1 (HSCMP1_TRIG)	32	See section	0000_003F
500	Input connections for HSCMP2 (HSCMP2_TRIG)	32	See section	0000_003F
520 - 56C	Trigger select for DMA0 channel (DMA0_ITRIG_INMUX32 - DMA0_ITRIG_INMUX51)	32	See section	0000_003F
740	Enable DMA0 requests (DMA0_REQEN0)	32	RW	FFFF_FFFF
744	Enable DMA0 requests (DMA0_REQEN1)	32	See section	See section
748	Set bits in DMA0_REQEN0 register (DMA0_REQEN0_SET)	32	WO	See section
74C	Set bits in DMA0_REQEN1 register (DMA0_REQEN1_SET)	32	See section	See section
750	Clear bits in DMA0_REQEN0 register (DMA0_REQEN0_CLR)	32	WO	See section
754	Clear bits in DMA0_REQEN1 register (DMA0_REQEN1_CLR)	32	See section	See section

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
760	Enable DMA1 requests (DMA1_REQEN)	32	See section	See section
768	Set bits in DMA1_REQEN register (DMA1_REQEN_SET)	32	See section	See section
770	Clear bits in DMA1_REQEN register (DMA1_REQEN_CLR)	32	See section	See section
780	Enable DMA0 triggers (DMA0_ITRIGEN0)	32	RW	FFFF_FFFF
784	Enable DMA0 triggers (DMA0_ITRIGEN1)	32	See section	000F_FFFF
788	Set bits in DMA0_ITRIGEN0 register (DMA0_ITRIGEN0_SET)	32	WO	See section
78C	Set bits in DMA0_ITRIGEN1 register (DMA0_ITRIGEN1_SET)	32	See section	See section
790	Clear bits in DMA0_ITRIGEN0 register (DMA0_ITRIGEN0_CLR)	32	WO	See section
794	Clear bits in DMA0_ITRIGEN1 register (DMA0_ITRIGEN1_CLR)	32	See section	See section
7A0	Enable DMA1 triggers (DMA1_ITRIGEN)	32	See section	0000_FFFF
7A8	Set bits in DMA1_ITRIGEN register (DMA1_ITRIGEN_SET)	32	See section	See section
7B0	Clear bits in DMA1_ITRIGEN register (DMA1_ITRIGEN_CLR)	32	See section	See section

14.5.1.1.1 Inputmux register for SCT0 input (SCT0_INMUX0 - SCT0_INMUX6)

This register selects one input source for each SCT0 input from 24 external and internal sources. (An exception is SCT0 input SCT0_IN7, which is directly connected to the SCTASYNCCLK PLL clock and not multiplexed with any other signals.)

The output of SCT0 Input multiplexing register 0 selects the source for SCT0 input 0. The output of SCT0 Input multiplexing register 1 selects the source for SCT0 input 1, and so forth up to SCT0 Input multiplexing register 6, which selects the input for SCT0 input 6.

Offset

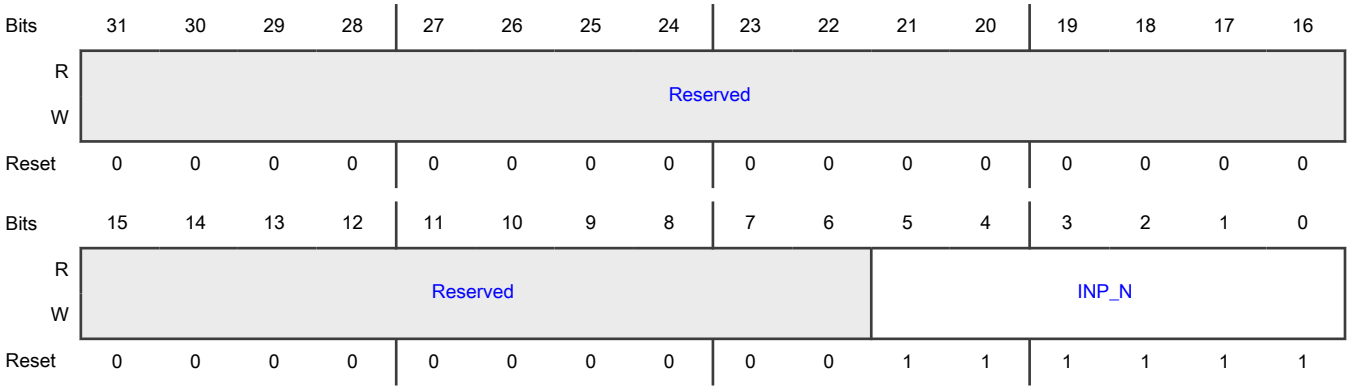
Register	Offset
SCT0_INMUX0	0h
SCT0_INMUX1	4h
SCT0_INMUX2	8h
SCT0_INMUX3	Ch

Table continues on the next page...

Table continued from the previous page...

Register	Offset
SCT0_INMUX4	10h
SCT0_INMUX5	14h
SCT0_INMUX6	18h

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 INP_N	Input number to SCT0 inputs 0 to 6. 00_0000 - SCT_GPIO_IN_A function selected from IOCON register 00_0001 - SCT_GPIO_IN_B function selected from IOCON register 00_0010 - SCT_GPIO_IN_C function selected from IOCON register 00_0011 - SCT_GPIO_IN_D function selected from IOCON register 00_0100 - SCT_GPIO_IN_E function selected from IOCON register 00_0101 - SCT_GPIO_IN_F function selected from IOCON register 00_0110 - SCT_GPIO_IN_G function selected from IOCON register 00_0111 - SCT_GPIO_IN_H function selected from IOCON register 00_1000 - T0_MAT0 ctimer 0 match[0] output 00_1001 - T1_MAT0 ctimer 1 match[0] output 00_1010 - T2_MAT0 ctimer 2 match[0] output 00_1011 - T3_MAT0 ctimer 3 match[0] output 00_1100 - T4_MAT0 ctimer 4 match[0] output 00_1101 - ADC0_IRQ interrupt request from ADC0

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00_1110 - GPIOINT_BMATCH 00_1111 - USB0_FRAME_TOGGLE 01_0000 - Reserved 01_0001 - ACMP0_OUT from analog comparator 01_0010 - SHARED_I2S_SCLK0 output from I2S pin sharing 01_0011 - SHARED_I2S_SCLK1 output from I2S pin sharing 01_0100 - SHARED_I2S_WS0 output from I2S pin sharing 01_0101 - SHARED_I2S_WS1 output from I2S pin sharing 01_0110 - ARM_TXEV interrupt event from CPU0 01_0111 - DEBUG_HALTED from CPU0 01_1000 - ADC1_IRQ interrupt request from ADC1 01_1001 - ADC0_tcomp[0] 01_1010 - ADC0_tcomp[1] 01_1011 - ADC0_tcomp[2] 01_1100 - ADC0_tcomp[3] 01_1101 - ADC1_tcomp[0] 01_1110 - ADC1_tcomp[1] 01_1111 - ADC1_tcomp[2] 10_0000 - ADC1_tcomp[3] 10_0001 - HSCMP0_OUT 10_0010 - HSCMP1_OUT 10_0011 - HSCMP2_OUT 10_0100 - PWM0_SM0_MUX_TRIG0 | PWM0_SM0_MUX_TRIG1 10_0101 - PWM0_SM1_MUX_TRIG0 | PWM0_SM1_MUX_TRIG1 10_0110 - PWM0_SM2_MUX_TRIG0 | PWM0_SM2_MUX_TRIG1 10_0111 - PWM0_SM3_MUX_TRIG0 | PWM0_SM3_MUX_TRIG1 10_1000 - PWM1_SM0_MUX_TRIG0 | PWM1_SM0_MUX_TRIG1 10_1001 - PWM1_SM1_MUX_TRIG0 | PWM1_SM1_MUX_TRIG1 10_1010 - PWM1_SM2_MUX_TRIG0 | PWM1_SM2_MUX_TRIG1 10_1011 - PWM1_SM3_MUX_TRIG0 | PWM1_SM3_MUX_TRIG1 10_1100 - ENC0_CMP/POS_MATCH 10_1101 - ENC1_CMP/POS_MATCH 10_1110 - AOI0_OUT0 10_1111 - AOI0_OUT1

Table continues on the next page...

Table continued from the previous page...

Field	Description
	11_0000 - AOI0_OUT2 11_0001 - AOI0_OUT3 11_0010 - AOI1_OUT0 11_0011 - AOI1_OUT1 11_0100 - AOI1_OUT2 11_0101 - AOI1_OUT3 11_0110 - FC3_SCK 11_0111 - FC3_RXD_SDA_MOSI_DATA 11_1000 - FC3_TXD_SCL_MISO_WS 11_1001 - FC3_CTS_DSA_SSEL0 11_1010 - TMPR_OUT 11_1011-11_1111 - None

14.5.1.1.2 Capture select register for TIMER0 inputs (TIMER0CAP0 - TIMER0CAP3)

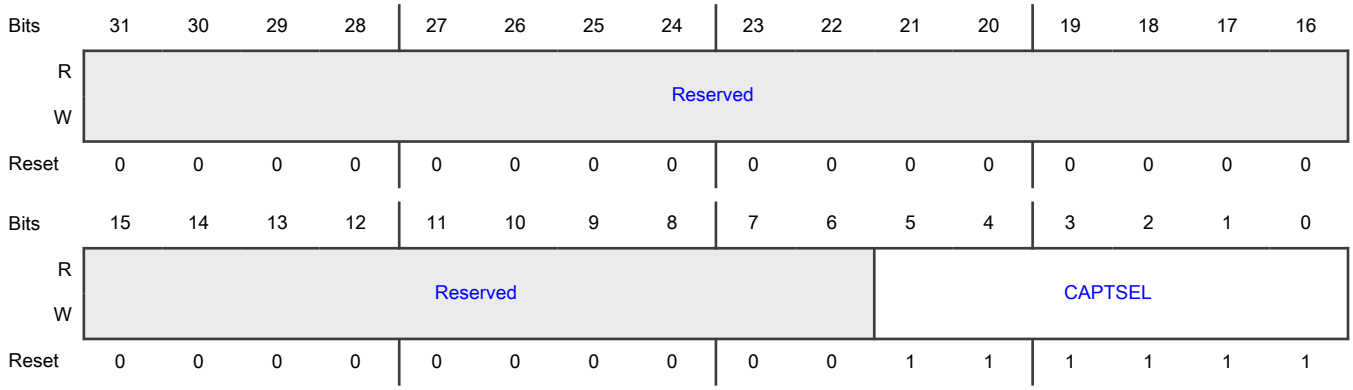
For each of the 5 standard timers, numbered i = 0 to 4 there are 4 TIMERiCAPTj, with j = 0 to 3, each allowing selecting between 25 external or internal input sources.

The output of TIMER0CAPT0 Input multiplexing register 0 selects the source for TIMER0 capture input 0. The output of TIMER0CAPT1 Input multiplexing register 1 selects the source for TIMER0 capture input 1, and so forth up to TIMER4CAPT3 Input multiplexing register 3, which selects the input for TIMER4 capture input 3.

Offset

Register	Offset
TIMER0CAP0	20h
TIMER0CAP1	24h
TIMER0CAP2	28h
TIMER0CAP3	2Ch

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 CAPTSEL	Input number to TIMER0 capture inputs 0 to 5 00_0000 - CTIMER_INP0 function selected from IOCON register 00_0001 - CTIMER_INP1 function selected from IOCON register 00_0010 - CTIMER_INP2 function selected from IOCON register 00_0011 - CTIMER_INP3 function selected from IOCON register 00_0100 - CTIMER_INP4 function selected from IOCON register 00_0101 - CTIMER_INP5 function selected from IOCON register 00_0110 - CTIMER_INP6 function selected from IOCON register 00_0111 - CTIMER_INP7 function selected from IOCON register 00_1000 - CTIMER_INP8 function selected from IOCON register 00_1001 - CTIMER_INP9 function selected from IOCON register 00_1010 - CTIMER_INP10 function selected from IOCON register 00_1011 - CTIMER_INP11 function selected from IOCON register 00_1100 - CTIMER_INP12 function selected from IOCON register 00_1101 - CTIMER_INP13 function selected from IOCON register 00_1110 - CTIMER_INP14 function selected from IOCON register 00_1111 - CTIMER_INP15 function selected from IOCON register 01_0000 - CTIMER_INP16 function selected from IOCON register 01_0001 - CTIMER_INP17 function selected from IOCON register 01_0010 - CTIMER_INP18 function selected from IOCON register 01_0011 - CTIMER_INP19 function selected from IOCON register

Table continues on the next page...

Table continued from the previous page...

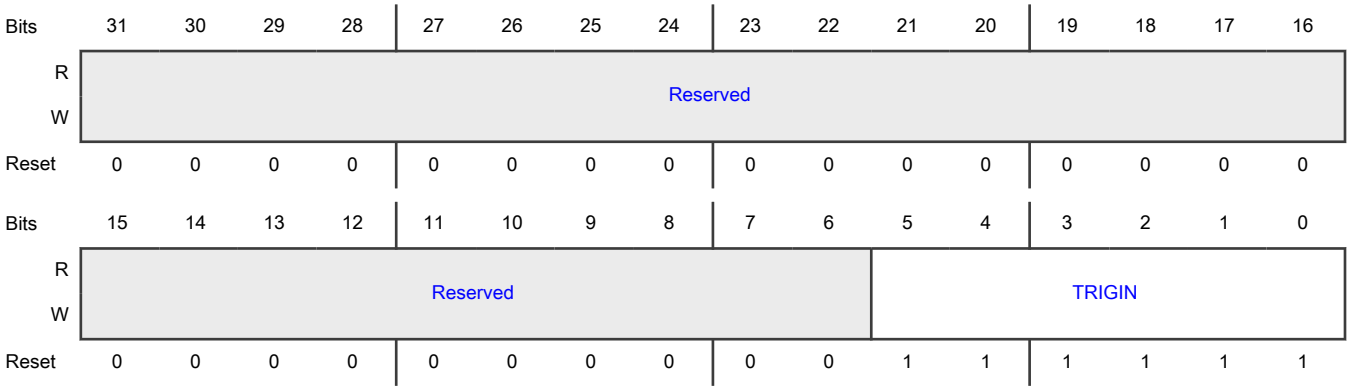
Field	Description
	01_0100 - USB0_FRAME_TOGGLE 01_0101 - Reserved 01_0110 - ACMP0_OUT from analog comparator 01_0111 - SHARED_I2S_WS0 output from I2S pin sharing 01_1000 - SHARED_I2S_WS1 output from I2S pin sharing 01_1001 - ADC0_IRQ 01_1010 - ADC1_IRQ 01_1011 - HSCMP0_OUT 01_1100 - HSCMP1_OUT 01_1101 - HSCMP2_OUT 01_1110 - PWM0_SM0_MUX_TRIG0 | PWM0_SM0_MUX_TRIG1 01_1111 - PWM0_SM1_MUX_TRIG0 | PWM0_SM1_MUX_TRIG1 10_0000 - PWM0_SM2_MUX_TRIG0 | PWM0_SM2_MUX_TRIG1 10_0001 - PWM0_SM3_MUX_TRIG0 | PWM0_SM3_MUX_TRIG1 10_0010 - PWM1_SM0_MUX_TRIG0 | PWM1_SM0_MUX_TRIG1 10_0011 - PWM1_SM1_MUX_TRIG0 | PWM1_SM1_MUX_TRIG1 10_0100 - PWM1_SM2_MUX_TRIG0 | PWM1_SM2_MUX_TRIG1 10_0101 - PWM1_SM3_MUX_TRIG0 | PWM1_SM3_MUX_TRIG1 10_0110 - ENC0_CMP/POS_MATCH 10_0111 - ENC1_CMP/POS_MATCH 10_1000 - AOI0_OUT0 10_1001 - AOI0_OUT1 10_1010 - AOI0_OUT2 10_1011 - AOI0_OUT3 10_1100 - AOI1_OUT0 10_1101 - AOI1_OUT1 10_1110 - AOI1_OUT2 10_1111 - AOI1_OUT3 11_0000 - TMPR_OUT 11_0001-11_1111 - None

14.5.1.1.3 Trigger register for TIMER0 (TIMER0TRIG)

Offset

Register	Offset
TIMER0TRIG	30h

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 TRIGIN	Input number to TIMER0 trigger inputs 00_0000 - CTIMER_INP0 function selected from IOCON register 00_0001 - CTIMER_INP1 function selected from IOCON register 00_0010 - CTIMER_INP2 function selected from IOCON register 00_0011 - CTIMER_INP3 function selected from IOCON register 00_0100 - CTIMER_INP4 function selected from IOCON register 00_0101 - CTIMER_INP5 function selected from IOCON register 00_0110 - CTIMER_INP6 function selected from IOCON register 00_0111 - CTIMER_INP7 function selected from IOCON register 00_1000 - CTIMER_INP8 function selected from IOCON register 00_1001 - CTIMER_INP9 function selected from IOCON register 00_1010 - CTIMER_INP10 function selected from IOCON register 00_1011 - CTIMER_INP11 function selected from IOCON register 00_1100 - CTIMER_INP12 function selected from IOCON register 00_1101 - CTIMER_INP13 function selected from IOCON register 00_1110 - CTIMER_INP14 function selected from IOCON register 00_1111 - CTIMER_INP15 function selected from IOCON register

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01_0000 - CTIMER_INP16 function selected from IOCON register 01_0001 - CTIMER_INP17 function selected from IOCON register 01_0010 - CTIMER_INP18 function selected from IOCON register 01_0011 - CTIMER_INP19 function selected from IOCON register 01_0100 - USB0_FRAME_TOGGLE 01_0101 - Reserved 01_0110 - ACMP0_OUT from analog comparator 01_0111 - SHARED_I2S_WS0 output from I2S pin sharing 01_1000 - SHARED_I2S_WS1 output from I2S pin sharing 01_1001 - ADC0_IRQ 01_1010 - ADC1_IRQ 01_1011 - HSCMP0_OUT 01_1100 - HSCMP1_OUT 01_1101 - HSCMP2_OUT 01_1110 - PWM0_SM0_MUX_TRIG0 | PWM0_SM0_MUX_TRIG1 01_1111 - PWM0_SM1_MUX_TRIG0 | PWM0_SM1_MUX_TRIG1 10_0000 - PWM0_SM2_MUX_TRIG0 | PWM0_SM2_MUX_TRIG1 10_0001 - PWM0_SM3_MUX_TRIG0 | PWM0_SM3_MUX_TRIG1 10_0010 - PWM1_SM0_MUX_TRIG0 | PWM1_SM0_MUX_TRIG1 10_0011 - PWM1_SM1_MUX_TRIG0 | PWM1_SM1_MUX_TRIG1 10_0100 - PWM1_SM2_MUX_TRIG0 | PWM1_SM2_MUX_TRIG1 10_0101 - PWM1_SM3_MUX_TRIG0 | PWM1_SM3_MUX_TRIG1 10_0110 - ENC0_CMP/POS_MATCH 10_0111 - ENC1_CMP/POS_MATCH 10_1000 - AOI0_OUT0 10_1001 - AOI0_OUT1 10_1010 - AOI0_OUT2 10_1011 - AOI0_OUT3 10_1100 - AOI1_OUT0 10_1101 - AOI1_OUT1 10_1110 - AOI1_OUT2 10_1111 - AOI1_OUT3 11_0000 - TMPR_OUT

Table continues on the next page...

Table continued from the previous page...

Field	Description
	11_0001-11_1111 - None

14.5.1.1.4 Capture select register for TIMER1 inputs (TIMER1CAP0 - TIMER1CAP3)

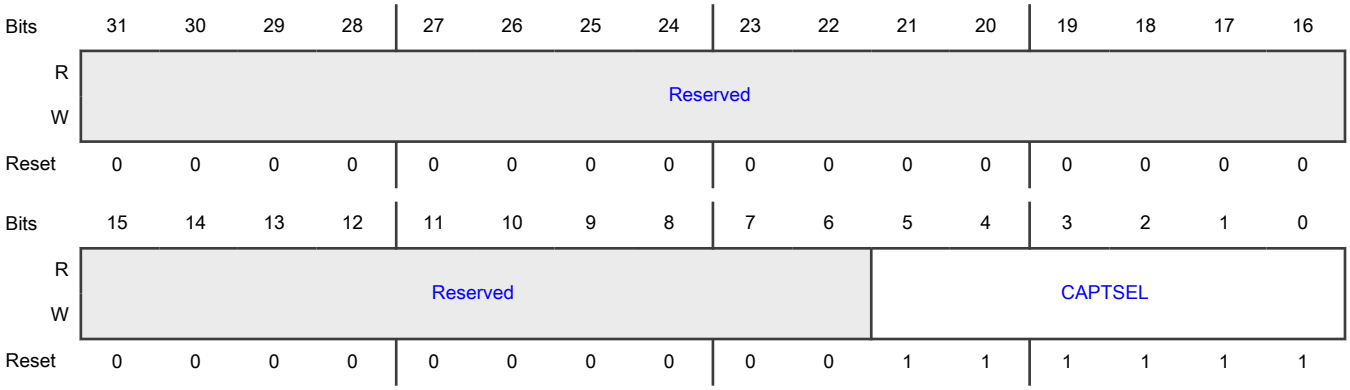
For each of the 5 standard timers, numbered i = 0 to 4 there are 4 TIMERiCAPTj, with j = 0 to 3, each allowing selecting between 25 external or internal input sources.

The output of TIMER0CAPT0 Input multiplexing register 0 selects the source for TIMER0 capture input 0. The output of TIMER0CAPT1 Input multiplexing register 1 selects the source for TIMER0 capture input 1, and so forth up to TIMER4CAPT3 Input multiplexing register 3, which selects the input for TIMER4 capture input 3.

Offset

Register	Offset
TIMER1CAP0	40h
TIMER1CAP1	44h
TIMER1CAP2	48h
TIMER1CAP3	4Ch

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 CAPTSEL	Input number to TIMER1 capture inputs 0 to 5 00_0000 - CTIMER_INP0 function selected from IOCON register 00_0001 - CTIMER_INP1 function selected from IOCON register

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00_0010 - CTIMER_INP2 function selected from IOCON register 00_0011 - CTIMER_INP3 function selected from IOCON register 00_0100 - CTIMER_INP4 function selected from IOCON register 00_0101 - CTIMER_INP5 function selected from IOCON register 00_0110 - CTIMER_INP6 function selected from IOCON register 00_0111 - CTIMER_INP7 function selected from IOCON register 00_1000 - CTIMER_INP8 function selected from IOCON register 00_1001 - CTIMER_INP9 function selected from IOCON register 00_1010 - CTIMER_INP10 function selected from IOCON register 00_1011 - CTIMER_INP11 function selected from IOCON register 00_1100 - CTIMER_INP12 function selected from IOCON register 00_1101 - CTIMER_INP13 function selected from IOCON register 00_1110 - CTIMER_INP14 function selected from IOCON register 00_1111 - CTIMER_INP15 function selected from IOCON register 01_0000 - CTIMER_INP16 function selected from IOCON register 01_0001 - CTIMER_INP17 function selected from IOCON register 01_0010 - CTIMER_INP18 function selected from IOCON register 01_0011 - CTIMER_INP19 function selected from IOCON register 01_0100 - USB0_FRAME_TOGGLE 01_0101 - Reserved 01_0110 - ACMP0_OUT from analog comparator 01_0111 - SHARED_I2S_WS0 output from I2S pin sharing 01_1000 - SHARED_I2S_WS1 output from I2S pin sharing 01_1001 - ADC0_IRQ 01_1010 - ADC1_IRQ 01_1011 - HSCMP0_OUT 01_1100 - HSCMP1_OUT 01_1101 - HSCMP2_OUT 01_1110 - PWM0_SM0_MUX_TRIG0 | PWM0_SM0_MUX_TRIG1 01_1111 - PWM0_SM1_MUX_TRIG0 | PWM0_SM1_MUX_TRIG1 10_0000 - PWM0_SM2_MUX_TRIG0 | PWM0_SM2_MUX_TRIG1 10_0001 - PWM0_SM3_MUX_TRIG0 | PWM0_SM3_MUX_TRIG1 10_0010 - PWM1_SM0_MUX_TRIG0 | PWM1_SM0_MUX_TRIG1 10_0011 - PWM1_SM1_MUX_TRIG0 | PWM1_SM1_MUX_TRIG1

Table continues on the next page...

Table continued from the previous page...

Field	Description
	10_0100 - PWM1_SM2_MUX_TRIG0 | PWM1_SM2_MUX_TRIG1 10_0101 - PWM1_SM3_MUX_TRIG0 | PWM1_SM3_MUX_TRIG1 10_0110 - ENC0_CMP/POS_MATCH 10_0111 - ENC1_CMP/POS_MATCH 10_1000 - AOI0_OUT0 10_1001 - AOI0_OUT1 10_1010 - AOI0_OUT2 10_1011 - AOI0_OUT3 10_1100 - AOI1_OUT0 10_1101 - AOI1_OUT1 10_1110 - AOI1_OUT2 10_1111 - AOI1_OUT3 11_0000 - TMPR_OUT 11_0001-11_1111 - None

14.5.1.1.5 Trigger register for TIMER1 (TIMER1TRIG)

Offset

Register	Offset
TIMER1TRIG	50h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	Reserved										TRIGIN					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Description
31-6 —	Reserved
5-0 TRIGIN	Input number to TIMER1 trigger inputs 00_0000 - CTIMER_INP0 function selected from IOCON register 00_0001 - CTIMER_INP1 function selected from IOCON register 00_0010 - CTIMER_INP2 function selected from IOCON register 00_0011 - CTIMER_INP3 function selected from IOCON register 00_0100 - CTIMER_INP4 function selected from IOCON register 00_0101 - CTIMER_INP5 function selected from IOCON register 00_0110 - CTIMER_INP6 function selected from IOCON register 00_0111 - CTIMER_INP7 function selected from IOCON register 00_1000 - CTIMER_INP8 function selected from IOCON register 00_1001 - CTIMER_INP9 function selected from IOCON register 00_1010 - CTIMER_INP10 function selected from IOCON register 00_1011 - CTIMER_INP11 function selected from IOCON register 00_1100 - CTIMER_INP12 function selected from IOCON register 00_1101 - CTIMER_INP13 function selected from IOCON register 00_1110 - CTIMER_INP14 function selected from IOCON register 00_1111 - CTIMER_INP15 function selected from IOCON register 01_0000 - CTIMER_INP16 function selected from IOCON register 01_0001 - CTIMER_INP17 function selected from IOCON register 01_0010 - CTIMER_INP18 function selected from IOCON register 01_0011 - CTIMER_INP19 function selected from IOCON register 01_0100 - USB0_FRAME_TOGGLE 01_0101 - Reserved 01_0110 - ACMP0_OUT from analog comparator 01_0111 - SHARED_I2S_WS0 output from I2S pin sharing 01_1000 - SHARED_I2S_WS1 output from I2S pin sharing 01_1001 - ADC0_IRQ 01_1010 - ADC1_IRQ 01_1011 - HSCMP0_OUT 01_1100 - HSCMP1_OUT 01_1101 - HSCMP2_OUT

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01_1110 - PWM0_SM0_MUX_TRIG0 | PWM0_SM0_MUX_TRIG1 01_1111 - PWM0_SM1_MUX_TRIG0 | PWM0_SM1_MUX_TRIG1 10_0000 - PWM0_SM2_MUX_TRIG0 | PWM0_SM2_MUX_TRIG1 10_0001 - PWM0_SM3_MUX_TRIG0 | PWM0_SM3_MUX_TRIG1 10_0010 - PWM1_SM0_MUX_TRIG0 | PWM1_SM0_MUX_TRIG1 10_0011 - PWM1_SM1_MUX_TRIG0 | PWM1_SM1_MUX_TRIG1 10_0100 - PWM1_SM2_MUX_TRIG0 | PWM1_SM2_MUX_TRIG1 10_0101 - PWM1_SM3_MUX_TRIG0 | PWM1_SM3_MUX_TRIG1 10_0110 - ENC0_CMP/POS_MATCH 10_0111 - ENC1_CMP/POS_MATCH 10_1000 - AOI0_OUT0 10_1001 - AOI0_OUT1 10_1010 - AOI0_OUT2 10_1011 - AOI0_OUT3 10_1100 - AOI1_OUT0 10_1101 - AOI1_OUT1 10_1110 - AOI1_OUT2 10_1111 - AOI1_OUT3 11_0000 - TMPR_OUT 11_0001-11_1111 - None

14.5.1.1.6 Capture select register for TIMER2 inputs (TIMER2CAP0 - TIMER2CAP3)

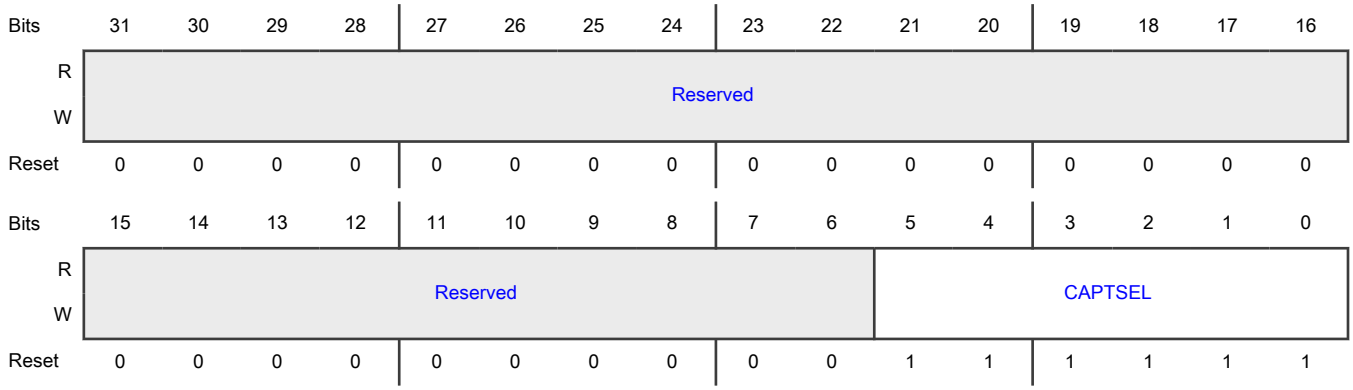
For each of the 5 standard timers, numbered i = 0 to 4 there are 4 TIMERiCAPTj, with j = 0 to 3, each allowing selecting between 25 external or internal input sources.

The output of TIMER0CAPT0 Input multiplexing register 0 selects the source for TIMER0 capture input 0. The output of TIMER0CAPT1 Input multiplexing register 1 selects the source for TIMER0 capture input 1, and so forth up to TIMER4CAPT3 Input multiplexing register 3, which selects the input for TIMER4 capture input 3.

Offset

Register	Offset
TIMER2CAP0	60h
TIMER2CAP1	64h
TIMER2CAP2	68h
TIMER2CAP3	6Ch

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 CAPTSEL	Input number to TIMER2 capture inputs 0 to 5 00_0000 - CTIMER_INP0 function selected from IOCON register 00_0001 - CTIMER_INP1 function selected from IOCON register 00_0010 - CTIMER_INP2 function selected from IOCON register 00_0011 - CTIMER_INP3 function selected from IOCON register 00_0100 - CTIMER_INP4 function selected from IOCON register 00_0101 - CTIMER_INP5 function selected from IOCON register 00_0110 - CTIMER_INP6 function selected from IOCON register 00_0111 - CTIMER_INP7 function selected from IOCON register 00_1000 - CTIMER_INP8 function selected from IOCON register 00_1001 - CTIMER_INP9 function selected from IOCON register 00_1010 - CTIMER_INP10 function selected from IOCON register 00_1011 - CTIMER_INP11 function selected from IOCON register 00_1100 - CTIMER_INP12 function selected from IOCON register 00_1101 - CTIMER_INP13 function selected from IOCON register 00_1110 - CTIMER_INP14 function selected from IOCON register 00_1111 - CTIMER_INP15 function selected from IOCON register 01_0000 - CTIMER_INP16 function selected from IOCON register 01_0001 - CTIMER_INP17 function selected from IOCON register 01_0010 - CTIMER_INP18 function selected from IOCON register 01_0011 - CTIMER_INP19 function selected from IOCON register

Table continues on the next page...

Table continued from the previous page...

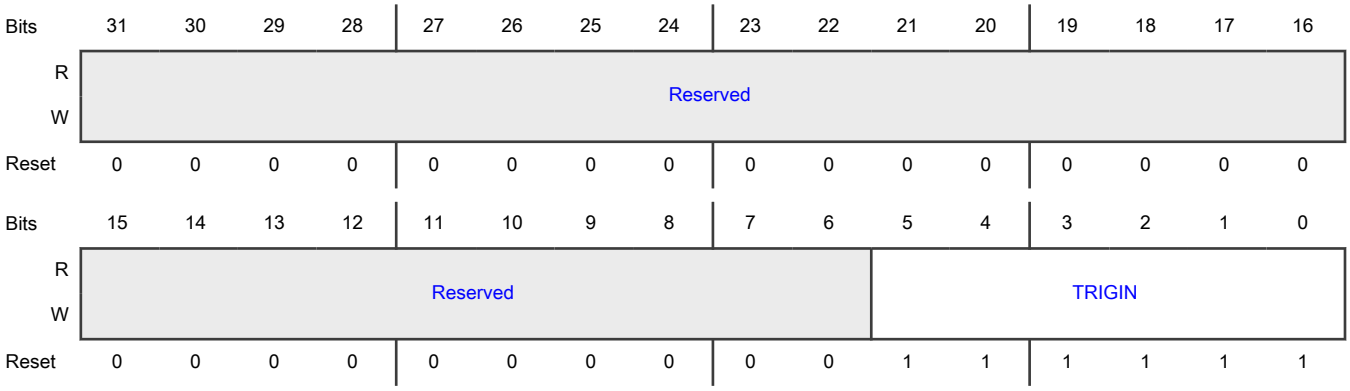
Field	Description
	01_0100 - USB0_FRAME_TOGGLE 01_0101 - Reserved 01_0110 - ACMP0_OUT from analog comparator 01_0111 - SHARED_I2S_WS0 output from I2S pin sharing 01_1000 - SHARED_I2S_WS1 output from I2S pin sharing 01_1001 - ADC0_IRQ 01_1010 - ADC1_IRQ 01_1011 - HSCMP0_OUT 01_1100 - HSCMP1_OUT 01_1101 - HSCMP2_OUT 01_1110 - PWM0_SM0_MUX_TRIG0 | PWM0_SM0_MUX_TRIG1 01_1111 - PWM0_SM1_MUX_TRIG0 | PWM0_SM1_MUX_TRIG1 10_0000 - PWM0_SM2_MUX_TRIG0 | PWM0_SM2_MUX_TRIG1 10_0001 - PWM0_SM3_MUX_TRIG0 | PWM0_SM3_MUX_TRIG1 10_0010 - PWM1_SM0_MUX_TRIG0 | PWM1_SM0_MUX_TRIG1 10_0011 - PWM1_SM1_MUX_TRIG0 | PWM1_SM1_MUX_TRIG1 10_0100 - PWM1_SM2_MUX_TRIG0 | PWM1_SM2_MUX_TRIG1 10_0101 - PWM1_SM3_MUX_TRIG0 | PWM1_SM3_MUX_TRIG1 10_0110 - ENC0_CMP/POS_MATCH 10_0111 - ENC1_CMP/POS_MATCH 10_1000 - AOI0_OUT0 10_1001 - AOI0_OUT1 10_1010 - AOI0_OUT2 10_1011 - AOI0_OUT3 10_1100 - AOI1_OUT0 10_1101 - AOI1_OUT1 10_1110 - AOI1_OUT2 10_1111 - AOI1_OUT3 11_0000 - TMPR_OUT 11_0001-11_1111 - None

14.5.1.1.7 Trigger register for TIMER2 (TIMER2TRIG)

Offset

Register	Offset
TIMER2TRIG	70h

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 TRIGIN	Input number to TIMER2 trigger inputs 00_0000 - CTIMER_INP0 function selected from IOCON register 00_0001 - CTIMER_INP1 function selected from IOCON register 00_0010 - CTIMER_INP2 function selected from IOCON register 00_0011 - CTIMER_INP3 function selected from IOCON register 00_0100 - CTIMER_INP4 function selected from IOCON register 00_0101 - CTIMER_INP5 function selected from IOCON register 00_0110 - CTIMER_INP6 function selected from IOCON register 00_0111 - CTIMER_INP7 function selected from IOCON register 00_1000 - CTIMER_INP8 function selected from IOCON register 00_1001 - CTIMER_INP9 function selected from IOCON register 00_1010 - CTIMER_INP10 function selected from IOCON register 00_1011 - CTIMER_INP11 function selected from IOCON register 00_1100 - CTIMER_INP12 function selected from IOCON register 00_1101 - CTIMER_INP13 function selected from IOCON register 00_1110 - CTIMER_INP14 function selected from IOCON register 00_1111 - CTIMER_INP15 function selected from IOCON register

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01_0000 - CTIMER_INP16 function selected from IOCON register 01_0001 - CTIMER_INP17 function selected from IOCON register 01_0010 - CTIMER_INP18 function selected from IOCON register 01_0011 - CTIMER_INP19 function selected from IOCON register 01_0100 - USB0_FRAME_TOGGLE 01_0101 - Reserved 01_0110 - ACMP0_OUT from analog comparator 01_0111 - SHARED_I2S_WS0 output from I2S pin sharing 01_1000 - SHARED_I2S_WS1 output from I2S pin sharing 01_1001 - ADC0_IRQ 01_1010 - ADC1_IRQ 01_1011 - HSCMP0_OUT 01_1100 - HSCMP1_OUT 01_1101 - HSCMP2_OUT 01_1110 - PWM0_SM0_MUX_TRIG0 | PWM0_SM0_MUX_TRIG1 01_1111 - PWM0_SM1_MUX_TRIG0 | PWM0_SM1_MUX_TRIG1 10_0000 - PWM0_SM2_MUX_TRIG0 | PWM0_SM2_MUX_TRIG1 10_0001 - PWM0_SM3_MUX_TRIG0 | PWM0_SM3_MUX_TRIG1 10_0010 - PWM1_SM0_MUX_TRIG0 | PWM1_SM0_MUX_TRIG1 10_0011 - PWM1_SM1_MUX_TRIG0 | PWM1_SM1_MUX_TRIG1 10_0100 - PWM1_SM2_MUX_TRIG0 | PWM1_SM2_MUX_TRIG1 10_0101 - PWM1_SM3_MUX_TRIG0 | PWM1_SM3_MUX_TRIG1 10_0110 - ENC0_CMP/POS_MATCH 10_0111 - ENC1_CMP/POS_MATCH 10_1000 - AOI0_OUT0 10_1001 - AOI0_OUT1 10_1010 - AOI0_OUT2 10_1011 - AOI0_OUT3 10_1100 - AOI1_OUT0 10_1101 - AOI1_OUT1 10_1110 - AOI1_OUT2 10_1111 - AOI1_OUT3 11_0000 - TMPR_OUT

Table continues on the next page...

Table continued from the previous page...

Field	Description
	11_0001-11_1111 - None

14.5.1.1.8 Pin interrupt select (PINTSEL0 - PINTSEL7)

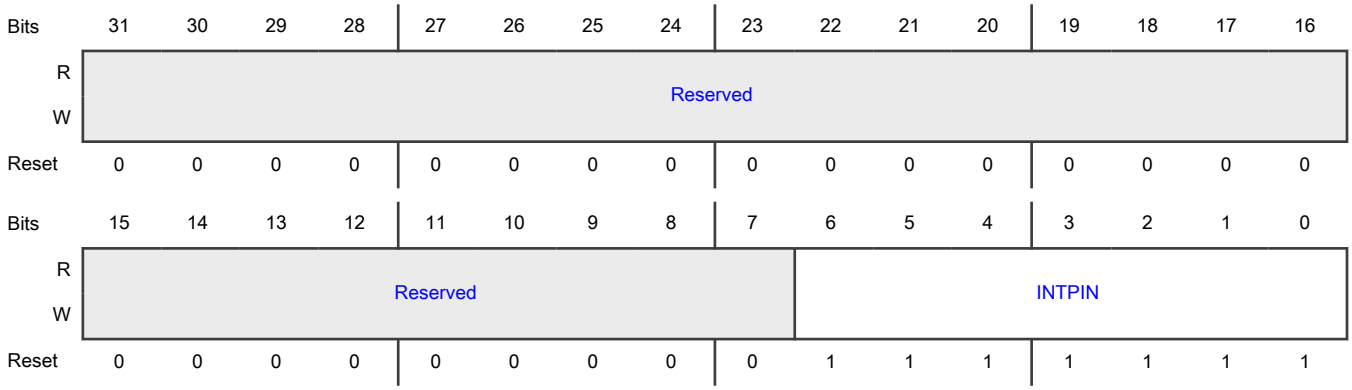
Each of these eight registers selects one pin from among ports 0 and 1 as the source of a pin interrupt or as the input to the pattern match engine. To select a pin for any of the 8 pin interrupts or pattern match engine inputs, write the GPIO port pin number as 0 to 31 for pins PIO0_0 to PIO0_31 to the INTPIN bits. Port 1 pins correspond to pin numbers 32 to 63. For example, setting INTPIN to 0x5 in PINTSEL0 selects pin PIO0_5 for pin interrupt 0. To determine the GPIO port pin number for a given device package, see the pin description table in the data sheet.

Offset

For a = 0 to 7:

Register	Offset
PINTSELa	C0h + (a × 4h)

Diagram



Fields

Field	Description
31-7 —	Reserved
6-0 INTPIN	Pin number select for pin interrupt or pattern match engine input. For PIOx_y: INTPIN = (x * 32) + y. PIO0_0 to PIO1_31 correspond to numbers 0 to 63.

14.5.1.1.9 Trigger select for DMA0 channel (DMA0_ITRIG_INMUX0 - DMA0_ITRIG_INMUX31)

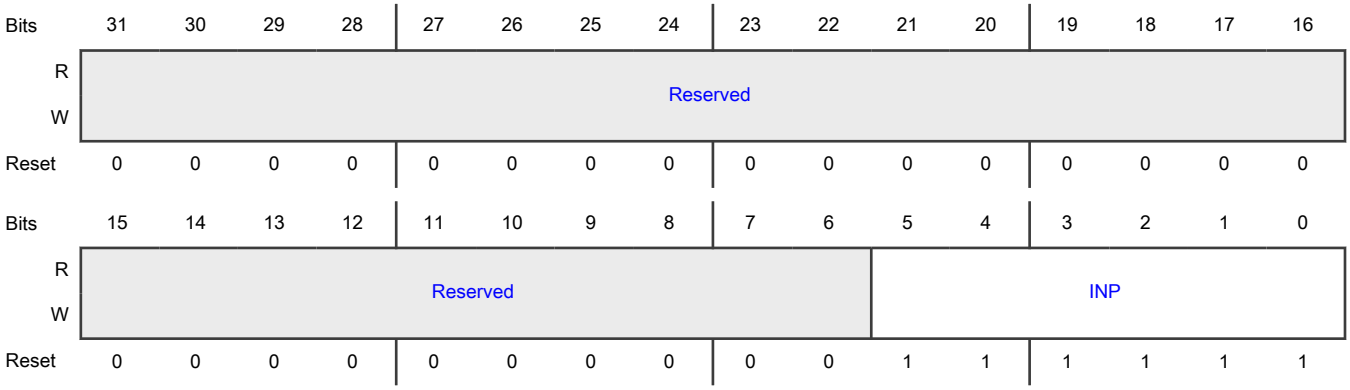
With the DMA trigger input multiplexing registers, one trigger input can be selected for each of the DMA channels from the potential internal sources. By default, none of the triggers are selected.

Offset

For a = 0 to 31:

Register	Offset
DMA0_ITRIG_INMUXa	E0h + (a × 4h)

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 INP	Trigger input number (binary value) for DMA channel n (n = 0 to 31). 00_0000 - FlexSPI_RX 00_0001 - FlexSPI_TX 00_0010 - GPIO_INT0 00_0011 - GPIO_INT1 00_0100 - GPIO_INT2 00_0101 - GPIO_INT3 00_0110 - T0_DMAREQ_M0 00_0111 - T0_DMAREQ_M1 00_1000 - T1_DMAREQ_M0 00_1001 - T1_DMAREQ_M1 00_1010 - T2_DMAREQ_M0 00_1011 - T2_DMAREQ_M1 00_1100 - T3_DMAREQ_M0 00_1101 - T3_DMAREQ_M1 00_1110 - T4_DMAREQ_M0

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00_1111 - T4_DMAREQ_M1 01_0000 - ACMP0_OUT 01_0001 - SDMA0_TRIGOUT_A 01_0010 - SDMA0_TRIGOUT_B 01_0011 - SDMA0_TRIGOUT_C 01_0100 - SDMA0_TRIGOUT_D 01_0101 - SCT_DMA0 01_0110 - SCT_DMA1 01_0111 - ADC0_tcomp[0] 01_1000 - ADC1_tcomp[0] 01_1001 - HSCMP0 01_1010 - HSCMP1 01_1011 - HSCMP2 01_1100 - AOI0_OUT0 01_1101 - AOI0_OUT1 01_1110 - AOI0_OUT2 01_1111 - AOI0_OUT3 10_0000 - AOI1_OUT0 10_0001 - AOI1_OUT1 10_0010 - AOI1_OUT2 10_0011 - AOI1_OUT3 10_0100 - FlexPWM0_req_capt0 10_0101 - FlexPWM0_req_capt1 10_0110 - FlexPWM0_req_capt2 10_0111 - FlexPWM0_req_capt3 10_1000 - FlexPWM0_req_val0 10_1001 - FlexPWM0_req_val1 10_1010 - FlexPWM0_req_val2 10_1011 - FlexPWM0_req_val3 10_1100 - FlexPWM1_req_capt0 10_1101 - FlexPWM1_req_capt1 10_1110 - FlexPWM1_req_capt2 10_1111 - FlexPWM1_req_capt3 11_0000 - FlexPWM1_req_val0

Table continues on the next page...

Table continued from the previous page...

Field	Description
	11_0001 - FlexPWM1_req_val1 11_0010 - FlexPWM1_req_val2 11_0011 - FlexPWM1_req_val3 11_0100 - TMPR_OUT 11_0110-11_1111 - Reserved

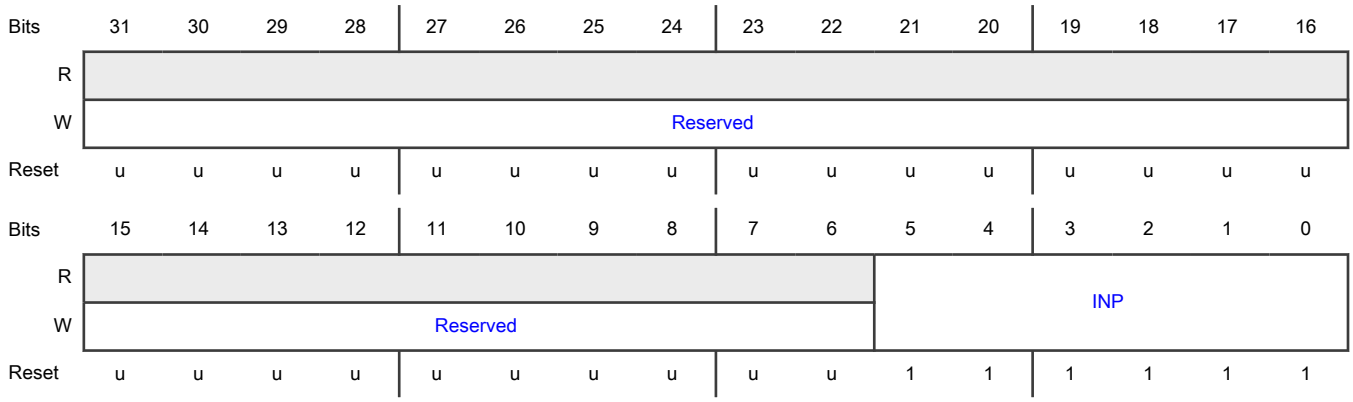
14.5.1.1.10 DMA0 output trigger selection for DMA0 input trigger (DMA0_OTRIG_INMUX0 - DMA0_OTRIG_INMUX6)

This register provides a multiplexer for inputs 15 to 18 of each DMA trigger input multiplexing register DMA0_ITRIG_INMUX. These inputs can be selected from among the trigger outputs generated by each DMA channel. By default, none of the triggers are selected.

Offset

Register	Offset
DMA0_OTRIG_INMUX0	160h
DMA0_OTRIG_INMUX1	164h
DMA0_OTRIG_INMUX2	168h
DMA0_OTRIG_INMUX3	16Ch
DMA0_OTRIG_INMUX4	170h
DMA0_OTRIG_INMUX5	174h
DMA0_OTRIG_INMUX6	178h

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 INP	DMA trigger output number (binary value) for DMA channel n (n = 0 to 52).

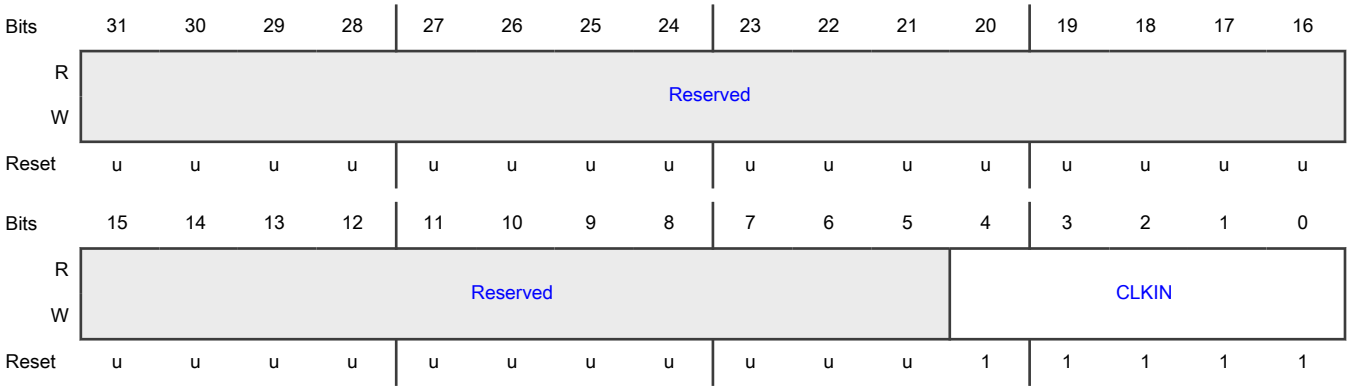
14.5.1.1.11 Selection for frequency measurement reference clock (FREQMEAS_REF)

This register selects a clock for the reference clock of the frequency measure function. By default, no clock is selected.

Offset

Register	Offset
FREQMEAS_REF	180h

Diagram



Fields

Field	Description
31-5 —	Reserved
4-0 CLKIN	Clock source number (binary value) for frequency measure function target clock. 0_0000 - XTAL32MHz 0_0001 - FRO_OSC_12M 0_0010 - FRO_OSC_96M 0_0011 - WDOSC (FRO1M) 0_0100 - 32KHZ_OSC

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0_0101 - MAIN_SYS_CLOCK 0_0110 - FREQME_GPIO_CLK_A 0_0111 - FREQME_GPIO_CLK_B 0_1000 - AOI0_OUT2 0_1001 - AOI1_OUT2

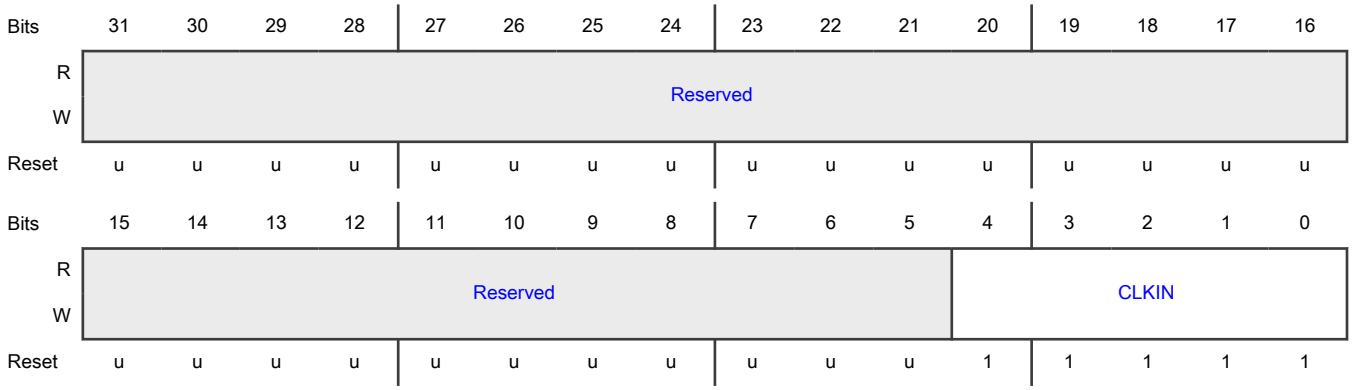
14.5.1.1.12 Selection for frequency measurement target clock (FREQMEAS_TAR)

This register selects a clock for the target clock of the frequency measure function. By default, no clock is selected.

Offset

Register	Offset
FREQMEAS_TAR	184h

Diagram



Fields

Field	Description
31-5 —	Reserved
4-0 CLKIN	Clock source number (binary value) for frequency measure function target clock 0_0000 - XTAL32MHz 0_0001 - FRO_OSC_12M 0_0010 - FRO_OSC_96M 0_0011 - WDOSC (FRO1M)

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0_0100 - 32KHZ_OSC 0_0101 - MAIN_SYS_CLOCK 0_0110 - FREQME_GPIO_CLK_A 0_0111 - FREQME_GPIO_CLK_B 0_1000 - AOI0_OUT2 0_1001 - AOI1_OUT2

14.5.1.1.13 Capture select register for TIMER3 inputs (TIMER3CAP0 - TIMER3CAP3)

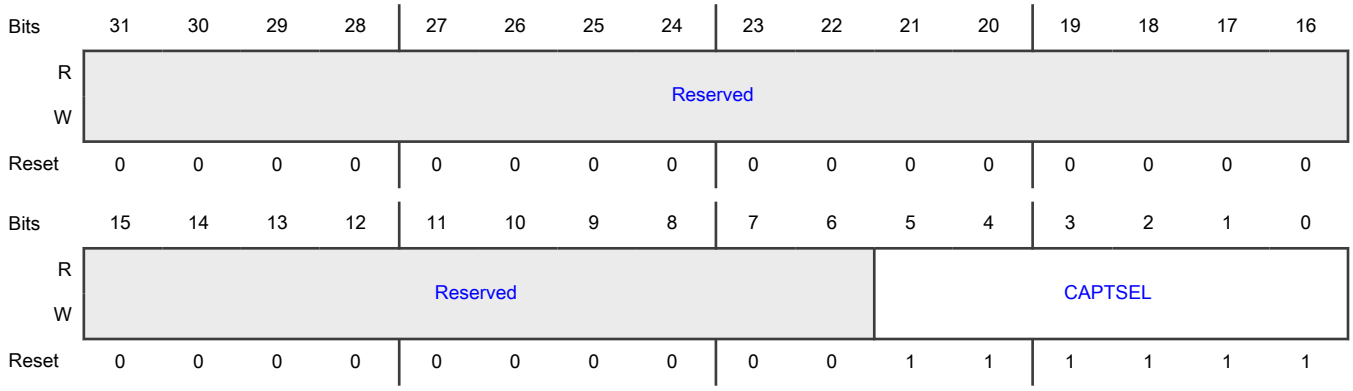
For each of the 5 standard timers, numbered i = 0 to 4 there are 4 TIMERiCAPTj, with j = 0 to 3, each allowing selecting between 25 external or internal input sources.

The output of TIMER0CAPT0 Input multiplexing register 0 selects the source for TIMER0 capture input 0. The output of TIMER0CAPT1 Input multiplexing register 1 selects the source for TIMER0 capture input 1, and so forth up to TIMER4CAPT3 Input multiplexing register 3, which selects the input for TIMER4 capture input 3.

Offset

Register	Offset
TIMER3CAP0	1A0h
TIMER3CAP1	1A4h
TIMER3CAP2	1A8h
TIMER3CAP3	1ACh

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 CAPTSEL	Input number to TIMER3 capture inputs 0 to 5 00_0000 - CTIMER_INP0 function selected from IOCON register 00_0001 - CTIMER_INP1 function selected from IOCON register 00_0010 - CTIMER_INP2 function selected from IOCON register 00_0011 - CTIMER_INP3 function selected from IOCON register 00_0100 - CTIMER_INP4 function selected from IOCON register 00_0101 - CTIMER_INP5 function selected from IOCON register 00_0110 - CTIMER_INP6 function selected from IOCON register 00_0111 - CTIMER_INP7 function selected from IOCON register 00_1000 - CTIMER_INP8 function selected from IOCON register 00_1001 - CTIMER_INP9 function selected from IOCON register 00_1010 - CTIMER_INP10 function selected from IOCON register 00_1011 - CTIMER_INP11 function selected from IOCON register 00_1100 - CTIMER_INP12 function selected from IOCON register 00_1101 - CTIMER_INP13 function selected from IOCON register 00_1110 - CTIMER_INP14 function selected from IOCON register 00_1111 - CTIMER_INP15 function selected from IOCON register 01_0000 - CTIMER_INP16 function selected from IOCON register 01_0001 - CTIMER_INP17 function selected from IOCON register 01_0010 - CTIMER_INP18 function selected from IOCON register 01_0011 - CTIMER_INP19 function selected from IOCON register 01_0100 - USB0_FRAME_TOGGLE 01_0101 - Reserved 01_0110 - ACMP0_OUT from analog comparator 01_0111 - SHARED_I2S_WS0 output from I2S pin sharing 01_1000 - SHARED_I2S_WS1 output from I2S pin sharing 01_1001 - ADC0_IRQ 01_1010 - ADC1_IRQ 01_1011 - HSCMP0_OUT 01_1100 - HSCMP1_OUT 01_1101 - HSCMP2_OUT

Table continues on the next page...

Table continued from the previous page...

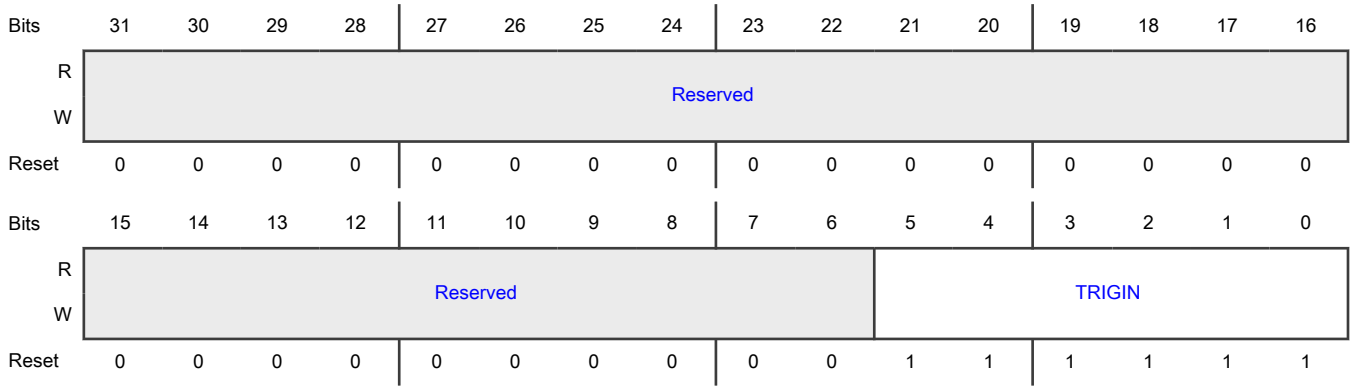
Field	Description
	01_1110 - PWM0_SM0_MUX_TRIG0 | PWM0_SM0_MUX_TRIG1 01_1111 - PWM0_SM1_MUX_TRIG0 | PWM0_SM1_MUX_TRIG1 10_0000 - PWM0_SM2_MUX_TRIG0 | PWM0_SM2_MUX_TRIG1 10_0001 - PWM0_SM3_MUX_TRIG0 | PWM0_SM3_MUX_TRIG1 10_0010 - PWM1_SM0_MUX_TRIG0 | PWM1_SM0_MUX_TRIG1 10_0011 - PWM1_SM1_MUX_TRIG0 | PWM1_SM1_MUX_TRIG1 10_0100 - PWM1_SM2_MUX_TRIG0 | PWM1_SM2_MUX_TRIG1 10_0101 - PWM1_SM3_MUX_TRIG0 | PWM1_SM3_MUX_TRIG1 10_0110 - ENC0_CMP/POS_MATCH 10_0111 - ENC1_CMP/POS_MATCH 10_1000 - AOI0_OUT0 10_1001 - AOI0_OUT1 10_1010 - AOI0_OUT2 10_1011 - AOI0_OUT3 10_1100 - AOI1_OUT0 10_1101 - AOI1_OUT1 10_1110 - AOI1_OUT2 10_1111 - AOI1_OUT3 11_0000 - TMPR_OUT 11_0001-11_1111 - None

14.5.1.1.14 Trigger register for TIMER3 (TIMER3TRIG)

Offset

Register	Offset
TIMER3TRIG	1B0h

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 TRIGIN	Input number to TIMER3 trigger inputs 00_0000 - CTIMER_INP0 function selected from IOCON register 00_0001 - CTIMER_INP1 function selected from IOCON register 00_0010 - CTIMER_INP2 function selected from IOCON register 00_0011 - CTIMER_INP3 function selected from IOCON register 00_0100 - CTIMER_INP4 function selected from IOCON register 00_0101 - CTIMER_INP5 function selected from IOCON register 00_0110 - CTIMER_INP6 function selected from IOCON register 00_0111 - CTIMER_INP7 function selected from IOCON register 00_1000 - CTIMER_INP8 function selected from IOCON register 00_1001 - CTIMER_INP9 function selected from IOCON register 00_1010 - CTIMER_INP10 function selected from IOCON register 00_1011 - CTIMER_INP11 function selected from IOCON register 00_1100 - CTIMER_INP12 function selected from IOCON register 00_1101 - CTIMER_INP13 function selected from IOCON register 00_1110 - CTIMER_INP14 function selected from IOCON register 00_1111 - CTIMER_INP15 function selected from IOCON register 01_0000 - CTIMER_INP16 function selected from IOCON register 01_0001 - CTIMER_INP17 function selected from IOCON register 01_0010 - CTIMER_INP18 function selected from IOCON register 01_0011 - CTIMER_INP19 function selected from IOCON register

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01_0100 - USB0_FRAME_TOGGLE 01_0101 - Reserved 01_0110 - ACMP0_OUT from analog comparator 01_0111 - SHARED_I2S_WS0 output from I2S pin sharing 01_1000 - SHARED_I2S_WS1 output from I2S pin sharing 01_1001 - ADC0_IRQ 01_1010 - ADC1_IRQ 01_1011 - HSCMP0_OUT 01_1100 - HSCMP1_OUT 01_1101 - HSCMP2_OUT 01_1110 - PWM0_SM0_MUX_TRIG0 | PWM0_SM0_MUX_TRIG1 01_1111 - PWM0_SM1_MUX_TRIG0 | PWM0_SM1_MUX_TRIG1 10_0000 - PWM0_SM2_MUX_TRIG0 | PWM0_SM2_MUX_TRIG1 10_0001 - PWM0_SM3_MUX_TRIG0 | PWM0_SM3_MUX_TRIG1 10_0010 - PWM1_SM0_MUX_TRIG0 | PWM1_SM0_MUX_TRIG1 10_0011 - PWM1_SM1_MUX_TRIG0 | PWM1_SM1_MUX_TRIG1 10_0100 - PWM1_SM2_MUX_TRIG0 | PWM1_SM2_MUX_TRIG1 10_0101 - PWM1_SM3_MUX_TRIG0 | PWM1_SM3_MUX_TRIG1 10_0110 - ENC0_CMP/POS_MATCH 10_0111 - ENC1_CMP/POS_MATCH 10_1000 - AOI0_OUT0 10_1001 - AOI0_OUT1 10_1010 - AOI0_OUT2 10_1011 - AOI0_OUT3 10_1100 - AOI1_OUT0 10_1101 - AOI1_OUT1 10_1110 - AOI1_OUT2 10_1111 - AOI1_OUT3 11_0000 - TMPR_OUT 11_0001-11_1111 - None

14.5.1.1.15 Capture select register for TIMER4 inputs (TIMER4CAP0 - TIMER4CAP3)

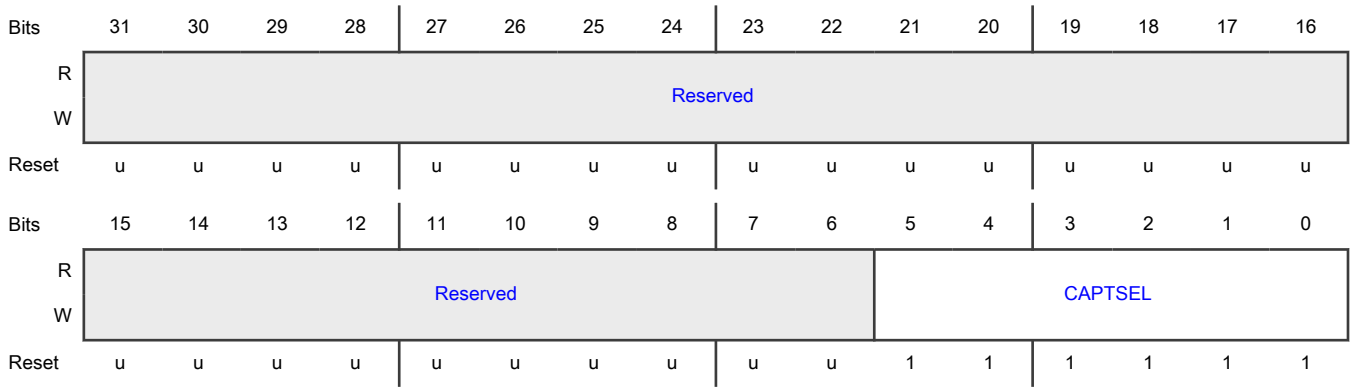
For each of the 5 standard timers, numbered i = 0 to 4 there are 4 TIMERiCAPTj, with j = 0 to 3, each allowing selecting between 25 external or internal input sources.

The output of TIMER0CAPT0 Input multiplexing register 0 selects the source for TIMER0 capture input 0. The output of TIMER0CAPT1 Input multiplexing register 1 selects the source for TIMER0 capture input 1, and so forth up to TIMER4CAPT3 Input multiplexing register 3, which selects the input for TIMER4 capture input 3.

Offset

Register	Offset
TIMER4CAP0	1C0h
TIMER4CAP1	1C4h
TIMER4CAP2	1C8h
TIMER4CAP3	1CCh

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 CAPTSEL	Input number to TIMER4 capture inputs 0 to 5 00_0000 - CTIMER_INP0 function selected from IOCON register 00_0001 - CTIMER_INP1 function selected from IOCON register 00_0010 - CTIMER_INP2 function selected from IOCON register 00_0011 - CTIMER_INP3 function selected from IOCON register 00_0100 - CTIMER_INP4 function selected from IOCON register 00_0101 - CTIMER_INP5 function selected from IOCON register 00_0110 - CTIMER_INP6 function selected from IOCON register 00_0111 - CTIMER_INP7 function selected from IOCON register 00_1000 - CTIMER_INP8 function selected from IOCON register 00_1001 - CTIMER_INP9 function selected from IOCON register

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00_1010 - CTIMER_INP10 function selected from IOCON register 00_1011 - CTIMER_INP11 function selected from IOCON register 00_1100 - CTIMER_INP12 function selected from IOCON register 00_1101 - CTIMER_INP13 function selected from IOCON register 00_1110 - CTIMER_INP14 function selected from IOCON register 00_1111 - CTIMER_INP15 function selected from IOCON register 01_0000 - CTIMER_INP16 function selected from IOCON register 01_0001 - CTIMER_INP17 function selected from IOCON register 01_0010 - CTIMER_INP18 function selected from IOCON register 01_0011 - CTIMER_INP19 function selected from IOCON register 01_0100 - USB0_FRAME_TOGGLE 01_0101 - Reserved 01_0110 - ACMP0_OUT from analog comparator 01_0111 - SHARED_I2S_WS0 output from I2S pin sharing 01_1000 - SHARED_I2S_WS1 output from I2S pin sharing 01_1001 - ADC0_IRQ 01_1010 - ADC1_IRQ 01_1011 - HSCMP0_OUT 01_1100 - HSCMP1_OUT 01_1101 - HSCMP2_OUT 01_1110 - PWM0_SM0_MUX_TRIG0 | PWM0_SM0_MUX_TRIG1 01_1111 - PWM0_SM1_MUX_TRIG0 | PWM0_SM1_MUX_TRIG1 10_0000 - PWM0_SM2_MUX_TRIG0 | PWM0_SM2_MUX_TRIG1 10_0001 - PWM0_SM3_MUX_TRIG0 | PWM0_SM3_MUX_TRIG1 10_0010 - PWM1_SM0_MUX_TRIG0 | PWM1_SM0_MUX_TRIG1 10_0011 - PWM1_SM1_MUX_TRIG0 | PWM1_SM1_MUX_TRIG1 10_0100 - PWM1_SM2_MUX_TRIG0 | PWM1_SM2_MUX_TRIG1 10_0101 - PWM1_SM3_MUX_TRIG0 | PWM1_SM3_MUX_TRIG1 10_0110 - ENC0_CMP/POS_MATCH 10_0111 - ENC1_CMP/POS_MATCH 10_1000 - AOI0_OUT0 10_1001 - AOI0_OUT1 10_1010 - AOI0_OUT2 10_1011 - AOI0_OUT3

Table continues on the next page...

Table continued from the previous page...

Field	Description
	10_1100 - AOI1_OUT0 10_1101 - AOI1_OUT1 10_1110 - AOI1_OUT2 10_1111 - AOI1_OUT3 11_0000 - TMPR_OUT 11_0001-11_1111 - None

14.5.1.1.16 Trigger register for TIMER4 (TIMER4TRIG)

Offset

Register	Offset
TIMER4TRIG	1D0h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	Reserved										TRIGIN					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Description
31-6 —	Reserved
5-0 TRIGIN	Input number to TIMER4 trigger inputs 00_0000 - CTIMER_INP0 function selected from IOCON register 00_0001 - CTIMER_INP1 function selected from IOCON register 00_0010 - CTIMER_INP2 function selected from IOCON register 00_0011 - CTIMER_INP3 function selected from IOCON register

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00_0100 - CTIMER_INP4 function selected from IOCON register 00_0101 - CTIMER_INP5 function selected from IOCON register 00_0110 - CTIMER_INP6 function selected from IOCON register 00_0111 - CTIMER_INP7 function selected from IOCON register 00_1000 - CTIMER_INP8 function selected from IOCON register 00_1001 - CTIMER_INP9 function selected from IOCON register 00_1010 - CTIMER_INP10 function selected from IOCON register 00_1011 - CTIMER_INP11 function selected from IOCON register 00_1100 - CTIMER_INP12 function selected from IOCON register 00_1101 - CTIMER_INP13 function selected from IOCON register 00_1110 - CTIMER_INP14 function selected from IOCON register 00_1111 - CTIMER_INP15 function selected from IOCON register 01_0000 - CTIMER_INP16 function selected from IOCON register 01_0001 - CTIMER_INP17 function selected from IOCON register 01_0010 - CTIMER_INP18 function selected from IOCON register 01_0011 - CTIMER_INP19 function selected from IOCON register 01_0100 - USB0_FRAME_TOGGLE 01_0101 - Reserved 01_0110 - ACMP0_OUT from analog comparator 01_0111 - SHARED_I2S_WS0 output from I2S pin sharing 01_1000 - SHARED_I2S_WS1 output from I2S pin sharing 01_1001 - ADC0_IRQ 01_1010 - ADC1_IRQ 01_1011 - HSCMP0_OUT 01_1100 - HSCMP1_OUT 01_1101 - HSCMP2_OUT 01_1110 - PWM0_SM0_MUX_TRIG0 | PWM0_SM0_MUX_TRIG1 01_1111 - PWM0_SM1_MUX_TRIG0 | PWM0_SM1_MUX_TRIG1 10_0000 - PWM0_SM2_MUX_TRIG0 | PWM0_SM2_MUX_TRIG1 10_0001 - PWM0_SM3_MUX_TRIG0 | PWM0_SM3_MUX_TRIG1 10_0010 - PWM1_SM0_MUX_TRIG0 | PWM1_SM0_MUX_TRIG1 10_0011 - PWM1_SM1_MUX_TRIG0 | PWM1_SM1_MUX_TRIG1 10_0100 - PWM1_SM2_MUX_TRIG0 | PWM1_SM2_MUX_TRIG1 10_0101 - PWM1_SM3_MUX_TRIG0 | PWM1_SM3_MUX_TRIG1

Table continues on the next page...

Table continued from the previous page...

Field	Description
	10_0110 - ENC0_CMP/POS_MATCH 10_0111 - ENC1_CMP/POS_MATCH 10_1000 - AOI0_OUT0 10_1001 - AOI0_OUT1 10_1010 - AOI0_OUT2 10_1011 - AOI0_OUT3 10_1100 - AOI1_OUT0 10_1101 - AOI1_OUT1 10_1110 - AOI1_OUT2 10_1111 - AOI1_OUT3 11_0000 - TMPR_OUT 11_0001-11_1111 - None

14.5.1.1.17 Trigger select for DMA1 channel (DMA1_ITRIG_INMUX0 - DMA1_ITRIG_INMUX15)

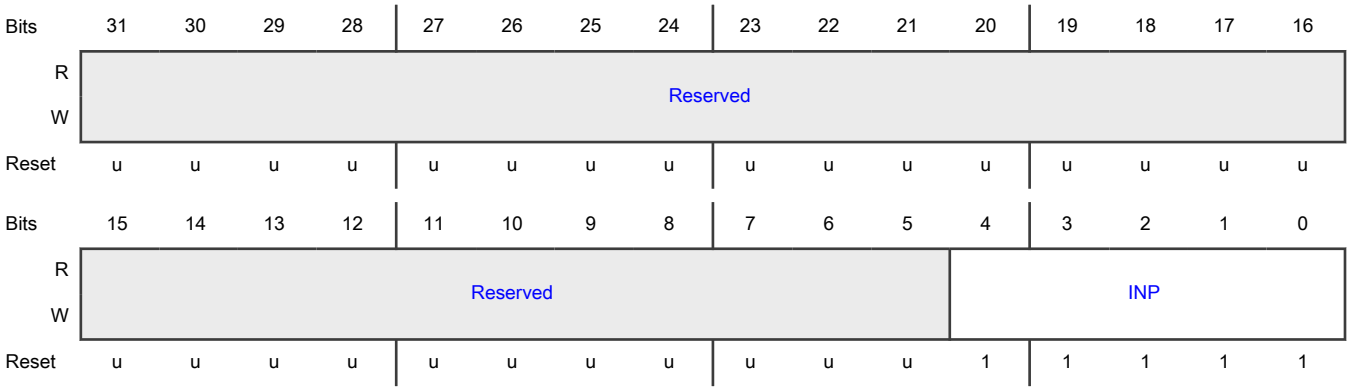
With the DMA trigger input multiplexing registers, one trigger input can be selected for each of the DMA channels from the potential internal sources. By default, none of the triggers are selected.

Offset

For n = 0 to 15:

Register	Offset
DMA1_ITRIG_INMUXn	200h + (n × 4h)

Diagram



Fields

Field	Description
31-5 —	Reserved
4-0 INP	Trigger input number (binary value) for DMA channel n (n = 0 to 14). 0_0000 - Pin interrupt 0 (GPIO_INT0) 0_0001 - Pin interrupt 1 (GPIO_INT1) 0_0010 - Pin interrupt 2 (GPIO_INT2) 0_0011 - Pin interrupt 3 (GPIO_INT3) 0_0100 - Timer CTIMER0 Match 0 (T0_DMAREQ_M0) 0_0101 - Timer CTIMER0 Match 1 (T0_DMAREQ_M1) 0_0110 - Timer CTIMER2 Match 0 (T2_DMAREQ_M0) 0_0111 - Timer CTIMER4 Match 0 (T4_DMAREQ_M0) 0_1000 - SDMA1_TRIGOUT_A 0_1001 - SDMA1_TRIGOUT_B 0_1010 - SDMA1_TRIGOUT_C 0_1011 - SDMA1_TRIGOUT_D 0_1100 - SCT_DMA_REQ0 0_1101 - SCT_DMA_REQ1 0_1110 - FlexSPI_RX 0_1111 - FlexSPI_TX 1_0000 - AOI0_OUT0 1_0001 - AOI0_OUT1 1_0010 - AOI0_OUT2 1_0011 - AOI0_OUT3 1_0100 - AOI1_OUT0 1_0101 - AOI1_OUT1 1_0110 - AOI1_OUT2 1_0111 - AOI1_OUT3 1_1000 - TMPR_OUT 1_1001-1_1111 - Reserved

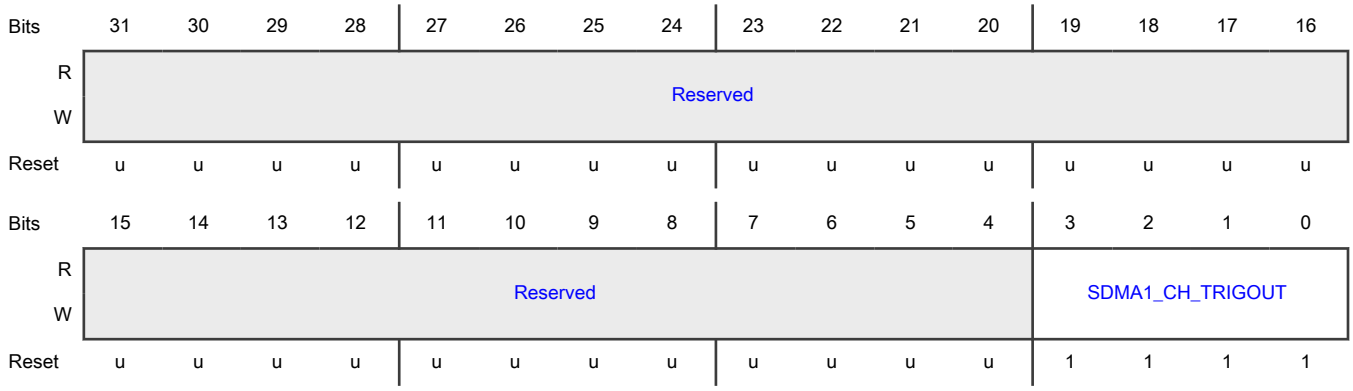
14.5.1.1.18 DMA1 output trigger selection for DMA1 input trigger (DMA1_OTRIG_INMUX0 - DMA1_OTRIG_INMUX3)

This register provides a multiplexer for inputs 8 to 11 of each DMA trigger input multiplexing register DMA1_ITRIG_INMUX. These inputs can be selected from among the trigger outputs generated by each DMA channel. By default, none of the triggers are selected.

Offset

Register	Offset
DMA1_OTRIG_INMUX0	240h
DMA1_OTRIG_INMUX1	244h
DMA1_OTRIG_INMUX2	248h
DMA1_OTRIG_INMUX3	24Ch

Diagram



Fields

Field	Description
31-4 —	Reserved
3-0 SDMA1_CH_TRIGOUT	DMA trigger output number (binary value) for DMA channel n (n = 0 to 15). 0000 - SDMA1_CH0_TRIGOUT 0001 - SDMA1_CH1_TRIGOUT 0010 - SDMA1_CH2_TRIGOUT 0011 - SDMA1_CH3_TRIGOUT 0100 - SDMA1_CH4_TRIGOUT 0101 - SDMA1_CH5_TRIGOUT 0110 - SDMA1_CH6_TRIGOUT 0111 - SDMA1_CH7_TRIGOUT 1000 - SDMA1_CH8_TRIGOUT 1001 - SDMA1_CH9_TRIGOUT 1010 - SDMA1_CH10_TRIGOUT 1011 - SDMA1_CH11_TRIGOUT

Table continues on the next page...

Table continued from the previous page...

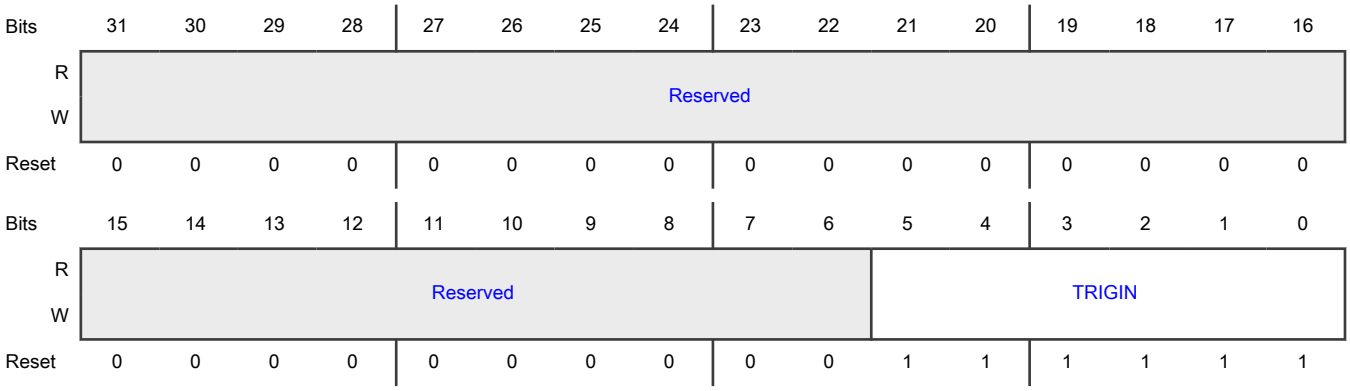
Field	Description
	1100 - SDMA1_CH12_TRIGOUT 1101 - SDMA1_CH13_TRIGOUT 1110-1111 - Reserved

14.5.1.1.19 Input connections for HSCMP0 (HSCMP0_TRIG)

Offset

Register	Offset
HSCMP0_TRIG	260h

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 TRIGIN	CMP0 input trigger 00_0000 - PIN_INT0 00_0001 - PIN_INT6 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT6 00_0101 - T0_MAT3 00_0110 - T1_MAT3

Table continues on the next page...

Table continued from the previous page...

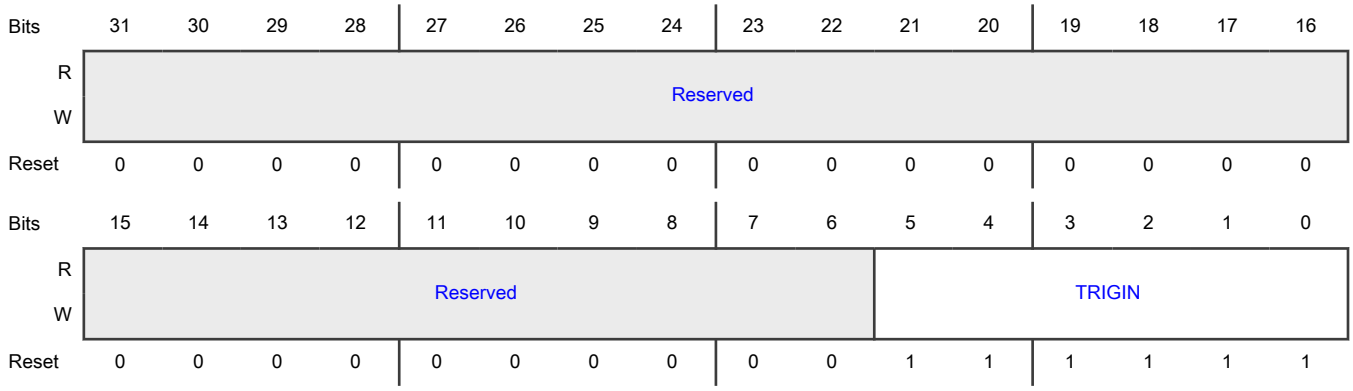
Field	Description
	00_0111 - T2_MAT3 00_1000 - T0_MAT0 00_1001 - T4_MAT0 00_1010 - Reserved 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC1_tcomp[0] 00_1111 - Reserved 01_0000 - Reserved 01_0001 - PWM0_SM0_MUX_TRIG0 | PWM0_SM0_MUX_TRIG1 01_0010 - PWM0_SM1_MUX_TRIG0 | PWM0_SM1_MUX_TRIG1 01_0011 - PWM0_SM2_MUX_TRIG0 | PWM0_SM2_MUX_TRIG1 01_0100 - PWM0_SM3_MUX_TRIG0 | PWM0_SM3_MUX_TRIG1 01_0101 - PWM1_SM0_MUX_TRIG0 | PWM1_SM0_MUX_TRIG1 01_0110 - PWM1_SM1_MUX_TRIG0 | PWM1_SM1_MUX_TRIG1 01_0111 - PWM1_SM2_MUX_TRIG0 | PWM1_SM2_MUX_TRIG1 01_1000 - PWM1_SM3_MUX_TRIG0 | PWM1_SM3_MUX_TRIG1 01_1001 - ENC0_CMP/POS_MATCH 01_1010 - ENC1_CMP/POS_MATCH 01_1011 - AOI0_OUT0 01_1100 - AOI0_OUT1 01_1101 - AOI0_OUT2 01_1110 - AOI0_OUT3 01_1111 - AOI1_OUT0 10_0000 - AOI1_OUT1 10_0001 - AOI1_OUT2 10_0010 - AOI1_OUT3 10_0011 - DMA0_TRIGOUT0 10_0100 - DMA0_TRIGOUT1 10_0101 - DMA0_TRIGOUT2

14.5.1.1.20 ADC0 Trigger input connections (ADC0_TRIG0 - ADC0_TRIG3)

Offset

Register	Offset
ADC0_TRIG0	280h
ADC0_TRIG1	284h
ADC0_TRIG2	288h
ADC0_TRIG3	28Ch

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 TRIGIN	ADC0 trigger inputs 00_0000 - PIN_INT0 00_0001 - PIN_INT1 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT9 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T3_MAT3 00_1001 - T4_MAT3 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2] 01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM0_SM0_MUX_TRIG0 01_1001 - PWM0_SM0_MUX_TRIG1 01_1010 - PWM0_SM1_MUX_TRIG0 01_1011 - PWM0_SM1_MUX_TRIG1 01_1100 - PWM0_SM2_MUX_TRIG0 01_1101 - PWM0_SM2_MUX_TRIG1 01_1110 - PWM0_SM3_MUX_TRIG0 01_1111 - PWM0_SM3_MUX_TRIG1 10_0000 - PWM1_SM0_MUX_TRIG0 10_0001 - PWM1_SM0_MUX_TRIG1 10_0010 - PWM1_SM1_MUX_TRIG0 10_0011 - PWM1_SM1_MUX_TRIG1 10_0100 - PWM1_SM2_MUX_TRIG0 10_0101 - PWM1_SM2_MUX_TRIG1 10_0110 - PWM1_SM3_MUX_TRIG0 10_0111 - PWM1_SM3_MUX_TRIG1 10_1000 - ENC0_CMP/POS_MATCH 10_1001 - ENC1_CMP/POS_MATCH 10_1010 - AOI0_OUT0 10_1011 - AOI0_OUT1 10_1100 - AOI0_OUT2 10_1101 - AOI0_OUT3

Table continues on the next page...

Table continued from the previous page...

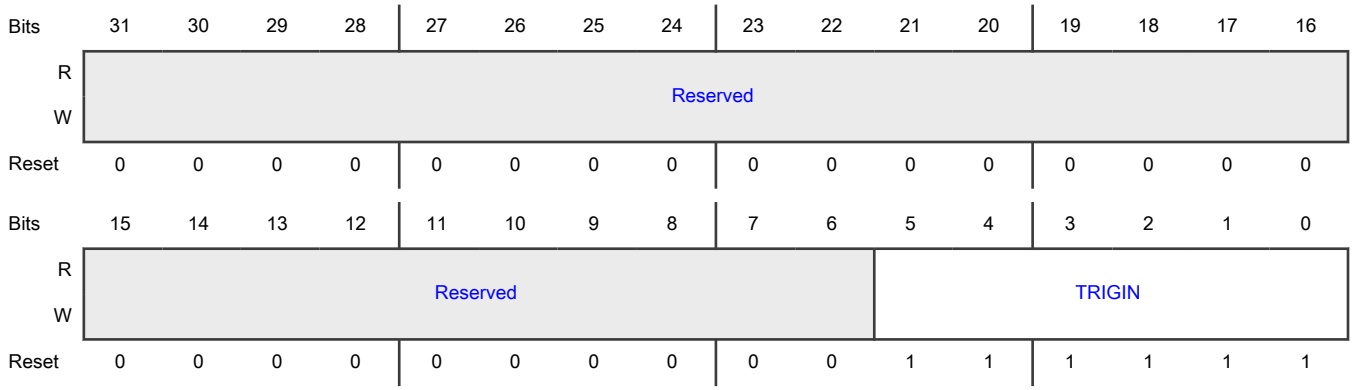
Field	Description
	10_1110 - AOI1_OUT0 10_1111 - AOI1_OUT1 11_0000 - AOI1_OUT2 11_0001 - AOI1_OUT3 11_0010 - DMA0_TRIGOUT0 11_0011 - DMA0_TRIGOUT1 11_0100 - DMA0_TRIGOUT2 11_0101-11_1111 - None

14.5.1.1.21 ADC1 Trigger input connections (ADC1_TRIG0 - ADC1_TRIG3)

Offset

Register	Offset
ADC1_TRIG0	2C0h
ADC1_TRIG1	2C4h
ADC1_TRIG2	2C8h
ADC1_TRIG3	2CCh

Diagram



Fields

Field	Description
31-6 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
5-0 TRIGIN	ADC1 trigger inputs 00_0000 - PIN_INT0 00_0001 - PIN_INT2 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT3 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T3_MAT2 00_1001 - T4_MAT1 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2] 01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM0_SM0_MUX_TRIG0 01_1001 - PWM0_SM0_MUX_TRIG1 01_1010 - PWM0_SM1_MUX_TRIG0 01_1011 - PWM0_SM1_MUX_TRIG1 01_1100 - PWM0_SM2_MUX_TRIG0 01_1101 - PWM0_SM2_MUX_TRIG1 01_1110 - PWM0_SM3_MUX_TRIG0 01_1111 - PWM0_SM3_MUX_TRIG1 10_0000 - PWM1_SM0_MUX_TRIG0

Table continues on the next page...

Table continued from the previous page...

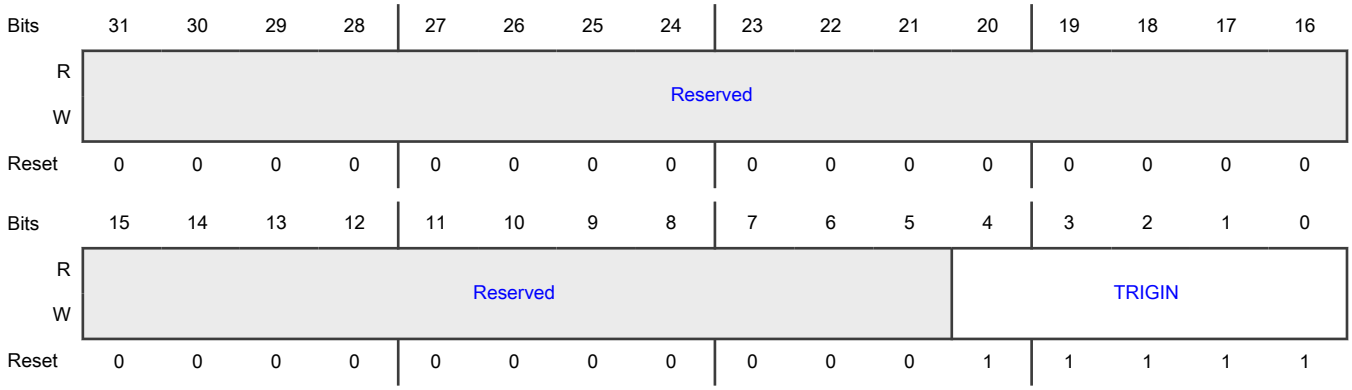
Field	Description
	10_0001 - PWM1_SM0_MUX_TRIG1 10_0010 - PWM1_SM1_MUX_TRIG0 10_0011 - PWM1_SM1_MUX_TRIG1 10_0100 - PWM1_SM2_MUX_TRIG0 10_0101 - PWM1_SM2_MUX_TRIG1 10_0110 - PWM1_SM3_MUX_TRIG0 10_0111 - PWM1_SM3_MUX_TRIG1 10_1000 - ENC0_CMP/POS_MATCH 10_1001 - ENC1_CMP/POS_MATCH 10_1010 - AOI0_OUT0 10_1011 - AOI0_OUT1 10_1100 - AOI0_OUT2 10_1101 - AOI0_OUT3 10_1110 - AOI1_OUT0 10_1111 - AOI1_OUT1 11_0000 - AOI1_OUT2 11_0001 - AOI1_OUT3 11_0010 - DMA0_TRIGOUT0 11_0011 - DMA0_TRIGOUT1 11_0100 - DMA0_TRIGOUT2 11_0101-11_1111 - None

14.5.1.1.22 DAC0 Trigger Inputs (DAC0_TRIG)

Offset

Register	Offset
DAC0_TRIG	300h

Diagram



Fields

Field	Description
31-5 —	Reserved
4-0 TRIGIN	DAC0 trigger input 0_0000 - PIN_INT0 0_0001 - PIN_INT3 0_0010 - SCT_OUT4 0_0011 - SCT_OUT5 0_0100 - SCT_OUT0 0_0101 - T0_MAT3 0_0110 - T1_MAT3 0_0111 - T2_MAT3 0_1000 - T2_MAT0 0_1001 - T3_MAT0 0_1010 - ACMP0_OUT 0_1011 - ARM_TXEV 0_1100 - GPIOINT_BMATCH 0_1101 - ADC0_tcomp[0] 0_1110 - ADC1_tcomp[0] 0_1111 - HSCMP0_OUT 1_0000 - HSCMP1_OUT 1_0001 - HSCMP2_OUT 1_0010 - AOI0_OUT0 1_0011 - AOI0_OUT1

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1_0100 - AOI0_OUT2 1_0101 - AOI0_OUT3 1_0110 - AOI1_OUT0 1_0111 - AOI1_OUT1 1_1000 - AOI1_OUT2 1_1001 - AOI1_OUT3 1_1010 - DMA0_TRIGOUT0 1_1011 - DMA0_TRIGOUT1 1_1100 - DMA0_TRIGOUT2

14.5.1.1.23 DAC1 Trigger Inputs (DAC1_TRIG)

Offset

Register	Offset
DAC1_TRIG	320h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	Reserved											TRIGIN				
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Fields

Field	Description
31-5 —	Reserved
4-0	DAC1 trigger input 0_0000 - PIN_INT0

Table continues on the next page...

Table continued from the previous page...

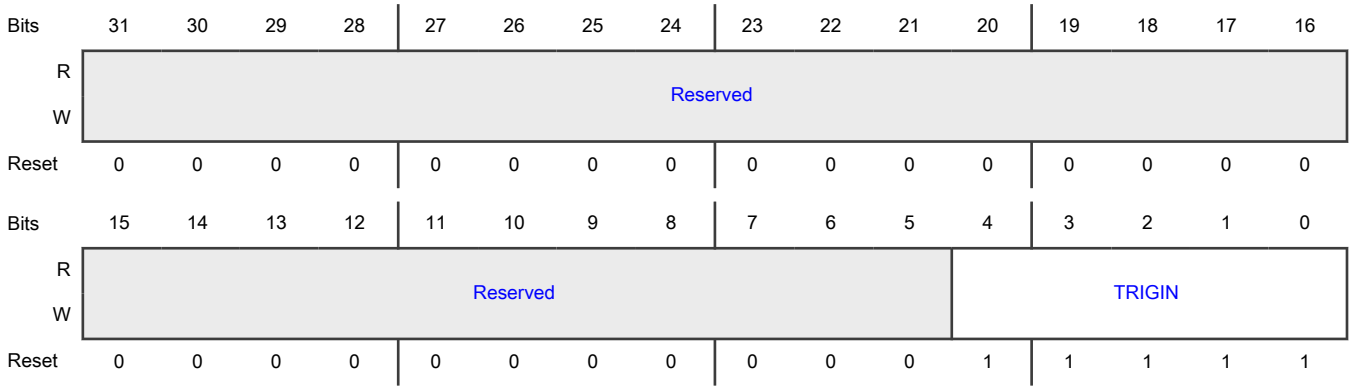
Field	Description
TRIGIN	0_0001 - PIN_INT4 0_0010 - SCT_OUT4 0_0011 - SCT_OUT5 0_0100 - SCT_OUT1 0_0101 - T0_MAT3 0_0110 - T1_MAT3 0_0111 - T2_MAT3 0_1000 - T2_MAT1 0_1001 - T3_MAT1 0_1010 - ACMP0_OUT 0_1011 - ARM_TXEV 0_1100 - GPIOINT_BMATCH 0_1101 - ADC0_tcomp[1] 0_1110 - ADC1_tcomp[1] 0_1111 - HSCMP0_OUT 1_0000 - HSCMP1_OUT 1_0001 - HSCMP2_OUT 1_0010 - AOI0_OUT0 1_0011 - AOI0_OUT1 1_0100 - AOI0_OUT2 1_0101 - AOI0_OUT3 1_0110 - AOI1_OUT0 1_0111 - AOI1_OUT1 1_1000 - AOI1_OUT2 1_1001 - AOI1_OUT3 1_1010 - DMA0_TRIGOUT0 1_1011 - DMA0_TRIGOUT1 1_1100 - DMA0_TRIGOUT2

14.5.1.1.24 DAC2 Trigger Inputs (DAC2_TRIG)

Offset

Register	Offset
DAC2_TRIG	340h

Diagram



Fields

Field	Description
31-5 —	Reserved
4-0 TRIGIN	DAC2 trigger input 0_0000 - PIN_INT0 0_0001 - PIN_INT5 0_0010 - SCT_OUT4 0_0011 - SCT_OUT5 0_0100 - SCT_OUT2 0_0101 - T0_MAT3 0_0110 - T1_MAT3 0_0111 - T2_MAT3 0_1000 - T2_MAT2 0_1001 - T3_MAT2 0_1010 - ACMP0_OUT 0_1011 - ARM_TXEV 0_1100 - GPIOINT_BMATCH 0_1101 - ADC0_tcomp[2] 0_1110 - ADC1_tcomp[2] 0_1111 - HSCMP0_OUT 1_0000 - HSCMP1_OUT 1_0001 - HSCMP2_OUT 1_0010 - AOI0_OUT0 1_0011 - AOI0_OUT1

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1_0100 - AOI0_OUT2 1_0101 - AOI0_OUT3 1_0110 - AOI1_OUT0 1_0111 - AOI1_OUT1 1_1000 - AOI1_OUT2 1_1001 - AOI1_OUT3 1_1010 - DMA0_TRIGOUT0 1_1011 - DMA0_TRIGOUT1 1_1100 - DMA0_TRIGOUT2

14.5.1.1.25 ENC0 Trigger Input Connections (ENC0_TRIG)

Offset

Register	Offset
ENC0_TRIG	360h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	Reserved										TRIGIN					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Description
31-6 —	Reserved
5-0	ENC0 input trigger 00_0000 - PIN_INT0

Table continues on the next page...

Table continued from the previous page...

Field	Description
TRIGIN	00_0001 - PIN_INT4 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT1 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T1_MAT0 00_1001 - T3_MAT0 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2] 01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM1_SM0_MUX_TRIG0 01_1001 - PWM1_SM0_MUX_TRIG1 01_1010 - PWM1_SM1_MUX_TRIG0 01_1011 - PWM1_SM1_MUX_TRIG1 01_1100 - PWM1_SM2_MUX_TRIG0 01_1101 - PWM1_SM2_MUX_TRIG1 01_1110 - PWM1_SM3_MUX_TRIG0 01_1111 - PWM1_SM3_MUX_TRIG1 10_0000 - ENC0_CMP/POS_MATCH 10_0001 - ENC1_CMP/POS_MATCH 10_0010 - AOI0_OUT0

Table continues on the next page...

Table continued from the previous page...

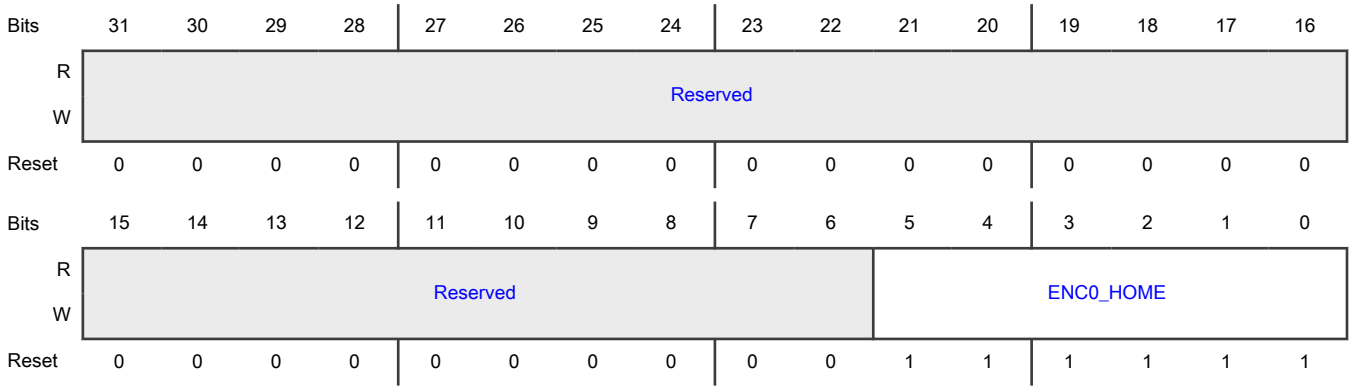
Field	Description
	10_0011 - AOI0_OUT1 10_0100 - AOI0_OUT2 10_0101 - AOI0_OUT3 10_0110 - AOI1_OUT0 10_0111 - AOI1_OUT1 10_1000 - AOI1_OUT2 10_1001 - AOI1_OUT3 10_1010 - EXTTRIG_IN0 10_1011 - EXTTRIG_IN1 10_1100 - EXTTRIG_IN2 10_1101 - EXTTRIG_IN3 10_1110 - EXTTRIG_IN4 10_1111 - EXTTRIG_IN5 11_0000 - EXTTRIG_IN6 11_0001 - EXTTRIG_IN7 11_0010 - EXTTRIG_IN8 11_0011 - EXTTRIG_IN9 11_0100 - DMA0_TRIGOUT0 11_0101 - DMA0_TRIGOUT1 11_0110 - DMA0_TRIGOUT2

14.5.1.1.26 ENC0 Input Connections (ENC0_HOME)

Offset

Register	Offset
ENC0_HOME	364h

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 ENC0_HOME	ENC0 Input Connections 00_0000 - PIN_INT0 00_0001 - PIN_INT4 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT1 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T1_MAT0 00_1001 - T3_MAT0 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2]

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM1_SM0_MUX_TRIG0 01_1001 - PWM1_SM0_MUX_TRIG1 01_1010 - PWM1_SM0_MUX_TRIG0 01_1011 - PWM1_SM1_MUX_TRIG1 01_1100 - PWM1_SM2_MUX_TRIG0 01_1101 - PWM1_SM2_MUX_TRIG1 01_1110 - PWM1_SM3_MUX_TRIG0 01_1111 - PWM1_SM3_MUX_TRIG1 10_0000 - ENC0_CMP/POS_MATCH 10_0001 - ENC1_CMP/POS_MATCH 10_0010 - AOI0_OUT0 10_0011 - AOI0_OUT1 10_0100 - AOI0_OUT2 10_0101 - AOI0_OUT3 10_0110 - AOI1_OUT0 10_0111 - AOI1_OUT1 10_1000 - AOI1_OUT2 10_1001 - AOI1_OUT3 10_1010 - EXTTRIG_IN0 10_1011 - EXTTRIG_IN1 10_1100 - EXTTRIG_IN2 10_1101 - EXTTRIG_IN3 10_1110 - EXTTRIG_IN4 10_1111 - EXTTRIG_IN5 11_0000 - EXTTRIG_IN6 11_0001 - EXTTRIG_IN7 11_0010 - EXTTRIG_IN8 11_0011 - EXTTRIG_IN9 11_0100 - DMA0_TRIGOUT0 11_0101 - DMA0_TRIGOUT1

Table continues on the next page...

Table continued from the previous page...

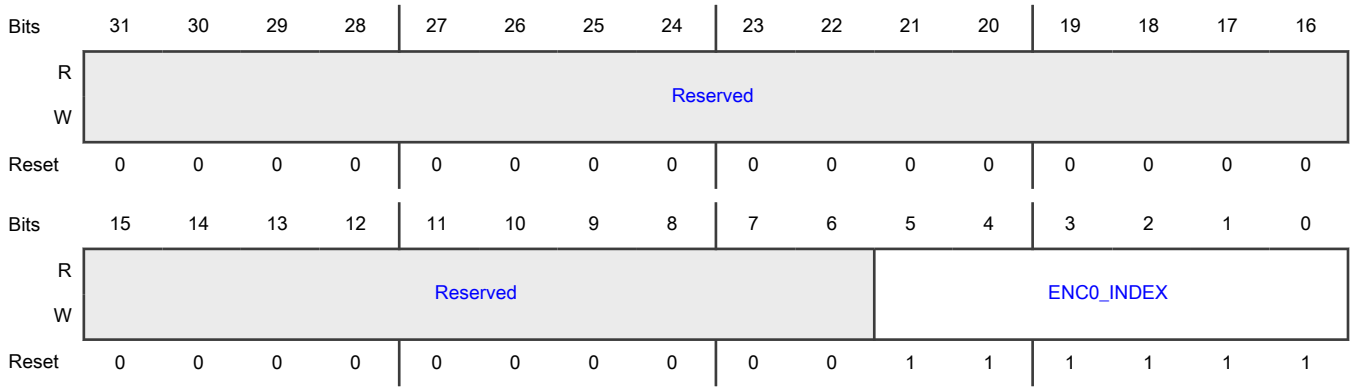
Field	Description
	11_0110 - DMA0_TRIGOUT2

14.5.1.1.27 ENC0 Input Connections (ENC0_INDEX)

Offset

Register	Offset
ENC0_INDEX	368h

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 ENC0_INDEX	ENC0 Input Connections 00_0000 - PIN_INT0 00_0001 - PIN_INT4 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT1 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T1_MAT0

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00_1001 - T3_MAT0 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2] 01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM1_SM0_MUX_TRIG0 01_1001 - PWM1_SM0_MUX_TRIG1 01_1010 - PWM1_SM1_MUX_TRIG0 01_1011 - PWM1_SM1_MUX_TRIG1 01_1100 - PWM1_SM2_MUX_TRIG0 01_1101 - PWM1_SM2_MUX_TRIG1 01_1110 - PWM1_SM3_MUX_TRIG0 01_1111 - PWM1_SM3_MUX_TRIG1 10_0000 - ENC0_CMP/POS_MATCH 10_0001 - ENC1_CMP/POS_MATCH 10_0010 - AOI0_OUT0 10_0011 - AOI0_OUT1 10_0100 - AOI0_OUT2 10_0101 - AOI0_OUT3 10_0110 - AOI1_OUT0 10_0111 - AOI1_OUT1 10_1000 - AOI1_OUT2 10_1001 - AOI1_OUT3 10_1010 - EXTTRIG_IN0

Table continues on the next page...

Table continued from the previous page...

Field	Description
	10_1011 - EXTTRIG_IN1 10_1100 - EXTTRIG_IN2 10_1101 - EXTTRIG_IN3 10_1110 - EXTTRIG_IN4 10_1111 - EXTTRIG_IN5 11_0000 - EXTTRIG_IN6 11_0001 - EXTTRIG_IN7 11_0010 - EXTTRIG_IN8 11_0011 - EXTTRIG_IN9 11_0100 - DMA0_TRIGOUT0 11_0101 - DMA0_TRIGOUT1 11_0110 - DMA0_TRIGOUT2

14.5.1.1.28 ENC0 Input Connections (ENC0_PHASEB)

Offset

Register	Offset
ENC0_PHASEB	36Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	Reserved										ENC0_PHASEB					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Description
31-6	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
5-0 ENC0_PHASEB	ENC0 Input Connections 00_0000 - PIN_INT0 00_0001 - PIN_INT4 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT1 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T1_MAT0 00_1001 - T3_MAT0 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2] 01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM1_SM0_MUX_TRIG0 01_1001 - PWM1_SM0_MUX_TRIG1 01_1010 - PWM1_SM1_MUX_TRIG0 01_1011 - PWM1_SM1_MUX_TRIG1 01_1100 - PWM1_SM2_MUX_TRIG0 01_1101 - PWM1_SM2_MUX_TRIG1 01_1110 - PWM1_SM3_MUX_TRIG0

Table continues on the next page...

Table continued from the previous page...

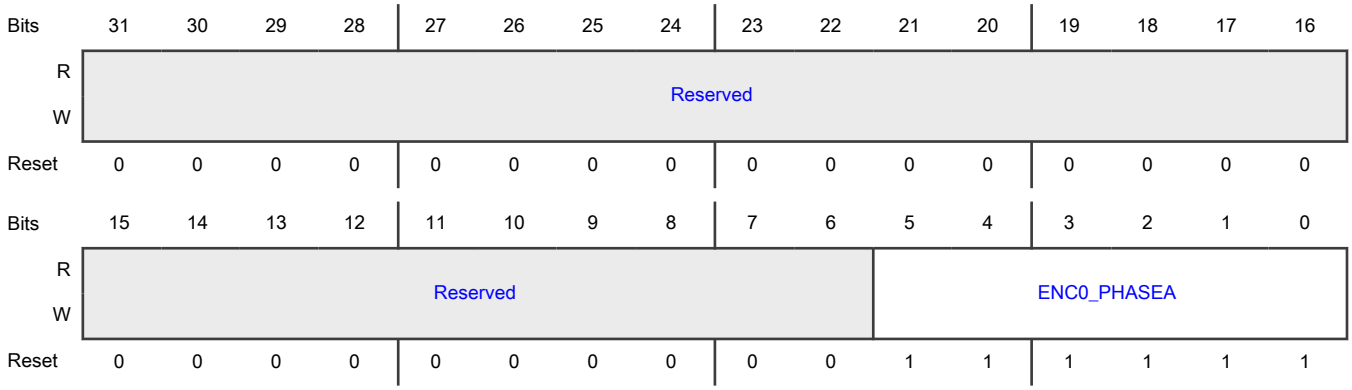
Field	Description
	01_1111 - PWM1_SM3_MUX_TRIG1 10_0000 - ENC0_CMP/POS_MATCH 10_0001 - ENC1_CMP/POS_MATCH 10_0010 - AOI0_OUT0 10_0011 - AOI0_OUT1 10_0100 - AOI0_OUT2 10_0101 - AOI0_OUT3 10_0110 - AOI1_OUT0 10_0111 - AOI1_OUT1 10_1000 - AOI1_OUT2 10_1001 - AOI1_OUT3 10_1010 - EXTTRIG_IN0 10_1011 - EXTTRIG_IN1 10_1100 - EXTTRIG_IN2 10_1101 - EXTTRIG_IN3 10_1110 - EXTTRIG_IN4 10_1111 - EXTTRIG_IN5 11_0000 - EXTTRIG_IN6 11_0001 - EXTTRIG_IN7 11_0010 - EXTTRIG_IN8 11_0011 - EXTTRIG_IN9 11_0100 - DMA0_TRIGOUT0 11_0101 - DMA0_TRIGOUT1 11_0110 - DMA0_TRIGOUT2

14.5.1.1.29 ENC0 Input Connections (ENC0_PHASEA)

Offset

Register	Offset
ENC0_PHASEA	370h

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 ENC0_PHASEA	ENC0 Input Connections 00_0000 - PIN_INT0 00_0001 - PIN_INT4 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT1 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T1_MAT0 00_1001 - T3_MAT0 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2]

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM1_SM0_MUX_TRIG0 01_1001 - PWM1_SM0_MUX_TRIG1 01_1010 - PWM1_SM1_MUX_TRIG0 01_1011 - PWM1_SM1_MUX_TRIG1 01_1100 - PWM1_SM2_MUX_TRIG0 01_1101 - PWM1_SM2_MUX_TRIG1 01_1110 - PWM1_SM3_MUX_TRIG0 01_1111 - PWM1_SM3_MUX_TRIG1 10_0000 - ENC0_CMP/POS_MATCH 10_0001 - ENC1_CMP/POS_MATCH 10_0010 - AOI0_OUT0 10_0011 - AOI0_OUT1 10_0100 - AOI0_OUT2 10_0101 - AOI0_OUT3 10_0110 - AOI1_OUT0 10_0111 - AOI1_OUT1 10_1000 - AOI1_OUT2 10_1001 - AOI1_OUT3 10_1010 - EXTTRIG_IN0 10_1011 - EXTTRIG_IN1 10_1100 - EXTTRIG_IN2 10_1101 - EXTTRIG_IN3 10_1110 - EXTTRIG_IN4 10_1111 - EXTTRIG_IN5 11_0000 - EXTTRIG_IN6 11_0001 - EXTTRIG_IN7 11_0010 - EXTTRIG_IN8 11_0011 - EXTTRIG_IN9 11_0100 - DMA0_TRIGOUT0 11_0101 - DMA0_TRIGOUT1

Table continues on the next page...

Table continued from the previous page...

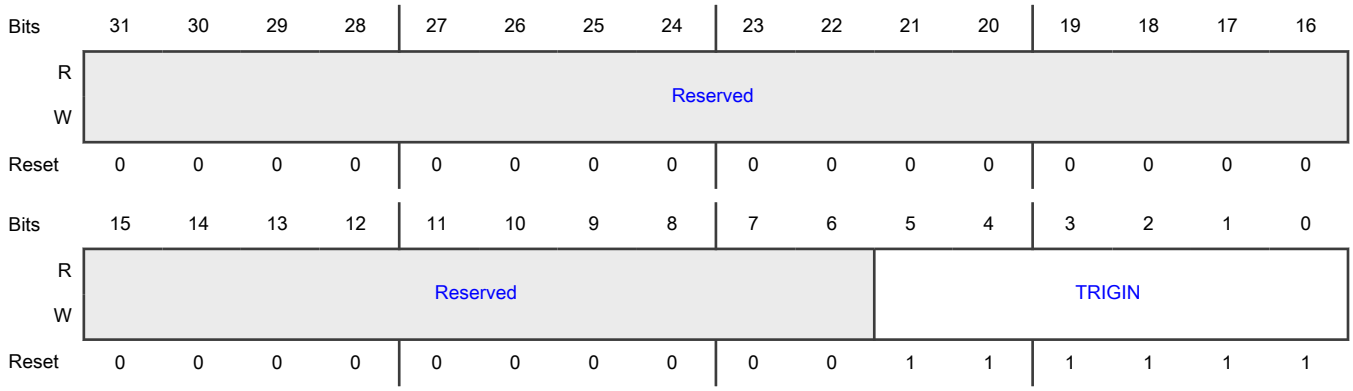
Field	Description
	11_0110 - DMA0_TRIGOUT2

14.5.1.1.30 ENC1 Trigger Input Connections (ENC1_TRIG)

Offset

Register	Offset
ENC1_TRIG	380h

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 TRIGIN	ENC1 input trigger 00_0000 - PIN_INT0 00_0001 - PIN_INT5 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT7 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T1_MAT1

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00_1001 - T3_MAT1 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2] 01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM1_SM0_MUX_TRIG0 01_1001 - PWM1_SM0_MUX_TRIG1 01_1010 - PWM1_SM1_MUX_TRIG0 01_1011 - PWM1_SM1_MUX_TRIG1 01_1100 - PWM1_SM2_MUX_TRIG0 01_1101 - PWM1_SM2_MUX_TRIG1 01_1110 - PWM1_SM3_MUX_TRIG0 01_1111 - PWM1_SM3_MUX_TRIG1 10_0000 - ENC0_CMP/POS_MATCH 10_0001 - ENC1_CMP/POS_MATCH 10_0010 - AOI0_OUT0 10_0011 - AOI0_OUT1 10_0100 - AOI0_OUT2 10_0101 - AOI0_OUT3 10_0110 - AOI1_OUT0 10_0111 - AOI1_OUT1 10_1000 - AOI1_OUT2 10_1001 - AOI1_OUT3 10_1010 - EXTTRIG_IN0

Table continues on the next page...

Table continued from the previous page...

Field	Description
	10_1011 - EXTTRIG_IN1 10_1100 - EXTTRIG_IN2 10_1101 - EXTTRIG_IN3 10_1110 - EXTTRIG_IN4 10_1111 - EXTTRIG_IN5 11_0000 - EXTTRIG_IN6 11_0001 - EXTTRIG_IN7 11_0010 - EXTTRIG_IN8 11_0011 - EXTTRIG_IN9 11_0100 - DMA0_TRIGOUT0 11_0101 - DMA0_TRIGOUT1 11_0110 - DMA0_TRIGOUT2

14.5.1.1.31 ENC1 Input Connections (ENC1_HOME)

Offset

Register	Offset
ENC1_HOME	384h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	Reserved										ENC1_HOME					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Description
31-6	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
5-0 ENC1_HOME	ENC1 input trigger 00_0000 - PIN_INT0 00_0001 - PIN_INT5 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT7 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T1_MAT1 00_1001 - T3_MAT1 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2] 01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM1_SM0_MUX_TRIG0 01_1001 - PWM1_SM0_MUX_TRIG1 01_1010 - PWM1_SM1_MUX_TRIG0 01_1011 - PWM1_SM1_MUX_TRIG1 01_1100 - PWM1_SM2_MUX_TRIG0 01_1101 - PWM1_SM2_MUX_TRIG1 01_1110 - PWM1_SM3_MUX_TRIG0

Table continues on the next page...

Table continued from the previous page...

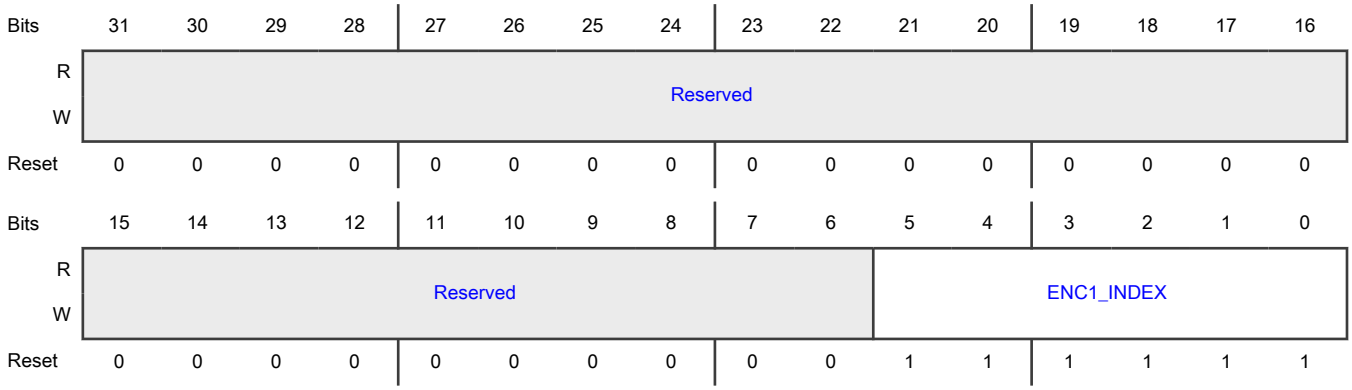
Field	Description
	01_1111 - PWM1_SM3_MUX_TRIG1 10_0000 - ENC0_CMP/POS_MATCH 10_0001 - ENC1_CMP/POS_MATCH 10_0010 - AOI0_OUT0 10_0011 - AOI0_OUT1 10_0100 - AOI0_OUT2 10_0101 - AOI0_OUT3 10_0110 - AOI1_OUT0 10_0111 - AOI1_OUT1 10_1000 - AOI1_OUT2 10_1001 - AOI1_OUT3 10_1010 - EXTTRIG_IN0 10_1011 - EXTTRIG_IN1 10_1100 - EXTTRIG_IN2 10_1101 - EXTTRIG_IN3 10_1110 - EXTTRIG_IN4 10_1111 - EXTTRIG_IN5 11_0000 - EXTTRIG_IN6 11_0001 - EXTTRIG_IN7 11_0010 - EXTTRIG_IN8 11_0011 - EXTTRIG_IN9 11_0100 - DMA0_TRIGOUT0 11_0101 - DMA0_TRIGOUT1 11_0110 - DMA0_TRIGOUT2

14.5.1.1.32 ENC1 Input Connections (ENC1_INDEX)

Offset

Register	Offset
ENC1_INDEX	388h

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 ENC1_INDEX	ENC1 input trigger 00_0000 - PIN_INT0 00_0001 - PIN_INT5 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT7 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T1_MAT1 00_1001 - T3_MAT1 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2]

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM1_SM0_MUX_TRIG0 01_1001 - PWM1_SM0_MUX_TRIG1 01_1010 - PWM1_SM1_MUX_TRIG0 01_1011 - PWM1_SM1_MUX_TRIG1 01_1100 - PWM1_SM2_MUX_TRIG0 01_1101 - PWM1_SM2_MUX_TRIG1 01_1110 - PWM1_SM3_MUX_TRIG0 01_1111 - PWM1_SM3_MUX_TRIG1 10_0000 - ENC0_CMP/POS_MATCH 10_0001 - ENC1_CMP/POS_MATCH 10_0010 - AOI0_OUT0 10_0011 - AOI0_OUT1 10_0100 - AOI0_OUT2 10_0101 - AOI0_OUT3 10_0110 - AOI1_OUT0 10_0111 - AOI1_OUT1 10_1000 - AOI1_OUT2 10_1001 - AOI1_OUT3 10_1010 - EXTTRIG_IN0 10_1011 - EXTTRIG_IN1 10_1100 - EXTTRIG_IN2 10_1101 - EXTTRIG_IN3 10_1110 - EXTTRIG_IN4 10_1111 - EXTTRIG_IN5 11_0000 - EXTTRIG_IN6 11_0001 - EXTTRIG_IN7 11_0010 - EXTTRIG_IN8 11_0011 - EXTTRIG_IN9 11_0100 - DMA0_TRIGOUT0 11_0101 - DMA0_TRIGOUT1

Table continues on the next page...

Table continued from the previous page...

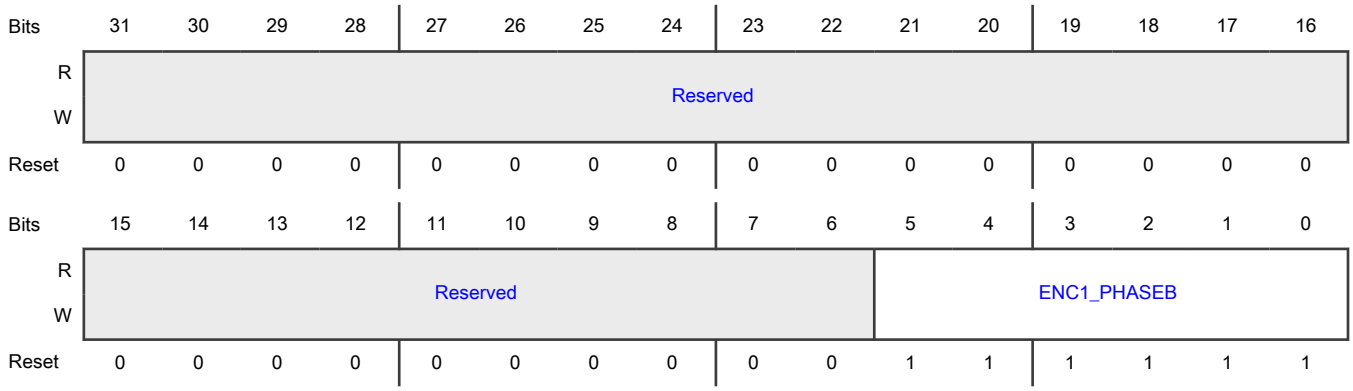
Field	Description
	11_0110 - DMA0_TRIGOUT2

14.5.1.1.33 ENC1 Input Connections (ENC1_PHASEB)

Offset

Register	Offset
ENC1_PHASEB	38Ch

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 ENC1_PHASEB	ENC1 input trigger 00_0000 - PIN_INT0 00_0001 - PIN_INT5 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT7 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T1_MAT1

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00_1001 - T3_MAT1 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2] 01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM1_SM0_MUX_TRIG0 01_1001 - PWM1_SM0_MUX_TRIG1 01_1010 - PWM1_SM1_MUX_TRIG0 01_1011 - PWM1_SM1_MUX_TRIG1 01_1100 - PWM1_SM2_MUX_TRIG0 01_1101 - PWM1_SM2_MUX_TRIG1 01_1110 - PWM1_SM3_MUX_TRIG0 01_1111 - PWM1_SM3_MUX_TRIG1 10_0000 - ENC0_CMP/POS_MATCH 10_0001 - ENC1_CMP/POS_MATCH 10_0010 - AOI0_OUT0 10_0011 - AOI0_OUT1 10_0100 - AOI0_OUT2 10_0101 - AOI0_OUT3 10_0110 - AOI1_OUT0 10_0111 - AOI1_OUT1 10_1000 - AOI1_OUT2 10_1001 - AOI1_OUT3 10_1010 - EXTTRIG_IN0

Table continues on the next page...

Table continued from the previous page...

Field	Description
	10_1011 - EXTTRIG_IN1 10_1100 - EXTTRIG_IN2 10_1101 - EXTTRIG_IN3 10_1110 - EXTTRIG_IN4 10_1111 - EXTTRIG_IN5 11_0000 - EXTTRIG_IN6 11_0001 - EXTTRIG_IN7 11_0010 - EXTTRIG_IN8 11_0011 - EXTTRIG_IN9 11_0100 - DMA0_TRIGOUT0 11_0101 - DMA0_TRIGOUT1 11_0110 - DMA0_TRIGOUT2

14.5.1.1.34 ENC1 Input Connections (ENC1_PHASEA)

Offset

Register	Offset
ENC1_PHASEA	390h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	Reserved										ENC1_PHASEA					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Description
31-6	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
5-0 ENC1_PHASEA	ENC1 input trigger 00_0000 - PIN_INT0 00_0001 - PIN_INT5 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT7 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T1_MAT1 00_1001 - T3_MAT1 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2] 01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM1_SM0_MUX_TRIG0 01_1001 - PWM1_SM0_MUX_TRIG1 01_1010 - PWM1_SM1_MUX_TRIG0 01_1011 - PWM1_SM1_MUX_TRIG1 01_1100 - PWM1_SM2_MUX_TRIG0 01_1101 - PWM1_SM2_MUX_TRIG1 01_1110 - PWM1_SM3_MUX_TRIG0

Table continues on the next page...

Table continued from the previous page...

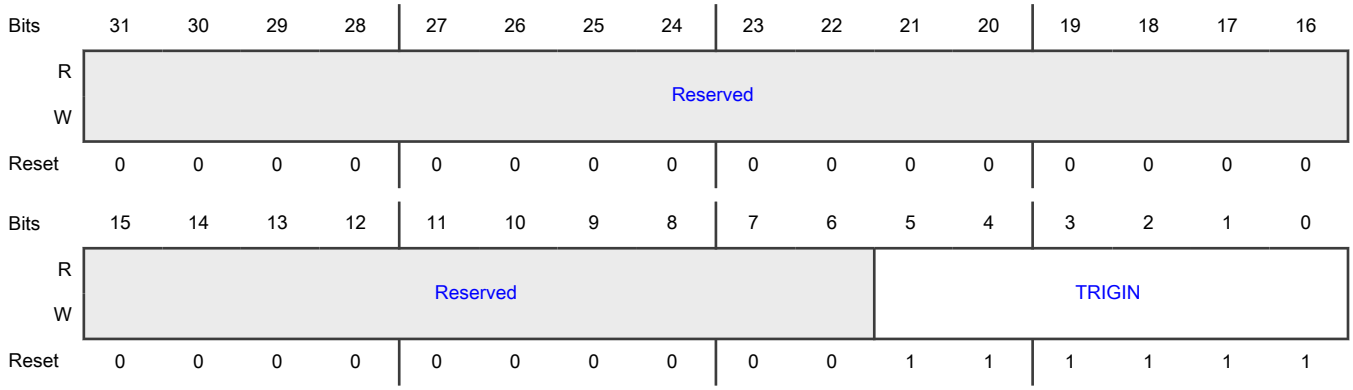
Field	Description
	01_1111 - PWM1_SM3_MUX_TRIG1 10_0000 - ENC0_CMP/POS_MATCH 10_0001 - ENC1_CMP/POS_MATCH 10_0010 - AOI0_OUT0 10_0011 - AOI0_OUT1 10_0100 - AOI0_OUT2 10_0101 - AOI0_OUT3 10_0110 - AOI1_OUT0 10_0111 - AOI1_OUT1 10_1000 - AOI1_OUT2 10_1001 - AOI1_OUT3 10_1010 - EXTTRIG_IN0 10_1011 - EXTTRIG_IN1 10_1100 - EXTTRIG_IN2 10_1101 - EXTTRIG_IN3 10_1110 - EXTTRIG_IN4 10_1111 - EXTTRIG_IN5 11_0000 - EXTTRIG_IN6 11_0001 - EXTTRIG_IN7 11_0010 - EXTTRIG_IN8 11_0011 - EXTTRIG_IN9 11_0100 - DMA0_TRIGOUT0 11_0101 - DMA0_TRIGOUT1 11_0110 - DMA0_TRIGOUT2

14.5.1.1.35 PWM0 external synchronization (PWM0_EXTSYNC0 - PWM0_EXTSYNC3)

Offset

Register	Offset
PWM0_EXTSYNC0	3A0h
PWM0_EXTSYNC1	3A4h
PWM0_EXTSYNC2	3A8h
PWM0_EXTSYNC3	3ACh

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 TRIGIN	Trigger input connections for PWM0 00_0000 - PIN_INT0 00_0001 - PIN_INT5 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT2 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T2_MAT0 00_1001 - T4_MAT0 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2]

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM1_SM0_MUX_TRIG0 01_1001 - PWM1_SM0_MUX_TRIG1 01_1010 - PWM1_SM1_MUX_TRIG0 01_1011 - PWM1_SM1_MUX_TRIG1 01_1100 - PWM1_SM2_MUX_TRIG0 01_1101 - PWM1_SM2_MUX_TRIG1 01_1110 - PWM1_SM3_MUX_TRIG0 01_1111 - PWM1_SM3_MUX_TRIG1 10_0000 - ENC0_CMP/POS_MATCH 10_0001 - ENC1_CMP/POS_MATCH 10_0010 - AOI0_OUT0 10_0011 - AOI0_OUT1 10_0100 - AOI0_OUT2 10_0101 - AOI0_OUT3 10_0110 - AOI1_OUT0 10_0111 - AOI1_OUT1 10_1000 - AOI1_OUT2 10_1001 - AOI1_OUT3 10_1010 - EXTTRIG_IN0 10_1011 - EXTTRIG_IN1 10_1100 - EXTTRIG_IN2 10_1101 - EXTTRIG_IN3 10_1110 - EXTTRIG_IN4 10_1111 - EXTTRIG_IN5 11_0000 - EXTTRIG_IN6 11_0001 - EXTTRIG_IN7 11_0010 - EXTTRIG_IN8 11_0011 - EXTTRIG_IN9 11_0100 - DMA0_TRIGOUT0 11_0101 - DMA0_TRIGOUT1

Table continues on the next page...

Table continued from the previous page...

Field	Description
	11_0110 - DMA0_TRIGOUT2

14.5.1.1.36 PWM0 input trigger connections (PWM0_EXTA0 - PWM0_EXTA3)

Offset

Register	Offset
PWM0_EXTA0	3B0h
PWM0_EXTA1	3B4h
PWM0_EXTA2	3B8h
PWM0_EXTA3	3BCh

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	Reserved										TRIGIN					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Description
31-6 —	Reserved
5-0 TRIGIN	Trigger input connections for PWM0 00_0000 - PIN_INT0 00_0001 - PIN_INT5 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT2 00_0101 - T0_MAT3

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T2_MAT0 00_1001 - T4_MAT0 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2] 01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM1_SM0_MUX_TRIG0 01_1001 - PWM1_SM0_MUX_TRIG1 01_1010 - PWM1_SM1_MUX_TRIG0 01_1011 - PWM1_SM1_MUX_TRIG1 01_1100 - PWM1_SM2_MUX_TRIG0 01_1101 - PWM1_SM2_MUX_TRIG1 01_1110 - PWM1_SM3_MUX_TRIG0 01_1111 - PWM1_SM3_MUX_TRIG1 10_0000 - ENC0_CMP/POS_MATCH 10_0001 - ENC1_CMP/POS_MATCH 10_0010 - AOI0_OUT0 10_0011 - AOI0_OUT1 10_0100 - AOI0_OUT2 10_0101 - AOI0_OUT3 10_0110 - AOI1_OUT0 10_0111 - AOI1_OUT1

Table continues on the next page...

Table continued from the previous page...

Field	Description
	10_1000 - AOI1_OUT2 10_1001 - AOI1_OUT3 10_1010 - EXTTRIG_IN0 10_1011 - EXTTRIG_IN1 10_1100 - EXTTRIG_IN2 10_1101 - EXTTRIG_IN3 10_1110 - EXTTRIG_IN4 10_1111 - EXTTRIG_IN5 11_0000 - EXTTRIG_IN6 11_0001 - EXTTRIG_IN7 11_0010 - EXTTRIG_IN8 11_0011 - EXTTRIG_IN9 11_0100 - DMA0_TRIGOUT0 11_0101 - DMA0_TRIGOUT1 11_0110 - DMA0_TRIGOUT2

14.5.1.1.37 PWM0 external force trigger connections (PWM0_EXTFORCE)

Offset

Register	Offset
PWM0_EXTFORCE	3C0h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	Reserved										TRIGIN					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Description
31-6 —	Reserved
5-0 TRIGIN	Trigger input connections for PWM0 00_0000 - PIN_INT0 00_0001 - PIN_INT5 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT2 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T2_MAT0 00_1001 - T4_MAT0 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2] 01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM1_SM0_MUX_TRIG0 01_1001 - PWM1_SM0_MUX_TRIG1 01_1010 - PWM1_SM1_MUX_TRIG0 01_1011 - PWM1_SM1_MUX_TRIG1 01_1100 - PWM1_SM2_MUX_TRIG0 01_1101 - PWM1_SM2_MUX_TRIG1

Table continues on the next page...

Table continued from the previous page...

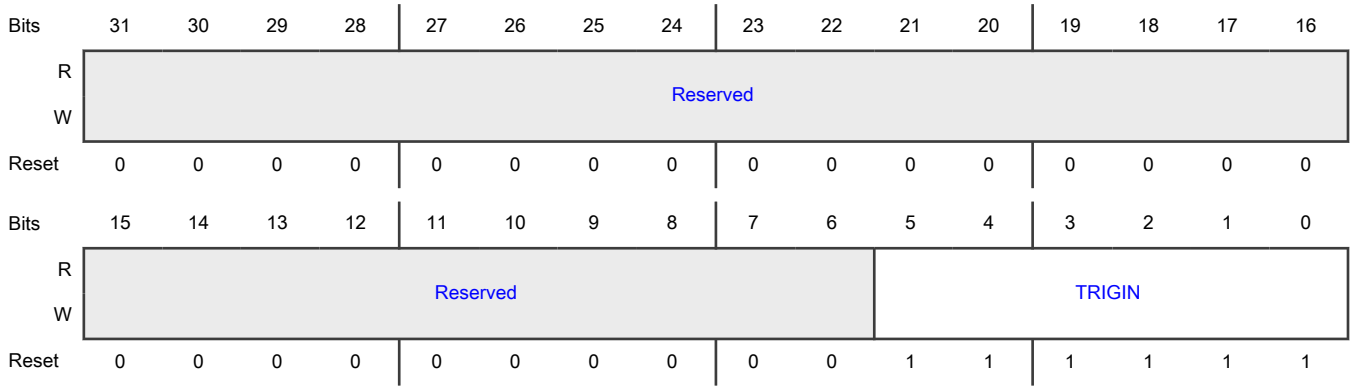
Field	Description
	01_1110 - PWM1_SM3_MUX_TRIG0 01_1111 - PWM1_SM3_MUX_TRIG1 10_0000 - ENC0_CMP/POS_MATCH 10_0001 - ENC1_CMP/POS_MATCH 10_0010 - AOI0_OUT0 10_0011 - AOI0_OUT1 10_0100 - AOI0_OUT2 10_0101 - AOI0_OUT3 10_0110 - AOI1_OUT0 10_0111 - AOI1_OUT1 10_1000 - AOI1_OUT2 10_1001 - AOI1_OUT3 10_1010 - EXTTRIG_IN0 10_1011 - EXTTRIG_IN1 10_1100 - EXTTRIG_IN2 10_1101 - EXTTRIG_IN3 10_1110 - EXTTRIG_IN4 10_1111 - EXTTRIG_IN5 11_0000 - EXTTRIG_IN6 11_0001 - EXTTRIG_IN7 11_0010 - EXTTRIG_IN8 11_0011 - EXTTRIG_IN9 11_0100 - DMA0_TRIGOUT0 11_0101 - DMA0_TRIGOUT1 11_0110 - DMA0_TRIGOUT2

14.5.1.1.38 PWM0 fault input trigger connections (PWM0_FAULT0 - PWM0_FAULT3)

Offset

Register	Offset
PWM0_FAULT0	3C4h
PWM0_FAULT1	3C8h
PWM0_FAULT2	3CCh
PWM0_FAULT3	3D0h

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 TRIGIN	Trigger input connections for PWM0 00_0000 - PIN_INT0 00_0001 - PIN_INT5 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT2 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T2_MAT0 00_1001 - T4_MAT0 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2]

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM1_SM0_MUX_TRIG0 01_1001 - PWM1_SM0_MUX_TRIG1 01_1010 - PWM1_SM1_MUX_TRIG0 01_1011 - PWM1_SM1_MUX_TRIG1 01_1100 - PWM1_SM2_MUX_TRIG0 01_1101 - PWM1_SM2_MUX_TRIG1 01_1110 - PWM1_SM3_MUX_TRIG0 01_1111 - PWM1_SM3_MUX_TRIG1 10_0000 - ENC0_CMP/POS_MATCH 10_0001 - ENC1_CMP/POS_MATCH 10_0010 - AOI0_OUT0 10_0011 - AOI0_OUT1 10_0100 - AOI0_OUT2 10_0101 - AOI0_OUT3 10_0110 - AOI1_OUT0 10_0111 - AOI1_OUT1 10_1000 - AOI1_OUT2 10_1001 - AOI1_OUT3 10_1010 - EXTTRIG_IN0 10_1011 - EXTTRIG_IN1 10_1100 - EXTTRIG_IN2 10_1101 - EXTTRIG_IN3 10_1110 - EXTTRIG_IN4 10_1111 - EXTTRIG_IN5 11_0000 - EXTTRIG_IN6 11_0001 - EXTTRIG_IN7 11_0010 - EXTTRIG_IN8 11_0011 - EXTTRIG_IN9 11_0100 - DMA0_TRIGOUT0 11_0101 - DMA0_TRIGOUT1

Table continues on the next page...

Table continued from the previous page...

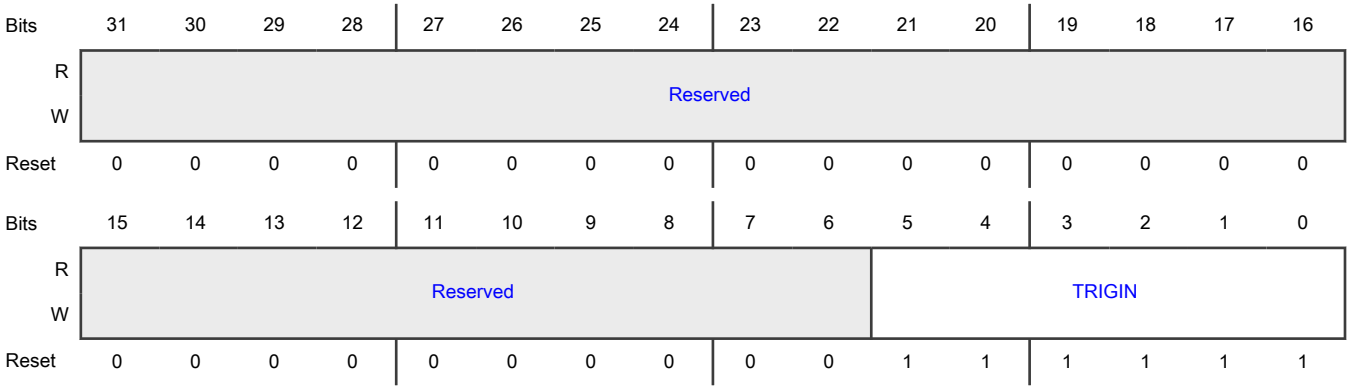
Field	Description
	11_0110 - DMA0_TRIGOUT2

14.5.1.1.39 PWM1 external synchronization (PWM1_EXTSYNC0 - PWM1_EXTSYNC3)

Offset

Register	Offset
PWM1_EXTSYNC0	3E0h
PWM1_EXTSYNC1	3E4h
PWM1_EXTSYNC2	3E8h
PWM1_EXTSYNC3	3ECh

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 TRIGIN	Trigger input connections for PWM1 00_0000 - PIN_INT0 00_0001 - PIN_INT2 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT3 00_0101 - T0_MAT3

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T2_MAT1 00_1001 - T4_MAT1 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2] 01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM0_SM0_MUX_TRIG0 01_1001 - PWM0_SM0_MUX_TRIG1 01_1010 - PWM0_SM1_MUX_TRIG0 01_1011 - PWM0_SM1_MUX_TRIG1 01_1100 - PWM0_SM2_MUX_TRIG0 01_1101 - PWM0_SM2_MUX_TRIG1 01_1110 - PWM0_SM3_MUX_TRIG0 01_1111 - PWM0_SM3_MUX_TRIG1 10_0000 - ENC0_CMP/POS_MATCH 10_0001 - ENC1_CMP/POS_MATCH 10_0010 - AOI0_OUT0 10_0011 - AOI0_OUT1 10_0100 - AOI0_OUT2 10_0101 - AOI0_OUT3 10_0110 - AOI1_OUT0 10_0111 - AOI1_OUT1

Table continues on the next page...

Table continued from the previous page...

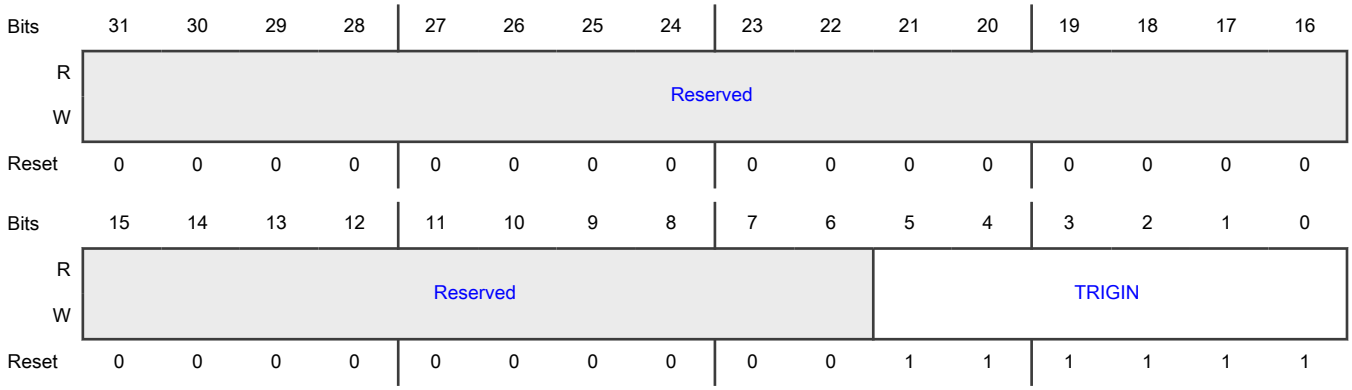
Field	Description
	10_1000 - AOI1_OUT2 10_1001 - AOI1_OUT3 10_1010 - EXTTRIG_IN0 10_1011 - EXTTRIG_IN1 10_1100 - EXTTRIG_IN2 10_1101 - EXTTRIG_IN3 10_1110 - EXTTRIG_IN4 10_1111 - EXTTRIG_IN5 11_0000 - EXTTRIG_IN6 11_0001 - EXTTRIG_IN7 11_0010 - EXTTRIG_IN8 11_0011 - EXTTRIG_IN9 11_0100 - DMA0_TRIGOUT0 11_0101 - DMA0_TRIGOUT1 11_0110 - DMA0_TRIGOUT2

14.5.1.1.40 PWM1 input trigger connections (PWM1_EXTA0 - PWM1_EXTA3)

Offset

Register	Offset
PWM1_EXTA0	3F0h
PWM1_EXTA1	3F4h
PWM1_EXTA2	3F8h
PWM1_EXTA3	3FCh

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 TRIGIN	Trigger input connections for PWM1 00_0000 - PIN_INT0 00_0001 - PIN_INT2 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT3 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T2_MAT1 00_1001 - T4_MAT1 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2] 01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM0_SM0_MUX_TRIG0 01_1001 - PWM0_SM0_MUX_TRIG1 01_1010 - PWM0_SM1_MUX_TRIG0 01_1011 - PWM0_SM1_MUX_TRIG1 01_1100 - PWM0_SM2_MUX_TRIG0 01_1101 - PWM0_SM2_MUX_TRIG1

Table continues on the next page...

Table continued from the previous page...

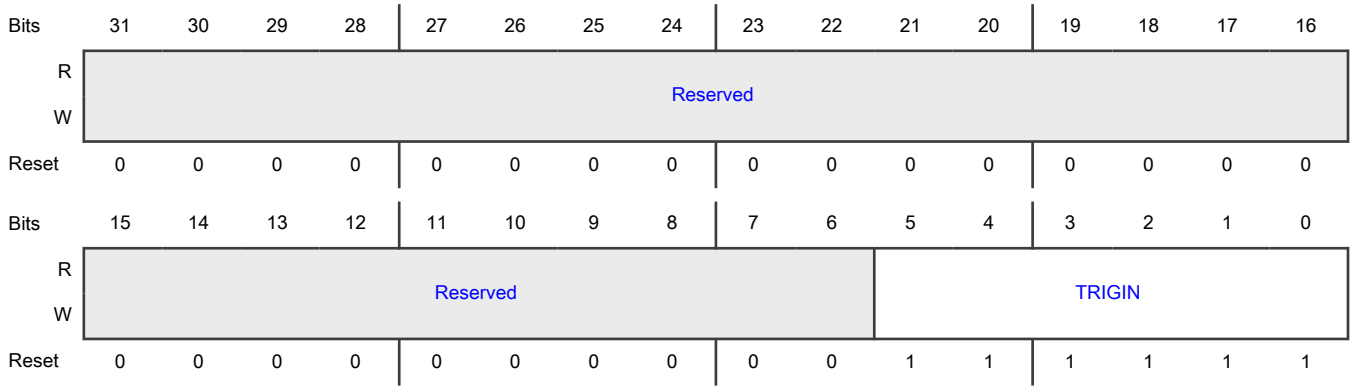
Field	Description
	01_1110 - PWM0_SM3_MUX_TRIG0 01_1111 - PWM0_SM3_MUX_TRIG1 10_0000 - ENC0_CMP/POS_MATCH 10_0001 - ENC1_CMP/POS_MATCH 10_0010 - AOI0_OUT0 10_0011 - AOI0_OUT1 10_0100 - AOI0_OUT2 10_0101 - AOI0_OUT3 10_0110 - AOI1_OUT0 10_0111 - AOI1_OUT1 10_1000 - AOI1_OUT2 10_1001 - AOI1_OUT3 10_1010 - EXTTRIG_IN0 10_1011 - EXTTRIG_IN1 10_1100 - EXTTRIG_IN2 10_1101 - EXTTRIG_IN3 10_1110 - EXTTRIG_IN4 10_1111 - EXTTRIG_IN5 11_0000 - EXTTRIG_IN6 11_0001 - EXTTRIG_IN7 11_0010 - EXTTRIG_IN8 11_0011 - EXTTRIG_IN9 11_0100 - DMA0_TRIGOUT0 11_0101 - DMA0_TRIGOUT1 11_0110 - DMA0_TRIGOUT2

14.5.1.1.41 PWM1 external force trigger connections (PWM1_EXTFORCE)

Offset

Register	Offset
PWM1_EXTFORCE	400h

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 TRIGIN	Trigger input connections for PWM1 00_0000 - PIN_INT0 00_0001 - PIN_INT2 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT3 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T2_MAT1 00_1001 - T4_MAT1 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2]

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM0_SM0_MUX_TRIG0 01_1001 - PWM0_SM0_MUX_TRIG1 01_1010 - PWM0_SM1_MUX_TRIG0 01_1011 - PWM0_SM1_MUX_TRIG1 01_1100 - PWM0_SM2_MUX_TRIG0 01_1101 - PWM0_SM2_MUX_TRIG1 01_1110 - PWM0_SM3_MUX_TRIG0 01_1111 - PWM0_SM3_MUX_TRIG1 10_0000 - ENC0_CMP/POS_MATCH 10_0001 - ENC1_CMP/POS_MATCH 10_0010 - AOI0_OUT0 10_0011 - AOI0_OUT1 10_0100 - AOI0_OUT2 10_0101 - AOI0_OUT3 10_0110 - AOI1_OUT0 10_0111 - AOI1_OUT1 10_1000 - AOI1_OUT2 10_1001 - AOI1_OUT3 10_1010 - EXTTRIG_IN0 10_1011 - EXTTRIG_IN1 10_1100 - EXTTRIG_IN2 10_1101 - EXTTRIG_IN3 10_1110 - EXTTRIG_IN4 10_1111 - EXTTRIG_IN5 11_0000 - EXTTRIG_IN6 11_0001 - EXTTRIG_IN7 11_0010 - EXTTRIG_IN8 11_0011 - EXTTRIG_IN9 11_0100 - DMA0_TRIGOUT0 11_0101 - DMA0_TRIGOUT1

Table continues on the next page...

Table continued from the previous page...

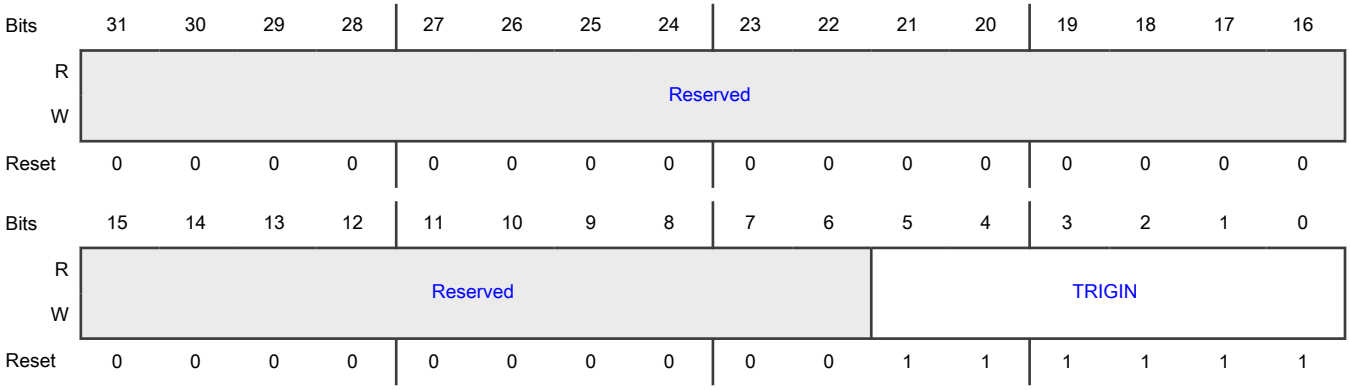
Field	Description
	11_0110 - DMA0_TRIGOUT2

14.5.1.1.42 PWM1 fault input trigger connections (PWM1_FAULT0 - PWM1_FAULT3)

Offset

Register	Offset
PWM1_FAULT0	404h
PWM1_FAULT1	408h
PWM1_FAULT2	40Ch
PWM1_FAULT3	410h

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 TRIGIN	Trigger input connections for PWM1 00_0000 - PIN_INT0 00_0001 - PIN_INT2 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT3 00_0101 - T0_MAT3

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T2_MAT1 00_1001 - T4_MAT1 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2] 01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM0_SM0_MUX_TRIG0 01_1001 - PWM0_SM0_MUX_TRIG1 01_1010 - PWM0_SM1_MUX_TRIG0 01_1011 - PWM0_SM1_MUX_TRIG1 01_1100 - PWM0_SM2_MUX_TRIG0 01_1101 - PWM0_SM2_MUX_TRIG1 01_1110 - PWM0_SM3_MUX_TRIG0 01_1111 - PWM0_SM3_MUX_TRIG1 10_0000 - ENC0_CMP/POS_MATCH 10_0001 - ENC1_CMP/POS_MATCH 10_0010 - AOI0_OUT0 10_0011 - AOI0_OUT1 10_0100 - AOI0_OUT2 10_0101 - AOI0_OUT3 10_0110 - AOI1_OUT0 10_0111 - AOI1_OUT1

Table continues on the next page...

Table continued from the previous page...

Field	Description
	10_1000 - AOI1_OUT2 10_1001 - AOI1_OUT3 10_1010 - EXTTRIG_IN0 10_1011 - EXTTRIG_IN1 10_1100 - EXTTRIG_IN2 10_1101 - EXTTRIG_IN3 10_1110 - EXTTRIG_IN4 10_1111 - EXTTRIG_IN5 11_0000 - EXTTRIG_IN6 11_0001 - EXTTRIG_IN7 11_0010 - EXTTRIG_IN8 11_0011 - EXTTRIG_IN9 11_0100 - DMA0_TRIGOUT0 11_0101 - DMA0_TRIGOUT1 11_0110 - DMA0_TRIGOUT2

14.5.1.1.43 PWM0 external clock trigger connections (PWM0_EXTCLK)

NOTE

Bits 3-5 are write only. Bits 0-2 are RW.

Offset

Register	Offset
PWM0_EXTCLK	420h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	Reserved										TRIGIN					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Fields

Field	Description
31-6 —	Reserved
5-0 TRIGIN	Trigger input connections for PWM0 00_0000 - PIN_INT0 00_0001 - PIN_INT5 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT2 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T2_MAT0 00_1001 - T4_MAT0 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2] 01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM1_SM0_MUX_TRIG0 01_1001 - PWM1_SM0_MUX_TRIG1 01_1010 - PWM1_SM1_MUX_TRIG0 01_1011 - PWM1_SM1_MUX_TRIG1 01_1100 - PWM1_SM2_MUX_TRIG0 01_1101 - PWM1_SM2_MUX_TRIG1

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01_1110 - PWM1_SM3_MUX_TRIG0 01_1111 - PWM1_SM3_MUX_TRIG1 10_0000 - ENC0_CMP/POS_MATCH 10_0001 - ENC1_CMP/POS_MATCH 10_0010 - AOI0_OUT0 10_0011 - AOI0_OUT1 10_0100 - AOI0_OUT2 10_0101 - AOI0_OUT3 10_0110 - AOI1_OUT0 10_0111 - AOI1_OUT1 10_1000 - AOI1_OUT2 10_1001 - AOI1_OUT3 10_1010 - EXTTRIG_IN0 10_1011 - EXTTRIG_IN1 10_1100 - EXTTRIG_IN2 10_1101 - EXTTRIG_IN3 10_1110 - EXTTRIG_IN4 10_1111 - EXTTRIG_IN5 11_0000 - EXTTRIG_IN6 11_0001 - EXTTRIG_IN7 11_0010 - EXTTRIG_IN8 11_0011 - EXTTRIG_IN9 11_0100 - DMA0_TRIGOUT0 11_0101 - DMA0_TRIGOUT1 11_0110 - DMA0_TRIGOUT2

14.5.1.1.44 PWM1 external clock trigger connections (PWM1_EXTCLK)

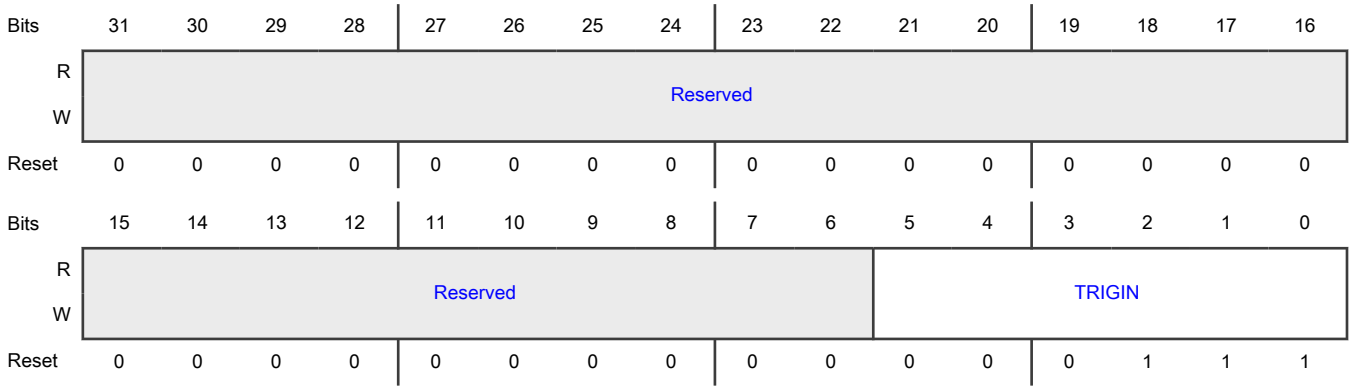
NOTE

Bits 3-5 are write only. Bits 0-2 are RW.

Offset

Register	Offset
PWM1_EXTCLK	424h

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 TRIGIN	Trigger input connections for PWM1 00_0000 - PIN_INT0 00_0001 - PIN_INT2 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT3 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T2_MAT1 00_1001 - T4_MAT1 00_1010 - ACMP0_OUT 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[0] 00_1110 - ADC0_tcomp[1] 00_1111 - ADC0_tcomp[2] 01_0000 - ADC0_tcomp[3] 01_0001 - ADC1_tcomp[0] 01_0010 - ADC1_tcomp[1] 01_0011 - ADC1_tcomp[2]

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01_0100 - ADC1_tcomp[3] 01_0101 - HSCMP0_OUT 01_0110 - HSCMP1_OUT 01_0111 - HSCMP2_OUT 01_1000 - PWM0_SM0_MUX_TRIG0 01_1001 - PWM0_SM0_MUX_TRIG1 01_1010 - PWM0_SM1_MUX_TRIG0 01_1011 - PWM0_SM1_MUX_TRIG1 01_1100 - PWM0_SM2_MUX_TRIG0 01_1101 - PWM0_SM2_MUX_TRIG1 01_1110 - PWM0_SM3_MUX_TRIG0 01_1111 - PWM0_SM3_MUX_TRIG1 10_0000 - ENC0_CMP/POS_MATCH 10_0001 - ENC1_CMP/POS_MATCH 10_0010 - AOI0_OUT0 10_0011 - AOI0_OUT1 10_0100 - AOI0_OUT2 10_0101 - AOI0_OUT3 10_0110 - AOI1_OUT0 10_0111 - AOI1_OUT1 10_1000 - AOI1_OUT2 10_1001 - AOI1_OUT3 10_1010 - EXTTRIG_IN0 10_1011 - EXTTRIG_IN1 10_1100 - EXTTRIG_IN2 10_1101 - EXTTRIG_IN3 10_1110 - EXTTRIG_IN4 10_1111 - EXTTRIG_IN5 11_0000 - EXTTRIG_IN6 11_0001 - EXTTRIG_IN7 11_0010 - EXTTRIG_IN8 11_0011 - EXTTRIG_IN9 11_0100 - DMA0_TRIGOUT0 11_0101 - DMA0_TRIGOUT1

Table continues on the next page...

Table continued from the previous page...

Field	Description
	11_0110 - DMA0_TRIGOUT2

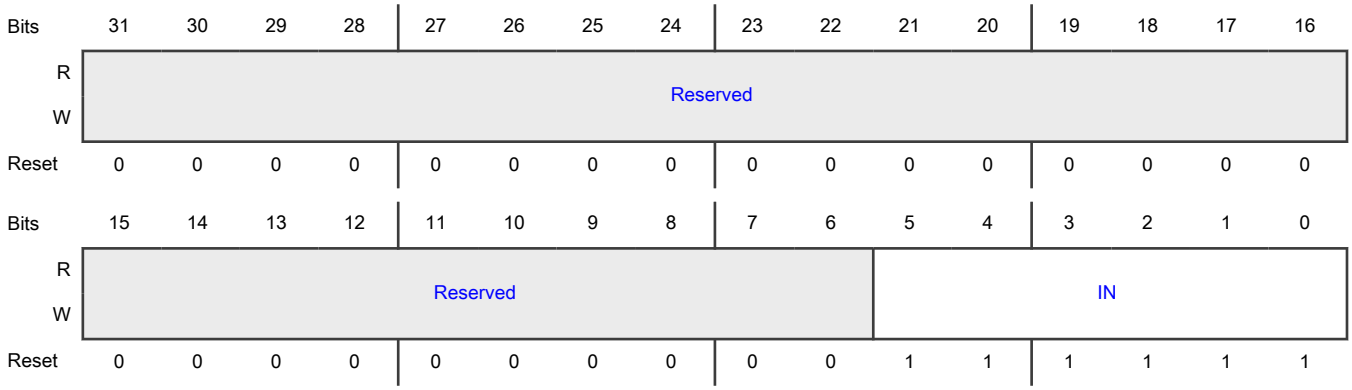
14.5.1.1.45 AOI0 trigger inputs (AOI0_IN0 - AOI0_IN15)

Offset

For m = 0 to 15:

Register	Offset
AOI0_INm	440h + (m × 4h)

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 IN	Input trigger assignments 00_0000 - PIN_INT0 00_0001 - PIN_INT1 00_0010 - SCT_OUT0 00_0011 - SCT_OUT1 00_0100 - SCT_OUT2 00_0101 - SCT_OUT3 00_0110 - T0_MAT3 00_0111 - T1_MAT3

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00_1000 - T2_MAT3 00_1001 - T2_MAT2 00_1010 - T3_MAT2 00_1011 - T4_MAT2 00_1100 - ACMP0_OUT 00_1101 - GPIOINT_BMATCH 00_1110 - ADC0_IRQ 00_1111 - ADC1_IRQ 01_0000 - ADC0_tcomp[0] 01_0001 - ADC0_tcomp[1] 01_0010 - ADC0_tcomp[2] 01_0011 - ADC0_tcomp[3] 01_0100 - ADC1_tcomp[0] 01_0101 - ADC1_tcomp[1] 01_0110 - ADC1_tcomp[2] 01_0111 - ADC1_tcomp[3] 01_1000 - HSCMP0_OUT 01_1001 - HSCMP1_OUT 01_1010 - HSCMP2_OUT 01_1011 - PWM0_SM0_MUX_TRIG0 01_1100 - PWM0_SM0_MUX_TRIG1 01_1101 - PWM0_SM1_MUX_TRIG0 01_1110 - PWM0_SM1_MUX_TRIG1 01_1111 - PWM0_SM2_MUX_TRIG0 10_0000 - PWM0_SM2_MUX_TRIG1 10_0001 - PWM0_SM3_MUX_TRIG0 10_0010 - PWM0_SM3_MUX_TRIG1 10_0011 - PWM1_SM0_MUX_TRIG0 10_0100 - PWM1_SM0_MUX_TRIG1 10_0101 - PWM1_SM1_MUX_TRIG0 10_0110 - PWM1_SM1_MUX_TRIG1 10_0111 - PWM1_SM2_MUX_TRIG0 10_1000 - PWM1_SM2_MUX_TRIG1 10_1001 - PWM1_SM3_MUX_TRIG0

Table continues on the next page...

Table continued from the previous page...

Field	Description
	10_1010 - PWM1_SM3_MUX_TRIG1 10_1011 - ENC0_CMP/POS_MATCH 10_1100 - ENC1_CMP/POS_MATCH 10_1101 - EXTTRIG_IN0 10_1110 - EXTTRIG_IN1 10_1111 - EXTTRIG_IN2 11_0000 - EXTTRIG_IN3 11_0001 - Reserved 11_0010 - Reserved 11_0011 - DMA0_TRIGOUT0 11_0100 - DMA0_TRIGOUT1 11_0101 - DMA0_TRIGOUT2 11_0110 - DMA0_TRIGOUT3 11_0111 - DMA0_TRIGOUT4 11_1000 - DMA0_TRIGOUT5 11_1001 - DMA0_TRIGOUT6 11_1010 - DMA1_TRIGOUT0 11_1011 - DMA1_TRIGOUT1 11_1100 - DMA1_TRIGOUT2

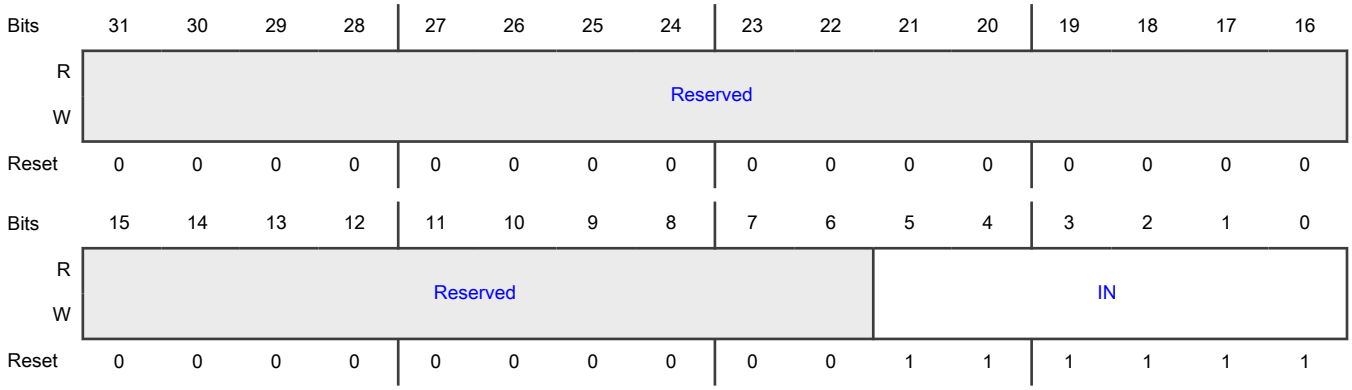
14.5.1.1.46 AOI1 trigger inputs (AOI1_IN0 - AOI1_IN15)

Offset

For m = 0 to 15:

Register	Offset
AOI1_INm	480h + (m × 4h)

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 IN	Input trigger assignments 00_0000 - PIN_INT0 00_0001 - PIN_INT1 00_0010 - SCT_OUT0 00_0011 - SCT_OUT1 00_0100 - SCT_OUT2 00_0101 - SCT_OUT3 00_0110 - T0_MAT3 00_0111 - T1_MAT3 00_1000 - T2_MAT3 00_1001 - T2_MAT2 00_1010 - T3_MAT2 00_1011 - T4_MAT2 00_1100 - ACMP0_OUT 00_1101 - GPIOINT_BMATCH 00_1110 - ADC0_IRQ 00_1111 - ADC1_IRQ 01_0000 - ADC0_tcomp[0] 01_0001 - ADC0_tcomp[1] 01_0010 - ADC0_tcomp[2] 01_0011 - ADC0_tcomp[3]

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01_0100 - ADC1_tcomp[0] 01_0101 - ADC1_tcomp[1] 01_0110 - ADC1_tcomp[2] 01_0111 - ADC1_tcomp[3] 01_1000 - HSCMP0_OUT 01_1001 - HSCMP1_OUT 01_1010 - HSCMP2_OUT 01_1011 - PWM0_SM0_MUX_TRIG0 01_1100 - PWM0_SM0_MUX_TRIG1 01_1101 - PWM0_SM1_MUX_TRIG0 01_1110 - PWM0_SM1_MUX_TRIG1 01_1111 - PWM0_SM2_MUX_TRIG0 10_0000 - PWM0_SM2_MUX_TRIG1 10_0001 - PWM0_SM3_MUX_TRIG0 10_0010 - PWM0_SM3_MUX_TRIG1 10_0011 - PWM1_SM0_MUX_TRIG0 10_0100 - PWM1_SM0_MUX_TRIG1 10_0101 - PWM1_SM1_MUX_TRIG0 10_0110 - PWM1_SM1_MUX_TRIG1 10_0111 - PWM1_SM2_MUX_TRIG0 10_1000 - PWM1_SM2_MUX_TRIG1 10_1001 - PWM1_SM3_MUX_TRIG0 10_1010 - PWM1_SM3_MUX_TRIG1 10_1011 - ENC0_CMP/POS_MATCH 10_1100 - ENC1_CMP/POS_MATCH 10_1101 - EXTTRIG_IN0 10_1110 - EXTTRIG_IN1 10_1111 - EXTTRIG_IN2 11_0000 - EXTTRIG_IN3 11_0001 - Reserved 11_0010 - Reserved 11_0011 - DMA0_TRIGOUT0 11_0100 - DMA0_TRIGOUT1 11_0101 - DMA0_TRIGOUT2

Table continues on the next page...

Table continued from the previous page...

Field	Description
	11_0110 - DMA0_TRIGOUT3 11_0111 - DMA0_TRIGOUT4 11_1000 - DMA0_TRIGOUT5 11_1001 - DMA0_TRIGOUT6 11_1010 - DMA1_TRIGOUT0 11_1011 - DMA1_TRIGOUT1 11_1100 - DMA1_TRIGOUT2

14.5.1.1.47 AOI External Trigger Inputs (AOI_EXT_TRIG0 - AOI_EXT_TRIG7)

Offset

For a = 0 to 7:

Register	Offset
AOI_EXT_TRIGa	4C0h + (a × 4h)

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	Reserved												TRIGIN			
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Fields

Field	Description
31-5 —	Reserved
4-0 TRIGIN	AOI external trigger inputs from 0 to 4. 0_0000 - PIN_INT0 0_0001 - PIN_INT1

Table continues on the next page...

Table continued from the previous page...

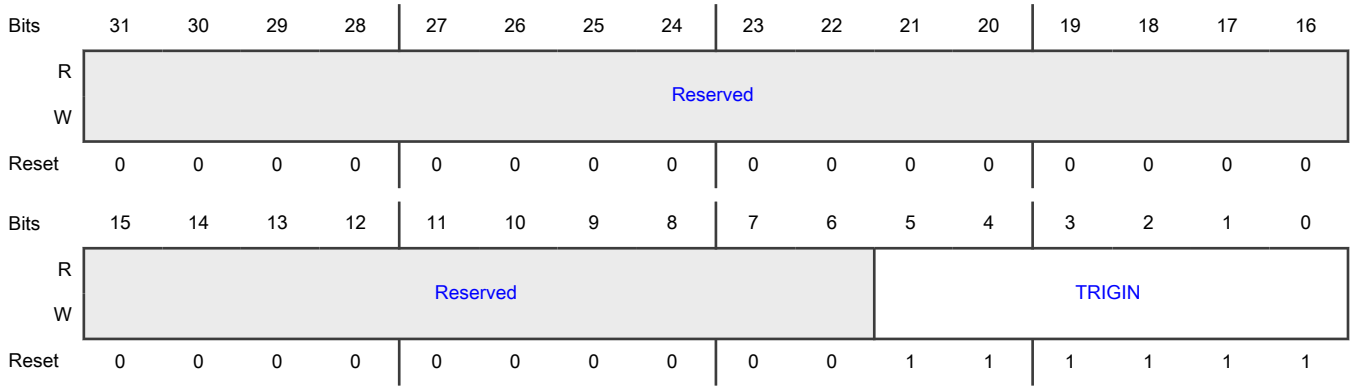
Field	Description
	0_0010 - ADC0_IRQ 0_0011 - ADC1_IRQ 0_0100 - ADC0_tcomp[0] 0_0101 - ADC1_tcomp[0] 0_0110 - PWM0_SM0_MUX_TRIG0 | PWM0_SM0_MUX_TRIG1 0_0111 - PWM0_SM1_MUX_TRIG0 | PWM0_SM1_MUX_TRIG1 0_1000 - PWM0_SM2_MUX_TRIG0 | PWM0_SM2_MUX_TRIG1 0_1001 - PWM0_SM3_MUX_TRIG0 | PWM0_SM3_MUX_TRIG1 0_1010 - PWM1_SM0_MUX_TRIG0 | PWM1_SM0_MUX_TRIG1 0_1011 - PWM1_SM1_MUX_TRIG0 | PWM1_SM1_MUX_TRIG1 0_1100 - PWM1_SM2_MUX_TRIG0 | PWM1_SM2_MUX_TRIG1 0_1101 - PWM1_SM3_MUX_TRIG0 | PWM1_SM3_MUX_TRIG1 0_1110 - ENC0_CMP/POS_MATCH 0_1111 - ENC1_CMP/POS_MATCH 1_0000 - AOI0_OUT0 1_0001 - AOI0_OUT1 1_0010 - AOI0_OUT2 1_0011 - AOI0_OUT3 1_0100 - AOI1_OUT0 1_0101 - AOI1_OUT1 1_0110 - AOI1_OUT2 1_0111 - AOI1_OUT3 1_1000 - TMPR_OUT 1_1001-1_1111 - None

14.5.1.1.48 Input connections for HSCMP1 (HSCMP1_TRIG)

Offset

Register	Offset
HSCMP1_TRIG	4E0h

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 TRIGIN	CMP1 input trigger 00_0000 - PIN_INT0 00_0001 - PIN_INT7 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT7 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T0_MAT1 00_1001 - T4_MAT1 00_1010 - Reserved 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[1] 00_1110 - ADC1_tcomp[1] 00_1111 - Reserved 01_0000 - Reserved 01_0001 - PWM0_SM0_MUX_TRIG0 | PWM0_SM0_MUX_TRIG1 01_0010 - PWM0_SM1_MUX_TRIG0 | PWM0_SM1_MUX_TRIG1 01_0011 - PWM0_SM2_MUX_TRIG0 | PWM0_SM2_MUX_TRIG1

Table continues on the next page...

Table continued from the previous page...

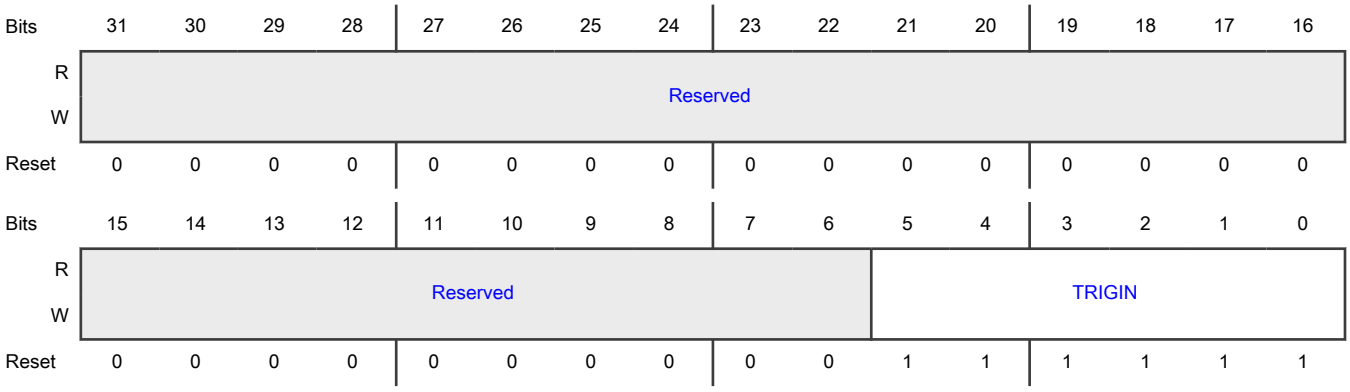
Field	Description
	01_0100 - PWM0_SM3_MUX_TRIG0 | PWM0_SM3_MUX_TRIG1 01_0101 - PWM1_SM0_MUX_TRIG0 | PWM1_SM0_MUX_TRIG1 01_0110 - PWM1_SM1_MUX_TRIG0 | PWM1_SM1_MUX_TRIG1 01_0111 - PWM1_SM2_MUX_TRIG0 | PWM1_SM2_MUX_TRIG1 01_1000 - PWM1_SM3_MUX_TRIG0 | PWM1_SM3_MUX_TRIG1 01_1001 - ENC0_CMP/POS_MATCH 01_1010 - ENC1_CMP/POS_MATCH 01_1011 - AOI0_OUT0 01_1100 - AOI0_OUT1 01_1101 - AOI0_OUT2 01_1110 - AOI0_OUT3 01_1111 - AOI1_OUT0 10_0000 - AOI1_OUT1 10_0001 - AOI1_OUT2 10_0010 - AOI1_OUT3 10_0011 - DMA0_TRIGOUT0 10_0100 - DMA0_TRIGOUT1 10_0101 - DMA0_TRIGOUT2

14.5.1.1.49 Input connections for HSCMP2 (HSCMP2_TRIG)

Offset

Register	Offset
HSCMP2_TRIG	500h

Diagram



Fields

Field	Description
31-6 —	Reserved
5-0 TRIGIN	CMP2 input trigger 00_0000 - PIN_INT0 00_0001 - PIN_INT4 00_0010 - SCT_OUT4 00_0011 - SCT_OUT5 00_0100 - SCT_OUT8 00_0101 - T0_MAT3 00_0110 - T1_MAT3 00_0111 - T2_MAT3 00_1000 - T0_MAT2 00_1001 - T4_MAT2 00_1010 - Reserved 00_1011 - ARM_TXEV 00_1100 - GPIOINT_BMATCH 00_1101 - ADC0_tcomp[2] 00_1110 - ADC1_tcomp[2] 00_1111 - Reserved 01_0000 - Reserved 01_0001 - PWM0_SM0_MUX_TRIG0 | PWM0_SM0_MUX_TRIG1 01_0010 - PWM0_SM1_MUX_TRIG0 | PWM0_SM1_MUX_TRIG1 01_0011 - PWM0_SM2_MUX_TRIG0 | PWM0_SM2_MUX_TRIG1 01_0100 - PWM0_SM3_MUX_TRIG0 | PWM0_SM3_MUX_TRIG1 01_0101 - PWM1_SM0_MUX_TRIG0 | PWM1_SM0_MUX_TRIG1 01_0110 - PWM1_SM1_MUX_TRIG0 | PWM1_SM1_MUX_TRIG1 01_0111 - PWM1_SM2_MUX_TRIG0 | PWM1_SM2_MUX_TRIG1 01_1000 - PWM1_SM3_MUX_TRIG0 | PWM1_SM3_MUX_TRIG1 01_1001 - ENC0_CMP/POS_MATCH 01_1010 - ENC1_CMP/POS_MATCH 01_1011 - AOI0_OUT0 01_1100 - AOI0_OUT1 01_1101 - AOI0_OUT2

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01_1110 - AOI0_OUT3 01_1111 - AOI1_OUT0 10_0000 - AOI1_OUT1 10_0001 - AOI1_OUT2 10_0010 - AOI1_OUT3 10_0011 - DMA0_TRIGOUT0 10_0100 - DMA0_TRIGOUT1 10_0101 - DMA0_TRIGOUT2

14.5.1.1.50 Trigger select for DMA0 channel (DMA0_ITRIG_INMUX32 - DMA0_ITRIG_INMUX51)

With the DMA trigger input multiplexing registers, one trigger input can be selected for each of the DMA channels from the potential internal sources. By default, none of the triggers are selected.

Offset

For a = 32 to 51:

Register	Offset
DMA0_ITRIG_INMUXa	4A0h + (a × 4h)

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	Reserved										INP					
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

Fields

Field	Description
31-6 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
5-0 INP	Trigger input number (binary value) for DMA channel n (n = 32 to 51). 00_0000 - FlexSPI_RX 00_0001 - FlexSPI_TX 00_0010 - GPIO_INT0 00_0011 - GPIO_INT1 00_0100 - GPIO_INT2 00_0101 - GPIO_INT3 00_0110 - T0_DMAREQ_M0 00_0111 - T0_DMAREQ_M1 00_1000 - T1_DMAREQ_M0 00_1001 - T1_DMAREQ_M1 00_1010 - T2_DMAREQ_M0 00_1011 - T2_DMAREQ_M1 00_1100 - T3_DMAREQ_M0 00_1101 - T3_DMAREQ_M1 00_1110 - T4_DMAREQ_M0 00_1111 - T4_DMAREQ_M1 01_0000 - ACMP0_OUT 01_0001 - SDMA0_TRIGOUT_A 01_0010 - SDMA0_TRIGOUT_B 01_0011 - SDMA0_TRIGOUT_C 01_0100 - SDMA0_TRIGOUT_D 01_0101 - SCT_DMA0 01_0110 - SCT_DMA1 01_0111 - ADC0_tcomp[0] 01_1000 - ADC1_tcomp[0] 01_1001 - HSCMP0 01_1010 - HSCMP1 01_1011 - HSCMP2 01_1100 - AOI0_OUT0 01_1101 - AOI0_OUT1 01_1110 - AOI0_OUT2 01_1111 - AOI0_OUT3 10_0000 - AOI1_OUT0

Table continues on the next page...

Table continued from the previous page...

Field	Description
	10_0001 - AOI1_OUT1 10_0010 - AOI1_OUT2 10_0011 - AOI1_OUT3 10_0100 - FlexPWM0_req_capt0 10_0101 - FlexPWM0_req_capt1 10_0110 - FlexPWM0_req_capt2 10_0111 - FlexPWM0_req_capt3 10_1000 - FlexPWM0_req_val0 10_1001 - FlexPWM0_req_val1 10_1010 - FlexPWM0_req_val2 10_1011 - FlexPWM0_req_val3 10_1100 - FlexPWM1_req_capt0 10_1101 - FlexPWM1_req_capt1 10_1110 - FlexPWM1_req_capt2 10_1111 - FlexPWM1_req_capt3 11_0000 - FlexPWM1_req_val0 11_0001 - FlexPWM1_req_val1 11_0010 - FlexPWM1_req_val2 11_0011 - FlexPWM1_req_val3 11_0100 - TMPR_OUT 11_0110-11_1111 - Reserved

14.5.1.1.51 Enable DMA0 requests (DMA0_REQEN0)

Inputs to DMA0 are enabled by the DMA0 request enable register. For each bit in this register, a 0 means that DMA request input is disabled and 1 means that DMA request input is enabled.

Offset

Register	Offset
DMA0_REQEN0	740h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...
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 W	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA9	REQ_ ENA8	REQ_ ENA7	REQ_ ENA6	REQ_ ENA5	REQ_ ENA4	REQ_ ENA3	REQ_ ENA2	REQ_ ENA1	REQ_ ENA0
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Description
31-0 REQ_ENAn	Controls the first 32 request inputs of DMA0. If bit i is '1' the DMA request input #i is enabled.

14.5.1.1.52 Enable DMA0 requests (DMA0_REQEN1)

Inputs to DMA0 are enabled by the DMA0 request enable register. For each bit in this register, a 0 means that DMA request input is disabled and 1 means that DMA request input is enabled.

Offset

Register	Offset
DMA0_REQEN1	744h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved												REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...
Reset	u	u	u	u	u	u	u	u	u	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 W	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA...	REQ_ ENA9	REQ_ ENA8	REQ_ ENA7	REQ_ ENA6	REQ_ ENA5	REQ_ ENA4	REQ_ ENA3	REQ_ ENA2	REQ_ ENA1	REQ_ ENA0
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Description
31-20 —	Reserved
19-0 REQ_ENAn	Controls the remaining 20 request inputs of DMA0. If bit i is '1' the DMA request input #(i+32) is enabled.

14.5.1.1.53 Set bits in DMA0_REQEN0 register (DMA0_REQEN0_SET)

Writing a 1 to a bit position in DMA0_REQEN0_SET, sets the corresponding position in DMA0_REQEN0. This is a write-only register.

Offset

Register	Offset
DMA0_REQEN0_SET	748h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	SET31	SET30	SET29	SET28	SET27	SET26	SET25	SET24	SET23	SET22	SET21	SET20	SET19	SET18	SET17	SET16
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																
W	SET15	SET14	SET13	SET12	SET11	SET10	SET9	SET8	SET7	SET6	SET5	SET4	SET3	SET2	SET1	SET0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-0 SETn	Write : If bit #i = 1, bit #i in DMA0_REQEN0 register is set to 1; if bit #i = 0, no change in DMA0_REQEN0 register.

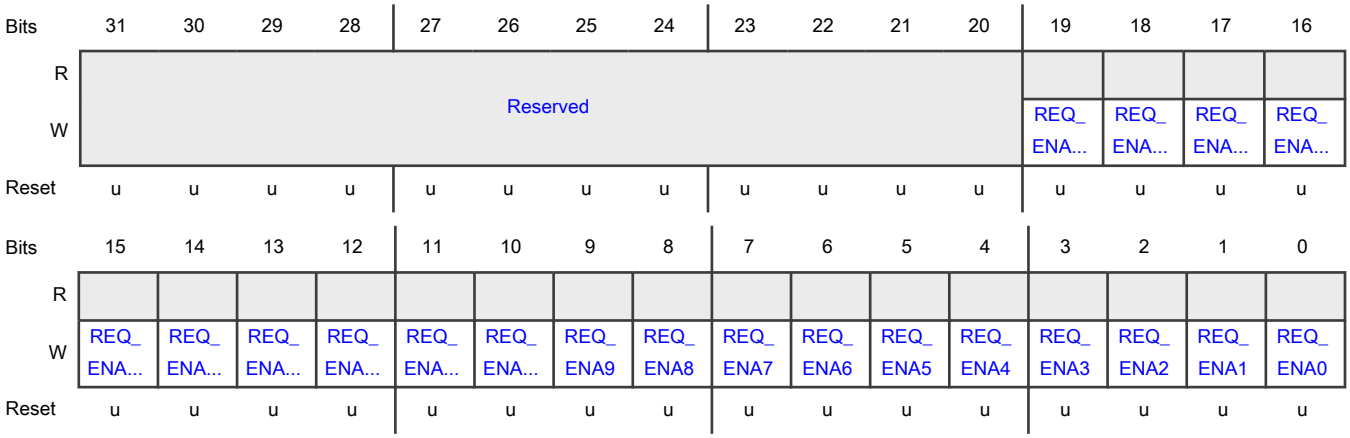
14.5.1.1.54 Set bits in DMA0_REQEN1 register (DMA0_REQEN1_SET)

Writing a 1 to a bit position in DMA0_REQEN1_SET, sets the corresponding position in DMA0_REQEN1.

Offset

Register	Offset
DMA0_REQEN1_SET	74Ch

Diagram



Fields

Field	Description
31-20 —	Reserved
19-0 REQ_ENAn	Write : If bit #i = 1, bit #i in DMA0_REQEN1 register is set to 1; if bit #i = 0, no change in DMA0_REQEN1 register.

14.5.1.1.55 Clear bits in DMA0_REQEN0 register (DMA0_REQEN0_CLR)

Writing a 1 to a bit position in DMA0_REQ_ENA_CLR, clears the corresponding position in DMA0_REQ_ENA. This is a write-only register.

Offset

Register	Offset
DMA0_REQEN0_CLR	750h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	CLR31	CLR30	CLR29	CLR28	CLR27	CLR26	CLR25	CLR24	CLR23	CLR22	CLR21	CLR20	CLR19	CLR18	CLR17	CLR16
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																
W	CLR15	CLR14	CLR13	CLR12	CLR11	CLR10	CLR9	CLR8	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-0 CLRn	Write : If bit #i = 1, bit #i in DMA0_REQEN0 register is reset to 0; if bit #i = 0 , no change in DMA0_REQEN0 register.

14.5.1.1.56 Clear bits in DMA0_REQEN1 register (DMA0_REQEN1_CLR)

Writing a 1 to a bit position in DMA0_REQEN1_CLR, clears the corresponding position in DMA0_REQENA.

Offset

Register	Offset
DMA0_REQEN1_CLR	754h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W													REQ_ENA...	REQ_ENA...	REQ_ENA...	REQ_ENA...
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																
W	REQ_ENA...	REQ_ENA...	REQ_ENA...	REQ_ENA...	REQ_ENA...	REQ_ENA...	REQ_ENA9	REQ_ENA8	REQ_ENA7	REQ_ENA6	REQ_ENA5	REQ_ENA4	REQ_ENA3	REQ_ENA2	REQ_ENA1	REQ_ENA0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-20 —	Reserved
19-0 REQ_ENAn	Write : If bit #i = 1, bit #i in DMA0_REQEN1 register is reset to 0; if bit #i = 0 , no change in DMA0_REQEN1 register.

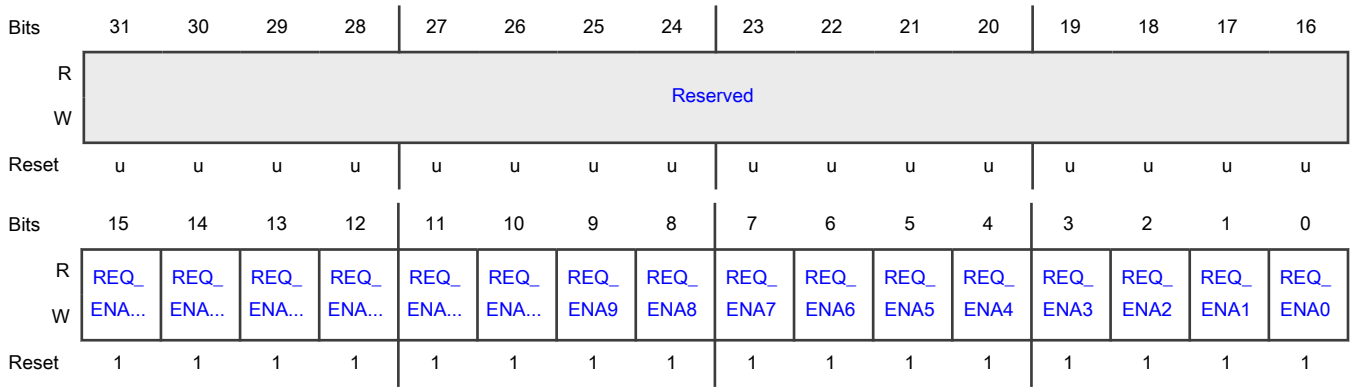
14.5.1.1.57 Enable DMA1 requests (DMA1_REQEN)

Inputs to DMA1 are enabled by the DMA1 request enable register. For each bit in this register, a 0 means that DMA request input is disabled and 1 means that DMA request input is enabled.

Offset

Register	Offset
DMA1_REQEN	760h

Diagram



Fields

Field	Description
31-16 —	Reserved
15-0 REQ_ENAn	Controls the 16 request inputs of DMA1. If bit i is '1' the DMA request input #i is enabled.

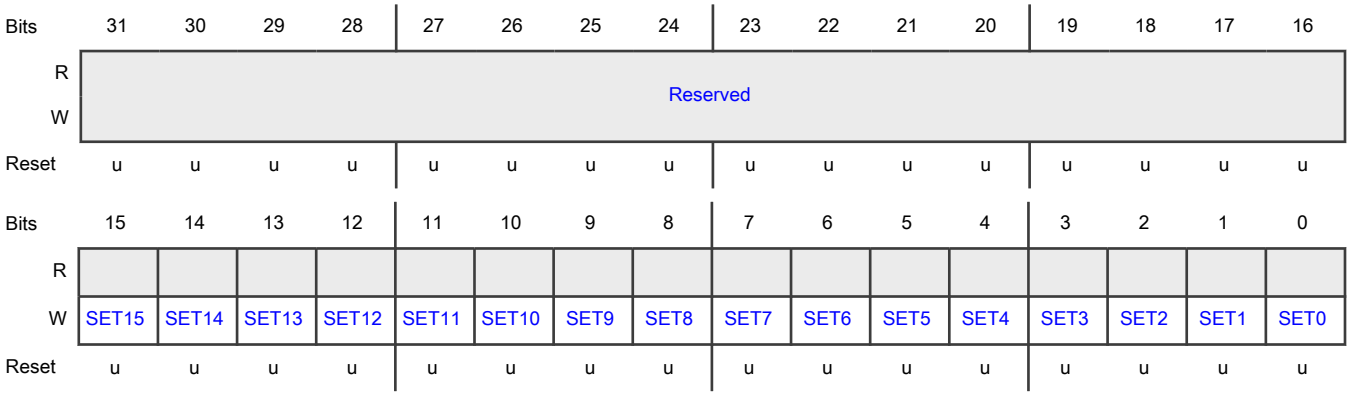
14.5.1.1.58 Set bits in DMA1_REQEN register (DMA1_REQEN_SET)

Writing a 1 to a bit position in DMA1_REQEN_SET, sets the corresponding position in DMA1_REQEN. This is a write-only register.

Offset

Register	Offset
DMA1_REQEN_SET	768h

Diagram



Fields

Field	Description
31-16 —	Reserved
15-0 SETn	Write : If bit #i = 1, bit #i in DMA1_REQEN register is set to 1; if bit #i = 0 , no change in DMA1_REQEN register

14.5.1.1.59 Clear bits in DMA1_REQEN register (DMA1_REQEN_CLR)

Writing a 1 to a bit position in DMA1_REQEN_CLR, clears the corresponding position in DMA1_REQEN. This is a write-only register.

Offset

Register	Offset
DMA1_REQEN_CLR	770h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
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																
W	CLR15	CLR14	CLR13	CLR12	CLR11	CLR10	CLR9	CLR8	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-16 —	Reserved
15-0 CLRn	Write : If bit #i = 1, bit #i in DMA1_REQEN register is reset to 0; if bit #i = 0 , no change in DMA1_REQEN register

14.5.1.1.60 Enable DMA0 triggers (DMA0_ITRIGEN0)

DMA triggers to each of the two DMA controllers are enabled separately so that the same trigger can be routed to only one DMA controller. Inputs to DMA0 are enabled by this register.

Offset

Register	Offset
DMA0_ITRIGEN0	780h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...
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	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN9	ITRIG EN8	ITRIG EN7	ITRIG EN6	ITRIG EN5	ITRIG EN4	ITRIG EN3	ITRIG EN2	ITRIG EN1	ITRIG EN0
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Description
31-0 ITRIGENn	Controls the 32 trigger inputs of DMA0. If bit i is '1' the DMA trigger input #i is enabled.

14.5.1.1.61 Enable DMA0 triggers (DMA0_ITRIGEN1)

DMA triggers to each of the two DMA controllers are enabled separately so that the same trigger can be routed to only one DMA controller. Inputs to DMA0 are enabled by this register.

Offset

Register	Offset
DMA0_ITRIGEN1	784h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved												ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...
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 W	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN9	ITRIG EN8	ITRIG EN7	ITRIG EN6	ITRIG EN5	ITRIG EN4	ITRIG EN3	ITRIG EN2	ITRIG EN1	ITRIG EN0
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Description
31-20 —	Reserved
19-0 ITRIGENn	Controls the remaining 20 trigger inputs of DMA0. If bit n is '1' the DMA trigger input #(n+32) is enabled.

14.5.1.1.62 Set bits in DMA0_ITRIGEN0 register (DMA0_ITRIGEN0_SET)

The DMA0 input trigger enable set register allows setting any combination of bits in the DMA0_ITRIGEN0 register.

Offset

Register	Offset
DMA0_ITRIGEN0_SET	788h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	SET31	SET30	SET29	SET28	SET27	SET26	SET25	SET24	SET23	SET22	SET21	SET20	SET19	SET18	SET17	SET16
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																
W	SET15	SET14	SET13	SET12	SET11	SET10	SET9	SET8	SET7	SET6	SET5	SET4	SET3	SET2	SET1	SET0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-0 SETn	Write : If bit #i = 1, bit #i in DMA0_ITRIGEN0 register is set to 1; if bit #i = 0 , no change in DMA0_ITRIGEN0 register.

14.5.1.1.63 Set bits in DMA0_ITRIGEN1 register (DMA0_ITRIGEN1_SET)

The DMA0 input trigger enable set register allows setting any combination of bits in the DMA0_ITRIGEN1 register.

Offset

Register	Offset
DMA0_ITRIGEN1_SET	78Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W													SET19	SET18	SET17	SET16
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																
W	SET15	SET14	SET13	SET12	SET11	SET10	SET9	SET8	SET7	SET6	SET5	SET4	SET3	SET2	SET1	SET0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-20 —	Reserved
19-0 SETn	Write : If bit #i = 1, bit #i in DMA0_ITRIGEN1 register is set to 1; if bit #i = 0 , no change in DMA0_ITRIGEN1 register.

14.5.1.1.64 Clear bits in DMA0_ITRIGEN0 register (DMA0_ITRIGEN0_CLR)

The DMA0 input trigger enable clear register allow clearing any combination of bits in the DMA0_ITRIGEN0 register.

Offset

Register	Offset
DMA0_ITRIGEN0_CLR	790h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	CLR31	CLR30	CLR29	CLR28	CLR27	CLR26	CLR25	CLR24	CLR23	CLR22	CLR21	CLR20	CLR19	CLR18	CLR17	CLR16
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																
W	CLR15	CLR14	CLR13	CLR12	CLR11	CLR10	CLR9	CLR8	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-0 CLRn	Write : If bit #i = 1, bit #i in DMA0_ITRIGEN0 register is reset to 0; if bit #i = 0 , no change in DMA0_ITRIGEN0 register.

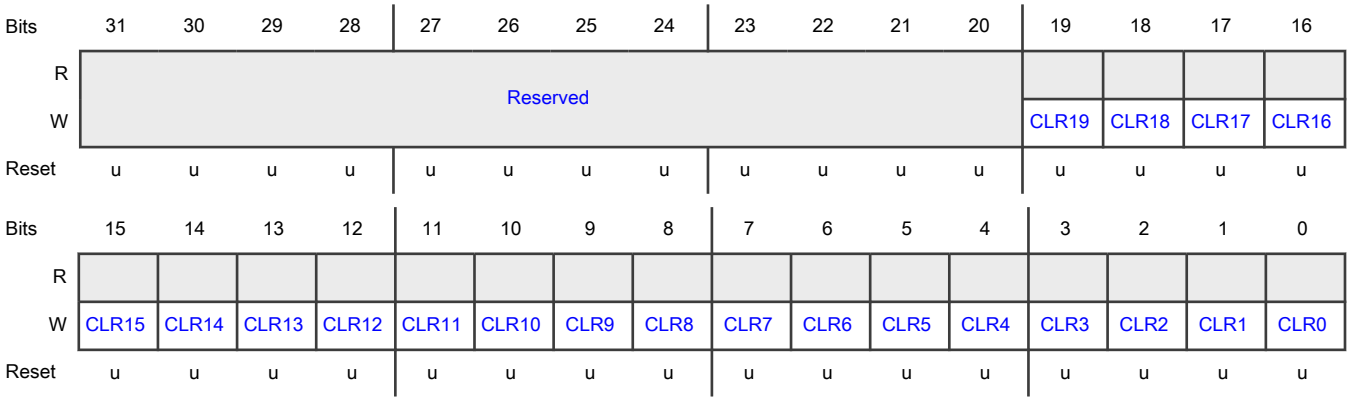
14.5.1.1.65 Clear bits in DMA0_ITRIGEN1 register (DMA0_ITRIGEN1_CLR)

The DMA0 input trigger enable clear register allow clearing any combination of bits in the DMA0_ITRIGEN1 register.

Offset

Register	Offset
DMA0_ITRIGEN1_CLR	794h

Diagram



Fields

Field	Description
31-20 —	Reserved
19-0 CLRn	Write : If bit #i = 1, bit #i in DMA0_ITRIGEN1 register is reset to 0; if bit #i = 0 , no change in DMA0_ITRIGEN1 register.

14.5.1.1.66 Enable DMA1 triggers (DMA1_ITRIGEN)

DMA triggers to each of the two DMA controllers are enabled separately so that the same trigger can be routed to only one DMA controller. Inputs to DMA1 are enabled by this register.

Offset

Register	Offset
DMA1_ITRIGEN	7A0h

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	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN...	ITRIG EN9	ITRIG EN8	ITRIG EN7	ITRIG EN6	ITRIG EN5	ITRIG EN4	ITRIG EN3	ITRIG EN2	ITRIG EN1	ITRIG EN0
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Description
31-16 —	Reserved
15-0 ITRIGENn	Controls the 16 trigger inputs of DMA1. If bit i is '1' the DMA trigger input #i is enabled.

14.5.1.1.67 Set bits in DMA1_ITRIGEN register (DMA1_ITRIGEN_SET)

The DMA1 input trigger enable set register allow setting any combination of bits in the DMA1_ITRIGEN register.

Offset

Register	Offset
DMA1_ITRIGEN_SET	7A8h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
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																
W	SET15	SET14	SET13	SET12	SET11	SET10	SET9	SET8	SET7	SET6	SET5	SET4	SET3	SET2	SET1	SET0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-16 —	Reserved
15-0 SETn	Write : If bit #i = 1, bit #i in DMA1_ITRIGEN register is set to 1; if bit #i = 0 , no change in DMA1_ITRIGEN register

14.5.1.1.68 Clear bits in DMA1_ITRIGEN register (DMA1_ITRIGEN_CLR)

The DMA1 input trigger enable clear register allow clearing any combination of bits in the DMA1_ITRIGEN register.

Offset

Register	Offset
DMA1_ITRIGEN_CLR	7B0h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
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																
W	CLR15	CLR14	CLR13	CLR12	CLR11	CLR10	CLR9	CLR8	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-16 —	Reserved
15-0 CLRn	Write : If bit #i = 1, bit #i in DMA1_ITRIGEN register is reset to 0; if bit #i = 0 , no change in DMA1_ITRIGEN register

Chapter 15
Flash Controller

15.1 Chip-specific FLASH information

Table 41. Reference links to related information

Topic	Related module	Reference
Full description	FLASH	FLASH
System memory map		System memory map
Clocking		Clock generation
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

15.1.1 Module instances

This device has only one instance of FLASH.

15.2 Overview

This section provides an introduction to the Flash Controller module.

15.2.1 Features

- Includes analog delay block to manage self-timed read operations.
- LPCAC with 8K READ Cache SRAM support with parity enabled.
- Read port designed as an interface to the FMC flash cache.
- APB registers interface (separate clock domain with respect to the read port).
- Auto initialization after reset.
- ECC management, including single bit correction and error correction logging.

15.2.2 Block diagram

Figure 28 shows a functional block diagram of the controller. Some connections between blocks are not presented for clarity. The actual design hierarchy does not correspond to this diagram; the controller top level instantiates the hard blocks (the flash and the analog delay block), and a block that contains all logic. The logic block is subdivided into a sub-block for each of the different clock domains, and an additional block that manages all clocks and resets.

The architecture is built around a sequencer, which transforms complex user and test commands into a sequence of basic memory operations. The sequencer implements a number of commands, for example, to change the content of the memory, check its content, and change the mode of operation.

NOTE

When performing AHB reads of the flash memory contents, a hardware fault occurs if an unrecoverable error is detected. Read operations performed using flash controller commands (see: Command listing (CMD)) will not cause a hardware fault.

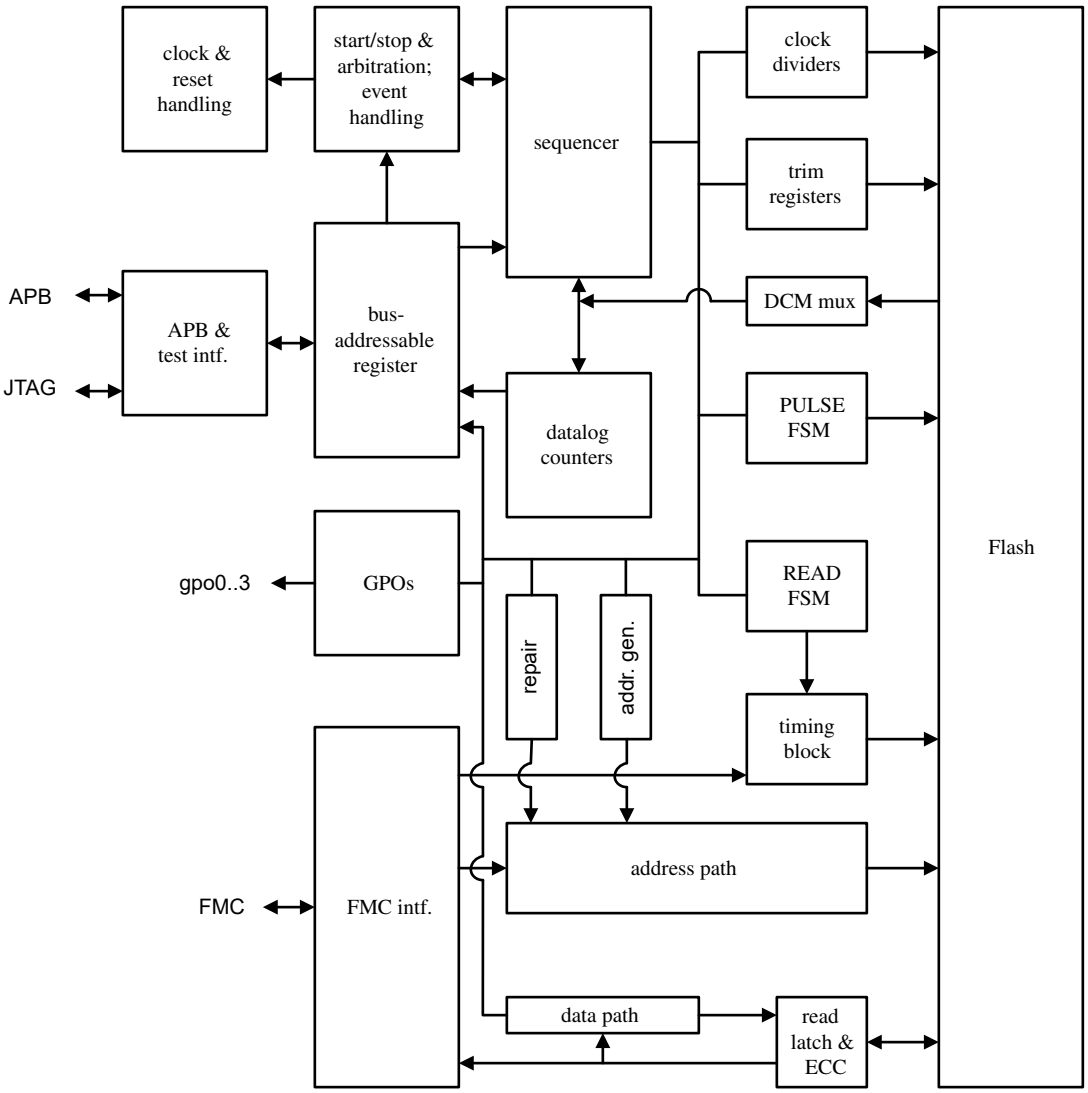


Figure 28. Block diagram

15.3 Functional description

This chapter contains the following information:

Throughout the chapter, [pseudo] code examples are also given. In these examples, it is assumed that register names are accessible through variables with the same name. Syntax is pseudo-C language.

- Detailed specification of the behavior of the controller (with the exception of commands, which are described in Command listing (CMD) .
- Constraints that must be followed while using the controller If these constraints are not met, the controller and/or the associated memory will not behave as specified.
- Instructions for the use of the controller (including usage examples), explanation of the rationale behind the architectural choices, caveats and warnings.

15.3.1 Basic principles of operation

This section lists information which is common to multiple controller functions.

15.3.1.1 Definitions

The memory managed by the controller can execute the following basic operations:

- Reading: it is the process of extracting the information contained at a specific memory location
- Writing: it is the process of updating temporary storage present in the memory, called *page register*, with data that must subsequently be programmed
- Program/erase: it is the process by which the memory will alter its nonvolatile content, by either clearing all selected bits to a default value (erase), or setting them to the value specified by the page register (program).
- Power down: the memory is put in a mode where a minimum amount of supply current is used; no (other) operation can be performed in this state
- When none of these operations is being performed, the memory is said to be idle.

In general, read and write operations on the memory are de-coupled by read and write requests to the controller: a single controller command can perform multiple memory operations.

In any case, no flash operation is initiated without a triggering event (e.g. a write to the CMD register, activation of the memory read, or a reset).

15.3.2 Initialization

When entering the reset mode (hard reset, or "1" written into the RST bit of the EVENT register), all controller registers will be initialized to the value specified in the relative register description. Any command or bus transaction in progress is interrupted as well, with no regards for data integrity.

Immediately after leaving reset mode, an initialization phase takes place, where some memory locations are read, and corresponding volatile locations are initialized depending on the value just read.

The controller reads 19 locations in the last two pages of the flash for the exact locations and their content). For each location read, it initializes the corresponding volatile storage in the controller: flash trim values, flash repair info, gpo trim bus.

If an un-correctable ECC error is detected, the corresponding volatile storage is not updated (so that the safe default values are kept), and the initialization is immediately terminated with the FAIL flag set.

A per-page checksum protects the integrity of information read by the Initialization command. An additional word is programmed with a value, such that the checksum of the words read (including the additional word) is 0. The checksum algorithm is the same as the one used by the *checksum page range* command.

If initialization reports a FAIL, the flash was not correctly configured, and must not be used (read data may be incorrect, and writing may corrupt the content). Security-conscious users should ensure that an application is not started with a failing init.

Although the controller will not ensure that reading is performed correctly and with the correct mode in case of an init error, reading is anyway permitted, to avoid ending up with inaccessible samples in the case of initialization issues.

15.3.3 Configuration

Controller configuration amounts to specifying options such as read speed, and caching/pre-fetching options in a way that best matches system operation. Configuration is normally performed by system software shortly after initialization, although default configuration values are normally chosen to allow safe operation with no further software intervention. When conditions change (for example, the system clock frequency is changed), configuration can be repeated.

Be aware that configuration errors may prevent correct working of the flash.

This is an example of code to perform configuration just after exiting from reset. It assumes execution from ROM by default, with comments specifying the differences in the case of booting from flash.

```
//check init status. Not needed if booting from flash: in that case, the safest
//option is to prevent fetching from flash if pin init_error=1.
while(!(INT_STATUS & 0x4 )) ; //wait until DONE is set. Not needed if CPU reset
```

```
//is only released when ctl_busy=0
if(INT_STATUS & 0x1) {
handle_boot_error();         //communicate to the external world that a
//non-recoverable error occurred.
while(1);         //handle_boot_error should not return. In any case, the flash
//cannot be used.
}
//end of init status checking
//begin interrupt configuration
INT_CLR_ENABLE = 0x1f;//clear all interrupt enables. Not needed just after
//reset, as this is the default state
INT_SET_ENABLE = 0x2; //only enable interrupt on ERR status.
//Correct code would never set this flag.
//Correctable ECC events can be managed by periodic checks.
//Most examples in this manual will poll the DONE bit
//and explicitly check for FAIL status, so no INT on these.
//end interrupt configuration
//begin of read mode configuration
//EXAMPLE CODE! values may also depend on target clock frequency
fmc_cache_controller_config = flash_location_containing_cache_controller_default_WS;
DATAW0 = flash_location_containing_flash_controller_default_WS;
CMD = CMD_SET_READ_MODE; //this starts the "set read modes" command
//no need to wait until command is completed: further accesses are stalled
//until the command is completed.
switch_to_target_clock_frequency();
//end of read mode configuration
//begin of program/erase configuration [optional]
DATAW0 = 0xf; //slowest clock for both program and erase
CMD = CMD_SET_WRITE_MODE; //no need to wait for completion
//end of program mode configuration
```

15.3.4 Memory power-down

In this controller, power-down is implemented as a command. See Command listing (CMD) for details on the power-down command.

During power-down, the memory will be placed in a mode where it draws a minimum amount of current.

During power-down (as with any other command) non-volatile memory controller is busy performing a command., and all memory read requests will be ignored.

Power-down is exited by a wake-up event, which can be triggered by writing a 1 in the WAKEUP bit of the EVENT register.

Power-down is also exited in case of a reset.

After that a wake-up event is triggered, the controller will wait for the memory to recover, and then end the power-down command, thus re-enabling read.

15.3.5 Code examples

In this example, powerdown is used as a low-power version of the CPU's WFI instruction. For this example to work, code is executed from flash, and the interrupt controller activates the *wake-up* input of a flash controller if a valid interrupt request arrives.

```
Enable_interrupt_sources(); //to be sure that wake-up will occur
CMD = CMD_POWERDOWN;
//Now the CPU will try to fetch the next word from the flash, which will stall
//because the flash is in powerdown mode. Whenever an interrupt request comes,
//the pending read will be completed, then (if CPU interrupts are enabled) the
```

```
//interrupt service routine is executed, then the following code is executed:
Process_interrupt_event();
```

In the following example, the CPU determines that it does need the flash for a period of time (all needed code is in ROM/RAM), and so temporarily turns it off. Be sure not to access flash when it is in power-down mode, otherwise the system will hang (a watchdog timer is recommended).

```
//executing from ROM/RAM:
INT_CLR_STATUS = 0x4; //clear the DONE status bit
CMD = CMD_POWERDOWN;
do_things_without_flash();
//when we need the flash again:
EVENT = 0x2; //WAKEUP event
while(!(INT_STATUS & 0x4 )) ; //wait until DONE is set
do_things_with_the_flash();
```

In the above example, the INT_STATUS register handling can be removed except when the flash is accessed after a wake-up is triggered and before the flash is ready, in which case the system may temporarily be stalled.

15.3.6 Reading

The memory is read through the AHB bus. Normal user memory is mapped on the AHB address space, as a contiguous address space, starting from address 0.

The Flash contains one additional word per page (the so-called "dmacc" word). Such words are not readable through the AHB bus. These words are managed internally by the controller in order to store a flag (all1), which can be used to verify whether a programming operation was prematurely terminated. See Command listing (CMD) .

Reading is not possible if the controller is executing a command.

15.3.7 Writing

A number of APB writes are required to fully define a memory word that is larger than 32 bits. The controller accumulates data inside its own internal storage, until the content of a full memory word has been specified. When this is done, the full word is transferred to the memory's page register (at the position specified by the STARTA register), as a single operation.

Data to be written is accumulated inside the controller's DATAW0-DATAW3 registers.

After specifying an address in the STARTA register and 128 bit of data in the DATAW0-3 registers, it's possible to activate the controller's *Write"* command, which transfers the data to the memory's page register, at the position indicated by the STARTA register (only the column part of the address is significant).

15.3.8 Erasing, programming, and verifying

Some controller commands can modify the content of the memory: program page, erase and page range. Other commands are targeted at verifying the content of the memory: checksum address range, blank check and margin check. Such commands operate either on a single address, specified by the STARTA register, or on an address range, specified by both STARTA and STOPA. Since all memory program/erase operations have a page granularity, column address bits are don't-care in the case of program, erase, and various other commands.

Additional command parameters may be required: they can be written in the DATAWx registers. Writes in STARTA, STOPA, and DATAWx registers can happen in any order, and have no other effect than modifying the register's content.

When all command parameters are set, the command can be started by writing the command's code into the CMD register.

During command execution, controller is busy, and access to some registers (CMD, STARTA, STOPA, DATAWx) is stalled. Other registers remain accessible so it is therefore possible to poll the INT_STATUS register and change INT_ENABLE. It is also possible to force an ERR or FAIL indication by writing to INT_SET_STATUS, in order to test the application's behavior, in the case of an error condition.

15.3.9 Code examples

This section presents an example of pseudocode to copy two pages (1024 bytes) of code from address src to address dst. Address dst is relative to the beginning of the flash address space, and is page-aligned (that is, a multiple of 512).

This code demonstrates the erase, write and program commands. If this code is not fetched from the flash itself (that is, it is fetched from RAM/ROM), accesses to the flash and controller never stall, therefore other masters are not prevented from accessing other resources on the bus. Interrupts are not needed, but they can be enabled and, as long as their service routines do not try to access the flash and controller, they retain their real-time performance.

```
int *src_i;
INT_CLR_STATUS = 0xf; //clear status register
STARTA = ((int)dst)>>4; //set start address. Assuming dst is a char*
STOPA = STARTA+32; //set end address. 1 page = 32 flash words.
CMD=CMD_ERASE_RANGE; //command: erase page range. Now erase starts.
while(!(INT_STATUS & 0x4 )) ; //wait until DONE is set
if(INT_STATUS & 3) handle_erase_errors();
//now write & program
src_i = (int *) src;
for(page=0; page < 2; page++) {
for(flashword=0; flashword<32; flashword++) {
INT_CLR_STATUS = 0xf; //clear status register
STARTA = flashword;
DATAW0 = *src_i++;
DATAW1 = *src_i++;
DATAW2 = *src_i++;
DATAW3 = *src_i++;
CMD=CMD_WRITE; //start write
while(!(INT_STATUS & 0x4 )) ; //wait until DONE is set
if(INT_STATUS & 3) handle_write_errors();
} //end of word loop
INT_CLR_STATUS = 0xf; //clear status register
STARTA = (((int)dst)>>4) + page*32
CMD=CMD_PROGRAM; //start program
while(!(INT_STATUS & 0x4 )) ; //wait until DONE is set
if(INT_STATUS & 3) handle_program_errors();
} //end of page loop
```

15.3.10 Command abort

Some commands can be aborted while they are executing if the application urgently requires access to the memory, such as in the case where there are erase and program commands which take a long time to complete.

An abort event can be specified through the ABORT bit of the EVENT register.

An aborted command flags unsuccessful completion by setting the FAIL bit in the INT_STATUS register. A failed program/erase command must be retried, even if the memory content appears to be intact (either the original one or the new one).

An abort request during the execution of a command that can be aborted does not necessarily result in a FAIL indication. When the request arrives very late in the command execution timeframe (i.e., when the command is already busy restoring safe read conditions) the request is ignored.

15.3.11 Verification

The flash and controller offer a number of commands to check whether the memory has been correctly programmed or erased. As a rule, there is no need to run any type of verification after programming or erasing, except for safety applications where the consequence of an error is known/deterministic. However, such commands come handy in order to verify whether a flash content modification has been allowed to complete successfully (for instance, a reset or power loss could interrupt an ongoing operation).

A simple example is provided below that shows how a small amount of regularly modified data can be handled to guarantee that, in case of power loss during a modification, valid data can always be retrieved (either the old data or the new data). In this example, the size of the data to be stored fits in a single flash page, leaving some room for locations required for algorithm management. Two pages are used: one normally contains the data, while the other is erased. When writing, new data is firstly programmed in the erased page, then old data is erased. The get_data() function returns the address of the page which contains valid data, performing cleanup of the other page if necessary (cleanup is necessary if programming or erasing was interrupted. In this case, one of the pages contains valid data while the other holds data halfway between programmed and erased levels). The put_data() function updates the stored data.

The concepts shown in this example can be adapted to different contexts (e.g., different data sizes), and optimizations can be performed (e.g., caching get_data() [intermediate] results to RAM, using multiple blank pages with one data page [to increase cycling endurance], sharing one backup page with multiple data pages [to reduce flash space – do not use a fixed backup page, otherwise it will be cycled too quickly], etc.

```
const char *page0 = address_of_1st_flash_page;
const char *page1 = address_of_2nd_flash_page;
char *get_data()
{
//DMACC words are all_0 for an erased page, all_1 for a programmed page
//doing a quick sanity check, to avoid time-consuming checks if not needed
//get_dmacc_status() returns 0 for all0, 1 for all1, 2 for any other content
int page0_status=get_dmacc_status(page0);
int page1_status=get_dmacc_status(page1);
if(page0_status==1 && page1_status==0) return page0;
if(page0_status==0 && page1_status==1) return page1;
//if we are here, the status of pages is not ideal... check full pages
//get_page_status returns 0 for a fully erased page, 1 for a correctly
//programmed page, 2 for a corrupted page
page0_status=get_page_status(page0, page0_status);
page1_status=get_page_status(page1, page1_status);
if(page0_status==2 && page1_status==2)
return do_recover(); //both pages marginal or KO
//at least one page is good (it's not possible that both are erased)
if(page0_status==1) {erase(page1); return page0;}
else {erase(page0); return page1;}
}
int get_dmacc_status(char *page)
{
int res;
STARTA = ((int)page)>>4;
DATAW0 = 0x8004; // read DMACC word, normal mode, ECC off
CMD= CMD_READ_SINGLE_WORD;
//the following access to DATAW0 is automatically stalled until the command completes
res=(DATAW0==0xffffffff)?1:(DATAW0==0)?0:2;
if(DATAW1!=DATAW0) return 2;
if(DATAW2!=DATAW0) return 2;
if(DATAW3!=DATAW0) return 2;
return res;
}
int get_page_status(char *page, int hint)
{
int res;
if(hint==2) return 2; //margin checks surely fail if usermode read is wrong
INT_CLR_STATUS=0x7; //clear DONE FAIL ERR (ERR is optional)
STARTA = STOPA = ((int)page)>>4;
CMD= hint? CMD_MARGIN_CHECK: CMD_BLANK_CHECK; //run the right command
while(!(INT_STATUS & 0x4 )) ; //wait until DONE: needed as INT_STATUS doesn't stall
return (INT_STATUS & 1)? 2: hint; //if the command does not fail, the hint was correct
```

```
}
void erase(char *page)
{
INT_CLR_STATUS=0x7; //clear DONE FAIL ERR (ERR is optional)
STARTA = STOPA = ((int)page)>>4;
CMD= CMD_ERASE_RANGE;
while(!(INT_STATUS & 0x4 )) ; //wait until DONE: needed as INT_STATUS doesn't stall
if(INT_STATUS&1) handle_hw_failure(); //erase of 1 page is always meant to pass
}
char *do_recover()
{
// check with the help of user_mode & signatures whether one of the pages still has
//valid data; re-write it to get better margin
char *good_page=NULL;
char *other_page;
char buf[512];
if(consistency_check(page0,buf)) good_page=page0;
else if (consistency_check(page1,buf)) good_page=page1;
if(!good_page) {
handle_hw_failure(); //we don't get consistent data anywhere
return NULL;//best would be that handle_hw_failure() does not return at all
}
//don't overwrite the ~good page, use the other one
erase(other_page);
program(buf,other_page); //the data which was previously read and found consistent
//is used to re-program; if we re-read good_page (which
//failed margin checks), we might get different data!
erase(good_page);
return other_page;
}
int consistency_check(char *page,char *buf)
{
int i;
int *ip; //assuming 32-bit integer
//If the optional check on many ECC corrections is performed (see below),
//this fragment of code is best executed from RAM/ROM, with other bus masters
//disabled, in order to avoid that other accesses cause additional ECC corrections
CMD=CMD_REPORT_ECC; //clear ECC datalog
ip=(int *)buf;
for(i=0;i<32;i++) { //32 words in a page
//use READ_SINGLE_WORD command to avoid bus errors on corrupt data
STARTA = ((int)page)>>4;
DATAW0 = 4; //read with ECC off
CMD= CMD_READ_SINGLE_WORD;
*ip++=DATAW0;
*ip++=DATAW1;
*ip++=DATAW2;
*ip++=DATAW3;
page+=16; //a word contains 16 bytes
}
CMD=CMD_REPORT_ECC; //get ECC datalog
//end of execution from RAM/ROM
if(DATAW1) return 0; //fail if there are uncorrectable words
//optional, if it's more risky to process dubious data than to report a data loss:
//if(DATAW2>treshold) return 0; //avoid too many corrections as well
return check_user_consistency(buf); //check based on the structure of the user payload
//For example, a checksum may have been added to the data; some values might be known
//to be within specific ranges; some fixed-content fields may be there; etc...
//Note that the check is performed on the buffer, not directly on the flash.
}
```

```
void put_data(char *src)
{
char *old_page=get_data();
char *new_page=(old_page==page1)? page0:page1;
//new_page is the page NOT returned by get_data(), the other one (expected blank)
INT_CLR_STATUS=0x7; //clear DONE FAIL ERR (ERR is optional)
STARTA = STOPA = ((int)new_page) >>4;
CMD= CMD_BLANK_CHECK; //needed to ensure that erase was properly completed: only
// the DMACC word was possibly checked
while(!(INT_STATUS & 0x4 )) ; //wait until DONE: needed as INT_STATUS doesn't stall
if(INT_STATUS&1) erase(new_page);
program(src,new_page); //copy the code into the new page.
erase(old_page);
}
```

The following example is targeted at verifying the correctness/integrity of a code area; it can be used for example after an application upgrade, or periodically to ensure that the correct code is still available (for example, not modified by a hacker, a programming error, and a HW failure). The area is delimited by start_address and end_address (end_address still included in the range). The content programmed in that page range has a known 128-bit checksum. Other than verifying the checksum, this example checks whether a high number of ECC corrections were found (an unexpected ECC uncorrectable error results in a failing checksum; *expected* errors can occur if erased pages are included in the checked range). Note that the "||" symbol indicates a concatenation of data.

```
//execute this code from RAM/ROM, so that fetching does not create
//additional ECC errors
CMD= CMD_REPORT_ECC; //clear ECC error count
STOPA = ((int)end_address) >> 4;
STARTA = ((int)start_address) >> 4;
CMD=CMD_CHECKSUM;
//the following access will stall until the checksum command is completed.
//if this is not desired, then either poll the DONE bit in the INT_STATUS register,
//as in previous code examples, or configure an interrupt to occur on DONE
//and wait for it before proceeding with execution.
if(DATAW0!=known_checksum[0] || DATAW1!=known_checksum[1] || DATAW2!=known_checksum[2]
|| DATAW3!=known_checksum[3]) return CHECK_FAILED;
CMD= CMD_REPORT_ECC; //get ECC error count
if(DATAW2>ECC_correction_treshold) return CHECK_FAILED; //singlebit corrections
return CHECK_PASSED;
```

15.3.12 ECC

The ECC function is normally transparent to the user. ECC abort function is enabled/disabled using ECC enable control bits in FMCCR register in SYSCON module.

When writing, parity is automatically computed and stored alongside user data.

When reading, data and parity are used to reconstruct correct data, even in the case of a 1-bit error.

ECC has to be taken into account only in the following contexts:

- In case of a correction or uncorrectable error, its condition and location are logged inside the controller, and an interrupt is optionally generated:
- In case of failure, the application must be able to take countermeasures, and even if execution is not endangered (when a correction is successfully performed), the application may choose to refresh memory data to avoid a subsequent error in the same word from causing a failure.
- Flags are made available, alongside read data and with the same timing, to identify ECC corrections and uncorrectable errors.

- When reading an erased location, an uncorrectable error is flagged. Use the "blank check" command to test for a successful erase.
- Due to the presence of ECC, over-programming an already programmed word will likely result in inconsistent parity bits. For this reason, programming a memory word, without first erasing it, is not allowed.
- If a program or erase operation is aborted, data and parity bits are unknown and probably inconsistent. In this case, the operation may result in unpredictable behavior and results (for example, when partially erasing a word, a bit which was previously already erased may be read as programmed, due to an inconsistent value of the parity bits).

Each data word has its associated parity bits, and only one wrong bit in the whole word (either in the data or in the parity) can be corrected. When more than one bit is wrong, the read result is unspecified (it is possible that no error is flagged, a correctable error is flagged, or an uncorrectable error is flagged).

Whenever a memory word is read by the controller, and a (correctable or uncorrectable) ECC error is identified, the address of the first occurrence of the most severe type of error is captured inside the controller; all other errors (correctable or uncorrectable) are separately counted (a saturating counter is used). A controller command allows this information to be read and contextually clears the logging information.

ECC is stored inverted, so that an ALL0 or ALL1 output from the memory is flagged as an uncorrectable error. This helps for safety and security, since most (hacker-induced) failures have a common-mode effect on all output bits.

15.3.13 Interrupts

There is a status register bit for each interrupt source, which is automatically set when the corresponding event occurs.

Each interrupt status bit has a corresponding interrupt enable bit; if the interrupt enable and status bits for at least one interrupt source are both set, an interrupt will be raised to the CPU (as long as the interrupt line number 0 is enabled inside the CPU registers).

The interrupt enable and status register bits are not writable directly: they are set by writing a 1 in the corresponding bit of the INT_SET_ENABLE and INT_SET_STATUS registers respectively, and are cleared by writing a 1 in the corresponding bit of the INT_CLR_ENABLE and INT_CLR_STATUS registers respectively.

If an enabled interrupt event occurs while the corresponding status register bit is being cleared, the interrupt request to the CPU is set high for at least one clock cycle.

The above provision is to ensure that no event is lost in the event that a new event occurs just before the CPU writes the INT_CLR_STATUS register. However, in this case, an interrupt can be triggered but it is not possible to determine its source among the ones available within the controller, since the status register would be cleared. It can be normally assumed that this is an ECC interrupt, since software is expected to first clear the indication of the completion status of a command, and only afterwards start a new operation of the same kind. The presence of an ECC error can be confirmed by clearing the ECC log information maintained inside the controller: in case that an ECC error indication is cleared in the status register before being processed, its presence would still be recorded in the ECC log info.

To maintain system integrity, the interrupt request to the CPU must be kept active until an interrupt service routine handles the interrupt, then the status register must be cleared while interrupts are disabled (either by means of the corresponding INT_ENABLE bit, or through some other viable means).

15.4 Memory map and register definition

Control and status information for the controller is mapped into register bits. All registers are 32 bit wide and can only be accessed as a whole word.

Registers are arranged in an address space that occupies a 4k byte space.

The Access field must be interpreted as follows: R = read, W = write, S = set (sets asserted bits, leaves others unchanged), C = clear, (clears asserted bits, leaves others unchanged), T= only accessible (R/W) in test mode (reserved in user mode).

S and C are special versions of write access, where the write data does not reflect the new register content, but indicates which bits must be set or cleared.

When multiple access types are supported, multiple characters are given. For example, R/W for registers that have both read and write access.

Within an otherwise accessible register, there may be reserved register bits, which can be neither read nor written. When the read access is not explicitly specified, read access is not supported.

Inside a register marked R/W there could be read-only bits.

15.4.1 Command listing (CMD)

This section lists all commands that can be specified in the CMD register. Irrespective of how command execution is triggered, any ongoing memory read is completed before the actual command execution starts. When command execution is triggered, but not yet started, the command is said to be pending.

When any command completes execution, it sets the DONE bit in the INTSTAT register. All commands report failure and error status bits as specified in their respective description; such flags are not listed in the command's output results. In general, when an error is detected (either by command execution or in the case where no command could be executed), no command result is defined, not even a fail status is generated. Therefore, if a command (or a CMD register write operation) sets the INTSTAT[ERR] bit, it will not modify the FAIL bits and the result registers.

NOTE

All registers capable of holding a command result (DATAWx) are always updated by a running command. If no specific result is listed for any of these registers, its content remains undefined after a command execution is attempted.

When a register (STARTA, STOPA, DATAWx, etc.,) contains an address, this is a physical word address inside the flash memory (that is, address 1 represents the second 128-bit word inside the flash memory, not the second byte in the first word). When a page address is required/returned, the five least significant bits of the address are don't-care (a flash page contains 32 user-accessible words).

Addresses and address ranges given as parameters have to be within the address range of the memory.

Table 42. CMD listing

Command	Value	Parameters	Output	Description
CMD_INIT	0	None	None	Initialization. Automatically triggered when exiting from Reset.
CMD_POWERDOWN	1	DATAW0 (See Table 43)		When this command is initiated, the flash and controller enter power-down mode. During power-down (as with any other command), the flash is not accessible and the power-down command waits indefinitely for a wake-up event. When such an event happens (triggered by the EVENT register), the controller will disable flash power-down and then will wait until the flash is ready for operation, with a time-out of 4096 clock cycles; FAIL is reported if the time-out is reached. Then the command terminates.
CMD_SET_READ_MODE	2	DATAW0 (see Table 43 for the meaning of each bit)	None	The flash data sheet reports the minimum duration of the pre-charge (Tp) and evaluation (Tdpd) as a function of the memory size, and depends on whether EWLE is active or not.

Table continues on the next page...

Table 42. CMD listing (continued)

Command	Value	Parameters	Output	Description
				Select the figures for EWLE=1, sum them up, add ~34ns (to take address path and ECC delay, wire delay, jitter, read delay uncertainty, data setup into account: the exact value is determined after synthesis), then divide the result by the clock period, rounding down the division result to an integer: this will give the values to specify in bits 3:0 of the DATAW0 register. The clock frequency should be kept constant while a controller command is being executed.
CMD_READ_SINGLE_WORD	3	STARTA (flash word address), DATAW0 (read mode). See Table 44.	DATAW0-3: Read Data	This command reads a single memory word, using a specified combination of read modes. For instance, it is possible to perform a read of the DMACC word with ECC disabled.The controller will respond to the command by setting the ERR flag if an illegal mode combination is requested. Depending on the chosen modes, the controller ensures that adequate settling times are met, both when the modes are activated and when they are deactivated.
CMD_ERASE_RANGE	4	STARTA, STOPA	None	The range from the page containing the STARTA address to the page containing STOPA (included) is erased. An abort event interrupts erasing, unless the event happens very late in the erase process (when the flash is discharging high voltages and reconfiguring itself for reading), in which case it would have no effect. If abort influences the erase process, the FAIL flag is set. When erasing completes, the controller waits until the flash is ready for operation, with a time-out of 4096 clock cycles; FAIL is reported if the time-out is reached. Then the command completes. The FAIL flag is also set in the case the flash reports an HV error (requested high voltages could not be reached). If STARTA points to a page following the one pointed by STOPA, no page is erased and the ERR flag is set.
CMD_BLANK_CHECK	5	STARTA, STOPA	DATAW0 contains address inside the first failing page (if any). If	The range from the page containing address STARTA to the page containing STOPA (included) is checked. The selected pages are checked for the erased condition (all0 including parity), with a specific margin read

Table continues on the next page...

Table 42. CMD listing (continued)

Command	Value	Parameters	Output	Description
			the FAIL flag is not set, the content of DATAW0 is not significant. Do not assume that this is the address of the first failing word; in the case of a DMACC word failure, such an address would not be representable.	mode. ECC is off during the check (single bit errors cause failures). If a page is found which is not correctly erased, the FAIL flag is set, the page is reported on DATAW0 and processing stops. The check is performed in incrementing address order, so that, in case of fail, it is known that pages at a lower address than the failing one are successfully verified. To know the individual status of all selected pages, when a fail is reported on a page which is not the last in the range, the command should be restarted with the page following the failing one being selected as start page. Checking a page range is more time-efficient than individually running the check command on single pages. If STARTA points to a page following the one pointed by STOPA, no page is checked and the ERR flag is set. As a side effect of this command, the ECC log is cleared. This is because the same HW resources are used to record the failing page.
CMD_MARGIN_CHECK	6	STARTA, STOPA	DATAW0: an address inside the first failing page (if any).	This command checks the selected page range for correct programming. If, for any reason, programming was interrupted or disturbed, or erase was performed without a subsequent programming, this check fails.
CMD_CHECKSUM	7	STARTA, STOPA	DATAW0-3, the computed checksum	
CMD_WRITE	8	STARTA, DATAW0-3:word to be written	None	The selected word is copied into the page register, at the specified position. STARTA is the column address of the word to be written.
CMD_WRITE_PROG	10	STARTA, DATAW0-3: word to be written	None	This command first performs a "write word" command, then, if the written word was the last of a page, it performs a "program page" command.
CMD_PROGRAM	12	STARTA	None	First, an all1 value (data+parity) is stored in the page register in the location corresponding to the DMACC word(*). Then, programming is started, which copies the page register content into the selected page. The controller waits until the flash is ready for operation, with a time out of 4096 clock cycles; FAIL is reported if the time out is reached.

Table continues on the next page...

Table 42. CMD listing (continued)

Command	Value	Parameters	Output	Description
CMD_REPORT_ECC	13	None	DATAW0: address of first word with ECC event DATAW1: number of uncorrectable errors found DATAW2: number of corrections performed	All ECC events are logged, both for reads performed by user code and for internally-generated reads (e.g. checksum and "read word" commands, initialization…). This command copies logging information to the DATAW0-2 registers, and then clears the log, zeroing the counters. 20-bit counters are used. When they reach their maximum value, further incrementing is prevented (that is, they saturate rather than wrapping around). As the DMACC word is not meant to contain ECC-encoded data, ECC errors are not logged for it.

Table 43. Bitfield description

Bit	Function
[31-4]	Reserved. Do not modify.
[3-0]	Number of extra wait states for controller-internal reads.

Table 44. Bitfield description

Bit	Function
[15]	Read DMACC word.
[14-12]	Reserved.
[11-10]	00: normal read. 01: margin vs program. 10: margin vs erase. 11: illegal bit combination.
[9-3]	Reserved.
[2]	Read with ECC off.
[1-0]	Reserved.

15.4.2 FLASH register descriptions

15.4.2.1 FLASH memory map

FLASH base address: 4003_4000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Command (CMD)	32	WO	0000_0000
4	Event (EVENT)	32	See section	See section
10	Start address for next flash command (STARTA)	32	See section	See section
14	End address for next flash command (STOPA)	32	See section	See section
80 - 8C	Data register (DATAW0 - DATAW3)	32	RW	0000_0000
FD8	Clear interrupt enables (INTEN_CLR)	32	See section	See section
FDC	Set interrupt enables (INTEN_SET)	32	See section	See section
FE0	Interrupt status (INTSTAT)	32	See section	See section
FE4	Interrupt enable (INTEN)	32	See section	See section
FE8	Clear interrupt status (INTSTAT_CLR)	32	See section	See section
FEC	Set interrupt status (INTSTAT_SET)	32	See section	See section
FFC	Module identification (MODULE_ID)	32	See section	C40A_2100

15.4.2.1.1 Command (CMD)

The controller manages the execution of *commands*. The *commands* encompass any action performed by the controller, for example, a mode change, or programming, erasing, or calculating a checksum over an address range. See Command listing (CMD) for a list of available commands.

A command usually has parameters, such as an address or address range, data to be written, and a mode specification. Parameters must be written into corresponding registers before the command is started. The writing of parameters has no effect until the command is started.

Command execution is triggered when writing to the CMD register.

When a command is executed, it sets the appropriate bits in the INTSTAT registers. Some commands also return additional information in other registers.

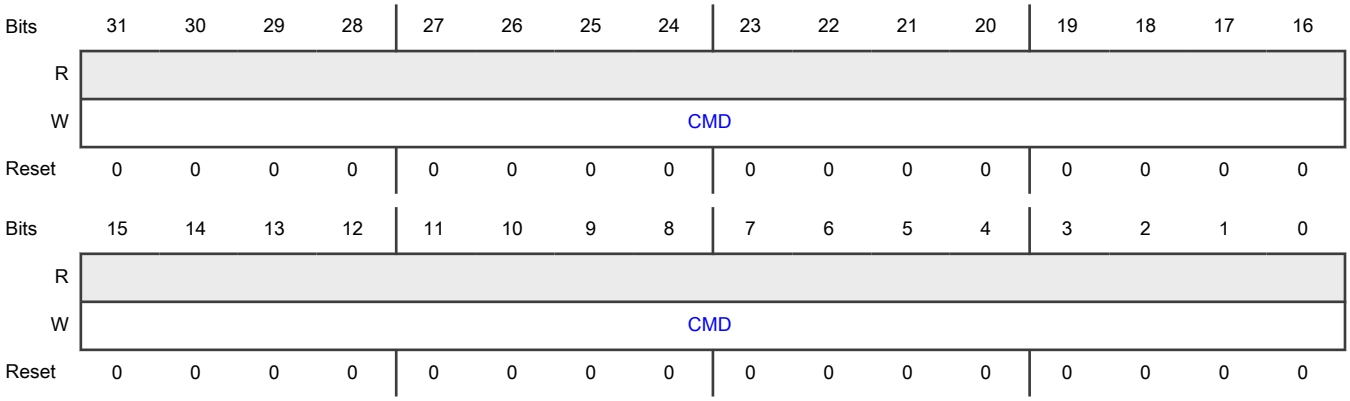
NOTE

Associated prefetches should be disabled before issuing any Flash commands.

Offset

Register	Offset
CMD	0h

Diagram



Fields

Field	Description
31-0 CMD	command register.

15.4.2.1.2 Event (EVENT)

As a general rule, when the controller is busy executing a command it is not possible to give further orders, and all registers involved in command execution cannot be used (an access would stall the APB bus).

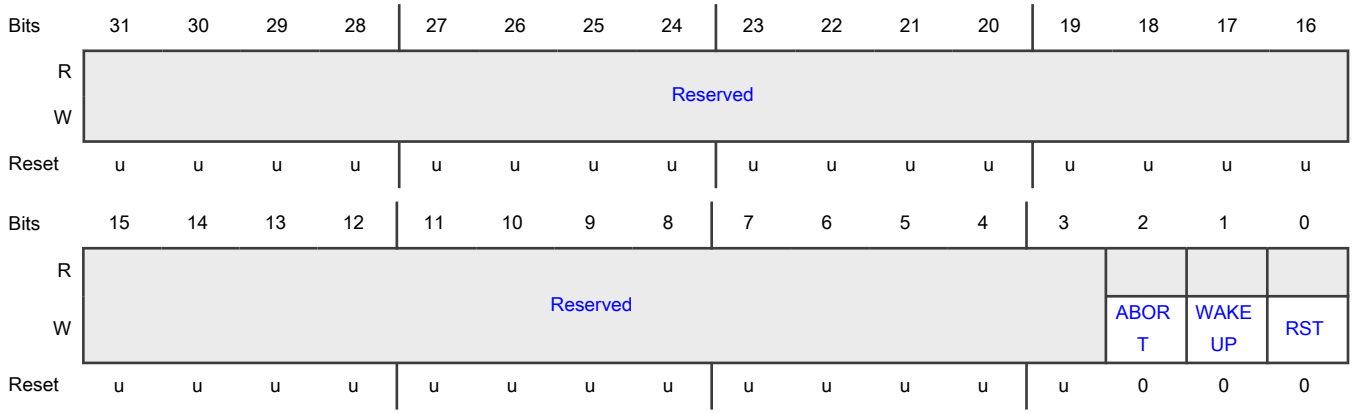
However, some events may also be generated when the controller is busy executing a command, and these events would influence the command being executed. Examples of such events are a reset, a command abort request, or a wake-up from a power-down.

The event register generates such events through software.The event register is write-only. The act of writing the register with one of the bits at 1 activates the generation of the corresponding event.

Offset

Register	Offset
EVENT	4h

Diagram



Fields

Field	Description
31-3 —	Reserved
2 ABORT	When bit is set, a running program/erase command is aborted.
1 WAKEUP	When bit is set, the controller wakes up from whatever low power or powerdown mode was active.
0 RST	When bit is set, the controller and flash are reset.

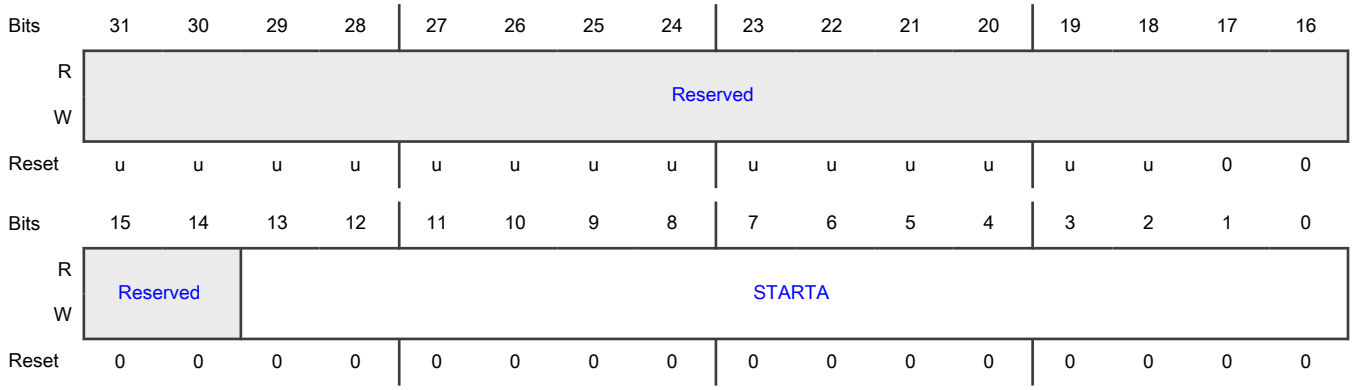
15.4.2.1.3 Start address for next flash command (STARTA)

STARTA contains a parameter that is never updated by a running command. STARTA is used in the command to specify the start address. This register contains the address in units of memory words and not bytes.

Offset

Register	Offset
STARTA	10h

Diagram



Fields

Field	Description
31-14 —	Reserved
13-0 STARTA	Address / Start address for commands that take an address (range) as a parameter.

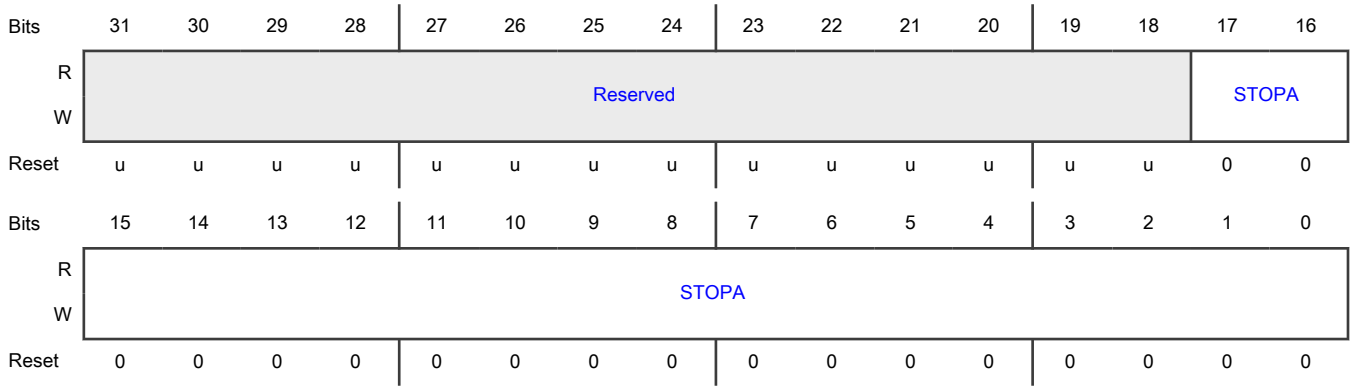
15.4.2.1.4 End address for next flash command (STOPA)

STOPA contains a parameter that is never updated by a running command. STOPA is used in the command to specify the end address. This register contains the address in units of memory words and not bytes.

Offset

Register	Offset
STOPA	14h

Diagram



Fields

Field	Description
31-18 —	Reserved
17-0 STOPA	Stop address for commands that take an address range as a parameter (the word specified by STOPA is included in the address range).

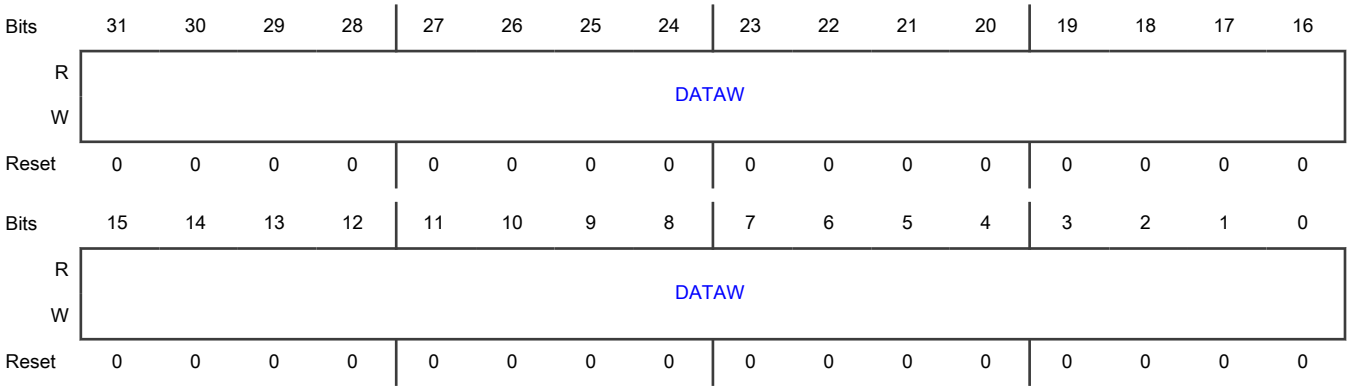
15.4.2.1.5 Data register (DATAW0 - DATAW3)

DATAWi registers are always updated as a result of executing a controller command, even if the command description does not report a result to be returned on some or all registers.

Offset

Register	Offset
DATAW0	80h
DATAW1	84h
DATAW2	88h
DATAW3	8Ch

Diagram



Fields

Field	Description
31-0 DATAW	Memory data, or command parameter, or command result.

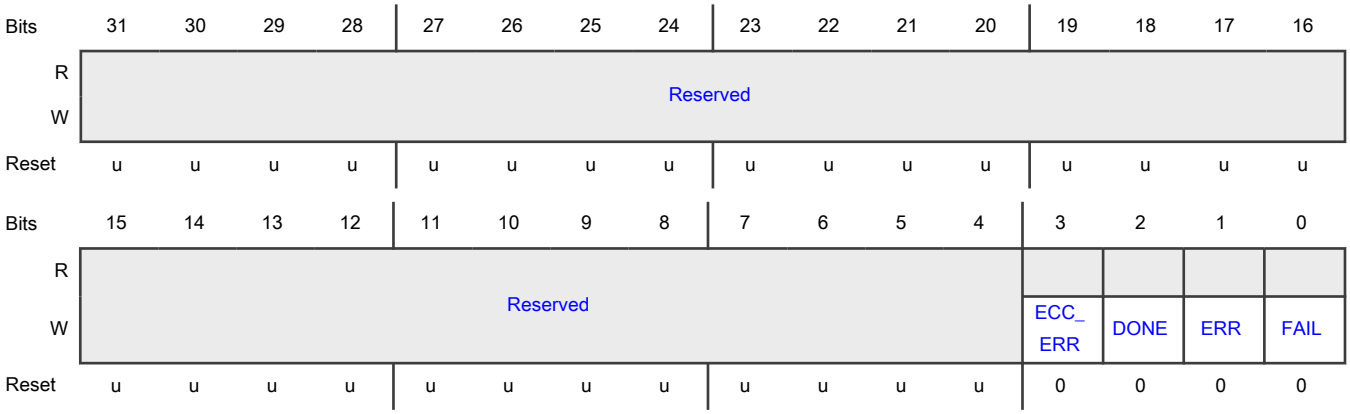
15.4.2.1.6 Clear interrupt enables (INTEN_CLR)

When an INTEN_CLR bit is set to 1, the corresponding INTEN bit is cleared.

Offset

Register	Offset
INTEN_CLR	FD8h

Diagram



Fields

Field	Description
31-4 —	Reserved
3 ECC_ERR	Clears the ECC error interrupt.
2 DONE	Clears the done interrupt.
1 ERR	Clears the error interrupt.
0 FAIL	Clears the fail interrupt.

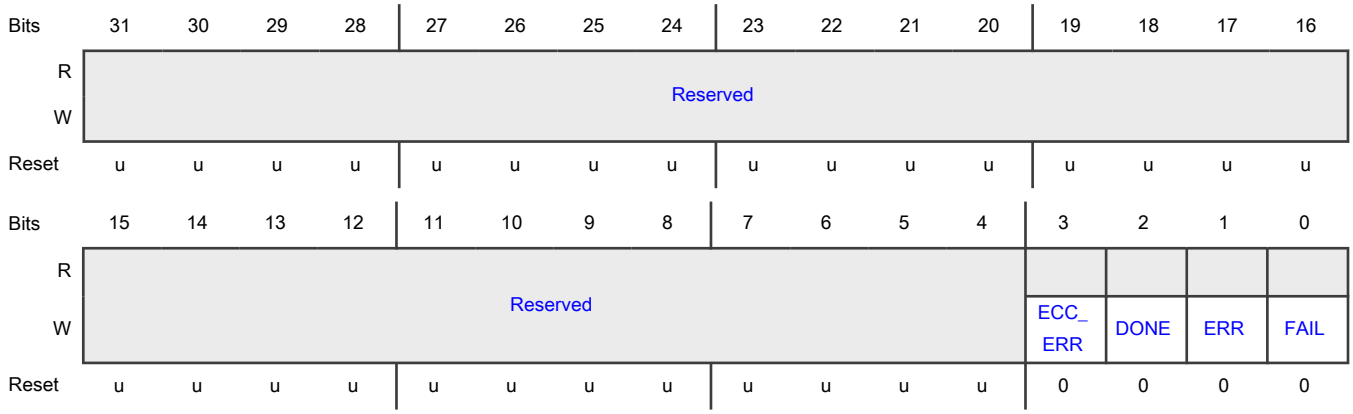
15.4.2.1.7 Set interrupt enables (INTEN_SET)

When a INTEN_SET bit is set to 1, the corresponding INTEN bit is set.

Offset

Register	Offset
INTEN_SET	FDCh

Diagram



Fields

Field	Description
31-4 —	Reserved
3 ECC_ERR	Sets ECC error interrupt.
2 DONE	Sets done interrupt.
1 ERR	Sets error interrupt
0 FAIL	Sets Fail interrupt.

15.4.2.1.8 Interrupt status (INTSTAT)

Interrupt registers determine when the controller generates an interrupt request. The interrupt is asserted when the bit-wise AND of INTSTAT and INTEN is nonzero.

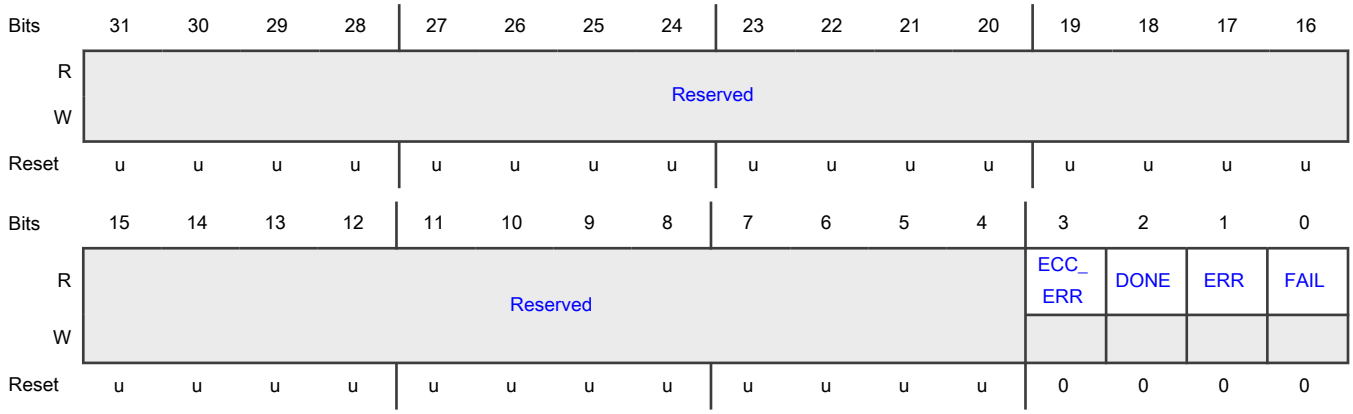
If the corresponding INTEN bit is zero, an INTSTAT register bit can be polled to test for the occurrence of an event.

The INTSTAT register can be set for software testing purpose, by writing into the INTSTAT_SET register.

Offset

Register	Offset
INTSTAT	FE0h

Diagram



Fields

Field	Description
31-4 —	Reserved
3 ECC_ERR	This status bit is set if, during a memory read operation (either a user-requested read, or a speculative read, or reads performed by a controller command), a correctable or uncorrectable error is detected by ECC decoding logic.
2 DONE	This status bit is set at the end of command execution.
1 ERR	This status bit is set if execution of an illegal command is detected.
0 FAIL	This status bit is set if execution of a (legal) command failed.

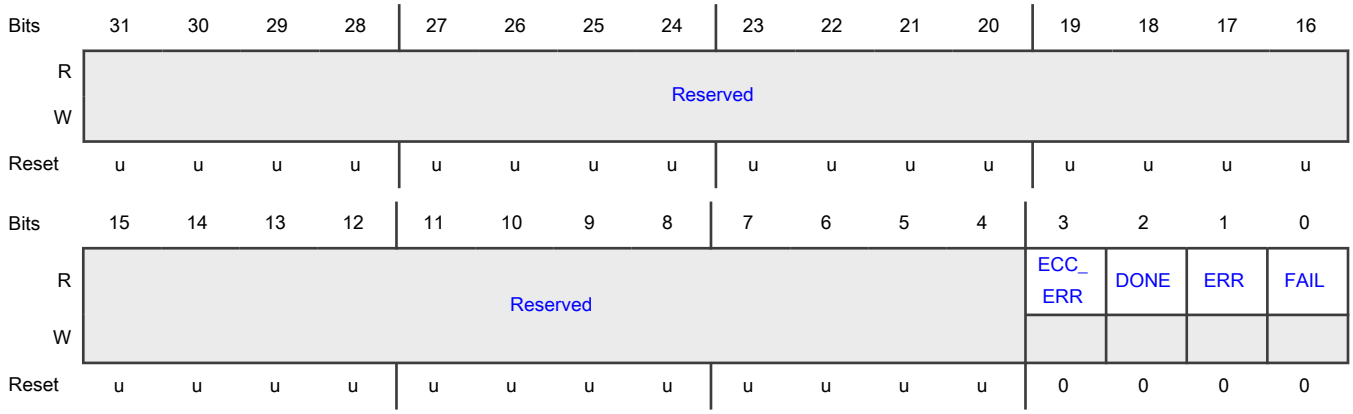
15.4.2.1.9 Interrupt enable (INTEN)

If an INTEN bit is set, an interrupt request will be generated if the corresponding INTSTAT bit is high

Offset

Register	Offset
INTEN	FE4h

Diagram



Fields

Field	Description
31-4 —	Reserved
3 ECC_ERR	Enables ECC error interrupt.
2 DONE	Enables done interrupt.
1 ERR	Enables error interrupt.
0 FAIL	Enables fail interrupt.

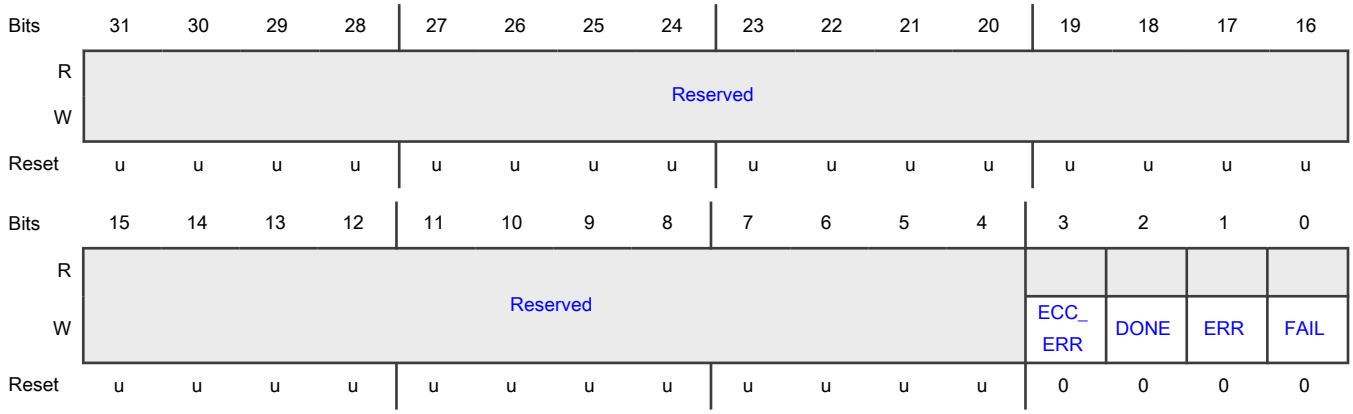
15.4.2.1.10 Clear interrupt status (INTSTAT_CLR)

When a INTSTAT_CLR bit is set to 1, the corresponding INTSTAT bit is cleared.

Offset

Register	Offset
INTSTAT_CLR	FE8h

Diagram



Fields

Field	Description
31-4 —	Reserved
3 ECC_ERR	Clears ECC error interrupt status.
2 DONE	Clears done interrupt status.
1 ERR	Clears error interrupt status.
0 FAIL	Clears fail interrupt status.

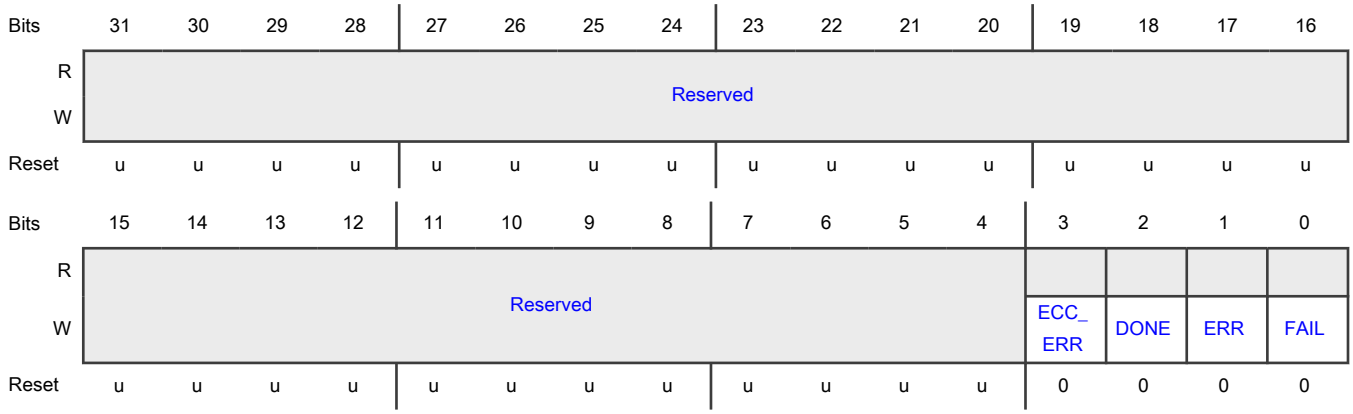
15.4.2.1.11 Set interrupt status (INTSTAT_SET)

When a INTSTAT_SET bit is set to 1, the corresponding INTSTAT bit is set.

Offset

Register	Offset
INTSTAT_SET	FECh

Diagram



Fields

Field	Description
31-4 —	Reserved
3 ECC_ERR	Sets ECC error interrupt status.
2 DONE	Sets done interrupt status.
1 ERR	Sets error interrupt status.
0 FAIL	Sets fail interrupt status.

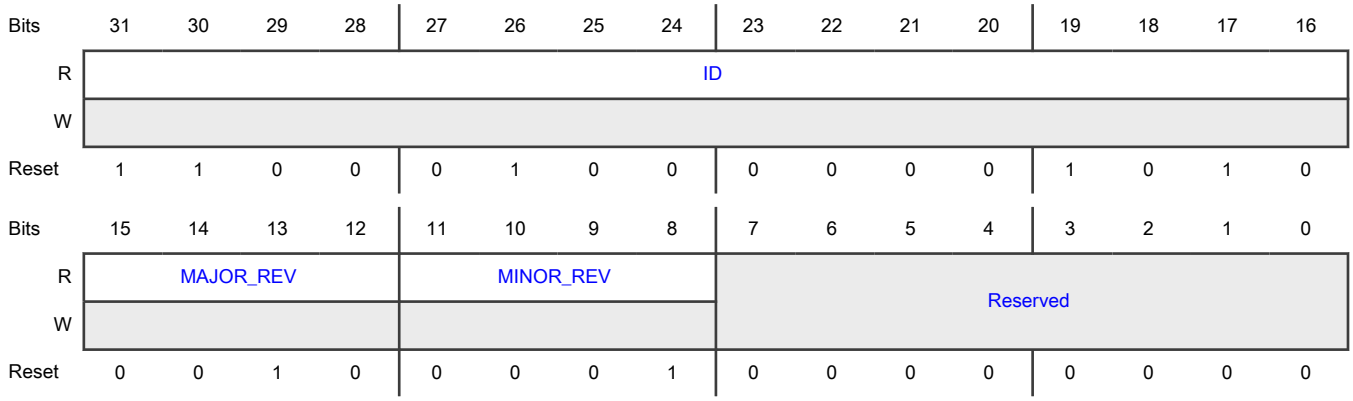
15.4.2.1.12 Module identification (MODULE_ID)

The purpose of this read-only register is to give information over the controller version.

Offset

Register	Offset
MODULE_ID	FFCh

Diagram



Fields

Field	Description
31-16 ID	Identifier.
15-12 MAJOR_REV	Major revision i.
11-8 MINOR_REV	Minor revision i.
7-0 —	Reserved

Chapter 16
Cache Controller

16.1 Chip-specific CACHE64 information

Table 45. Reference links to related information

Topic	Related module	Reference
Full description	CACHE64 Controller	CACHE64 Controller
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

Cache Controller register bitfield CSAR[28] is always 1’b0.

16.1.1 Module instances

This device has only one instance of the CACHE64 module, CACHE64_CTRL0. The 8 KB of cache is for accesses to external memory through the FlexSPI peripheral. The cache policy is programmable using the POLSEL.

16.2 Overview

The Cache Memory Controller (CACHE64) is a general purpose AMBA AHB bus protocol cache.

16.2.1 Block Diagram

The following figure shows the block diagram:

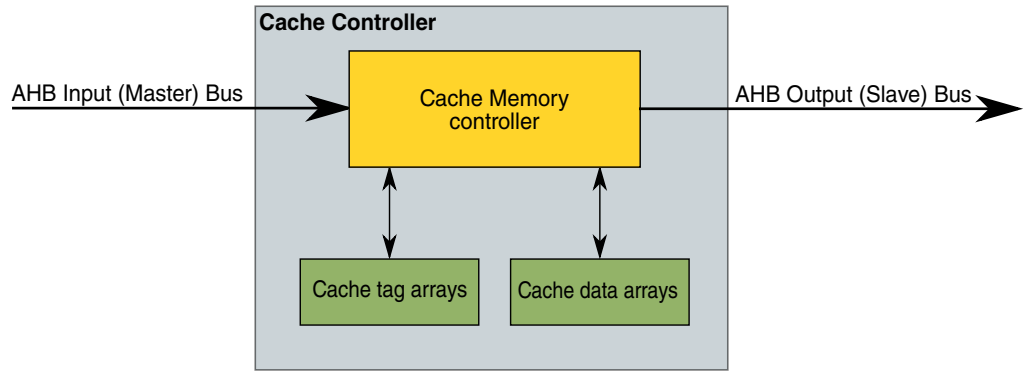


Figure 29. Cache controller block diagram

16.2.2 Features

The cache controller is designed to work with AMBA AHB bus systems. It has:

- An input (master) AHB bus. Accesses have cache mode attributes for cacheable copyback, cacheable writethrough, or non-cacheable mode operation and also write bufferable mode operation on an access by access basis.

- An output (slave) 64-bit data path AHB bus for downstream cache read miss, cache writethrough write, and non-cacheable accesses.
- A peripheral interface for accessing the cache's program model.

16.3 Functional Description

A cache is a block of high-speed memory locations containing address information (commonly known as a tag) and the associated data. The purpose is to decrease the average time of a memory access. Caches operate on two principles of locality:

- Spatial locality — An access to one location is likely to be followed by accesses from adjacent locations (for example, sequential instruction execution or usage of a data structure).
- Temporal locality — An access to an area of memory is likely to be repeated within a short time period (for example, execution of a code loop).

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

Temporal locality is achieved by keeping recently accessed lines in the cache.

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

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

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

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

The cache controller supports the following modes of operation:

1. Write-through — access to address spaces with this cache mode are cacheable.
    - If all cacheable spaces are read-only spaces, the cache will contain read-only data and all writes to the cache will fault. See the chip-specific cacheable space information.
    - A read miss on the input bus causes a line read on the output bus of a 32-byte-aligned memory address containing the desired address. This miss data is loaded into the cache and is marked as valid and not modified.
    - A read hit to a valid cache location returns data from the cache with no output bus access.
    - A write miss bypasses the cache and writes to the output bus (no allocate on write miss policy for write-through mode spaces).
    - A write-through write hit updates the cache hit line with the write data and writes to the output bus.
    - The caches are bus-master-local and do not support hardware cache coherency. If the bus master has accessed write-through regions and another bus master that can access the same target memory without going through the cache (such as DMA) then needs update these regions, software must first perform explicit cache clears to any needed cache memory range to ensure all modified cache lines update their associated memories before being modified by external masters and subsequent cache bus master accesses will get the updated memory.
2. Write-back — access to address spaces with this cache mode are cacheable.
    - A read miss on the input bus will cause a line read on the output bus of a 32-byte-aligned memory address containing the desired address. This miss data is loaded into the cache and marked as valid and not modified.
    - A read hit to a valid cache location will return data from the cache with no output bus access.
    - A write miss will do a "read-to-write" (allocate on write miss policy for write-back mode spaces). A line read on the output bus of a 32 byte aligned memory address containing the desired write address is performed. This miss data

is loaded into the cache and marked as valid and modified; and the write data will then update the appropriate cache data locations.

- A write hit updates the cache hit line with the write data plus marks the line as modified.
- The caches do not support hardware cache coherency. If the cache's master bus has written to write-back regions and another bus master that can access the same target memory without going through the cache then needs to see these updates, software must perform explicit cache pushes to any needed cache memory range to ensure all modified cache lines update their associated memories before being read by this master. Likewise, if the cache's master bus has accessed write-back regions and another bus master that can access the same target memory without going through the cache then needs update these regions, software must first perform explicit cache clears to any needed cache memory range to ensure all modified cache lines update their associated memories before being modified by this master and subsequent cache master bus accesses will get the updated memory.

3. Non-cacheable — access to address spaces with this cache mode are not cacheable. These accesses bypass the cache and access the output bus.

16.3.1 Cache Configuration

The Cache Controller receives the following request:

- Master bus requests on the Master (M) bus

The programming model for the Cache is accessed via the cache controller's AMBA APB bus.

This controller then processes the cacheable accesses as needed, while bypassing the non-cacheable, cache write-through, cache miss, and cache maintenance accesses to its slave bus.

The cache on this device is structured as follows. The cache has a 2-way set-associative cache structure with a total size of 8 KBytes for the cache. The cache has 32-bit address, 64-bit data paths and a 32-byte line size. The cache tags and data storage use single-port, synchronous RAMs.

For this 8-KByte cache, each cache TAG function uses two 128 x 22-bit RAM arrays and the cache DATA function uses two 512 x 64-bit RAM arrays. The cache TAG entries store 20 bits of upper address as well as a modified and valid bit per cache line. The cache DATA entries store eight byes of code or data.

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

CACHE - 8 KByte size = (128 sets) x (32-byte lines) x (2-way set associative)

Table 46. Tag Cache Address Use

Tag Cache Address Use	Cache
Tag Hit Address Range	Address[31:12]
Tag Set Select Address Range	Address[11:5] used to select 1 of 128 sets
Not Used	Address[4:0]

Table 47. Data Cache Address Use

Data Cache Address Use	System Cache
Not Used	Address[31:12]
Data Set Select Address Range	Address[11:5] used to select one of 128 sets
64-bit word select	Address[4:3] used to select one of four 64-bit words within a set
Byte select	Address[2:0] used to select the byte within the 64-bit word

16.3.2 Cache Control

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

16.3.2.1 Cache set commands

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

- all of way 0,
- all of way 1
- all of both ways (complete cache)

Cache set commands are specified using the upper bits in the CCR register. Cache set commands perform their operation on the cache independent of the cache enable bit, CCR[ENCACHE].

A cache set command is initiated by setting the CCR[GO] bit. This bit also acts as a busy bit for set commands. It stays set while the command is active and is cleared by the hardware when the set command completes.

Supported cache set commands are given in Table 48. Set commands work as follows:

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

Table 48. Cache Set Commands

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

After a reset, complete an invalidate cache command before using the cache. It is possible to combine the cache invalidate command with the cache enable. That is, setting CCR to 0x8500_0003 will invalidate the cache and enable the cache and write buffer.

16.3.2.2 Cache line commands

Cache line commands operate on a single line in the cache at a time. Cache line commands can be performed using a physical or cache address.

- A cache address consists of a set address and a way select. The line command acts on the specified cache line.
- Cache line commands with physical addresses first search both ways of the cache set specified by the following physical address bits. If they hit, the commands perform their action on the hit way:
  - For Cache - [11:5]

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

A line cache command is initiated by setting the line command go bit (CLCR[LGO] or CSAR[LGO]). This bit also acts a a busy bit for line commands. It stays set while the command is active and is cleared by the hardware when the command completes.

The CLCR[27:24] bits select the line command as follows:

Table 49. Cache Line Commands

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

16.3.2.2.1 Executing a series of line commands using cache addresses

A series of line commands with incremental cache addresses can be performed by writing to the CLCR.

1. Place the command in CLCR[27:24]
2. Set the way (CLCR[WSEL]) and tag/data (CLCR[TDSEL]) controls as needed
3. Place the cache address in CLCR[CACHEADDR]

4. Set the line command go bit (CLCR[LGO])
5. Wait for the command to complete (CLCR[LGO] clears)
6. Increment the cache address (at bit 3 to step through data or at bit 5 to step through lines), and
7. Set the line command go bit (CLCR[LGO]).
8. Wait for the command to complete (CLCR[LGO] clears)
9. Repeat steps 5 - 8 as needed

16.3.2.2.2 Executing a series of line commands using physical addresses

Perform a series of line commands with incremental physical addresses using the following steps:

1. Write to the CLCR.
   - Place the command in CLCR[27:24]
   - Set the tag/data (CLCR[TDSEL]) control
2. Place the physical address in CSAR[PHYADDR] and set the line command go bit (CSAR[LGO])
3. Wait for the command to complete (CLCR[LGO] clears)
4. Increment the physical address (at bit 3 to step through data or at bit 5 to step through lines), and
5. Set the line command go bit (CLCR[LGO]).
6. Wait for the command to complete (CLCR[LGO] clears)
7. Repeat steps 3 - 6 as needed

The line command go bit is shared between the CLCR and CSAR registers, so that the above steps can be completed in a single write to the CSAR register.

16.3.2.2.3 Line command results

At completion of a line command, the CLCR register contains information on the initial state of the line targeted by the command. For line commands with cache addresses, this information is read before the line command action is performed from the targeted cache line. For line commands with physical addresses, this information is read on a hit before the line command action is performed from the hit cache line. For line commands with physical address that miss in the cache, the initial valid bit (LCIVB) is cleared and no line command action is performed. In general, if the valid indicator (CLCR[LCIVB] is cleared, the targeted line was invalid at the start of the line command and no line operation is performed.

Table 50. Line command results

CLCR[22:20]			For cache address commands	For physical address commands
LCWAY	LCIMB	LCIVB		
0	0	0	Way 0 line was invalid	No hit
0	0	1	Way 0 valid, not modified	Way 0 valid, not modified
0	1	0	Way 0 line was invalid	No hit
0	1	1	Way 0 valid and modified	Way 0 valid and modified
1	0	0	Way 1 line was invalid	No hit
1	0	1	Way 1 valid, not modified	Way 1 valid, not modified
1	1	0	Way 1 line was invalid	No hit
1	1	1	Way 1 valid and modified	Way 1 valid and modified

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

The cache does not have line lock capability.

16.3.3 Clocks

The Cache uses the AHB master bus clock.

16.3.4 Resets

The Cache is disabled at reset. Cache tag and data arrays are not cleared at reset. See Initialization and Cache Control for more information regarding Cache initialization and configuration.

16.4 External Signals

There are no external signals for the Cache Controller.

16.5 Initialization

The cache initialization is as follows:

1. Set INVW1 and INVW0 in the CCR register
2. Set GO in the CCR register
3. Enable ENWRBUF and ENCACHE bits in the CCR register

NOTE

The above can be done simultaneously. That is, setting CCR to 0x8500_0003 will invalidate the cache and enable the cache and write buffer.

16.6 Memory Map and Registers

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

NOTE

The Cache registers are accessible in supervisor mode only.

16.6.1 CACHE64 register descriptions

16.6.1.1 CACHE64 memory map

CACHE64_CTRL0 base address: 4002_E000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
800	Cache control register (CCR)	32	See section	0000_0000
804	Cache line control register (CLCR)	32	See section	0000_0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
808	Cache search address register (CSAR)	32	RW	0000_0000
80C	Cache read/write value register (CCVR)	32	RW	0000_0000

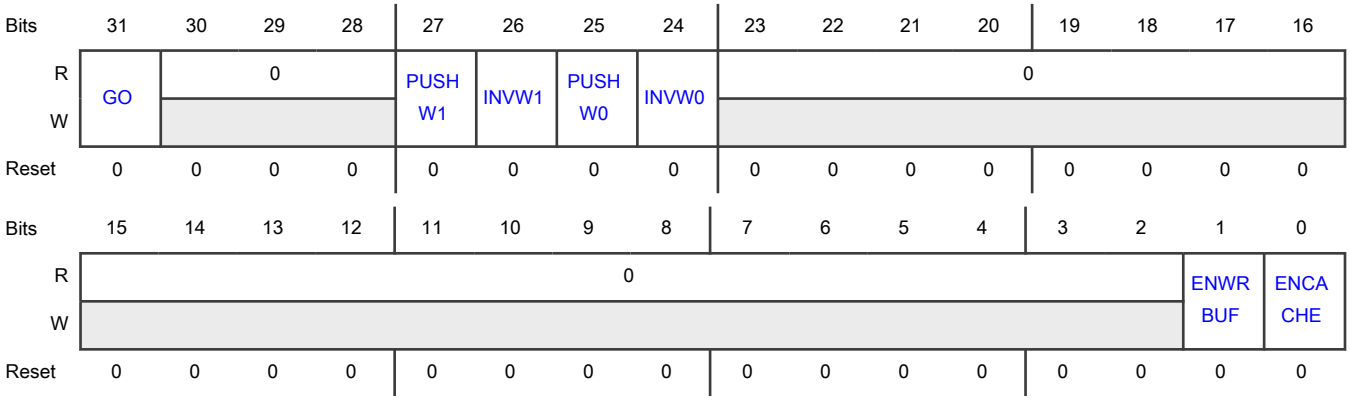
16.6.1.1.1 Cache control register (CCR)

The CCR is used to enable the cache and write buffer. The CCR also contains the fields used for Cache set commands.

Offset

Register	Offset
CCR	800h

Diagram



Fields

Field	Description
31 GO	Initiate Cache Command Setting this bit initiates the cache command indicated by bits 27-24. Reading this bit indicates if a command is active. **NOTE** This bit stays set until the command completes. Writing zero has no effect. 0 - Write: no effect. Read: no cache command active. 1 - Write: initiate command indicated by bits 27-24. Read: cache command active.
30-28 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
27 PUSHW1	Push Way 1 Along with bits INVW0, PUSHW0, and INVW1, this bit is used to specify the cache command to execute the next time the GO bit is set. 0 - No operation 1 - When setting the GO bit, push all modified lines in way 1
26 INVW1	Invalidate Way 1 Along with bits INVW0, PUSHW0, and PUSHW1, this bit is used to specify the cache command to execute the next time the GO bit is set. **NOTE** If the PUSHW1 and INVW1 bits are set, then after setting the GO bit, push all modified lines in way 1 and invalidate all lines in way 1 (clear way 1). 0 - No operation 1 - When setting the GO bit, invalidate all lines in way 1
25 PUSHW0	Push Way 0 Along with bits INVW0, INVW1, and PUSHW1, this bit is used to specify the cache command to execute the next time the GO bit is set. 0 - No operation 1 - When setting the GO bit, push all modified lines in way 0
24 INVW0	Invalidate Way 0 Along with bits PUSHW0, INVW1, and PUSHW1, this bit is used to specify the cache command to execute the next time the GO bit is set. **NOTE** If the PUSHW0 and INVW0 bits are set, then after setting the GO bit, push all modified lines in way 0 and invalidate all lines in way 0 (clear way 0). 0 - No operation 1 - When setting the GO bit, invalidate all lines in way 0.
23-2 —	Reserved
1 ENWRBUF	Enable Write Buffer This bit enables/disables the write buffer. 0 - Write buffer disabled 1 - Write buffer enabled
0	Cache enable

Table continues on the next page...

Table continued from the previous page...

Field	Description
ENCACHE	This bit enables/disables the cache. 0 - Cache disabled 1 - Cache enabled

16.6.1.1.2 Cache line control register (CLCR)

This register defines cache line commands to be performed using a specific cache line address or a physical address.

If a physical address is specified, both ways of the cache are searched, and the command is only performed on the way which hits.

The LCWAY, LCIMB and LCIVB bits are status bits that are read/write. These bits can be written, but written values to these bits have no functional purpose.

Offset

Register	Offset
CLCR	804h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				LACC	LADSEL	LCMD		Reserved	LCWAY	LCIMB	LCIVB	0			TDSEL
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	0		CACHEADDR										0	LGO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-28 —	Reserved
27 LACC	Line access type When doing a search command, this bit specifies if a read or write operation should be performed after a hit. **NOTE** Only used for search commands (LCMD =00).

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - Read 1 - Write
26 LADSEL	Line Address Select When using the cache address, the way must also be specified in CLCR[WSEL]. When using the physical address, both ways are searched and the command is performed only if a hit. 0 - Cache address. WSEL and CACHEADDR bitfields specify the address. 1 - Physical address. CSAR register specifies the address.
25-24 LCMD	Line Command 00 - Search and read or write. LACC bit determines if operation is a read or write. 01 - Invalidate 10 - Push 11 - Clear
23 —	Reserved
22 LCWAY	Line Command Way Indicates the way used by the line command. Only applies if valid bit LCIVB = 1. 0 - Way 0 1 - Way 1
21 LCIMB	Line Command Initial Modified Bit If command used cache address and way, then this bit shows the initial state of the modified bit If command used physical address and a hit, then this bit shows the initial state of the modified bit. If a miss, this bit reads zero. 0 - If command uses Cache Address and Way, then the initial state of the modified bit is 0. If command uses a Physical Address and is a hit, then the initial state of the modified bit is 0. If command uses a Physical Address and is a miss, then this bit reads 0. 1 - If command uses Cache Address and Way, then the initial state of the modified bit is 1. If command uses a Physical Address and is a hit, then the initial state of the modified bit is 1.
20 LCIVB	Line Command Initial Valid Bit If command used cache address and way, then this bit shows the initial state of the valid bit If command used physical address and a hit, then this bit shows the initial state of the valid bit. If a miss, this bit reads zero.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - If command uses Cache Address and Way, then the initial state of the valid bit is 0. If command uses a Physical Address and is a hit, then the initial state of the valid bit is 0. If command uses a Physical Address and is a miss, then this bit reads 0. 1 - If command uses Cache Address and Way, then the initial state of the valid bit is 1. If command uses a Physical Address and is a hit, then the initial state of the valid bit is 1.
19-17 —	Reserved
16 TDSEL	Tag/Data Select Selects tag or data for search and read or write commands. 0 - Data 1 - Tag
15 —	Reserved
14 WSEL	Way select Selects the way for line commands. 0 - Way 0 1 - Way 1
13-12 —	Reserved
11-2 CACHEADDR	Cache address CLCR [11:5] bits are used to access the tag arrays CLCR [11:3] bits are used to access the data arrays CLCR[2] bit is needed to decide which half of the 64 bits of data to put in the Cache read/write value registers (CCVR)
1 —	Reserved
0 LGO	Initiate Cache Line Command Setting this bit initiates the cache line command indicated by bits 27-24. Reading this bit indicates if a line command is active **NOTE** This bit stays set until the command completes. Writing zero has no effect.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	NOTE This bit is shared with CSAR[LGO]. Either register address bit[0] can be written interchangeably. This saves xcache writes (helps performance) when doing cache operations. Cache line operations can be started by writing to CLCR with the LGO bit asserted or the cache search address can be changed and a search started by writing to CSAR with the LGO bit asserted. 0 - Write: no effect. Read: no line command active. 1 - Write: initiate line command indicated by bits 27-24. Read: line command active.

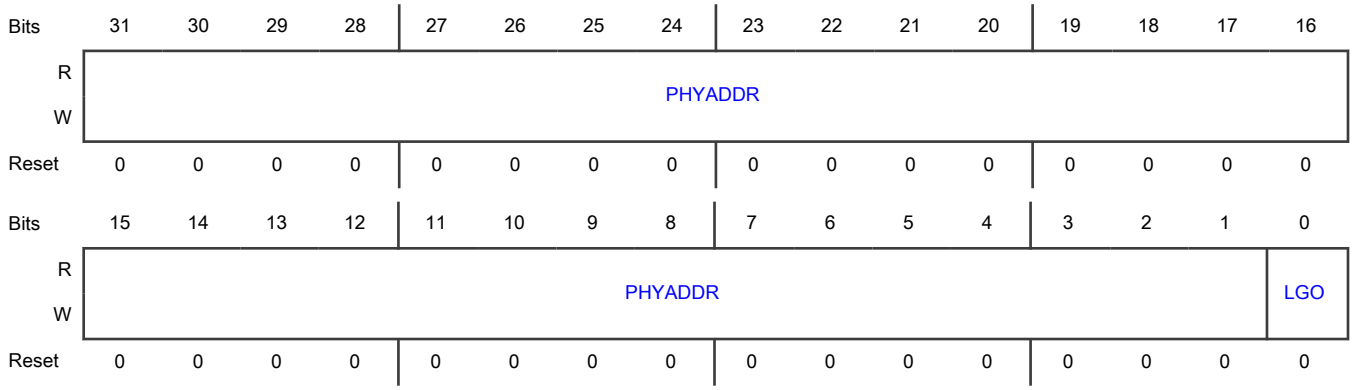
16.6.1.1.3 Cache search address register (CSAR)

The CSAR register is used to define the physical address for cache line commands when the CLCR[LADSEL] bit is set.

Offset

Register	Offset
CSAR	808h

Diagram



Fields

Field	Description
31-1 PHYADDR	Physical Address PHYADDR represents bits [31:1] of the system address. CSAR [31:12] bits are used for tag compare CSAR [11:5] bits are used to access the tag arrays CSAR [11:3] bits are used to access the data arrays

Table continues on the next page...

Table continued from the previous page...

Field	Description
	CSAR[2] bit is needed to decide which half of the 64 bits of data to put in the Cache read/write value registers (CCVR) CSAR[1] bit is not used. Writes to this bit are allowed, and the read value will be whatever value was previously written. This bit is is not functionally used.
0 LGO	Initiate Cache Line Command Setting this bit initiates the cache line command indicated by CLCR bits 27-24. Reading this bit indicates if a line command is active **NOTE** This bit stays set until the command completes. Writing zero has no effect. **NOTE** This bit is shared with CLCR[LGO] 0 - Write: no effect. Read: no line command active. 1 - Write: initiate line command indicated by bits CLCR[27:24]. Read: line command active.

16.6.1.1.4 Cache read/write value register (CCVR)

The CCVR register is used to source write data or return read data when a search line command is requested (CLCR[LCMD] = 00). CLCR[LACC] determines if the access is a read or write, and CLCR[TDSEL] determines if the access is to the data or tag arrays.

Offset

Register	Offset
CCVR	80Ch

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	Description
31-0 DATA	Cache read/write Data For tag search, read or write: • CCVR [31:12] bits are used for tag array R/W value • CCVR [11:5] bits are used for tag set address on reads; unused on writes • CCVR[4:2] bits are reserved • CCVR[1] tag modify bit • CCVR[0] tag valid bit For data search, read or write: • CCVR[31:0] bits are used for data array R/W value

Chapter 17
AHB Low Power Cache Controller (AHB_LPCAC)

17.1 Chip-specific LPCAC information

Table 51. Reference links to related information

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

LPCAC is 8 KB of cache to accelerate accesses to the internal flash.

17.1.1 Cache control and status

In this device, cache control and status are provided from outside the LPCAC through SYSCON register LPCAC_CTRL.

17.2 Overview

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

AHB_LPCAC has a 32-bit datapath AMBA-AHB input bus and a 32-bit datapath AMBA-AHB output bus.

17.2.1 Block diagram

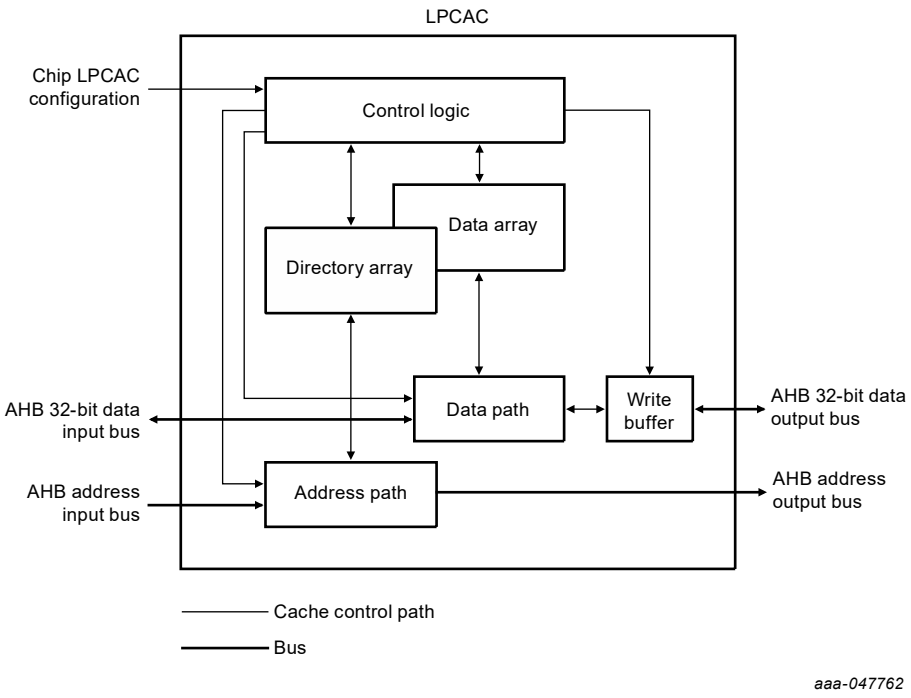


Figure 30. Block diagram

17.2.2 Features

AHB_LPCAC supports the following features:

- Nonblocking and write-through cache mode
- 8 KB total cache size
- Cache organization as an 8-way, 4-set-associative design based on 256-byte superpages
- Each way or superpage contains up to 16 sequential 16-byte pages

17.3 Functional description

This section provides further information on the AHB_LPCAC module operation.

A reset signal clears and enables this module.

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

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

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

17.3.1 Cache functional description

The AHB_LPCAC cache is an 8-way, 4-set-associative, 256-byte per line write-through design. It supports a total cache capacity of 8 KB for a 32-bit wide cache miss datapath.

17.3.1.1 Cache controls

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

The cache control inputs are as follows:

- clr_lpcac: clear lpcac
- dis_lpcac: disable lpcac
- frc_no_alloc: force no allocation
- mode_ctl_hprot: ignore AHB_LPCAC Memory Regions

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

17.3.2 Clocks

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

17.3.3 Reset

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

17.3.4 Interrupts

This module has no interrupts.

17.4 Signal descriptions

This module has no external signals.

17.5 Memory regions and input control description

17.5.1 Memory regions

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

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

The following table defines the memory map.

Table 52. Memory map

Cache memory range	Cache memory range	Cache total memory address space
1D00_0000h–1D03_FFFFh (256 KB)	2D00_0000h–2D03_FFFFh (256 KB)	512 KB

17.5.2 Input controls

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

Table 53. Control inputs for AHB_LPCAC

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

Chapter 18
CACHE64 Policy Select (POLSEL)

18.1 Chip-specific POLSEL information

Table 54. Reference links to related information

Topic	Related module	Reference
Full description	POLSEL	POLSEL
System memory map		System memory map
Clocking		Clock generation
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

18.1.1 Chip-specific CACHE64 POLSEL information

Each of the CACHE64 modules interface to a FLEXSPI octal/quad SPI module for access to off-chip memory. The 128 Mbyte space divided into three policy regions, is allocated for each CACHE64 POLSEL interface. POLICY_SEL register has three 2-bit fields associated with the three defined regions. Each field designates the policy to be applied to addresses falling within that region. The three policies are completely independent so, it is possible to assign the same policy to all three regions.

Name	Default value	Description
APB_ADDR_WIDTH	12	APB Address bus width.
APB_DATA_WIDTH	32	APB Data bus width.
BASE_ADDR	32'h000	Memory Map base address position for this IP.
CACHE64_SIZE	32	Cache64 Size limit in KByte.
CACHE64_BPW	32	Cache64 bits per word.

18.2 Overview

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

18.2.1 Features

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

18.3 Functional description

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

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

18.3.1 Region mapping

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

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

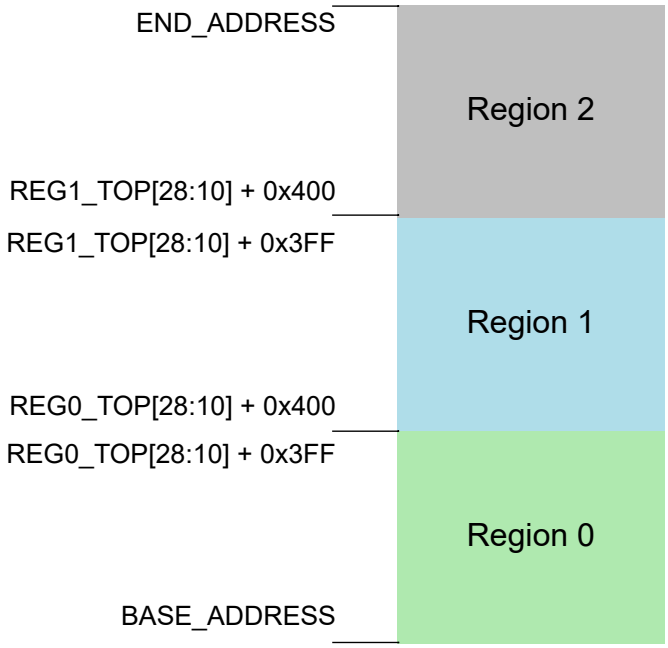


Figure 31. Cache memory regions

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

18.3.2 Region priority

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

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

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

NOTE

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

18.4 Application information

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

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

The following register settings are used to achieve this configuration:

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

The following figure shows the resulting memory configuration.

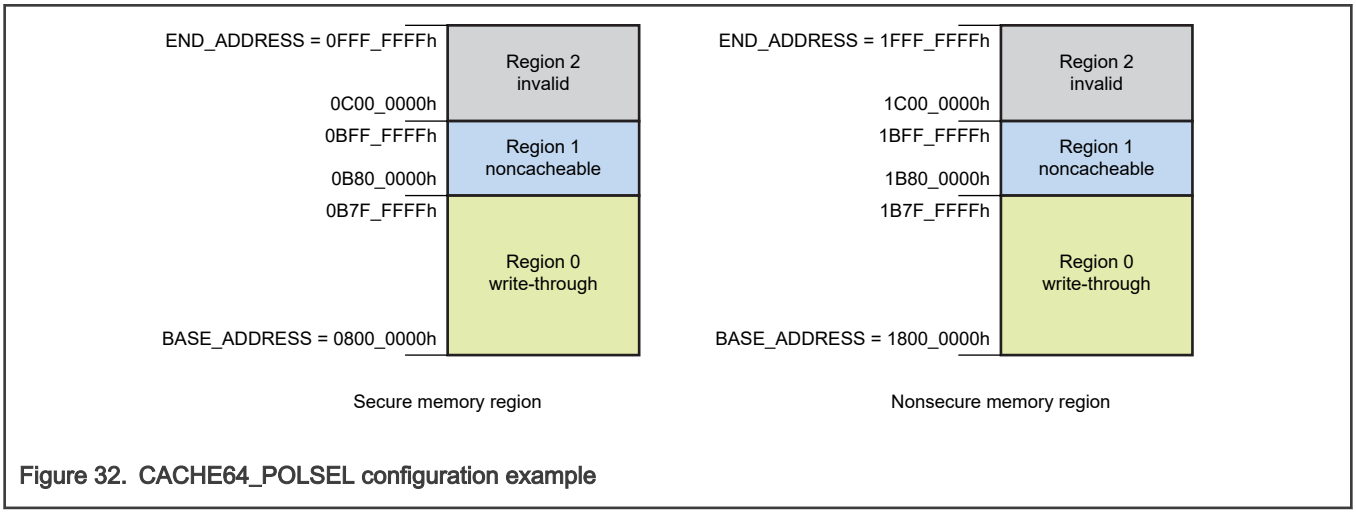


Figure 32. CACHE64_POLSEL configuration example

18.5 Memory map and register definition

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

18.5.1 CACHE64_POLSEL register descriptions

18.5.1.1 CACHE64_POLSEL memory map

CACHE64_POLSEL0 base address: 4002_E000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
14	Region 0 Top Boundary (REG0_TOP)	32	See section	02AA_A800
18	Region 1 Top Boundary (REG1_TOP)	32	See section	0555_5400
1C	Policy Select (POLSEL)	32	See section	0000_0000

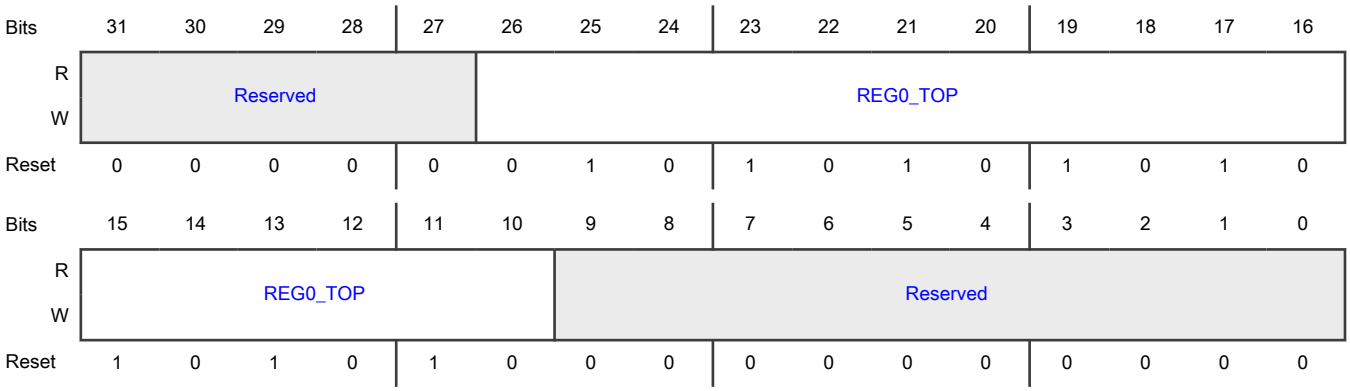
18.5.1.1.1 Region 0 Top Boundary (REG0_TOP)

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

Offset

Register	Offset
REG0_TOP	14h

Diagram



Fields

Field	Description
31-27 —	Reserved
26-10 REG0_TOP	Upper Limit Of Region 0 Provides bits 26-10 of the region 0 compare address. The value provided here is concatenated with 3FFh to create the full compare address. For Example, if REG0_TOP[26:10] = 0b00000000000_111111, the region 0 end address is BASE_ADDRESS + 0000_FFFFh (64 KB).

Table continues on the next page...

Table continued from the previous page...

Field	Description
9-0 —	Reserved

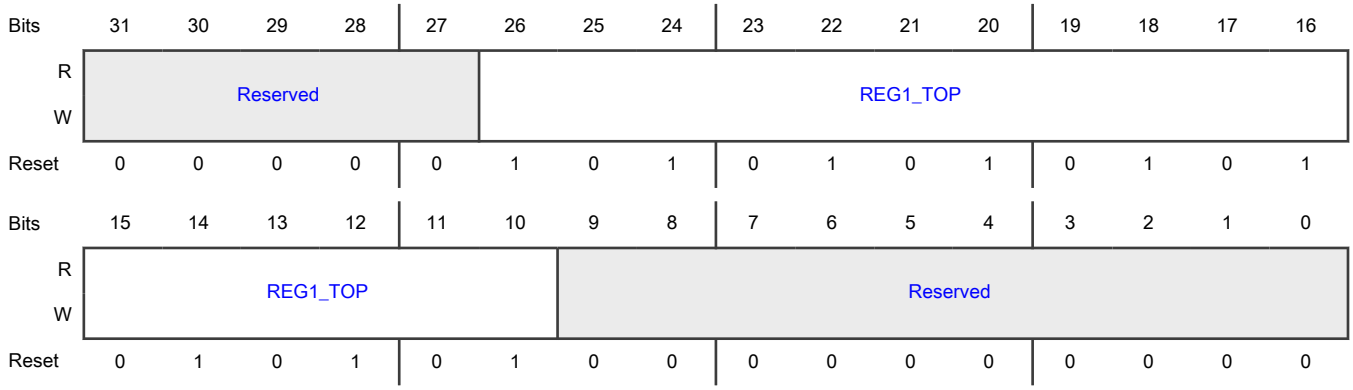
18.5.1.1.2 Region 1 Top Boundary (REG1_TOP)

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

Offset

Register	Offset
REG1_TOP	18h

Diagram



Fields

Field	Description
31-27 —	Reserved
26-10 REG1_TOP	Upper Limit Of Region 1 Provides bits 26-10 of the region 1 compare address. The value provided here is concatenated with 3FFh to create the full compare address. For example, if REG1_TOP[26:10] = 0b00000000000_111111, the region 0 end address is BASE_ADDRESS + 0000_FFFFh (64 KB).
9-0 —	Reserved

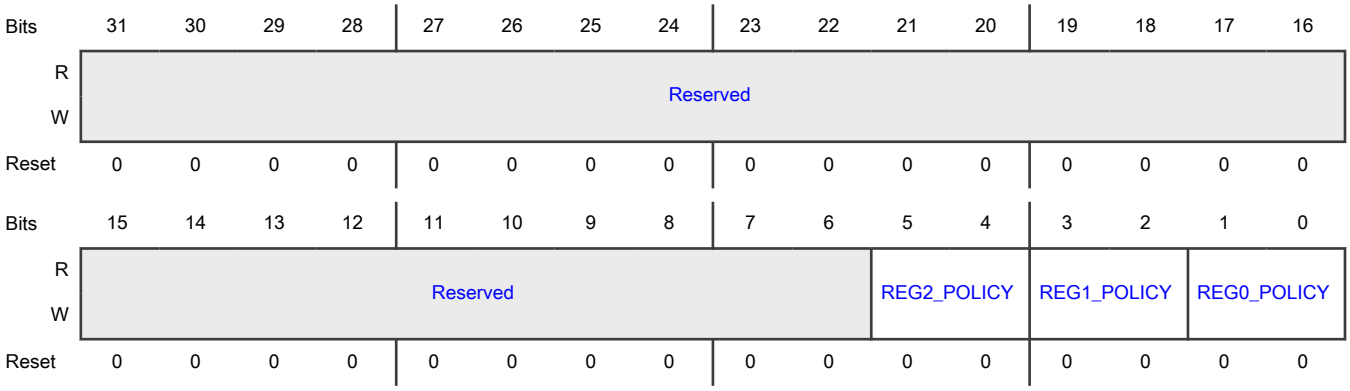
18.5.1.1.3 Policy Select (POLSEL)

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

Offset

Register	Offset
POLSEL	1Ch

Diagram



Fields

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

Table continues on the next page...

Table continued from the previous page...

Field	Description
	10 - Write-back 11 - Invalid

Chapter 19
FlexSPI

19.1 Chip-specific FLEXSPI information

Table 55. Reference links to related information

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

19.1.1 Module instances

This device has one instance of the FLEXSPI module, FLEXSPI0. FlexSPI0 has an AHB RX Buffer size of 1KB. FlexSPI0 supports Port A only. Port A can be connected to 2 memory devices. The FlexSPI interface supports HyperFlash, HyperRAM and Xccela memory types, among others.The FlexSPI interface includes a 8 KB cache with CACHE64 controller, and programmable cache policies are supported using POLSEL.

19.1.2 Pin name mapping

Below table maps the signal name in FlexSPI with the pin names in the chip.

FlexSPI signal	SoC signal
A_SS0_N	FLEXSPI0_SS0_B
A_SS1_N	FLEXSPI0_SS1_B
A_SCLK	FLEXSPI0_SCLK
A_DATAn	FLEXSPI0_DATAn
A_DQS	FLEXSPI0_DQS

19.1.3 Master ID Assignments

Table 56. FlexSPI Master ID Assignments

Master	Bus	Master Port	Master ID
CPU0	C-AHB bus	M0	0000
CPU0	S-AHB bus	M1	0001
SDMA0		M2	0010
SDMA1		M3	0011
Powerquad		M10	1010

19.1.4 Initialization

The FlexSPI interface supports HyperFlash, HyperRAM and Xccela memory types, among others. The FlexSPI interface includes a 8 KB cache with CACHE64 controller, and programmable cache policies are supported using POLSEL.

FlexSPI controller initialization sequence is as following:

- Power the FlexSPI SRAM using PDRUNCFG registers.
- Release FlexSPI from reset using PRSTCRL registers.

See Initialization for more details on block level initialization.

19.2 Overview

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

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

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

NOTE

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

Table 57. Updated terms

Updated term	Deprecated term
Controller	Master
Target	Slave

19.2.1 Block diagram

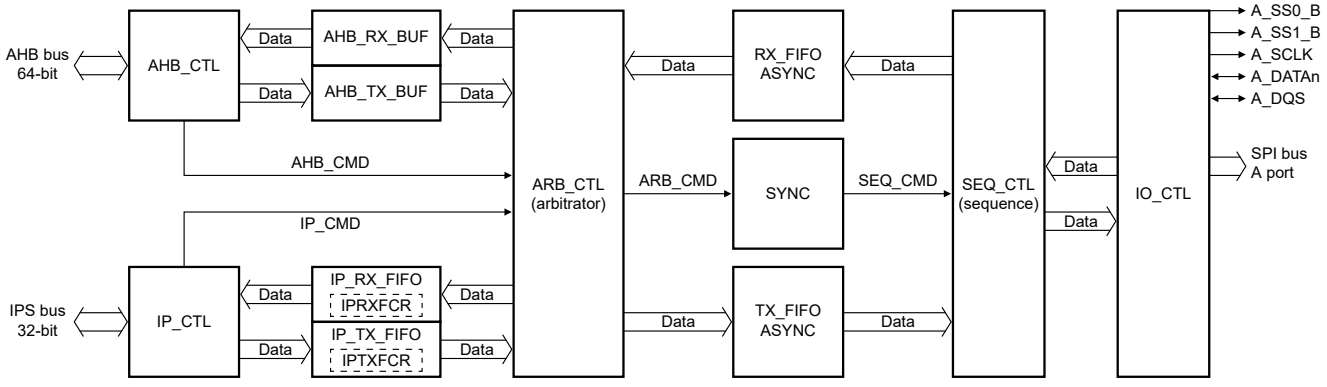


Figure 33. Block diagram

19.2.2 Features

FlexSPI supports:

- Flexible sequence engine (lookup table) to support various vendor devices

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

NOTE

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

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

19.3 Functional description

19.3.1 Flash connection

There is one FlexSPI interface port (port A) that supports up to two flash devices by providing two chip select outputs.

NOTE

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

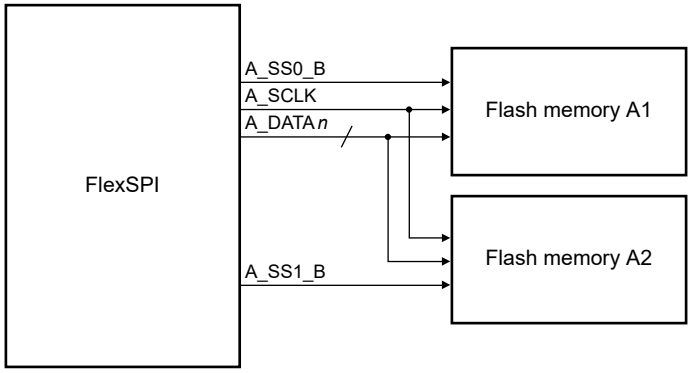


Figure 34. FlexSPI connection diagram with two devices

NOTE

- Flash A1 and Flash A2 can be two flash chip packages or two flash dies in the same package.
- In individual mode, flash devices cannot be accessed at the same time. But these two devices can be accessed separately.

19.3.2 Flash memory access mode

19.3.2.1 SPI clock mode

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

19.3.2.2 Individual mode

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

NOTE

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

19.3.2.3 SDR mode and DDR mode

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

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

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

19.3.2.4 Single, Dual, Quad, and Octal mode

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

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

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

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

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

19.3.3 Modes of operation

FlexSPI operates with these modes.

Table 58. Operating modes

Mode	Description
Module Disable	This mode is a low-power mode for FlexSPI. In Module Disable mode, the AHB clock and serial clock domains are gated off internally, but the IPS bus clock domain is not gated off. Read and write access to the control and status register are available, but the LUT, IP receive FIFO, and IP transmit FIFO cannot be accessed. Serial memory access is also not available. Write 1 to MCR0[MDIS] to enter this mode. Write 0 to MCR0[MDIS] to exit this mode.
Doze	This mode is a low-power mode for the chip. When the chip requires FlexSPI to enter Doze mode and MCR0[DOZEEN] = 1, FlexSPI enters Doze mode after all transactions are completed (STS0[ARBIDLE] = 1). In Doze mode, the AHB clock and serial clock domains are gated off internally, but the IPS bus clock is not gated off. Read and write access to the control and status register are available, but the LUT, IP receive FIFO, the IP transmit FIFO cannot be accessed. Serial memory access is also not available. This mode is entered via system request, and exited by deasserting this system request.
Stop	This mode is a low-power mode for the chip. When the chip requires the FlexSPI to enter Stop mode, FlexSPI waits for all transactions to complete (STS0[ARBIDLE] = 1) and return ACK handshake to the system. After the ACK handshake is returned, FlexSPI gates off the AHB clock and serial clock domains internally. The system can gate off the AHB bus clock, IPS bus clock, and serial clock at the system level. This mode is entered via system request. This mode is exited by deasserting this system request, and the ACK handshake message is also deasserted immediately.
Normal	No clock is gated off internally. Normal register access and serial memory access are available.

19.3.4 AHB access memory map

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

NOTE

The maximum flash size supported for each device is 128 MB. The maximum total flash size supported (for all four devices) is also 128 MB.

When MCR2[SAMEDEVICEEN] = 1:

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

When MCR2[SAMEDEVICEEN] = 0:

- FA1_SIZE = FLSHA1CR0[FLSHSZ] × 1 KB
- FA2_SIZE = FLSHA2CR0[FLSHSZ] × 1 KB

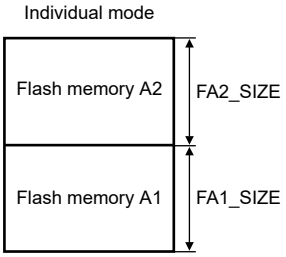


Figure 35. AHB access memory map in individual mode

AHB access memory map in individual mode:

- Flash A1 address range: 0000_0000h to FA1_SIZE
- Flash A2 address range: FA1_SIZE to (FA1_SIZE + FA2_SIZE)

NOTE

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

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

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

NOTE

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

Table 59. FlexSPI address translation

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

NOTE

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

19.3.4.2 Flash address sent to flash memory devices

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

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

For word-addressable flash devices, the last bit of the address is not needed, because flash memory is read and programmed in terms of two bytes.

19.3.5 Lookup table (LUT)

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

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

Figure 36 shows the LUT structures, sequences, and instructions.

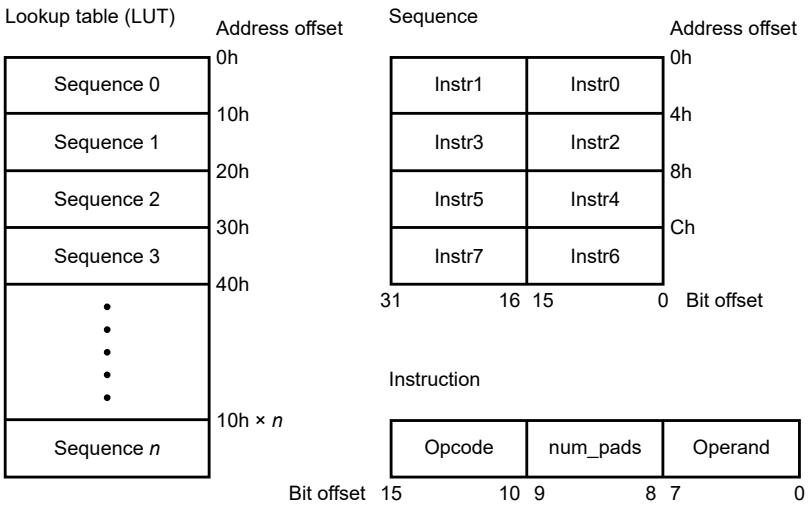


Figure 36. LUT structure, sequences, and instructions

NOTE

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

For IP and AHB write commands, the FlexSPI controller always executes from instruction pointer 0. For AHB read commands, the FlexSPI controller executes from a saved instruction start pointer. The FlexSPI controller saves the instruction start pointers separately for each flash device. All these saved instruction pointers are zero before the JMP_ON_CS instruction is executed.

When the JMP_ON_CS instruction is executed, the operand in the JMP_ON_CS instruction is saved as instruction start pointer. See Execute-In-Place (XIP) enhanced mode.

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

To lock the LUT:

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

To unlock the LUT:

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

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

19.3.6 Programmable sequence engine

The FlexSPI controller implements a programmable sequence engine that executes the sequence from the LUT. The FlexSPI controller executes the instructions sequentially, and generates flash transactions on the SPI interface accordingly. Table 60 is a complete list of the supported instructions.

Table 60. Instruction set

Name	Opcode	num_pads	Action on SPI interface	Transmit data	Bits/bytes/cycle number
CMD_SDR	00_0001h	00h - one pad (Single mode) 01h - two pads (Dual mode) 02h - four pads (Quad mode) 03h - eight pads (Octal mode)	Transmit command code to flash memory	Command code: Operand[7:0]	Bit number: 8
CMD_DDR	00_0021h				
RADDR_SDR	00_0002h		Transmit row address to flash memory See Flash address sent to flash memory devices	Row_Address[31:0] Row_Address comes from AHB bus (AHB command) or IPCR0[SFAR] (IP command).	Bit number: operand[7:0] Value of operand determines number of bits sent in Row_Address.
RADDR_DDR	00_0022h				
CADDR_SDR	00_0003h		Transmit column address to flash memory. See Flash address sent to flash memory devices.	Column_Address[31:0] Column_Address comes from AHB bus (AHB command) or IPCR0[SFAR] (IP command).	Bit number: operand[7:0] Value of operand determines number of bits sent in Column_Address.
CADDR_DDR	00_0023h				
MODE1_SDR	00_0004h		Transmit mode bits to flash memory	Mode bits: Operand[0]	Bit number: 1
MODE1_DDR	00_0024h				

Table continues on the next page...

Table 60. Instruction set (continued)

Name	Opcode	num_pads	Action on SPI interface	Transmit data	Bits/bytes/cycle number
MODE2_SDR	00_0005h			Mode bits: Operand[1:0]	Bit number: 2
MODE2_DDR	00_0025h				
MODE4_SDR	00_0006h			Mode bits: Operand[3:0]	Bit number: 4
MODE4_DDR	00_0026h				
MODE8_SDR	00_0007h			Mode bits: Operand[7:0]	Bit number: 8
MODE8_DDR	00_0027h				
WRITE_SDR	00_0008h		Transmit program data to flash device	Program data in IP_TX_FIFO or AHB_TX_BUF	AHB burst size and burst type (AHB command) or IPCR1[IDATSZ] (IP command) determines byte number (data size). For details about flash read or program data size, see Flash memory access via AHB command and Flash memory access via IP command.
WRITE_DDR	00_0028h				
READ_SDR	00_0009h		Receive read data from flash device Read data is put into AHB_RX_BUF or IP_RX_FIFO.	-	
READ_DDR	00_0029h				
DATSZ_SDR	00_000Bh		Transmit read or program data size (byte number) to flash device	Internal logic Read or program data size for current command sequence	Bit number: operand[7:0] Never set operand to zero or greater than 64 for DATSZ instruction.
DATSZ_DDR	00_002Bh				
DUMMY_SDR	00_000Ch		Leave data lines undriven by FlexSPI controller. Turnaround cycles are provided from host driving to device driving. num_pads determines number of pads in input mode.	-	Dummy cycle number (in serial root clock): Operand[7:0] Dummy cycle (N), described in data sheet of flash device, is in number of SCLK cycles. This number may be configurable. In SDR mode, SCLK cycle is same as serial root clock. Operand value must be set to N. In DDR mode, SCLK cycle is double serial root clock cycle. Operand value must be set to 2N, 2N-1 or 2N+1 depending on definition of dummy in data sheet of flash device. See Flash memory access sequence examples and dummy cycle
DUMMY_DDR	00_002Ch				

Table continues on the next page...

Table 60. Instruction set (continued)

Name	Opcode	num_pads	Action on SPI interface	Transmit data	Bits/bytes/cycle number
					definition in data sheet of external memory.
DUMMY_RWDS_SDR	00_000Dh		Similar to DUMMY_SDR/ DUMMY_DDR instruction. Difference is in dummy cycle number. DQS pin is called RWDS in HyperBus specification. See Dummy instruction for details. **Set operand to "Latency count" for HyperBus devices.**	-	For read command, dummy cycle number (in serial root clock): (operand[7:0] × 4 - 1) if RWDS (DQS pin) is high; (operand[7:0] × 2 - 1) if RWDS (DQS pin) is low; For write command, dummy cycle number (in serial root clock): (operand[7:0] × 4 - 2) if RWDS (DQS pin) is high; (operand[7:0] × 2 - 2) if RWDS (DQS pin) is low;
DUMMY_RWDS_DDR	00_002Dh				
LEARN_SDR	00_000Ah	00h - one pad (Single mode) 01h - two pads (Dual mode) 02h - four pads (Quad mode) 03h - eight pads (Octal mode)	Receive read data or preamble bit from flash device FlexSPI controller compares data line bits with the DLPR register to determine a correct sampling clock phase.	-	Bit number: operand[7:0] Never set operand to zero for LEARN instruction. Value of operand indicates number of bits to receive and compare to DLPR value. For example, 8-bit pattern 5Ah on each data line must set operand to 8.
LEARN_DDR	00_002Ah				
JMP_ON_CS	00_001Fh	Num_pads setting ignored.	Stop execution, deassert CS and save operand[7:0] as instruction start pointer for next sequence. Normally this instruction is used to support Execute-In-Place enhanced mode. See Execute-In-Place (XIP) enhanced mode. This instruction is only allowed for	-	No transaction on SPI interface.

Table continues on the next page...

Table 60. Instruction set (continued)

Name	Opcode	num_pads	Action on SPI interface	Transmit data	Bits/bytes/cycle number
			AHB read commands. When using this instruction in IP command or AHB write command, interrupt status flag set (INTR[IPCMDERR] or INTR[AHBCMDERR]).		
STOP	00_0000h		Stop execution and deassert CS. Next command sequence (to same flash device) starts from instruction pointer 0.	-	

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

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

19.3.6.1 Executing instructions on SPI interface

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

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

This order is shown in Figure 37.

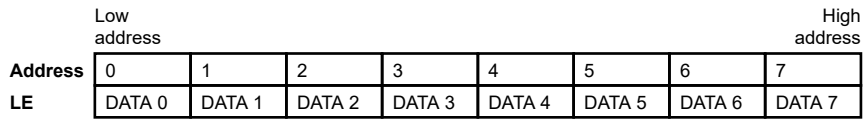


Figure 37. Bus bit order

Table 61. SPI instructions

Instructions	Description
Command (CMD_SDR, CMD_DDR)	Normally used to transmit a command code to the external device. The command code is the 8-bit operand in the instructions. See Flash memory access sequence examples.
Address (RADDR_SDR, RADDR_DDR, CADDR_SDR, CADDR_DDR)	Normally used to send the flash memory access start address (row address or column address) to an external device. FlexSPI determines the bits of the row address or column address according to the AHB access address or IP command address. See Flash address sent to flash memory devices. The operand value in the instruction code represents the number of address bits to send. For example, when NUM_PADSn = 8, a memory reading the flash memory address on four SCK edges requires the sum of bits in CADDR and RADDR instructions to be 32. Otherwise, FlexSPI does not generate four SCK edges of address phase. See Flash memory access sequence examples.
Mode (MODEx_SDR, MODEx_DDR)	Normally used to send mode bits to external devices. Mode bits are the lower bits of an operand. For example, the bit number is 1 for MODE1_SDR and MODE1_DDR, 2 for MODE2_SDR and MODE2_DDR, 4 for MODE4_SDR and MODE4_DDR, and 8 for MODE8_SDR and MODE8_DDR. The pad number should not be greater than the mode bit number. For example, you cannot set NUM_PADSn to 11b (Octal mode) for MODE4_* instructions. See Flash memory access sequence examples for more details.
Data Size (DATSZ_SDR, DATSZ_DDR)	Used to send the size of program data or read data (byte number) to external devices. This instruction is normally used in FPGA applications where the memory space in external device acts like a FIFO. The external device requires data size information to determine how much data are popped from or pushed into the internal FIFO. The operation value in the data size instructions is the bit number. See Flash memory access sequence examples.
Write (WRITE_SDR, WRITE_DDR)	Normally used to send program data to external device. Programming data is fetched from IP_TX_FIFO (IP Command) or AHB_TX_Buffer (AHB command). For details about flash program data size, see Flash memory access via AHB command and Flash memory access via IP command. The byte order for program data is always from low to high. See Flash memory access sequence examples for more details.
Read (READ_SDR, READ_DDR)	Normally used to receive flash memory data from external devices. Received data is put into IP_RX_FIFO (IP Command) or AHB_RX_Buffer (AHB command). For information about flash memory read data size, see Flash memory access via AHB command and Flash memory access via IP command. The byte order for reading data is always from low to high. See Flash memory access sequence examples.
Dummy (DUMMY_SDR, DUMMY_DDR, DUMMY_RWDS_SDR, DUMMY_RWDS_DDR)	Used to provide turnaround cycles on the SPI interface. During dummy instruction, the FlexSPI controller and external devices do not drive the SPI interface. See Programmable sequence engine for details about dummy cycle number. DUMMY_RWDS_DDR can be used for a HyperBus device that uses RWDS pin to indicate whether extra latency is needed. DUMMY_RWDS_SDR is reserved. The FlexSPI controller checks the DQS pin input level at the fourth cycle after SCLK output toggling is enabled. The DQS pin is called RWDS in HyperBus specification. See Flash memory access sequence examples.

Table continues on the next page...

Table 61. SPI instructions (continued)

Instructions	Description
	NOTE FlexSPI releases the bus after at least one cycle. To avoid data contention, the NUM_PADSn value for the dummy commands should be configured to match the number of data lines used for the external memory.
Learn (LEARN_SDR, LEARN_DDR)	Used to determine the correct sample clock phase for flash memory read data sampling. External device drives read data (or data learning pattern) bits on the FlexSPI interface. The FlexSPI controller compares the data line bits to the DLPR register to determine a correct sampling clock phase. Clock phase selection is automatically updated after executing learn instructions. See Data learning for more details. The operand value in learn instructions is the byte number. FlexSPI checks the same data pattern on each data line. The byte order is byte 0, byte 1, byte 2, byte 3, byte0, byte 1, and so on. • Byte 0 is DLPR register bits 7–0. • Byte 1 is DLPR register bits 15–8. • Byte 2 is DLPR register bits 23–16. • Byte 3 is DLPR register bit 31–24. The bit order is from high to low in each byte. See Flash memory access sequence examples.

19.3.6.2 Flash memory access sequence examples

Diagrams below all assume MCR0[SERCLKDIV] = 0.

Figure 38 shows an example SDR single I/O read sequence (Cypress Serial Nor Flash S25FS512S) in individual mode.

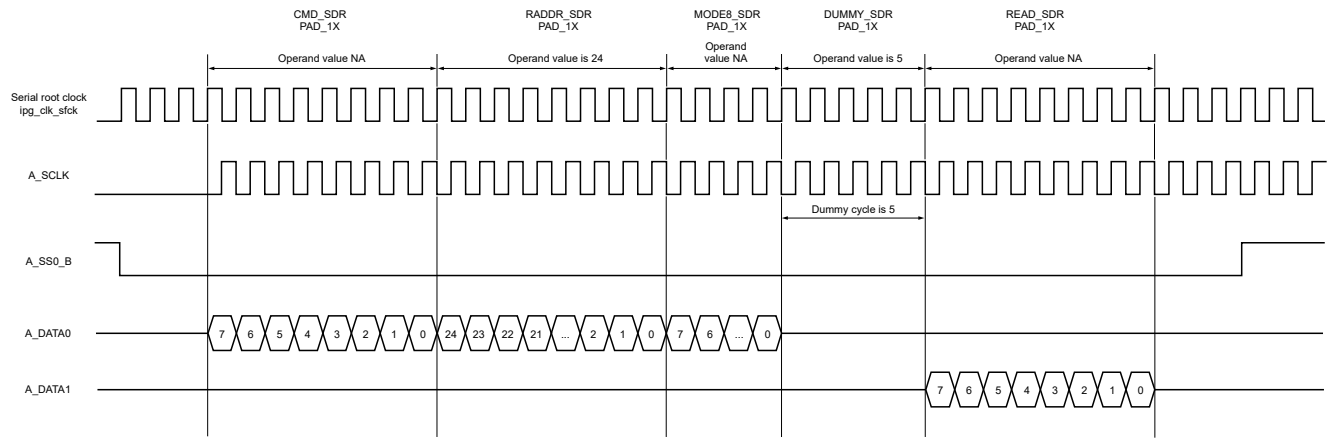


Figure 38. Flash memory access sequence example (SDR single I/O read sequence)

NOTE

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

Figure 39 shows an example SDR quad I/O read sequence (Cypress Serial Nor Flash S25FS512S) in individual mode.

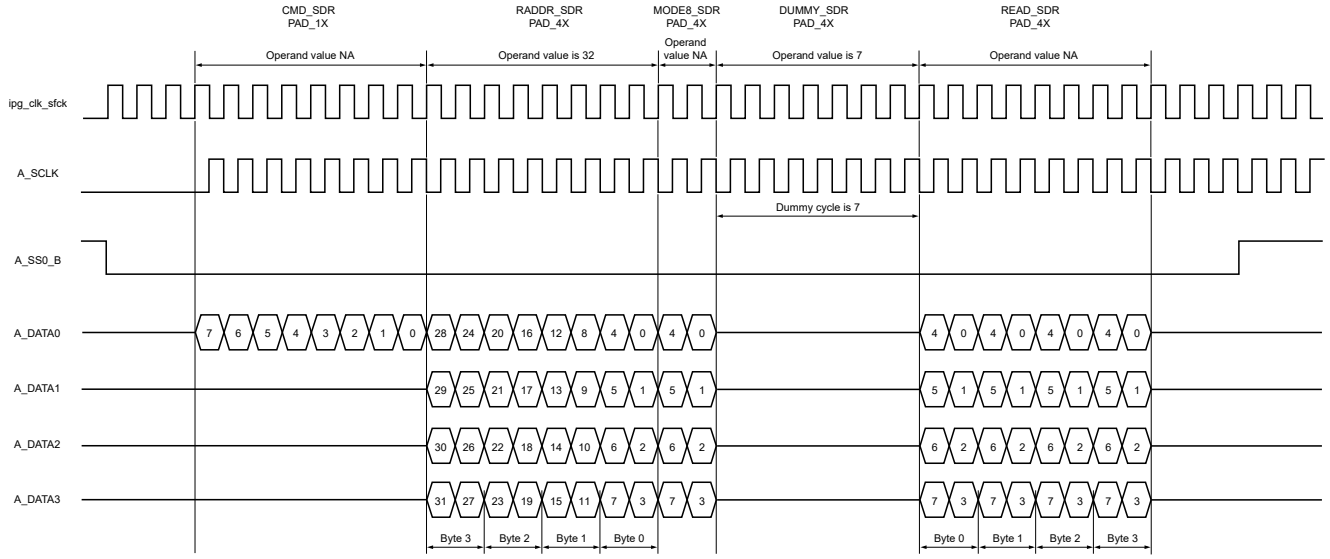


Figure 39. Flash memory access sequence example (SDR quad I/O read sequence)

NOTE

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

Figure 40 shows an example learn instruction (not for a specified flash device) in individual mode.

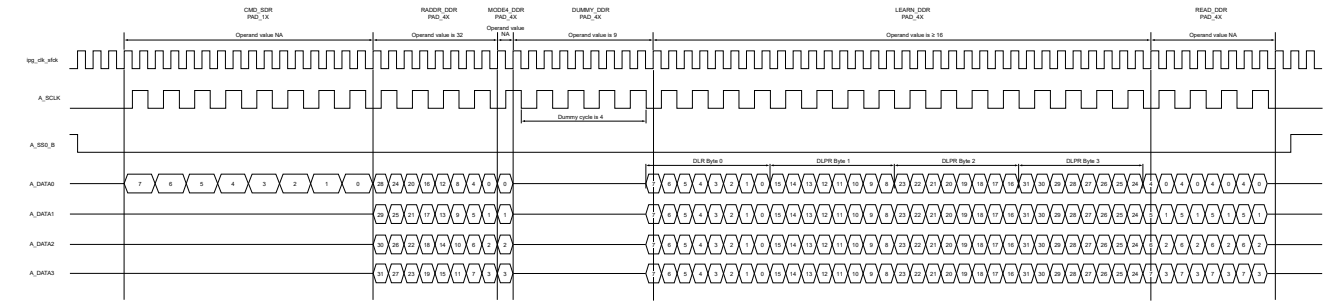


Figure 40. Flash memory access sequence example (data learning)

NOTE

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

Figure 41 shows an example HyperBus device read transaction (single latency count) in individual mode.

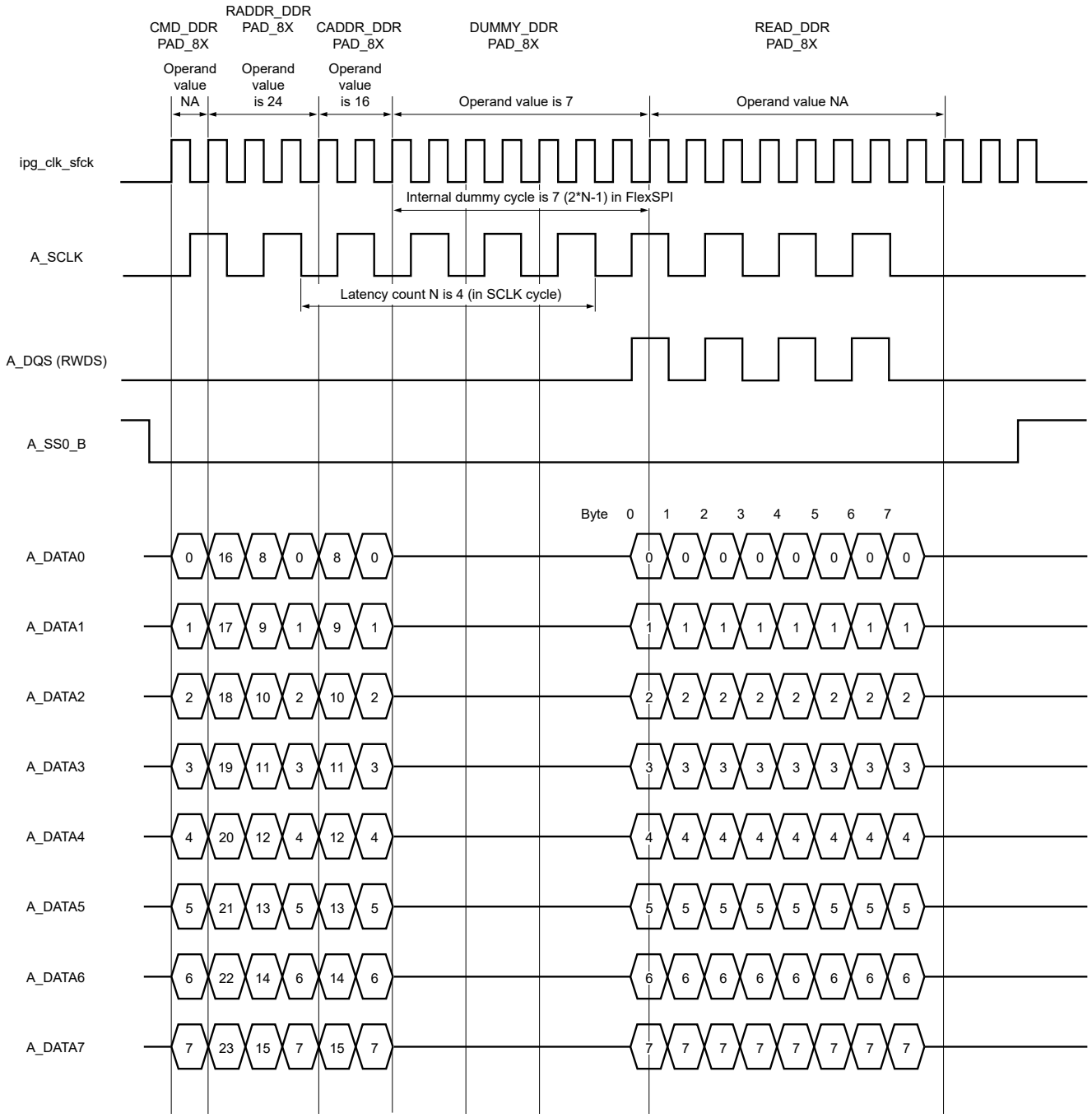


Figure 41. HyperBus device read transaction with single latency count

NOTE

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

Figure 42 shows an example HyperBus device read transaction (additional latency count) in individual mode.

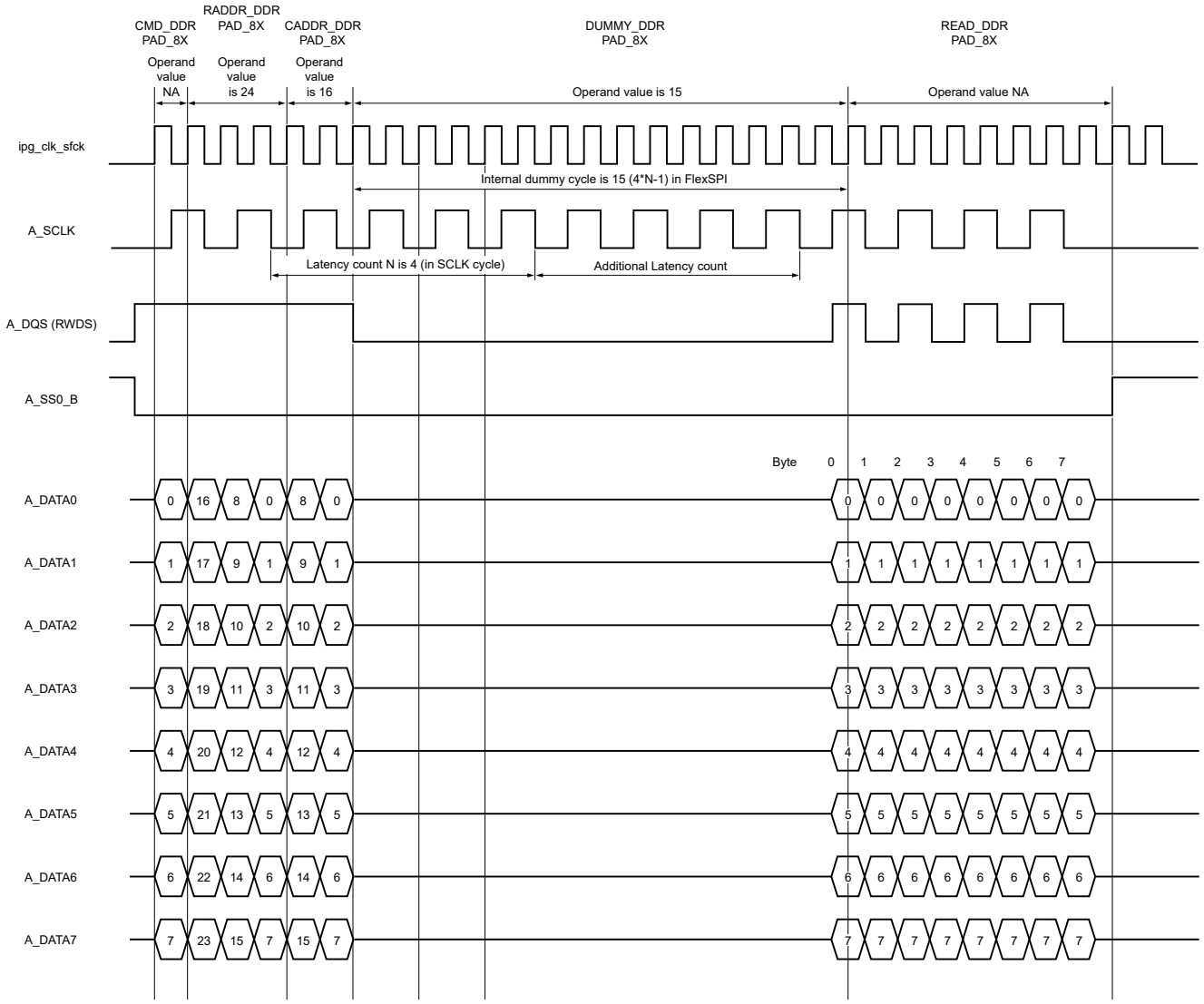


Figure 42. HyperBus device read transaction with additional latency count

Figure 43 shows an example HyperBus device write transaction (single latency count) in individual mode.

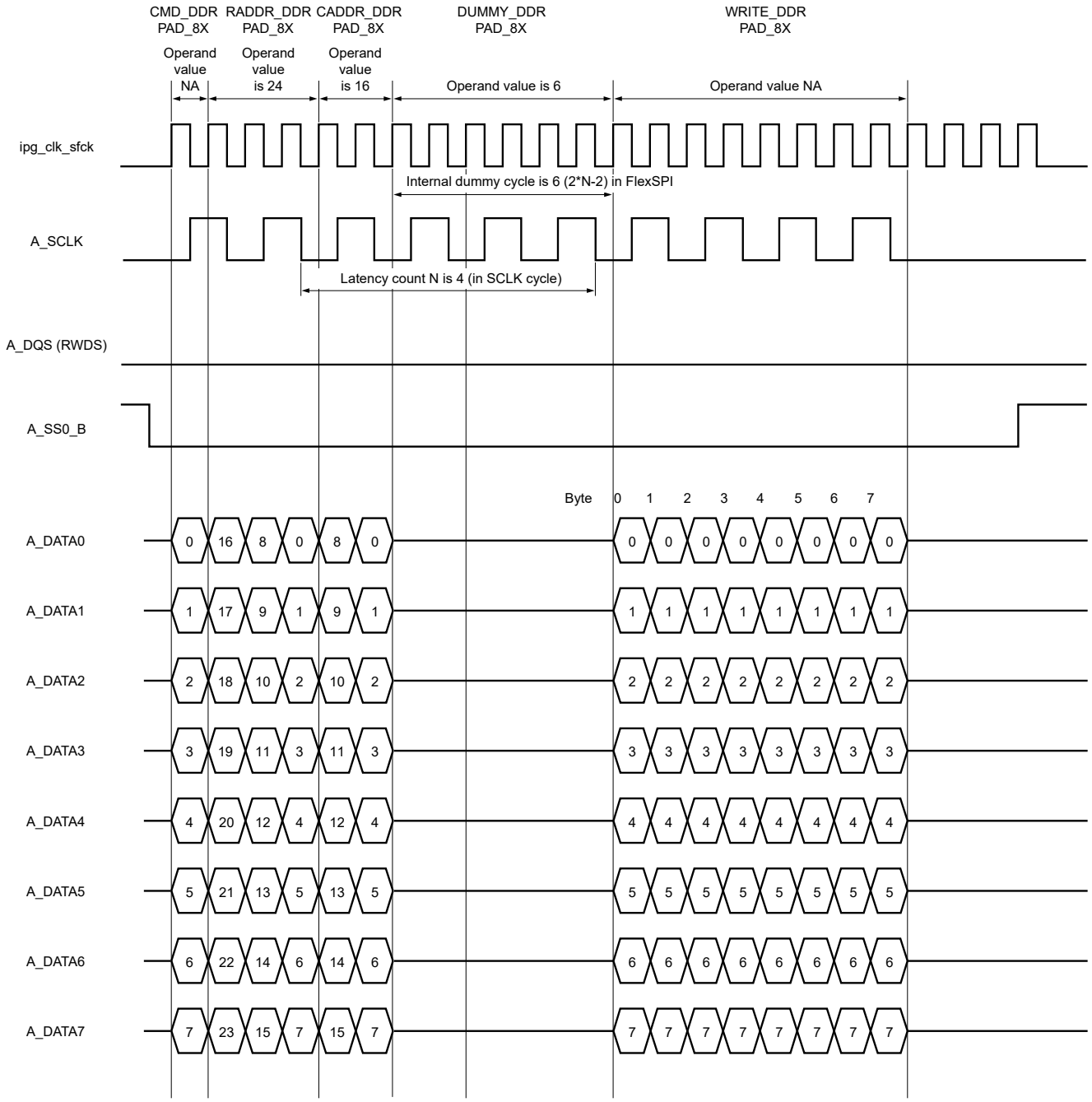


Figure 43. HyperBus device write transaction with single latency count

Figure 44 shows an example HyperBus device write transaction (additional latency count) in individual mode.

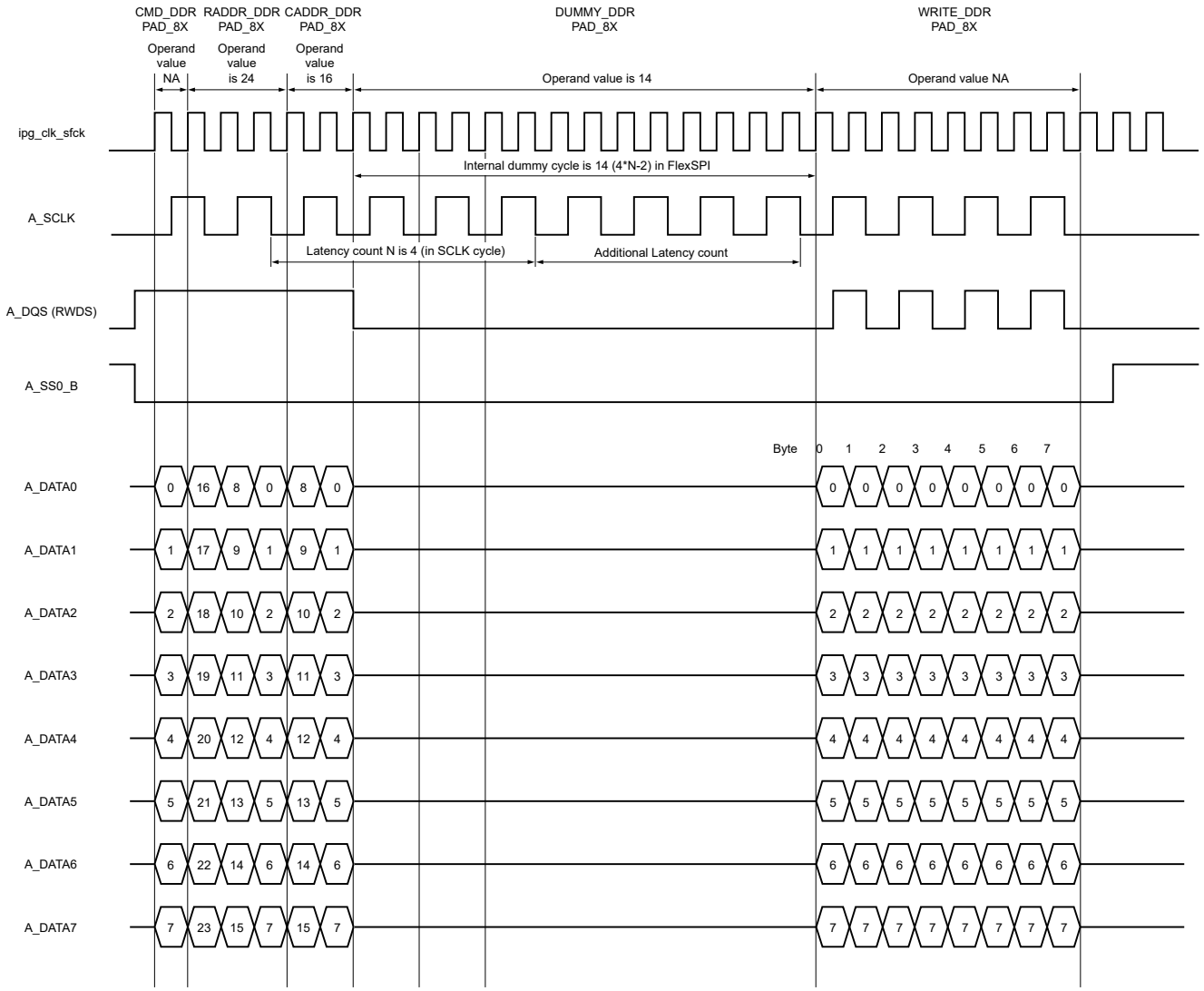


Figure 44. HyperBus device write transaction with additional latency count

19.3.7 Clocking

This section describes FlexSPI clocks and special clocking requirements.

Table 62. Clock usage

Clock name	Description	Comment
serial clock root (ipg_clk_sfck)	Root clock for Serial domain	FlexSPI core clock, used to generate the serial clock output. The clock is optionally divided using MCR0[SERCLKDIV] to obtain the SCLK.
ahb clock (hclk)	AHB bus clock	hclk frequency is higher than 1/4 of serial root clock, so internal async FIFO

Table continues on the next page...

Table 62. Clock usage (continued)

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

19.3.8 Interrupts

Table 63 describes all interrupts that FlexSPI generates.

Table 63. Flash memory address, row address, and column address

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

Table continues on the next page...

Table 63. Flash memory address, row address, and column address (continued)

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

19.3.9 Inline PRINCE encryption and decryption (IPED) CTR

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

NOTE

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

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

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

To use IPED, apply these settings:

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

When IPEDCTRL[CONFIG] = 1, additional settings are needed:

- When IPEDCTRL[AHBRD_EN] = 1, MISCCR3[RX_BUF_WR_EN_DELAY1] should be 1.
- When IPEDCTRL[AHBWR_EN] = 1 or IPEDCTRL[IPWR_EN] = 1, MISCCR3[TX_FIFO_WR_DELAY] should be 1.

FlexSPI supports a maximum of 4 address ranges for different 64-bit IPED.i_iv to encrypt or decrypt data.

The IPEDCTXnSTART and IPEDCTXnEND registers control the 4 ranges. There must be no overlap among these 4 ranges. Otherwise behavior of IP is undefined. Any single read or write request to flash memory must not cross one context boundary. For AHB read and write operations, hardware can automatically keep the boundary by writing 1 to AHBCR[ALIGNMENT]. Software must maintain the IP command write boundary. Any read or write request across context boundary leads to wrong data to or from the external device.

IPED context control 0 (IPEDCTXCTRL0) and IPED context control 1 (IPEDCTXCTRL1) control the write ability of IPEDCTXnSTART and IPEDCTXnEND. By default, IPEDCTXnSTART and IPEDCTXnEND cannot be modified after reset is enabled. Table 64 lists details.

Table 64. IPEDCTXnSTART and IPEDCTXnEND write ability

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

19.3.10 Flash memory access via IP command

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

1. If this command is a programing command, fill the IP transmit FIFO with programming data (for instance, programming flash data and flash memory status registers.)
2. Write the flash memory access start address to IPCR0[SFAR].
3. Write the read or program data size to IPCR1[IDATSZ]. Write the sequence index to IPCR1[ISEQID]. Write the sequence number to IPCR1[ISEQNUM].

4. Write 1 to IPCMD[TRG] to trigger a flash memory access command.
5. Wait for the INTR[IPCMDDONE] flag to set or for the IP command done interrupt to fire, indicating the command has completed on the FlexSPI interface.

NOTE

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

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

A triggered serial flash command contains:

- A flash memory access start address. IPCR0[SFAR] determines this address.
- A flash chip select. The flash memory access address and flash memory size setting (FLSHxCR0[FLSHSZ]) determine the chip select.
- A flash command sequence index and sequence number. FlexSPI sequentially executes the sequences indexed from IPCR1[ISEQID] to (IPCR1[ISEQID] + IPCR1[ISEQNUM]) in LUT.
- Flash individual memory access mode:
- The flash memory read or program data size, in bytes.
  - If the value of IPCR1[IDATSZ] is nonzero, the data size is IPCR1[IDATSZ].
  - If the value of IPCR1[IDATSZ] is zero, the operand value in the read or write instruction determines the data size.

NOTE

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

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

19.3.10.1 Reading data from IP receive FIFO

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

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

- 100h–17Ch (by IPS bus)

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

The processor or DMA can read the IP receive FIFO.

19.3.10.1.1 Reading via processor

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

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

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

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

Figure 45 shows the reading flow from the IP receive FIFO via processor.

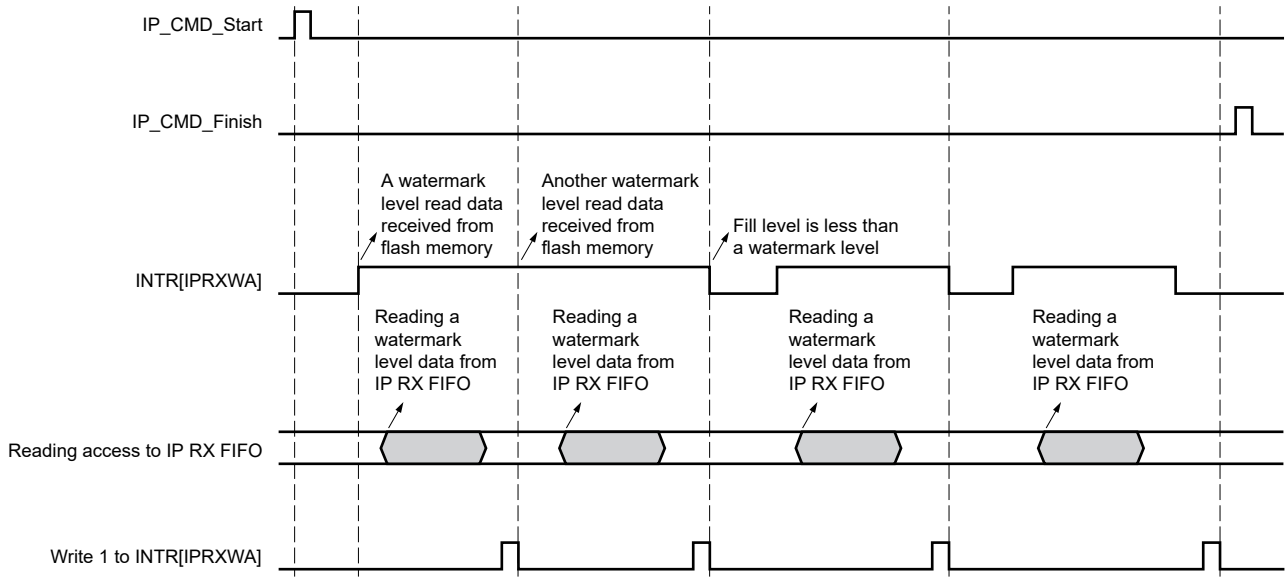


Figure 45. Reading IP receive FIFO via processor

NOTE

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

19.3.10.1.2 Reading via DMA

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

- Write 1 to IPRXFCR[RXDMAEN].
- Write the watermark level to IPRXFCR[RXWMRK]. The watermark level is (IPRXFCR[RXWMRK] + 1) × 8 bytes.
- Configure the DMA to transfer the same amount of data indicated by the watermark level: (XFERCFGa[XFERCOUNT] + 1) × XFERCFGa[WIDTH].

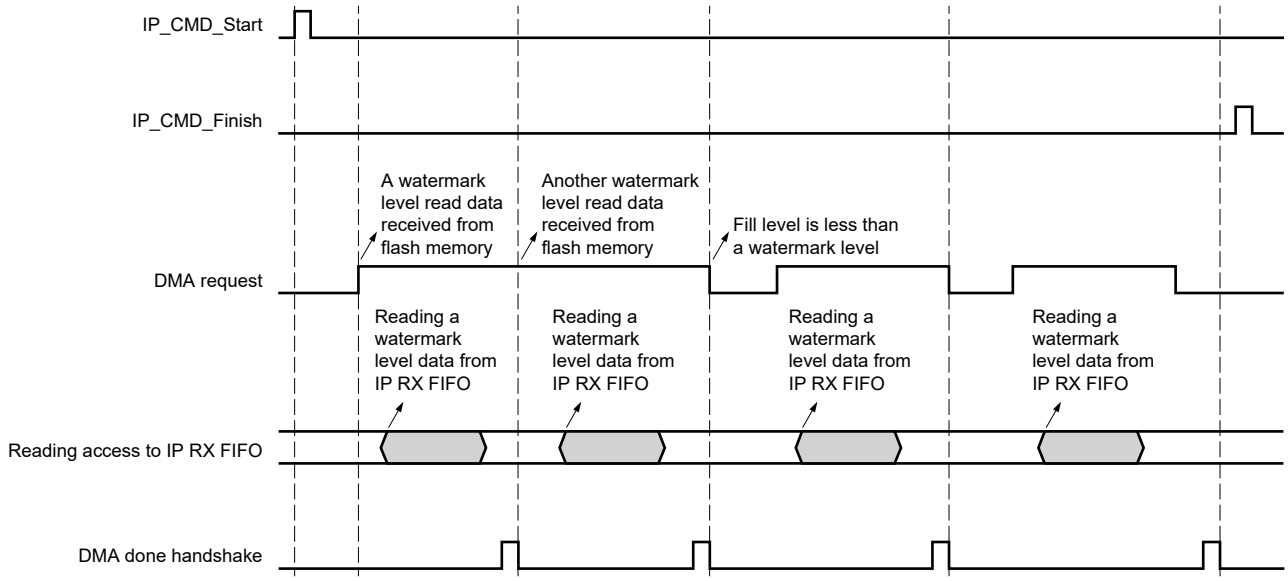


Figure 46. Reading from IP receive FIFO via DMA

NOTE

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

19.3.10.2 Writing data to IP transmit FIFO

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

- 400C_8180h–400C_81FCh (via IP bus)

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

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

19.3.10.2.1 Writing via processor

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

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

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

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

NOTE

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

Figure 47 shows the writing flow to the IP transmit FIFO via processor.

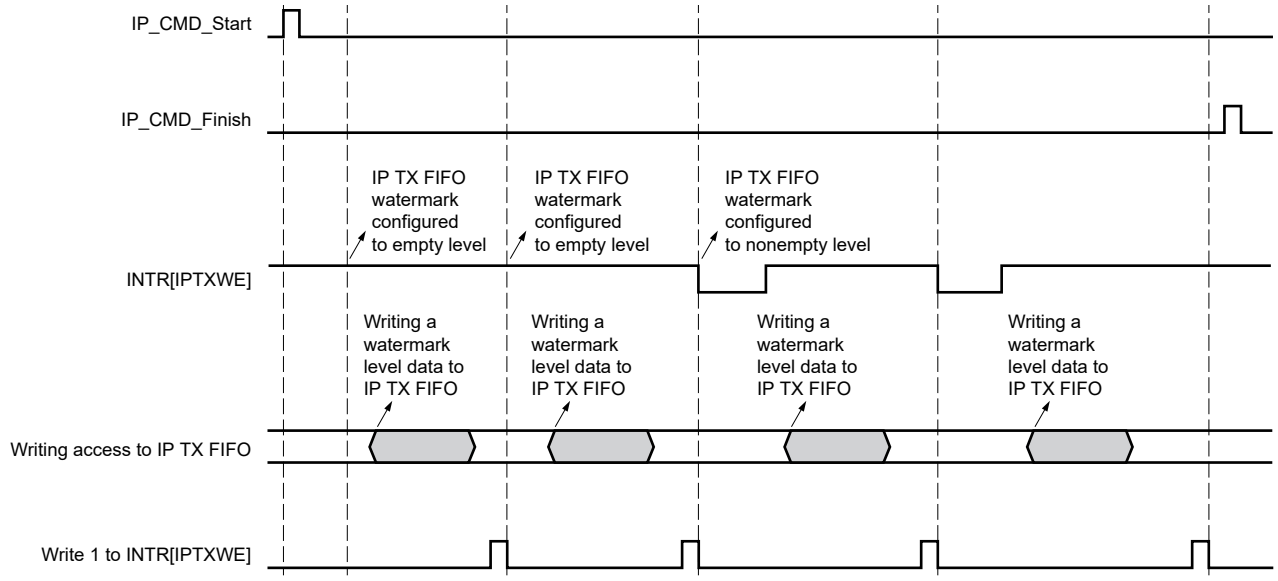


Figure 47. Writing to IP transmit FIFO via processor

NOTE

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

19.3.10.2.2 Writing via DMA

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

- Write 1 to IPTXFCR[TXDMAEN].
- Write the watermark level to IPTXFCR[TXWMRK]. The watermark level is (IPTXFCR[TXWMRK] + 1) × 8 bytes.
- Configure the DMA to transfer the same amount of data that the watermark level indicates: (XFERCFGa[XFERCOUNT] + 1) × XFERCFGa[WIDTH].

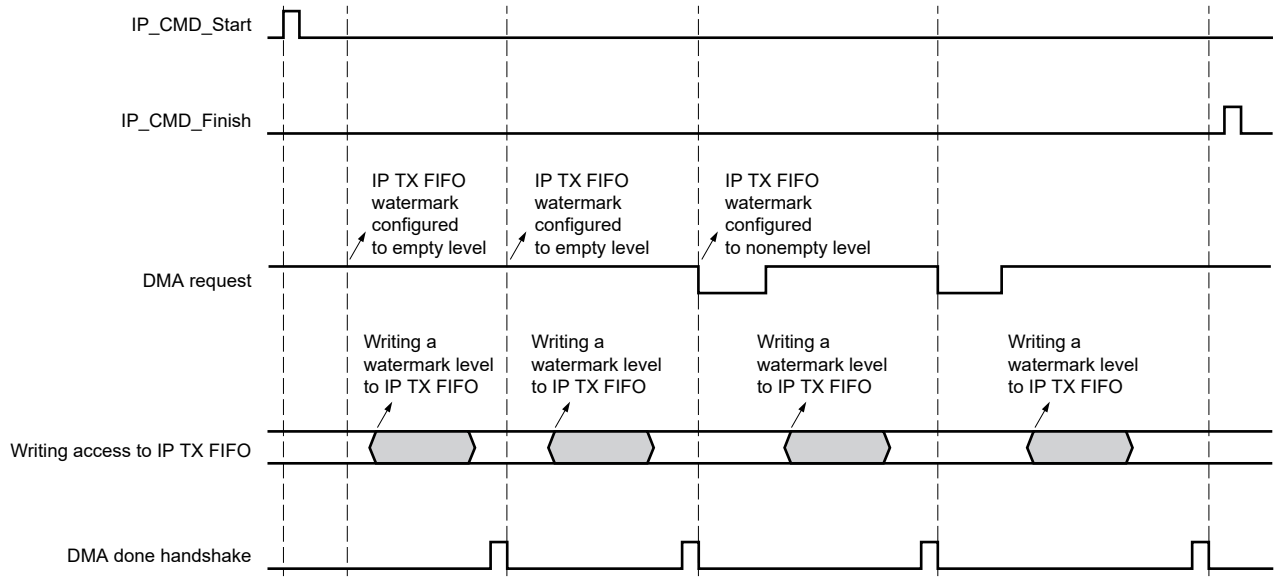


Figure 48. Writing into IP transfer FIFO via DMA

NOTE

- When there is empty space in the IP transmit FIFO greater than the watermark level, a DMA request is generated. This request is level-valid, not pulse-valid.
- DMA writes a watermark level amount of data into the IP transmit FIFO each time. (Configure the DMA transfer size to the same value that the watermark level indicates).
- DMA returns a done handshake (pulse valid) to FlexSPI each time it finishes writing a watermark level amount of data.
- The DMA done handshake (not each write access) pushes the IP transmit FIFO data.
- The total program data size (when executing multiple DMA transfers) is not required to be a multiple of the watermark level. After writing all data into the IP transmit FIFO and all command sequences to flash memory are finished (INTR[IPCMDDONE] = 1), write 1 to IPTXFCR[CLRIPTXF] to clear the FIFO. Failure to clear the FIFO results in incorrect programming data for the next programming command to flash memory.
- For transfers larger than 1 kB using the transmit or receive FIFO, you must use DMA chained transfers. (See DMA controller.)
- DMA transfer using FlexSPI is channel-bounded, even if the transfer uses triggers instead of requests. For FlexSPI:
  - DMA_FLEXSPI0_RX: 22
  - DMA_FLEXSPI0_TX: 23
  - DMA_FLEXSPI1_RX: 25
  - DMA_FLEXSPI1_TX: 26

19.3.11 Flash memory access via AHB command

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

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

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

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

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

19.3.11.1 AHB write access to flash memory

For AHB write access to flash memory, FlexSPI buffers the write data from the AHB bus into the internal AHB transmit buffer. FlexSPI then transmits the data to flash memory. FlexSPI only buffers write data for one AHB burst. Figure 49 shows the hardware operation in response to an AHB write access to flash memory.

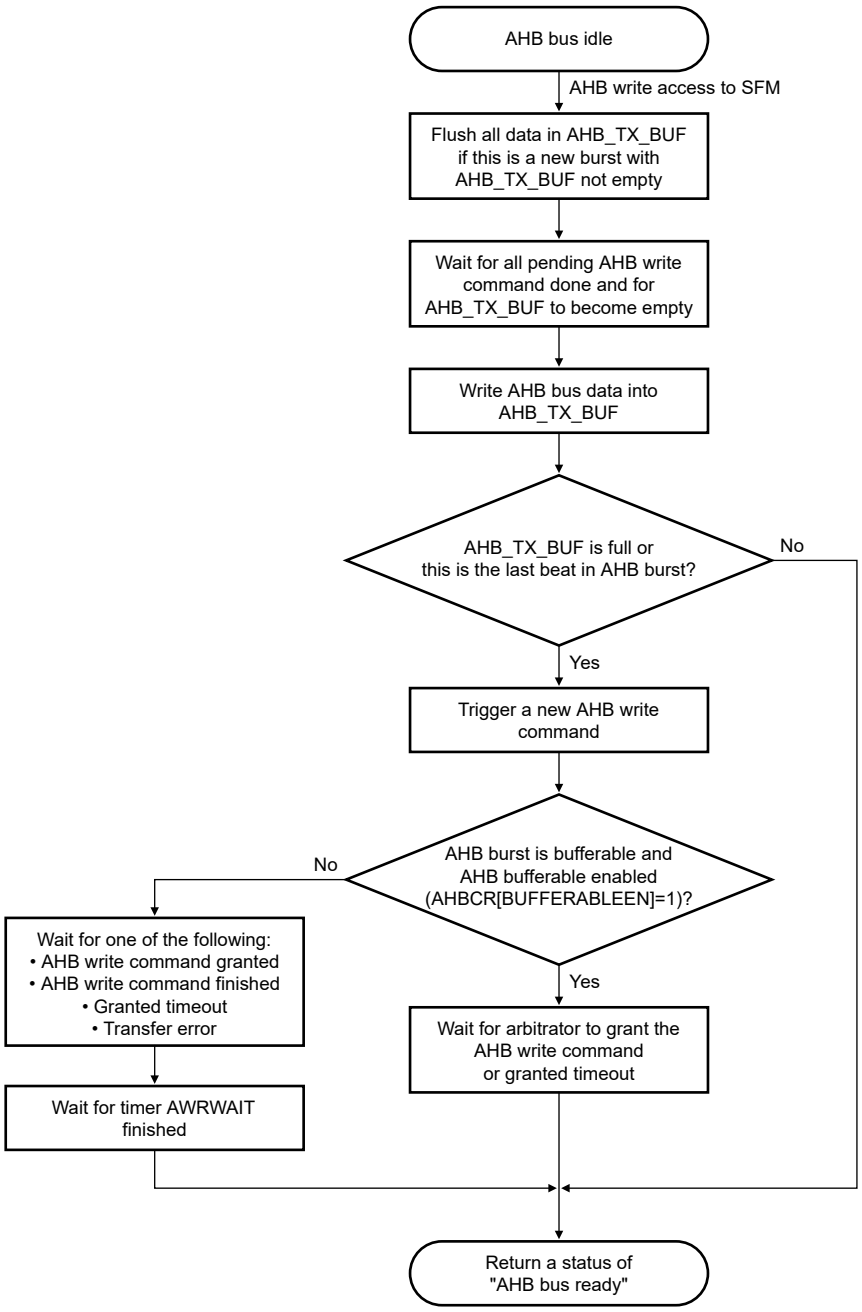


Figure 49. Hardware operation in response to AHB write access to flash memory

NOTE

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

FlexSPI triggers a new AHB write command in these cases:

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

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

Table 65 describes the details of a triggered AHB write command.

Table 65. Triggered AHB write command

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

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

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

  Two AHB write commands are triggered: the first command with flash memory start address 8h and data size 40h; the second command with flash memory start address 48h and data size 10h. See Figure 50.

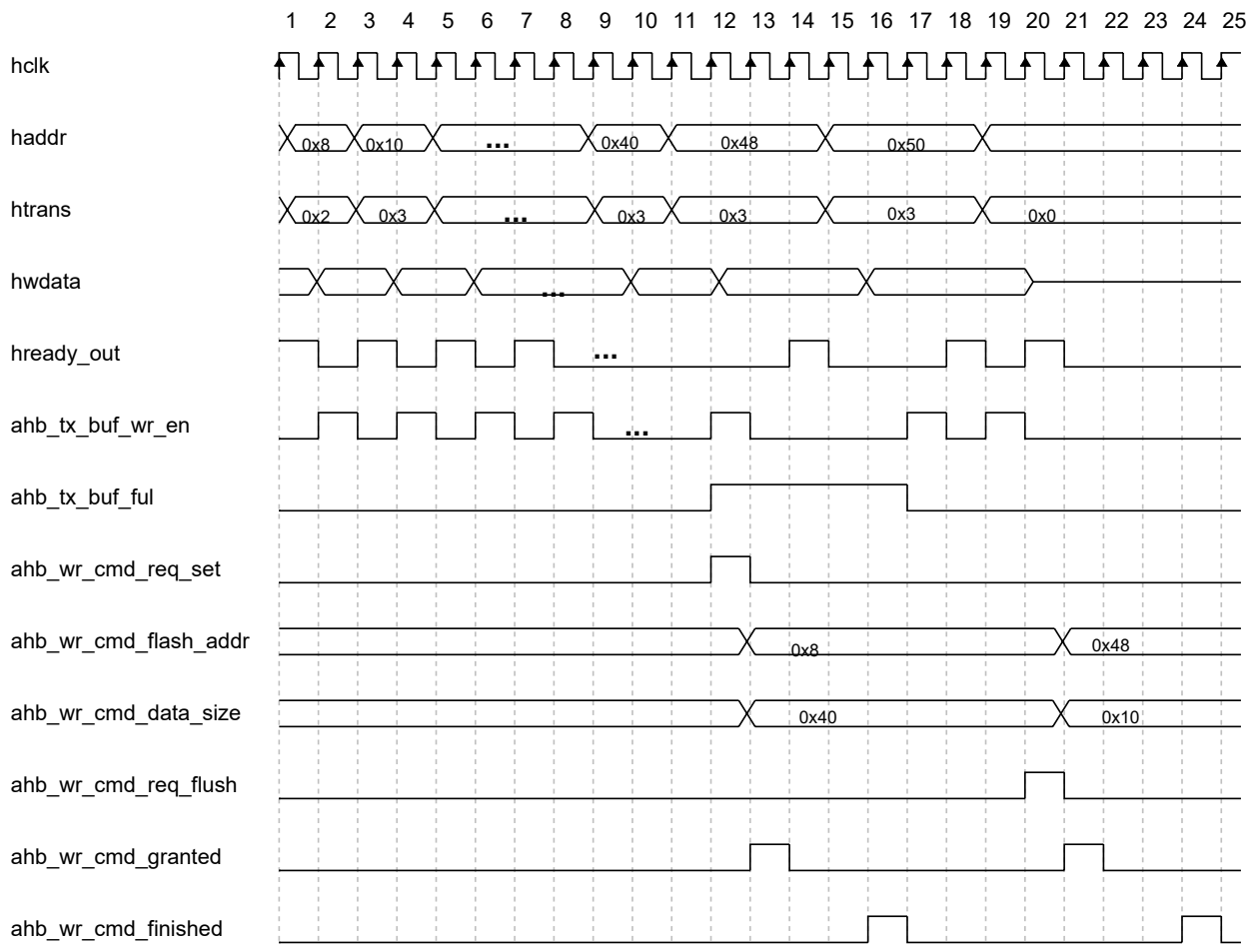


Figure 50. AHB write access (INCR, 64-bit, bufferable)

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

  One AHB write command is triggered with flash memory start address 0h and data size 40h. See Figure 51.

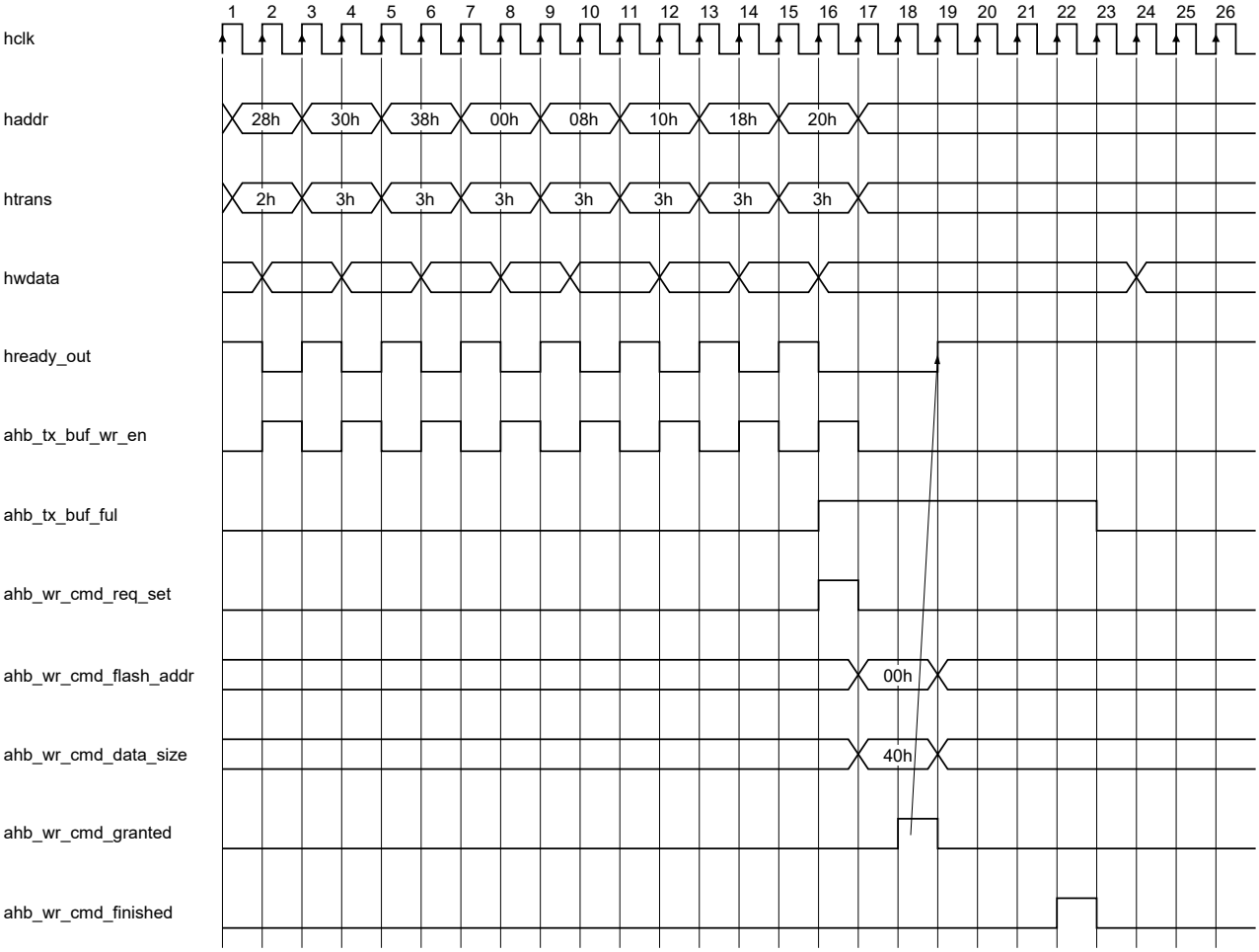


Figure 51. AHB write access (WRAP8, 64-bit, bufferable)

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

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

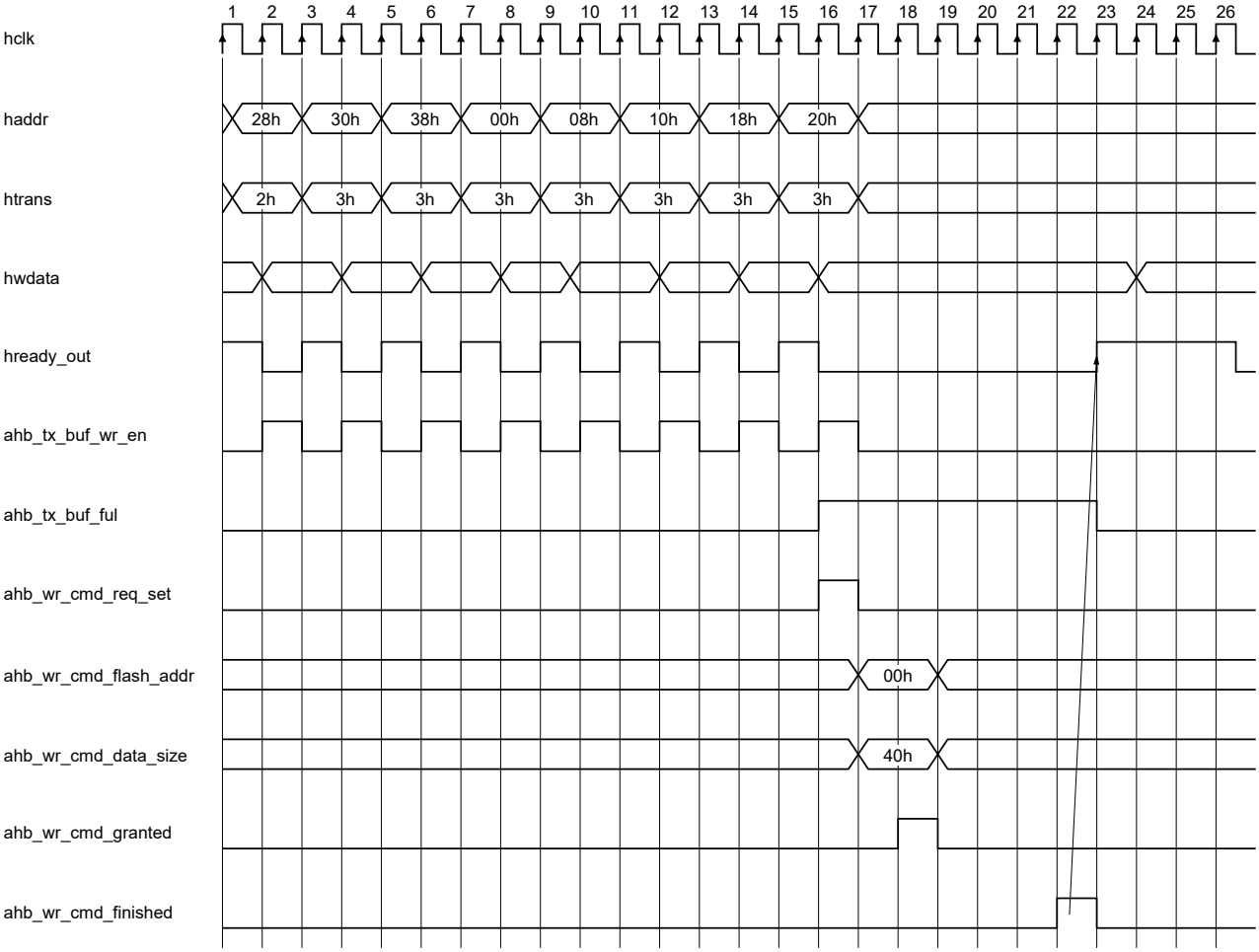


Figure 52. AHB write access (WRAP8, 64-bit, non-bufferable)

NOTE

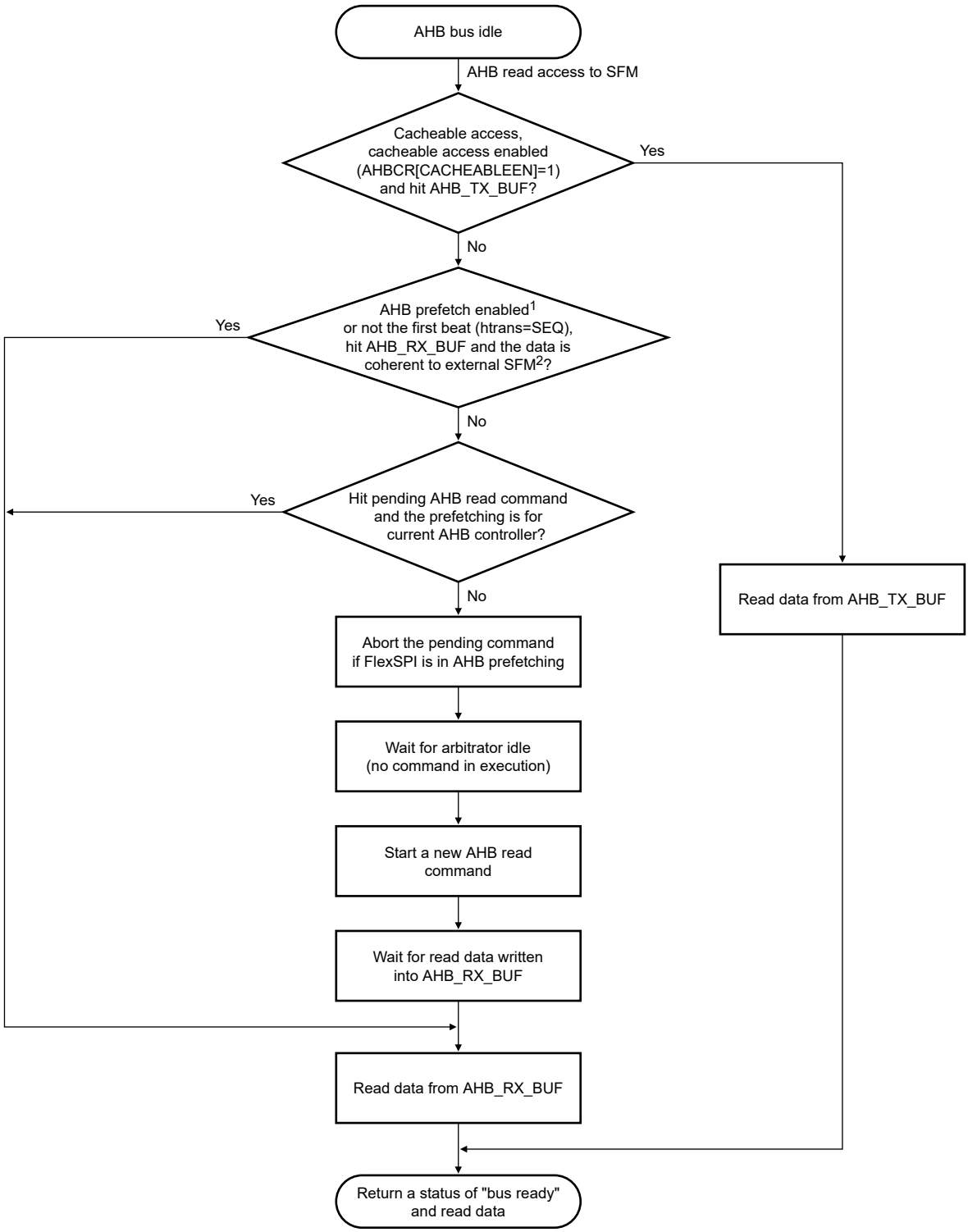
If the burst data size (in bytes) is greater than the AHB transmit buffer size, wrapper burst is not supported. For example, if AHB_TX_BUF is 64 bytes, AHB WRAP16 (16 × 64 bits = 128 bytes) write burst access is not supported.

19.3.11.2 AHB read access to flash memory

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

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

If all of these options are missed, FlexSPI triggers a new AHB read command to fetch data from flash memory. Figure 53 shows the hardware operation in response to AHB read access to flash memory.



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

Figure 53. Hardware operation in response to AHB read access to flash memory

Table 66 describes the details of a triggered AHB read command.

Table 66. Triggered AHB read command

Item	Description
Flash memory access start address and data size	Determined by AHB address, burst type, and burst size. FlexSPI fetches read data from the start address for the current burst or beat. See Table 67 for details.
Flash memory chip select	FlexSPI uses the settings of the flash memory access start address and flash memory size to determine the chip select.
Flash memory command sequence index	Determined by FLSHxCR2[AWRSEQID].
Flash memory command sequence number	Determined by FLSHxCR2[AWRSEQNUM]. If AWRSEQNUM is not zero, multiple flash memory access command sequences are triggered every time for an AHB read command. The sequences indexed from AWRSEQID to (AWRSEQID + AWRSEQNUM) in LUT are executed sequentially.
Flash memory access mode (Individual)	

NOTE

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

Table 67. AHB read command flash memory start address and data size

Prefetch enable	Cross flash boundary	Burst type	Flash memory start address [26:0]	Data size
0	Never cross flash boundary, because AHB burst never crosses 1 KB boundary.	SINGLE, INCR4, INCR8, INCR16	hbeat_start_address	(hburst_end_address-hbeat_start_address)
		INCR	hbeat_start_address	byte size of current beat For INCR burst with prefetch disabled, each beat is handled same as SINGLE burst.
		WRAP4, WRAP8, WRAP16	hburst_start_address	(hburst_end_address-hburst_start_address)
1	No	INCR, SINGLE, INCR4, INCR8, INCR16	hbeat_start_address	ahb_rx_buf_sz
	No	WRAP4, WRAP8, WRAP16	hburst_start_address	ahb_rx_buf_sz

Table continues on the next page...

Table 67. AHB read command flash memory start address and data size (continued)

Prefetch enable	Cross flash boundary	Burst type	Flash memory start address [26:0]	Data size
	Yes	INCR, SINGLE, INCR4, INCR8, INCR16	hbeat_start_address	(flash_top_address-hbeat_start_address)
	Yes	WRAP4, WRAP8, WRAP16	hburst_start_address	(flash_top_address-hburst_start_address)

NOTE

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

19.3.11.3 AHB receive buffer management

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

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

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

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

NOTE

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

19.3.11.3.1 Prefetch buffer enhancement

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

If the switches between task A and task B happen frequently, then FLEXSPI can end up re-reading items it had previously stored in the prefetch buffer as the buffer ends up being flushed often due to the task switches.

To increase the efficiency of FLEXSPI in situations like those described in the example above, the prefetch buffer enhancement feature can be enabled. When prefetch buffer enhancement is enabled (AHBRXBUFnCR0[REGIONEN]=1), then multiple prefetch buffers can be assigned to the same controller with an address range (AHBBUFREGIONSTARTn/ENDn) used as an additional filter to determine which buffer is used.

FLEXSPI compares the controller ID along with corresponding address range of each buffer to select one buffer. Other buffers are not impacted. This way, the AHB controller can prefetch the data from multiple address ranges in each of the buffers.

Program sequence:

- Configure AHBBUFREGIONSTARTn and AHBBUFREGIONENDn with the desired address ranges. Note that the addresses here use the AHB address including the system base address, not the serial flash address with the base address removed
- END must be larger than START
- Set AHBRXBUFnCR0[REGIONEN] = 1
- Set AHBRXBUFnCR0[MSTRID] with the MSTRID corresponding to the controller that will own multiple buffers, for example 4'b0011

NOTE

- This function is only meaningful when prefetch is enabled.
- Only Buffer 0 - 3 support this enhancement when REGIONEN=1, other buffers work like before. Buffer 0 - 3 with REGIONEN=0 works like before.

Prefetch buffer enhancement example:

The core in the system uses MSTRID = 0. The core will frequently switch between task A at 0x8000_1000-0x8000_1FFF and task B at 0x8000_2000-0x8000_2FFF, so dedicated prefetch buffers will be used for these tasks. The core will access other addresses in the FLEXSPI region using a third prefetch buffer.

To achieve this configuration:

- AHBBUFREGIONSTART0 = 0x8000_1000 and AHBBUFREGIONEND0 = 0x8000_2000. Configure Buffer 0 to the address range used by task A.
- AHBBUFREGIONSTART1 = 0x8000_2000 and AHBBUFREGIONEND1 = 0x8000_3000. Configure Buffer 1 to the address range used by task B.
- AHBRXBUF0CR0 = 0xC000_0004. Buffer 0 is a 256 bytes buffer assigned to the core (MSTRID = 0) with the prefetch buffer enhancement feature enabled (REGIONEN = 1), so only task A addresses will use buffer 0.
- AHBRXBUF1CR0 = 0xC000_0004. Buffer 1 is a second 256 bytes buffer assigned to the core (MSTRID = 0) with the prefetch buffer enhancement feature enabled (REGIONEN = 1), so only task B addresses will use buffer 1.
- AHBRXBUF4CR0 = 0x8000_0004. Buffer 4 is a third 256 bytes buffer assigned to the core (MSTRID = 0). The prefetch buffer enhancement feature will not be used for this buffer (and is not available for buffer 4 anyway). Core accesses that do not hit in Buffer 0 or Buffer 1 will use Buffer 4.

19.3.12 Command arbitration

There are four flash memory access command sources:

1. AHB command, that AHB write access to serial flash memory space triggers
2. AHB command, that AHB read access to serial flash memory space triggers
3. IP command, that writing IPCMD[TRG] triggers
4. Suspended command, an AHB read prefetch sequence which is suspended

An AHB bus access never triggers a write command and a read command at the same time.

The AHB prefetch sequence is an AHB command sequence triggered via AHB read access when AHB prefetch is enabled. After all read data is fetched for the current AHB read burst, FlexSPI prefetches more data to reduce read latency for the next AHB read access. An AHB command for a read operation is never aborted while fetching read data for the current AHB read burst. However, a new IP command or AHB command request while prefetching data (not for the current read burst) can abort an AHB read command.

When the arbitrator is idle (STS0[ARBIDLE] = 1), the granted priority of these command sources is:

1. AHB command (read or write)
2. IP command
3. Suspended command

NOTE

When the arbitrator is idle and there is no AHB or IP command request, a suspended command is not granted immediately. The arbitrator waits MCR2[RESUMEWAIT] AHB clock cycle idle states before resuming the suspended command to avoid AHB prefetch sequence being resumed and suspended frequently.

If the arbitrator is busy executing AHB or IP commands (not suspended commands), no new command requests are granted. If the grant times out, an AHB or IP command granted error occurs.

If a new AHB or IP command request comes while arbitrator is executing AHB read prefetch sequence, AHB read prefetch sequence is aborted (but not immediately). Arbitrator grants the AHB or IP command request after AHB read prefetch sequence is aborted on the FlexSPI interface and saved all internal data pointers.

19.3.12.1 Command abort and suspend

Table 68 describes the command abort and suspend mechanism.

Table 68. Command abort and suspend

Action	Description
Aborting a command	As mentioned above, if a new AHB or IP command request arrives, an AHB read prefetch sequence can be aborted.
Suspending a command	When an AHB read prefetch sequence is aborted on the FlexSPI interface, FlexSPI saves this suspended sequence in the cases listed below. FlexSPI resumes this sequence after the arbitrator has been idle for enough time. • There is no valid suspended command (AHBSPNDSTS[ACTIVE] = 0). This condition can occur when no command is suspended or when the suspended command is resumed. • The priority of aborted AHB read prefetch sequence is higher than the active suspend sequence. **NOTE** FlexSPI ignores and never resumes the original suspended sequence.
Activating a suspended command	The suspended command (internal status) becomes active when an AHB prefetch command is aborted and suspended. It becomes inactive in the following cases: • The arbitrator resumes the suspended command. • The AHB controller triggers a new AHB read command request using the same AHB receive buffer (buffer ID). **NOTE** MCR0[SWRESET] can be used to clear any suspended command. The saved suspended command is dropped and no resuming occurs. The suspended command must be cleared after the flash page program or erase. Otherwise, another command may lead to a period in which the flash memory cannot support a read command.

Figure 54 shows an example of the flow of command abort, suspend, or resume.

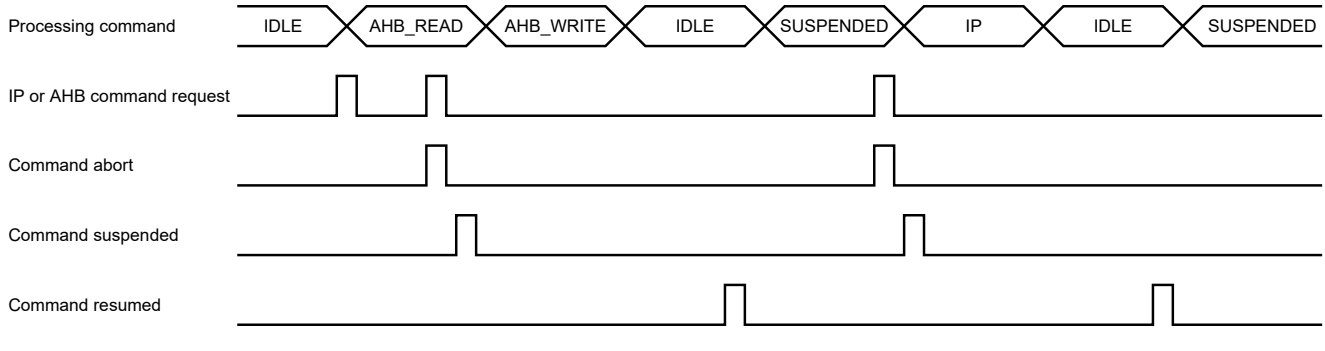


Figure 54. Command instruction execution on FlexSPI interface

NOTE

- The AHB prefetch sequence is aborted, suspended, or resumed twice in Figure 54. A new AHB write command request aborts the sequence the first time, and a new IP command request aborts the sequence the second time.
- The suspended sequence is resumed after a wait time of idle state. If there are frequent IP or AHB command requests and the wait time is not set reasonably, the suspend and resume activities may be triggered frequently.

19.3.13 SCLK stop

FlexSPI stops SCLK output toggling:

- When programming data is not ready for a programming command sequence.
- When an internal FIFO has no space to receive data for reading command sequence.

Certain devices may not support the stopping of SCLK during a command sequence (chip select is valid). SCLK stopping can be avoided via the methods described below.

- For flash memory reading that an IP command triggers:
  - Never trigger a read command with data size larger than IP receive FIFO size.
  - Internal asynchronous FIFO for flash memory reading should never be full.

    FlexSPI pops 64 bits of data per AHB clock cycle from this FIFO, and receives data from FlexSPI interface on the serial root clock. The flash memory access mode (Single, Dual, Quad, or Octal mode and Individual mode) determines the receiving speed. For example, in Octal mode, FlexSPI receives 16 bits per serial root clock cycle. If the AHB clock frequency is faster than 1/4 of the serial root clock, this asynchronous FIFO is never full.
- For flash memory programming that an IP command triggers:
  - Never trigger a program command with data size larger than the IP transmit FIFO size. Write all programming data into the IP transmit FIFO before triggering the IP command.
  - The internal asynchronous FIFO for flash memory programming must never be empty.

    FlexSPI fetches 64 bits of programming data per AHB clock cycle into this FIFO, and transmits data to the FlexSPI interface on the serial root clock. The flash memory access mode (Single, Dual, Quad, or Octal mode and Individual mode) determines the transmitting speed. For example, in Octal mode, FlexSPI transmits 16 bits per serial root clock cycle. If the AHB clock frequency is faster than 1/4 of the serial root clock, this asynchronous FIFO is never empty.
- For flash memory reading or programming that an AHB command triggers:
  - The internal asynchronous FIFO for flash memory reading and programming must never be full or empty. The frequency ratio limitation is same as it is for flash memory reading or flash memory programming that an IP command triggers.

NOTE

- FlexSPI never triggers an AHB read command with data size larger than the internal AHB receive buffer size.
- FlexSPI never triggers an AHB program command with data size larger than the internal AHB transmit buffer size.
- All programming data is buffered into the AHB transmit buffer before triggering an AHB program command in FlexSPI.

19.3.14 FlexSPI output timing

19.3.14.1 Output timing between data and SCLK

This section describes the output timing relationship of data (on A_DATA) and SCLK. There are three cases.

- **SDR instruction in SDR sequence**

  An SDR sequence contains only SDR instructions. In this case, all data bits last one serial root clock cycle on the FlexSPI interface. Figure 55 shows the relationship between the serial root clock, data, and SCLK:

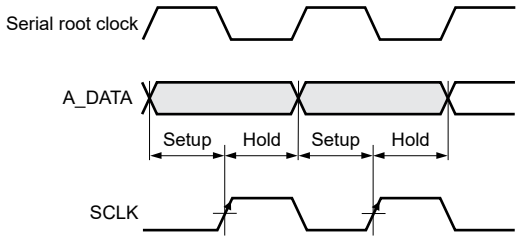


Figure 55. SDR instruction in SDR sequence

- **SDR instructions in a DDR sequence**

  A DDR sequence is a flash memory access command sequence that contains DDR instructions other than DUMMY_DDR. It may contain SDR instructions. If there are SDR instructions in a DDR sequence, all data bits last for two serial root clock cycles on the FlexSPI interface. Figure 56 shows the relationships between the serial root clock, data, and SCLK.

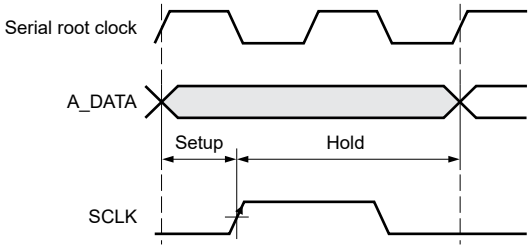


Figure 56. SDR instruction in DDR sequence

- **DDR instructions in a DDR sequence**

  For DDR instructions in a DDR sequence, all data bits last one serial root clock cycle on the FlexSPI interface. Figure 57 shows the relationships between the serial root clock, data, and SCLK.

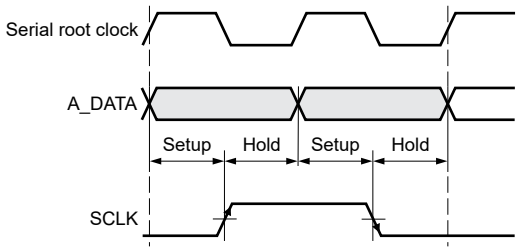


Figure 57. DDR instruction in DDR sequence

19.3.14.2 Output timing between chip select and SCLK

This section describes the output timing relationship between chip select (on A_SS0_B, A_SS1_B) and SCLK. The timing relationship for the SDR sequence differs a little from the timing relationship for the DDR sequence.

- **Chip select timing in SDR sequence**

  For an SDR sequence, the delay from chip select assertion to the SCLK rising edge is (FLSHxCR1[TCSS] + 0.5) cycles of the serial root clock. The delay from SCLK falling edge to chip select deassertion is FLSHxCR1[TCSH] cycles of serial root clock. Figure 58 shows the timing relationship between the chip select and SCLK.

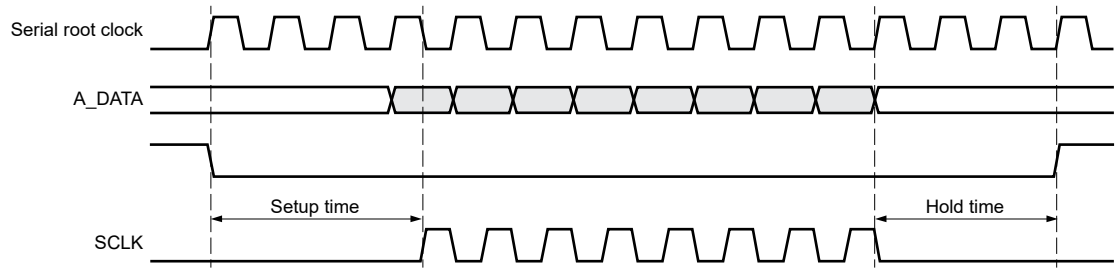


Figure 58. Chip select output timing for SDR sequence

- **Chip select timing in DDR sequence**

  For a DDR sequence, the delay from chip select assertion to the SCLK rising edge is **(FLSHxCR1[TCSS] + 0.5)** cycles of the serial root clock. The delay from the SCLK falling edge to chip select deassertion is **(FLSHxCR1[TCSH] + 0.5)** cycles of serial root clock.

  NOTE

  When AHB receive prefetch is enabled, and prefetch can be aborted, configure TCSH to 1 or greater. Configuring TCSH in this way guarantees a positive chip select hold time after the SCLK falling edge.

  Figure 59 shows the timing relationship between chip select and SCLK.

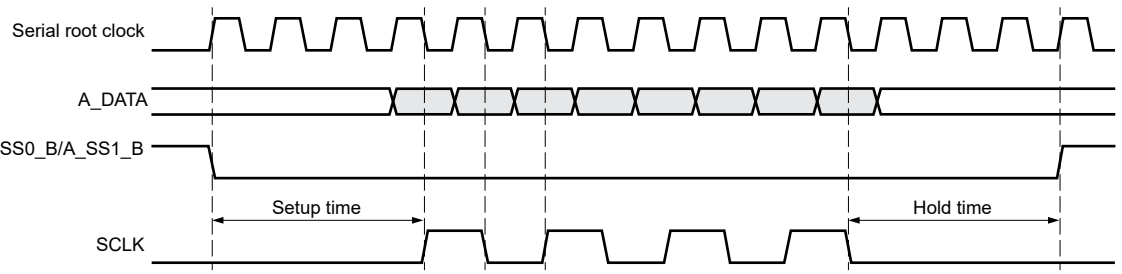


Figure 59. Chip select output timing for DDR sequence

For devices such as an FPGA device, a minimum chip select negation time (time between two chip select assertions) may be required. If the value of FLSHxCR1[CSINTERVAL] is nonzero, FlexSPI ensures a delay between chip select valid assertions. The delay time is CSINTERVAL × 256 cycles of serial root clock for an SDR or DDR sequence. If the external device has no limitation, configure the value of this field to zero. Figure 60 shows the timing of the chip select interval.

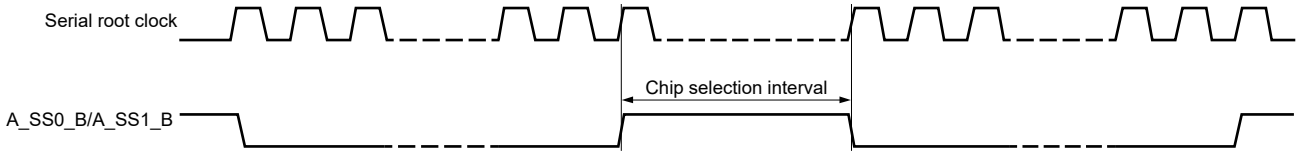


Figure 60. Chip select valid interval

19.3.15 FlexSPI input timing

This section describes the input timing of FlexSPI.

19.3.15.1 Receiving block

This section describes the receiving block in FlexSPI.

FlexSPI decodes instruction code to determine SDR mode or DDR mode. It also determines whether both the rising and falling edges are used for flash data sampling. FlexSPI samples the data line for learn instructions and read instructions.

MCR0[RXCLKSRC] selects the clock source from these four sources for the sampling clock:

- Internal dummy read strobe and looped back internally
- Internal dummy read strobe and looped back from DQS pad
- SCK output and looped back from SCK pad
- Flash-memory-provided read strobe

For details about DLL configuration, see DLL configuration for sampling.

There are 16 clock phases for sampling data. An internal delay chain delays the clock phases.

See Data learning.

FlexSPI does not support read operations with a zero-turnaround cycle for direction switching on I/O pins. This read operation often occurs in QuadSPI configuration register reads. In Figure 61 and Figure 62, SIO[3:0] is an output from FlexSPI in the CMD phase, and is an input to FlexSPI in the DATA phase. No time remains for FlexSPI to switch the direction. This kind of read is not supported. FlexSPI and flash memory drive the I/O pins at the same time, which may cause a problem. To avoid this behavior on I/O pins, using SPI mode or reading with dummy cycles is required.

NOTE

The cycle numbers are not accurate in the examples below. See the device spec for accurate cycle numbers.

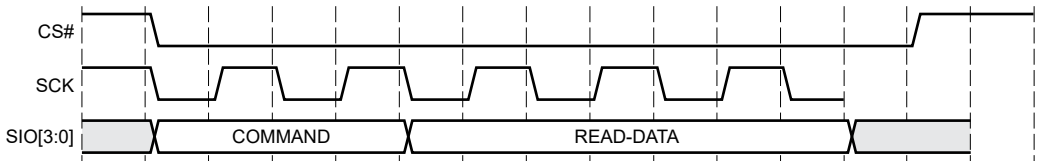


Figure 61. Unsupported read behavior

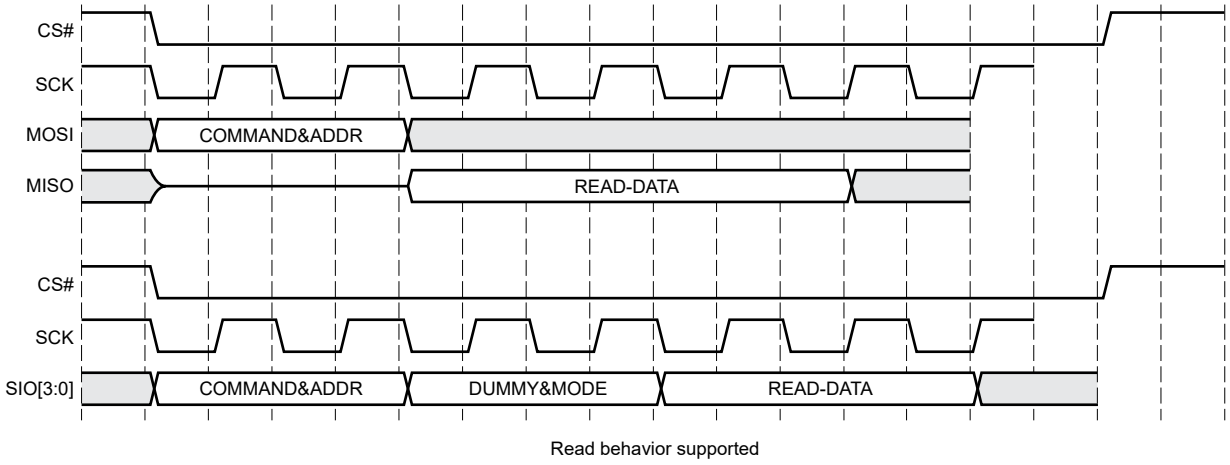


Figure 62. Supported read behavior

19.3.15.2 Receive clock source features

FlexSPI uses a read strobe to sample incoming data. There are several options for generating the read clock strobe that MCR0[RXCLKSRC] selects. Using the receive clock source method has implications for the board connections and can affect the

maximum frequency that the interface supports. See the device data sheet for details on the maximum frequency supported for each RXCLKSRC option.

Table 69. RXCLKSRC options

MCR0[RXCLKSRC]	Description	Board connection	Max Frequency
0	Internal dummy read strobe and internal loopback	DQS pin is not used. The DQS pin can be configured for an alternate signal function.	Lowest
1	Internal dummy read strobe and loopback from DQS pad	FlexSPI uses the DQS pin, and it must be configured for its FlexSPI function. The internally generated read strobe is sent to the DQS pin and is sampled at the pin to match more closely the data pin timings. The DQS pin is typically left floating. External capacitance can be added to adjust timing.	Medium
2	SCK output and loopback from SCK pad	SCK is used as the read strobe. FlexSPI uses the signal looped back from the SCK pad as the read strobe to match more closely the data pin timings. DQS pin is not used and can be configured for an alternate signal function.	Medium
3	Flash-memory-provided read strobe	DQS pin is connected to the DQS or RWDS signal that the memory provides.	Highest

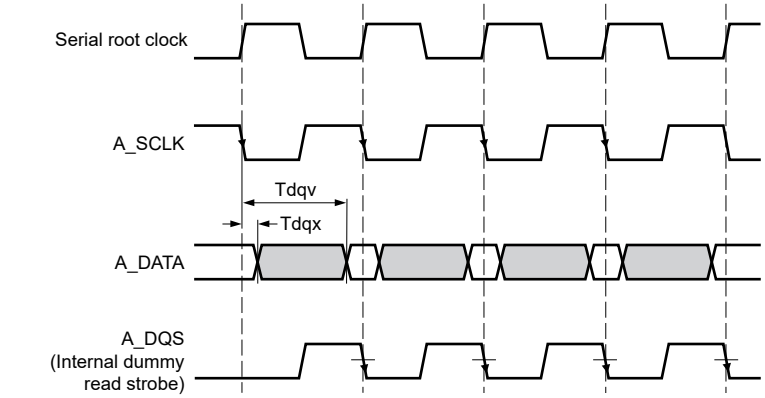
19.3.15.3 Input timing for sampling with dummy read strobe

This section describes the input timing when sampling with the internal dummy read strobe (MCR0[RXCLKSRC] = 0 or 1). The timing for sampling with an internal dummy read strobe loopback is very similar to the timing for loopback from pad. Sampling with a dummy read strobe loopback from the DQS pad can achieve a higher read frequency. It compensates for the delay of the SCLK output path and data pin input path.

The input timing is different for SDR mode and DDR mode.

- **Input timing for sampling with dummy read strobe in SDR mode**

  For SDR read or learn instructions, FlexSPI samples input data pins with the falling edge of the dummy read strobe. Figure 63 shows the input timing for sampling with dummy read strobe in SDR mode.



Tdqv is the setup time from SCLK to the next valid data.
Tdqx is the hold time from SCLK to the current valid data.

Figure 63. Input timing for sampling with dummy read strobe in SDR mode

- **Input timing for sampling with dummy read strobe in DDR mode**

  For DDR read or learn instructions, FlexSPI samples input data pins on rising and falling edges of the dummy read strobe. Figure 64 shows the input timing for sampling with dummy read strobe in DDR mode.

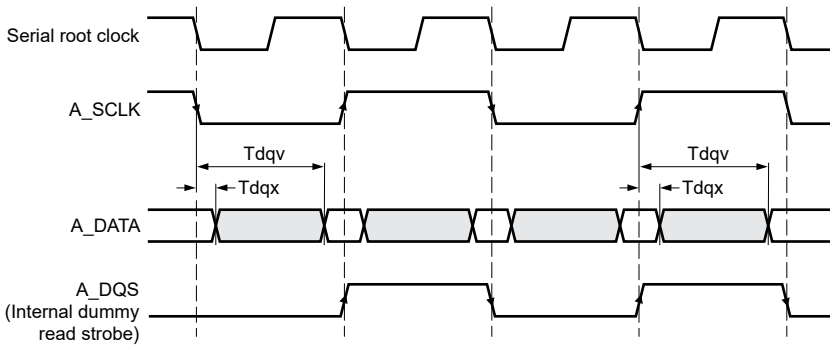


Figure 64. Input timing for sampling with dummy read strobe in DDR mode

19.3.15.4 Input timing for sampling with SCK output looped back from SCK pad

This section describes the input timing when sampling with the SCK output looped back from the SCK pad (MCR0[RXCLKSRC] = 2). The input timing is similar to sampling with a dummy read strobe. SCK output toggles for all types of instructions, but internal dummy strobe only toggles for read and learn instructions. In this case, FlexSPI receives more data bits when sampling with SCK output. FlexSPI automatically ignores the redundant data bits sampled.

19.3.15.5 Input timing for sampling with flash-memory-provided read strobe

This section describes the input timing when sampling with flash-memory-provided read strobe (MCR0[RXCLKSRC] = 3). The input timing is different for SDR mode and DDR mode.

NOTE

There are no known devices that provide a read strobe and support SDR mode operation.

- **Flash-memory-provided read strobe with SCLK2**

  Certain flash devices provide a read strobe with SCLK2 and provide read data with SCLK. Then the read strobe edge is in the center of a valid data window. The FlexSPI controller samples the read data with read strobe directly (DLL is bypassed). See

DLL configuration for sampling. Figure 65 and Figure 66 show the input timing for sampling with flash memory read strobe in SDR mode and DDR mode.

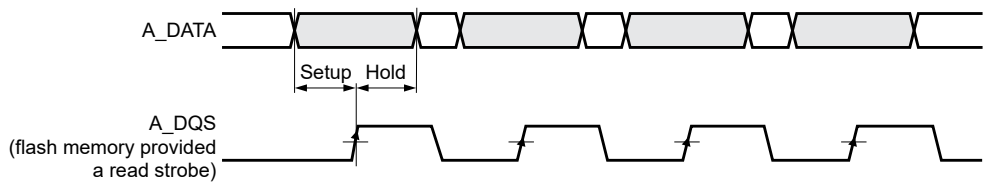


Figure 65. Input timing 1 for flash-memory-provided read strobe in SDR mode

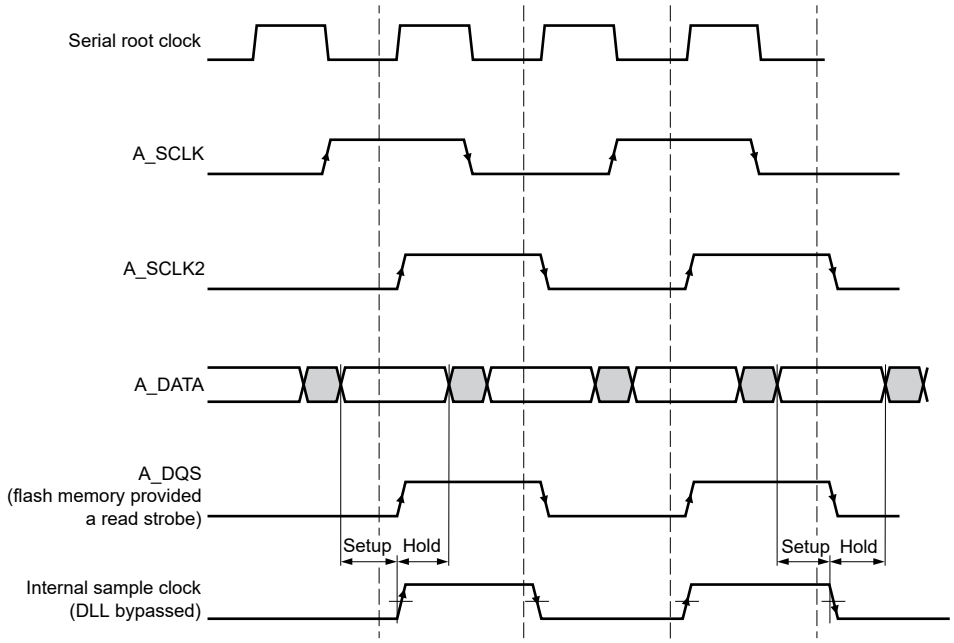


Figure 66. Input timing 1 for flash-memory-provided read strobe in DDR mode

- **Flash-memory-provided read strobe with SCLK**

  Certain flash devices provide both read data and read strobes with SCLK. Then the read strobe edge is aligned with the read data change. The FlexSPI controller delays the read strobe for one half cycle of the serial root clock (with DLL), then samples read data with the delayed strobe. See DLL configuration for sampling. Figure 67 and Figure 68 show the input timing for sampling with flash memory read strobe in SDR mode and DDR mode.

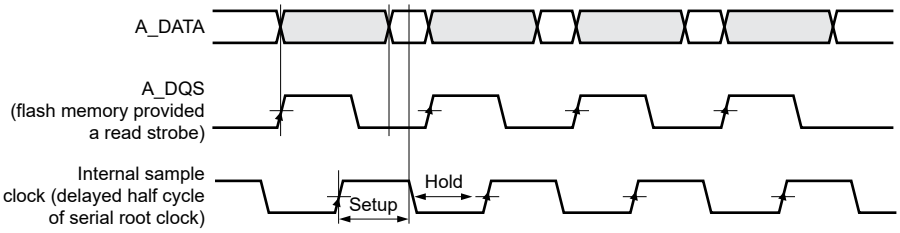


Figure 67. Input timing 2 for flash-memory-provided read strobe in SDR mode

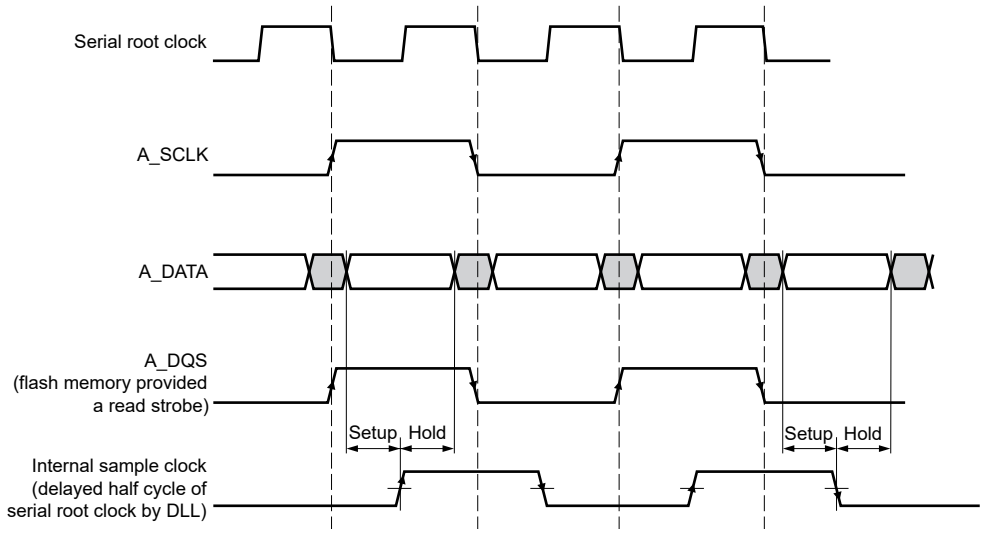


Figure 68. Input timing 2 for flash-memory-provided read strobe in DDR mode

19.3.15.6 DLL configuration for sampling

For the four sampling clock sources described previously, the input timing differs depending on the DLLxCR register configuration, according to the sampling clock source mode. The DLL is a delay-line chain that can be set to a fixed number of delay cells. It can also be auto-adjusted to lock on a certain phase delay to the reference clock.

- DLLxCR must be set to 0000_0100h (one fixed delay cell in DLL delay chain) in the following cases:
  - Sampling data with dummy read strobe looped back internally (MCR0[RXCLKSRC] = 0h)
  - Sampling data with dummy read strobe looped back from DQS pad (MCR0[RXCLKSRC] = 1h)
  - Sampling data with SCLK output looped back from SCLK pad (MCR0[RXCLKSRC] = 2h)
  - Sampling data with flash-memory-provided read strobe (MCR0[RXCLKSRC] = 3h), and flash memory provides the read strobe with SCLK2
- When data is sampled with a flash-memory-provided read strobe (MCR0[RXCLKSRC] = 3h) and flash memory provides the read strobe with SCLK, DLL must be configured as follows. These settings ensure a lock on a half cycle of the reference clock (serial root clock).
  - DLLxCR[SLVDLYTARGET] = Fh
  - DLLxCR[DLLEN] = 1
  - DLLxCR[OVRDEN] = 0
  - Other fields in DLLxCR must be kept at their reset values (all zeroes).

NOTE

If the serial root clock is slower than 100 MHz, DLL cannot lock on a half cycle of serial root clock. The delay cell number is limited in the delay chain. DLL must be configured as follows instead:

- DLLxCR[OVRDEN] = 1
- DLLxCR[OVRDVAL] = N

  Each delay cell in DLL is about 75 ps to 225 ps. The delay of DLL delay chain is (N × Delay_cell_delay). N is set based on the maximum DDR frequency (N ≤ equation). This value is calculated based on the MAX_DDR_FREQ parameter. N = 18. This value is a recommended value. If a failure occurs, the value might require adjustment in a real application.
- Other fields in DLLxCR must be kept at their reset values (all zeroes).

19.3.16 Data learning

The FlexSPI controller generates 16 clock phases (Phase 0–15) via delay cell line (DCL) with the selected sample clock. Clock phase 0 is the selected sample clock (DQS_IN) with no delay cell. 16 sampling blocks are implemented for Port A.

During a learn instruction, FlexSPI compares the sampled data bits with the internal data learn pattern in the Data Learning Pattern (DLPR) register and determines the correct sampling clock phase. The phase selection is automatically updated after a learn instruction and applied to subsequent read instructions or sequences.

When data learning is disabled (MCR0[LEARNEN] = 0), FlexSPI always uses clock phase 0 to sample FlexSPI data lines. Attempting to execute a learn instruction results in an error flag (INTR[IPCMDERR] or INTR[AHBCMDERR]).

FlexSPI supports data learning in both SDR mode and DDR mode. Data learning is not supported if the sampling clock source is a flash-memory-provided read strobe (MCR0[RXCLKSRC] = 3h).

Data learning can achieve high read frequency, but the flash memory input timing still limits the highest frequency. Flash memory must receive the command code and flash address bits.

When data learning is enabled, FlexSPI uses clock phase 0 after reset and before executing a learn instruction. The clock phase selection is updated after FlexSPI executes the learn instruction. The selected sample clock phase can be polled via STS0[DATALEARNPHASEA]. When data learning fails, the INTR[DATALEARNFAIL] interrupt flag is set, and the previous clock phase selection is kept. You can write 1 to MCR2[CLRLEARNPHASE] to reset the internal clock phase selection to clock phase 0.

19.3.16.1 Data learning with flash-memory-provided preamble bit

Certain flash devices support driving with preamble bits before driving read data in each read command sequence. Preamble bits are also called the data-learning pattern (DLP). The DLP is programmable via configuration register in the flash device.

For the specified read command sequence, flash memory returns preamble bits after dummy cycles and before returning read data.

For these flash devices, the operation flow with data learning is:

1. Configure the data learning pattern in Data Learning Pattern (DLPR) register.
2. Configure the same data learning pattern in the flash device by triggering an IP command.
3. Enable data learning in the flash device by triggering an IP command.
4. Configure LUT sequence with valid read command sequence containing the learn instruction.
5. Trigger a flash memory read command via AHB or IP command as normal.

NOTE

The first four steps are executed only once. They are not needed for every flash memory read command.

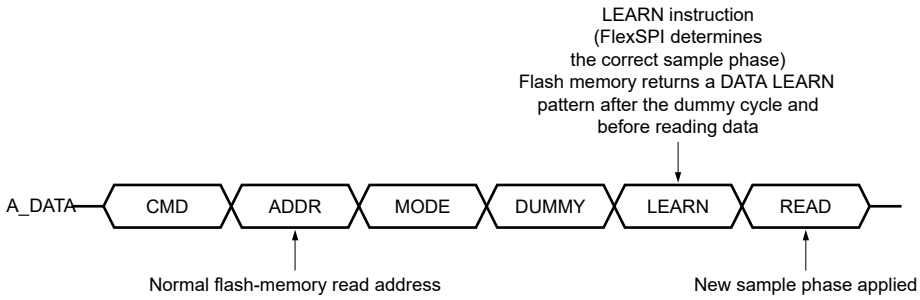


Figure 69. Data learning flow with flash-memory-provided preamble bit

19.3.16.2 Data learning without flash-memory-provided preamble bit

For flash devices that do not provide a preamble bit, the DLP cannot be returned automatically for each read command sequence. For these flash devices, the data learning sequence is:

1. Set data learning pattern in Data Learning Pattern (DLPR) register.
2. Reserve a certain flash memory area and program data (according to data learning pattern and flash memory access mode) to this area via IP command.
3. Configure LUT sequence with read sequence using a learn instruction instead of a read instruction. Trigger the read sequence to the reserved flash memory area via IP command.
4. Trigger the flash memory read command via AHB or IP command as normal. No learn instruction is needed in this read sequence.

NOTE

- The first three steps are executed only once. They are not needed for every flash memory read command.
- Step 4 should be executed with a certain interval to ensure that the internal sampling clock phase is adjusted in time.

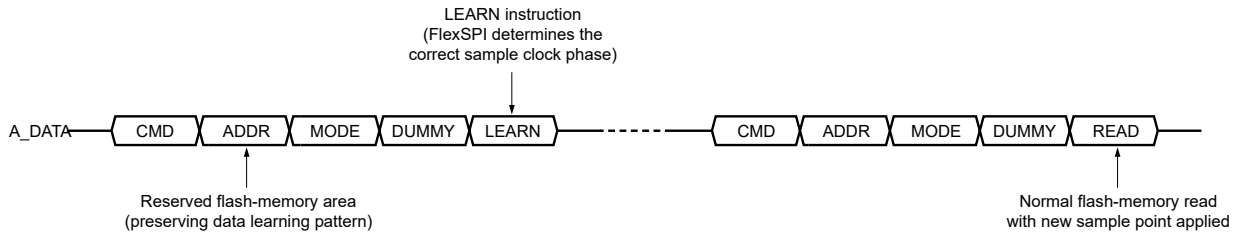


Figure 70. Data learning flow without flash-memory-provided preamble bit

NOTE

- For flash devices that do not provide a preamble bit, software overhead is required to read the reserved flash memory area at the interval.

These items determine the preprogramming data:

- Data Learning Pattern (DLPR register setting)
- DLP bit length
- Individual mode for read command
- Octal, Quad, Dual, or Single mode for read command

Figure 71 shows a preprogramming data example where the DLPR value is 43h, eight bits, and flash memory is read in Quad mode.

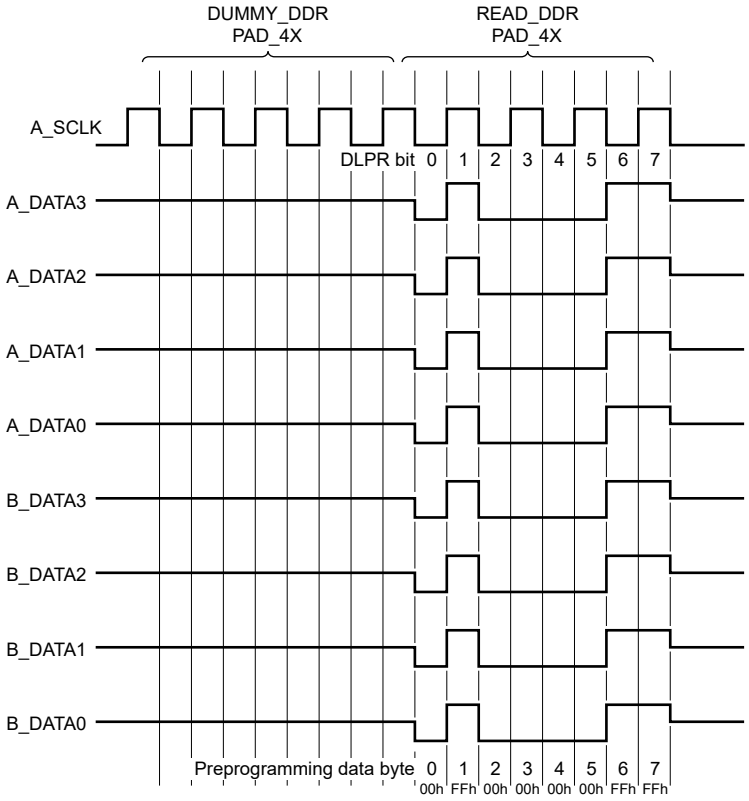


Figure 71. Preprogramming data for data learning

19.3.17 Execute-In-Place (XIP) enhanced mode

FlexSPI always supports XIP, regardless of whether external devices provide XIP enhanced mode. To support XIP, put program codes on an external device, then read or execute these codes directly via AHB read access to serial flash memory space. There is no configuration or status polling needed during AHB read access to the memory of the external device. The AHB receive buffer is fully transparent to software.

Certain devices provide XIP enhanced mode to improve code execution. In this mode, the command code is only sent in the first read instruction. It saves many cycles for command instructions and improves code execution. Devices enter or exit XIP enhanced mode via a special sequence that external devices specify. See the external device data sheet for more details.

Normally, a device enters XIP enhanced mode via this sequence:

1. Enable XIP enhanced mode in external flash device via IP command.
2. Send the first read sequence to external flash device with correct mode bits. Command code is included in this read sequence.
3. Send the subsequent read sequences to external flash device with correct mode bits. Command code is not included in these read sequences. Mode bits must be sent according to the flash specified; otherwise, the flash exits XIP enhanced mode.

The instruction JMP_ON_CS in FlexSPI must be used to support XIP enhanced mode in external devices. This instruction is only allowed in the AHB read command; otherwise, the instruction generates an IPCMDERR or AHBCDMERR interrupt (when the interrupt is enabled). When external devices do not support XIP enhanced mode, JMP_ON_CS must never be used. To support XIP enhanced mode:

- The first instruction in the read sequence must be the command instruction.
- The last valid instruction must be JMP_ON_CS (with operand 1h).

For the first AHB read command triggered, FlexSPI executes the instructions from the instruction pointer 0 in the sequence (which is the command instruction). After this sequence is executed, FlexSPI saves the operand in the JMP_ON_CS instruction as the start pointer for the next command to current device internally. For the next AHB read command triggered, FlexSPI executes from the instruction pointer 1h, bypassing the command instruction.

After XIP enhanced mode is started, it is possible to access flash via other commands like program commands. You must, however, use a special sequence to exit enhanced XIP mode before you can run a different command. Figure 72 indicates XIP operation with Flash XIP enhanced mode.

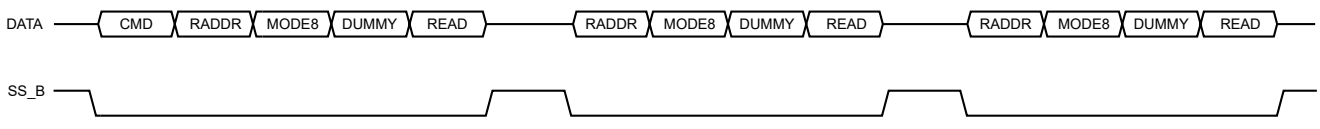


Figure 72. XIP Enhanced Mode Operation

19.3.18 Domain control

FLEXSPI supports two kinds of protection modes to manage nonsecure controller IP command access:

- IP command trigger
- LUT write

The chip must maintain domain control of the AHB region.

19.3.18.1 LUT write protection

LUTCR[PROTECT] controls LUT write protection. By default, no protection is applied on the LUT Control (LUTCR) register and LUT memory, and any controller can access the LUT freely.

When LUTCR[PROTECT] = 1, only a secure controller can change the value of LUTCR and the content of the LUT table. Unless hardware resets FlexSPI to return to the LUT write-protection-disabled state, nonsecure controllers cannot write to the LUT table or LUTCR register.

No error or interrupt occurs if a non-secure controller attempts to write to the protected LUT; the write operation is ignored.

19.4 External signals

This section provides the external signal information for the FlexSPI module.

Table 70. External Signal List

Signal name	Function	Direction	Description
A_SS0_B	Peripheral Chip Select Flash A1	O	Chip select for the serial flash device A1
A_SS1_B	Peripheral Chip Select Flash A2	O	Chip select for the serial flash device A2
A_SCLK	Serial Clock Flash A	O	Serial clock output to the serial flash device A. This clock runs at half the frequency of serial clock root in DDR mode, and at the same frequency as serial clock root in SDR mode. Clock

Table continues on the next page...

Table 70. External Signal List (continued)

Signal name	Function	Direction	Description
			output toggles during the entire flash memory access sequence.
A_SCLK2	Serial Clock 2 Flash A	O	Serial clock output 2 to the serial flash device A. A_SCLK2 is 90-degree phase shifted from A_SCLK in DDR mode. No clock output in SDR mode (clock gated). See FlexSPI input timing.
A_DATAn	Serial I/O Flash A	I/O	Data I/O lines to and from the serial flash device A
A_DQS	Data Strobe Signal Flash A	I/O	Data strobe signal for port A. This signal has three functions: • Driven with read strobe by external device. Some flash devices provide the read strobe signal together with read data. In this case, if the external device drives this pad only when reading flash memory data, this pad may require a **pull-down** resistor. • Driven with latency information by external device. Some devices use this pin to indicate the dummy cycles needed (before program or read data transfer) such as HyperRAM or HyperFlash. • Loopback dummy read strobe. The FlexSPI controller provides internal dummy read strobe for flash memory read data. Higher read frequency can be achieved by looping back this dummy read strobe from the pad. This pin can be floated or have some capacitive load added at the board level to compensate for load on DATA or SCLK pins.

19.5 Initialization

The FlexSPI controller initialization sequence is:

1. Enable controller clocks (AHB clock, IP bus clock, or serial root clock) at system level.
2. To enter module stop mode, write 1 to MCR0[MDIS].
3. Configure module control registers: MCR0, MCR1, and MCR2. Do not change MCR0[MDIS].
4. When AHB commands are used, configure AHB Bus Control (AHBCR) and AHB Receive Buffer Control (AHBRXBUFxCR0) registers.
5. Configure flash control registers (FLSHxCR0, FLSHxCR1, or FLSHxCR2) according to external device type.
6. Configure DLL control register (DLLxCR) according to sample clock source selection.

7. To exit module stop mode, write 0 to MCR0[MDIS].
8. Configure LUT as needed for AHB command or IP command.
9. Optionally, to reset controller, write 1 to MCR0[SWRESET].

External devices must be configured via IP command normally after initialization. For example, the WRITE STATUS command performs device configuration for most serial NOR flash memory.

19.6 Application information

This section describes applications that FlexSPI supports.

19.6.1 Application on serial NOR flash device

This section provides the example LUT instruction sequences for serial NOR flash device (Cypress Flash S25FS128S).

19.6.1.1 Write enable command

Table 71 shows the WRITE ENABLE command sequence.

Table 71. WRITE ENABLE command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h	06h	Command name: WREN
1-7	STOP (00h)	0h	00h	

19.6.1.2 Write registers command

Table 72 shows the Write Registers command sequence.

Table 72. Write Registers command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	01h	Command name: WRR
1	WRITE_SDR	0h (Single)	1h or 2h	Data size is 1 byte or 2 bytes. Byte 0 is write data for Status Register 1; Byte 1 is write data for Configuration Register 1. IPCR1[IDATSZ] can override this value.
2-7	STOP (00h)	0h	00h	

19.6.1.3 Page Program command

Table 73 shows Page Program command sequence.

Table 73. PAGE PROGRAM command (Cypress serial NOR flash)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	02h or 12h	Command name: PP or 4PP

Table continues on the next page...

Table 73. PAGE PROGRAM command (Cypress serial NOR flash) (continued)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
				PP is 3-byte address mode. 4PP is 4-byte address mode.
1	ADDR_SDR	0h (Single)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	WRITE_SDR	0h (Single)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default programming data size. This value is ignored for AHB command.
3-7	STOP (00h)	0h	00h	

Table 74 shows Page Program command sequence (QPI mode).

Table 74. PAGE PROGRAM (QPI mode) command (Cypress serial NOR flash)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	2h (Quad)	02h or 12h	Command name: PP or 4PP PP is 3-byte address mode. 4PP is 4-byte address mode.
1	ADDR_SDR	2h (Quad)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	WRITE_SDR	2h (Quad)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default programming data size. This value is ignored for AHB command.
3-7	STOP (00h)	0h	00h	

19.6.1.4 Read Status 1 command

Table 75 shows READ STATUS 1 command sequence.

Table 75. READ STATUS 1 command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	05h	Command name: RDSR1
1	READ_SDR	0h (Single)	1h	1 byte for Status register 1
2-7	STOP (00h)	0h	00h	

19.6.1.5 Read command

Table 76 shows READ command sequence.

Table 76. READ command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	03h or 13h	Command name: READ or 4READ READ is 3-byte address mode. 4READ is 4-byte address mode.
1	ADDR_SDR	0h (Single)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	READ_SDR	0h (Single)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.
3-7	STOP (00h)	0h	00h	

19.6.1.6 Fast Read command

Table 77 shows Fast Read command sequence. This table shows Cypress SPI Configuration Register bits CR2V[7] = 0, CR2V[3:0] = 8h.

Table 77. FAST_READ command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	0Bh or 0Ch	Command name: FAST_READ or 4FAST_READ FAST_READ is 3-byte address mode. 4FAST_READ is 4-byte address mode.
1	ADDR_SDR	0h (Single)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	DUMMY_SDR	0h (Single)	08h	Dummy cycle is 8 (in serial root clock) as CR2V[3:0] = 8h.
3	READ_SDR	0h (Single)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.
4-7	STOP (00h)	0h	00h	

19.6.1.7 Dual IO Fast Read command

Table 78 shows Dual IO FAST_READ command sequence. This table shows Cypress SPI Configuration Register bits CR2V[7] = 0, CR2V[3:0] = 8h, Continuous Read mode.

Table 78. Dual IO FAST_READ command (Continuous Read mode)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	BBh or BCh	Command name: DIOR or 4DIOR DIOR is 3-byte address mode. 4DIOR is 4-byte address mode.
1	ADDR_SDR	1h (Dual)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	MODE8_SDR	1h (Dual)	Axh	Enter continuous read mode or remain in continuous read mode.
3	DUMMY_SDR	1h (Dual)	08h	Dummy cycle is 8 (in serial root clock) as CR2V[3:0] = 8h.
4	READ_SDR	1h (Dual)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.
5	JMP_ON_CS	0h (or Don't care)	01h	CMD instruction is bypassed after the first read access.
6-7	STOP (00h)	0h	00h	

Table 79 shows Dual IO FAST_READ command sequence. This table shows Cypress SPI Configuration Register bits CR2V[7] = 0, CR2V[3:0] = 8h, Noncontinuous Read mode.

Table 79. Dual IO FAST_READ command (Noncontinuous Read mode)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	BBh or BCh	Command name: DIOR or 4DIOR DIOR is 3-byte address mode. 4DIOR is 4-byte address mode.
1	ADDR_SDR	1h (Dual)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	MODE8_SDR	1h (Dual)	Any value other than Axh	Exit continuous read mode or remain in noncontinuous read mode.
3	DUMMY_SDR	1h (Dual)	08h	Dummy cycle is 8 (in serial root clock) as CR2V[3:0] = 8h.
4	READ_SDR	1h (Dual)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.
5-7	STOP (00h)	0h	00h	

19.6.1.8 Quad IO Fast Read command

Table 80 shows Quad IO FAST_READ command sequence. This table shows Cypress SPI Configuration Register bits CR2V[7] = 0, CR2V[6] = 0, CR2V[3:0] = 8h, Non-QPI mode, Noncontinuous read mode.

Table 80. Quad IO FAST_READ command (Non-QPI mode, Noncontinuous read mode)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	EBh or ECh	Command name: QIOR or 4QIOR QIOR is 3-byte address mode. 4QIOR is 4-byte address mode.
1	ADDR_SDR	2h (Quad)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	MODE8_SDR	2h (Quad)	Any value other than Axh	Exit continuous read mode or remain in noncontinuous read mode.
3	DUMMY_SDR	2h (Quad)	08h	Dummy cycle is 8 (in serial root clock) as CR2V[3:0] = 8h.
4	READ_SDR	2h (Quad)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.
5-7	STOP (00h)	0h	00h	

Table 81 shows Quad IO FAST_READ command sequence. This table shows Cypress SPI Configuration Register bits CR2V[7] = 0, CR2V[6] = 0, CR2V[3:0] = 8h, Non-QPI mode, Continuous read mode.

Table 81. Quad IO FAST_READ command (Non-QPI mode, Continuous read mode)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	EBh or ECh	Command name: QIOR or 4QIOR QIOR is 3-byte address mode. 4QIOR is 4-byte address mode.
1	ADDR_SDR	2h (Quad)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	MODE8_SDR	2h (Quad)	A0h	Enter continuous read mode or remain in continuous read mode.
3	DUMMY_SDR	2h (Quad)	08h	Dummy cycle is 8 (in serial root clock) as CR2V[3:0] = 8h.
4	READ_SDR	2h (Quad)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.

Table continues on the next page...

Table 81. Quad IO FAST_READ command (Non-QPI mode, Continuous read mode) (continued)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
5	JMP_ON_CS	0h (or Don't care)	01h	CMD instruction is bypassed after the first read access.
6-7	STOP (00h)	0h	00h	

Table 82 shows Quad IO FAST_READ command sequence. This table shows Cypress SPI Configuration Register bits CR2V[7] = 0, CR2V[6] = 1, CR2V[3:0] = 8h, QPI mode, Continuous read mode.

Table 82. Quad IO FAST_READ command (QPI mode, Continuous read mode)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	2h (Quad)	EBh or ECh	Command name: QIOR or 4QIOR QIOR is 3-byte address mode. 4QIOR is 4-byte address mode.
1	ADDR_SDR	2h (Quad)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	MODE8_SDR	2h (Quad)	A0h	Enter continuous read mode or remain in continuous read mode.
3	DUMMY_SDR	2h (Quad)	08h	Dummy cycle is 8 (in serial root clock) as CR2V[3:0] = 8h.
4	READ_SDR	2h (Quad)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.
5	JMP_ON_CS	0h (or Don't care)	01h	CMD instruction is bypassed after the first read access.
6-7	STOP (00h)	0h	00h	

19.6.1.9 DDR Quad IO Fast Read command

Table 83 shows DDR Quad IO FAST_READ command sequence. This table shows Cypress SPI Configuration Register bits CR2V[7] = 0, CR2V[6] = 0, CR2V[3:0] = 8h, Non-QPI mode, Non-continuous read mode.

Table 83. DDR Quad IO FAST_READ command (Non-QPI mode, Noncontinuous read mode)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	EDh or EEh	Command name: QIOR_DDR or 4QIOR_DDR QIOR_DDR is 3-byte address mode. 4QIOR_DDR is 4-byte address mode.

Table continues on the next page...

Table 83. DDR Quad IO FAST_READ command (Non-QPI mode, Noncontinuous read mode) (continued)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
1	ADDR_DDR	2h (Quad)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	MODE8_DDR	2h (Quad)	Any value other than Axh	Exit continuous read mode or keep in noncontinuous read mode.
3	DUMMY_DDR	2h (Quad)	08h	Dummy cycle is 8 (in serial root clock) as CR2V[3:0] = 8h.
4	LEARN_DDR	2h (Quad)	1h	DLP is 8 bits.
5	READ_DDR	2h (Quad)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.
6-7	STOP (00h)	0h	00h	

Table 84 shows DDR Quad IO FAST_READ command sequence. This table shows Cypress SPI Configuration register bits CR2V[7] = 0, CR2V[6] = 0, CR2V[3:0] = 8h, Non-QPI mode, Continuous read mode.

Table 84. DDR Quad IO FAST_READ command (Non-QPI mode, Continuous read mode)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	EDh or EEh	Command name: QIOR_DDR or 4QIOR_DDR QIOR_DDR is 3-byte address mode. 4QIOR_DDR is 4-byte address mode.
1	ADDR_DDR	2h (Quad)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	MODE8_DDR	2h (Quad)	A0h	Enter continuous read mode or remain in continuous read mode.
3	DUMMY_DDR	2h (Quad)	08h	Dummy cycle is 8 (in serial root clock) as CR2V[3:0] = 8h.
4	LEARN_DDR	2h (Quad)	1h	DLP is 8 bits.
5	READ_DDR	2h (Quad)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.
6	JMP_ON_CS	0h (or Don't care)	01h	CMD instruction is bypassed after the first read access.
7	STOP (00h)	0h	00h	

Table 85 shows DDR Quad IO FAST_READ command sequence. This table shows Cypress SPI Configuration Register bits CR2V[7] = 0, CR2V[6] = 1, CR2V[3:0] = 8h, QPI mode, Continuous read mode.

Table 85. DDR Quad IO FAST_READ command (QPI mode, Continuous read mode)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	2h (Quad)	EDh or EEh	Command name: QIOR_DDR or 4QIOR_DDR QIOR_DDR is 3-byte address mode. 4QIOR_DDR is 4-byte address mode.
1	ADDR_DDR	2h (Quad)	18h or 20h	Address bit number (operand value) should be 24 in 3-byte address mode and 32 in 4-byte address mode.
2	MODE8_DDR	2h (Quad)	A0h	Enter continuous read mode or remain in continuous read mode.
3	DUMMY_DDR	2h (Quad)	08h	Dummy cycle is 8 (in serial root clock) as CR2V[3:0] = 8h.
4	LEARN_DDR	2h (Quad)	1h	DLP is 8 bits.
5	READ_DDR	2h (Quad)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.
6	JMP_ON_CS	0h (or Don't care)	01h	CMD instruction is bypassed after the first read access.
7	STOP (00h)	0h	00h	

19.6.2 Application on HyperBus device

This section provides the example LUT instruction sequences for HyperBus device (Cypress RPC flash, HyperRam, or HyperFlash).

19.6.2.1 HyperFlash

This section provides example sequences for HyperFlash devices (Cypress S26KS series).

Table 86 shows the read status command sequence.

Table 86. Read status command

Seq num	Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0 (Write - Addr = 555h, Data = 70h)	0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W# = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	CMD_DDR	3h (Octal)	00h	Row Address: 0000AAh (24 bit)
	2	CMD_DDR	3h (Octal)	00h	
	3	CMD_DDR	3h (Octal)	AAh	

Table continues on the next page...

Table 86. Read status command (continued)

Seq num	Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
	4	CMD_DDR	3h (Octal)	00h	Column Address: 05h (13 zero bits + 3 valid bits)
	5	CMD_DDR	3h (Octal)	05h	
	6	CMD_DDR	3h (Octal)	00h	Write Data: 0070h
	7	CMD_DDR	3h (Octal)	70h	
1 (Read - Addr = xxx, Data = Status register data)	0	CMD_DDR	3h (Octal)	A0h	CA bit 47: (R/W#) = 1h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	RADDR_DDR	3h (Octal)	18h	Row Address: 24 bits
	2	CADDR_DDR	3h (Octal)	10h	Column Address: (13 zero bits + 3 valid bits)
	3	DUMMY_RWDS_DDR	3h (Octal)	0Bh	When latency count = 11
	4	READ_DDR	0h (Octal)	4h	4-Byte read
	5-7	STOP (00h)	0h	00h	

Table 87 shows the read (memory) command sequence.

Table 87. Read (memory) command

Seq num	Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0 (Read - Addr = xxx, Data = memory data)	0	CMD_DDR	3h (Octal)	A0h	CA bit 47: (R/W#) = 1h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	RADDR_DDR	3h (Octal)	18h	Row Address: 24 bits
	2	CADDR_DDR	3h (Octal)	10h	Column Address: (13 zero bits + 3 valid bits)
	3	DUMMY_RWDS_DDR	3h (Octal)	0Bh	When latency count = 11
	4	READ_DDR	3h (Octal)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size. This value is ignored for AHB command.
	5-7	STOP (00h)	0h	00h	

Table 88 shows the word program command sequence.

Table 88. Word program command

Seq num	Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0 (Write - Addr = 555h, Data = AAh)	0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	CMD_DDR	3h (Octal)	00h	Row Address: 0000AAh (24 bit)
	2	CMD_DDR	3h (Octal)	00h	
	3	CMD_DDR	3h (Octal)	AAh	
	4	CMD_DDR	3h (Octal)	00h	Column Address: 05h (13 zero bits + 3 valid bits)
	5	CMD_DDR	3h (Octal)	05h	
	6	CMD_DDR	3h (Octal)	00h	Write Data: 00AAh
	7	CMD_DDR	3h (Octal)	AAh	
1 (Write - Addr = 2AAh, Data = 55h)	0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	CMD_DDR	3h (Octal)	00h	Row Address: 000055h (24 bit)
	2	CMD_DDR	3h (Octal)	00h	
	3	CMD_DDR	3h (Octal)	55h	
	4	CMD_DDR	3h (Octal)	00h	Column Address: 02h (13 zero bits + 3 valid bits)
	5	CMD_DDR	3h (Octal)	02h	
	6	CMD_DDR	3h (Octal)	00h	Write Data: 0055h
	7	CMD_DDR	3h (Octal)	55h	
2 (Write - Addr = 555h, Data = A0h)	0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	CMD_DDR	3h (Octal)	00h	Row Address: 0000AAh (24 bit)
	2	CMD_DDR	3h (Octal)	00h	
	3	CMD_DDR	3h (Octal)	AAh	
	4	CMD_DDR	3h (Octal)	00h	Column Address: 05h (13 zero bits + 3 valid bits)
	5	CMD_DDR	3h (Octal)	55h	
	6	CMD_DDR	3h (Octal)	00h	Write Data: 00A0h

Table continues on the next page...

Table 88. Word program command (continued)

Seq num	Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
	7	CMD_DDR	3h (Octal)	A0h	
3 (Word Program)	0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	RADDR_DDR	3h (Octal)	18h	Row Address: 24 bits
	2	CADDR_DDR	3h (Octal)	10h	Column Address: (13 zero bits + 3 valid bits)
	3	WRITE_DDR	3h (Octal)	02h	2-Byte written data
	4-7	STOP (0h)	0h	00h	

Table 89 shows Write-to-Buffer and Program-Buffer-to-Flash command sequence.

Table 89. Write-to-Buffer and Program-Buffer-to-Flash command

Seq num	Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0 (Write - Addr = 555h, Data = AAh)	0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	CMD_DDR	3h (Octal)	00h	Row Address: 0000AAh (24 bit)
	2	CMD_DDR	3h (Octal)	00h	
	3	CMD_DDR	3h (Octal)	AAh	
	4	CMD_DDR	3h (Octal)	00h	Column Address: 05h (13 zero bits + 3 valid bits)
	5	CMD_DDR	3h (Octal)	05h	
	6	CMD_DDR	3h (Octal)	00h	Write Data: 00AAh
	7	CMD_DDR	3h (Octal)	AAh	
1 (Write - Addr = 2AAh, Data = 55h)	0	CMD_DDR	0h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	CMD_DDR	3h (Octal)	00h	Row Address: 000055h (24 bit)
	2	CMD_DDR	3h (Octal)	00h	
	3	CMD_DDR	3h (Octal)	55h	

Table continues on the next page...

Table 89. Write-to-Buffer and Program-Buffer-to-Flash command (continued)

Seq num	Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
	4	CMD_DDR	3h (Octal)	00h	Column Address: 02h (13 zero bits + 3 valid bits)
	5	CMD_DDR	3h (Octal)	02h	
	6	CMD_DDR	3h (Octal)	00h	Write Data: 0055h
	7	CMD_DDR	3h (Octal)	55h	
2 (Write - Addr = SA, Data = 25h)	0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	RADDR_DDR	3h (Octal)	18h	Row Address: SA (24 bit) SA is sector address. Write SA to IPCR0[SFAR].
	2	CADDR_DDR	3h (Octal)	10h	Column Address: 13 zero bits + 3 valid bits
	3	CMD_DDR	3h (Octal)	00h	Write Data: 0025h
	4	CMD_DDR	3h (Octal)	25h	
2 (Write - Addr = SA, Data = WC)	0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	RADDR_DDR	3h (Octal)	18h	Row Address: SA (24 bit) SA is sector address. Write SA to IPCR0[SFAR].
	2	CADDR_DDR	3h (Octal)	10h	Column Address: 13 zero bits + 3 valid bits
	3	CMD_DDR	3h (Octal)	WC	Write Data: WC WC is word count.
	4	CMD_DDR	3h (Octal)		
3 - N (Write - Addr = WBL, Data = PD) N is the word count + 2	0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	RADDR_DDR	3h (Octal)	18h	Row Address: WBL (24 bit) WBL is write buffer location. Write WBL to IPCR0[SFAR].

Table continues on the next page...

Table 89. Write-to-Buffer and Program-Buffer-to-Flash command (continued)

Seq num	Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
	2	CADDR_DDR	3h (Octal)	10h	Column Address: 13 zero bits + 3 valid bits
	3	WRITE_DDR	3h (Octal)	02h	2-Byte write data
	4-7	STOP (0h)	0h	00h	
N + 1 (Write - Addr = SA, Data = 29) Program Buffer to Flash	0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
	1	RADDR_DDR	3h (Octal)	18h	Row Address: SA (24 bit) SA is sector address. Write SA to IPCR0[SFAR].
	2	CADDR_DDR	3h (Octal)	10h	Column Address: 13 zero bits + 3 valid bits
	3	CMD_DDR	3h (Octal)	00h	Write Data: 29h
	4	CMD_DDR	3h (Octal)	29h	
	5-7	STOP (0h)	0h	00h	

19.6.2.2 HyperRAM

This section provides example sequences for HyperRAM (Cypress S27KL series).

The Read (memory) command sequence is same as HyperFlash. Table 90 shows the Write (memory) command sequence.

Table 90. Write (memory) command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_DDR	3h (Octal)	20h	CA bit 47: (R/W#) = 0h CA bit 46: (Target) = 0h CA bit 45: (Burst Type) = 1h CA bit 44-40: All reserved = 0h
1	RADDR_DDR	3h (Octal)	18h	Row Address: 24 bits
2	CADDR_DDR	3h (Octal)	10h	Column Address: (13 zero bits + 3 valid bits)
3	DUMMY_RWDS_DDR	3h (Octal)	0Bh	When latency count = 11
4	WRITE_DDR	3h (Octal)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default writing

Table continues on the next page...

Table 90. Write (memory) command (continued)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
				data size. This value is ignored for AHB command.
5-7	STOP (00h)	0h	00h	

19.6.3 Application on Serial NAND flash device

This section provides example LUT instruction sequences for serial NAND flash device (Micron Flash MT29 series). The operation of serial NAND flash is similar to serial NOR flash.

The read operation sequence is:

- Page read. Transfer the data from the NAND flash array to the cache register.
- Get feature to read the status.
- Random data read

Table 91 shows the page read command sequence.

Table 91. Page Read command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	1h (Single)	13h	Command code: 13h
1	RADDR_SDR	1h (Single)	18h	Row address: 24 bit
2-7	STOP (0h)	0h	00h	

Table 92 shows Get Feature command sequence.

Table 92. Get Feature command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0h (Single)	0Fh	Command code: 0Fh
1	CMD_SDR	1h (Single)	C0h	Status register address (C0h)
2	READ_SDR	0x1 (Single)	02h	2 bytes read data
3-7	STOP (0h)	0h	00h	

Table 93 shows Random Data Read command sequence.

Table 93. Read from Cache x4 command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	1h (Single)	6Bh	Command code: 6Bh
1	MODE4_SDR	1h (Single)	0h or 1h	If plane selection is one, software should decode the flash address and write 1h to mode bits. If plane selection

Table continues on the next page...

Table 93. Read from Cache x4 command (continued)

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
				is zero, software should write 0h to mode bits. Plane selection bit is 18th bit of flash address. If NAND flash size is less than 4 Gbit, plane selection is always zero.
2	CADDR_SDR	1h (Single)	0Ch	Column address: 12 bit
3	DUMMY_SDR	2h (Quad)	08h	Dummy cycle number: 8 (serial root clock)
4	READ_SDR	2h (Quad)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default reading data size.
5-7	STOP (0x0)	0x0	0x00	

The program operation sequence is:

- Write enable
- Program Load. Transfer the write data to the cache register.
- Program Execute
- Get Feature to read the status.

Table 94 shows Program Load command sequence.

Table 94. Program Load command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	1h (Single)	02h	Command code: 02h
1	MODE4_SDR	1h (Single)	0h or 1h	If plane selection is one, software should decode the flash address and write 1h to mode bits. If plane selection is zero, software should write 0h to mode bits. Plane selection bit is 18th bit of flash address. If NAND flash size is less than 4 Gbit, plane selection is always zero.
2	CADDR_SDR	1h (Single)	0Ch	Column address: 12 bit
3	WRITE_SDR	1h (Single)	Any nonzero value	If IPCR1[IDATSZ] is zero, this operand value can be used as default writing data size.
4-7	STOP (0h)	0h	00h	

Table 95 shows Program Execute command sequence.

Table 95. Program Execute command

Instruction number	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	1h (Single)	10h	Command code: 10h
1	RADDR_SDR	1h (Single)	18h	Row Address: 24 bit
2-7	STOP (0h)	0h	00h	

19.6.4 Application on FPGA device

An FPGA device must be accessed via AHB command. All AHB accesses to FPGA are transparent to the software driver (no software intervention). An FPGA device may have some special requirements.

Table 96. Special requirements for FPGA devices

Condition	How to manage the condition
Device type may be different on A1, A2, B1, or B2.	Write 0 to MCR2[SAMEDEVICEEN] and configure the FLSHxCR0 and FLSHxCR1 registers separately for up to four external devices.
Device requires different wait cycle for programming.	The AHB write wait cycle number can be set separately for these four external devices (via FLSHxCR2[AWRWAIT]). Software can configure the sequences in LUT with different dummy instructions (operand determines dummy cycle). FlexSPI holds the AHB bus ready for this wait time; AHB bus performance may decrease when this wait time is very long.
Device requires different wait cycle for reading.	The AHB read sequence index and sequence number can be set separately for these four external devices (via FLSHxCR2[ARDSEQID] and FLSHxCR2[ARDSEQNUM]). Software can configure the sequences in LUT with different dummy instructions (operand determines dummy cycle).
Device may be sensitive to read instruction clock cycle number.	If its internal memory is implemented similar to a FIFO, the device is sensitive to the read instruction clock cycle number. Software can send the data size information to the external device via DATSZ instruction. The FPGA device decodes the data size information and determines how many data bytes must be popped.
Device may require interval time between chip select valid.	This interval can be managed via FLSHxCR1[CSINTERVAL].
Device may use SCLK as reference clock for its internal PLL.	SCLK must be free-running and the clock frequency must be stable. To achieve this configuration, write 1 to MCR0[SCKFREERUNEN] and only use SDR sequences.

19.6.5 Overview of error categories, flags, and triggered sources

Table 97 provides an overview of the categories, flags, and triggered sources of errors in FlexSPI.

Table 97. Error category, triggered sources, and flags

Error Category	Triggered Sources	Description	Error Flags
Command grant error	AHB write command	Command grant timeout	INTR[AHBCMDGE] is set. AHB bus error response

Table continues on the next page...

Table 97. Error category, triggered sources, and flags (continued)

Error Category	Triggered Sources	Description	Error Flags
	AHB read command		INTR[AHBCMDGE] is set. AHB bus error response
	IP command		INTR[IPCMDGE] is set.
Command check error	AHB write command	• AHB write command with JMP_ON_CS instruction used in the sequence • Unknown instruction opcode in the sequence. • Instruction DUMMY_SDR or DUMMY_RWDS_SDR used in DDR sequence. • Instruction DUMMY_DDR or DUMMY_RWDS_DDR used in SDR sequence.	INTR[AHBCMDERR] is set. Command is not executed when an error is detected in command check
	AHB read command	• Unknown instruction opcode in the sequence. • Instruction DUMMY_SDR or DUMMY_RWDS_SDR used in DDR sequence. • Instruction DUMMY_DDR or DUMMY_RWDS_DDR used in SDR sequence.	INTR[AHBCMDERR] is set. Command is not executed when an error is detected in command check
	IP command	• IP command with JMP_ON_CS instruction used in the sequence. • Unknown instruction opcode in the sequence • Instruction DUMMY_SDR or DUMMY_RWDS_SDR used in DDR sequence. • Instruction DUMMY_DDR or DUMMY_RWDS_DDR used in SDR sequence. • Flash boundary across	INTR[IPCMDERR] is set. Command is not executed when an error is detected in command check.
Command execution error	AHB write command	Command timeout during execution	INTR[AHBCMDERR] is set. INTR[SEQTIMEOUT] is set.

Table continues on the next page...

Table 97. Error category, triggered sources, and flags (continued)

Error Category	Triggered Sources	Description	Error Flags
			An AHB bus error response occurs, except in following cases: • Flush triggers AHB write command (INCR burst ended with AHB_TX_BUF not empty). • AHB bufferable write access and bufferable enabled (AHBCR[BUFFERABLEEN] = 1).
	AHB read command		INTR[AHBCMDERR] is set. INTR[SEQTIMEOUT] is set. AHB bus error response
	IP command		INTR[IPCMDERR] is set. INTR[SEQTIMEOUT] is set.
AHB bus timeout	AHB write command	AHB bus timeout (no bus ready return)	INTR[AHBBUSTIMEOUT] is set. AHB bus error response
	AHB read command		
Data learning failed	Any command	No valid sample clock phase found after learn instruction is executed.	INTR[DATALEARNFAIL] is set.
Security error	IP command	Nonsecure controller requests IP command access to a secure region	INTR[IPCMDSECUREVIO] is set.

19.7 Memory map and register definition

This section includes the FlexSPI module memory map and detailed descriptions of all registers.

19.7.1 Register access

All registers can be accessed with 8-bit, 16-bit, and 32-bit width operations. Never change the values of reserved fields in control registers. Changing the values of reserved fields may impact the normal functioning of the controller.

NOTE

When FlexSPI can access sensitive memory contents, the FlexSPI controller should protect those contents using resource isolation such as XRDC or TrustZone, or equivalent partition control via SCFW. Ensure that only trusted software is allowed to access the FlexSPI controller register interface.

19.7.2 FlexSPI register descriptions

19.7.2.1 FlexSPI memory map

FLEXSPI0 base address: 400C_0000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Module Control 0 (MCR0)	32	See section	FFFF_80C2
4	Module Control 1 (MCR1)	32	RW	FFFF_FFFF
8	Module Control 2 (MCR2)	32	See section	2000_81F7
C	AHB Bus Control (AHBCR)	32	See section	0000_0018
10	Interrupt Enable (INTEN)	32	See section	0000_0000
14	Interrupt (INTR)	32	See section	0000_0000
18	LUT Key (LUTKEY)	32	RW	5AF0_5AF0
1C	LUT Control (LUTCR)	32	See section	0000_0002
20	AHB Receive Buffer 0 Control 0 (AHBRXBUF0CR0)	32	See section	8000_0010
24	AHB Receive Buffer 1 Control 0 (AHBRXBUF1CR0)	32	See section	8001_0010
28	AHB Receive Buffer 2 Control 0 (AHBRXBUF2CR0)	32	See section	8002_0010
2C	AHB Receive Buffer 3 Control 0 (AHBRXBUF3CR0)	32	See section	8003_0010
30	AHB Receive Buffer 4 Control 0 (AHBRXBUF4CR0)	32	See section	8004_0010
34	AHB Receive Buffer 5 Control 0 (AHBRXBUF5CR0)	32	See section	8005_0010
38	AHB Receive Buffer 6 Control 0 (AHBRXBUF6CR0)	32	See section	8006_0010

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
3C	AHB Receive Buffer 7 Control 0 (AHBRXBUF7CR0)	32	See section	8007_0010
60	Flash Control 0 (FLSHA1CR0)	32	See section	0001_0000
64	Flash Control 0 (FLSHA2CR0)	32	See section	0001_0000
70	Flash Control 1 (FLSHA1CR1)	32	RW	0000_0063
74	Flash Control 1 (FLSHA2CR1)	32	RW	0000_0063
80	Flash Control 2 (FLSHA1CR2)	32	See section	0000_0000
84	Flash Control 2 (FLSHA2CR2)	32	See section	0000_0000
94	Flash Control 4 (FLSHCR4)	32	See section	0000_0000
A0	IP Control 0 (IPCR0)	32	RW	0000_0000
A4	IP Control 1 (IPCR1)	32	See section	0000_0000
B0	IP Command (IPCMD)	32	See section	0000_0000
B4	Data Learning Pattern (DLPR)	32	RW	0000_0000
B8	IP Receive FIFO Control (IPRXFCR)	32	See section	0000_0000
BC	IP Transmit FIFO Control (IPTXFCR)	32	See section	0000_0000
C0	DLL Control 0 (DLLACR)	32	See section	0000_0100
C4	DLL Control 0 (DLLBCR)	32	See section	0000_0100
E0	Status 0 (STS0)	32	RO	0000_0002
E4	Status 1 (STS1)	32	RO	0000_0000
E8	Status 2 (STS2)	32	RO	0100_0100
EC	AHB Suspend Status (AHBSPNDSTS)	32	RO	0000_0000
F0	IP Receive FIFO Status (IPRXFSTS)	32	RO	0000_0000
F4	IP Transmit FIFO Status (IPTXFSTS)	32	RO	0000_0000
100 - 17C	IP Receive FIFO Data a (RFDR0 - RFDR31)	32	RO	0000_0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
180 - 1FC	IP TX FIFO Data a (TFDR0 - TFDR31)	32	WO	0000_0000
200 - 2FC	Lookup Table a (LUT0 - LUT63)	32	RW	See section
420	HADDR REMAP Start Address (HADDRSTART)	32	See section	0000_0000
424	HADDR REMAP END ADDR (HADDREND)	32	See section	0000_0000
428	HADDR Remap Offset (HADDROFFSET)	32	See section	0000_0000
42C	IPED Function Control (IPEDCTRL)	32	See section	0000_0000
500	IPED context control 0 (IPEDCTXCTRL0)	32	See section	5555_5555
504	IPED context control 1 (IPEDCTXCTRL1)	32	See section	AAAA_AAAA
520	IPED Context0 IV0 (IPEDCTX0IV0)	32	RW	0000_0000
524	IPED Context0 IV1 (IPEDCTX0IV1)	32	RW	0000_0000
528	Start Address of Region (IPEDCTX0START)	32	See section	0000_0000
52C	End Address of Region (IPEDCTX0END)	32	See section	0000_0000
540	IPED Context1 IV0 (IPEDCTX1IV0)	32	RW	0000_0000
544	IPED Context1 IV1 (IPEDCTX1IV1)	32	RW	0000_0000
548	Start Address of Region (IPEDCTX1START)	32	See section	0000_0000
54C	End Address of Region (IPEDCTX1END)	32	See section	0000_0000
560	IPED Context2 IV0 (IPEDCTX2IV0)	32	RW	0000_0000
564	IPED Context2 IV1 (IPEDCTX2IV1)	32	RW	0000_0000
568	Start Address of Region (IPEDCTX2START)	32	See section	0000_0000
56C	End Address of Region (IPEDCTX2END)	32	See section	0000_0000
580	IPED Context3 IV0 (IPEDCTX3IV0)	32	RW	0000_0000
584	IPED Context3 IV1 (IPEDCTX3IV1)	32	RW	0000_0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
588	Start Address of Region (IPEDCTX3START)	32	See section	0000_0000
58C	End Address of Region (IPEDCTX3END)	32	See section	0000_0000

19.7.2.1.1 Module Control 0 (MCR0)

Controls basic functions of FlexSPI module

Offset

Register	Offset
MCR0	0h

Diagram

Fields

Field	Description
31-24 AHBGRANTWAIT	Timeouts Wait Cycle for AHB command Grant Sets duration of timeout wait cycle for AHB commands. If the arbitrator does not grant the AHB-triggered command, it times out after (AHBGRANTTIMEOUT × 1024) AHB clock cycles. When the pending command sequence is IP-triggered and the read or write data size is too large, this grant timeout may occur. When an AHB command grant timeout occurs, an INTR[AHBCMDGE] interrupt is generated. When INTEN[AHBCMDGEEN] = 1, the arbitrator ignores AHB commands. **NOTE** This field is for debugging only. Do not change the value from its default.
23-16	Timeout Wait Cycle for IP Command Grant

Table continues on the next page...

Table continued from the previous page...

Field	Description
IPGRANTWAIT	Sets duration of timeout wait cycle for IP commands. If the arbitrator does not grant the IP-triggered command, it times out after (IPGRANTTIMEOUT × 1024) AHB clock cycles. When the pending command sequence is AHB-triggered and the read or write data size is too large, this grant timeout may occur. When IP command grant timeout occurs, an INTR[IPCMDGE] interrupt is generated. When INTEN[IPCMDGEEN] = 1, the arbitrator ignores IP commands. **NOTE** This field is for debugging only. Do not change the value from its default.
15 LEARNEN	Data Learning Enable Enables data learning feature. When data learning is disabled, the sampling clock phase 0 is always used for receive data sampling, even when a learn instruction is correctly executed. See Data learning. 0 - Disable 1 - Enable
14 SCKFREERUNEN	SCLK Free-running Enable Enables free-running SCLK output. For FPGA applications, the external device may use SCLK as a reference clock to its internal PLL. If SCLK free-running is enabled, data sampling with loopback clock from SCLK pad is not supported (MCR0[RXCLKSRC] = 2). 0 - Disable 1 - Enable
13 —	Reserved
12 DOZEEN	Doze Mode Enable Enables Doze mode. When enabled, AHB clock and serial clock are gated off when there is a Doze-mode request from the system. 0 - Disable 1 - Enable
11 HSEN	Half Speed Serial Flash Memory Access Enable Enables the clock divider to provide a half-speed clock to external serial flash devices (A_SCLK/ B_SCLK) for all commands in SDR and DDR mode. Write 1 to MCR0[MDIS] before changing the value of this field. Failure to do so may cause issues in the internal logic or state machine. 0 - Disable 1 - Enable
10-8 SERCLKDIV	Serial Root Clock Divider Sets divider value for serial root clock. The serial root clock can be divided inside FlexSPI . See Clocking.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	NOTE Do not modify this field while FlexSPI is active (MCR0[MDIS] = 0). 000 - Divided by 1 001 - Divided by 2 010 - Divided by 3 011 - Divided by 4 100 - Divided by 5 101 - Divided by 6 110 - Divided by 7 111 - Divided by 8
7 —	Reserved
6 —	Reserved
5-4 RXCLKSRC	Sample Clock Source for Flash Reading Selects sampling clock source for flash memory reading. See Receive clock source features. 00 - Dummy Read strobe that FlexSPI generates, looped back internally 01 - Dummy Read strobe that FlexSPI generates, looped back from DQS pad 10 - SCLK output clock and looped back from SCLK pad 11 - Flash-memory-provided read strobe and input from DQS pad
3-2 —	Reserved
1 MDIS	Module Disable Disables FlexSPI module. When the module is disabled, AHB and serial clock are gated off internally to save power. Only register access is allowed, except for the LUT, IP receive FIFO, and IP transmit FIFO. 0 - No impact 1 - Module disable
0 SWRESET	Software Reset Resets the internal FlexSPI state machine and aborts any transactions in progress. Hardware automatically writes 0 to this field after software reset is done. Configuration registers are not reset. 0 - No impact

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Software reset

19.7.2.1.2 Module Control 1 (MCR1)

Controls basic functions of FlexSPI module

Offset

Register	Offset
MCR1	4h

Diagram

Fields

Field	Description
31-16 SEQWAIT	Command Sequence Wait Sets wait time for command sequence. Command sequence execution times out and aborts after (SEQWAIT × 1024) serial root clock cycles. When this timeout occurs, if the interrupt is enabled (INTEN[SEQTIMEOUTEN] = 1), an INTR[SEQTIMEOUT] interrupt is generated. Also, the arbitrator ignores AHB commands. **NOTE** You cannot write 0 to this field.
15-0 AHBBUSWAIT	AHB Bus Wait Sets wait time for AHB bus. When data is not received from or transmitted to serial flash memory space after (AHBBUSWAIT × 1024) AHB clock cycles, the read or write access times out. When this timeout occurs, the AHB bus receives an error response, and an interrupt is generated (INTR[AHBBUSTIMEOUT]). When INTR[AHBBUSTIMEOUT] = 1, the arbitrator ignores AHB commands.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	NOTE You cannot write 0 to this field.

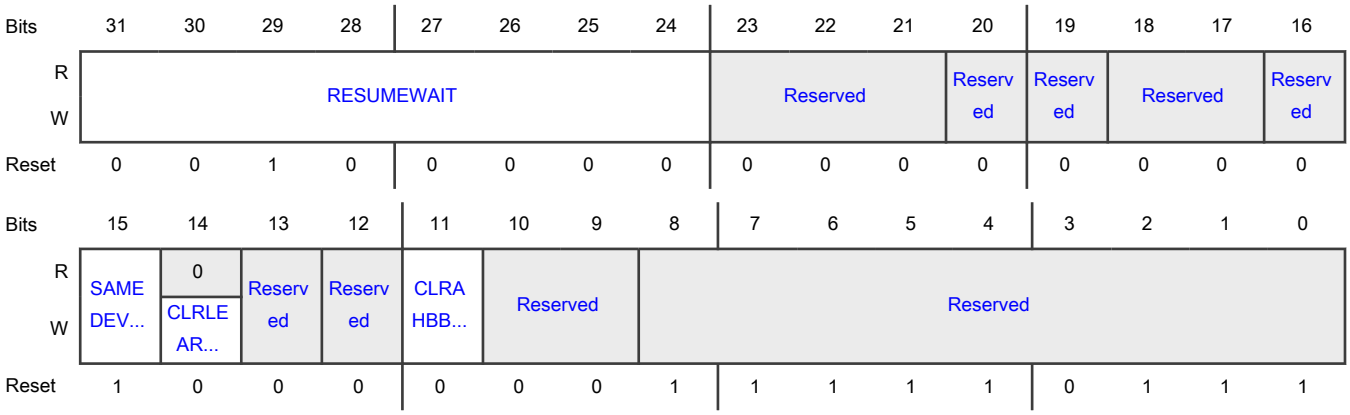
19.7.2.1.3 Module Control 2 (MCR2)

Controls basic functions of FlexSPI module.

Offset

Register	Offset
MCR2	8h

Diagram



Fields

Field	Description
31-24 RESUMEWAIT	Resume Wait Duration Determines duration (in AHB clock cycles) to remain in idle state before suspended command sequence is resumed. See Command abort and suspend.
23-21 —	Reserved
20 —	Reserved
19 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
18-17 —	Reserved
16 —	Reserved
15 SAMEDEVICEEN	Same Device Enable Sets all external devices (A1, A2, B1, and B2) to be the same devices (in type and in size). 0 - In Individual mode, FLSHA1CRx and FLSHA2CRx settings are applied to Flash A1, A2 separately. 1 - FLSHA1CR0, FLSHA1CR1, and FLSHA1CR2 register settings are applied to Flash A1, A2. FLSHA2CRx settings are ignored.
14 CLRLEARNPHASE	Clear Learn Phase Selection Resets the sampling clock phase selection to 0. When 1 is written to this field, it becomes 0 again immediately. 0 - No impact 1 - Reset sample clock phase selection to 0
13 —	Reserved
12 —	Reserved
11 CLRAHBBUFOPT	Clear AHB Buffer Determines whether AHB receive and transmit buffers are cleared automatically when FlexSPI returns Stop mode ACK. If an AHB receive buffer or transmit buffer will be powered off in Stop mode, software should write 1 to this field. Otherwise, an AHB read access after exiting Stop mode may hit either buffer, but their data entries are invalid. 0 - Not cleared automatically 1 - Cleared automatically
10-9 —	Reserved
8-0 —	Reserved

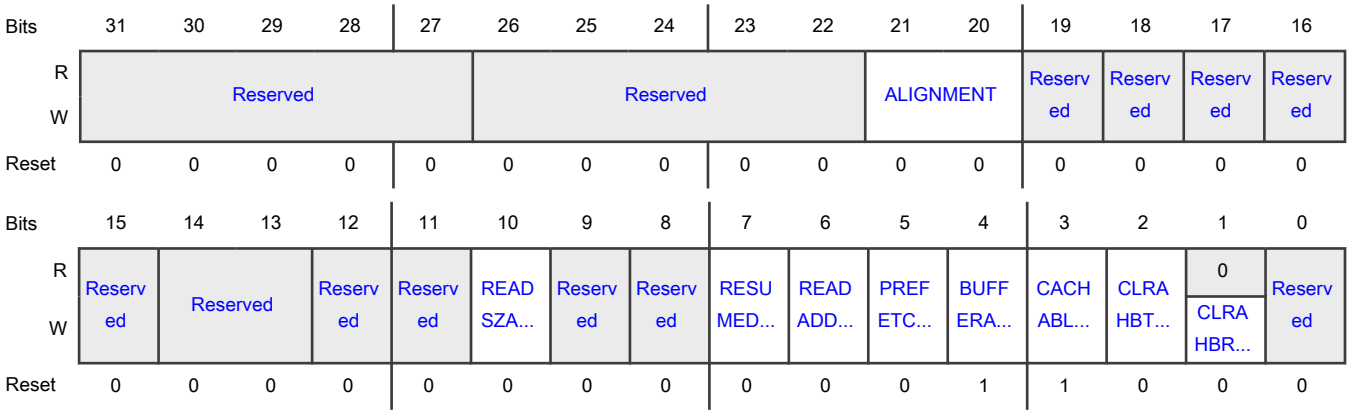
19.7.2.1.4 AHB Bus Control (AHBCR)

Controls AHB bus interface functions.

Offset

Register	Offset
AHBCR	Ch

Diagram



Fields

Field	Description
31-27 —	Reserved
26-22 —	Reserved
21-20 ALIGNMENT	AHB Boundary Alignment Configures the size of AHB read and write boundary. All accesses that cross the boundary are divided into smaller sub accesses. 00 - No limit 01 - 1 KB 10 - 512 bytes 11 - 256 bytes
19 —	Reserved
18 —	Reserved
17 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
16 —	Reserved
15 —	Reserved
14-13 —	Reserved
12 —	Reserved
11 —	Reserved
10 READSZALIGN	AHB Read Size Alignment Configures how AHB read size is determined. See Table 67. 0 - Register settings such as PREFETCH_EN determine AHB read size. 1 - AHB read size to up size to 8 bytes aligned, no prefetching
9 —	Reserved
8 —	Reserved
7 RESUMEDISABLE	AHB Read Resume Disable Disables the resumption of AHB read prefetch. 0 - Suspended AHB read prefetch resumes when AHB is IDLE. 1 - Suspended AHB read prefetch does not resume once aborted,
6 READADDROPT	AHB Read Address Option Removes AHB burst start address alignment limitation. When the FlexSPI controller is used for FPGA application, this field may be required for FlexSPI to fetch the exact byte number as an AHB burst. In this case, FPGA device should be non-word-addressable and this field must be 0. 0 - AHB read burst start address alignment is limited when flash memory is accessed in flash is word-addressable. 1 - AHB read burst start address alignment is not limited. FlexSPI fetches more data than the AHB burst requires for address alignment.
5	AHB Read Prefetch Enable

Table continues on the next page...

Table continued from the previous page...

Field	Description
PREFETCHEN	Enables AHB read prefetch. When enabled, FlexSPI fetches more flash read data than the current AHB burst requires. This action reduces the read latency for next AHB read access. **NOTE** AHB read prefetch is enabled only when both this field and AHBRXBUFnCR0[PREFETCHEN] are 1. 0 - Disable 1 - Enable
4 BUFFERABLEEN	Bufferable Write Access Enable Enables AHB bus bufferable write access. This field affects the last beat of an AHB write access. See AHB write access to flash memory. 0 - Disabled. For all AHB write accesses (bufferable or nonbufferable), FlexSPI returns AHB Bus Ready after transmitting all data and finishing command. 1 - Enabled. For AHB bufferable write access, FlexSPI returns AHB Bus Ready when the arbitrator grants the AHB command. FlexSPI does not wait for the AHB command to finish.
3 CACHABLEEN	Cacheable Read Access Enable Enables AHB bus cacheable read access. 0 - Disabled. When an AHB bus cacheable read access occurs, FlexSPI does not check whether it hit the AHB transmit buffer. 1 - Enabled. When an AHB bus cacheable read access occurs, FlexSPI first checks whether the access hit the AHB transmit buffer.
2 CLRAHBTXBUF	Clear AHB Transmit Buffer Clears status and pointers of AHB transmit buffer. When the clear operation completes, this field clears automatically, so it always reads as 0. 0 - No impact. 1 - Enable clear operation.
1 CLRAHBRXBUF	Clear AHB Receive Buffer Clears status and pointers of AHB receive buffer. When the clear operation completes, this field clears automatically, so it always reads as 0. 0 - No impact. 1 - Enable clear operation.
0 —	Reserved

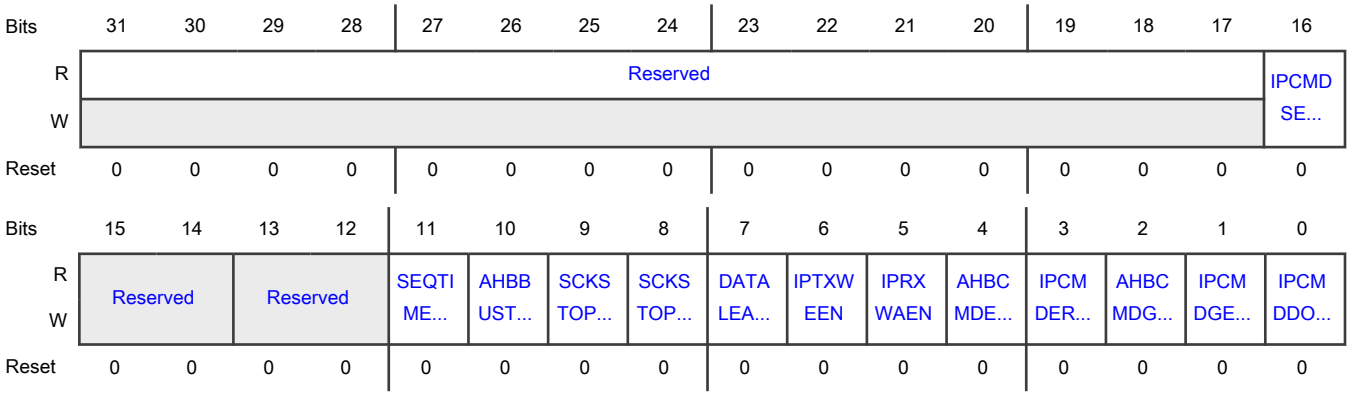
19.7.2.1.5 Interrupt Enable (INTEN)

Includes AHB error response, IPS error response, and ECC error interrupt enables. See Interrupts.

Offset

Register	Offset
INTEN	10h

Diagram



Fields

Field	Description
31-17 —	Reserved
16 IPCMDSECUREVIOEN	IP Command Security Violation Interrupt Enable 0 - Disable interrupt or no impact 1 - Enable interrupt
15-14 —	Reserved
13-12 —	Reserved
11 SEQTIMEOUTEN	Sequence execution Timeout Interrupt Enable 0 - Disable interrupt or no impact 1 - Enable interrupt
10 AHBBUSTIMEOUTEN	AHB Bus Timeout Interrupt Enable 0 - Disable interrupt or no impact 1 - Enable interrupt
9	SCLK Stopped By Write Interrupt Enable

Table continues on the next page...

Table continued from the previous page...

Field	Description
SCKSTOPBYWREN	Enables interrupt that indicates SCLK is stopped during command sequence because asynchronous transmit FIFO is empty. 0 - Disable interrupt or no impact 1 - Enable interrupt
8 SCKSTOPBYRDEN	SCLK Stopped By Read Interrupt Enable Enables interrupt that indicates SCLK is stopped during command sequence because asynchronous receive FIFO is full. 0 - Disable interrupt or no impact 1 - Enable interrupt
7 DATALEARNFAILEN	Data Learning Failed Interrupt Enable 0 - Disable interrupt or no impact 1 - Enable interrupt
6 IPTXWEEN	IP Transmit FIFO Watermark Empty Interrupt Enable Enables interrupt that indicates IP transmit FIFO contains more empty space than watermark level. 0 - Disable interrupt or no impact 1 - Enable interrupt
5 IPRXWAEN	IP Receive FIFO Watermark Available Interrupt Enable Enables interrupt that indicates IP receive FIFO contains more valid data than the watermark level. 0 - Disable interrupt or no impact 1 - Enable interrupt
4 AHBCMDERREN	AHB-Triggered Command Sequences Error Detected Interrupt Enable 0 - Disable interrupt or no impact 1 - Enable interrupt
3 IPCMDERREN	IP-Triggered Command Sequences Error Detected Interrupt Enable 0 - Disable interrupt or no impact 1 - Enable interrupt
2 AHBCMDGEEN	AHB-Triggered Command Sequences Grant Timeout Interrupt Enable. 0 - Disable interrupt or no impact 1 - Enable interrupt
1 IPCMDGEEN	IP-Triggered Command Sequences Grant Timeout Interrupt Enable 0 - Disable interrupt or no impact 1 - Enable interrupt

Table continues on the next page...

Table continued from the previous page...

Field	Description
0 IPCMDDONEEN	IP-Triggered Command Sequences Execution Finished Interrupt Enable 0 - Disable interrupt or no impact 1 - Enable interrupt

19.7.2.1.6 Interrupt (INTR)

Includes AHB error response, IPS error response, and ECC error interrupts. See Interrupts.

Offset

Register	Offset
INTR	14h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															IPCMDSE...
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	Reserved		Reserved		SEQTIME...	AHBBUST...	SCKSTOP...	SCKSTOP...	DATALEA...	IPTXWE	IPRXWA	AHBCMDE...	IPCMDERR	AHBCMDGE	IPCMDGE	IPCMDDO...
W					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	Description
31-17 —	Reserved
16 IPCMDSECUREVIO	IP Command Security Violation **NOTE** This field behaves differently for read and write operations. When reading 0 - Interrupt condition has not occurred 1 - Interrupt condition has occurred

Table continues on the next page...

Table continued from the previous page...

Field	Description
	When writing 0 - No effect 1 - Clear the flag
15-14 —	Reserved
13-12 —	Reserved
11 SEQTIMEOUT	Sequence Execution Timeout **NOTE** This field behaves differently for read and write operations. When reading 0 - Interrupt condition has not occurred 1 - Interrupt condition has occurred When writing 0 - No effect 1 - Clear the flag
10 AHBBUSTIMEOUT	AHB Bus Timeout **NOTE** This field behaves differently for read and write operations. When reading 0 - Interrupt condition has not occurred 1 - Interrupt condition has occurred When writing 0 - No effect 1 - Clear the flag
9 SCKSTOPBYWR	SCLK Stopped Due To Empty Transmit FIFO Generated when SCLK is stopped during command sequence because the asynchronous transmit FIFO is empty. **NOTE** This field behaves differently for read and write operations. When reading 0 - Interrupt condition has not occurred

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Interrupt condition has occurred When writing 0 - No effect 1 - Clear the flag
8 SCKSTOPBYRD	SCLK Stopped Due To Full Receive FIFO Generated when SCLK is stopped during command sequence because the asynchronous receive FIFO is full. **NOTE** This field behaves differently for read and write operations. When reading 0 - Interrupt condition has not occurred 1 - Interrupt condition has occurred When writing 0 - No effect 1 - Clear the flag
7 DATALEARNFAIL	Data Learning Failed **NOTE** This field behaves differently for read and write operations. When reading 0 - Interrupt condition has not occurred 1 - Interrupt condition has occurred When writing 0 - No effect 1 - Clear the flag
6 IPTXWE	IP Transmit FIFO Watermark Empty Generated when the IP transmit FIFO contains more empty space than the watermark level. **NOTE** This field behaves differently for read and write operations. When reading 0 - Interrupt condition has not occurred 1 - Interrupt condition has occurred When writing 0 - No effect

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Clear the flag
5 IPRXWA	IP Receive FIFO Watermark Available Generated when the IP receive FIFO contains more valid data than the watermark level. **NOTE** This field behaves differently for read and write operations. When reading 0 - Interrupt condition has not occurred 1 - Interrupt condition has occurred When writing 0 - No effect 1 - Clear the flag
4 AHBCMDERR	AHB-Triggered Command Sequences Error Generated upon an AHB-triggered command sequence error. When an error is detected for an AHB command, the command is ignored and not executed. **NOTE** This field behaves differently for read and write operations. When reading 0 - Interrupt condition has not occurred 1 - Interrupt condition has occurred When writing 0 - No effect 1 - Clear the flag
3 IPCMDERR	IP-Triggered Command Sequences Error Generated upon an IP-triggered command sequence error. When an error is detected for an IP command, the command is ignored and not executed. **NOTE** This field behaves differently for read and write operations. When reading 0 - Interrupt condition has not occurred 1 - Interrupt condition has occurred When writing 0 - No effect 1 - Clear the flag

Table continues on the next page...

Table continued from the previous page...

Field	Description
2 AHBCMDGE	AHB-Triggered Command Sequences Grant Timeout **NOTE** This field behaves differently for read and write operations. When reading 0 - Interrupt condition has not occurred 1 - Interrupt condition has occurred When writing 0 - No effect 1 - Clear the flag
1 IPCMDGE	IP-Triggered Command Sequences Grant Timeout **NOTE** This field behaves differently for read and write operations. When reading 0 - Interrupt condition has not occurred 1 - Interrupt condition has occurred When writing 0 - No effect 1 - Clear the flag
0 IPCMDDONE	IP-Triggered Command Sequences Execution Finished Generated upon completion of an IP-triggered command sequence. Also generated when IPCMDGE or IPCMDERR interrupt is generated **NOTE** This field behaves differently for read and write operations. When reading 0 - Interrupt condition has not occurred 1 - Interrupt condition has occurred When writing 0 - No effect 1 - Clear the flag

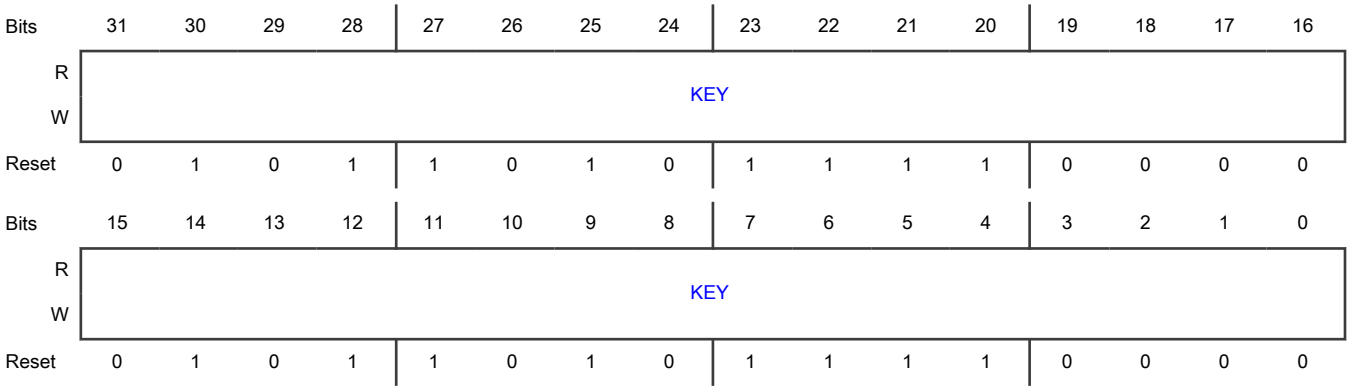
19.7.2.1.7 LUT Key (LUTKEY)

Contains the key to lock and unlock LUT. See Lookup table (LUT).

Offset

Register	Offset
LUTKEY	18h

Diagram



Fields

Field	Description
31-0 KEY	LUT Key Contains key to lock or unlock LUT. The key is 5AF05AF0h. Read value is always 5AF05AF0h.

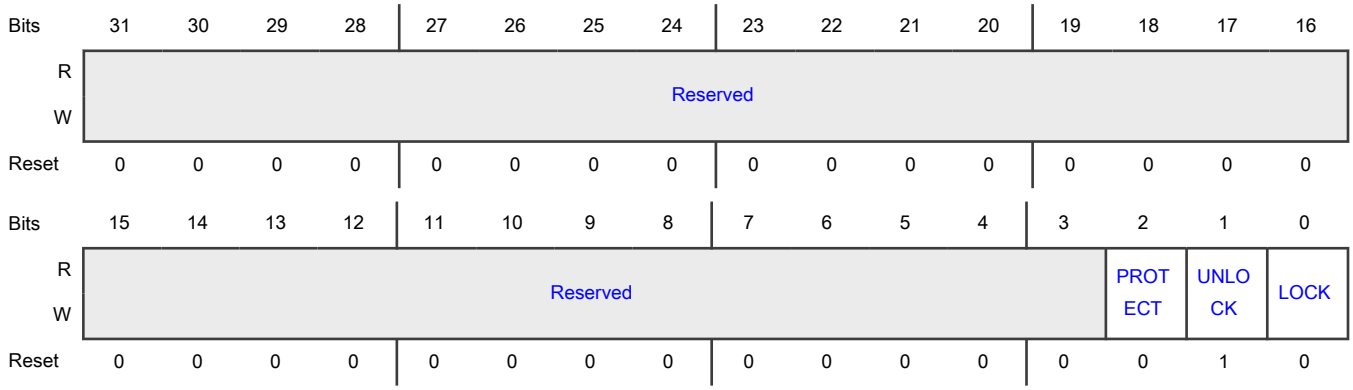
19.7.2.1.8 LUT Control (LUTCR)

Used with LUTKEY register to lock or unlock LUT. For the lock or unlock operation to be successful, this register must be written immediately after writing 5AF05AF0h to the LUTKEY register. See Lookup table (LUT) for details on locking and unlocking LUT. You cannot write 00 or 11 to the LOCK and UNLOCK fields.

Offset

Register	Offset
LUTCR	1Ch

Diagram



Fields

Field	Description
31-3 —	Reserved
2 PROTECT	LUT Protection Controls lookup table protection. 0 - Not protected. All IPS controllers can access LUTCR and LUT memory. 1 - Protected. Only secure IPS controller can change the value of LUTCR and write to LUT memory.
1 UNLOCK	Unlock LUT 0 - LUT is locked (LUTCR[LOCK] must be 1) 1 - LUT is unlocked and can be written
0 LOCK	Lock LUT 0 - LUT is unlocked (LUTCR[UNLOCK] must be 1) 1 - LUT is locked and cannot be written

19.7.2.1.9 AHB Receive Buffer n Control 0 (AHBRXBUF0CR0 - AHBRXBUF7CR0)

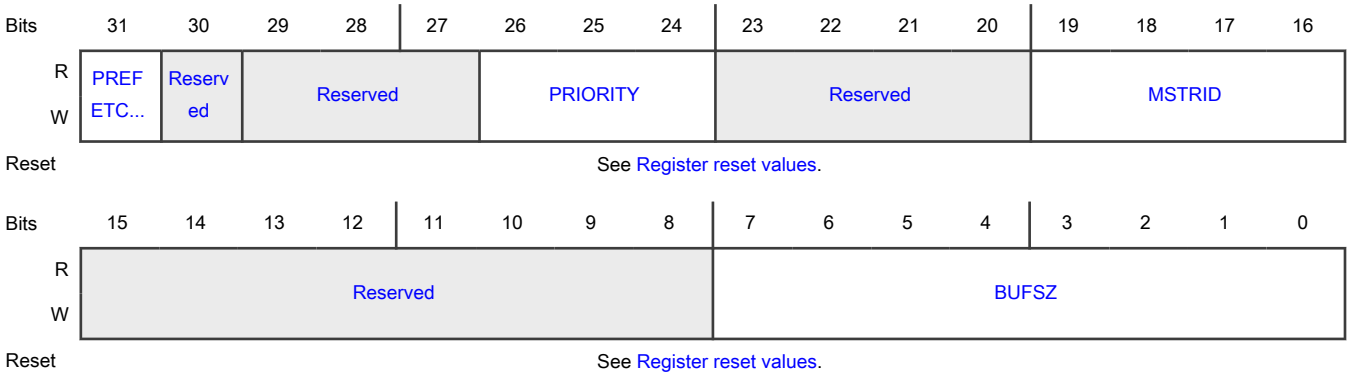
Stores the read data from the SPI interface.

Offset

For n = 0 to 7:

Register	Offset
AHBRXBUFnCR0	20h + (n × 4h)

Diagram



Register reset values

Register	Reset value
AHBRXBUF0CR0	8000_0010h
AHBRXBUF1CR0	8001_0010h
AHBRXBUF2CR0	8002_0010h
AHBRXBUF3CR0	8003_0010h
AHBRXBUF4CR0	8004_0010h
AHBRXBUF5CR0	8005_0010h
AHBRXBUF6CR0	8006_0010h
AHBRXBUF7CR0	8007_0010h

Fields

Field	Description
31 PREFETCHEN	AHB Read Prefetch Enable Enables AHB read prefetch for the controller corresponding to the current AHB receive buffer. The prefetch feature is disabled when AHBCR[PREFETCHEN] is 0. You can use this field to enable or disable prefetch separately for each controller. 0 - Disabled 1 - Enabled when AHBCR[PREFETCHEN] is enabled.
30 —	Reserved
29-27 —	Reserved
26-24	AHB Controller Read Priority

Table continues on the next page...

Table continued from the previous page...

Field	Description
PRIORITY	Configure the priority for the AHB Controller Read to which this AHB receive buffer is assigned. 7 is the highest priority, 0 the lowest. See Command abort and suspend.
23-20 —	Reserved
19-16 MSTRID	AHB Controller ID Configures the ID of the AHB controller to which this AHB receive buffer is assigned. See AHB receive buffer management.
15-8 —	Reserved
7-0 BUFSZ	AHB Receive Buffer Size Configures the size of the AHB receive buffer in multiples of 64 bits. See AHB receive buffer management.

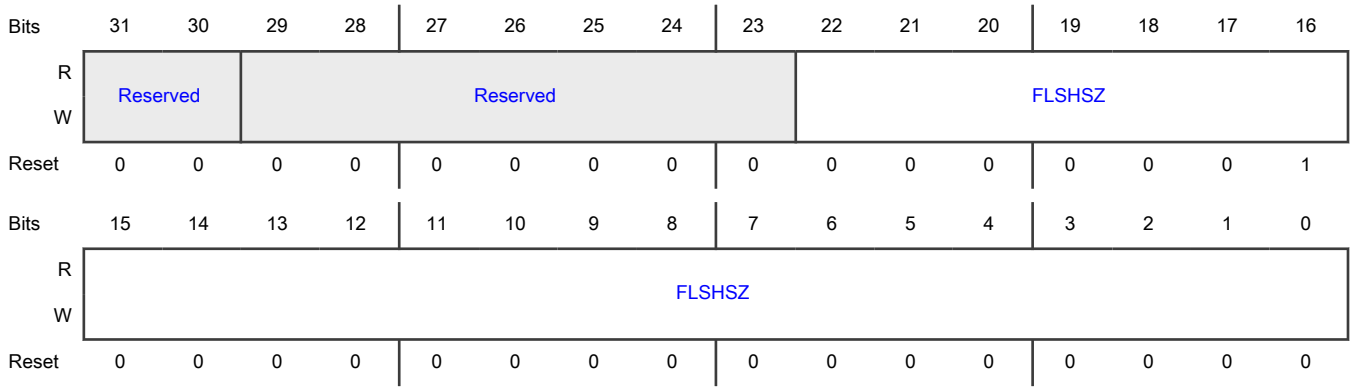
19.7.2.1.10 Flash Control 0 (FLSHA1CR0 - FLSHA2CR0)

Contains flash memory size setting. FlexSPI determines which device is accessed (Chip Select) via this register.

Offset

Register	Offset
FLSHA1CR0	60h
FLSHA2CR0	64h

Diagram



Fields

Field	Description
31-30 —	Reserved
29-23 —	Reserved
22-0 FLSHSZ	Flash Size in KB Configures the maximum flash memory size. The maximum flash size supported for each device is 4 GB. When the value of this field is greater than 400000h, the device flash size is taken as 4 GB. The max total flash size supported (for all 4 devices) is also 4 GB. If the total flash size is larger than 4 GB, only 4 GB of address space is accessible.

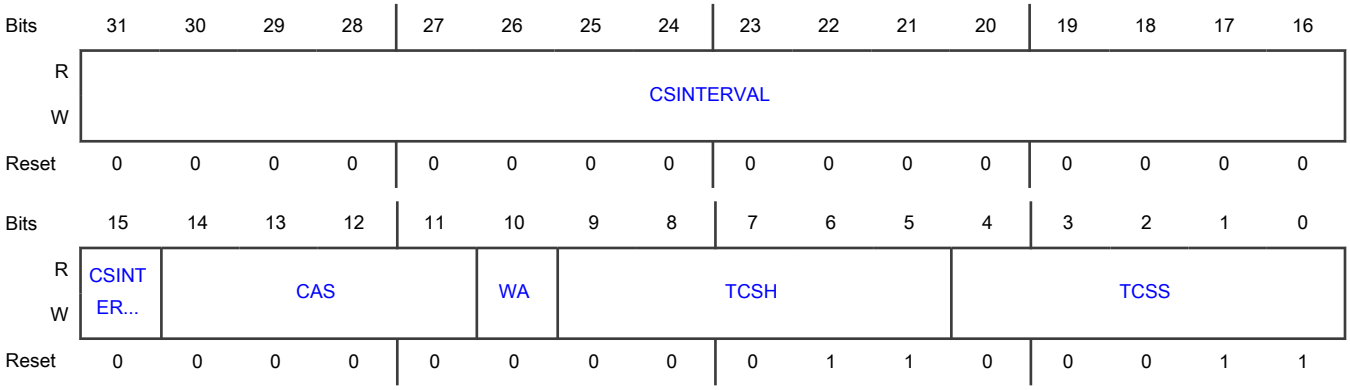
19.7.2.1.11 Flash Control 1 (FLSHA1CR1 - FLSHA2CR1)

Contains settings for flash device-specific timings and flash internal address space.

Offset

Register	Offset
FLSHA1CR1	70h
FLSHA2CR1	74h

Diagram



Fields

Field	Description
31-16 CSINTERVAL	Chip Select Interval

Table continues on the next page...

Table continued from the previous page...

Field	Description
	Configures the minimum interval between flash device chip select deassertion and chip select assertion. If the external flash device has a limitation on the interval between command sequences, configure this field accordingly. If there is no limitation, write 0h to this field. When CSINTERVALUNIT = 0, the chip select invalid interval is: CSINTERVAL × 1 serial clock cycle; When CSINTERVALUNIT = 1, the chip select invalid interval is: CSINTERVAL × 256 serial clock cycles. **NOTE** The minimum chip select interval is 2 cycles, even when the value of CSINTERVAL is less than 2.
15 CSINTERVALUNIT	Chip Select Interval Unit Configures the interval unit for chip select. 0 - 1 serial clock cycle 1 - 256 serial clock cycles
14-11 CAS	Column Address Size When external flash memory has a separate address field for rows and columns, this field configures the flash column address bit width. FlexSPI automatically splits a flash-mapped address into row address and column address according to the values of this field and the WA field. When the external flash memory does not support column address, write 0 to this field. FlexSPI transmits all flash address bits as Row address. For flash address mapping, see Flash address sent to flash memory devices.
10 WA	Word-Addressable Configures whether external flash memory is word-addressable or byte-addressable. If flash memory is word-addressable, it should be accessed in multiples of 16 bits. Currently, FlexSPI does not transmit flash address bit 0 to external flash memory. For flash address mapping, see Flash address sent to flash memory devices. 0 - Byte-addressable 1 - Word-addressable
9-5 TCSH	Serial Flash CS Hold Time Used to meet flash TCSH timing requirement. Serial flash CS Hold time that FlexSPI promises is: (TCSH + 1/2) serial clock cycles for DDR mode, and TCSH serial clock cycles for SDR mode. See Output timing between chip select and SCLK.
4-0 TCSS	Serial Flash CS Setup Time Used to meet flash TCSS timing requirement. Serial flash CS Setup time that FlexSPI promises is: (TCSS + 1/2) serial root clock cycles for SDR and DDR mode. See Output timing between chip select and SCLK.

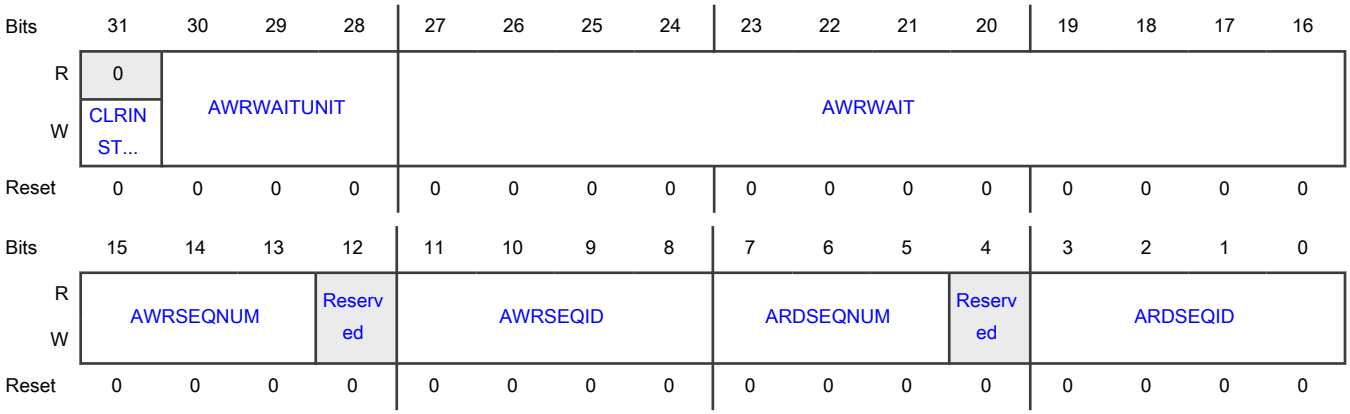
19.7.2.1.12 Flash Control 2 (FLSHA1CR2 - FLSHA2CR2)

Contains fields to configure AHB bus access. If the four external devices are different types, AHB read and write commands may use different command sequences. AHB bus ready wait time may also differ.

Offset

Register	Offset
FLSHA1CR2	80h
FLSHA2CR2	84h

Diagram



Fields

Field	Description
31 CLRINSTRPTR	Clear Instruction Pointer Clears the instruction pointer, which is the pointer that JMP_ON_CS saves internally. See Programmable sequence engine. This field is used for AHB read access to external flash memory supporting Execute-In-Place (XIP) mode.
30-28 AWRWAITUNIT	AWRWAIT Unit Configures the unit of AHB write wait time, as the value of AWRWAIT determines, in terms of AHB clock cycles. 000 - 2 001 - 8 010 - 32 011 - 128 100 - 512 101 - 2048 110 - 8192 111 - 32768
27-16 AWRWAIT	AHB Write Wait Configures AHB write wait time, with the AWRWAITUNIT field. Certain devices (such as FPGA) require time to write data into internal memory after the command sequences finished on the FlexSPI

Table continues on the next page...

Table continued from the previous page...

Field	Description
	interface. If another read command sequence arrives before the current programming finishes internally, the read data may be wrong. This field is used to hold the AHB bus ready for AHB write access, waiting until the programming is finished in the external device. This hold ensures that no AHB read command is triggered before the programming finishes in the external device. The wait cycle between an AHB-triggered command sequence finishing on FlexSPI and the AHB return bus being ready is: AWRWAIT × AWRWAITUNIT.
15-13 AWRSEQNUM	Sequence Number for AHB Write-Triggered Command Configures the sequence number of an AHB read-triggered command. For certain flash devices (for example, HyperFlash, HyperRam, and Serial NAND flash), a flash programming access is done via several command sequences. An AHB write command triggers (AWRSEQNUM + 1) command sequences to external flash memory each time. FlexSPI executes the sequences in LUT incrementally. **NOTE** • Software should ensure that the last sequence index never exceeds LUT sequence numbers: AWRSEQID + AWRSEQNUM < 16 • Software must ensure that the AWRSEQNUM and LUT fields are configured correctly according to the external device specification. FlexSPI does not check the sequence; it executes the sequences one by one.
12 —	Reserved
11-8 AWRSEQID	Sequence Index for AHB Write-Triggered Command
7-5 ARDSEQNUM	Sequence Number for AHB Read-Triggered Command Configures the sequence number of an AHB read-triggered command in the lookup table. For certain flash devices (for example, HyperFlash, HyperRam, and Serial NAND flash), a flash read access is done via several command sequences. An AHB read command triggers (ARDSEQNUM + 1) command sequences to external flash memory each time. FlexSPI executes the sequences in LUT incrementally. **NOTE** • Software should ensure that the last sequence index never exceeds LUT sequence numbers: ARDSEQID + ARDSEQNUM ≤ 16 • Software must ensure that the ARDSEQNUM and LUT fields are configured correctly according to the external device specification. FlexSPI does not check the sequence; it executes the sequences one by one.
4 —	Reserved
3-0 ARDSEQID	Sequence Index for AHB Read-Triggered Command in LUT.

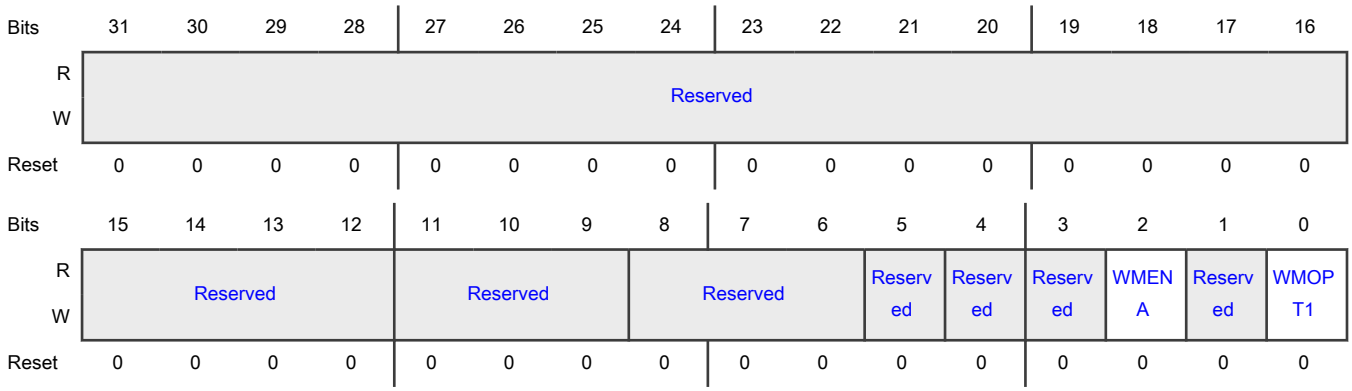
19.7.2.1.13 Flash Control 4 (FLSHCR4)

Provides configuration for all external devices.

Offset

Register	Offset
FLSHCR4	94h

Diagram



Fields

Field	Description
31-12 —	Reserved
11-9 —	Reserved
8-6 —	Reserved
5 —	Reserved
4 —	Reserved
3 —	Reserved
2 WMENA	Write Mask Enable for Port A Enables write mask for flash device on port A.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - Disabled. When writing to external device, DQS(RWDS) pin is not driven. 1 - Enabled. When writing to external device, FlexSPI drives DQS(RWDS) pin as write mask output.
1 —	Reserved
0 WMOPT1	Write Mask Option 1 Used to remove AHB and IP write burst start address alignment limitation. 0 - When writing to an external device, DQS pin is used as write mask. When flash memory is accessed in individual mode, AHB or IP write burst start address alignment is not limited. 1 - When writing to an external device, DQS pin is not used as write mask. When flash memory is accessed in individual mode, AHB or IP write burst start address alignment is limited.

19.7.2.1.14 IP Control 0 (IPCR0)

Provides all configuration required for IP commands. Provides the start address for the flash device, instead of the chip address, to be accessed for IP command. FlexSPI determines the chip select automatically according to this start address.

NOTE

- You cannot issue an IP command that crosses flash device boundaries. If you do so, it generates an IPCMDERR interrupt.
- Configure this register before an IP command is triggered.
- Do not change the values in this register while an IP command is in progress.

Offset

Register	Offset
IPCR0	A0h

Diagram

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

Fields

Field	Description
31-0 SFAR	Serial Flash Address Configures the serial flash address for IP commands. The address should be the address of the flash device without the base address.

19.7.2.1.15 IP Control 1 (IPCR1)

Provides all configuration required for IP commands. Provides the flash read and program data size, sequence index in LUT, sequence number and individual mode settings for IP commands.

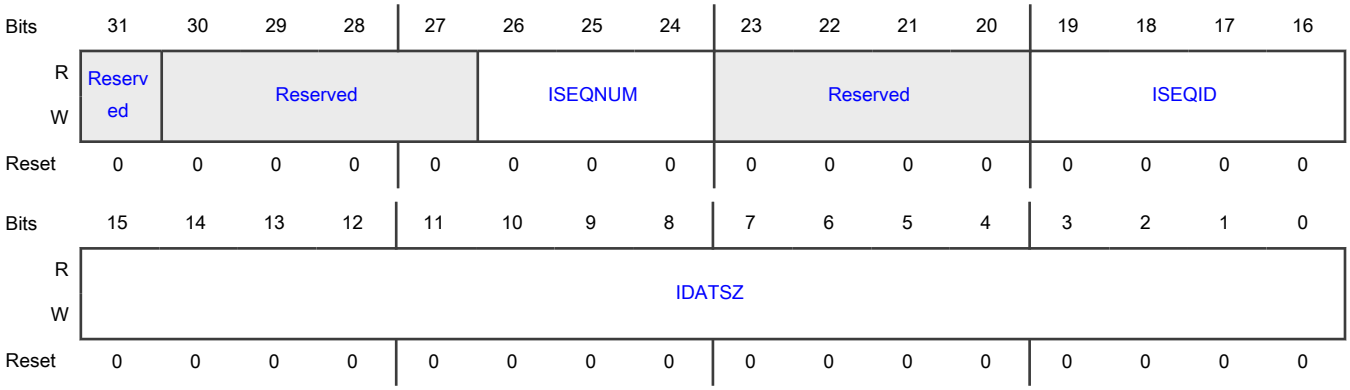
NOTE

- Configure this register before an IP command is triggered.
- Do not change the values in this register while an IP command is in progress.

Offset

Register	Offset
IPCR1	A4h

Diagram



Fields

Field	Description
31 —	Reserved
30-27 —	Reserved
26-24	Sequence Number for IP command: ISEQNUM+1.

Table continues on the next page...

Table continued from the previous page...

Field	Description
ISEQNUM	
23-20 —	Reserved
19-16 ISEQID	Sequence Index in LUT for IP command.
15-0 IDATSZ	Flash Read/Program Data Size (in bytes) for IP command.

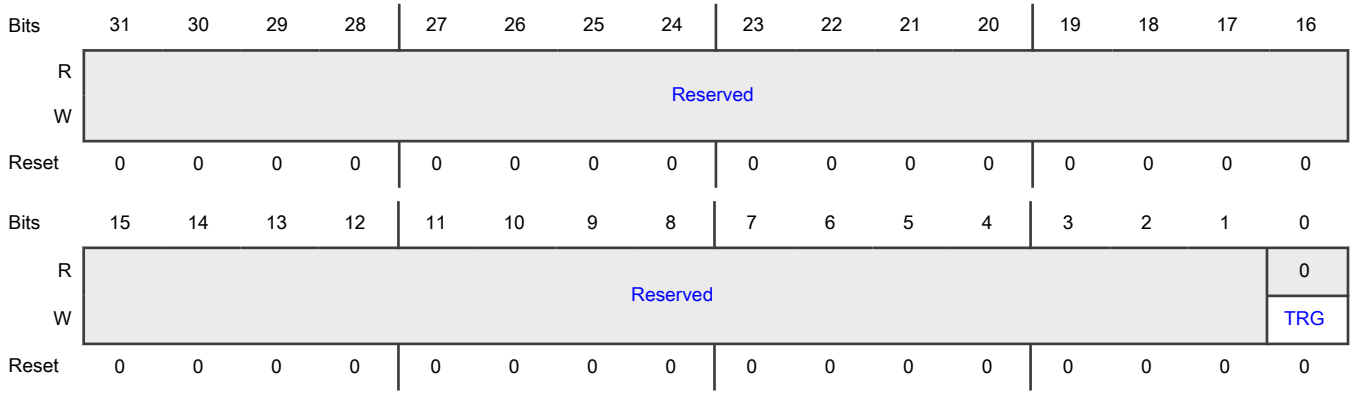
19.7.2.1.16 IP Command (IPCMD)

Used to trigger an IP command to access external flash device. When the arbitrator grants the IP command, the command is executed on the FlexSPI interface.

Offset

Register	Offset
IPCMD	B0h

Diagram



Fields

Field	Description
31-1 —	Reserved
0	Command Trigger

Table continues on the next page...

Table continued from the previous page...

Field	Description
TRG	Triggers an IP command. This field clears automatically after it has been written and always reads as 0. **NOTE** You cannot trigger another IP command before the previous IP command finishes on the FlexSPI interface. Software must poll INTR[IPCMDDONE] or wait for this interrupt. 0 - No action 1 - Start the IP command that the IPCR0 and IPCR1 registers define.

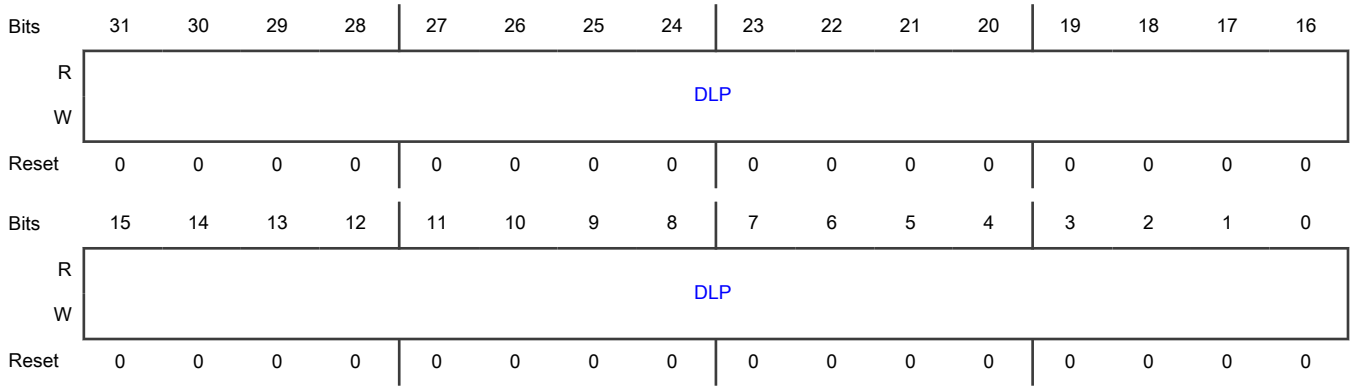
19.7.2.1.17 Data Learning Pattern (DLPR)

Provides the pattern to be used during Data Learning in FlexSPI.

Offset

Register	Offset
DLPR	B4h

Diagram



Fields

Field	Description
31-0 DLP	Data Learning Pattern The operand in the LEARN_SDR or LEARN_DDR instruction code determines the data learning pattern bit number. This number never exceeds 32 bits. If the operand in the instruction code is greater than 32, a 32-bit pattern is used. See Data learning.

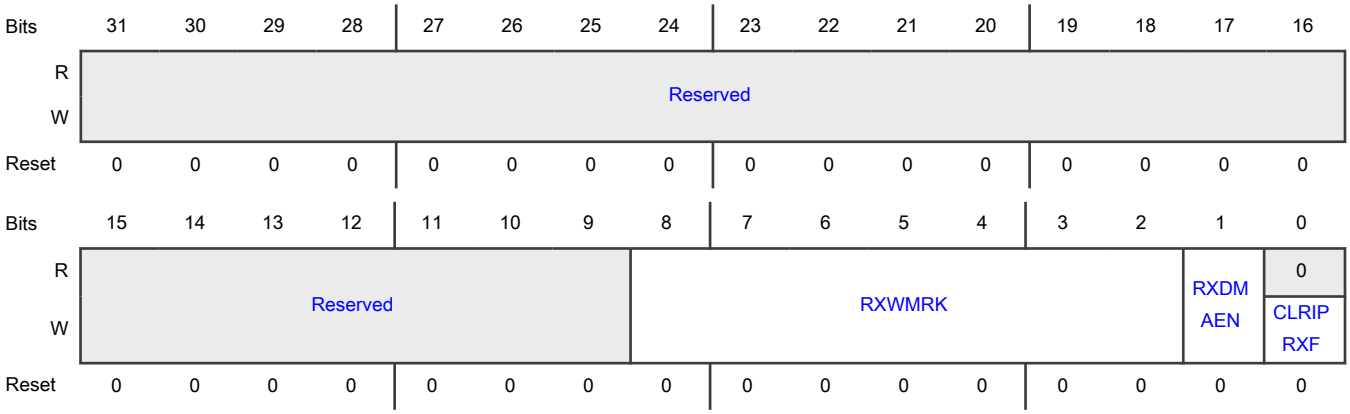
19.7.2.1.18 IP Receive FIFO Control (IPRXFCR)

Provides the configuration fields for IP receive FIFO management.

Offset

Register	Offset
IPRXFCR	B8h

Diagram



Fields

Field	Description
31-9 —	Reserved
8-2 RXWMRK	IP Receive FIFO Watermark Level Configures the watermark level for the IP receive FIFO. The watermark level is (RXWMRK + 1) × 64 bits. The INTR[IPRXWA] interrupt is set when FlexSPI fills the IP receive FIFO ≥ the watermark level. A DMA request occurs when the fill level is ≥ the watermark level and IPRXFCR[RXDMAEN] = 1. When the fill level is ≥ the watermark level and INTEN[IPRXWAEN] = 1, an INTR[IPRXWA] interrupt is generated. **NOTE** After writing 1 to INTR[IPRXWA] to clear it, the read address should be rolled back to the start address (memory mapped).
1 RXDMAEN	IP Receive FIFO Reading by DMA Enable 0 - Disabled. The processor reads the FIFO. 1 - Enabled. DMA reads the FIFO.
0 CLRIPRXF	Clear IP Receive FIFO Clears all valid data entries in IP receive FIFO. Resets the read and write pointers in IP receive FIFO. 0 - No function 1 - A clock cycle pulse clears all valid data entries in IP receive FIFO.

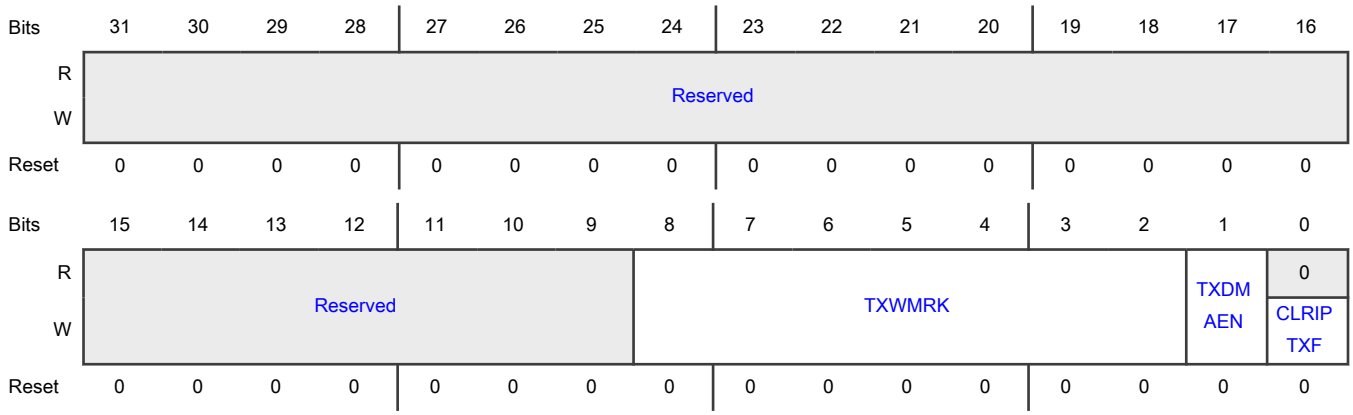
19.7.2.1.19 IP Transmit FIFO Control (IPTXFCR)

Provides the configuration fields for IP transmit FIFO management.

Offset

Register	Offset
IPTXFCR	BCh

Diagram



Fields

Field	Description
31-9 —	Reserved
8-2 TXWMRK	Transmit Watermark Level Sets the transmit watermark level. The watermark level is (TXWMRK + 1) × 64 bits. INTR[IPTXWE] is set when FlexSPI empties the IP transmit FIFO ≥ the watermark level. When the empty level ≥ the watermark level and IPTXFCR[TXDMAEN] = 1, a DMA request occurs. When the empty level ≥ the watermark level and INTEN[IPTXWEEN] = 1, an INTR[IPTXWE] interrupt is generated. **NOTE** • The watermark level should not be larger than the write window. • The watermark level should not be larger than the IP transmit FIFO size. • The write address to the IP receive FIFO should roll back to the start address of the write window. After pushing the IP transmit fifo, you should write 1 to INTR[IPTXWE] to clear it, which will perform the rollback.
1 TXDMAEN	Transmit FIFO DMA Enable Selects whether DMA or processor fills IP transmit FIFO. 0 - Processor

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - DMA
0 CLRIPTXF	Clear IP Transmit FIFO Clears all valid data entries in IP transmit FIFO. The read and write pointers in IP transmit FIFO are reset. 0 - No function 1 - A clock cycle pulse clears all valid data entries in the IP transmit FIFO.

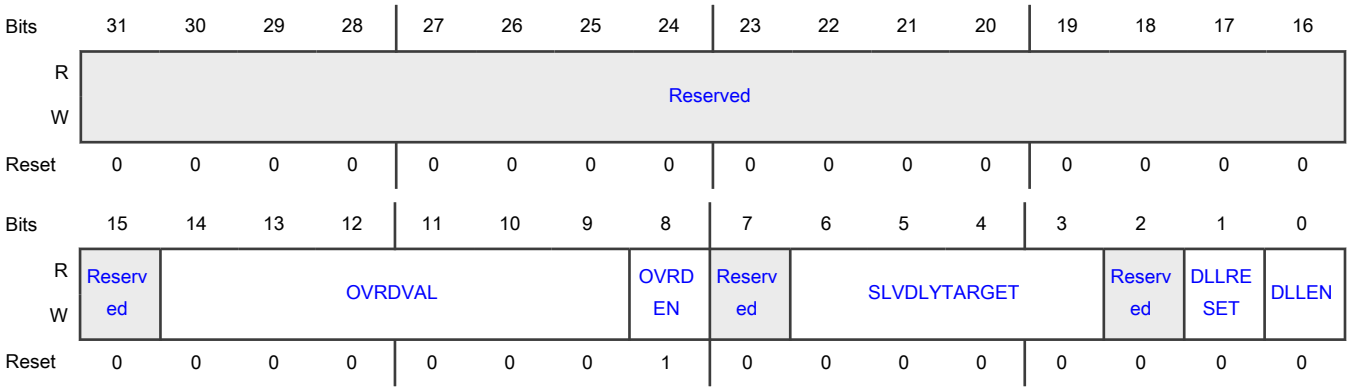
19.7.2.1.20 DLL Control 0 (DLLACR - DLLBCR)

Configures Flash A and B sample clock DLL.

Offset

Register	Offset
DLLACR	C0h
DLLBCR	C4h

Diagram



Fields

Field	Description
31-15 —	Reserved
14-9 OVRDVAL	Target Clock Delay Line Override Value Configures the override value for the target clock line delay cell number. When OVRDEN = 1, the delay cell number in DLL is OVRDVAL + 1. See DLL configuration for sampling.

Table continues on the next page...

Table continued from the previous page...

Field	Description
8 OVRDEN	Target Clock Delay Line Override Value Enable Enables the override value for the target clock line delay cell number. 0 - Disable 1 - Enable
7 —	Reserved
6-3 SLVDLYTARGET	Target Delay Line For Target Configures the target delay line for the target. The delay target for target delay line is: ((SLVDLYTARGET+1) × 1/32 × clock cycle of reference clock (serial root clock). If serial root clock is ≥ 100 MHz, DLLEN set to 1, OVRDEN set to 0, then SLVDLYTARGET setting of 0xF is recommended. This value is a recommended value. If a failure occurs, the value might require adjustment in a real application.
2 —	Reserved
1 DLLRESET	DLL reset Forces a DLL reset. The forced reset causes the DLL to lose lock and recalibrate to detect a ref_clock half-period phase shift. The reset action is edge-triggered, so software must write 0 to this field after writing 1 to it (no delay limitation). 0 - No function 1 - Force DLL reset.
0 DLLEN	DLL Calibration Enable Enables DLL calibration. When DLL calibration is disabled, the delay cell number in the target delay line is always 1. When DLLxCR[OVRDEN] = 1, the target delay line is overridden and this field is ignored. 0 - Disable 1 - Enable

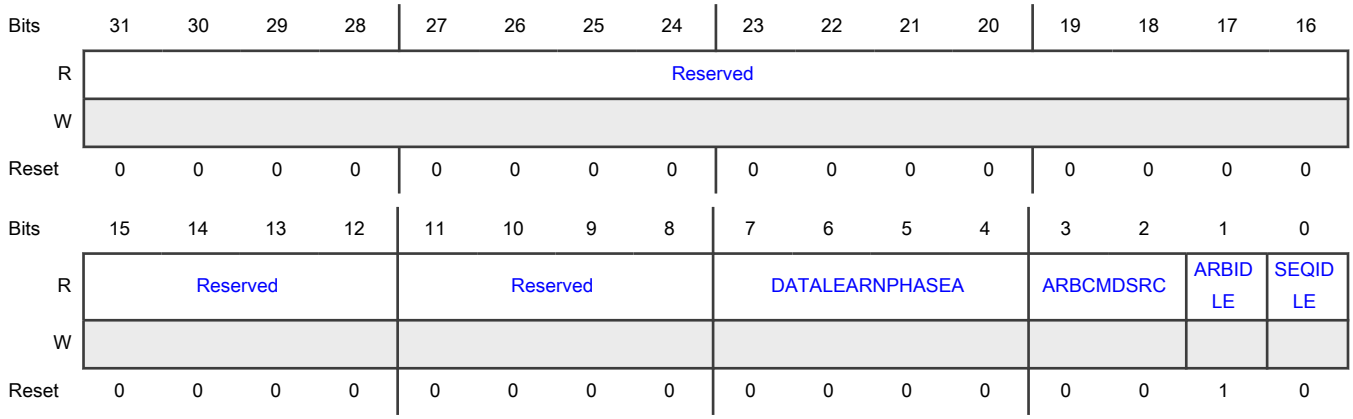
19.7.2.1.21 Status 0 (STS0)

Indicates the status of the internal state machine.

Offset

Register	Offset
STS0	E0h

Diagram



Fields

Field	Description
31-12 —	Reserved
11-8 —	Reserved
7-4 DATALEARNPHASEA	Data Learning Phase Selection on Port A Indicates the sampling clock phase selection on Port A after data learning. There are 16 clock phases for the sampling clock that the delay cell line generates. When data learning is not enabled, the default clock phase 0 is used to sample flash read data. When data learning is enabled and a LEARN_SDR or LEARN_DDR instruction has executed correctly, FlexSPI determines the correct clock phase to sample read data. See Data learning.
3-2 ARBCMDSRC	ARB Command Source Indicates the trigger source of current command sequence that the arbitrator has granted. When ARB_CTL is not busy (STS0[ARBIDLE] = 1), the value of this field does not matter. 00 - Trigger source is AHB read command. 01 - Trigger source is AHB write command. 10 - Trigger source is IP command (by writing 1 to IPCMD[TRG]). 11 - Trigger source is a suspended command that has resumed.
1 ARBIDLE	ARB_CTL State Machine Idle Indicates whether the state machine in ARB_CTL is idle, or a command sequence that the arbitrator granted has not finished on the FlexSPI interface. When idle, no transaction is occurring on the FlexSPI interface (STS0[SEQIDLE] = 1). When waiting for the FlexSPI controller to become idle, poll this field instead of STS0[SEQIDLE]. 0 - Not idle 1 - Idle

Table continues on the next page...

Table continued from the previous page...

Field	Description
0 SEQIDLE	SEQ_CTL State Machine Idle Indicates whether the state machine in SEQ_CTL is idle, or a command sequence is executing on the FlexSPI interface. 0 - Not idle 1 - Idle

19.7.2.1.22 Status 1 (STS1)

Indicates the error code of the AHB or IPS interface error response.

Offset

Register	Offset
STS1	E4h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				IPCMDERRCODE				Reserved				IPCMDERRID			
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				AHBCMDERRCODE				Reserved				AHBCMDERRID			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-28 —	Reserved
27-24 IPCMDERRCODE	IP Command Error Code When an IP command Error is detected, indicates the error code. This field returns to zero when INTR[IPCMDERR] is cleared. 0000 - No error 0010 - IP command with JMP_ON_CS instruction used in the sequence

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0011 - Unknown instruction opcode in the sequence 0100 - DUMMY_SDR or DUMMY_RWDS_SDR instruction used in DDR sequence 0101 - DUMMY_DDR or DUMMY_RWDS_DDR instruction used in SDR sequence 0110 - Flash memory access start address exceeds entire flash address range (A1, A2, B1, and B2) 1110 - Sequence execution timeout 1111 - Flash boundary crossed
23-20 —	Reserved
19-16 IPCMDERRID	IP Command Error ID When an IP command error is detected, indicates the sequence index. This field returns to zero when INTR[IPCMDERR] is cleared.
15-12 —	Reserved
11-8 AHBCMDERRCODE	AHB Command Error Code Contains the error code when an AHB command error is detected. This field returns to zero when INTR[AHBCMDERR] is cleared. 0000 - No error 0010 - AHB Write command with JMP_ON_CS instruction used in the sequence 0011 - Unknown instruction opcode in the sequence 0100 - DUMMY_SDR or DUMMY_RWDS_SDR instruction used in DDR sequence 0101 - DUMMY_DDR or DUMMY_RWDS_DDR instruction used in SDR sequence 1110 - Sequence execution timeout
7-4 —	Reserved
3-0 AHBCMDERRID	AHB Command Error ID When an AHB command error is detected, indicates the sequence index. This field returns to zero when INTR[AHBCMDERR] is cleared.

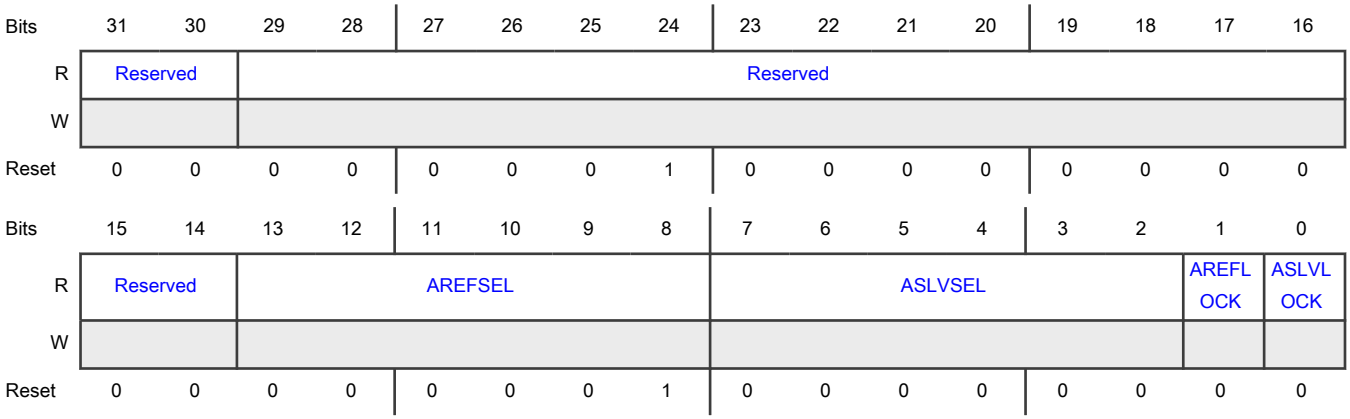
19.7.2.1.23 Status 2 (STS2)

Indicates the status of Flash A sample clock DLLs.

Offset

Register	Offset
STS2	E8h

Diagram



Fields

Field	Description
31-30 —	Reserved
29-16 —	Reserved
15-14 —	Reserved
13-8 AREFSEL	Flash A Sample Clock Reference Delay Line Delay Cell Number Indicates the sample clock reference delay line delay cell number for Flash A. There are 0-63 phases. • 000001b - Indicates lock phase 0. • 100000b - Indicates lock phase 63.
7-2 ASLVSEL	Flash A Sample Clock Target Delay Line Delay Cell Number Indicates the sample clock target delay line delay cell number for Flash A. There are 0-63 phases. • 000001b - Indicates lock phase 0. • 100000b - Indicates lock phase 63.
1 AREFLOCK	Flash A Sample Clock Reference Delay Line Locked 0 - Not locked 1 - Locked

Table continues on the next page...

Table continued from the previous page...

Field	Description
0 ASLVLOCK	Flash A Sample Target Delay Line Locked 0 - Not locked 1 - Locked

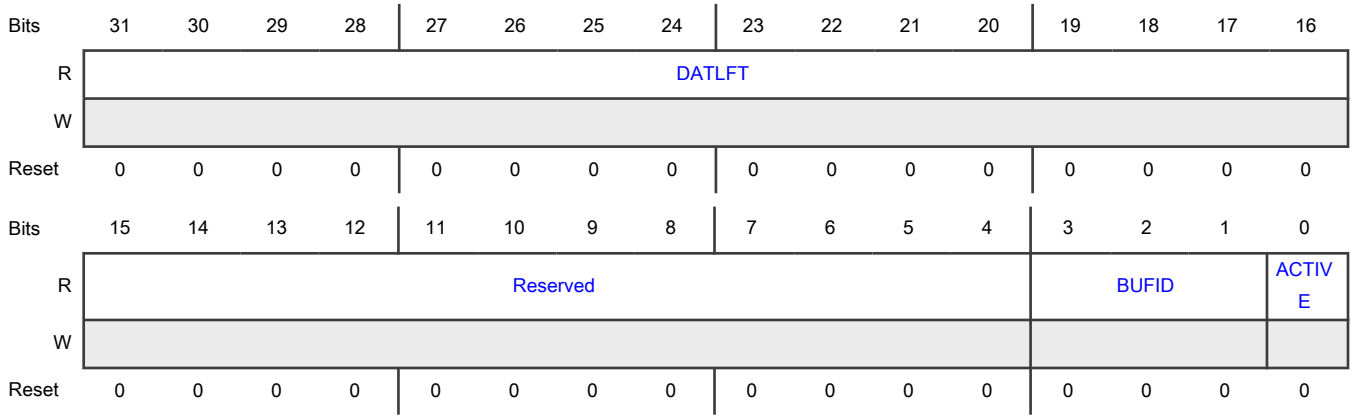
19.7.2.1.24 AHB Suspend Status (AHBSPNDSTS)

Indicates the status of suspended AHB read prefetch command sequence. When an IP or AHB command is triggered while the arbitrator processes an AHB read sequence (prefetching additional data not for current AHB burst), the prefetch sequence is suspended. When there are no longer transactions on FlexSPI, the prefetch sequence may be resumed. FlexSPI saves only one AHB prefetch sequence. When a new prefetch sequence is suspended with an active sequence already suspended, the previous suspended sequence is removed and never resumed. See Command abort and suspend.

Offset

Register	Offset
AHBSPNDSTS	ECh

Diagram



Fields

Field	Description
31-16 DATLFT	Data Left Contains the data size remaining (in bytes) for a suspended command sequence.
15-4 —	Reserved
3-1	AHB Receive Buffer ID for Suspended Command Sequence

Table continues on the next page...

Table continued from the previous page...

Field	Description
BUFID	
0 ACTIVE	Active AHB Read Prefetch Suspended Indicates whether an AHB read prefetch command sequence has been suspended. 0 - No suspended AHB read prefetch command. 1 - An AHB read prefetch command sequence has been suspended.

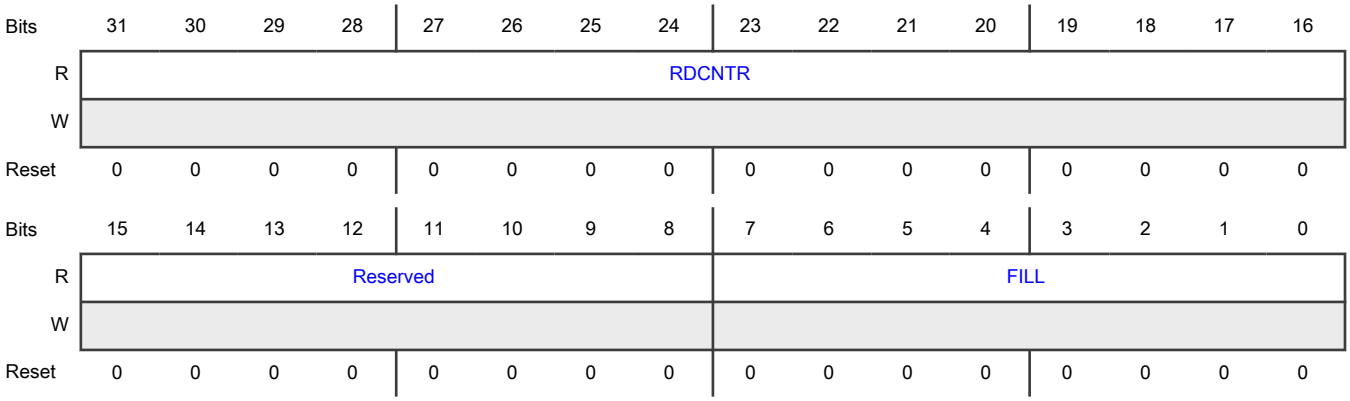
19.7.2.1.25 IP Receive FIFO Status (IPRXFSTS)

Indicates the status of IP receive FIFO.

Offset

Register	Offset
IPRXFSTS	F0h

Diagram



Fields

Field	Description
31-16 RDCNTR	Read Data Counter Contains counter for total data read. The data read is the value of this field × 64 bits.
15-8 —	Reserved
7-0 FILL	Fill Level of IP Receive FIFO Indicates how full the IP receive FIFO is. The fill level is the value of this field × 64 bits.

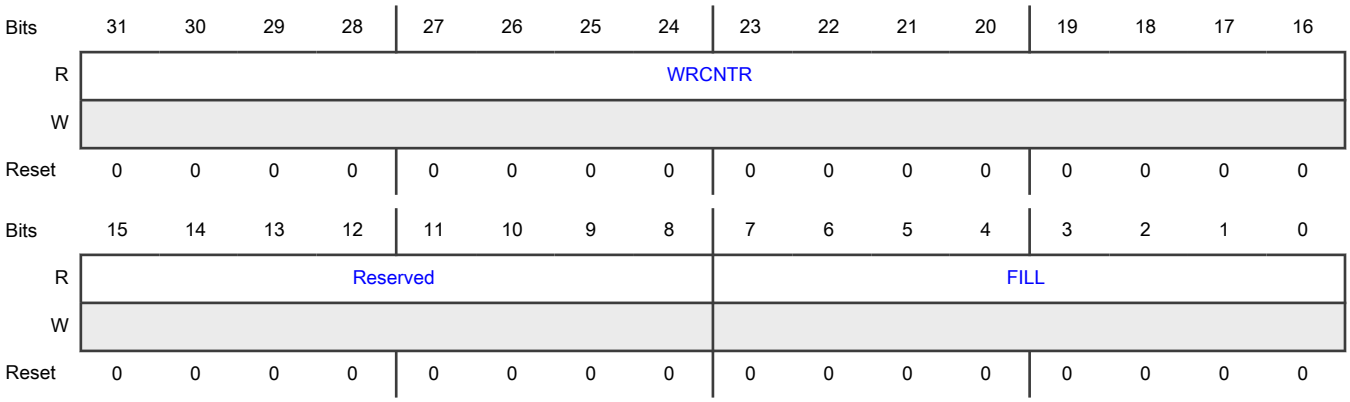
19.7.2.1.26 IP Transmit FIFO Status (IPTXFSTS)

Indicates the status of IP transmit FIFO.

Offset

Register	Offset
IPTXFSTS	F4h

Diagram



Fields

Field	Description
31-16 WRCNTR	Write Data Counter Contains counter for total data written. The data written is the value of this field × 64 bits.
15-8 —	Reserved
7-0 FILL	Fill Level of IP Transmit FIFO Indicates how full the IP transmit FIFO is. The fill level is the value of this field × 64 bits.

19.7.2.1.27 IP Receive FIFO Data a (RFDR0 - RFDR31)

Provides read access to IP receive FIFO via IPS bus. The read value is unknown for read access to invalid entries in the IP receive FIFO.

Offset

For a = 0 to 31:

Register	Offset
RFDRa	100h + (a × 4h)

Diagram

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

Fields

Field	Description
31-0 RXDATA	Receive Data Contains receive data. See Reading data from IP receive FIFO.

19.7.2.1.28 IP TX FIFO Data a (TFDR0 - TFDR31)

Provides write access to IP transmit FIFO via IPS bus.

Offset

For a = 0 to 31:

Register	Offset
TFDRa	180h + (a × 4h)

Diagram

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

Fields

Field	Description
31-0 TXDATA	Transmit Data Contains transmit data. See Writing data to IP transmit FIFO.

19.7.2.1.29 Lookup Table a (LUT0 - LUT63)

Contains a lookup table for command sequences. Software should set the sequence index before triggering an IP command or AHB command. FlexSPI fetches the command sequence from LUT when an IP or AHB command is triggered. There are 16 command sequences in LUT. See Lookup table (LUT) and Programmable sequence engine.

NOTE

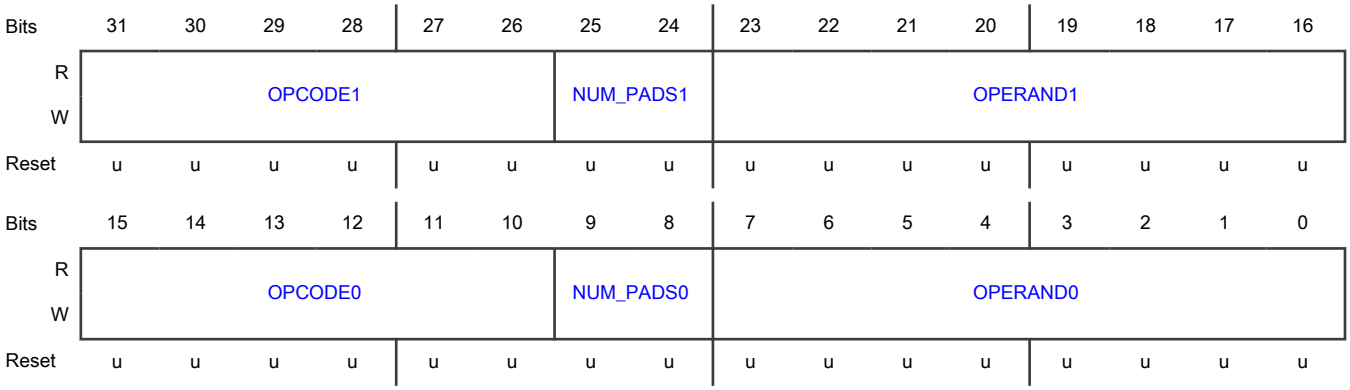
LUT is implemented as memory, so the reset value is unknown.

Offset

For a = 0 to 63:

Register	Offset
LUTa	200h + (a × 4h)

Diagram



Fields

Field	Description
31-26 OPCODE1	OPCODE1
25-24 NUM_PADS1	NUM_PADS1
23-16	OPERAND1

Table continues on the next page...

Table continued from the previous page...

Field	Description
OPERAND1	
15-10 OPCODE0	OPCODE
9-8 NUM_PADS0	NUM_PADS0
7-0 OPERAND0	OPERAND0

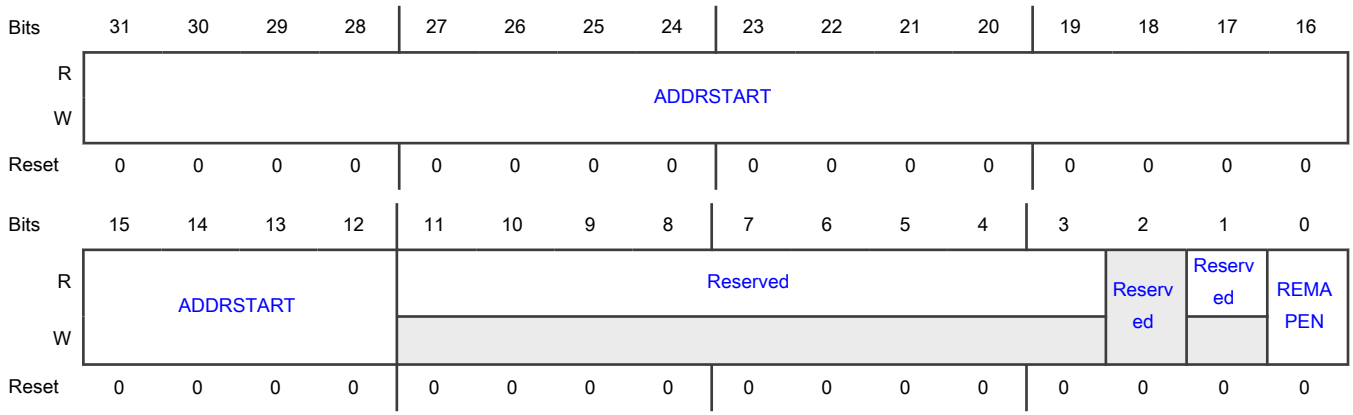
19.7.2.1.30 HADDR REMAP Start Address (HADDRSTART)

Used to configure options for the dual-image remap feature. See Dual image use case using HADDRSTART, HADDREND, and HADDROFFSET registers.

Offset

Register	Offset
HADDRSTART	420h

Diagram



Fields

Field	Description
31-12 ADDRSTART	HADDR Start Address Contains the start address of the HADDR remap range, aligned to 4 KB.
11-3	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
2 —	Reserved
1 —	Reserved
0 REMAPEN	AHB Bus Address Remap Enable 0 - HADDR REMAP Disabled 1 - HADDR REMAP Enabled

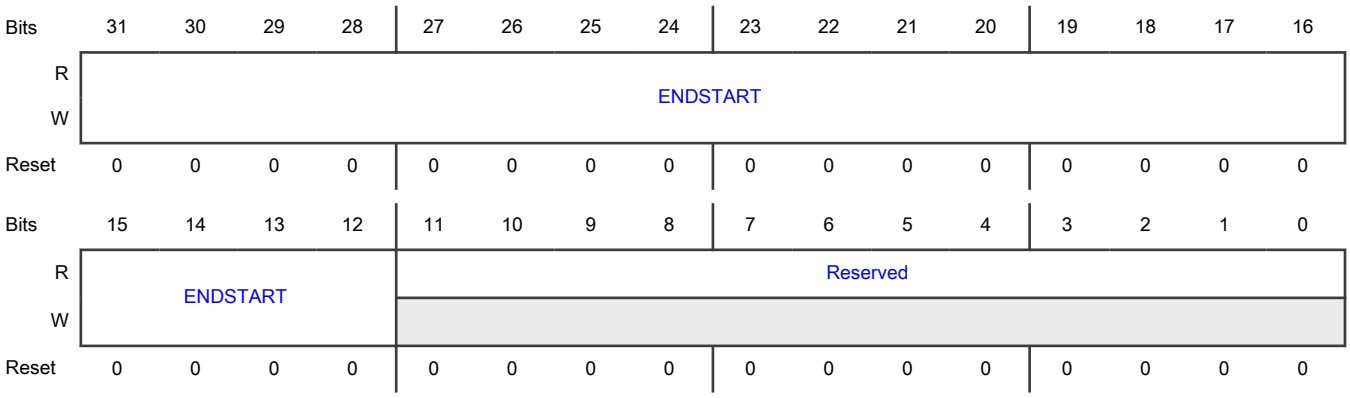
19.7.2.1.31 HADDR REMAP END ADDR (HADDREND)

Used to configure options for the dual-image remap feature. See Dual image use case using HADDRSTART, HADDREND, and HADDROFFSET registers.

Offset

Register	Offset
HADDREND	424h

Diagram



Fields

Field	Description
31-12 ENDSTART	End Address of HADDR Remap Range Contains the end address of the HADDR remap range, aligned to 4 KB.

Table continues on the next page...

Table continued from the previous page...

Field	Description
11-0 —	Reserved

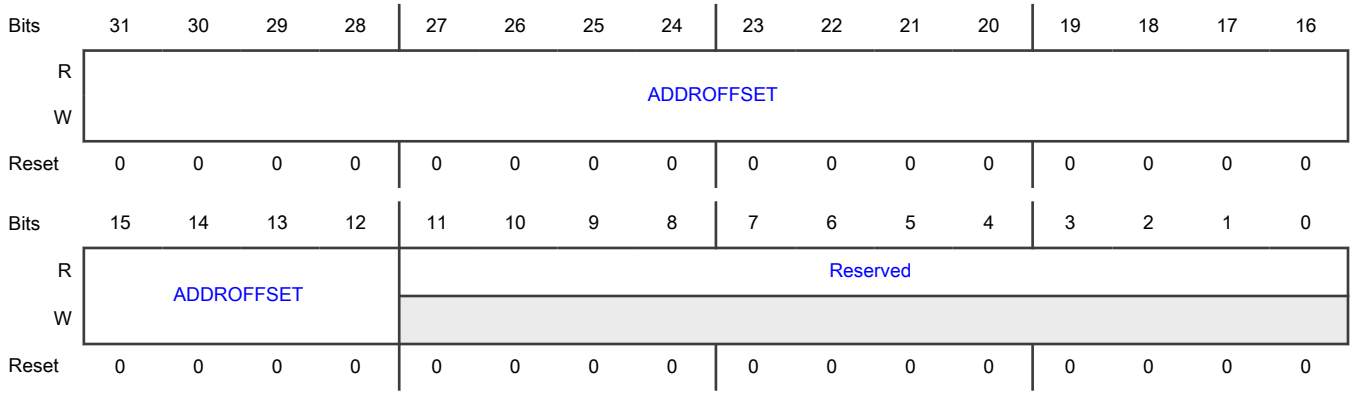
19.7.2.1.32 HADDR Remap Offset (HADDROFFSET)

Used to configure options for the dual-image remap feature. See Dual image use case using HADDRSTART, HADDREND, and HADDROFFSET registers.

Offset

Register	Offset
HADDROFFSET	428h

Diagram



Fields

Field	Description
31-12 ADDROFFSET	HADDR Offset Contains the offset for HADDR. The remapped address is ADDR[31:12] = ADDR_original[31:12] + ADDROFFSET.
11-0 —	Reserved

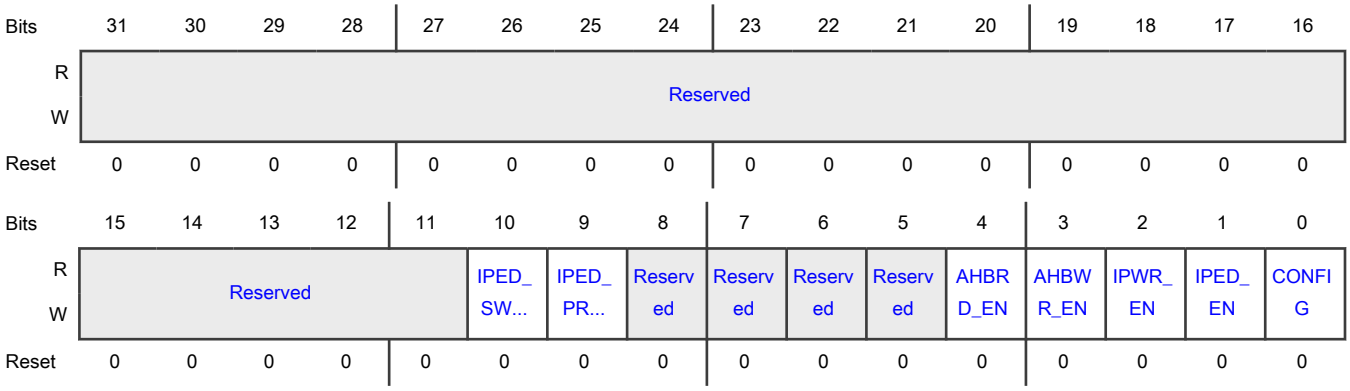
19.7.2.1.33 IPED Function Control (IPEDCTRL)

Selects IPED mode for encryption and decryption.

Offset

Register	Offset
IPEDCTRL	42Ch

Diagram



Fields

Field	Description
31-11 —	Reserved
10 IPED_SWRESET	Abort Current Decryption or Encryption 0 - No function. 1 - Aborts current decryption or encryption and waits for the next start operation.
9 IPED_PROTECT	IPED Protection Can be used to limit access to IPED configuration registers. 0 - No restrictions 1 - Only privileged controllers can write IPED registers.
8 —	Reserved
7 —	Reserved
6 —	Reserved
5 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
4 AHBRD_EN	AHB Read IPED CTR Mode Decryption Enable 0 - Disable 1 - Enable
3 AHBWR_EN	AHB Write IPED CTR Mode Encryption Enable. 0 - Disable 1 - Enable
2 IPWR_EN	IP Write IPED CTR Mode Encryption Enable 0 - Disable 1 - Enable
1 IPED_EN	IPED Encryption and Decryption Enable 0 - Disable 1 - Enable
0 CONFIG	IPED Mode Select Configures whether encryption and decryption are fully pipelined. 0 - Fully pipelined 1 - Not fully pipelined

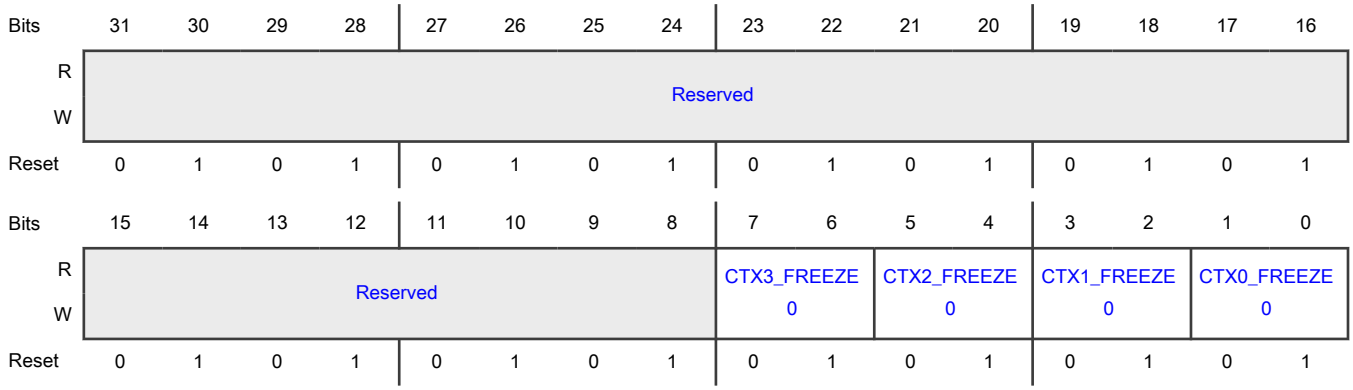
19.7.2.1.34 IPED context control 0 (IPEDCTXCTRL0)

Freezes the IPEDCTXnSTART and IPEDCTXnEND registers. If bit1 and bit0 of the CTXn_FREEZE0 field are different, and IPEDCTXCTRL1[CTXn_FREEZE1] = 10b, CTXn_FREEZE0 is writable. See Inline PRINCE encryption and decryption (IPED) CTR.

Offset

Register	Offset
IPEDCTXCTRL0	500h

Diagram



Fields

Field	Description
31-8 —	Reserved
7-6 CTX3_FREEZE0	Context Register Freeze for Region 3 Controls the read and write properties of this field and region 3 context registers (IPEDCTX3xxxx). This field freezes the IPEDCTX3START and IPEDCTX3END registers. If bit1 and bit0 of CTX3_FREEZE0 field are different, and IPEDCTXCTRL1[CTX3_FREEZE1] = 10b, CTX3_FREEZE0 is writable.
5-4 CTX2_FREEZE0	Context Register Freeze for Region 2 Controls the read and write properties of this field and region 2 context registers (IPEDCTX2xxxx). This field freezes the IPEDCTX2START and IPEDCTX2END registers. If bit1 and bit0 of CTX2_FREEZE0 field are different, and IPEDCTXCTRL1[CTX2_FREEZE1] = 10b, CTX2_FREEZE0 is writable.
3-2 CTX1_FREEZE0	Context Register Freeze for Region 1 Controls the read and write properties of this field and region 1 context registers (IPEDCTX1xxxx). This field freezes the IPEDCTX1START and IPEDCTX1END registers. If bit1 and bit0 of CTX1_FREEZE0 field are different, and IPEDCTXCTRL1[CTX1_FREEZE1] = 10b, CTX1_FREEZE0 is writable.
1-0 CTX0_FREEZE0	Context Register Freeze for Region 0 Controls the read and write properties of this field and region 0 context registers (IPEDCTX0xxxx). This field freezes the IPEDCTX0START and IPEDCTX0END registers. If bit1 and bit0 of CTX0_FREEZE0 field are different, and IPEDCTXCTRL1[CTX0_FREEZE1] = 10b, CTX0_FREEZE0 is writable.

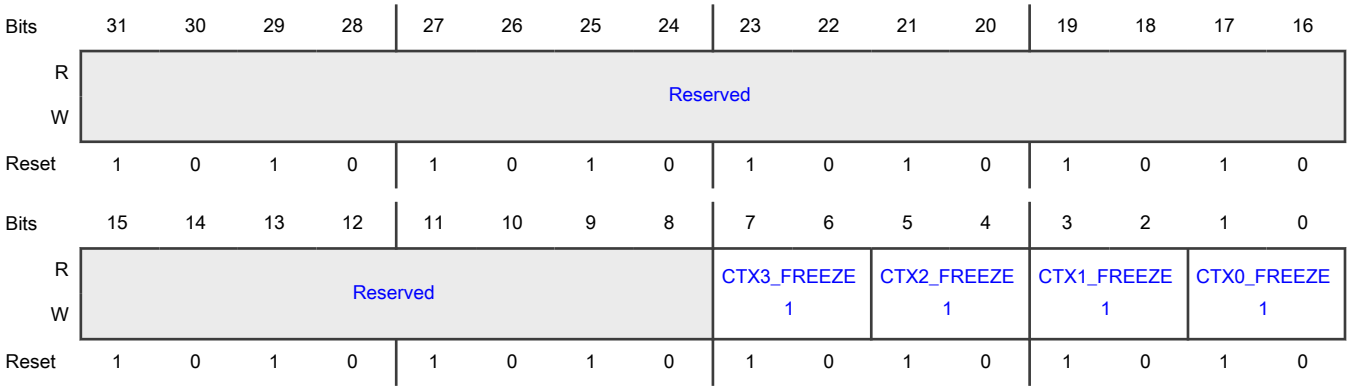
19.7.2.1.35 IPED context control 1 (IPEDCTXCTRL1)

Freezes the IPEDCTXnSTART and IPEDCTXnEND registers. If bit1 and bit0 of the CTXn_FREEZE0 field are different, and IPEDCTXCTRL1[CTXn_FREEZE1] = 10b, CTXn_FREEZE1 is writable. See Inline PRINCE encryption and decryption (IPED) CTR.

Offset

Register	Offset
IPEDCTXCTRL1	504h

Diagram



Fields

Field	Description
31-8 —	Reserved
7-6 CTX3_FREEZE1	Context Register Freeze for Region 3 Controls the read and write properties of this field and region 3 context registers (IPEDCTX3xxxx). This field freezes the IPEDCTX3START and IPEDCTX3END registers. If bit1 and bit0 of CTX3_FREEZE0 field are different, and IPEDCTXCTRL1[CTX3_FREEZE1] = 10b, CTX3_FREEZE1 is writable.
5-4 CTX2_FREEZE1	Context Register Freeze for Region 2 Controls the read and write properties of this field and region 2 context registers (IPEDCTX2xxxx). This field freezes the IPEDCTX2START and IPEDCTX2END registers. If bit1 and bit0 of CTX2_FREEZE0 field are different, and IPEDCTXCTRL1[CTX2_FREEZE1] = 10b, CTX2_FREEZE1 is writable.
3-2 CTX1_FREEZE1	Context Register Freeze for Region 1 Controls the read and write properties of this field and region 1 context registers (IPEDCTX1xxxx). This field freezes the IPEDCTX1START and IPEDCTX1END registers. If bit1 and bit0 of CTX1_FREEZE0 field are different, and IPEDCTXCTRL1[CTX1_FREEZE1] = 10b, CTX1_FREEZE1 is writable.
1-0 CTX0_FREEZE1	Context Register Freeze for Region 0 Controls the read and write properties of this field and region 0 context registers (IPEDCTX0xxxx). This field freezes the IPEDCTX0START and IPEDCTX0END registers. If bit1 and bit0 of CTX0_FREEZE0 field are different, and IPEDCTXCTRL1[CTX0_FREEZE1] = 10b, CTX0_FREEZE1 is writable.

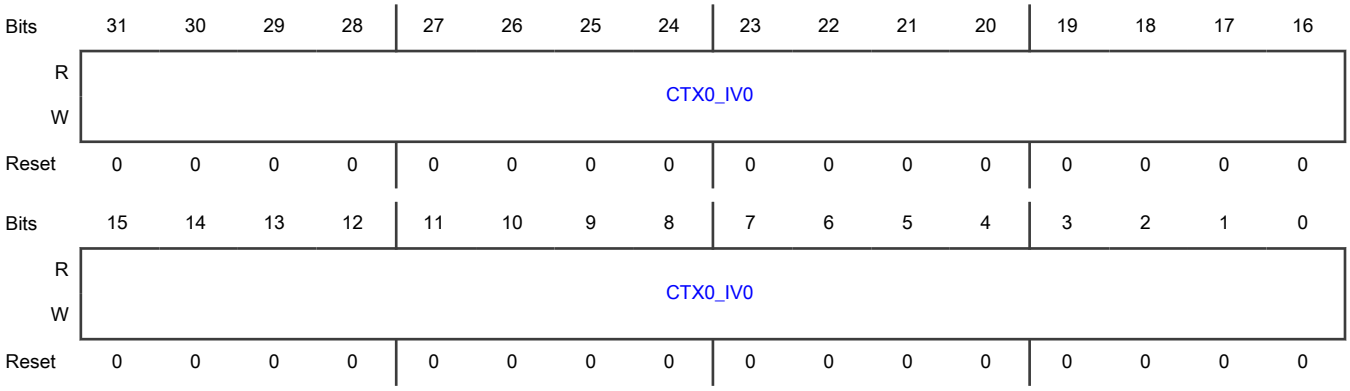
19.7.2.1.36 IPED Context0 IV0 (IPEDCTX0IV0)

Sets the IPED context IV for encryption and decryption.

Offset

Register	Offset
IPEDCTX0IV0	520h

Diagram



Fields

Field	Description
31-0 CTX0_IV0	Lowest 32 bits of IV for region 0.

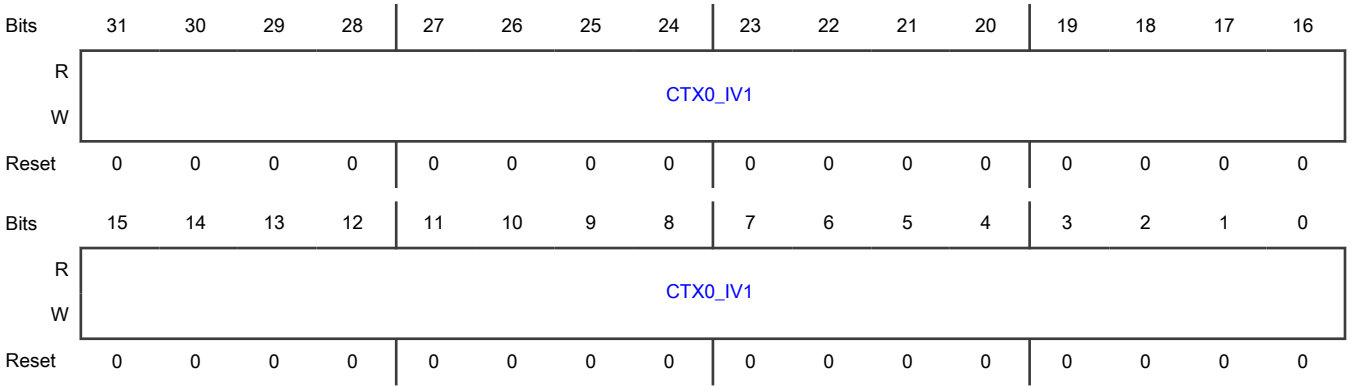
19.7.2.1.37 IPED Context0 IV1 (IPEDCTX0IV1)

Sets the IPED context IV for encryption and decryption.

Offset

Register	Offset
IPEDCTX0IV1	524h

Diagram



Fields

Field	Description
31-0 CTX0_IV1	Highest 32 bits of IV for region 0.

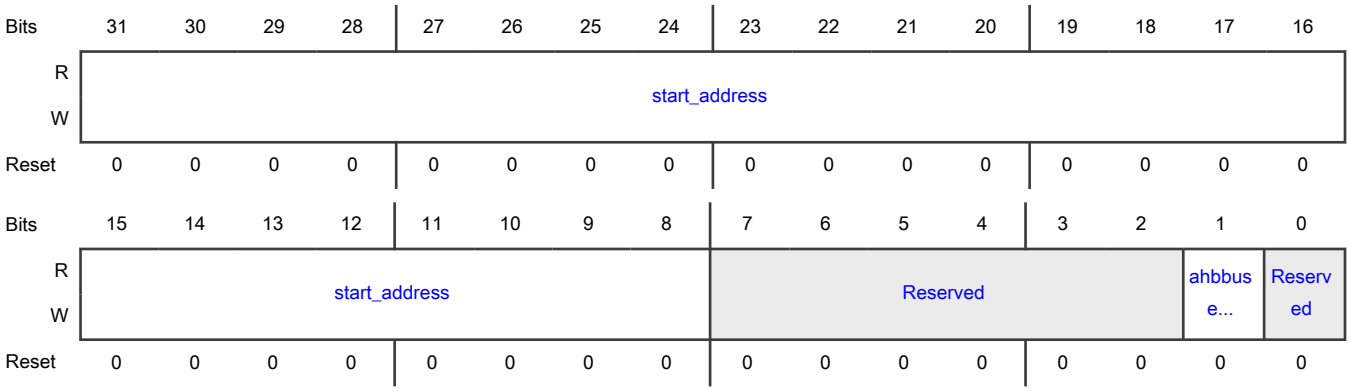
19.7.2.1.38 Start Address of Region (IPEDCTX0START - IPEDCTX3START)

Indicates start address of region. If the CTXn_FREEZE0 fields of the IPEDCTXCTRL0 register are 10b and CTXn_FREEZE1 fields of IPEDCTXCTRL1 register are 10b, then IPEDCTXnSTART is writable.

Offset

Register	Offset
IPEDCTX0START	528h
IPEDCTX1START	548h
IPEDCTX2START	568h
IPEDCTX3START	588h

Diagram



Fields

Field	Description
31-8 start_address	Start Address Indicates start address of IPED region. The start address must be at least 256-byte aligned. The address used here is the AHB address that is compared with the system bus address.
7-2 —	Reserved
1	AHB Bus Error Disable

Table continues on the next page...

Table continued from the previous page...

Field	Description
ahbbuserror_dis	Used to configure whether IPED errors generate bus errors on an AHB access. 0 - AHB bus errors enabled 1 - AHB bus errors disabled
0 —	Reserved

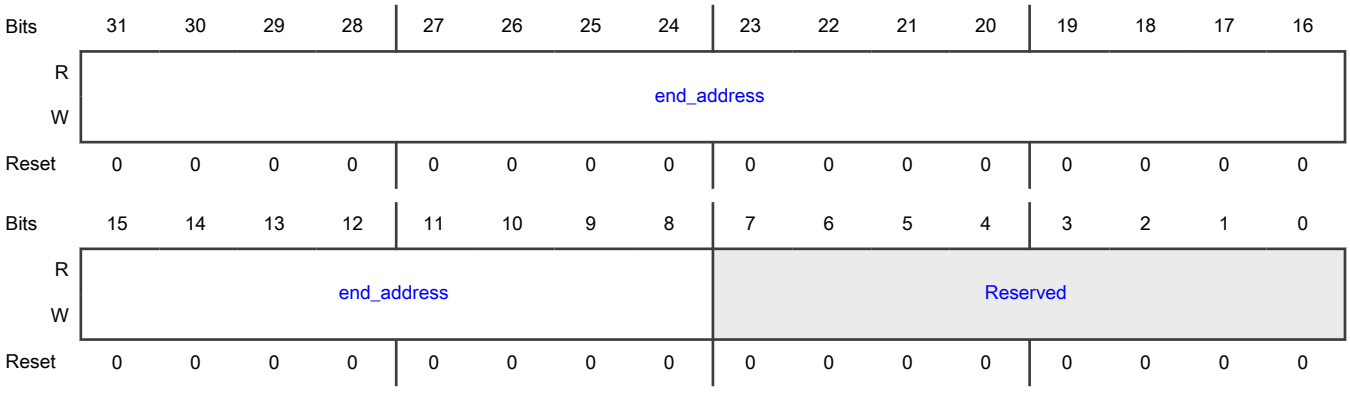
19.7.2.1.39 End Address of Region (IPEDCTX0END - IPEDCTX3END)

Indicates end address of region. If IPEDCTXCTRL0[CTXn_FREEZE0] = 10b and IPEDCTXCTRL1[CTXn_FREEZE1] = 10b, IPEDCTXnEND is writable.

Offset

Register	Offset
IPEDCTX0END	52Ch
IPEDCTX1END	54Ch
IPEDCTX2END	56Ch
IPEDCTX3END	58Ch

Diagram



Fields

Field	Description
31-8 end_address	End Address of IPED Region Contains the end address of the IPED region. Must be at least 256-byte aligned. This address is the AHB address that is compared with the system bus address.

Table continues on the next page...

Table continued from the previous page...

Field	Description
7-0 —	Reserved

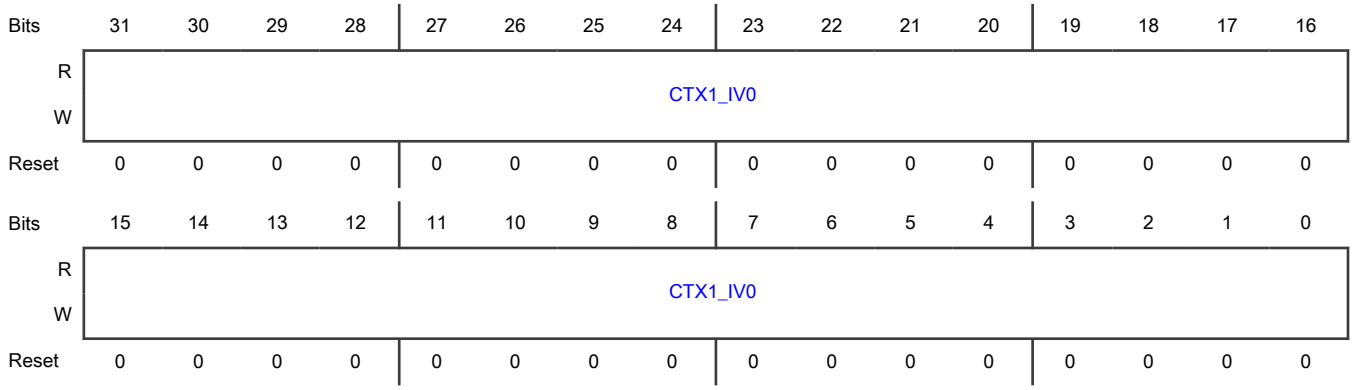
19.7.2.1.40 IPED Context1 IV0 (IPEDCTX1IV0)

Sets the IPED context IV for encryption and decryption.

Offset

Register	Offset
IPEDCTX1IV0	540h

Diagram



Fields

Field	Description
31-0 CTX1_IV0	Lowest 32 bits of IV for region 1.

19.7.2.1.41 IPED Context1 IV1 (IPEDCTX1IV1)

Sets the IPED context IV for encryption and decryption.

Offset

Register	Offset
IPEDCTX1IV1	544h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CTX1_IV1															
Reset	0	0	0	0	0	0	0	0	0	0	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	CTX1_IV1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 CTX1_IV1	Highest 32 bits of IV for region 1.

19.7.2.1.42 IPED Context2 IV0 (IPEDCTX2IV0)

Sets the IPED context IV for encryption and decryption.

Offset

Register	Offset
IPEDCTX2IV0	560h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CTX2_IV0															
Reset	0	0	0	0	0	0	0	0	0	0	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	CTX2_IV0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 CTX2_IV0	Lowest 32 bits of IV for region 2.

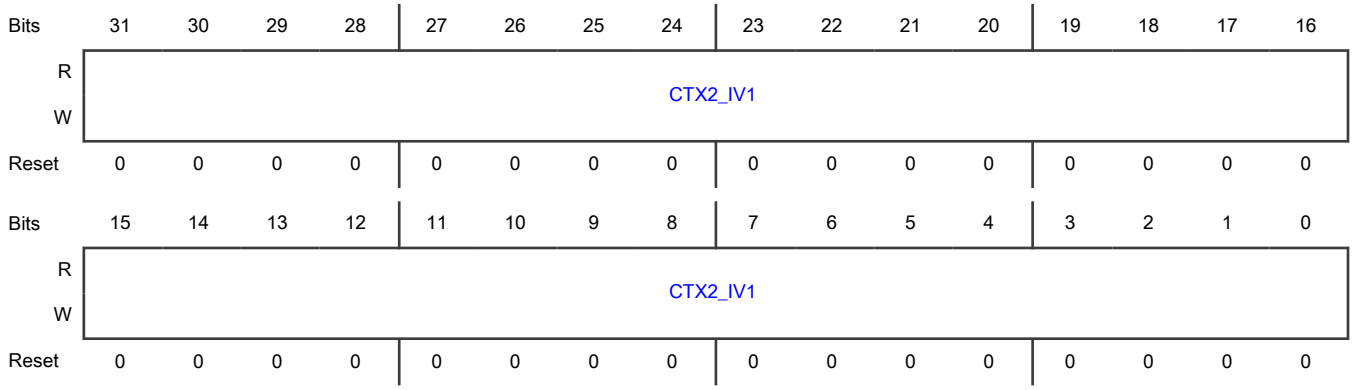
19.7.2.1.43 IPED Context2 IV1 (IPEDCTX2IV1)

Sets the IPED context IV for encryption and decryption.

Offset

Register	Offset
IPEDCTX2IV1	564h

Diagram



Fields

Field	Description
31-0 CTX2_IV1	Highest 32 bits of IV for region 2.

19.7.2.1.44 IPED Context3 IV0 (IPEDCTX3IV0)

Sets the IPED context IV for encryption and decryption.

Offset

Register	Offset
IPEDCTX3IV0	580h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CTX3_IV0															
Reset	0	0	0	0	0	0	0	0	0	0	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	CTX3_IV0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 CTX3_IV0	Lowest 32 bits of IV for region 3.

19.7.2.1.45 IPED Context3 IV1 (IPEDCTX3IV1)

Sets the IPED context IV for encryption and decryption.

Offset

Register	Offset
IPEDCTX3IV1	584h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CTX3_IV1															
Reset	0	0	0	0	0	0	0	0	0	0	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	CTX3_IV1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 CTX3_IV1	Highest 32 bits of IV for region 3.

19.8 AHB memory map definition

This section describes FlexSPI module AHB memory map in detail.

19.8.1 AHB memory map for serial flash memory and RAM access

AHB read and write access for serial flash and RAM memory are mapped to a specific address range. See the system memory map for specific address ranges supported.

AHB bus features supported for serial flash and RAM memory reading:

- Cacheable and non-cacheable access
- Prefetch enable and disable
- Burst size: 8, 16, 32, 64 bits
- All burst types: types: SINGLE, INCR, WRAP4, INCR4, WRAP8, INCR8, WRAP16, INCR16

AHB bus features for Serial RAM writing:

- Bufferable and Non-Bufferable access
- Burst size: 8, 16, 32, 64 bits
- All burst types: SINGLE, INCR, WRAP4, INCR4, WRAP8, INCR8, WRAP16, INCR16

See Flash memory access via AHB command for details about AHB access to serial flash memory and RAM.

Chapter 20
DMA Controller

20.1 Chip-specific DMA controller information

Table 98. Reference links to related information

Topic	Related module	Reference
Full description	DMA Controller	DMA Controller
System memory map	System memory map	System memory map
Clocking	SYSCON	Clock distribution
Power management	Power management	Power management
Signal multiplexing	Port control	Signal multiplexing

20.1.1 Module Instances

The chip includes two DMA controllers. DMA controller 0 supports 52 channels and DMA controller 1 supports 16 channels.

20.1.2 Channel Descriptors for DMA channels

Table 99. Channel Descriptor offsets for DMA channels on the chip

Descriptor	Offset
Channel Descriptor for DMA channel 0	0x000
Channel Descriptor for DMA channel 1	0x010
Channel Descriptor for DMA channel 2	0x020
Channel Descriptor for DMA channel 3	0x030
Channel Descriptor for DMA channel 4	0x040
Channel Descriptor for DMA channel 5	0x050
Channel Descriptor for DMA channel 6	0x060
Channel Descriptor for DMA channel 7	0x070
Channel Descriptor for DMA channel 8	0x080
Channel Descriptor for DMA channel 9	0x090
Channel Descriptor for DMA channel 10	0x0A0
Channel Descriptor for DMA channel 11	0x0B0
Channel Descriptor for DMA channel 12	0x0C0

Table continues on the next page...

Table 99. Channel Descriptor offsets for DMA channels on the chip (continued)

Descriptor	Offset
Channel Descriptor for DMA channel 13	0x0D0
Channel Descriptor for DMA channel 14	0x0E0
Channel Descriptor for DMA channel 15	0x0F0
Channel Descriptor for DMA channel 16	0x100
Channel Descriptor for DMA channel 17	0x110
Channel Descriptor for DMA channel 18	0x120
Channel Descriptor for DMA channel 19	0x130
Channel Descriptor for DMA channel 20	0x140
Channel Descriptor for DMA channel 21	0x150
Channel Descriptor for DMA channel 22	0x160
Channel Descriptor for DMA channel 23	0x170
Channel Descriptor for DMA channel 24	0x180
Channel Descriptor for DMA channel 25	0x190
Channel Descriptor for DMA channel 26	0x1A0
Channel Descriptor for DMA channel 27	0x1B0
Channel Descriptor for DMA channel 28	0x1C0
Channel Descriptor for DMA channel 29	0x1D0
Channel Descriptor for DMA channel 30	0x1E0
Channel Descriptor for DMA channel 31	0x1F0
Channel Descriptor for DMA channel 32	0x200
Channel Descriptor for DMA channel 33	0x210
Channel Descriptor for DMA channel 34	0x220
Channel Descriptor for DMA channel 35	0x230
Channel Descriptor for DMA channel 36	0x240
Channel Descriptor for DMA channel 37	0x250
Channel Descriptor for DMA channel 38	0x260
Channel Descriptor for DMA channel 39	0x270
Channel Descriptor for DMA channel 40	0x280
Channel Descriptor for DMA channel 41	0x290

Table continues on the next page...

Table 99. Channel Descriptor offsets for DMA channels on the chip (continued)

Descriptor	Offset
Channel Descriptor for DMA channel 42	0x2A0
Channel Descriptor for DMA channel 43	0x2B0
Channel Descriptor for DMA channel 44	0x2C0
Channel Descriptor for DMA channel 45	0x2D0
Channel Descriptor for DMA channel 46	0x2E0
Channel Descriptor for DMA channel 47	0x2F0
Channel Descriptor for DMA channel 48	0x300
Channel Descriptor for DMA channel 49	0x310
Channel Descriptor for DMA channel 50	0x320
Channel Descriptor for DMA channel 51	0x330
Channel Descriptor for DMA channel 52	0x340

20.1.3 Chip-specific DMA Initialization

Configure each DMA controller as follows:

- Use the AHBCLKCTRL0 and AHBCLKCTRL1 register (See SYSCON) to enable the clock to the DMA0 and DMA1 registers interface.
- Clear the appropriate DMA peripheral reset in the PRESETCTRL register by writing to the PRESETCTRL_CLR register (See SYSCON).
- The DMA controllers provide interrupts to the NVIC, (see NVIC). These interrupts can alternatively be connected to the DSP (See INPUTMUX).
- Most peripherals that support DMA, the ADC being an exception, have at least one DMA request line associated with them. The related channel(s) must be set up according to the desired operation. ADC uses a trigger instead of a DMA request. DMA requests and triggers are described in detail in DMA Requests and Hardware Triggers.
- For peripherals using DMA requests, DMA operation must be triggered before any transfer will occur. This can be done by software, or can optionally be signalled by one of the hardware triggers, through the input mux registers DMA_ITRIG_SEL (See INPUTMUX).
- Trigger outputs may optionally cause other DMA channels to be triggered for more complex DMA functions. Trigger outputs are connected to DMA_OTRIG_SEL (See INPUTMUX)as inputs to DMA triggers.
- The HWWAKE feature may be used in conjunction with certain peripherals to allow DMA to operate while the device is in the deep-sleep reduced power mode, without waking up the CPU. (See SYSCTL).
- The DMA controllers have an AHB master function. The priority of this and other AHB bus masters can be adjusted if needed to achieve system performance needed by an application by using the SYSCTL0_AHBMATRIXPRIOR register (See SYSCTL).

For details on the trigger input and output multiplexing, see INPUTMUX.

20.2 Overview

This section provides an introduction to the DMA Controller.

20.2.1 Block diagram

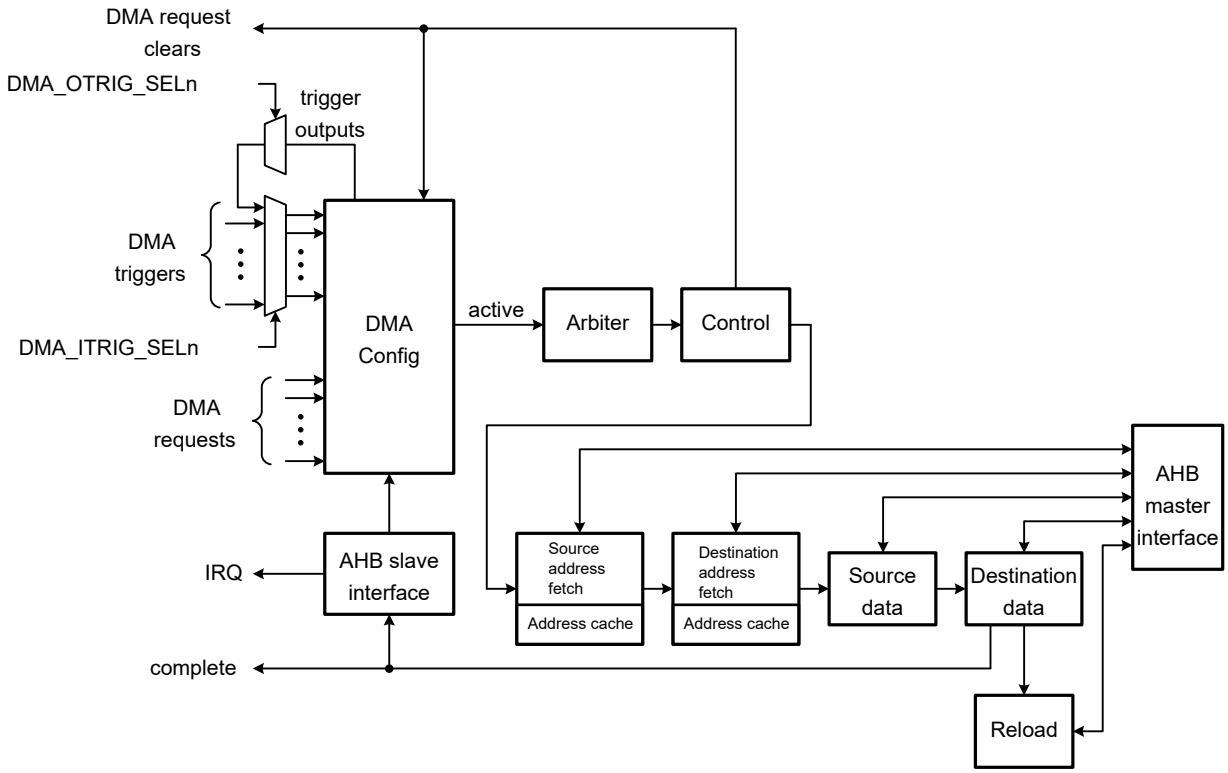


Figure 73. DMA block diagram

20.2.2 Features

DMA controller has the features below:

- Up to 52 channels, 52 of which are connected to fixed peripheral DMA requests, dedicated to that particular channel. DMA channels with no DMA request connected or that are otherwise unused can be used for functions such as memory-to-memory moves, or triggered transfers.
- 52 potential DMA triggers for each channel: Some peripherals use the trigger instead of a DMA request for pacing the transfer. Other peripherals can have their transfer started by a trigger, then paced by the DMA request. Simple transfers, such as memory-to-memory, can be triggered by (for instance) a timer match or other event. Trigger sources include ADC interrupts, Timer interrupts, pin interrupts, and the SCT DMA request lines, among others.
- Channels can be chained together, allowing one channel to be triggered by a completion of an operation on another channel. For instance, a memory-to-memory transfer could be triggered by completion of a USART buffer being filled.
- Priority is user selectable for each channel (up to eight priority levels).
- Continuous priority arbitration.
- Address cache with eight entries (each entry is a pair of transfer addresses).
- Supports single transfers up to 1,024 words. A single transfer is defined as one read from the source address, followed by one write to the destination address.
- Address increment options allow packing and/or unpacking data.

20.3 Signals

The DMA controller is not associated with any device pins. However, some DMA triggers can be taken from device pins (see Hardware triggers).

20.4 Functional description

The following sections describe functional details of the DMA controller.

20.4.1 Modes of operation

The DMA controller does not have separate operating modes.

Using any specific DMA channel requires initializing the device registers associated with that channel, and setting up the Channel Descriptor. The Channel Descriptor is located in system memory, typically in on-chip SRAM (see SRAM address of the channel configuration table (SRAMBASE)).

The memory locations in a Channel Descriptor are actively used by the DMA controller when it operates on the related channel. So it must be re-initialized if the channel is used again. This is not true for additional reload descriptors that are linked to from the Channel Descriptor.

Table 100. Memory locations in a Channel Descriptor

Offset	Description
+ 0x0	Reserved
+ 0x4	Source data end address
+ 0x8	Destination end address
+ 0xC	Link to next descriptor

Table 101. Reload descriptors

Offset	Description
+ 0x0	Transfer configuration.
+ 0x4	Source end address. This points to the address of the last entry of the source address range if the address is incremented. The address to be used in the transfer is calculated from the end address, data width, and transfer size.
+ 0x8	Destination end address. This points to the address of the last entry of the destination address range if the address is incremented. The address to be used in the transfer is calculated from the end address, data width, and transfer size.
+ 0xC	Link to next descriptor. If used, this address must be aligned to a multiple of 16 bytes (the size of a descriptor).

20.4.1.1 DMA in low power modes

DMA in sleep mode:

- In sleep mode, the DMA can operate and access all enabled SRAM blocks without waking up the CPU.

DMA in deep-sleep mode:

- Some peripherals support DMA during deep-sleep mode without waking up the CPU or the rest of the device. These peripherals are the Flexcomm functions that include FIFO support (USART, SPI, and I2S), and the DMIC.
- These wake-ups are based on peripheral FIFO levels, not directly related to peripheral DMA requests and interrupts. See SYSCTL for more information.

20.4.2 DMA Transfers

DMA transfer is initiated by writing Configuration register for DMA channel (CFG0 - CFG51) and RELOAD registers and setting ENA bits in Channel Enable read and set for all DMA channels (ENABLESET0). DMA transfer can be started by a hardware or software trigger.

For data transfer from memory to a peripheral or from a peripheral to memory, the address used for SRCINC or DSTINC, whichever corresponds to the fixed peripheral data address, is the address of the peripheral data register. See Transfer configuration register for DMA channel (XFERCFG0 - XFERCFG51). The memory address for such a transfer is based on the end (upper) address of the memory buffer. The value can be calculated from the starting address of the buffer and the length of the buffer, where the transfer increment is the value specified by SRCINC or DSTINC (whichever corresponds to the memory buffer)

Buffer ending address = Buffer starting address + (XFERCOUNT * Transfer increment)

NOTE

XFERCOUNT = Actual count - 1

After the channel has had a sufficient number of DMA requests and/or triggers, depending on its configuration, the initial descriptor will be exhausted. At that point, if the transfer configuration directs it, the Channel Descriptor will be reloaded with data from memory pointed to by the "Link to next descriptor" entry of the Channel Descriptor. Descriptors loaded in this manner are shown in Reload Descriptors. The difference is that a new transfer configuration is specified in the reload descriptor instead of being written to the XFERCFG register for that channel.

This process repeats as each descriptor is exhausted as long as reload is selected in the transfer configuration for each new descriptor.

20.4.2.1 Single buffer

This generally applies to memory to memory transfers, and peripheral DMA that occurs only occasionally and is set up for each transfer. For this operation, only the initial Channel Descriptor as shown in the table below is needed.

Table 102. Channel Descriptor for a single buffer

Offset	Description
+ 0x0	Reserved
+ 0x4	Source data end address
+ 0x8	Destination data end address
+ 0xC	(not used)

For single buffer transfer, RELOAD bit is 0. When the DMA channel receives a DMA request or trigger (depending on how it is configured), it performs one or more transfers as configured, then stops. Once the Channel Descriptor is exhausted, additional DMA requests or triggers will have no effect until the channel configuration is updated by software.

20.4.2.2 Interleaved transfers

One use for the SRCINC and DSTINC configurations (located in the channel transfer configuration registers, XFERCFGn) is to handle data in a buffer such that it is interleaved with other data.

For example, if 4 data samples from several peripherals need to be interleaved into a single data structure, this may be done while the data is being read in by the DMA. Setting SRCINC to 4x width for each channel involved will allow room for 4 samples in a row in the buffer memory. The DMA will place data for each successive value at the next location for that peripheral.

The reverse of this process could be done using DSTINC to de-interleave combined data from the buffer and send it to several peripherals or locations.

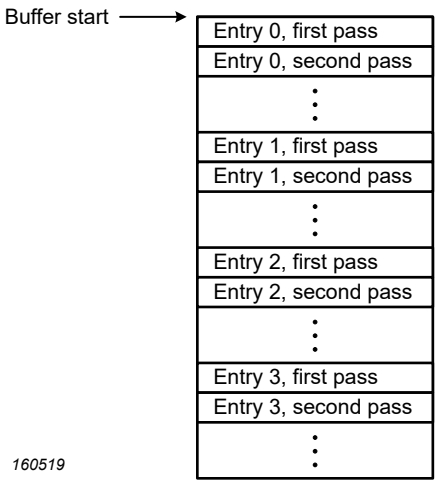


Figure 74. Interleaved transfer in a single buffer

20.4.2.3 Linked Transfers (Linked List)

A linked transfer is transfer across more than one descriptor. A single DMA request or trigger initiates a single transfer, a portion of a transfer, one whole descriptor, or an entire structure of links.

An example of a linked transfer could start out like the example for a ping-pong transfer (Table 103). The difference would be that descriptor B would not link back to descriptor A, but would continue on to another different descriptor. This could continue as long as desired, and can be ended anywhere, or linked back to any point to repeat a sequence of descriptors. Any descriptor not currently in use can be altered by software.

20.4.2.3.1 Ping-Pong

Ping-pong is a special and more frequently used application of linked transfers. A ping-pong transfer uses two buffers alternately. At any one time, one buffer is being loaded or unloaded by DMA operations. The other buffer has the opposite operation being handled by software, readying the buffer for use when the buffer currently being used by the DMA controller is full or empty. The table below shows an example of descriptors for ping-pong from a peripheral to two buffers in memory.

Table 103. Example descriptors for ping-pong operation: peripheral to buffer

Channel Descriptor			Descriptor B			Descriptor A	
+ 0x0	(not used)		+ 0x0	Buffer B transfer configuration		+ 0x0	Buffer A transfer configuration
+ 0x4	Peripheral data end address		+ 0x4	Peripheral data end address		+ 0x4	Peripheral data end address
+ 0x8	Buffer A memory end address		+ 0x8	Buffer B memory end address		+ 0x8	Buffer A memory end address
+ 0xC	Address of descriptor B		+ 0xC	Address of descriptor A		+ 0xC	Address of descriptor B

In addition to the Channel Descriptor, two reload descriptors (A and B) are needed because the Channel Descriptor is actively modified during DMA operation. Descriptor A is typically a copy of the original Channel Descriptor plus the transfer configuration information for buffer A.

The Channel Descriptor is used first, with a first buffer in memory called buffer A. The configuration of the DMA channel must have been set to indicate a reload. Similarly, both descriptor A and descriptor B must also specify reload. When the Channel Descriptor is exhausted, a transfer interrupt informs the CPU that buffer A is available for processing. Descriptor B is then loaded using the link to descriptor B, and a new transfer begins using buffer B.

When Descriptor B is exhausted, a transfer interrupt informs the CPU that buffer B is available for processing. Descriptor A is then loaded using the link to descriptor A. This ping-pong process repeats, alternating between the two memory buffers.

20.4.2.4 Address alignment for data transfers

Transfers of 16 bit width require an address alignment to a multiple of 2 bytes. Transfers of 32 bit width require an address alignment to a multiple of 4 bytes. Transfers of 8 bit width can be at any address.

20.4.2.5 Channel chaining

Channel chaining is a feature which allows completion of a DMA transfer on channel x to trigger a DMA transfer on channel y. For example, this can be used to have DMA channel x reading n bytes from UART to memory, and then have DMA channel y transferring the received bytes to the CRC engine, without any action required from the core.

To use channel chaining, first configure DMA channels x and y as if no channel chaining would be used. Then:

- For channel x:
  - If channel x is configured to auto reload the descriptor on exhausting of the descriptor, RELOAD is set, then enable CLRTRIG in the channel's transfer configuration in the descriptor.
- For channel y:
  - Configure the DMA_ITRIG_SEL for channel y to use any of the available DMA trigger muxes (see INPUTMUX and DMA requests).
  - Configure the chosen DMA trigger mux to select DMA channel x.
  - Enable hardware triggering by setting HWTRIGEN.
  - Set the trigger type to edge sensitive by clearing bit TRIGTYPE in the channel configuration register.
  - Configure the trigger edge to falling edge by clearing bit TRIGPOL in the channel configuration register.

NOTE

After completion of channel x the descriptor may be reloaded (if configured so), but remains un-triggered. To configure the chain to auto-trigger itself, setup channels x and y for channel chaining as described above.

- A ping-pong configuration for both channel x and y is recommended, so that data currently moved by channel y is not altered by channel x.
- For channel x:
  - Configure the DMA_ITRIG_SEL (See INPUTMUX) for channel y to use the same DMA trigger mux as chosen for channel y.
  - Enable hardware triggering by setting HWTRIGEN bit in the channel configuration register.
  - Set the trigger type to edge sensitive by clearing the TRIGTYPE bit in the channel configuration register.
  - Configure the trigger edge to falling edge by clearing the TRIGPOL bit in the channel configuration register.

20.4.3 DMA requests

DMA requests are directly connected to the peripherals. Each channel supports one DMA request and one trigger input which is multiplexed to many possible input sources.

DMA requests are intended to pace transfers to match what the peripheral (including its FIFO if it has one) can do. For example, the USART will issue a transmit DMA request when its transmit FIFO is not full, and a receive DMA request when its receive FIFO is not empty.

20.4.3.1 DMA with I2C monitor mode

The I^2C monitor function may be used with DMA if one of the channels related to the same Flexcomm module is available.

Table 104. DMA with the I2C Monitor function

I2C Master DMA	I2C Slave DMA	I2C Monitor DMA
Not enabled	-	If I2C Monitor DMA is enabled, it will use the DMA channel for the Master function of the same Flexcomm module.
Enabled	Not enabled	If I2C Monitor DMA is enabled, it will use the DMA channel for the Slave function of the same Flexcomm module.
Enabled	Enabled	The I2C Monitor function cannot use DMA.

20.4.4 Hardware triggers

Triggers start the transfer. DMA transfer happens for any particular DMA channel after it is triggered by software or hardware. In typical cases, only a software trigger is used. Other possibilities are occurence of certain timer or pin related events. Those transfers would usually still be paced by a peripheral DMA request if a peripheral is involved in the transfer. Once triggered by software or hardware, a DMA operation on a specific channel is initiated by a DMA request if it is enabled for that channel.

The DMA channel using a trigger can respond by moving data from any memory address to any other memory address. This can include fixed peripheral data registers, or incrementing through RAM buffers. The amount of data moved by a single trigger event can range from a single transfer to many transfers. A transfer that is started by a trigger can still be paced using the channel's DMA request. This allows sending a string to a serial peripheral, for instance, without overrunning the peripheral's transmit buffer.

Each DMA channel also has an output that can be used as a trigger input to another channel. The trigger outputs appear in the trigger source list for each channel and can be selected through the DMA_INMUX registers as inputs to other channels.

Each DMA channel can use one trigger that is independent of the request input for the channel. The trigger input is selected in the DMA_ITRIG_SEL registers (See INPUTMUX). In addition, the DMA trigger output can be routed to the trigger input of another channel through the trigger input multiplexing. See DMA requests and INPUTMUX.

20.4.4.1 Trigger operation detail

A trigger is always needed to start a transfer on a DMA channel. The trigger can be a hardware or software trigger, and can be used in several ways.

If a channel is configured with the SWTRIG bit equal to 0, the channel can be later triggered by hardware or software. Software triggering is accomplished by writing a 1 to the appropriate SETTRIG0 bit. Hardware triggering requires setup of the HWTRIGEN, TRIGPOL, TRIGTYPE, and TRIGBURST fields in the Configuration register for DMA channel (CFG0 - CFG51) for the related channel. When a channel is initially set up, the SWTRIG bit can be set, causing the transfer to begin immediately.

Once triggered, transfer on a channel will be paced by DMA requests if the PERIPHREQEN bit in the related CFG register is set. Otherwise, the transfer will proceed at full speed.

The TRIG bit can be cleared at the end of a transfer, determined by the value CLRTRIG. When CLRTRIG = 1 , the trigger is cleared when the descriptor is exhausted.

20.4.4.2 Trigger output detail

Each DMA channel provides a trigger output. This allows the possibility of using trigger outputs as a trigger source to a different channel in order to support complex transfers on selected peripherals. This kind of transfer can, use more than one peripheral DMA request. An example use would be to input data to a holding buffer from one peripheral, and then output the data to another peripheral, with both transfers being paced by the appropriate peripheral DMA request. This kind of operation is called "chained operation" or "channel chaining".

20.5 Application information

The two intended scenarios for the use of DMA controllers are:

- One DMA controller (DMA0) is used by the CM33, the other (DMA1) is used by the DSP. This scenario can apply to systems where there is no need to differentiate security between the CPUs or between different tasks.
- One DMA controller (DMA0) is secured and has access to secure spaces and peripherals, the other (DMA1) is not secured and does not have access to secure spaces and peripherals. In this scenario, only the secure code running on the CM33 has access to the secure DMA controller. The non-secure DMA controller will be shared by other code and by the DSP (if needed).

20.6 Memory map and register definition

This section includes the DMA Controller memory map and detailed descriptions of all registers for one DMA controller. DMA transfers are controlled by a set of three registers per channel, the CFG, CTRLSTAT, and XFERCFG registers.

20.6.1 DMA0 controller register descriptions

20.6.1.1 DMA memory map

DMA0 base address: 4008_2000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	DMA control (CTRL)	32	See section	See section
4	Interrupt status (INTSTAT)	32	See section	See section
8	SRAM address of the channel configuration table (SRAMBASE)	32	See section	See section
20	Channel Enable read and set for all DMA channels (ENABLESET0)	32	RW	0000_0000
24	Channel Enable read and set for all DMA channels (ENABLESET1)	32	RW	0000_0000
28	Channel Enable Clear for all DMA channels (ENABLECLR0)	32	See section	See section
2C	Channel Enable Clear for all DMA channels (ENABLECLR1)	32	See section	See section
30	Channel Active status for all DMA channels (ACTIVE0)	32	RO	0000_0000
34	Channel Active status for all DMA channels (ACTIVE1)	32	See section	0000_0000
38	Channel Busy status for all DMA channels (BUSY0)	32	RO	0000_0000
3C	Channel Busy status for all DMA channels (BUSY1)	32	See section	0000_0000
40	Error Interrupt status for all DMA channels (ERRINT0)	32	RW	0000_0000
44	Error Interrupt status for all DMA channels (ERRINT1)	32	See section	0000_0000
48	Interrupt Enable read and Set for all DMA channels (INTENSET0)	32	RW	0000_0000
4C	Interrupt Enable read and Set for all DMA channels (INTENSET1)	32	See section	0000_0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
50	Interrupt Enable Clear for all DMA channels (INTENCLR0)	32	WO	See section
54	Interrupt Enable Clear for all DMA channels (INTENCLR1)	32	See section	See section
58	Interrupt A status for all DMA channels (INTA0)	32	RW	0000_0000
5C	Interrupt A status for all DMA channels (INTA1)	32	See section	0000_0000
60	Interrupt B status for all DMA channels (INTB0)	32	RW	0000_0000
64	Interrupt B status for all DMA channels (INTB1)	32	See section	0000_0000
68	Set ValidPending control bits for all DMA channels (SETVALID0)	32	WO	See section
6C	Set ValidPending control bits for all DMA channels (SETVALID1)	32	See section	See section
70	Set Trigger control bits for all DMA channels (SETTRIG0)	32	WO	See section
74	Set Trigger control bits for all DMA channels (SETTRIG1)	32	See section	See section
78	Channel Abort control for all DMA channels (ABORT0)	32	WO	See section
7C	Channel Abort control for all DMA channels (ABORT1)	32	See section	See section
400 - 730	Configuration register for DMA channel (CFG0 - CFG51)	32	See section	See section
404 - 734	Control and status register for DMA channel (CTLSTAT0 - CTLSTAT51)	32	RO	See section
408 - 738	Transfer configuration register for DMA channel (XFERCFG0 - XFERCFG51)	32	See section	See section

20.6.1.1.1 DMA control (CTRL)

This register has the global control bit for a enabling the DMA controller.

Offset

Register	Offset
CTRL	0h

Diagram

Fields

Field	Description
31-1 —	Reserved
0 ENABLE	DMA controller master enable. 0 - DMA controller is disabled. This clears any triggers that were asserted at the point when disabled, but does not prevent re-triggering when the DMA controller is re-enabled. 1 - Enabled. DMA controller is enabled.

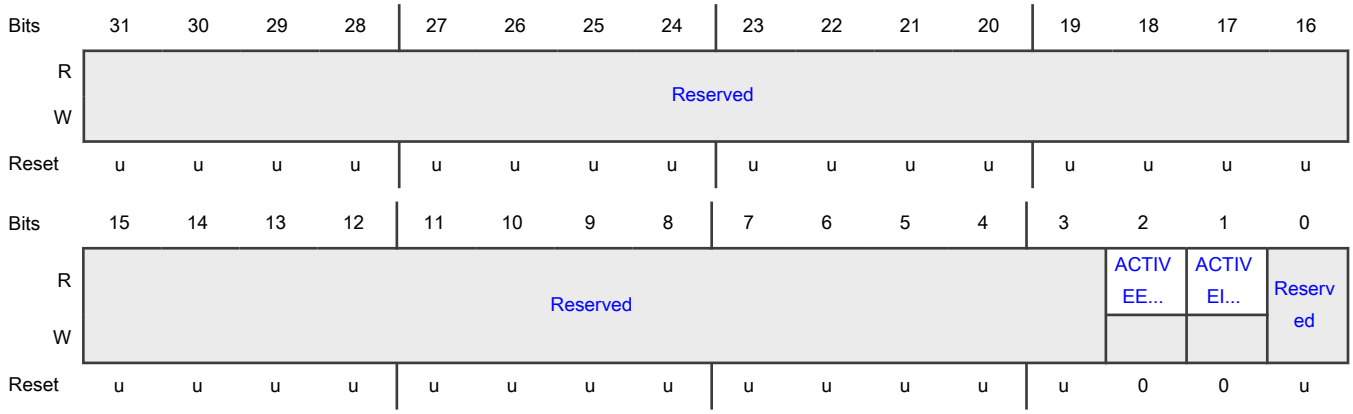
20.6.1.1.2 Interrupt status (INTSTAT)

It provides an overview of DMA status and allows quick determination of whether any enabled interrupts are pending. For details regarding the channels involved see Interrupt A status for all DMA channels (INTA0) and Interrupt B status for all DMA channels (INTB0)

Offset

Register	Offset
INTSTAT	4h

Diagram



Fields

Field	Description
31-3 —	Reserved
2 ACTIVEERRINT	Summarizes whether any error interrupts are pending. 0 - No error interrupts are pending. 1 - At least one error interrupt is pending.
1 ACTIVEINT	Summarizes whether any enabled interrupts (other than error interrupts) are pending. 0 - No enabled interrupts are pending. 1 - At least one enabled interrupt is pending.
0 —	Reserved

20.6.1.1.3 SRAM address of the channel configuration table (SRAMBASE)

It must be configured with an address (preferably in on-chip SRAM) where DMA descriptors will be stored. Software must set up the descriptors for the DMA channels that will be used in the application.

Each DMA channel has an entry for the Channel Descriptor in the SRAM table. See Chip-specific DMA information section.

Offset

Register	Offset
SRAMBASE	8h

Diagram

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

Fields

Field	Description
31-10 OFFSET	Offset Address of the beginning of the DMA descriptor table. The table must begin on a 1KB boundary.
9-0 —	Reserved

20.6.1.1.4 Channel Enable read and set for all DMA channels (ENABLESET0)

The ENABLESET0 register determines whether each DMA channel is enabled or disabled. Disabling a DMA channel does not reset the channel in any way. A channel can be paused and restarted by clearing, then setting the Enable bit for that channel. Reading ENABLESET0 provides the current state of all of the DMA channels represented by that register. Writing a 1 to a bit position in ENABLESET0 that corresponds to an implemented DMA channel sets the bit, enabling the related DMA channel. Writing a 0 to any bit has no effect. Enables are cleared by writing to ENABLECLR0.

Offset

Register	Offset
ENABLESET0	20h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ENABLE31	ENABLE30	ENABLE29	ENABLE28	ENABLE27	ENABLE26	ENABLE25	ENABLE24	ENABLE23	ENABLE22	ENABLE21	ENABLE20	ENABLE19	ENABLE18	ENABLE17	ENABLE16
Reset	0	0	0	0	0	0	0	0	0	0	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	ENABLE15	ENABLE14	ENABLE13	ENABLE12	ENABLE11	ENABLE10	ENABLE9	ENABLE8	ENABLE7	ENABLE6	ENABLE5	ENABLE4	ENABLE3	ENABLE2	ENABLE1	ENABLE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 ENABLEn	Enable for DMA channel 0 - DMA channel is disabled. 1 - DMA channel is enabled.

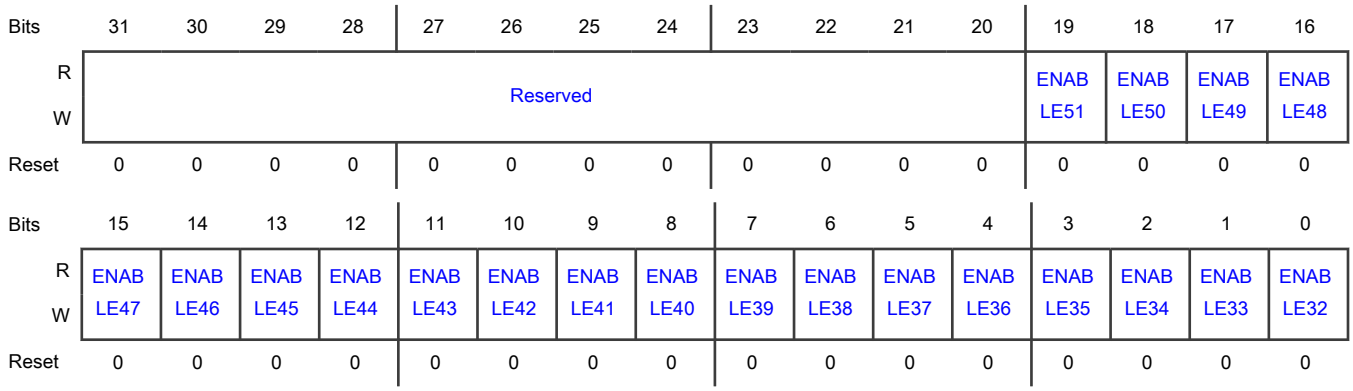
20.6.1.1.5 Channel Enable read and set for all DMA channels (ENABLESET1)

The ENABLESET1 register determines whether each DMA channel is enabled or disabled. Disabling a DMA channel does not reset the channel in any way. A channel can be paused and restarted by clearing, then setting the Enable bit for that channel. Reading ENABLESET1 provides the current state of all of the DMA channels represented by that register. Writing a 1 to a bit position in ENABLESET1 that corresponds to an implemented DMA channel sets the bit, enabling the related DMA channel. Writing a 0 to any bit has no effect. Enables are cleared by writing to ENABLECLR1.

Offset

Register	Offset
ENABLESET1	24h

Diagram



Fields

Field	Description
31-20 —	Reserved
19-0 ENABLEn	Enable for DMA channel 0 - DMA channel is disabled. 1 - DMA channel is enabled.

20.6.1.1.6 Channel Enable Clear for all DMA channels (ENABLECLR0)

The ENABLECLR0 register is used to clear the enable of one or more DMA channels.

Offset

Register	Offset
ENABLECLR0	28h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CLR31	CLR30	CLR29	CLR28	CLR27	CLR26	CLR25	CLR24	CLR23	CLR22	CLR21	CLR20	CLR19	CLR18	CLR17	CLR16
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
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	CLR15	CLR14	CLR13	CLR12	CLR11	CLR10	CLR9	CLR8	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-0 CLRn	Writing ones to this register clears the corresponding bits in ENABLESET0. 0 - No effect. 1 - DMA channel is cleared.

20.6.1.1.7 Channel Enable Clear for all DMA channels (ENABLECLR1)

The ENABLECLR1 register is used to clear the enable of one or more DMA channels.

Offset

Register	Offset
ENABLECLR1	2Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved												CLR51	CLR50	CLR49	CLR48
W													W1C	W1C	W1C	W1C
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	CLR47	CLR46	CLR45	CLR44	CLR43	CLR42	CLR41	CLR40	CLR39	CLR38	CLR37	CLR36	CLR35	CLR34	CLR33	CLR32
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-20 —	Reserved
19-0 CLRn	Writing ones to this register clears the corresponding bits in ENABLESET1. 0 - No effect. 1 - DMA channel is cleared.

20.6.1.1.8 Channel Active status for all DMA channels (ACTIVE0)

It indicates which DMA channels are active at the point when the read occurs. A DMA channel is considered active when a DMA operation has been started but not yet fully completed. The active status persists another DMA operation from being started, until the pipeline is empty. After end of the last descriptor (when there is no reload), a new DMA operation can be started. An active channel may be aborted by software by setting the appropriate bit in the ABORT register.

Offset

Register	Offset
ACTIVE0	30h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ACTIVE31	ACTIVE30	ACTIVE29	ACTIVE28	ACTIVE27	ACTIVE26	ACTIVE25	ACTIVE24	ACTIVE23	ACTIVE22	ACTIVE21	ACTIVE20	ACTIVE19	ACTIVE18	ACTIVE17	ACTIVE16
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	ACTIVE15	ACTIVE14	ACTIVE13	ACTIVE12	ACTIVE11	ACTIVE10	ACTIVE9	ACTIVE8	ACTIVE7	ACTIVE6	ACTIVE5	ACTIVE4	ACTIVE3	ACTIVE2	ACTIVE1	ACTIVE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 ACTIVEn	Active flag for DMA channel. 0 - DMA channel is not active. 1 - DMA channel is active.

20.6.1.1.9 Channel Active status for all DMA channels (ACTIVE1)

It indicates which DMA channels are active at the point when the read occurs. A DMA channel is considered active when a DMA operation has been started but not yet fully completed. The active status persists another DMA operation from being started, until the pipeline is empty. After end of the last descriptor (when there is no reload), a new DMA operation can be started. An active channel may be aborted by software by setting the appropriate bit in the ABORT register.

Offset

Register	Offset
ACTIVE1	34h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved												ACTIVE51	ACTIVE50	ACTIVE49	ACTIVE48
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	ACTIVE47	ACTIVE46	ACTIVE45	ACTIVE44	ACTIVE43	ACTIVE42	ACTIVE41	ACTIVE40	ACTIVE39	ACTIVE38	ACTIVE37	ACTIVE36	ACTIVE35	ACTIVE34	ACTIVE33	ACTIVE32
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-20 —	Reserved
19-0 ACTIVEn	Active flag for DMA channel. 0 - DMA channel is not active. 1 - DMA channel is active.

20.6.1.1.10 Channel Busy status for all DMA channels (BUSY0)

The BUSY0 register indicates which DMA channels is busy at the point when the read occurs. A DMA channel is considered busy when there is any operation related to that channel in the DMA controller's internal pipeline. This information can be used after a DMA channel is disabled by software (but still active), allowing confirmation that there are no remaining operations in progress for that channel.

Offset

Register	Offset
BUSY0	38h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BUSY 31	BUSY 30	BUSY 29	BUSY 28	BUSY 27	BUSY 26	BUSY 25	BUSY 24	BUSY 23	BUSY 22	BUSY 21	BUSY 20	BUSY 19	BUSY 18	BUSY 17	BUSY 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	BUSY 15	BUSY 14	BUSY 13	BUSY 12	BUSY 11	BUSY 10	BUSY 9	BUSY 8	BUSY 7	BUSY 6	BUSY 5	BUSY 4	BUSY 3	BUSY 2	BUSY 1	BUSY 0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 BUSYn	Busy flag for DMA channel. 0 - DMA channel is not busy. 1 - DMA channel is busy.

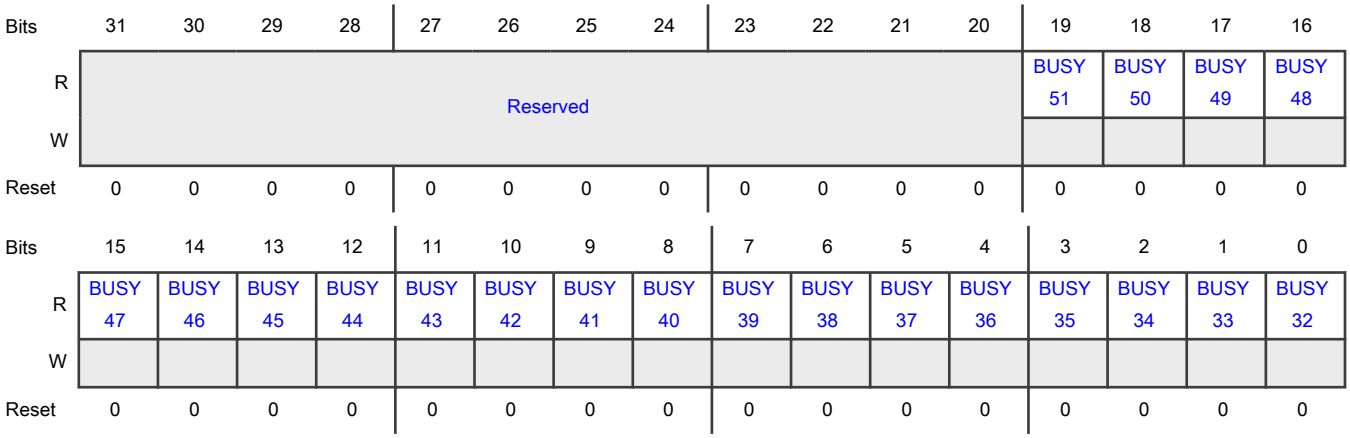
20.6.1.1.11 Channel Busy status for all DMA channels (BUSY1)

The BUSY0 register indicates which DMA channels is busy at the point when the read occurs. A DMA channel is considered busy when there is any operation related to that channel in the DMA controller's internal pipeline. This information can be used after a DMA channel is disabled by software (but still active), allowing confirmation that there are no remaining operations in progress for that channel.

Offset

Register	Offset
BUSY1	3Ch

Diagram



Fields

Field	Description
31-20 —	Reserved
19-0 BUSYn	Busy flag for DMA channel. 0 - DMA channel is not busy. 1 - DMA channel is busy.

20.6.1.1.12 Error Interrupt status for all DMA channels (ERRINT0)

It contains flags for each DMA channel's Error Interrupt. Any pending interrupt flag in the register will be reflected on the DMA interrupt output. Reading the registers provides the current state of all DMA channel error interrupts. Writing a 1 to a bit position in ERRINT0 that corresponds to an implemented DMA channel clears the bit, removing the interrupt for the related DMA channel. Writing a 0 to any bit has no effect.

Offset

Register	Offset
ERRINT0	40h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ERR31	ERR30	ERR29	ERR28	ERR27	ERR26	ERR25	ERR24	ERR23	ERR22	ERR21	ERR20	ERR19	ERR18	ERR17	ERR16
Reset	0	0	0	0	0	0	0	0	0	0	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	ERR15	ERR14	ERR13	ERR12	ERR11	ERR10	ERR9	ERR8	ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 ERRn	Error Interrupt flag for DMA channel. 0 - The Error Interrupt is not active for DMA channel. 1 - The Error Interrupt is pending for DMA channel.

20.6.1.1.13 Error Interrupt status for all DMA channels (ERRINT1)

It contains flags for each DMA channel's Error Interrupt. Any pending interrupt flag in the register will be reflected on the DMA interrupt output. Reading the registers provides the current state of all DMA channel error interrupts. Writing a 1 to a bit position in ERRINT0 that corresponds to an implemented DMA channel clears the bit, removing the interrupt for the related DMA channel. Writing a 0 to any bit has no effect.

Offset

Register	Offset
ERRINT1	44h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved												ERR51	ERR50	ERR49	ERR48
Reset	0	0	0	0	0	0	0	0	0	0	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	ERR47	ERR46	ERR45	ERR44	ERR43	ERR42	ERR41	ERR40	ERR39	ERR38	ERR37	ERR36	ERR35	ERR34	ERR33	ERR32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-20 —	Reserved
19-0 ERRn	Error Interrupt flag for DMA channel. 0 - The Error Interrupt is not active for DMA channel. 1 - The Error Interrupt is pending for DMA channel.

20.6.1.1.14 Interrupt Enable read and Set for all DMA channels (INTENSET0)

It controls whether the individual Interrupts for DMA channels contribute to the DMA interrupt output. Reading the registers provides the current state of all DMA channel interrupt enables. Writing a 1 to a bit position in INTENSET0 that corresponds to an implemented DMA channel sets the bit, enabling the interrupt for the related DMA channel. Writing a 0 to any bit has no effect. Interrupt enables are cleared by writing to INTENCLR0.

Offset

Register	Offset
INTENSET0	48h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INTEN 31	INTEN 30	INTEN 29	INTEN 28	INTEN 27	INTEN 26	INTEN 25	INTEN 24	INTEN 23	INTEN 22	INTEN 21	INTEN 20	INTEN 19	INTEN 18	INTEN 17	INTEN 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 W	INTEN 15	INTEN 14	INTEN 13	INTEN 12	INTEN 11	INTEN 10	INTEN 9	INTEN 8	INTEN 7	INTEN 6	INTEN 5	INTEN 4	INTEN 3	INTEN 2	INTEN 1	INTEN 0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 INTENn	Interrupt Enable read and set for DMA channel. 0 - The Interrupt for DMA channel is disabled. 1 - The Interrupt for DMA channel is enabled.

20.6.1.1.15 Interrupt Enable read and Set for all DMA channels (INTENSET1)

It controls whether the individual Interrupts for DMA channels contribute to the DMA interrupt output. Reading the registers provides the current state of all DMA channel interrupt enables. Writing a 1 to a bit position in INTENSET0 that corresponds to an implemented DMA channel sets the bit, enabling the interrupt for the related DMA channel. Writing a 0 to any bit has no effect. Interrupt enables are cleared by writing to INTENCLR0.

Offset

Register	Offset
INTENSET1	4Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved												INTEN 51	INTEN 50	INTEN 49	INTEN 48
Reset	0	0	0	0	0	0	0	0	0	0	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	INTEN 47	INTEN 46	INTEN 45	INTEN 44	INTEN 43	INTEN 42	INTEN 41	INTEN 40	INTEN 39	INTEN 38	INTEN 37	INTEN 36	INTEN 35	INTEN 34	INTEN 33	INTEN 32
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-20 —	Reserved
19-0 INTENn	Interrupt Enable read and set for DMA channel. 0 - The Interrupt for DMA channel is disabled. 1 - The Interrupt for DMA channel is enabled.

20.6.1.1.16 Interrupt Enable Clear for all DMA channels (INTENCLR0)

This register is used to clear interrupt enable bits in INTENSET0. The register is write-only.

Offset

Register	Offset
INTENCLR0	50h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	CLR31	CLR30	CLR29	CLR28	CLR27	CLR26	CLR25	CLR24	CLR23	CLR22	CLR21	CLR20	CLR19	CLR18	CLR17	CLR16
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																
W	CLR15	CLR14	CLR13	CLR12	CLR11	CLR10	CLR9	CLR8	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-0 CLRn	Writing ones to this register clears corresponding bits in the DMAIntEnSet0.

20.6.1.1.17 Interrupt Enable Clear for all DMA channels (INTENCLR1)

This register is used to clear interrupt enable bits in INTENSET1. The register is write-only.

Offset

Register	Offset
INTENCLR1	54h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W													CLR51	CLR50	CLR49	CLR48
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																
W	CLR47	CLR46	CLR45	CLR44	CLR43	CLR42	CLR41	CLR40	CLR39	CLR38	CLR37	CLR36	CLR35	CLR34	CLR33	CLR32
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-20 —	Reserved
19-0 CLRn	Writing ones to this register clears corresponding bits in the DMAIntEnSet1.

20.6.1.1.18 Interrupt A status for all DMA channels (INTA0)

It contains the interrupt A status for each DMA channel. The status will be set when the SETINTA bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in this register clears the related INTA flag. Writing 0 has no effect. Any interrupt pending status in the registers will be reflected on the DMA interrupt output if it is enabled in the related INTENSET register.

Offset

Register	Offset
INTA0	58h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INTA31	INTA30	INTA29	INTA28	INTA27	INTA26	INTA25	INTA24	INTA23	INTA22	INTA21	INTA20	INTA19	INTA18	INTA17	INTA16
Reset	0	0	0	0	0	0	0	0	0	0	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	INTA15	INTA14	INTA13	INTA12	INTA11	INTA10	INTA9	INTA8	INTA7	INTA6	INTA5	INTA4	INTA3	INTA2	INTA1	INTA0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 INTAn	Interrupt A status for DMA channel. 0 - The DMA channel interrupt A is not active. 1 - The DMA channel interrupt A is active.

20.6.1.1.19 Interrupt A status for all DMA channels (INTA1)

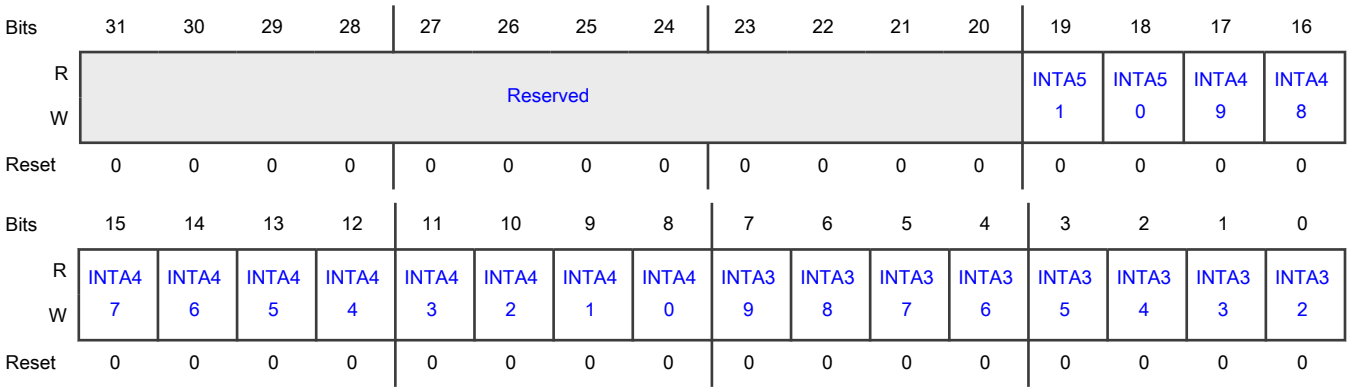
It contains the interrupt A status for each DMA channel. The status will be set when the SETINTA bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in this register clears the related INTA

flag. Writing 0 has no effect. Any interrupt pending status in the registers will be reflected on the DMA interrupt output if it is enabled in the related INTENSET register.

Offset

Register	Offset
INTA1	5Ch

Diagram



Fields

Field	Description
31-20 —	Reserved
19-0 INTAn	Interrupt A status for DMA channel. 0 - The DMA channel interrupt A is not active. 1 - The DMA channel interrupt A is active.

20.6.1.1.20 Interrupt B status for all DMA channels (INTB0)

It contains the interrupt B status for each DMA channel. The status will be set when the SETINTB bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in the register clears the related INTB flag. Writing 0 has no effect. Any interrupt pending status in this register will be reflected on the DMA interrupt output if it is enabled in the INTENSET register.

Offset

Register	Offset
INTB0	60h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INTB31	INTB30	INTB29	INTB28	INTB27	INTB26	INTB25	INTB24	INTB23	INTB22	INTB21	INTB20	INTB19	INTB18	INTB17	INTB16
Reset	0	0	0	0	0	0	0	0	0	0	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	INTB15	INTB14	INTB13	INTB12	INTB11	INTB10	INTB9	INTB8	INTB7	INTB6	INTB5	INTB4	INTB3	INTB2	INTB1	INTB0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 INTBn	Interrupt B status for DMA channel. 0 - The DMA channel interrupt B is not active. 1 - The DMA channel interrupt B is active.

20.6.1.1.21 Interrupt B status for all DMA channels (INTB1)

It contains the interrupt B status for each DMA channel. The status will be set when the SETINTB bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in the register clears the related INTB flag. Writing 0 has no effect. Any interrupt pending status in this register will be reflected on the DMA interrupt output if it is enabled in the INTENSET register.

Offset

Register	Offset
INTB1	64h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved												INTB19	INTB18	INTB17	INTB16
Reset	0	0	0	0	0	0	0	0	0	0	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	INTB15	INTB14	INTB13	INTB12	INTB11	INTB10	INTB9	INTB8	INTB7	INTB6	INTB5	INTB4	INTB3	INTB2	INTB1	INTB0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-20 —	Reserved
19-0 INTBn	Interrupt B status for DMA channel. 0 - The DMA channel interrupt B is not active. 1 - The DMA channel interrupt B is active.

20.6.1.1.22 Set ValidPending control bits for all DMA channels (SETVALID0)

The SETVALID0 register allows setting the Valid bit in the CTRLSTAT register for one or more DMA channels. The CFGVALID and SV (set valid) bits allow more direct DMA block timing control by software. Each Channel Descriptor, in a sequence of descriptors, can be validated by either the setting of the CFGVALID bit or by setting the channel's SETVALID flag. Normally, the CFGVALID bit is set. This tells the DMA that the Channel Descriptor is active and can be executed. The DMA will continue sequencing through descriptor blocks whose CFGVALID bit are set without further software intervention. Leaving a CFGVALID bit set to 0 allows the DMA sequence to pause at the Descriptor until software triggers the continuation. If, during DMA transmission, a Channel Descriptor is found with CFGVALID set to 0, the DMA checks for a previously buffered SETVALID0 setting for the channel. If found, the DMA will set the descriptor valid, clear the SV setting, and resume processing the descriptor. Otherwise, the DMA pauses until the channels SETVALID0 bit is set.

Offset

Register	Offset
SETVALID0	68h

Diagram

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

Fields

Field	Description
31-0	SetValid control for DMA channel.

Table continues on the next page...

Field	Description
SETVALIDn	0 - No effect. 1 - Sets the ValidPending control bit for DMA channel.

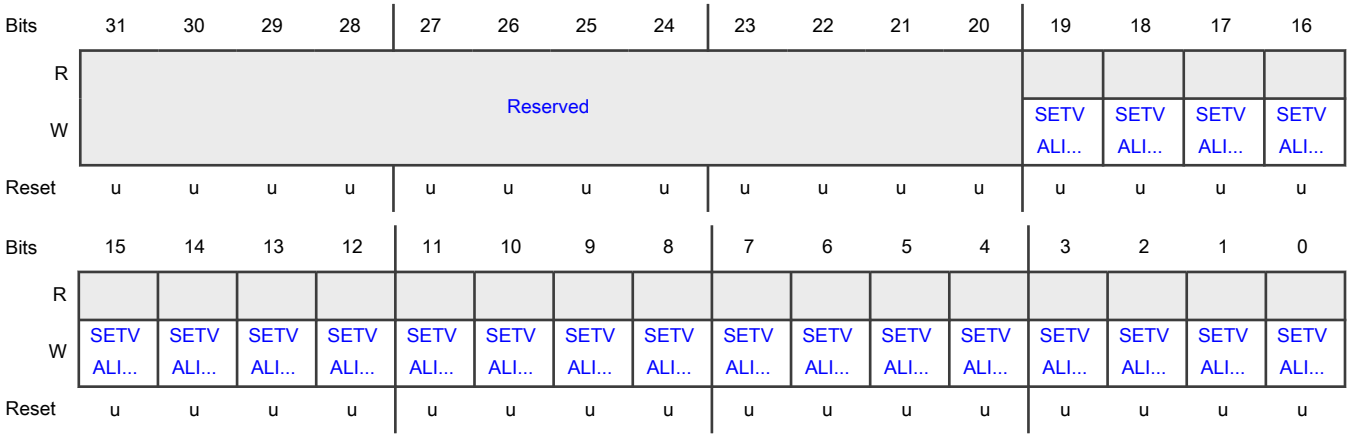
20.6.1.1.23 Set ValidPending control bits for all DMA channels (SETVALID1)

The SETVALID1 register allows setting the Valid bit in the CTRLSTAT register for one or more DMA channels. The CFGVALID and SV (set valid) bits allow more direct DMA block timing control by software. Each Channel Descriptor, in a sequence of descriptors, can be validated by either the setting of the CFGVALID bit or by setting the channel's SETVALID flag. Normally, the CFGVALID bit is set. This tells the DMA that the Channel Descriptor is active and can be executed. The DMA will continue sequencing through descriptor blocks whose CFGVALID bit are set without further software intervention. Leaving a CFGVALID bit set to 0 allows the DMA sequence to pause at the Descriptor until software triggers the continuation. If, during DMA transmission, a Channel Descriptor is found with CFGVALID set to 0, the DMA checks for a previously buffered SETVALID1 setting for the channel. If found, the DMA will set the descriptor valid, clear the SV setting, and resume processing the descriptor. Otherwise, the DMA pauses until the channels SETVALID1 bit is set.

Offset

Register	Offset
SETVALID1	6Ch

Diagram



Fields

Field	Description
31-20 —	Reserved
19-0 SETVALIDn	SetValid control for DMA channel. 0 - No effect. 1 - Sets the ValidPending control bit for DMA channel.

20.6.1.1.24 Set Trigger control bits for all DMA channels (SETTRIG0)

This register allows setting the TRIG bit in the CTRLSTAT register for one or more DMA channel.

Offset

Register	Offset
SETTRIG0	70h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...
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																
W	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG9	SETTRIG8	SETTRIG7	SETTRIG6	SETTRIG5	SETTRIG4	SETTRIG3	SETTRIG2	SETTRIG1	SETTRIG0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-0 SETTRIGn	Set Trigger control bit for DMA channel. 0 - No effect. 1 - Sets the Trig bit for DMA channel.

20.6.1.1.25 Set Trigger control bits for all DMA channels (SETTRIG1)

register allows setting the TRIG bit in the CTRLSTAT register for one or more DMA channel.

Offset

Register	Offset
SETTRIG1	74h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W													SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...
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																
W	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-20 —	Reserved
19-0 SETTRIGn	Set Trigger control bit for DMA channel. 0 - No effect. 1 - Sets the Trig bit for DMA channel.

20.6.1.1.26 Channel Abort control for all DMA channels (ABORT0)

The Abort 0 register allows aborting operation of a DMA channel if needed. To abort a selected channel, the channel should first be disabled by clearing the corresponding Enable bit by writing a 1 to the proper bit in ENABLECLR. Then wait until the channel is no longer busy by checking the corresponding bit in BUSY. Finally, write a 1 to the proper bit of ABORT. This prevents the channel from restarting an incomplete operation when it is enabled again.

Offset

Register	Offset
ABORT0	78h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	ABORT31	ABORT30	ABORT29	ABORT28	ABORT27	ABORT26	ABORT25	ABORT24	ABORT23	ABORT22	ABORT21	ABORT20	ABORT19	ABORT18	ABORT17	ABORT16
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																
W	ABORT15	ABORT14	ABORT13	ABORT12	ABORT11	ABORT10	ABORT9	ABORT8	ABORT7	ABORT6	ABORT5	ABORT4	ABORT3	ABORT2	ABORT1	ABORT0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-0 ABORTn	Abort control for DMA channel. 0 - No effect. 1 - Aborts DMA operations on channel.

20.6.1.1.27 Channel Abort control for all DMA channels (ABORT1)

The Abort 1 register allows aborting operation of a DMA channel if needed. To abort a selected channel, the channel should first be disabled by clearing the corresponding Enable bit by writing a 1 to the proper bit in ENABLECLR. Then wait until the channel is no longer busy by checking the corresponding bit in BUSY. Finally, write a 1 to the proper bit of ABORT. This prevents the channel from restarting an incomplete operation when it is enabled again.

Offset

Register	Offset
ABORT1	7Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W													ABORT51	ABORT50	ABORT49	ABORT48
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																
W	ABORT47	ABORT46	ABORT45	ABORT44	ABORT43	ABORT42	ABORT41	ABORT40	ABORT39	ABORT38	ABORT37	ABORT36	ABORT35	ABORT34	ABORT33	ABORT32
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-20 —	Reserved
19-0 ABORTn	Abort control for DMA channel. 0 - No effect. 1 - Aborts DMA operations on channel.

20.6.1.1.28 Configuration register for DMA channel (CFG0 - CFG51)

The CFG register contains various configuration options for DMA channels.

Table 105. Trigger setting summary

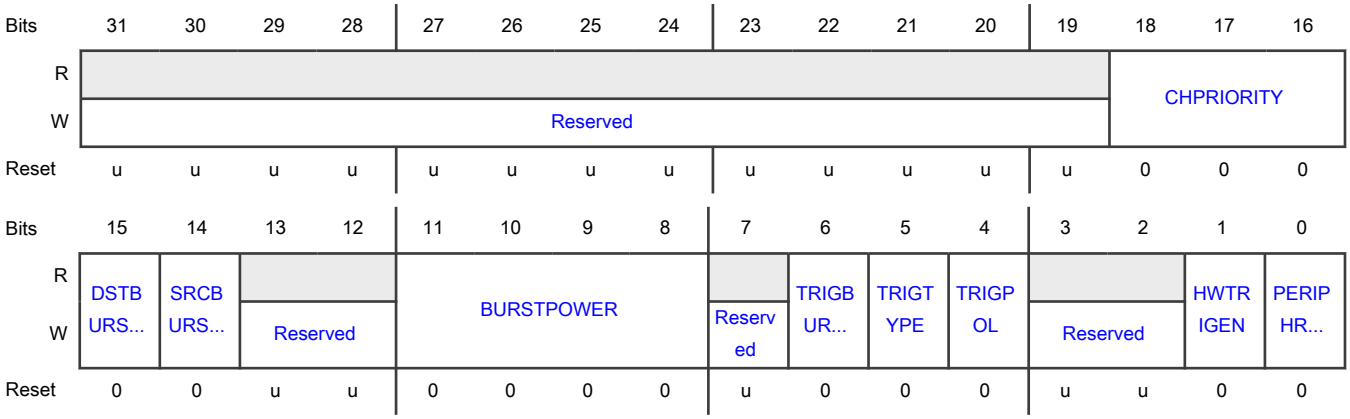
TrigBurst	TrigType	TrigPol	Description
0	0	0	Hardware DMA trigger is falling edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	0	1	Hardware DMA trigger is rising edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	1	0	Hardware DMA trigger is low level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	1	1	Hardware DMA trigger is high level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
1	0	0	Hardware DMA trigger is falling edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	0	1	Hardware DMA trigger is rising edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	1	0	Hardware DMA trigger is low level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	1	1	Hardware DMA trigger is high level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.

Offset

For a = 0 to 51:

Register	Offset
CFGa	400h + (a × 10h)

Diagram



Fields

Field	Description
31-19 —	Reserved Read value is undefined, only zero should be written.
18-16 CHPRIORITY	Priority of channel when multiple DMA requests are pending. Eight priority levels are supported: 0x0 = highest priority. 0x7 = lowest priority.
15 DSTBURSTWRAP	Destination Burst Wrap. When enabled, the destination data address for the DMA is 'wrapped', meaning that the destination address range for each burst will be the same. As an example, this could be used to write several sequential registers to a peripheral for each DMA burst, writing the same registers again for each burst. 0 - Disabled. Destination burst wrapping is not enabled for this DMA channel. 1 - Enabled. Destination burst wrapping is enabled for this DMA channel.
14 SRCBURSTWRAP	Source Burst Wrap. When enabled, the source data address for the DMA is 'wrapped', meaning that the source address range for each burst will be the same. As an example, this could be used to read several sequential registers from a peripheral for each DMA burst, reading the same registers again for each burst. 0 - Disabled. Source burst wrapping is not enabled DMA channel. 1 - Enabled. Source burst wrapping is enabled DMA channel.
13-12	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
11-8 BURSTPOWER	Burst Power. It always selects the address wrap size when SRCBURSTWRAP and/or DSTBURSTWRAP modes are selected. When the TRIGBURST field = 1, Burst Power selects how many transfers are performed for each DMA trigger. This can be used, for example, with peripherals that contain a FIFO that can initiate a DMA operation when the FIFO reaches a certain level. 0000: Burst size = 1 (2^0). This corresponds to a single transfer. 0001: Burst size = 2 (2^1). 0010: Burst size = 4 (2^2). 1010: Burst size = 1024 (2^{10}). This corresponds to the maximum supported transfer count. Others: not supported. The total transfer length as defined in the XFERCOUNT bits in the XFERCFG register must be an integer multiple of the burst size. **NOTE** Total number of bytes transferred is: (XFERCOUNT + 1) x data width (Transfer configuration register for DMA channel (XFERCFG0 - XFERCFG51)).
7 —	Reserved Read value is undefined, only zero should be written.
6 TRIGBURST	Trigger Burst. Selects whether hardware triggers cause a single or burst transfer. 0 - Single transfer. Hardware trigger causes a single transfer. 1 - Burst transfer. When the trigger for this channel is set to edge triggered, a hardware trigger causes a burst transfer, as defined by BURSTPOWER. When the trigger for this channel is set to level triggered, a hardware trigger causes transfers to continue as long as the trigger is asserted, unless the transfer is complete.
5 TRIGTYPE	Trigger Type. Selects hardware trigger as edge triggered or level triggered. 0 - Edge. Hardware trigger is edge triggered. Transfers will be initiated and completed, as specified for a single trigger. 1 - Level. Hardware trigger is level triggered. Note that when level triggering without burst (BURSTPOWER = 0) is selected, only hardware triggers should be used on that channel. Transfers continue as long as the trigger level is asserted. Once the trigger is de-asserted, the transfer will be paused until the trigger is, again, asserted. However, the transfer will not be paused until any remaining transfers within the current BURSTPOWER length are completed.
4	Trigger Polarity.

Table continues on the next page...

Table continued from the previous page...

Field	Description
TRIGPOL	Selects the polarity of a hardware trigger for this channel. 0 - Active low - falling edge. Hardware trigger is active low or falling edge triggered, based on TRIGTYPE. 1 - Active high - rising edge. Hardware trigger is active high or rising edge triggered, based on TRIGTYPE.
3-2 —	Read value is undefined, only zero should be written.
1 HWTRIGEN	Hardware Triggering Enable for channel. 0 - Hardware triggering not used for channel. 1 - Hardware triggering used for channel.
0 PERIPHREQEN	Peripheral request Enable. If a DMA channel is used to perform a memory-to-memory transfer, any peripheral DMA request associated with that channel can be disabled to prevent any interaction between the peripheral and the DMA controller. 0 - Peripheral DMA requests disabled. 1 - Peripheral DMA requests enabled.

20.6.1.1.29 Control and status register for DMA channel (CTLSTAT0 - CTLSTAT51)

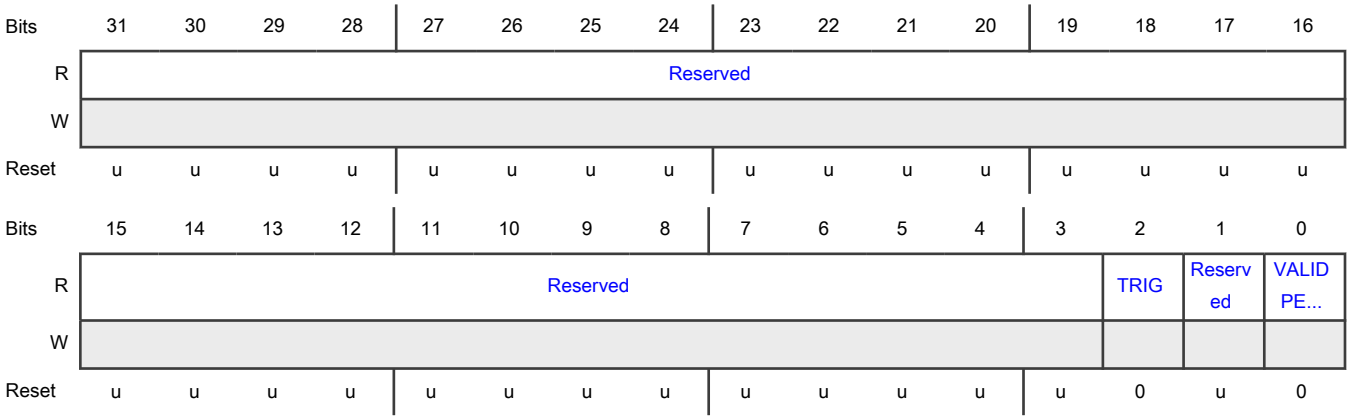
It provides status flags specific to the DMA channel.

Offset

For a = 0 to 51:

Register	Offset
CTLSTATa	404h + (a × 10h)

Diagram



Fields

Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2 TRIG	Trigger flag. Indicates that the trigger for this channel is currently set. This bit is cleared at the end of an entire transfer or upon reload when CLRTRIG = 1. 0 - Not triggered. The trigger for this DMA channel is not set. DMA operations will not be carried out. 1 - Triggered. The trigger for this DMA channel is set. DMA operations will be carried out.
1 —	Reserved Read value is undefined, only zero should be written.
0 VALIDPENDING	Valid pending flag for this channel. This bit is set when a 1 is written to the corresponding bit in the related SETVALID register when CFGVALID = 1 for the same channel. 0 - No effect on DMA operation. 1 - Valid pending.

20.6.1.1.30 Transfer configuration register for DMA channel (XFERCFG0 - XFERCFG51)

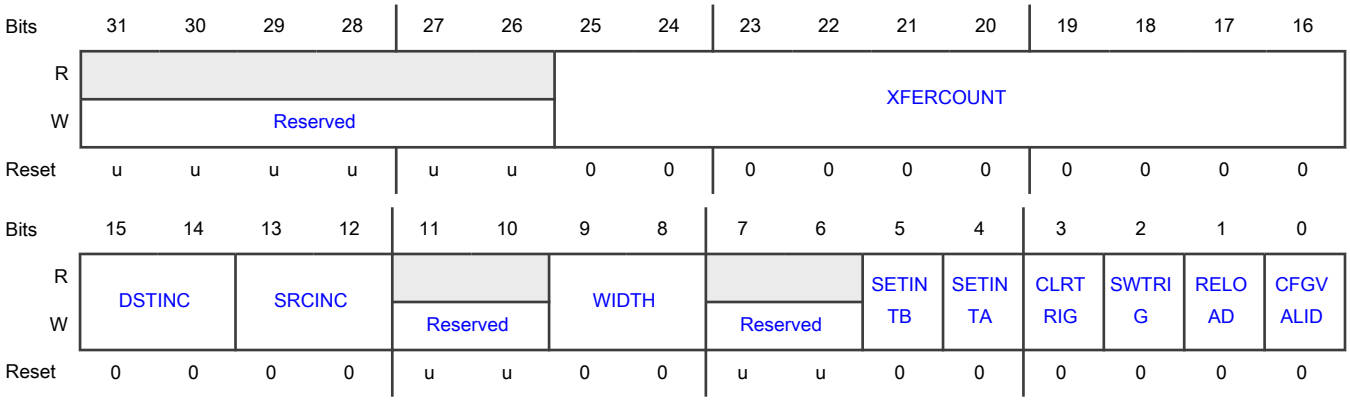
The registers contains transfer related configuration information for the DMA channel. Using the Reload bit, the register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Offset

For a = 0 to 51:

Register	Offset
XFERCFGa	408h + (a × 10h)

Diagram



Fields

Field	Description
31-26 —	Reserved Read value is undefined, only zero should be written.
25-16 XFERCOUNT	Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 = a total of 1 transfer will be performed. 0x1 = a total of 2 transfers will be performed. 0x3FF = a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination address increment Determines whether the destination address is incremented for each DMA transfer. 00 - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01 - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10 - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11 - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.
13-12 SRCINC	Source address increment Determines whether the source address is incremented for each DMA transfer. 00 - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01 - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10 - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11 - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	Reserved Read value is undefined, only zero should be written.
9-8 WIDTH	Transfer width used for this DMA channel. 00 - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01 - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10 - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11 - Reserved.
7-6	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
5 SETINTB	Set Interrupt flag B for channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0 - No effect. 1 - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt flag A for channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0 - No effect. 1 - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.
3 CLRTRIG	Clear Trigger. 0 - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1 - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger. This bit is always read as 0. 0 - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1 - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Reload. Indicates whether the channel's control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0 - Disabled. The channels' control structure should not be reloaded when the current descriptor is exhausted. 1 - Enabled. The channels' control structure should be reloaded when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid flag. This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0 - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1 - Valid. The current channel descriptor is considered valid.

20.6.2 DMA1 controller register descriptions

20.6.2.1 DMA memory map

DMA1 base address: 400A_7000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	DMA control (CTRL)	32	See section	See section
4	Interrupt status (INTSTAT)	32	See section	See section
8	SRAM address of the channel configuration table (SRAMBASE)	32	See section	See section
20	Channel Enable read and set for all DMA channels (ENABLESET0)	32	RW	0000_0000
28	Channel Enable Clear for all DMA channels (ENABLECLR0)	32	See section	See section
30	Channel Active status for all DMA channels (ACTIVE0)	32	See section	0000_0000
38	Channel Busy status for all DMA channels (BUSY0)	32	See section	0000_0000
40	Error Interrupt status for all DMA channels (ERRINT0)	32	See section	0000_0000
48	Interrupt Enable read and Set for all DMA channels (INTENSET0)	32	See section	0000_0000
50	Interrupt Enable Clear for all DMA channels (INTENCLR0)	32	See section	See section
58	Interrupt A status for all DMA channels (INTA0)	32	See section	0000_0000
60	Interrupt B status for all DMA channels (INTB0)	32	See section	0000_0000
68	Set ValidPending control bits for all DMA channels (SETVALID0)	32	See section	See section
70	Set Trigger control bits for all DMA channels (SETTRIG0)	32	See section	See section
78	Channel Abort control for all DMA channels (ABORT0)	32	See section	See section
400 - 4F0	Configuration register for DMA channel (CFG0 - CFG15)	32	See section	See section
404 - 4F4	Control and status register for DMA channel (CTLSTAT0 - CTLSTAT15)	32	RO	See section
408 - 4F8	Transfer configuration register for DMA channel (XFERCFG0 - XFERCFG15)	32	See section	See section

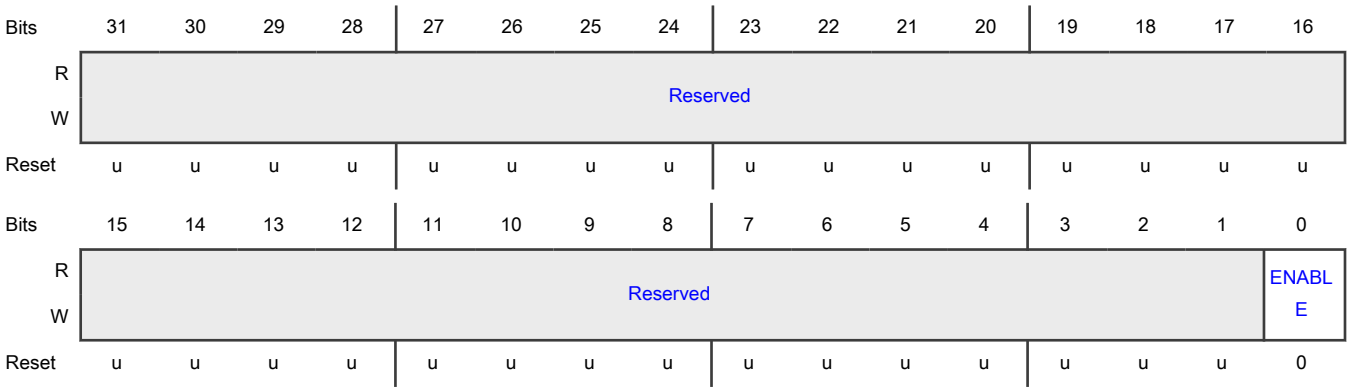
20.6.2.1.1 DMA control (CTRL)

This register has the global control bit for a enabling the DMA controller.

Offset

Register	Offset
CTRL	0h

Diagram



Fields

Field	Description
31-1 —	Reserved
0 ENABLE	DMA controller master enable. 0 - DMA controller is disabled. This clears any triggers that were asserted at the point when disabled, but does not prevent re-triggering when the DMA controller is re-enabled. 1 - Enabled. DMA controller is enabled.

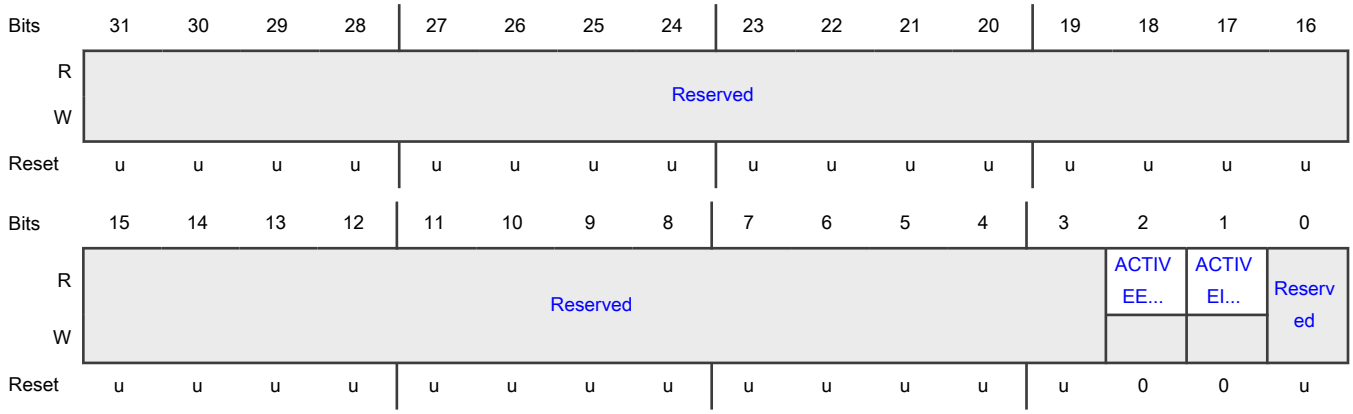
20.6.2.1.2 Interrupt status (INTSTAT)

It provides an overview of DMA status and allows quick determination of whether any enabled interrupts are pending. For details regarding the channels involved see Interrupt A status for all DMA channels (INTA0) and Interrupt B status for all DMA channels (INTB0)

Offset

Register	Offset
INTSTAT	4h

Diagram



Fields

Field	Description
31-3 —	Reserved
2 ACTIVEERRINT	Summarizes whether any error interrupts are pending. 0 - No error interrupts are pending. 1 - At least one error interrupt is pending.
1 ACTIVEINT	Summarizes whether any enabled interrupts (other than error interrupts) are pending. 0 - No enabled interrupts are pending. 1 - At least one enabled interrupt is pending.
0 —	Reserved

20.6.2.1.3 SRAM address of the channel configuration table (SRAMBASE)

It must be configured with an address (preferably in on-chip SRAM) where DMA descriptors will be stored. Software must set up the descriptors for the DMA channels that will be used in the application.

Each DMA channel has an entry for the Channel Descriptor in the SRAM table. See Chip-specific DMA information section.

Offset

Register	Offset
SRAMBASE	8h

Diagram

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

Fields

Field	Description
31-8 OFFSET	Offset Address of the beginning of the DMA descriptor table. The table must begin on a 256 byte boundary.
7-0 —	Reserved

20.6.2.1.4 Channel Enable read and set for all DMA channels (ENABLESET0)

The ENABLESET0 register determines whether each DMA channel is enabled or disabled. Disabling a DMA channel does not reset the channel in any way. A channel can be paused and restarted by clearing, then setting the Enable bit for that channel. Reading ENABLESET0 provides the current state of all of the DMA channels represented by that register. Writing a 1 to a bit position in ENABLESET0 that corresponds to an implemented DMA channel sets the bit, enabling the related DMA channel. Writing a 0 to any bit has no effect. Enables are cleared by writing to ENABLECLR0.

Offset

Register	Offset
ENABLESET0	20h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	ENABLE15	ENABLE14	ENABLE13	ENABLE12	ENABLE11	ENABLE10	ENABLE9	ENABLE8	ENABLE7	ENABLE6	ENABLE5	ENABLE4	ENABLE3	ENABLE2	ENABLE1	ENABLE0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-16 —	Reserved
15-0 ENABLEn	Enable for DMA channel 0 - DMA channel is disabled. 1 - DMA channel is enabled.

20.6.2.1.5 Channel Enable Clear for all DMA channels (ENABLECLR0)

The ENABLECLR0 register is used to clear the enable of one or more DMA channels.

Offset

Register	Offset
ENABLECLR0	28h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
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	CLR15	CLR14	CLR13	CLR12	CLR11	CLR10	CLR9	CLR8	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-16 —	Reserved
15-0 CLRn	Writing ones to this register clears the corresponding bits in ENABLESET0. 0 - No effect. 1 - DMA channel is cleared.

20.6.2.1.6 Channel Active status for all DMA channels (ACTIVE0)

It indicates which DMA channels are active at the point when the read occurs. A DMA channel is considered active when a DMA operation has been started but not yet fully completed. The active status persists another DMA operation from being started, until the pipeline is empty. After end of the last descriptor (when there is no reload), a new DMA operation can be started. An active channel may be aborted by software by setting the appropriate bit in the ABORT register.

Offset

Register	Offset
ACTIVE0	30h

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	ACTIVE15	ACTIVE14	ACTIVE13	ACTIVE12	ACTIVE11	ACTIVE10	ACTIVE9	ACTIVE8	ACTIVE7	ACTIVE6	ACTIVE5	ACTIVE4	ACTIVE3	ACTIVE2	ACTIVE1	ACTIVE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-16 —	Reserved
15-0 ACTIVEn	Active flag for DMA channel. 0 - DMA channel is not active. 1 - DMA channel is active.

20.6.2.1.7 Channel Busy status for all DMA channels (BUSY0)

The BUSY0 register indicates which DMA channels is busy at the point when the read occurs. A DMA channel is considered busy when there is any operation related to that channel in the DMA controller's internal pipeline. This information can be used after a DMA channel is disabled by software (but still active), allowing confirmation that there are no remaining operations in progress for that channel.

Offset

Register	Offset
BUSY0	38h

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

Fields

Field	Description
31-16 —	Reserved
15-0 BUSYn	Busy flag for DMA channel. 0 - DMA channel is not busy. 1 - DMA channel is busy.

20.6.2.1.8 Error Interrupt status for all DMA channels (ERRINT0)

It contains flags for each DMA channel's Error Interrupt. Any pending interrupt flag in the register will be reflected on the DMA interrupt output. Reading the registers provides the current state of all DMA channel error interrupts. Writing a 1 to a bit position in ERRINT0 that corresponds to an implemented DMA channel clears the bit, removing the interrupt for the related DMA channel. Writing a 0 to any bit has no effect.

Offset

Register	Offset
ERRINT0	40h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	ERR15	ERR14	ERR13	ERR12	ERR11	ERR10	ERR9	ERR8	ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-16 —	Reserved
15-0 ERRn	Error Interrupt flag for DMA channel. 0 - The Error Interrupt is not active for DMA channel. 1 - The Error Interrupt is pending for DMA channel.

20.6.2.1.9 Interrupt Enable read and Set for all DMA channels (INTENSET0)

It controls whether the individual Interrupts for DMA channels contribute to the DMA interrupt output. Reading the registers provides the current state of all DMA channel interrupt enables. Writing a 1 to a bit position in INTENSET0 that corresponds to an implemented DMA channel sets the bit, enabling the interrupt for the related DMA channel. Writing a 0 to any bit has no effect. Interrupt enables are cleared by writing to INTENCLR0.

Offset

Register	Offset
INTENSET0	48h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	INTEN15	INTEN14	INTEN13	INTEN12	INTEN11	INTEN10	INTEN9	INTEN8	INTEN7	INTEN6	INTEN5	INTEN4	INTEN3	INTEN2	INTEN1	INTEN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-16 —	Reserved
15-0 INTENn	Interrupt Enable read and set for DMA channel. 0 - The Interrupt for DMA channel is disabled. 1 - The Interrupt for DMA channel is enabled.

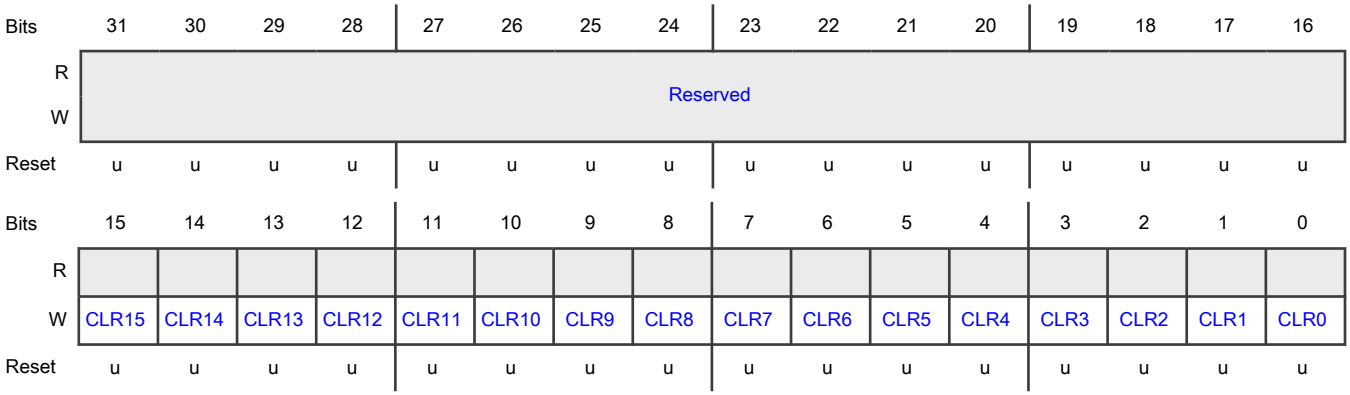
20.6.2.1.10 Interrupt Enable Clear for all DMA channels (INTENCLR0)

This register is used to clear interrupt enable bits in INTENSET0. The register is write-only.

Offset

Register	Offset
INTENCLR0	50h

Diagram



Fields

Field	Description
31-16 —	Reserved
15-0 CLRn	Writing ones to this register clears corresponding bits in the DMAIntEnSet0.

20.6.2.1.11 Interrupt A status for all DMA channels (INTA0)

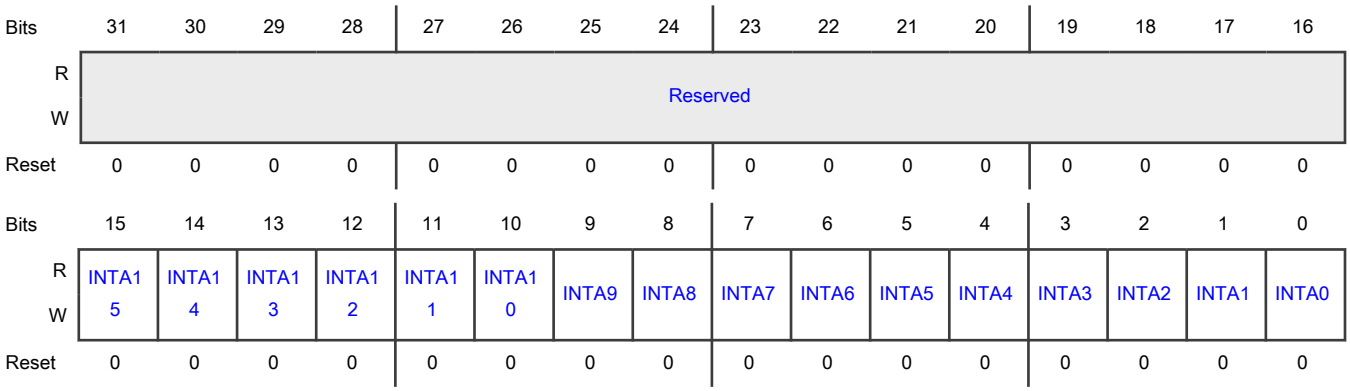
It contains the interrupt A status for each DMA channel. The status will be set when the SETINTA bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in this register clears the related INTA

flag. Writing 0 has no effect. Any interrupt pending status in the registers will be reflected on the DMA interrupt output if it is enabled in the related INTENSET register.

Offset

Register	Offset
INTA0	58h

Diagram



Fields

Field	Description
31-16 —	Reserved
15-0 INTAn	Interrupt A status for DMA channel. 0 - The DMA channel interrupt A is not active. 1 - The DMA channel interrupt A is active.

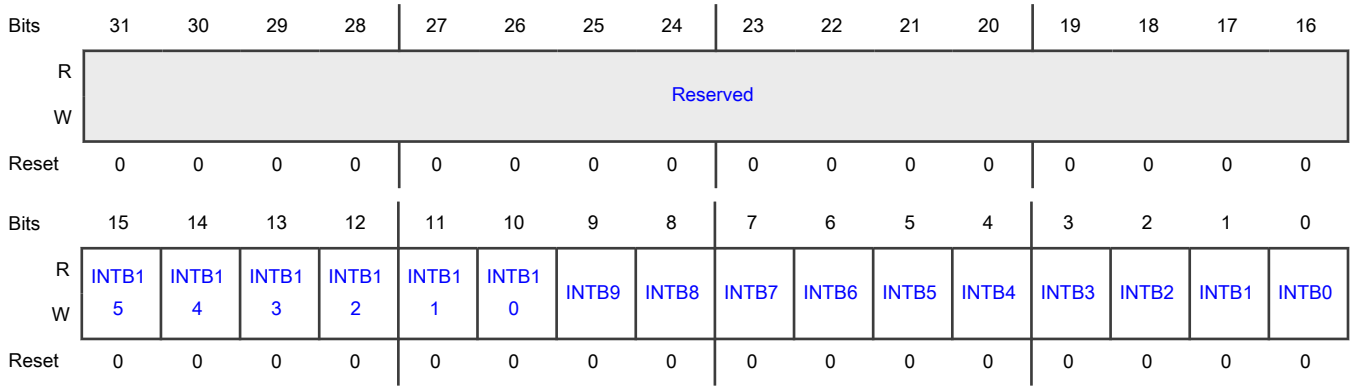
20.6.2.1.12 Interrupt B status for all DMA channels (INTB0)

It contains the interrupt B status for each DMA channel. The status will be set when the SETINTB bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in the register clears the related INTB flag. Writing 0 has no effect. Any interrupt pending status in this register will be reflected on the DMA interrupt output if it is enabled in the INTENSET register.

Offset

Register	Offset
INTB0	60h

Diagram



Fields

Field	Description
31-16 —	Reserved
15-0 INTBn	Interrupt B status for DMA channel. 0 - The DMA channel interrupt B is not active. 1 - The DMA channel interrupt B is active.

20.6.2.1.13 Set ValidPending control bits for all DMA channels (SETVALID0)

The SETVALID0 register allows setting the Valid bit in the CTRLSTAT register for one or more DMA channels. The CFGVALID and SV (set valid) bits allow more direct DMA block timing control by software. Each Channel Descriptor, in a sequence of descriptors, can be validated by either the setting of the CFGVALID bit or by setting the channel's SETVALID flag. Normally, the CFGVALID bit is set. This tells the DMA that the Channel Descriptor is active and can be executed. The DMA will continue sequencing through descriptor blocks whose CFGVALID bit are set without further software intervention. Leaving a CFGVALID bit set to 0 allows the DMA sequence to pause at the Descriptor until software triggers the continuation. If, during DMA transmission, a Channel Descriptor is found with CFGVALID set to 0, the DMA checks for a previously buffered SETVALID0 setting for the channel. If found, the DMA will set the descriptor valid, clear the SV setting, and resume processing the descriptor. Otherwise, the DMA pauses until the channels SETVALID0 bit is set.

Offset

Register	Offset
SETVALID0	68h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
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																
W	SETVALI...	SETVALI...	SETVALI...	SETVALI...	SETVALI...	SETVALI...	SETVALI...	SETVALI...	SETVALI...	SETVALI...	SETVALI...	SETVALI...	SETVALI...	SETVALI...	SETVALI...	SETVALI...
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-16 —	Reserved
15-0 SETVALIDn	SetValid control for DMA channel. 0 - No effect. 1 - Sets the ValidPending control bit for DMA channel.

20.6.2.1.14 Set Trigger control bits for all DMA channels (SETTRIG0)

This register allows setting the TRIG bit in the CTRLSTAT register for one or more DMA channel.

Offset

Register	Offset
SETTRIG0	70h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
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																
W	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG...	SETTRIG9	SETTRIG8	SETTRIG7	SETTRIG6	SETTRIG5	SETTRIG4	SETTRIG3	SETTRIG2	SETTRIG1	SETTRIG0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-16 —	Reserved
15-0 SETTRIGn	Set Trigger control bit for DMA channel. 0 - No effect. 1 - Sets the Trig bit for DMA channel.

20.6.2.1.15 Channel Abort control for all DMA channels (ABORT0)

The Abort 0 register allows aborting operation of a DMA channel if needed. To abort a selected channel, the channel should first be disabled by clearing the corresponding Enable bit by writing a 1 to the proper bit in ENABLECLR. Then wait until the channel is no longer busy by checking the corresponding bit in BUSY. Finally, write a 1 to the proper bit of ABORT. This prevents the channel from restarting an incomplete operation when it is enabled again.

Offset

Register	Offset
ABORT0	78h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
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																
W	ABORT15	ABORT14	ABORT13	ABORT12	ABORT11	ABORT10	ABORT9	ABORT8	ABORT7	ABORT6	ABORT5	ABORT4	ABORT3	ABORT2	ABORT1	ABORT0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-16 —	Reserved
15-0 ABORTn	Abort control for DMA channel. 0 - No effect. 1 - Aborts DMA operations on channel.

20.6.2.1.16 Configuration register for DMA channel (CFG0 - CFG15)

The CFG register contains various configuration options for DMA channels.

Table 106. Trigger setting summary

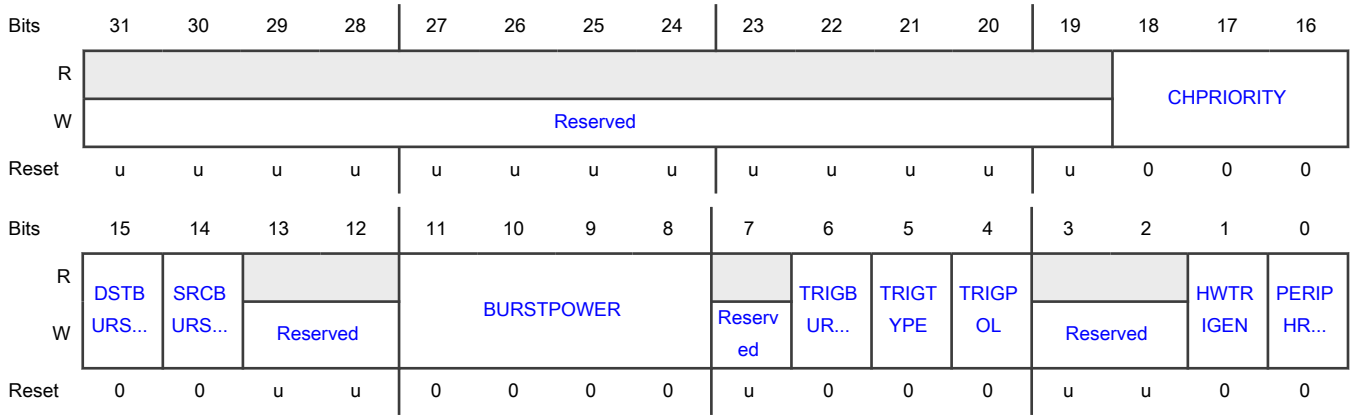
TrigBurst	TrigType	TrigPol	Description
0	0	0	Hardware DMA trigger is falling edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	0	1	Hardware DMA trigger is rising edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	1	0	Hardware DMA trigger is low level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	1	1	Hardware DMA trigger is high level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
1	0	0	Hardware DMA trigger is falling edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	0	1	Hardware DMA trigger is rising edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	1	0	Hardware DMA trigger is low level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	1	1	Hardware DMA trigger is high level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.

Offset

For a = 0 to 15:

Register	Offset
CFGa	400h + (a × 10h)

Diagram



Fields

Field	Description
31-19 —	Reserved Read value is undefined, only zero should be written.
18-16 CHPRIORITY	Priority of channel when multiple DMA requests are pending. Eight priority levels are supported: 0x0 = highest priority. 0x7 = lowest priority.
15 DSTBURSTWRAP	Destination Burst Wrap. When enabled, the destination data address for the DMA is 'wrapped', meaning that the destination address range for each burst will be the same. As an example, this could be used to write several sequential registers to a peripheral for each DMA burst, writing the same registers again for each burst. 0 - Disabled. Destination burst wrapping is not enabled for this DMA channel. 1 - Enabled. Destination burst wrapping is enabled for this DMA channel.
14 SRCBURSTWRAP	Source Burst Wrap. When enabled, the source data address for the DMA is 'wrapped', meaning that the source address range for each burst will be the same. As an example, this could be used to read several sequential registers from a peripheral for each DMA burst, reading the same registers again for each burst. 0 - Disabled. Source burst wrapping is not enabled DMA channel. 1 - Enabled. Source burst wrapping is enabled DMA channel.
13-12 —	Reserved Read value is undefined, only zero should be written.
11-8 BURSTPOWER	Burst Power. It always selects the address wrap size when SRCBURSTWRAP and/or DSTBURSTWRAP modes are selected.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	When the TRIGBURST field = 1, Burst Power selects how many transfers are performed for each DMA trigger. This can be used, for example, with peripherals that contain a FIFO that can initiate a DMA operation when the FIFO reaches a certain level. 0000: Burst size = 1 (20). This corresponds to a single transfer. 0001: Burst size = 2 (2^1). 0010: Burst size = 4 (2^2). 1010: Burst size = 1024 (2^{10}). This corresponds to the maximum supported transfer count. Others: not supported. The total transfer length as defined in the XFERCOUNT bits in the XFERCFG register must be an integer multiple of the burst size. **NOTE** Total number of bytes transferred is: (XFERCOUNT + 1) x data width (Transfer configuration register for DMA channel (XFERCFG0 - XFERCFG15)).
7 —	Reserved Read value is undefined, only zero should be written.
6 TRIGBURST	Trigger Burst. Selects whether hardware triggers cause a single or burst transfer. 0 - Single transfer. Hardware trigger causes a single transfer. 1 - Burst transfer. When the trigger for this channel is set to edge triggered, a hardware trigger causes a burst transfer, as defined by BURSTPOWER. When the trigger for this channel is set to level triggered, a hardware trigger causes transfers to continue as long as the trigger is asserted, unless the transfer is complete.
5 TRIGTYPE	Trigger Type. Selects hardware trigger as edge triggered or level triggered. 0 - Edge. Hardware trigger is edge triggered. Transfers will be initiated and completed, as specified for a single trigger. 1 - Level. Hardware trigger is level triggered. Note that when level triggering without burst (BURSTPOWER = 0) is selected, only hardware triggers should be used on that channel. Transfers continue as long as the trigger level is asserted. Once the trigger is de-asserted, the transfer will be paused until the trigger is, again, asserted. However, the transfer will not be paused until any remaining transfers within the current BURSTPOWER length are completed.
4 TRIGPOL	Trigger Polarity. Selects the polarity of a hardware trigger for this channel. 0 - Active low - falling edge. Hardware trigger is active low or falling edge triggered, based on TRIGTYPE. 1 - Active high - rising edge. Hardware trigger is active high or rising edge triggered, based on TRIGTYPE.
3-2	Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
1 HWTRIGEN	Hardware Triggering Enable for channel. 0 - Hardware triggering not used for channel. 1 - Hardware triggering used for channel.
0 PERIPHREQEN	Peripheral request Enable. If a DMA channel is used to perform a memory-to-memory transfer, any peripheral DMA request associated with that channel can be disabled to prevent any interaction between the peripheral and the DMA controller. 0 - Peripheral DMA requests disabled. 1 - Peripheral DMA requests enabled.

20.6.2.1.17 Control and status register for DMA channel (CTLSTAT0 - CTLSTAT15)

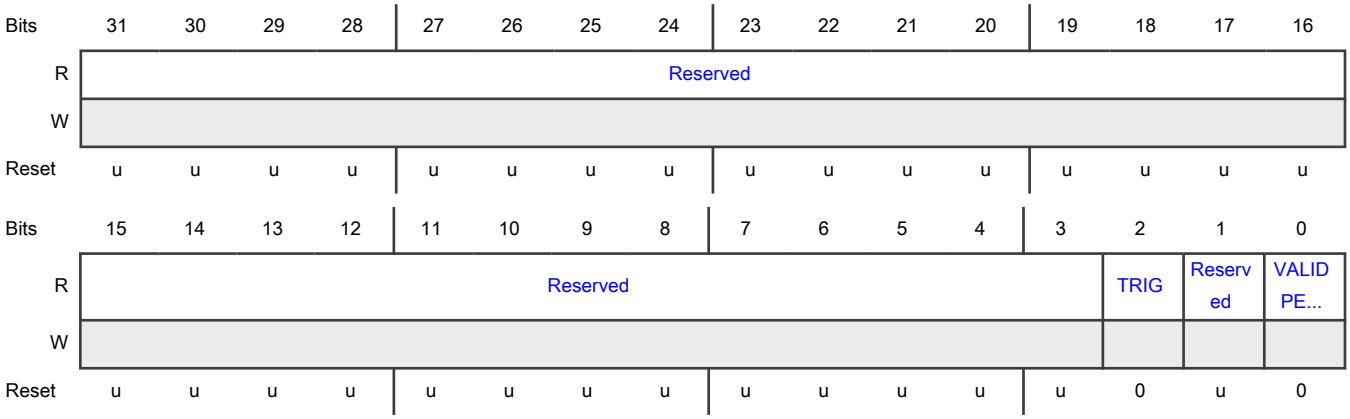
It provides status flags specific to the DMA channel.

Offset

For a = 0 to 15:

Register	Offset
CTLSTATa	404h + (a × 10h)

Diagram



Fields

Field	Description
31-3	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
2 TRIG	Trigger flag. Indicates that the trigger for this channel is currently set. This bit is cleared at the end of an entire transfer or upon reload when CLRTRIG = 1. 0 - Not triggered. The trigger for this DMA channel is not set. DMA operations will not be carried out. 1 - Triggered. The trigger for this DMA channel is set. DMA operations will be carried out.
1 —	Reserved Read value is undefined, only zero should be written.
0 VALIDPENDING	Valid pending flag for this channel. This bit is set when a 1 is written to the corresponding bit in the related SETVALID register when CFGVALID = 1 for the same channel. 0 - No effect on DMA operation. 1 - Valid pending.

20.6.2.1.18 Transfer configuration register for DMA channel (XFERCFG0 - XFERCFG15)

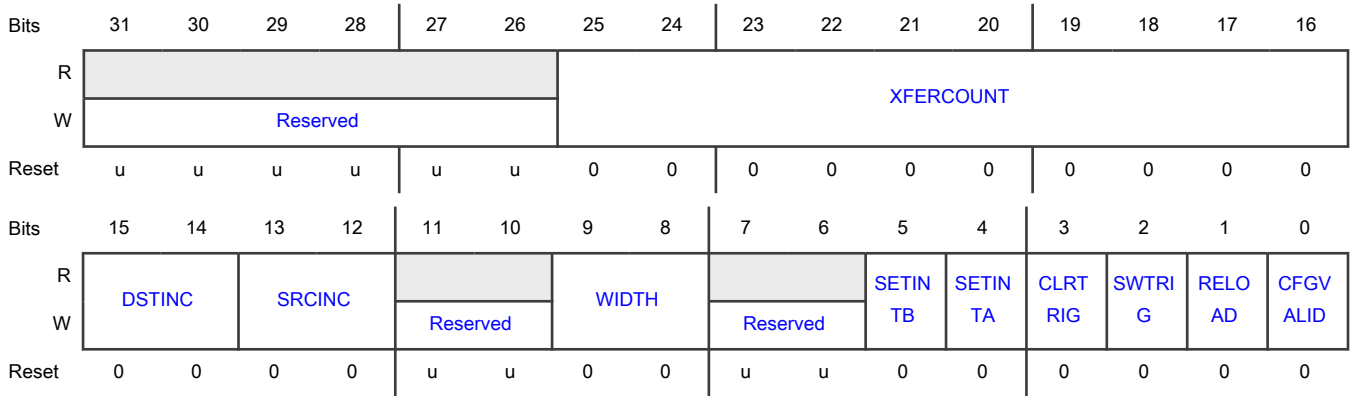
The registers contains transfer related configuration information for the DMA channel. Using the Reload bit, the register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

Offset

For a = 0 to 15:

Register	Offset
XFERCFGa	408h + (a × 10h)

Diagram



Fields

Field	Description
31-26 —	Reserved Read value is undefined, only zero should be written.
25-16 XFERCOUNT	Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 = a total of 1 transfer will be performed. 0x1 = a total of 2 transfers will be performed. 0x3FF = a total of 1,024 transfers will be performed.
15-14 DSTINC	Destination address increment Determines whether the destination address is incremented for each DMA transfer. 00 - No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. 01 - 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. 10 - 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. 11 - 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.
13-12 SRCINC	Source address increment Determines whether the source address is incremented for each DMA transfer. 00 - No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. 01 - 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. 10 - 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. 11 - 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.
11-10 —	Reserved Read value is undefined, only zero should be written.
9-8 WIDTH	Transfer width used for this DMA channel. 00 - 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). 01 - 16-bit. 16-bit transfers are performed (16-bit source reads and destination writes). 10 - 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). 11 - Reserved.
7-6	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
5 SETINTB	Set Interrupt flag B for channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0 - No effect. 1 - Set. The INTB flag for this channel will be set when the current descriptor is exhausted.
4 SETINTA	Set Interrupt flag A for channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. 0 - No effect. 1 - Set. The INTA flag for this channel will be set when the current descriptor is exhausted.
3 CLRTRIG	Clear Trigger. 0 - Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. 1 - Cleared. The trigger is cleared when this descriptor is exhausted.
2 SWTRIG	Software Trigger. This bit is always read as 0. 0 - Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. 1 - Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.
1 RELOAD	Reload. Indicates whether the channel's control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. 0 - Disabled. The channels' control structure should not be reloaded when the current descriptor is exhausted. 1 - Enabled. The channels' control structure should be reloaded when the current descriptor is exhausted.
0 CFGVALID	Configuration Valid flag. This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. 0 - Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. 1 - Valid. The current channel descriptor is considered valid.

Chapter 21
And-Or-Inverter (AOI)

21.1 Chip-specific AOI information

Table 107. Reference links to related information

Topic	Related module	Reference
Full description	AOI	AOI
System memory map		System memory map
Clocking		Clock distribution
Internal signal connections	INPUTMUX	INPUTMUX
Signal multiplexing	Port control	Signal multiplexing

21.1.1 Module instances

This device has two instances of the AOI module, AOI0 and AOI1.

21.2 Overview

The AND/OR/INVERT module (known simply as the AOI module) supports the generation of a configurable number of EVENT signals. Each output EVENTn is a configurable and/or/invert function of four associated AOI inputs: An, Bn, Cn, and Dn.

This module is designed to be integrated in conjuction with one or more inter-peripheral crossbar switch (XBAR) modules. A crossbar switch is typically used to select the 4*n AOI inputs from among available peripheral outputs and GPIO signals. The n EVENTn outputs from the AOI module are typically used as additional inputs to a second crossbar switch, adding to it the ability to connect to its outputs an arbitrary 4-input boolean function of its other inputs.

The AOI controller is a slave peripheral module connecting event input indicators from a variety of device modules and generating event output signals that can be routed to an inter-peripheral crossbar switch or other peripherals. Its programming model is accessed through the standard IPS (Sky Blue) slave interface. The module is designed to be very configurable in terms of the functionality of its integrated AOI functions.

21.2.1 Block diagram

The AOI module supports a configurable number of event outputs, where each event output represents a user-programmed combinational boolean function based on four event inputs. The key features of this module include:

- Four dedicated inputs for each event output
- User-programmable combinational boolean function evaluation for each event output
- Memory-mapped device connected to a slave peripheral (IPS) bus
- Configurable number of event outputs

NOTE

The connections from the AOI module outputs to other functions is SoC-specific.

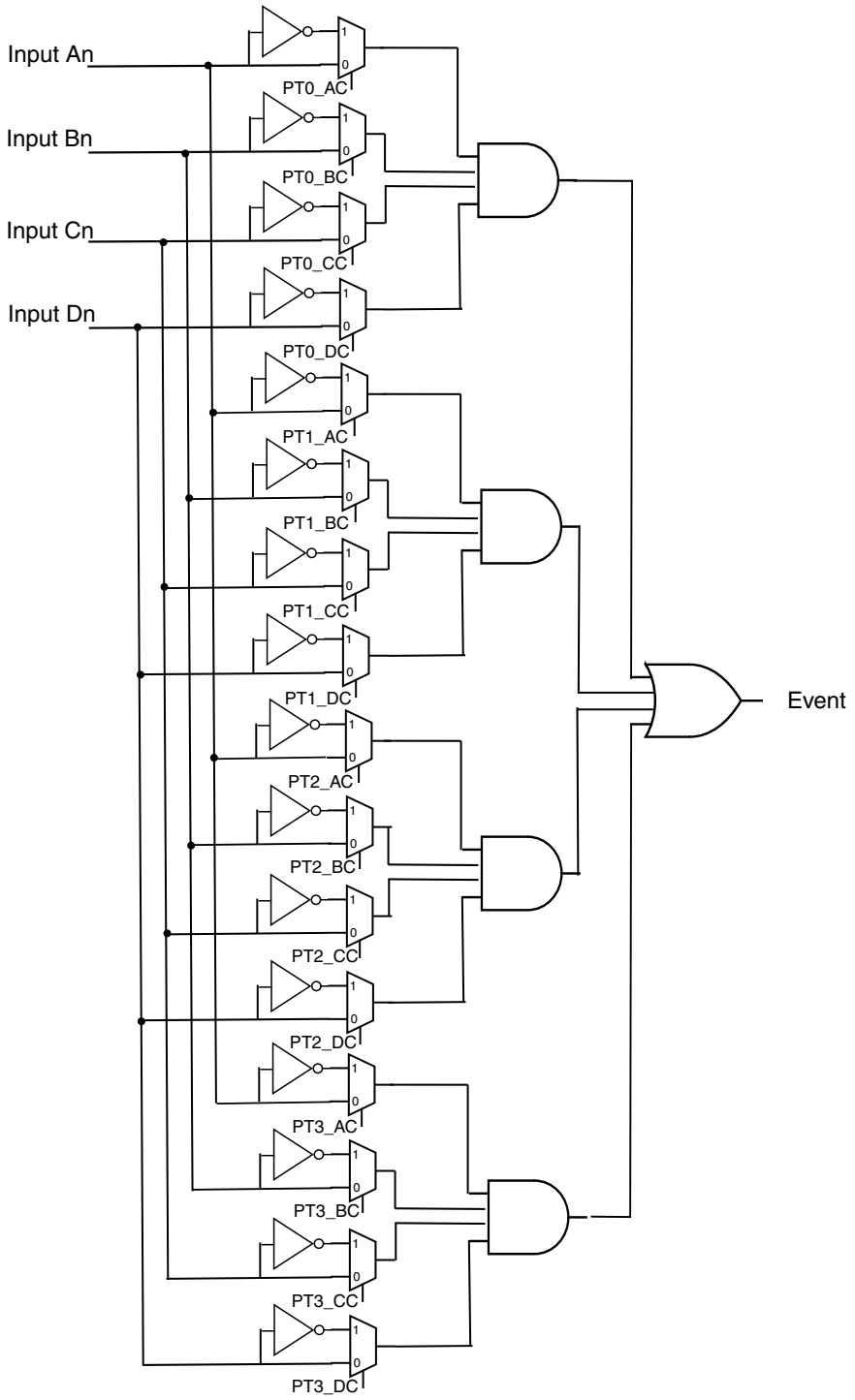


Figure 75. Simplified AOI Block Diagram

21.2.2 Features

The major features of the AOI module are summarized below:

- Highly programmable module for creating combinational boolean events for use as hardware triggers
  - Each channel has four event inputs and one output

  - Evaluates a combinational boolean expression as the sum of four products where each product term includes all four selected input sources available as true or complement values
  - Event output is formed as purely combinational logic and operates as a hardware trigger
- Memory-mapped device connected to the slave peripheral (IPS) bus
  - Programming model organized per channel for simplified software

21.2.3 Modes of Operation

The AOI module does not support any special modes of operation. As shown in Figure 75, its operation is primarily controlled by the selected event inputs and outputs. Additionally, as a memory-mapped device located on the slave peripheral bus, it responds based strictly on memory address for accesses to its programming model.

The AOI module resides in the slave peripheral *bus clock domain*.

21.3 Functional Description

The AOI is a highly programmable module for creating combinational boolean outputs for use as hardware triggers. Each AOI output channel, as shown in Figure 75, has one logic function:

- Evaluation of a combinational boolean expression as a sum of four products where each product term includes all four selected input sources available as true or complement values

A typical application of the AOI module is to be integrated with one or more inter-peripheral crossbar switch modules as illustrated in the following figure. The 20 external inputs are shared by two crossbar switch modules. The crossbar switch on the top is used to select the inputs to four 4-input AOI functions in the AOI module. The outputs of these four AOI functions are output from the AOI module and are added to the original 20 external inputs to provide a total of 24 inputs to the bottom crossbar switch. As a result, the bottom crossbar can not only direct any of the original 20 external inputs to any of its outputs, it can also now direct any one of four 4-input AOI functions of those external inputs to any of its outputs.

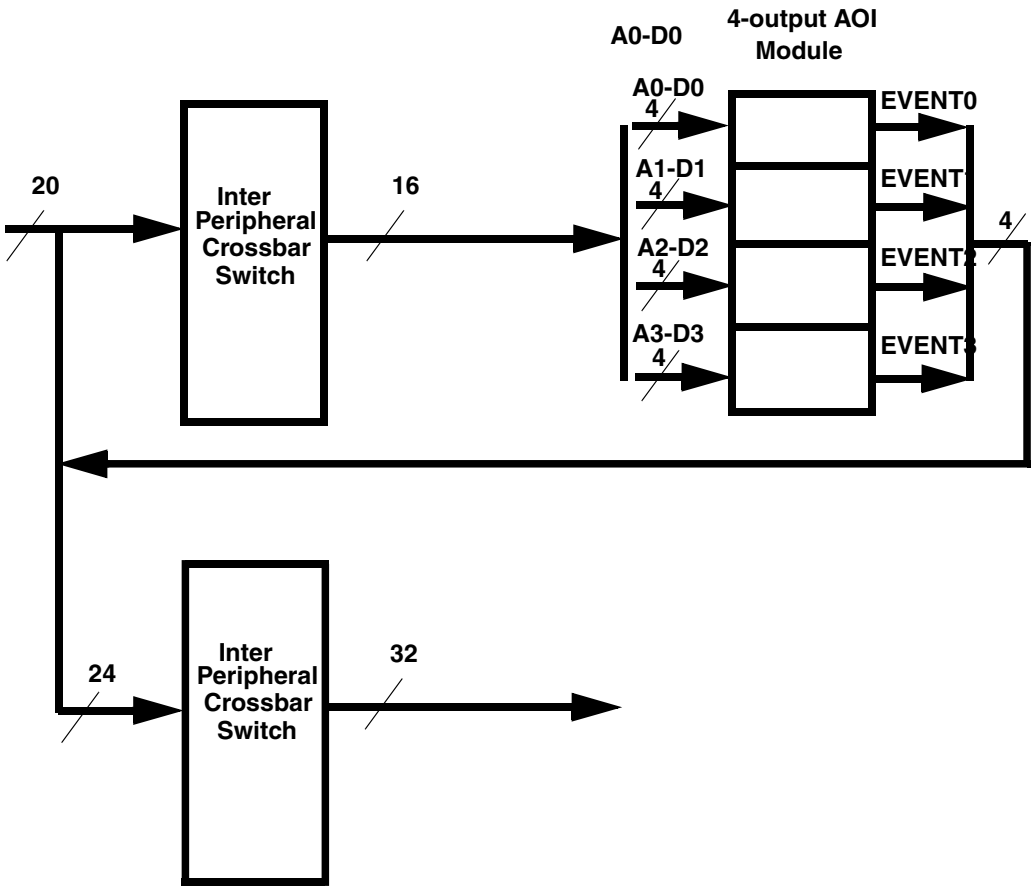


Figure 76. Integration Example of AOI with two Inter-Peripheral Crossbar Switches

21.3.1 Configuration Examples for the Boolean Function Evaluation

This section presents examples of the programming model configuration for simple boolean expressions.

The AOI module provides a universal boolean function generator using a four-term sum of products expression with each product term containing true or complement values of the four selected event inputs (A, B, C, D). Specifically, the event output is defined by the following “4 x 4” boolean expression:

```
EVENTn
    = (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 0
    | (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 1
    | (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 2
    | (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 3
```

where each selected input term in each product term can be configured to produce a logical 0 or 1 or pass the true or complement of the selected event input. Each product term uses eight bits of configuration information, two bits for each of the four selected event inputs. The actual boolean expression implemented in each channel is:

```
EVENTn
    = (PT0_AC[0] & An |  PT0_AC[1] & ~An)// product term 0
```

```
                    & (PT0_BC[0] &  Bn | PTO_BC[1] & ~Bn)
                    & (PT0_CC[0] &  Cn | PTO_CC[1] & ~Cn)
                    & (PT0_DC[0] &  Dn | PTO_DC[1] & ~Dn)

    | (PT1_AC[0] & An |  PT1_AC[1] & ~An)// product term 1
                    & (PT1_BC[0] &  Bn | PT1_BC[1] & ~Bn)
                    & (PT1_CC[0] &  Cn | PT1_CC[1] & ~Cn)
                    & (PT1_DC[0] &  Dn | PT1_DC[1] & ~Dn)

    | (PT2_AC[0] & An |  PT2_AC[1] & ~An)// product term 2
                    & (PT2_BC[0] &  Bn | PT2_BC[1] & ~Bn)
                    & (PT2_CC[0] &  Cn | PT2_CC[1] & ~Cn)
                    & (PT2_DC[0] &  Dn | PT2_DC[1] & ~Dn)

    | (PT3_AC[0] & An |  PT3_AC[1] & ~An)// product term 3
                    & (PT3_BC[0] &  Bn | PT3_BC[1] & ~Bn)
                    & (PT3_CC[0] &  Cn | PT3_CC[1] & ~Cn)
                    & (PT3_DC[0] &  Dn | PT3_DC[1] & ~Dn)
```

where the bits of the combined {BFCRT01n,BFCRT23n} registers correspond to the `PT{0-3}_{A,B,C,D}C[1:0]` terms in the equation.

Consider the settings of the combined 32-bit {BFCRT01n,BFCRT23n} registers for several simple boolean expressions as shown in Table 108.

Table 108. BFCRTn Values for Simple Boolean Expressions

Event Output Expression	PT0	PT1	PT2	PT3	{BFCRT01, BFCRT23}
A & B	A & B	0	0	0	01011111_00000000_00000000_00000000
A & B & C	A & B & C	0	0	0	01010111_00000000_00000000_00000000
(A & B & C) + D	A & B & C	D	0	0	01010111_11111101_00000000_00000000
A + B + C + D	A	B	C	D	01111111_11011111_11110111_11111101
(A & ~B) + (~A & B)	A & ~B	~A & B	0	0	01101111_10011111_00000000_00000000

As can be seen in these examples, the resulting logic provides a simple yet powerful boolean function evaluation for defining an event output.

21.3.2 AOI Timing Between Inputs and Outputs

Each EVENTn output of the AOI module is a combination function of its four dedicated inputs An, Bn, Cn, and Dn. Propagation through the AOI and any associated inter-peripheral crossbar switch modules is intended to be single bus clock cycle.

21.3.3 Clocks

The bus clock is used for register accesses by the system, but AOI functions themselves are asynchronous and do not use a clock.

21.4 External Signal Description

The AOI module does not directly support any external interfaces. There may be package input signals (indirectly) connected to the module as event inputs, but since the *AOI does not include any input synchronization hardware*, this function must be handled before the event input signals are routed into the module.

21.5 Memory Map and Register Descriptions

The AOI module supports access to its programming model via a 16-bit peripheral bus connection. The module is designed to support 16-bit accesses only. Functionality for accesses of other widths is undefined.

21.5.1 AOI register descriptions

The AOI module supports a specific number of event outputs. Each output EVENTn outputs a four-term AOI function of four binary inputs: An, Bn, Cn, and Dn. A pair of 16-bit registers configures this four-term AOI function: The two registers BFCRT01n and BFCRT23n define the configuration for the evaluation of the Boolean function defining EVENTn, where *n* is the event output channel number. The BFCRT01n register defines the configuration of product terms 0 and 1, and the BFCRT23n register defines the configuration of product terms 2 and 3.

The AOI module provides a universal Boolean function generator using a four-term sum of products expression with each product term containing true or complement values of the four selected event inputs (An, Bn, Cn, Dn). Specifically, the EVENTn output is defined by the following "4 x 4" Boolean expression:

```
EVENTn
    = (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 0
    | (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 1
    | (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 2
    | (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 3
```

where each selected input of each product term can be configured to produce a logical 0 or 1 or pass the true or complement of the selected event input. Each product term uses 8 bits of configuration information, 2 bits for each of the four selected event inputs. The resulting logic provides a simple yet powerful Boolean function evaluation for defining an event output.

These AOI functions are combinational in nature and are intended to be sampled and used synchronously.

21.5.1.1 AOI memory map

AOI0 base address: 400C_7000h

AOI1 base address: 400C_8000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Boolean Function Term 0 and 1 Configuration Register for EVENTn (BFCRT010)	16	RW	0000
2	Boolean Function Term 2 and 3 Configuration Register for EVENTn (BFCRT230)	16	RW	0000
4	Boolean Function Term 0 and 1 Configuration Register for EVENTn (BFCRT011)	16	RW	0000
6	Boolean Function Term 2 and 3 Configuration Register for EVENTn (BFCRT231)	16	RW	0000
8	Boolean Function Term 0 and 1 Configuration Register for EVENTn (BFCRT012)	16	RW	0000
A	Boolean Function Term 2 and 3 Configuration Register for EVENTn (BFCRT232)	16	RW	0000
C	Boolean Function Term 0 and 1 Configuration Register for EVENTn (BFCRT013)	16	RW	0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
E	Boolean Function Term 2 and 3 Configuration Register for EVENTn (BFCRT233)	16	RW	0000

21.5.1.1.1 Boolean Function Term 0 and 1 Configuration Register for EVENTn (BFCRT010 - BFCRT013)

Offset

Register	Offset
BFCRT010	0h
BFCRT011	4h
BFCRT012	8h
BFCRT013	Ch

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PT0_AC		PT0_BC		PT0_CC		PT0_DC		PT1_AC		PT1_BC		PT1_CC		PT1_DC	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
15-14 PT0_AC	Product term 0, A input configuration This 2-bit field defines the Boolean evaluation associated with the selected input A in product term 0. 00 - Force the A input in this product term to a logical zero 01 - Pass the A input in this product term 10 - Complement the A input in this product term 11 - Force the A input in this product term to a logical one
13-12 PT0_BC	Product term 0, B input configuration This 2-bit field defines the Boolean evaluation associated with the selected input B in product term 0. 00 - Force the B input in this product term to a logical zero 01 - Pass the B input in this product term 10 - Complement the B input in this product term

Table continues on the next page...

Table continued from the previous page...

Field	Description
	11 - Force the B input in this product term to a logical one
11-10 PT0_CC	Product term 0, C input configuration This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 0. 00 - Force the C input in this product term to a logical zero 01 - Pass the C input in this product term 10 - Complement the C input in this product term 11 - Force the C input in this product term to a logical one
9-8 PT0_DC	Product term 0, D input configuration This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 0. 00 - Force the D input in this product term to a logical zero 01 - Pass the D input in this product term 10 - Complement the D input in this product term 11 - Force the D input in this product term to a logical one
7-6 PT1_AC	Product term 1, A input configuration This 2-bit field defines the Boolean evaluation associated with the selected input A in product term 1. 00 - Force the A input in this product term to a logical zero 01 - Pass the A input in this product term 10 - Complement the A input in this product term 11 - Force the A input in this product term to a logical one
5-4 PT1_BC	Product term 1, B input configuration This 2-bit field defines the Boolean evaluation associated with the selected input B in product term 1. 00 - Force the B input in this product term to a logical zero 01 - Pass the B input in this product term 10 - Complement the B input in this product term 11 - Force the B input in this product term to a logical one
3-2 PT1_CC	Product term 1, C input configuration This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 1. 00 - Force the C input in this product term to a logical zero 01 - Pass the C input in this product term 10 - Complement the C input in this product term 11 - Force the C input in this product term to a logical one
1-0	Product term 1, D input configuration

Table continues on the next page...

Table continued from the previous page...

Field	Description
PT1_DC	This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 1. 00 - Force the D input in this product term to a logical zero 01 - Pass the D input in this product term 10 - Complement the D input in this product term 11 - Force the D input in this product term to a logical one

21.5.1.1.2 Boolean Function Term 2 and 3 Configuration Register for EVENTn (BFCRT230 - BFCRT233)

Offset

Register	Offset
BFCRT230	2h
BFCRT231	6h
BFCRT232	Ah
BFCRT233	Eh

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PT2_AC		PT2_BC		PT2_CC		PT2_DC		PT3_AC		PT3_BC		PT3_CC		PT3_DC	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
15-14 PT2_AC	Product term 2, A input configuration This 2-bit field defines the Boolean evaluation associated with the selected input A in product term 2. 00 - Force the A input in this product term to a logical zero 01 - Pass the A input in this product term 10 - Complement the A input in this product term 11 - Force the A input in this product term to a logical one
13-12 PT2_BC	Product term 2, B input configuration This 2-bit field defines the Boolean evaluation associated with the selected input B in product term 2. 00 - Force the B input in this product term to a logical zero

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01 - Pass the B input in this product term 10 - Complement the B input in this product term 11 - Force the B input in this product term to a logical one
11-10 PT2_CC	Product term 2, C input configuration This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 2. 00 - Force the C input in this product term to a logical zero 01 - Pass the C input in this product term 10 - Complement the C input in this product term 11 - Force the C input in this product term to a logical one
9-8 PT2_DC	Product term 2, D input configuration This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 2. 00 - Force the D input in this product term to a logical zero 01 - Pass the D input in this product term 10 - Complement the D input in this product term 11 - Force the D input in this product term to a logical one
7-6 PT3_AC	Product term 3, A input configuration This 2-bit field defines the Boolean evaluation associated with the selected input A in product term 3. 00 - Force the A input in this product term to a logical zero 01 - Pass the A input in this product term 10 - Complement the A input in this product term 11 - Force the A input in this product term to a logical one
5-4 PT3_BC	Product term 3, B input configuration This 2-bit field defines the Boolean evaluation associated with the selected input B in product term 3. 00 - Force the B input in this product term to a logical zero 01 - Pass the B input in this product term 10 - Complement the B input in this product term 11 - Force the B input in this product term to a logical one
3-2 PT3_CC	Product term 3, C input configuration This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 3. 00 - Force the C input in this product term to a logical zero 01 - Pass the C input in this product term 10 - Complement the C input in this product term

Table continues on the next page...

Table continued from the previous page...

Field	Description
	11 - Force the C input in this product term to a logical one
1-0 PT3_DC	Product term 3, D input configuration This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 3. 00 - Force the D input in this product term to a logical zero 01 - Pass the D input in this product term 10 - Complement the D input in this product term 11 - Force the D input in this product term to a logical one

Chapter 22
Frequency Measurement (FREQMEASURE)

22.1 Chip-specific Frequency Measurement information

Table 109. Reference links to related information

Topic	Related module	Reference
Full description	FREQMEASURE	Frequency Measurement
System memory map		System memory map
Clocking	SYSCON	Clock generation
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

22.1.1 Module instances

This device has one instance of the Frequency Measurement module.

22.1.2 Chip-specific initialization

To initialize the Frequency Measurement module:

- Enable the clock to the Frequency Measurement function (AHBCLKCTRL2[FREQME] = 1) in SYSCON registers. This enables the register interface and the peripheral function clock.
- Reset the Frequency Measurement function peripheral reset (PRESETCTRL2[FREQME] = 1) in SYSCON regiters.
- Use the IOPCTL registers to connect the FREQME_GPIO_CLK input (if it is used) to an external pin.
- Make sure the clocks used as the reference and target are enabled.
- Select reference and target clocks using input mux block.
  - Reference clock selection: FREQMEAS_REF[CLKIN]
  - Target clock selection: FREQMEAS_TAR[CLKIN]

22.2 Overview

FREQME accurately measures the frequency of an on- or off-chip target clock signal using a selectable on-chip reference clock. For example, it can accurately determine the frequency of a low-power oscillator that varies depending on process and temperature.

22.2.1 Block diagram

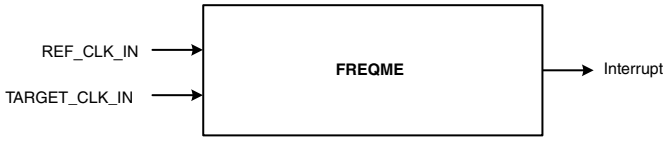


Figure 77. Block diagram

See the chip-specific FREQME information for the target and reference clock selection options.

22.2.2 Features

- High-accuracy Frequency Measurement mode for on- and off-chip clocks
- Pulse Width Measurement (PWM) mode
- Reference and target clock inputs selectable from among various chip-specific options
- Optional measurement complete interrupt
- Result out-of-range detection with optional interrupt

22.3 Functional description

The following sections describe FREQME functional details.

22.3.1 Frequency Measurement mode

In Frequency Measurement mode (CTL_W[PULSE_MODE] = 0), FREQME counts the number of target clock cycles that occur during a specified number of cycles from a reference clock that has a known frequency. You can then calculate the target clock frequency based on the frequency of the reference clock, the number of reference clock cycles, and the number of target clock cycles (see Equation 1).

The frequency measurement circuit is based on two 31-bit counters—one clocked by the selected reference clock and one by the selected target clock. You can only read the target clock counter. FREQME synchronizes the clocks at the start and end of each count sequence during preparation for the next count sequence. A count sequence consists of incrementing the target clock counter for each target clock pulse that occurs during the time defined by $2^{CTRL\_W[REF\_SCALE]}$ periods of the reference clock.

After selecting reference and target clocks, you initiate a measurement cycle by writing 1 to CTRL_W[MEASURE_IN_PROGRESS]. You can then poll this same field, which automatically transitions to 0 when the measurement operation completes. Alternatively, the completion of the measurement operation can generate an interrupt.

The measurement cycle terminates when the reference counter equals the value $2^{CTRLSTAT[REFSCALE]}$. If CTRLSTAT[REFSCALE] = 0, the reference counter counts to one and stops. You can use this feature to measure the frequency of a fast target clock using a slow reference clock such as the 32 kHz clock without taking much time for the measurement to complete. The penalty is reduced accuracy in the measurement.

When the counting operation completes, the state of the target counter is loaded into CTRL_R[RESULT], and the CTRL_W[MEASURE_IN_PROGRESS] field is 0. You can then read the value and calculate the target frequency as follows, with the frequencies given in MHz.

$$F_{target} = (CTRL\_R[RESULT] - 2) \times F_{reference} \div 2^{CTRLSTAT[REFSCALE]}$$

Equation 1. Calculating the target clock frequency

22.3.1.1 Accuracy

The Frequency Measurement mode can measure the frequency of any on-chip (or off-chip) clock (referred to as the target clock) with a high degree of accuracy by using an on-chip clock of known frequency as a reference clock.

Uncertainty in the reference clock (for example a ±1 % accuracy of the clock) adds to the measurement error of the target clock. In general, though, this additional error is less than the uncertainty of the reference clock.

22.3.2 Pulse Measurement mode

When you write 1 to CTRL_W[PULSE_MODE] and write 1 to CTRL_W[MEASURE_IN_PROGRESS] to start a measurement cycle, FREQME counts target clock pulses while the reference clock is in a specific state (high or low), selected by writing to CTRL_W[PULSE_POL]. See the description of these fields for more details.

22.3.3 Interrupts

The following interrupts can optionally be generated at the completion of a frequency measurement cycle.

Table 110. Interrupts

Condition	Interrupt enable field	Interrupt status field
Result is ready to read	CTRL_W[RESULT_READY_INT_EN]	CTRLSTAT[RESULT_READY_INT_EN]
Result is greater than MAX[MAX_VALUE]	CTRL_W[GT_MAX_INT_EN]	CTRLSTAT[GT_MAX_INT_EN]
Result is less than MIN[MIN_VALUE]	CTRL_W[LT_MIN_INT_EN]	CTRLSTAT[LT_MIN_INT_EN]

22.4 External signals

Table 111. External signals

Signal	Description	Direction
REF_CLK_IN	Reference clock. On-chip clock with known frequency chosen as reference.	Input
TARGET_CLK_IN	Target clock. On- or off-chip clock to be measured by FREQME.	Input

22.5 Initialization

Initialize FREQME by performing the following steps:

1. Enable the clock that drives FREQME in the chip-level clock control register. This step enables the register interface and the peripheral function clock.
2. Clear the FREQME peripheral reset in the chip-level reset control register.
3. If measuring an external clock, use the Input/Output pin configuration registers to connect the FREQME target clock input (TARGET_CLK_IN) to an external pin.
4. Ensure that the reference and target clocks are enabled.
5. Select the reference and target clocks.

22.6 Memory map and register definition

This section includes the FREQME memory map and detailed descriptions of all registers.

22.6.1 FREQME register descriptions

22.6.1.1 FREQME memory map

FREQME base address: 4001_3140h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Control (in Read mode) (CTRL_R)	32	RO	0000_0000
0	Control (in Write mode) (CTRL_W)	32	See section	0000_0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
4	Control Status (CTRLSTAT)	32	See section	0000_0000
8	Minimum (MIN)	32	See section	0000_0000
C	Maximum (MAX)	32	See section	7FFF_FFFF

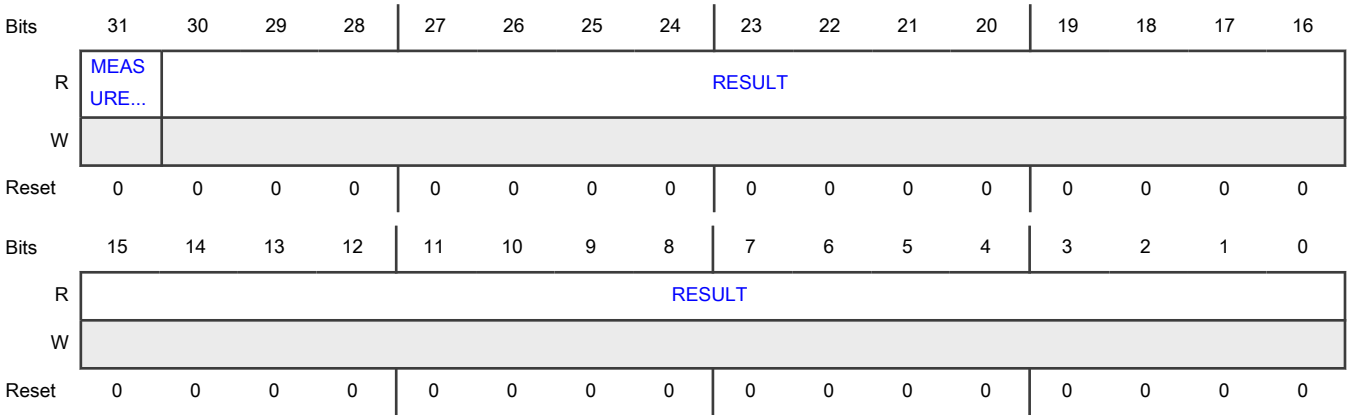
22.6.1.1.1 Control (in Read mode) (CTRL_R)

Contains different fields depending on whether you are performing a read or a write. Reading this register provides the measurement results and the status of the measurement cycle.

Offset

Register	Offset
CTRL_R	0h

Diagram



Fields

Field	Description
31 MEASURE_IN_PROGRESS	Measurement In Progress Indicates whether measurement is in progress or complete. If complete, you can read the result from RESULT. 0 - Complete 1 - In progress

Table continues on the next page...

Table continued from the previous page...

Field	Description
30-0 RESULT	Indicates the measurement result—either the target clock counter value (for Frequency Measurement mode) or pulse width measurement (for Pulse Width Measurement mode). This field is valid only when MEASURE_IN_PROGRESS = 0, In Continuous mode (CTRL_W[CONTINUOUS_MODE_EN] = 1), the value is the result from the most-recently completed measurement.

22.6.1.1.2 Control (in Write mode) (CTRL_W)

Writes to this register control and configure the measurement cycle.

Offset

Register	Offset
CTRL_W	0h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	MEASURE...	CONTINU...	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																
W	0	RESULT_...	GT_MAX_...	LT_MIN_...	0		PULSE_P...	PULSE_M...	0			REF_SCALE				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31 MEASURE_IN_PROGRESS	Measurement In Progress Initiates a frequency measurement cycle or terminates a measurement cycle that is in progress. Writing 1 to this field initiates a frequency or pulse width measurement process. Hardware automatically writes 0 to the MEASURE_IN_PROGRESS field when the measurement cycle completes. If there is an active measurement in progress, a new measurement starts. Writing 0 to this field forces the termination of any measurement cycle currently in progress and resets CTRL_R[RESULT] or just resets CTRL_R[RESULT] if idle. 0 - Terminates measurement

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Initiates measurement
30 CONTINUOUS_MODE_EN	Continuous Mode Enable When you write 1 to both MEASURE_IN_PROGRESS and this field, measurement is performed continuously. The result for the most-recently completed measurement is available in CTRL_R[RESULT]. When the result is out of range (that is, CTRLSTAT[GT_MAX_STAT] = 1 or CTRLSTAT[LT_MIN_STAT] = 1), this field automatically transitions to 0 and the continuous measurement is suspended. After clearing both status bits, you must write 1 to both MEASURE_IN_PROGRESS and this field restart measurement in continuous mode. 0 - Disable 1 - Enable
29-15 —	Reserved
14 RESULT_READY_INT_EN	Result Ready Interrupt Enable Generates interrupt when a measurement completes and the result is ready (CTRLSTAT[RESULT_READY_STAT] = 1). 0 - Disable 1 - Enable
13 GT_MAX_INT_EN	Greater Than Maximum Interrupt Enable Generates an interrupt when the result is greater than MAX[MAX_VALUE] (CTRLSTAT[GT_MAX_STAT] = 1). 0 - Disable 1 - Enable
12 LT_MIN_INT_EN	Less Than Minimum Interrupt Enable Generates an interrupt when the result is less than MIN[MIN_VALUE] (CTRLSTAT[LT_MIN_STAT] = 1). 0 - Disable 1 - Enable
11-10 —	Reserved
9 PULSE_POL	Pulse Polarity Specifies whether a pulse width (high period or low period) of the reference clock is measured in the Pulse Width Measurement mode. **NOTE** PULSE_POL is valid only in the Pulse Width Measurement mode.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	• A high period measurement is triggered by the rising edge on the reference clock input. • A low period measurement is triggered by the falling edge on the reference clock input. 0 - High period 1 - Low period
8 PULSE_MODE	Pulse Width Measurement Mode Select Selects the measurement mode—either Frequency Measurement mode or Pulse Width Measurement mode. In Frequency Measurement mode, measurement begins at the rising edge of the selected reference clock and continues until a count of 2^{REF_SCALE} reference clock pulses is reached. In Pulse Width Measurement mode, the counter starts incrementing when the selected trigger edge (rising edge for high period measurement or falling edge for low period) occurs, and stops at the next edge (falling edge for a high period measurement or rising edge for a low period measurement). Select high or low period measurement by writing the desired value to PULSE_POL. 0 - Frequency Measurement mode 1 - Pulse Width Measurement mode
7-5 —	Reserved
4-0 REF_SCALE	Reference Clock Scaling Factor Specifies the reference clock scaling factor in Frequency Measurement mode. The reference count cycle is 2^{REF_SCALE}. A higher number provides better accuracy but consumes more processing time. This field is valid only in Frequency Measurement mode.

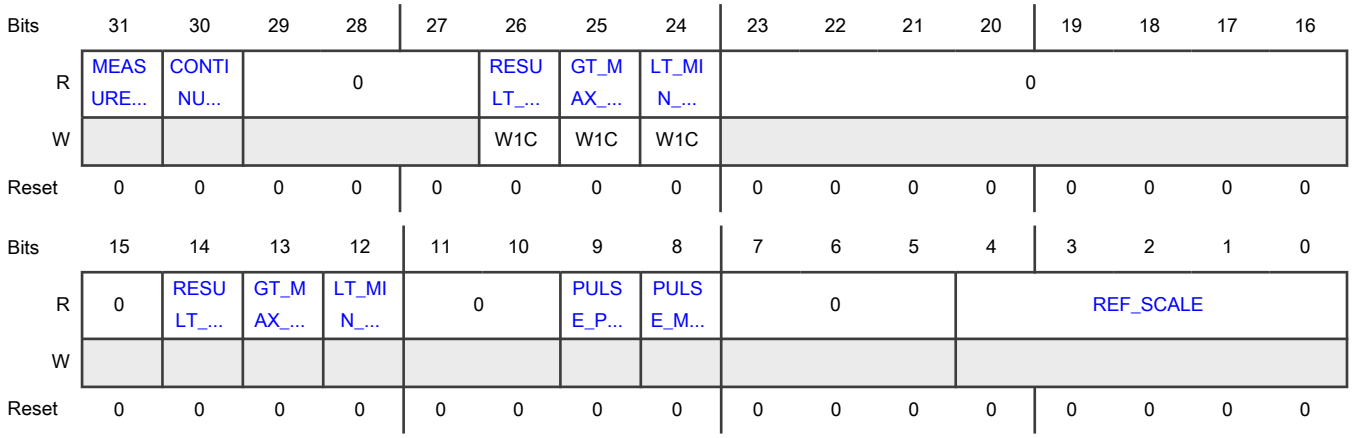
22.6.1.1.3 Control Status (CTRLSTAT)

Contains the current CTRL_W register configuration fields.

Offset

Register	Offset
CTRLSTAT	4h

Diagram



Fields

Field	Description
31 MEASURE_IN_PROGRESS	Measurement in Progress Status Indicates the current CTRL_W[MEASURE_IN_PROGRESS] value. 0 - Not in progress 1 - In progress
30 CONTINUOUS_MODE_EN	Continuous Mode Enable Status Indicates the current CTRL_W[CONTINUOUS_MODE_EN] value. 0 - Disabled 1 - Enabled
29-27 —	Reserved
26 RESULT_READY_STAT	Result Ready Status Indicates that a measurement is complete and CTRL_R[RESULT] is ready to read. Write 1 to this field to clear. 0 - Not complete 1 - Complete
25 GT_MAX_STAT	Greater Than Maximum Result Status Indicates that a measurement is complete and the result is greater than the maximum expected value (CTRL_R[RESULT] > MAX[MAX_VALUE]). Write 1 to this field to clear. 0 - Less than MAX[MAX_VALUE] 1 - Greater than MAX[MAX_VALUE]
24	Less Than Minimum Results Status

Table continues on the next page...

Table continued from the previous page...

Field	Description
LT_MIN_STAT	Indicates that a measurement is complete and the result is less than the minimum expected value (CTRL_R[RESULT] < MIN[MIN_VALUE]). Write 1 to this field to clear. 0 - Greater than MIN[MIN_VALUE] 1 - Less than MIN[MIN_VALUE]
23-15 —	Reserved
14 RESULT_READY_INT_EN	Result Ready Interrupt Enable Indicates the current CTRL_W[RESULT_READY_INT_EN] value. 0 - Disabled 1 - Enabled
13 GT_MAX_INT_EN	Greater Than Maximum Interrupt Enable Indicates the current CTRL_W[GT_MAX_INT_EN] value. 0 - Disabled 1 - Enabled
12 LT_MIN_INT_EN	Less Than Minimum Interrupt Enable Indicates the current CTRL_W[LT_MIN_INT_EN] value. 0 - Disabled 1 - Enabled
11-10 —	Reserved
9 PULSE_POL	Pulse Polarity Indicates the current CTRL_W[PULSE_POL] value. 0 - High period 1 - Low period
8 PULSE_MODE	Pulse Mode Indicates the current CTRL_W[PULSE_MODE] value. 0 - Frequency Measurement mode 1 - Pulse Width Measurement mode
7-5 —	Reserved
4-0	Reference Scale

Table continues on the next page...

Table continued from the previous page...

Field	Description
REF_SCALE	Indicates the current CTRL_W[REF_SCALE] value.

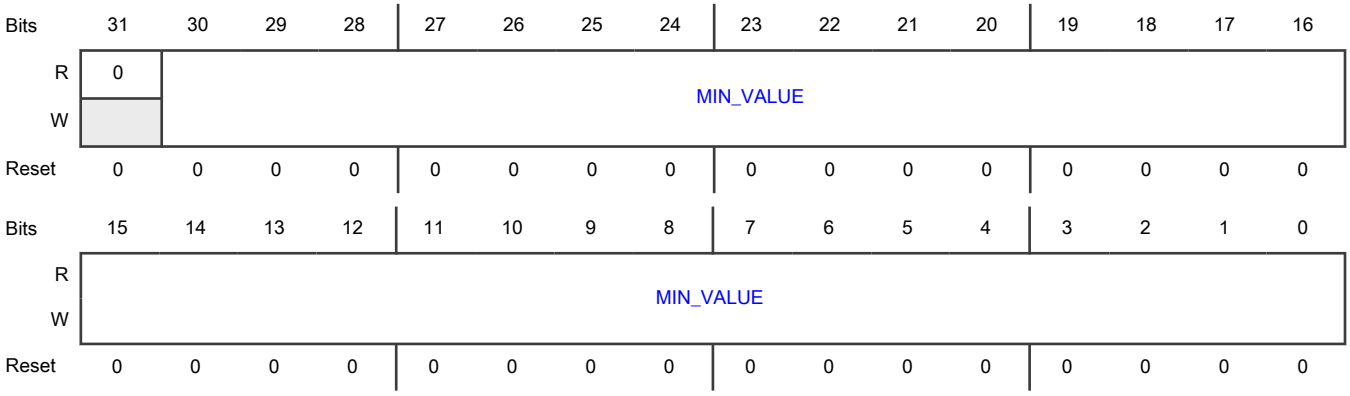
22.6.1.1.4 Minimum (MIN)

Specifies the minimum expected value for the measurement result.

Offset

Register	Offset
MIN	8h

Diagram



Fields

Field	Description
31 —	Reserved
30-0 MIN_VALUE	Minimum Value Minimum expected value for the measurement result.

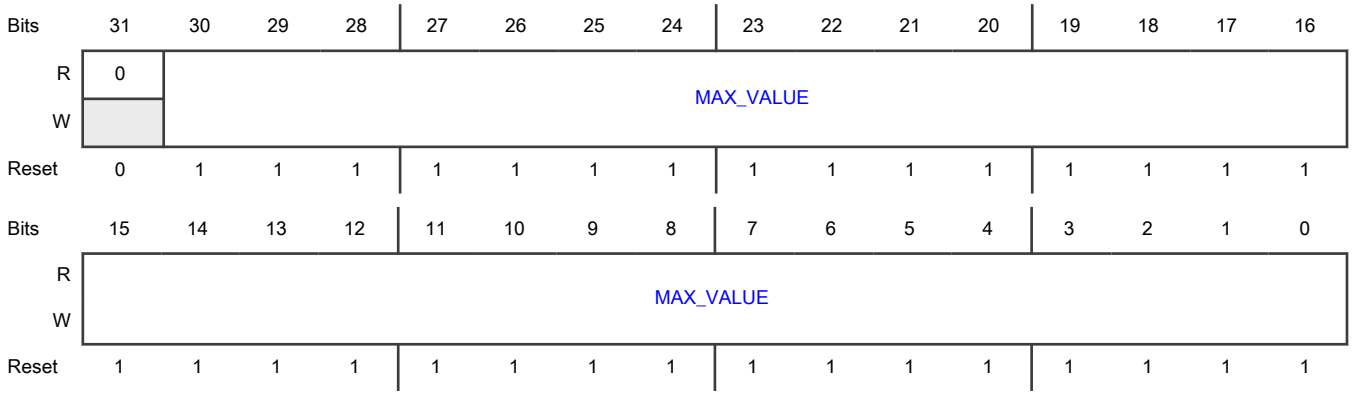
22.6.1.1.5 Maximum (MAX)

Specifies the maximum expected value for the measurement result.

Offset

Register	Offset
MAX	Ch

Diagram



Fields

Field	Description
31 —	Reserved
30-0 MAX_VALUE	Maximum Value Maximum expected value for the measurement result.

Chapter 23
General Purpose Input/Output (GPIO)

23.1 Chip-specific GPIO information

Table 112. Reference links to related information

Topic	Related module	Reference
Full description	GPIO	GPIO
System memory map		System memory map
Clocking	SYSCON	Clock generation
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

This table describes the GPIO signals implemented on the chip.

Signal	I/O	Description
PIO0_0 through PIO0_31	IO	HS GPIO port 0
PIO1_0 through PIO1_31	IO	HS GPIO port 1
PIO2_0 through PIO2_31	IO	HS GPIO port 2
PIO3_0 through PIO3_6	IO	HS GPIO port 3

23.1.1 Module instances

The chip includes one instance of GPIO module- HS GPIO (Ports 0-3).

23.1.2 Initialization

Initial configuration of the GPIO can be accomplished as follows:

- For the GPIO ports, enable clocks in the AHBCLKCTRL0 register. For the secure GPIO port, enable the clock in the AHBCLKCTRL2 register (See CLKCTL).
- Use the IOCON registers to connect the (inputs/outputs) to external pins.

23.2 Overview

This section provides an introduction to the GPIO module. Each GPIO port supports up to 32 pins. The GPIO port registers configure each GPIO pin as input or output and read the state of each pin if the pin is configured as input or set the state of each pin if the pin is configured as output. The GPIOs can be used as external interrupts together with the pin interrupt and group interrupt blocks.

A single additional GPIO port, called Secure GPIO, can be optionally secured by software separately from the non-secure GPIO ports. The Secure GPIO has all of the features of non-secure GPIO and has its own instance of the GPIO interrupts.

23.2.1 Features

Each GPIO pin includes the following features:

- It can be configured as input or output by software.

  - Direction (input/output) can be set and cleared individually.
- It can be sensed and set individually by GPIO registers.
- Default is tri-state, some can be set as pull-up or pull-down.
- It can optionally contribute to one of two GPIO interrupts, with selection of polarity, level or edge detection.
  - Enable for contributing to interrupt A (contributing pin states ORed together)
  - Enable for contributing to interrupt B (contributing pin states ORed together)
  - Status flag for each pin interrupt A
  - Status flag for each pin interrupt B
  - Ability to clear status flag for each pin interrupt A
  - Ability to clear status flag for each pin interrupt B
  - Interrupts are disabled at reset.
- Secure GPIO pins overlay non-secure port 0 pins and are intended to be secured in the application if used.

23.3 Application information

The following lists some recommended uses for using the GPIO port registers:

- For initial setup after Reset or re-initialization, write the PIN registers.
- To change the state of one pin, write a Byte Pin or Word Pin register.
- To change the state of multiple pins at a time, write the SET and/or CLR registers.
- To change the state of multiple pins in a tightly controlled environment like a software state machine, consider using the NOT register. This can require less write operations than SET and CLR.
- To read the state of one pin, read a Byte Pin or Word Pin register.
- To make a decision based on multiple pins, read and mask a PIN register.

23.4 Functional description

The following sections describe functional details of the GPIO module.

23.4.1 Reading pin state

Software can read the state of all GPIO pins except those selected for analog input or output in the I/O Configuration logic. A pin does not have to be selected for GPIO in I/O Configuration in order to read its state. The Input Enable bit of the pad must be set in I/O Configuration otherwise its value is 0. There are four ways to read pin state:

- The state of a single pin can be read with 7 high-order zeros from a Byte Pin register.
- The state of a single pin can be read in all bits of a byte, halfword, or word from a Word Pin register.
- The state of multiple pins in a port can be read as a byte, halfword, or word from a PIN register.
- The state of a selected subset of the pins in a port can be read from a Masked Port (MPIN) register. Pins having a 1 in the port's Mask register will read as 0 from its MPIN register.

23.4.2 GPIO output

Each GPIO pin has an output bit in the GPIO. These output bits are the targets of write operations to the pins. Two conditions must be met in order for a pin's output bit to be driven onto the pin:

1. The pin must be selected for GPIO operation via IOPCTL (this is the default), and
2. the pin must be selected for output by a 1 in its port's DIR register.

If either or both of these conditions is (are) not met, writing to the pin has no effect.

There are seven ways to change GPIO output bits:

- Writing to a Byte Pin register loads the output bit from the least significant bit.
- Writing to a Word Pin register loads the output bit with the OR of all of the bits written. (This feature follows the definition of truth of a multi-bit value in programming languages.)
- Writing to a port's PIN register loads the output bits of all the pins written to.
- Writing to a port's MPIN register loads the output bits of pins identified by zeros in corresponding positions of the port's MASK register.
- Writing ones to a port's SET register sets output bits.
- Writing ones to a port's CLR register clears output bits.
- Writing ones to a port's NOT register toggles/complements/inverts output bits.

The state of a port's output bits can be read from its SET register. Reading any of the registers returns the state of pins, regardless of their direction or alternate functions.

23.4.3 Masked I/O

A port's MASK register defines which of its pins should be accessible in its MPIN register. Zeros in MASK enable the corresponding pins to be read from and written to MPIN. Ones in MASK force a pin to read as 0 and its output bit to be unaffected by writes to MPIN. When a port's MASK register contains all zeros, its PIN and MPIN registers operate identically for reading and writing.

Applications in which interrupts can result in Masked GPIO operation, or in task switching among tasks that do Masked GPIO operation, must treat code that uses the Mask register as a protected/restricted region. This can be done by interrupt disabling or by using a semaphore.

The simpler way to protect a block of code that uses a MASK register is to disable interrupts before setting the MASK register, and re-enable them after the last operation that uses the MPIN or MASK register.

More efficiently, software can dedicate a semaphore to the MASK registers, and set/capture the semaphore controlling exclusive use of the MASK registers before setting the MASK registers, and release the semaphore after the last operation that uses the MPIN or MASK registers.

23.4.4 GPIO interrupts

The GPIO function includes two interrupts, A and B, that can be used with any GPIO pin. Each interrupt has enables for each pin (INTENAn and INTENBn registers, one for each port), and a status for each pin (INTSTATAn and INTSTATBn registers, one for each port). Polarity and edge versus level trigger can also be set for each pin (INTPOLn and INTEDGn registers, one for each port).

Any GPIO pin whose configured condition is satisfied, and is enabled for one of the two GPIO interrupts, will send an interrupt to the CM33 and the Fusion DSP interrupt controllers. All contributing conditions need to be cleared to cause the interrupt to be deasserted.

The single secure GPIO port also supports it own GPIO interrupts.

23.5 Memory map and register definition

This section includes the GPIO memory map and detailed descriptions of all registers.

23.5.1 SECGPIO_HS General Purpose I/O (GPIO) register descriptions

GPIO port addresses can be read and written as bytes, halfwords, or words. Registers within each kind of GPIO are functionally the same.

NOTE

Supported pins depends on the specific device and package.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

23.5.1.1 GPIO memory map

SECGPIO_HS base address: 400A_8000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0 - 1F	Byte pin registers for all port GPIO pins (B0_0 - B0_31)	8	See section	See section
1000 - 107C	Word pin registers for all port GPIO pins (W0_0 - W0_31)	32	RW	0000_0000
2000	Port direction (DIR0)	32	WO	0000_0000
2080	Port mask (MASK0)	32	RW	0000_0000
2100	Port pin (PIN0)	32	RW	0000_0000
2180	Masked Port Pin (MPIN0)	32	RW	0000_0000
2200	Port set (SET0)	32	RW	0000_0000
2280	Port clear (CLR0)	32	See section	See section
2300	Port toggle (NOT0)	32	WO	See section
2380	Port direction set (DIRSET0)	32	WO	See section
2400	Port direction clear (DIRCLR0)	32	See section	See section
2480	Port direction toggle (DIRNOT0)	32	WO	See section
2500	Interrupt A enable control (INTENA0)	32	RW	0000_0000
2580	Interrupt B enable control (INTENB0)	32	RW	0000_0000
2600	Interupt polarity control (INTPOL0)	32	RW	0000_0000
2680	Interrupt edge select (INTEDG0)	32	RW	0000_0000
2700	Interrupt status for interrupt A (INTSTATA0)	32	See section	0000_0000
2780	Interrupt status for interrupt B (INTSTATB0)	32	See section	0000_0000

23.5.1.1.1 Byte pin registers for all port GPIO pins (B0_0 - B0_31)

Each GPIO pin has a byte register. Software reads and writes bytes to access individual pins. It can read or write halfwords to sense or set the state of two pins, and read or write words to sense or set the state of four pins.

NOTE

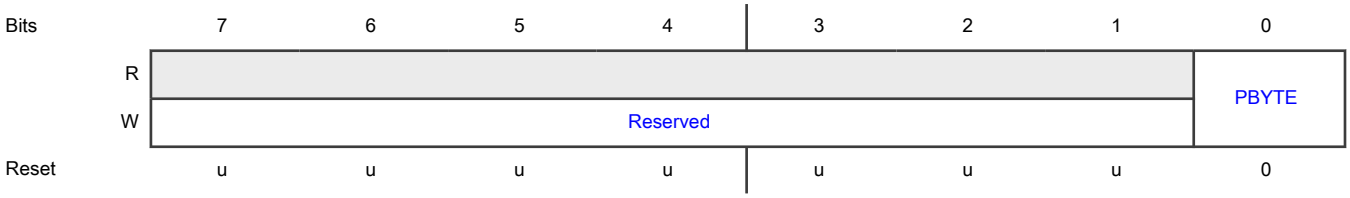
The data read after reset depends on the state of the pin, which in turn may depend on an external source.

Offset

For b = 0 to 31:

Register	Offset
B0_b	0h + (b × 1h)

Diagram



Fields

Field	Description
7-1 —	Reserved (0 on read, ignored on write)
0 PBYTE	Port Byte Read- specifies the state of GPIO pins(PIOa_b), regardless of direction, masking, or alternate function. Pins configured as analog I/O always read as 0. Write- loads the pin's output bit.

23.5.1.1.2 Word pin registers for all port GPIO pins (W0_0 - W0_31)

Each GPIO pin has a word register in this address range. Any byte, halfword, or word read in this range will be all zeros if the pin is low or all ones if the pin is high, regardless of direction, masking, or alternate function. Pins configured as analog I/O always read as zeros. Any write will clear the pin’s output bit if the value written is all zeros, else it will set the pin’s output bit.

NOTE

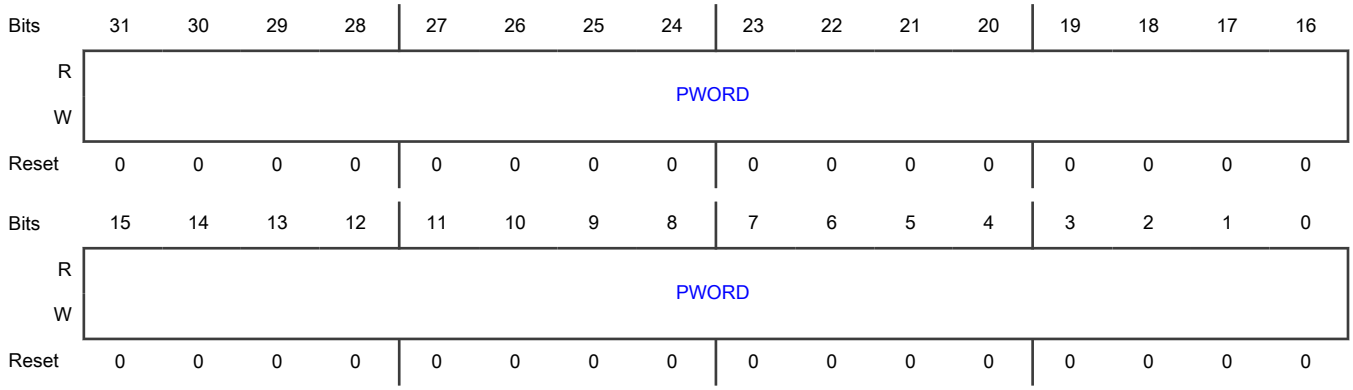
The data read after reset depends on the state of the pin, which in turn may depend on an external source.

Offset

For b = 0 to 31:

Register	Offset
W0_b	1000h + (b × 4h)

Diagram



Fields

Field	Description
31-0 PWORD	PWORD Read 0- pin PIOa_b is LOW. Write 0- clear output bit. Read 0xFFFF FFFF- pin PIOa_b is HIGH. Write any value 0x0000 0001 to 0xFFFF FFFF- sets output bit. Only 0 or 0xFFFF FFFF can be read. Writing any value other than 0 will set the output bit.

23.5.1.1.3 Port direction (DIR0)

The DIR registers configure each pin in a GPIO port as input or output. Each GPIO port has one direction register for configuring the port pins as inputs or outputs.

Offset

Register	Offset
DIR0	2000h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	DIRP31	DIRP30	DIRP29	DIRP28	DIRP27	DIRP26	DIRP25	DIRP24	DIRP23	DIRP22	DIRP21	DIRP20	DIRP19	DIRP18	DIRP17	DIRP16
Reset	0	0	0	0	0	0	0	0	0	0	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	DIRP15	DIRP14	DIRP13	DIRP12	DIRP11	DIRP10	DIRP9	DIRP8	DIRP7	DIRP6	DIRP5	DIRP4	DIRP3	DIRP2	DIRP1	DIRP0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 DIRPn	Selects pin direction for pin PIOa_b. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0 - Input 1 - Output

23.5.1.1.4 Port mask (MASK0)

It defines which port pins are accessible in its MPIN register. Zeros in these registers enable reading and writing; ones disable writing and result in zeros in corresponding positions when reading.

Offset

Register	Offset
MASK0	2080h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MASKP31	MASKP30	MASKP29	MASKP28	MASKP27	MASKP26	MASKP25	MASKP24	MASKP23	MASKP22	MASKP21	MASKP20	MASKP19	MASKP18	MASKP17	MASKP16
Reset	0	0	0	0	0	0	0	0	0	0	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	MASKP15	MASKP14	MASKP13	MASKP12	MASKP11	MASKP10	MASKP9	MASKP8	MASKP7	MASKP6	MASKP5	MASKP4	MASKP3	MASKP2	MASKP1	MASKP0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 MASKPn	Port Mask Controls which bits corresponding to PIOa_b are active in the MPIN register. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0 - Read MPIN: pin state; write MPIN: load output bit 1 - Read MPIN: 0; write MPIN: output bit not affected

23.5.1.1.5 Port pin (PIN0)

Reading these registers returns the current state of the pins, regardless of direction, masking, or alternate digital functions. Pins configured as analog I/O always read as 0s. Writing these registers loads the output bits of the pins written to, regardless of the Mask register.

NOTE

The data read after reset depends on the state of the pin, which in turn may depend on an external source.

Offset

Register	Offset
PIN0	2100h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PORT 31	PORT 30	PORT 29	PORT 28	PORT 27	PORT 26	PORT 25	PORT 24	PORT 23	PORT 22	PORT 21	PORT 20	PORT 19	PORT 18	PORT 17	PORT 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 W	PORT 15	PORT 14	PORT 13	PORT 12	PORT 11	PORT 10	PORT 9	PORT 8	PORT 7	PORT 6	PORT 5	PORT 4	PORT 3	PORT 2	PORT 1	PORT 0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 PORTn	Port pins Reads pin states or loads output bits. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0 - Read- pin is low; Write- clear output bit 1 - Read- pin is high; Write- set output bit

23.5.1.1.6 Masked Port Pin (MPIN0)

These registers are similar to the PIN registers, except that the value read is masked by ANDing with the inverted contents of the corresponding MASK register, and writing to one of these registers only affects output register bits that are enabled by zeros in the corresponding MASK register.

Offset

Register	Offset
MPIN0	2180h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MPORTP31	MPORTP30	MPORTP29	MPORTP28	MPORTP27	MPORTP26	MPORTP25	MPORTP24	MPORTP23	MPORTP22	MPORTP21	MPORTP20	MPORTP19	MPORTP18	MPORTP17	MPORTP16
Reset	0	0	0	0	0	0	0	0	0	0	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	MPORTP15	MPORTP14	MPORTP13	MPORTP12	MPORTP11	MPORTP10	MPORTP9	MPORTP8	MPORTP7	MPORTP6	MPORTP5	MPORTP4	MPORTP3	MPORTP2	MPORTP1	MPORTP0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 MPORTPn	Mask bits for port pins Controls which bits corresponding to PIOa_b are active in the MPIN register. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. **NOTE** The data read after reset depends on the state of the pin, which in turn may depend on an external source. 0 - Read- pin is LOW and/or the corresponding bit in the MASK register is 1; write- clear output bit if the corresponding bit in the MASK register is 0 1 - Read- pin is HIGH and the corresponding bit in the MASK register is 0; write- set output bit if the corresponding bit in the MASK register is 0

23.5.1.1.7 Port set (SET0)

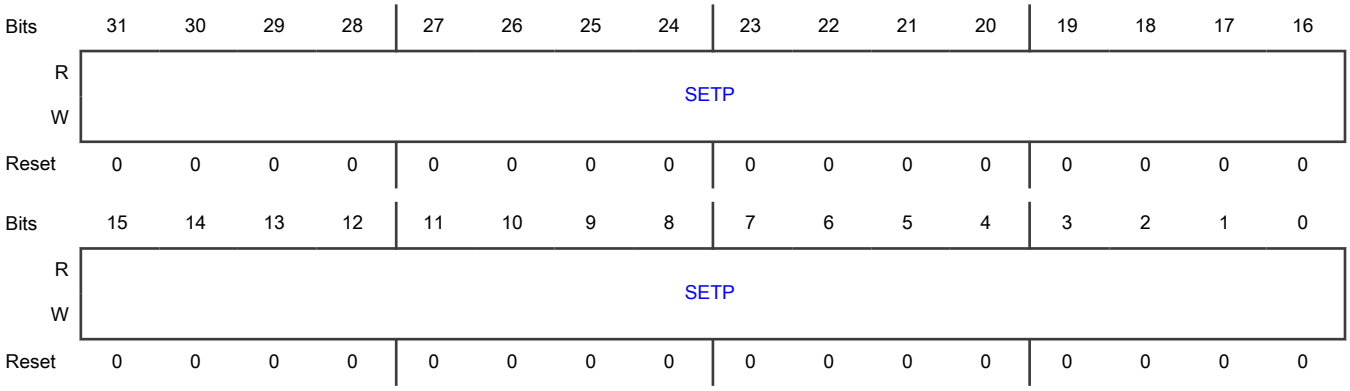
Output bits can be set by writing ones to these registers, regardless of MASK registers. Reading from these register returns the port's output bits, regardless of pin directions.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

Offset

Register	Offset
SET0	2200h

Diagram



Fields

Field	Description
31-0 SETP	Read or set output bits Sets output bits for port pins 0000_0000_0000_0000_0000_0000_0000_0000 - Read- output bit; write- no operation 0000_0000_0000_0000_0000_0000_0000_0001 - Read- output bit; write- set output bit

23.5.1.1.8 Port clear (CLR0)

Output bits can be cleared by writing ones to these write-only registers, regardless of MASK registers.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

Offset

Register	Offset
CLR0	2280h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CLRP 31	CLRP 30	CLRP 29	CLRP 28	CLRP 27	CLRP 26	CLRP 25	CLRP 24	CLRP 23	CLRP 22	CLRP 21	CLRP 20	CLRP 19	CLRP 18	CLRP 17	CLRP 16
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
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	CLRP 15	CLRP 14	CLRP 13	CLRP 12	CLRP 11	CLRP 10	CLRP 9	CLRP 8	CLRP 7	CLRP 6	CLRP 5	CLRP 4	CLRP 3	CLRP 2	CLRP 1	CLRP 0
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-0 CLRPn	Clear output bits Clear output bits for port pins 0 - No operation 1 - Clears output bit

23.5.1.1.9 Port toggle (NOT0)

Output bits can be toggled by writing ones to these write-only registers.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

Offset

Register	Offset
NOT0	2300h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	NOTP 31	NOTP 30	NOTP 29	NOTP 28	NOTP 27	NOTP 26	NOTP 25	NOTP 24	NOTP 23	NOTP 22	NOTP 21	NOTP 20	NOTP 19	NOTP 18	NOTP 17	NOTP 16
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																
W	NOTP 15	NOTP 14	NOTP 13	NOTP 12	NOTP 11	NOTP 10	NOTP 9	NOTP 8	NOTP 7	NOTP 6	NOTP 5	NOTP 4	NOTP 3	NOTP 2	NOTP 1	NOTP 0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-0 NOTPn	Toggle output bits Toggle direction bits for port pins. 0 - No operation 1 - Toggle output bit

23.5.1.1.10 Port direction set (DIRSET0)

Direction bits can be set by writing ones to these registers.

Offset

Register	Offset
DIRSET0	2380h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...
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																
W	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP9	DIRSETP8	DIRSETP7	DIRSETP6	DIRSETP5	DIRSETP4	DIRSETP3	DIRSETP2	DIRSETP1	DIRSETP0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-0 DIRSETPn	Direction set bits for Port pins Set pin direction for port pins. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0 - No operation 1 - Sets direction bit

23.5.1.1.11 Port direction clear (DIRCLR0)

Direction bits can be cleared by writing ones to these write-only registers.

Offset

Register	Offset
DIRCLR0	2400h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
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	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP9	DIRCLRP8	DIRCLRP7	DIRCLRP6	DIRCLRP5	DIRCLRP4	DIRCLRP3	DIRCLRP2	DIRCLRP1	DIRCLRP0
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-0 DIRCLRPn	Clear direction bits. Clear pin direction bits for Port pins. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0 - No operation 1 - Clears direction bits

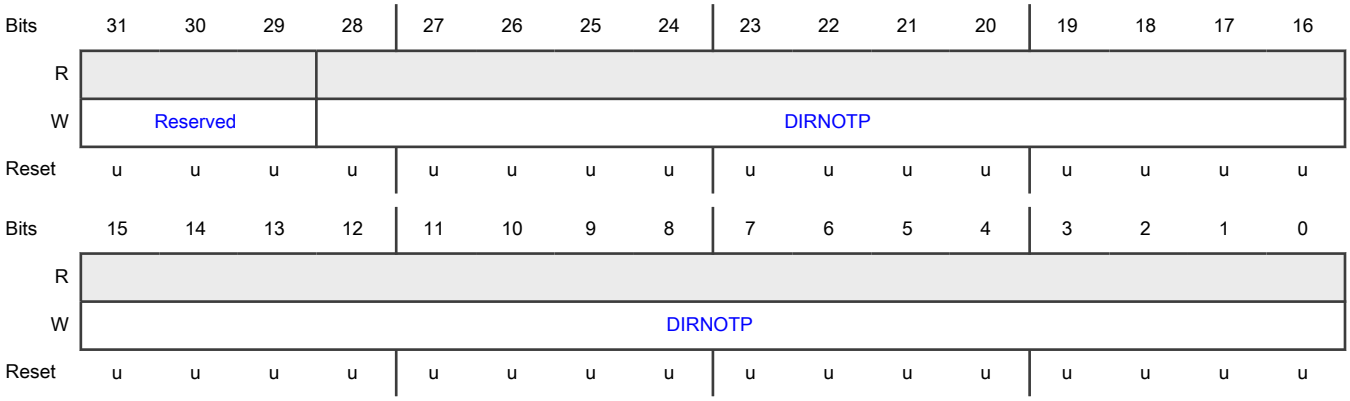
23.5.1.1.12 Port direction toggle (DIRNOT0)

Direction bits can be toggled by writing ones to these write-only registers.

Offset

Register	Offset
DIRNOT0	2480h

Diagram



Fields

Field	Description
31-29 —	Reserved
28-0 DIRNOTP	Toggle direction bits. Toggle direction bits for each pin. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0_0000_0000_0000_0000_0000_0000_0000 - No operation 0_0000_0000_0000_0000_0000_0000_0001 - Toggles direction bit

23.5.1.1.13 Interrupt A enable control (INTENA0)

These registers allow the enabling or disabling of individual pins contributions to GPIO interrupt A.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

Offset

Register	Offset
INTENA0	2500h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INT_EN31	INT_EN30	INT_EN29	INT_EN28	INT_EN27	INT_EN26	INT_EN25	INT_EN24	INT_EN23	INT_EN22	INT_EN21	INT_EN20	INT_EN19	INT_EN18	INT_EN17	INT_EN16
Reset	0	0	0	0	0	0	0	0	0	0	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	INT_EN15	INT_EN14	INT_EN13	INT_EN12	INT_EN11	INT_EN10	INT_EN9	INT_EN8	INT_EN7	INT_EN6	INT_EN5	INT_EN4	INT_EN3	INT_EN2	INT_EN1	INT_EN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 INT_ENn	Interrupt A enable bits. Interrupt enable control for each pin. 0 - Pin does not contribute to GPIO interrupt A 1 - Pin contributes to GPIO interrupt A

23.5.1.1.14 Interrupt B enable control (INTENB0)

These registers allow the enabling or disabling of individual pins contributions to GPIO interrupt B.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

Offset

Register	Offset
INTENB0	2580h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INT_EN31	INT_EN30	INT_EN29	INT_EN28	INT_EN27	INT_EN26	INT_EN25	INT_EN24	INT_EN23	INT_EN22	INT_EN21	INT_EN20	INT_EN19	INT_EN18	INT_EN17	INT_EN16
Reset	0	0	0	0	0	0	0	0	0	0	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	INT_EN15	INT_EN14	INT_EN13	INT_EN12	INT_EN11	INT_EN10	INT_EN9	INT_EN8	INT_EN7	INT_EN6	INT_EN5	INT_EN4	INT_EN3	INT_EN2	INT_EN1	INT_EN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 INT_ENn	Interrupt B enable bits. Interrupt enable control for each pin. 0 - Pin does not contribute to GPIO interrupt B 1 - Pin contributes to GPIO interrupt B

23.5.1.1.15 Interupt polarity control (INTPOL0)

These registers select the polarity of GPIO interrupts.

Offset

Register	Offset
INTPOL0	2600h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...
Reset	0	0	0	0	0	0	0	0	0	0	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	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL9	POL_CTL8	POL_CTL7	POL_CTL6	POL_CTL5	POL_CTL4	POL_CTL3	POL_CTL2	POL_CTL1	POL_CTL0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 POL_CTLn	Polarity control for each pin Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0 - High level or rising edge triggered 1 - Low level or falling edge triggered

23.5.1.1.16 Interrupt edge select (INTEDG0)

These registers allow GPIO interrupts to be selected as level or edge triggered.

Offset

Register	Offset
INTEDG0	2680h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE
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	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE	EDGE
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	Description
31-0 EDGEn	Edge or level mode select bits. These bits allow GPIO interrupts to be selected as level or edge triggered. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0 - Level mode 1 - Edge mode

23.5.1.1.17 Interrupt status for interrupt A (INTSTATA0)

These registers report the status of interrupt group A for each GPIO port.

Offset

Register	Offset
INTSTATA0	2700h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	STATUS															
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	STATUS															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 STATUS	Interrupt status. Interrupt group A status bits Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. Read: 0 - not pending; Read: 1 - pending; Write: 0 - no action. Write: 1 - clear status bit.

23.5.1.1.18 Interrupt status for interrupt B (INTSTATB0)

These registers report the status of interrupt group B for each GPIO port.

Offset

Register	Offset
INTSTATB0	2780h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	STATUS															
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	STATUS															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 STATUS	Interrupt status Interrupt group B status bits. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. Read 0 - not pending; Read 1 - pending; Write 0 - no action. Write 1 - clear status bit.

23.5.2 GPIO_HS General Purpose I/O (GPIO) register descriptions

GPIO port addresses can be read and written as bytes, halfwords, or words. Registers within each kind of GPIO are functionally the same.

NOTE

Supported pins depends on the specific device and package.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

23.5.2.1 GPIO memory map

GPIO_HS base address: 4008_C000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0 - 66	Byte pin registers for all port GPIO pins (B0_0 - B3_6)	8	See section	See section
1000 - 11FC	Word pin registers for all port GPIO pins (W0_0 - W3_31)	32	RW	0000_0000
2000 - 200C	Port direction (DIR0 - DIR3)	32	WO	0000_0000
2080 - 208C	Port mask (MASK0 - MASK3)	32	RW	0000_0000
2100 - 210C	Port pin (PIN0 - PIN3)	32	RW	0000_0000
2180 - 218C	Masked Port Pin (MPIN0 - MPIN3)	32	RW	0000_0000
2200 - 220C	Port set (SET0 - SET3)	32	RW	0000_0000
2280 - 228C	Port clear (CLR0 - CLR3)	32	See section	See section
2300 - 230C	Port toggle (NOT0 - NOT3)	32	WO	See section
2380 - 238C	Port direction set (DIRSET0 - DIRSET3)	32	WO	See section
2400 - 240C	Port direction clear (DIRCLR0 - DIRCLR3)	32	See section	See section

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
2480 - 248C	Port direction toggle (DIRNOT0 - DIRNOT3)	32	WO	See section
2500 - 250C	Interrupt A enable control (INTENA0 - INTENA3)	32	RW	0000_0000
2580 - 258C	Interrupt B enable control (INTENB0 - INTENB3)	32	RW	0000_0000
2600 - 260C	Interupt polarity control (INTPOL0 - INTPOL3)	32	RW	0000_0000
2680 - 268C	Interrupt edge select (INTEDG0 - INTEDG3)	32	RW	0000_0000
2700 - 270C	Interrupt status for interrupt A (INTSTATA0 - INTSTATA3)	32	See section	0000_0000
2780 - 278C	Interrupt status for interrupt B (INTSTATB0 - INTSTATB3)	32	See section	0000_0000

23.5.2.1.1 Byte pin registers for all port GPIO pins (B0_0 - B3_6)

Each GPIO pin has a byte register. Software reads and writes bytes to access individual pins. It can read or write halfwords to sense or set the state of two pins, and read or write words to sense or set the state of four pins.

NOTE

The data read after reset depends on the state of the pin, which in turn may depend on an external source.

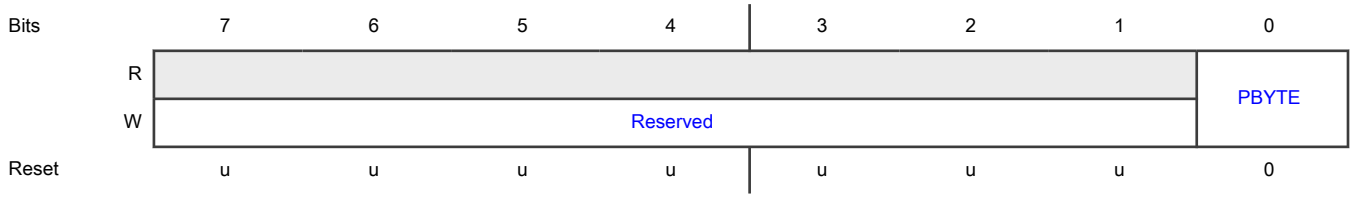
Offset

Registers in this array exist only for the following combinations of index values.

Index *a*	Index *b*
0–2	0–31
3	0–6

Register	Offset
Ba_b	0h + (a × 20h) + (b × 1h)

Diagram



Fields

Field	Description
7-1 —	Reserved (0 on read, ignored on write)
0 PBYTE	Port Byte Read- specifies the state of GPIO pins(PIOa_b), regardless of direction, masking, or alternate function. Pins configured as analog I/O always read as 0. Write- loads the pin's output bit.

23.5.2.1.2 Word pin registers for all port GPIO pins (W0_0 - W3_31)

Each GPIO pin has a word register in this address range. Any byte, halfword, or word read in this range will be all zeros if the pin is low or all ones if the pin is high, regardless of direction, masking, or alternate function. Pins configured as analog I/O always read as zeros. Any write will clear the pin’s output bit if the value written is all zeros, else it will set the pin’s output bit.

NOTE

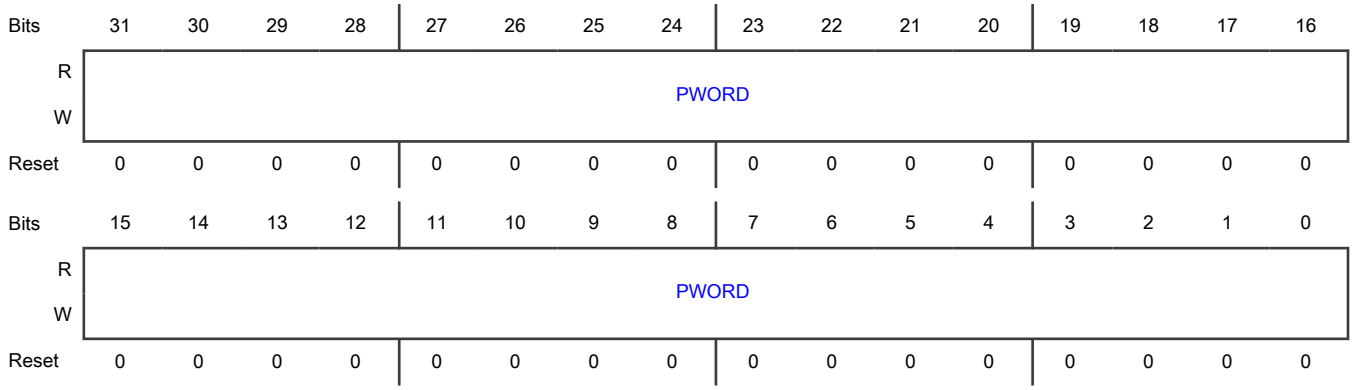
The data read after reset depends on the state of the pin, which in turn may depend on an external source.

Offset

For a = 0 to 3; b = 0 to 31:

Register	Offset
Wa_b	1000h + (a × 80h) + (b × 4h)

Diagram



Fields

Field	Description
31-0 PWORD	PWORD Read 0- pin PIOa_b is LOW. Write 0- clear output bit. Read 0xFFFF FFFF- pin PIOa_b is HIGH. Write any value 0x0000 0001 to 0xFFFF FFFF- sets output bit. Only 0 or 0xFFFF FFFF can be read. Writing any value other than 0 will set the output bit.

23.5.2.1.3 Port direction (DIR0 - DIR3)

The DIR registers configure each pin in a GPIO port as input or output. Each GPIO port has one direction register for configuring the port pins as inputs or outputs.

Offset

Register	Offset
DIR0	2000h
DIR1	2004h
DIR2	2008h
DIR3	200Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	DIRP31	DIRP30	DIRP29	DIRP28	DIRP27	DIRP26	DIRP25	DIRP24	DIRP23	DIRP22	DIRP21	DIRP20	DIRP19	DIRP18	DIRP17	DIRP16
Reset	0	0	0	0	0	0	0	0	0	0	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	DIRP15	DIRP14	DIRP13	DIRP12	DIRP11	DIRP10	DIRP9	DIRP8	DIRP7	DIRP6	DIRP5	DIRP4	DIRP3	DIRP2	DIRP1	DIRP0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0	Selects pin direction for pin PIOa_b.

Table continues on the next page...

Field	Description
DIRPn	Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0 - Input 1 - Output

23.5.2.1.4 Port mask (MASK0 - MASK3)

It defines which port pins are accessible in its MPIN register. Zeros in these registers enable reading and writing; ones disable writing and result in zeros in corresponding positions when reading.

Offset

Register	Offset
MASK0	2080h
MASK1	2084h
MASK2	2088h
MASK3	208Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MASKP31	MASKP30	MASKP29	MASKP28	MASKP27	MASKP26	MASKP25	MASKP24	MASKP23	MASKP22	MASKP21	MASKP20	MASKP19	MASKP18	MASKP17	MASKP16
Reset	0	0	0	0	0	0	0	0	0	0	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	MASKP15	MASKP14	MASKP13	MASKP12	MASKP11	MASKP10	MASKP9	MASKP8	MASKP7	MASKP6	MASKP5	MASKP4	MASKP3	MASKP2	MASKP1	MASKP0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 MASKPn	Port Mask Controls which bits corresponding to PIOa_b are active in the MPIN register. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0 - Read MPIN: pin state; write MPIN: load output bit 1 - Read MPIN: 0; write MPIN: output bit not affected

23.5.2.1.5 Port pin (PIN0 - PIN3)

Reading these registers returns the current state of the pins, regardless of direction, masking, or alternate digital functions. Pins configured as analog I/O always read as 0s. Writing these registers loads the output bits of the pins written to, regardless of the Mask register.

NOTE

The data read after reset depends on the state of the pin, which in turn may depend on an external source.

Offset

Register	Offset
PIN0	2100h
PIN1	2104h
PIN2	2108h
PIN3	210Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PORT 31	PORT 30	PORT 29	PORT 28	PORT 27	PORT 26	PORT 25	PORT 24	PORT 23	PORT 22	PORT 21	PORT 20	PORT 19	PORT 18	PORT 17	PORT 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 W	PORT 15	PORT 14	PORT 13	PORT 12	PORT 11	PORT 10	PORT 9	PORT 8	PORT 7	PORT 6	PORT 5	PORT 4	PORT 3	PORT 2	PORT 1	PORT 0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 PORTn	Port pins Reads pin states or loads output bits. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0 - Read- pin is low; Write- clear output bit 1 - Read- pin is high; Write- set output bit

23.5.2.1.6 Masked Port Pin (MPIN0 - MPIN3)

These registers are similar to the PIN registers, except that the value read is masked by ANDing with the inverted contents of the corresponding MASK register, and writing to one of these registers only affects output register bits that are enabled by zeros in the corresponding MASK register.

Offset

Register	Offset
MPIN0	2180h
MPIN1	2184h
MPIN2	2188h
MPIN3	218Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MPORTP31	MPORTP30	MPORTP29	MPORTP28	MPORTP27	MPORTP26	MPORTP25	MPORTP24	MPORTP23	MPORTP22	MPORTP21	MPORTP20	MPORTP19	MPORTP18	MPORTP17	MPORTP16
Reset	0	0	0	0	0	0	0	0	0	0	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	MPORTP15	MPORTP14	MPORTP13	MPORTP12	MPORTP11	MPORTP10	MPORTP9	MPORTP8	MPORTP7	MPORTP6	MPORTP5	MPORTP4	MPORTP3	MPORTP2	MPORTP1	MPORTP0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 MPORTPn	Mask bits for port pins Controls which bits corresponding to PIOa_b are active in the MPIN register. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. **NOTE** The data read after reset depends on the state of the pin, which in turn may depend on an external source. 0 - Read- pin is LOW and/or the corresponding bit in the MASK register is 1; write- clear output bit if the corresponding bit in the MASK register is 0 1 - Read- pin is HIGH and the corresponding bit in the MASK register is 0; write- set output bit if the corresponding bit in the MASK register is 0

23.5.2.1.7 Port set (SET0 - SET3)

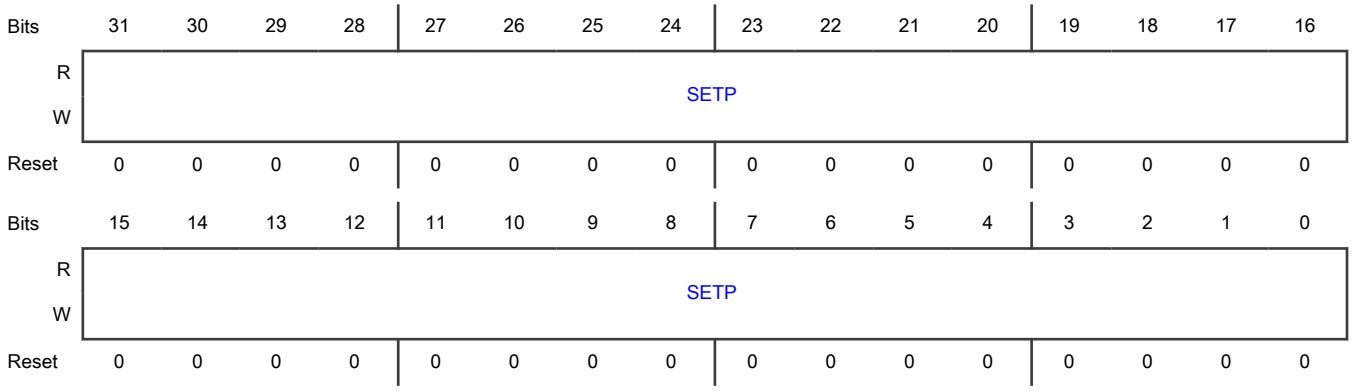
Output bits can be set by writing ones to these registers, regardless of MASK registers. Reading from these register returns the port's output bits, regardless of pin directions.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

Offset

Register	Offset
SET0	2200h
SET1	2204h
SET2	2208h
SET3	220Ch

Diagram



Fields

Field	Description
31-0 SETP	Read or set output bits Sets output bits for port pins 0000_0000_0000_0000_0000_0000_0000_0000 - Read- output bit; write- no operation 0000_0000_0000_0000_0000_0000_0000_0001 - Read- output bit; write- set output bit

23.5.2.1.8 Port clear (CLR0 - CLR3)

Output bits can be cleared by writing ones to these write-only registers, regardless of MASK registers.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

Offset

Register	Offset
CLR0	2280h
CLR1	2284h
CLR2	2288h
CLR3	228Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CLRP 31	CLRP 30	CLRP 29	CLRP 28	CLRP 27	CLRP 26	CLRP 25	CLRP 24	CLRP 23	CLRP 22	CLRP 21	CLRP 20	CLRP 19	CLRP 18	CLRP 17	CLRP 16
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
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	CLRP 15	CLRP 14	CLRP 13	CLRP 12	CLRP 11	CLRP 10	CLRP 9	CLRP 8	CLRP 7	CLRP 6	CLRP 5	CLRP 4	CLRP 3	CLRP 2	CLRP 1	CLRP 0
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-0 CLRPn	Clear output bits Clear output bits for port pins 0 - No operation 1 - Clears output bit

23.5.2.1.9 Port toggle (NOT0 - NOT3)

Output bits can be toggled by writing ones to these write-only registers.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

Offset

Register	Offset
NOT0	2300h
NOT1	2304h
NOT2	2308h
NOT3	230Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	NOTP 31	NOTP 30	NOTP 29	NOTP 28	NOTP 27	NOTP 26	NOTP 25	NOTP 24	NOTP 23	NOTP 22	NOTP 21	NOTP 20	NOTP 19	NOTP 18	NOTP 17	NOTP 16
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																
W	NOTP 15	NOTP 14	NOTP 13	NOTP 12	NOTP 11	NOTP 10	NOTP 9	NOTP 8	NOTP 7	NOTP 6	NOTP 5	NOTP 4	NOTP 3	NOTP 2	NOTP 1	NOTP 0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-0 NOTPn	Toggle output bits Toggle direction bits for port pins. 0 - No operation 1 - Toggle output bit

23.5.2.1.10 Port direction set (DIRSET0 - DIRSET3)

Direction bits can be set by writing ones to these registers.

Offset

Register	Offset
DIRSET0	2380h
DIRSET1	2384h
DIRSET2	2388h
DIRSET3	238Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...
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																
W	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP...	DIRSETP9	DIRSETP8	DIRSETP7	DIRSETP6	DIRSETP5	DIRSETP4	DIRSETP3	DIRSETP2	DIRSETP1	DIRSETP0
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-0 DIRSETPn	Direction set bits for Port pins Set pin direction for port pins. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0 - No operation 1 - Sets direction bit

23.5.2.1.11 Port direction clear (DIRCLR0 - DIRCLR3)

Direction bits can be cleared by writing ones to these write-only registers.

Offset

Register	Offset
DIRCLR0	2400h
DIRCLR1	2404h
DIRCLR2	2408h
DIRCLR3	240Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
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	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP...	DIRCLRP9	DIRCLRP8	DIRCLRP7	DIRCLRP6	DIRCLRP5	DIRCLRP4	DIRCLRP3	DIRCLRP2	DIRCLRP1	DIRCLRP0
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-0 DIRCLRPn	Clear direction bits. Clear pin direction bits for Port pins. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0 - No operation 1 - Clears direction bits

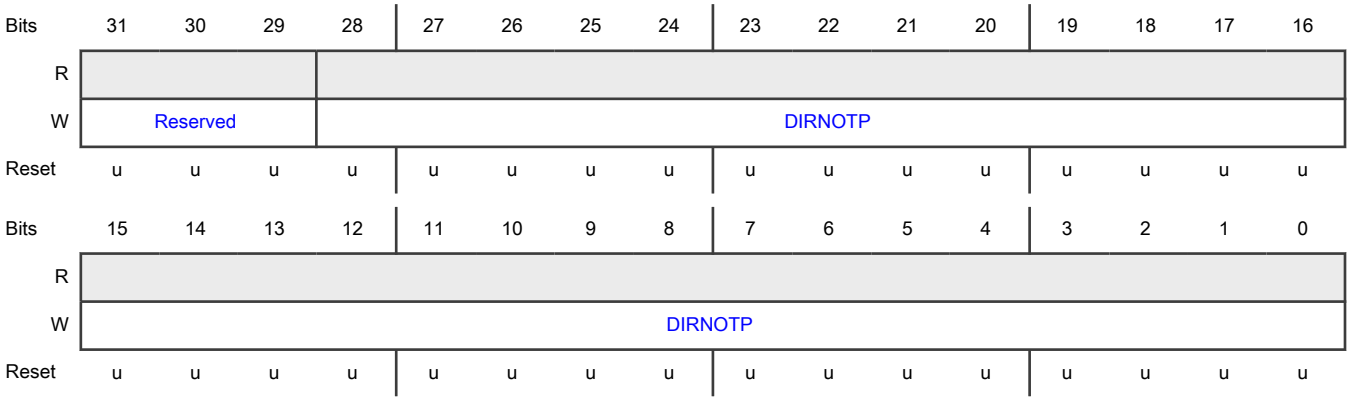
23.5.2.1.12 Port direction toggle (DIRNOT0 - DIRNOT3)

Direction bits can be toggled by writing ones to these write-only registers.

Offset

Register	Offset
DIRNOT0	2480h
DIRNOT1	2484h
DIRNOT2	2488h
DIRNOT3	248Ch

Diagram



Fields

Field	Description
31-29 —	Reserved
28-0 DIRNOTP	Toggle direction bits. Toggle direction bits for each pin. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0_0000_0000_0000_0000_0000_0000_0000 - No operation 0_0000_0000_0000_0000_0000_0000_0001 - Toggles direction bit

23.5.2.1.13 Interrupt A enable control (INTENA0 - INTENA3)

These registers allow the enabling or disabling of individual pins contributions to GPIO interrupt A.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

Offset

Register	Offset
INTENA0	2500h
INTENA1	2504h
INTENA2	2508h
INTENA3	250Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INT_EN31	INT_EN30	INT_EN29	INT_EN28	INT_EN27	INT_EN26	INT_EN25	INT_EN24	INT_EN23	INT_EN22	INT_EN21	INT_EN20	INT_EN19	INT_EN18	INT_EN17	INT_EN16
Reset	0	0	0	0	0	0	0	0	0	0	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	INT_EN15	INT_EN14	INT_EN13	INT_EN12	INT_EN11	INT_EN10	INT_EN9	INT_EN8	INT_EN7	INT_EN6	INT_EN5	INT_EN4	INT_EN3	INT_EN2	INT_EN1	INT_EN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 INT_ENn	Interrupt A enable bits. Interrupt enable control for each pin. 0 - Pin does not contribute to GPIO interrupt A 1 - Pin contributes to GPIO interrupt A

23.5.2.1.14 Interrupt B enable control (INTENB0 - INTENB3)

These registers allow the enabling or disabling of individual pins contributions to GPIO interrupt B.

Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31.

Offset

Register	Offset
INTENB0	2580h
INTENB1	2584h
INTENB2	2588h
INTENB3	258Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INT_EN31	INT_EN30	INT_EN29	INT_EN28	INT_EN27	INT_EN26	INT_EN25	INT_EN24	INT_EN23	INT_EN22	INT_EN21	INT_EN20	INT_EN19	INT_EN18	INT_EN17	INT_EN16
Reset	0	0	0	0	0	0	0	0	0	0	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	INT_EN15	INT_EN14	INT_EN13	INT_EN12	INT_EN11	INT_EN10	INT_EN9	INT_EN8	INT_EN7	INT_EN6	INT_EN5	INT_EN4	INT_EN3	INT_EN2	INT_EN1	INT_EN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 INT_ENn	Interrupt B enable bits. Interrupt enable control for each pin. 0 - Pin does not contribute to GPIO interrupt B 1 - Pin contributes to GPIO interrupt B

23.5.2.1.15 Interupt polarity control (INTPOL0 - INTPOL3)

These registers select the polarity of GPIO interrupts.

Offset

Register	Offset
INTPOL0	2600h
INTPOL1	2604h
INTPOL2	2608h
INTPOL3	260Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...
Reset	0	0	0	0	0	0	0	0	0	0	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	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL...	POL_CTL9	POL_CTL8	POL_CTL7	POL_CTL6	POL_CTL5	POL_CTL4	POL_CTL3	POL_CTL2	POL_CTL1	POL_CTL0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 POL_CTLn	Polarity control for each pin Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0 - High level or rising edge triggered 1 - Low level or falling edge triggered

23.5.2.1.16 Interrupt edge select (INTEDG0 - INTEDG3)

These registers allow GPIO interrupts to be selected as level or edge triggered.

Offset

Register	Offset
INTEDG0	2680h
INTEDG1	2684h
INTEDG2	2688h
INTEDG3	268Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	EDGE 31	EDGE 30	EDGE 29	EDGE 28	EDGE 27	EDGE 26	EDGE 25	EDGE 24	EDGE 23	EDGE 22	EDGE 21	EDGE 20	EDGE 19	EDGE 18	EDGE 17	EDGE 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 W	EDGE 15	EDGE 14	EDGE 13	EDGE 12	EDGE 11	EDGE 10	EDGE 9	EDGE 8	EDGE 7	EDGE 6	EDGE 5	EDGE 4	EDGE 3	EDGE 2	EDGE 1	EDGE 0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 EDGEn	Edge or level mode select bits. These bits allow GPIO interrupts to be selected as level or edge triggered. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. 0 - Level mode 1 - Edge mode

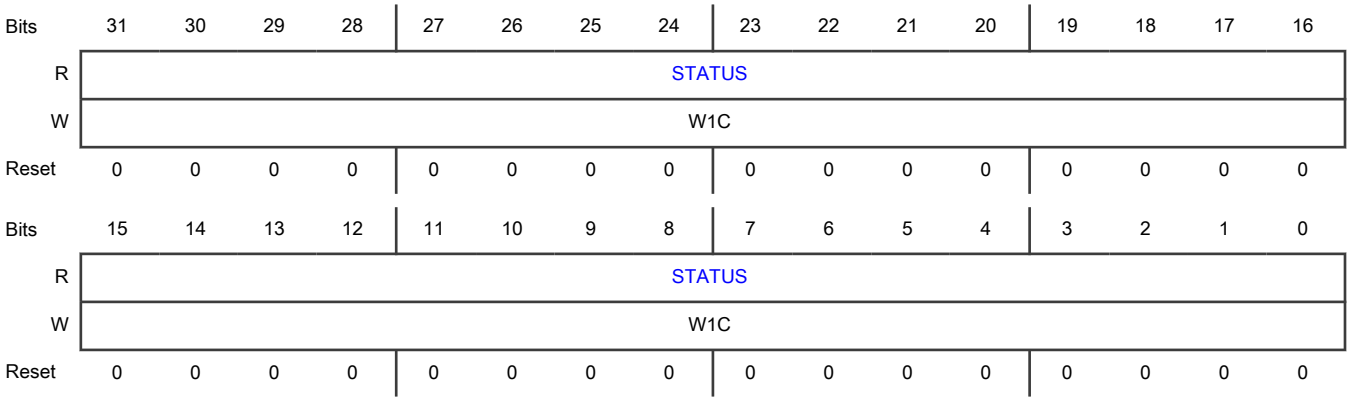
23.5.2.1.17 Interrupt status for interrupt A (INTSTATA0 - INTSTATA3)

These registers report the status of interrupt group A for each GPIO port.

Offset

Register	Offset
INTSTATA0	2700h
INTSTATA1	2704h
INTSTATA2	2708h
INTSTATA3	270Ch

Diagram



Fields

Field	Description
31-0 STATUS	Interrupt status. Interrupt group A status bits Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. Read: 0 - not pending; Read: 1 - pending; Write: 0 - no action. Write: 1 - clear status bit.

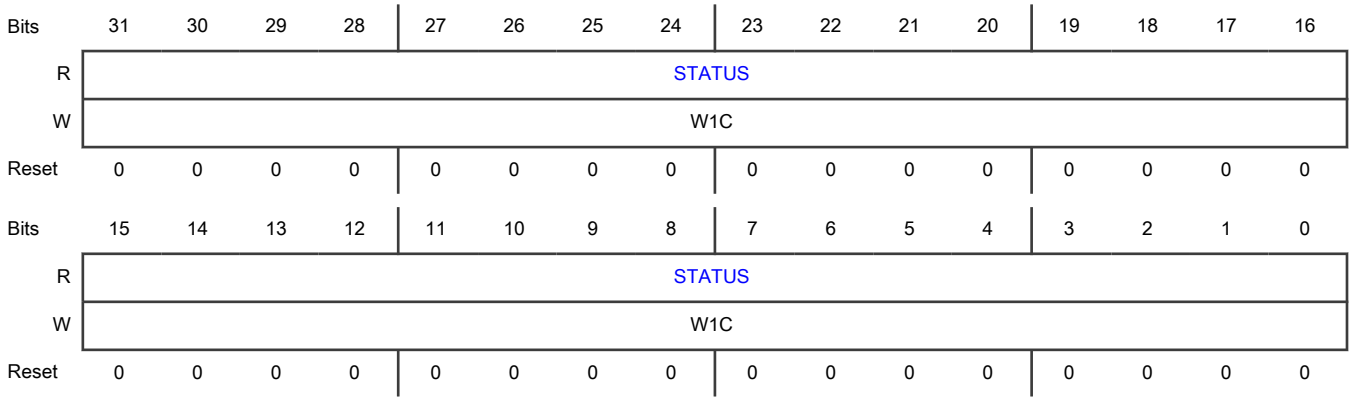
23.5.2.1.18 Interrupt status for interrupt B (INTSTATB0 - INTSTATB3)

These registers report the status of interrupt group B for each GPIO port.

Offset

Register	Offset
INTSTATB0	2780h
INTSTATB1	2784h
INTSTATB2	2788h
INTSTATB3	278Ch

Diagram



Fields

Field	Description
31-0 STATUS	Interrupt status Interrupt group B status bits. Bit 0- PIOa_0, Bit 1- PIOa_1,.....,Bit 31- PIOa_31. Read 0 - not pending; Read 1 - pending; Write 0 - no action. Write 1 - clear status bit.

Chapter 24
Group GPIO Input Interrupt (GINT)

24.1 Chip-specific GINT information

Table 113. Reference links to related information

Topic	Related module	Reference
Full description	GINT	GINT
System memory map		System memory map
Clocking		Clock generation
Signal multiplexing	Port control	Signal multiplexing

24.1.1 Module instances

The chip includes two instances of GINT module, GINT0 and GINT1.

24.1.2 Initialization

- For the group interrupt feature, enable the clock to both the GROUP0 and GROUP1 register interfaces in the AHBCLKCTRL0[GINT].
- For sleep mode, the group interrupt wake-up feature is enabled in the CPU NVIC.
- For deep-sleep and power-down low power modes, the group interrupt wake-up feature is enabled via the relevant low power API.
- The pins can be configured as GPIO pins through IOCON.
- The GINT block reads the input from the pin bypassing IOCON multiplexing.
- Analog function should not be selected on pins that are input to the group interrupts. Selecting an analog function in IOCON disables the digital pad and the digital signal is tied to 0.

24.2 Introduction

GPIO ports/pins are connected to GINT. GPIO pins set as inputs can be used as combinations of level and edge sensitive interrupts.

The GINT generates an interrupt when the designated pattern is detected on the selected input pins. If the chip is in a low-power mode, it first asynchronously wakes the chip up prior to asserting the interrupt request.

24.3 Features

- Inputs from digital pins can be enabled to contribute to a combined group interrupt.
- Polarity of each input enabled for the group interrupt can be configured as HIGH or LOW.
- Enabled interrupts can be logically combined through an OR or AND operation.
- Two group interrupts are supported to reflect two distinct interrupt patterns.
- The group interrupts can wake up the part from sleep, deep-sleep mode, and power-down modes.

24.4 Functional description

A subset of GPIO pins in each port can be selected to contribute to a common group interrupt (GINT) and can be enabled to wake the part up from sleep, deep-sleep and power-down modes.

An interrupt can be requested for each port, based on a selected subset of pins within each port. The level on each pin is exclusive-ORed with its polarity bit, and the result is ANDed with its enable bit. These results are then inclusive-ORed among all the pins in the port to create the raw interrupt request.

The raw interrupt request from each of the two group interrupts is sent to the NVIC, which can be programmed to treat it as level- or edge-sensitive, or it can be edge-detected by the wake-up interrupt logic.

24.5 Memory Map and register definition

This section includes the GINT module memory map and detailed descriptions of all registers.

24.5.1 Group GPIO input interrupt (GINT0/1) register descriptions

The GINT registers determine the pins that are enabled to generate interrupts and the active polarities of each of those inputs. These registers also select if the interrupt output will be level or edge triggered and if it will be based on the OR or the AND of all of the enabled inputs.

24.5.1.1 GINT memory map

GINT0 base address: 4000_2000h

GINT1 base address: 4000_3000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	GPIO grouped interrupt control (CTRL)	32	See section	See section
20 - 24	Port polarity (PORT_POL0 - PORT_POL1)	32	RW	FFFF_FFFF
40 - 44	GPIO grouped interrupt port 0 enable register (PORT_ENA0 - PORT_ENA1)	32	RW	0000_0000

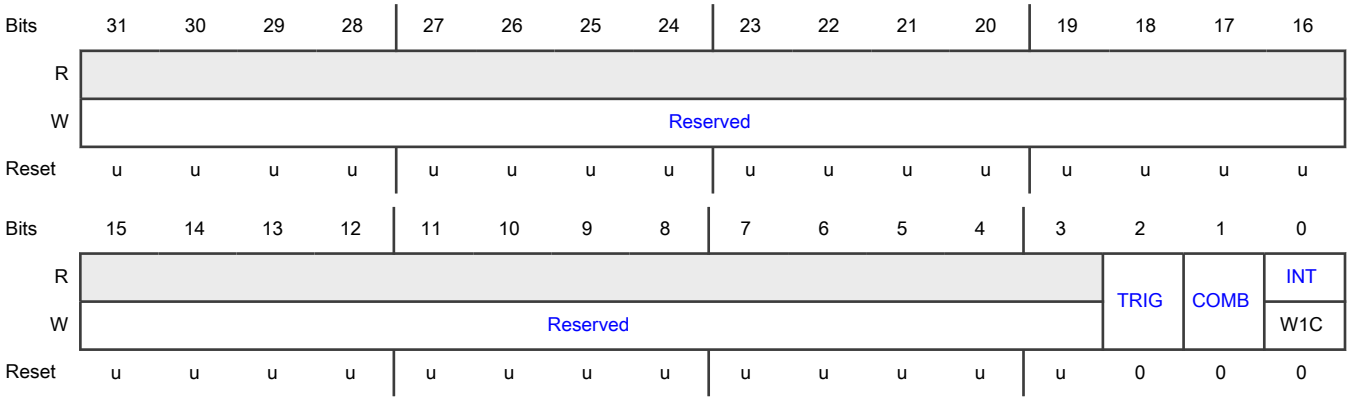
24.5.1.1.1 GPIO grouped interrupt control (CTRL)

This register controls the GPIO group interrupts.

Offset

Register	Offset
CTRL	0h

Diagram



Fields

Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2 TRIG	Group interrupt trigger 0 - Edge-triggered 1 - Level-triggered
1 COMB	Combine enabled inputs for group interrupt Selects the function (AND or OR) applied to the enabled inputs used to generate the grouped interrupt. 0 - OR functionality 1 - AND functionality
0 INT	Group interrupt status Cleared by writing a one. Writing zero has no effect. Interrupt request line can be cleared by writing a one to the interrupt status bit. 0 - No interrupt request is pending. 1 - Interrupt request is pending.

24.5.1.1.2 Port polarity (PORT_POL0 - PORT_POL1)

These registers determine the polarity of each enabled pin contributing to the grouped interrupt. Each port is associated with its own port polarity register, and the values of the registers together determine the grouped interrupt. Each register controls the polarity of pins in a port.

Offset

Register	Offset
PORT_POL0	20h
PORT_POL1	24h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	POL31	POL30	POL29	POL28	POL27	POL26	POL25	POL24	POL23	POL22	POL21	POL20	POL19	POL18	POL17	POL16
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 W	POL15	POL14	POL13	POL12	POL11	POL10	POL9	POL8	POL7	POL6	POL5	POL4	POL3	POL2	POL1	POL0
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Description
31-0 POLn	Polarity of pin n of the port Configures pin polarity of port pins for group interrupt. 0 - Pin is active LOW. If the level on this pin is LOW, the pin contributes to the group interrupt. 1 - Pin is active HIGH. If the level on this pin is HIGH, the pin contributes to the group interrupt.

24.5.1.1.3 GPIO grouped interrupt port 0 enable register (PORT_ENA0 - PORT_ENA1)

These registers enable pins to contribute to the group interrupt. Each port is associated with a port enable register, and the values of these registers together determine which pins contribute to the group interrupt.

Offset

Register	Offset
PORT_ENA0	40h
PORT_ENA1	44h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ENA31	ENA30	ENA29	ENA28	ENA27	ENA26	ENA25	ENA24	ENA23	ENA22	ENA21	ENA20	ENA19	ENA18	ENA17	ENA16
Reset	0	0	0	0	0	0	0	0	0	0	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	ENA15	ENA14	ENA13	ENA12	ENA11	ENA10	ENA9	ENA8	ENA7	ENA6	ENA5	ENA4	ENA3	ENA2	ENA1	ENA0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 ENAn	Enables port pin n to contribute to the group interrupt 0 - Pin is disabled and does not contribute to the grouped interrupt 1 - Pin is enabled and contributes to the grouped interrupt

Chapter 25
Pin Interrupt and Pattern Match (PINT)

25.1 Chip-specific PINT information

Table 114. Reference links to related information

Topic	Related module	Reference
Full description	PINT	PINT
System memory map		System memory map
Clocking		Clock generation
Signal multiplexing	Port control	Signal multiplexing

25.1.1 Module instances

The chip includes two instances of PINT module, PINT0 and PINT1.

25.1.2 Initialization

- Select up to eight external interrupt pins from all available digital port pins on ports 0 and 1 in INPUTMUX. The pin selection process is the same for pin interrupts and the pattern match engine. The two features are mutually exclusive.
- Enable the clock to the PINT module via AHBCLKCTRL0[PINT] bit (See SYSCON).
- Clear the PINT peripheral reset via PRESETCTRL0 register bit .

25.1.2.1 Pin Interrupts Initialization

In addition to the initialization steps described above, for Pin Interrupts initialization do the following -

- Each selected pin interrupt is assigned to one interrupt in the NVIC (see NVIC). Note that only the first 4 are available to the Cortex M33.

25.1.2.2 Pattern Match Initialization

In addition to the initialization steps described above, for Pattern match engine initialization do the following -

- Each bit slice of the pattern match engine is assigned to one interrupt (see NVIC). Note that only the first 4 are available to the Cortex M33, which is present on selected devices.

25.1.2.3 Configure pins as pin interrupts or as inputs to the pattern match engine

1. Determine the pins that you want to serve as pin interrupts or pattern match inputs. See the data sheet for determining the GPIO port pin number associated with the package pin.
2. For each pin selected, program the GPIO port pin number from ports 0 and 1 into one of the eight PINT_SEL registers in the INPUTMUX module.

   NOTE

   The port pin number serves to identify the pin to the PINT_SEL register. Any function, including GPIO, can be assigned to this pin via IOCON.

3. Enable each pin interrupt in the NVIC.

Once the pin interrupts or pattern match inputs are configured, the pin interrupt detection levels or the pattern match boolean expression can be set up.

NOTE

The inputs to the Pin interrupt select registers bypass the IOCON function selection. They do not have to be selected as GPIO in IOCON. Make sure that no analog function is selected on pins that are input to the pin interrupts.

25.2 Overview

PINT uses standard GPIO functions as inputs. Pins with configurable functions can serve as external interrupts or inputs to the pattern match engine.

25.2.1 Features

- Pin interrupts -
  - Up to 8 pins can be selected from all GPIO pins on ports 0 and 1 as edge- or level-sensitive interrupt requests. Each request creates a separate interrupt in the NVIC.
  - Edge-sensitive interrupt pins can interrupt on rising or falling edges or both.
  - Level-sensitive interrupt pins can be HIGH- or LOW-active.
- Pattern match engine -
  - Up to 8 pins can be selected from all digital pins on ports 0 and 1 to contribute to a boolean expression. The boolean expression consists of specified levels and/or transitions on various combinations of these pins.
  - Each bit slice minterm (product term) comprising the specified boolean expression can generate its own, dedicated interrupt request.
  - Any occurrence of a pattern match can be programmed to also generate an RXEV notification to the CPU.
  - Pattern match can be used, in conjunction with software, to create complex state machines based on pin inputs.

25.2.2 Block diagram

From all available GPIO pins, up to 8 pins can be selected in the INPUTMUX module to serve as external interrupt pins (see INPUTMUX for more details). The external interrupt pins are connected to eight individual interrupts in the NVIC and are created based on rising or falling edges or on the input level on the pin.

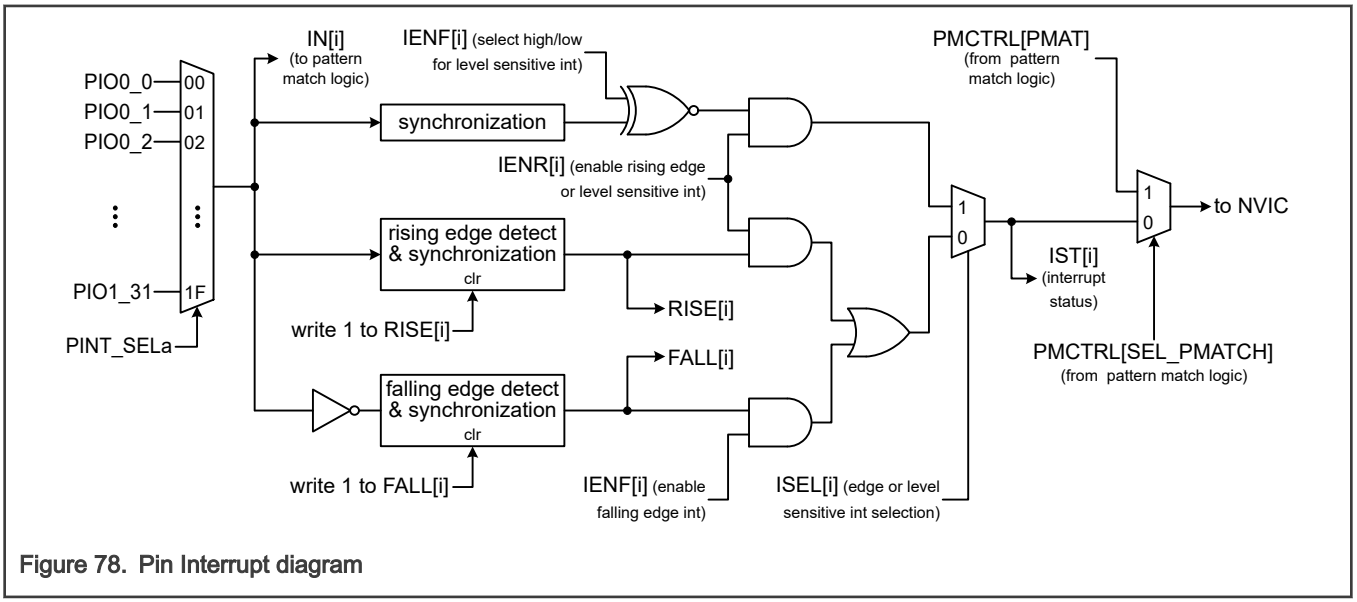


Figure 78. Pin Interrupt diagram

25.3 Functional description

25.3.1 Pattern match engine

The pattern match feature allows complex boolean expressions to be constructed from the same set of 8 GPIO pins that were selected for the GPIO pin interrupts. Each term in the boolean expression is implemented as one slice of the pattern match engine. A slice consists of an input selector and detection logic that monitors the selected input continuously and creates a HIGH output (true). If the input qualifies as detected, several terms can be combined to a minterm, and a pin interrupt is asserted when the minterm evaluates as true.

The detection logic of each slice can detect the following events on the selected input:

- Edge with memory (sticky): A rising edge, a falling edge, or a rising or falling edge that is detected at any time after the edge-detection mechanism has been cleared. The input qualifies as detected (the detection logic output remains HIGH) until the pattern match engine detect logic is cleared again.
- Event (non-sticky): Every time an edge (rising or falling) is detected, the detection logic output goes HIGH. This bit is cleared after one clock cycle, and the detection logic can detect another edge.
- Level: A HIGH or LOW level on the selected input.

The figure below shows the details of the edge detection logic for each slice.

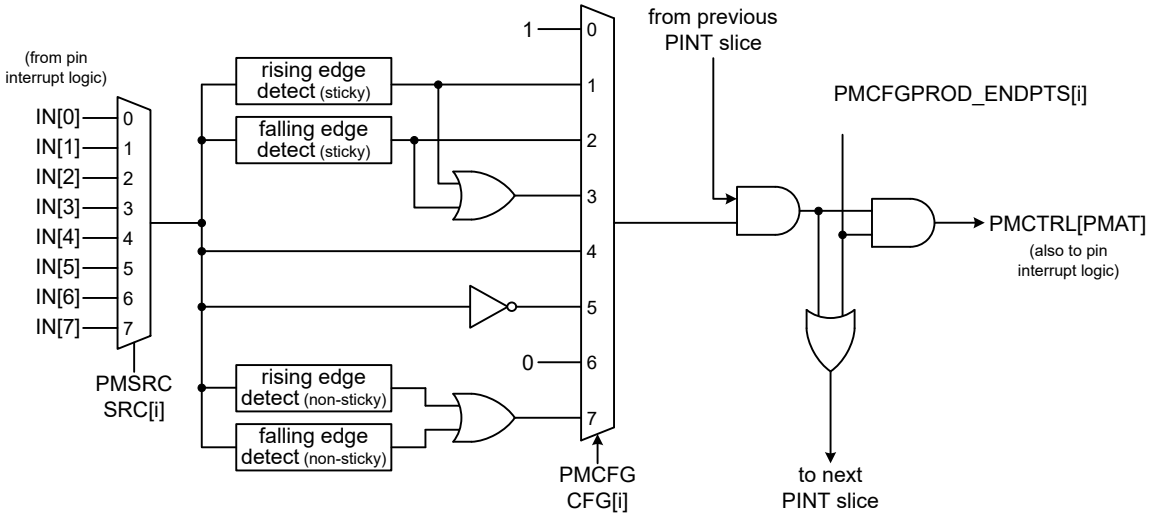


Figure 79. Pattern match bit slice

Sticky events can be combined with non-sticky events to create a pin interrupt whenever a rising or falling edge occurs after a qualifying edge event.

A time window can be created during which rising or falling edges can create a pin interrupt by combining a level detect with an event detect. See Pattern match engine edge detect examples for details.

The following figure shows connections between the pins and the pattern match engine. All pins that are inputs to the pattern match engine are selected in the INPUTMUX block and can be GPIO port pins or other pin functions depending on the configuration.

NOTE

The pattern match feature requires clocks in order to operate, and can thus not generate an interrupt or wake up the device during reduced power modes below sleep mode.

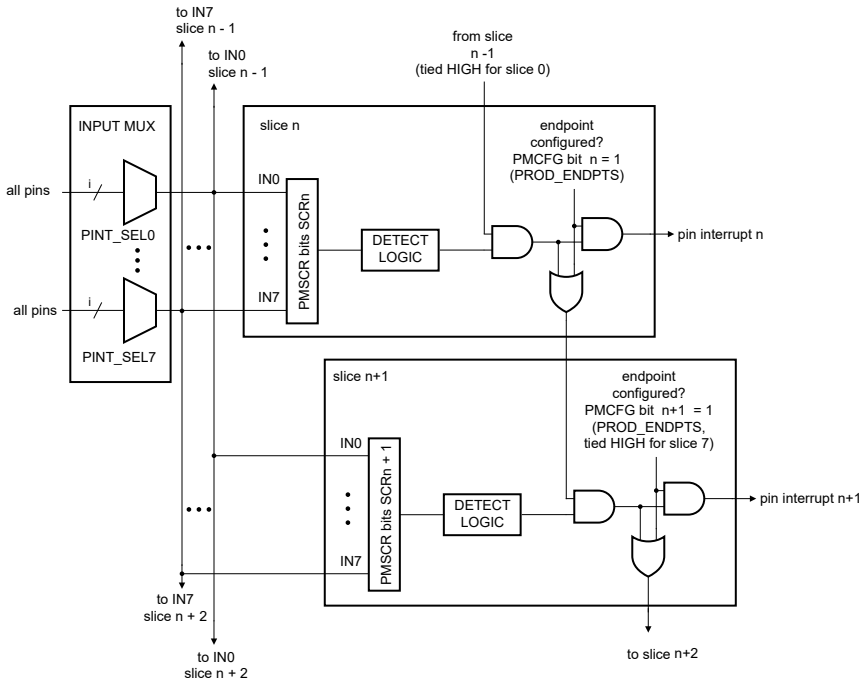


Figure 80. Pattern match engine connections

The pattern match logic continuously monitors the 8 inputs and generates interrupts when any one or more minterms (product terms) of the specified boolean expression is matched. A separate interrupt request is generated for each individual minterm.

In addition, the pattern match module can be enabled to generate a Receive Event (RXEV) output to the CPU when the entire boolean expression is true, when any minterm is matched.

The pattern match function utilizes the same 8 interrupt request lines as the pin interrupts so these two features are mutually exclusive as far as interrupt generation is concerned. A control bit is provided to select whether interrupt requests are generated in response to the standard pin interrupts or to pattern matches. Note that, if the pin interrupts are selected, the RXEV request to the CPU can still be enabled for pattern matches.

NOTE

Pattern matching cannot be used to wake the part up from deep-sleep mode. Pin interrupts must be selected in order to use the GPIO for wake-up.

The pattern match module is constructed of 8 bit-slice elements. Each bit slice is programmed to represent one component of one minterm (product term) within the boolean expression. The interrupt request associated with the last bit slice for a particular minterm will be asserted whenever that minterm is matched.

The pattern match capability can be used to create complex software state machines. Each minterm (and its corresponding individual interrupt) represents a different transition event to a new state. Software can then establish the new set of conditions (that is a new boolean expression) that will cause a transition out of the current state.

25.4 Signals

The inputs to the pin interrupt and pattern match engine are determined by the pin interrupt select registers in the INPUTMUX. See INPUTMUX for more details.

25.5 Application information

The following expression is specified through the registers Pattern Match Interrupt Bit-Slice Source (PMSRC) and Pattern Match Interrupt Bit Slice Configuration (PMCFG) :

```
IN0 AND NOT IN1 AND IN3 rising edge OR IN1 AND IN2 OR IN0 AND NOT IN3 AND NOT IN4
```

Each term in the boolean expression, IN0, NOT IN1, IN3 rising edge, etc., represents one bit slice of the pattern match engine.

- In the first AND function IN0 AND NOT IN1 AND IN3 rising edge, bit slice 0 monitors for a high-level on input IN0, bit slice 1 monitors for a low level on input IN1 and bit slice 2 monitors for a rising-edge on input IN3. If this combination is detected (all three terms are true) the interrupt associated with bit slice 2 will be asserted.
- In the second AND function IN1 AND IN2, bit slice 3 monitors input IN1 for a high level, bit slice 4 monitors input IN2 for a high level. If this combination is detected, the interrupt associated with bit slice 4 will be asserted.
- In the third AND function IN0 AND NOT IN3 AND NOT IN4, bit slice 5 monitors input IN0 for a high level, bit slice 6 monitors input IN3 for a low level, and bit slice 7 monitors input IN4 for a low level. If this combination is detected, the interrupt associated with bit slice 7 will be asserted.
- The ORed result of all three AND functions asserts the RXEV request to the CPU. That is, if any of the three terms are true, the output is asserted.

25.5.1 Pattern Match engine example

Suppose the desired boolean pattern to be matched is:

(IN1) + (IN1 * IN2) + (~IN2 * ~IN3 * IN6fe) + (IN5 * IN7ev)

with:

IN6fe = (sticky) falling-edge on input 6

IN7ev = (non-sticky) event (rising or falling edge) on input 7

Each individual term in the expression shown above is controlled by one bit-slice. To specify this expression, program the pattern match bit slice source and configuration register fields as follows:

- PMSRC register Pattern Match Interrupt Bit-Slice Source (PMSRC)
  - Since bit slice 5 will be used to detect a sticky event on input 6, a 1 can be written to the SRC5 bits to clear any pre-existing edge detects on bit slice 5.
  - SRC0: 001 - select input 1 for bit slice 0
  - SRC1: 001 - select input 1 for bit slice 1
  - SRC2: 010 - select input 2 for bit slice 2
  - SRC3: 010 - select input 2 for bit slice 3
  - SRC4: 011 - select input 3 for bit slice 4
  - SRC5: 110 - select input 6 for bit slice 5
  - SRC6: 101 - select input 5 for bit slice 6
  - SRC7: 111 - select input 7 for bit slice 7
- PMCFG register Pattern Match Interrupt Bit Slice Configuration (PMCFG)
  - PROD_ENDPTS0 = 1
  - PROD_ENDPTS2 = 1
  - PROD_ENDPTS5 = 1

- All other slices are not product term endpoints and their PROD_ENDPTS bits are 0. Slice 7 is always a product term endpoint and does not have a register bit associated with it.
- PROD_ENDPTS= 0100101 - bit slices 0, 2, 5, and 7 are product-term endpoints. (Bit slice 7 is an endpoint by default - no associated register bit).
- CFG0: 000 - high level on the selected input (input 1) for bit slice 0
- CFG1: 000 - high level on the selected input (input 1) for bit slice 1
- CFG2: 000 - high level on the selected input (input 2) for bit slice 2
- CFG3: 101 - low level on the selected input (input 2) for bit slice 3
- CFG4: 101 - low level on the selected input (input 3) for bit slice 4
- CFG5: 010 - (sticky) falling edge on the selected input (input 6) for bit slice 5
- CFG6: 000 - high level on the selected input (input 5) for bit slice 6
- CFG7: 111 - event (any edge, non-sticky) on the selected input (input 7) for bit slice 7

- PMCTRL register Pattern Match Interrupt Control (PMCTRL)
  - Bit0: Setting this bit will select pattern matches to generate the pin interrupts in place of the standard pin interrupt mechanism.

    For this example, pin interrupt 0 will be asserted when a match is detected on the first product term (which, in this case, is just a high level on input 1).

    Pin interrupt 2 will be asserted in response to a match on the second product term.

    Pin interrupt 5 will be asserted when there is a match on the third product term.

    Pin interrupt 7 will be asserted on a match on the last term.
  - Bit1: Setting this bit will cause the RxEv signal to the CPU to be asserted whenever a match occurs on ANY of the product terms in the expression. Otherwise, the RXEV line will not be used.
  - Bit31:24: At any given time, bits 0, 2, 5 and/or 7 may be high if the corresponding product terms are currently matching.
  - The remaining bits will always be low.

25.5.2 Pattern match engine edge detect examples

The figures below show pattern match functionality only; accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

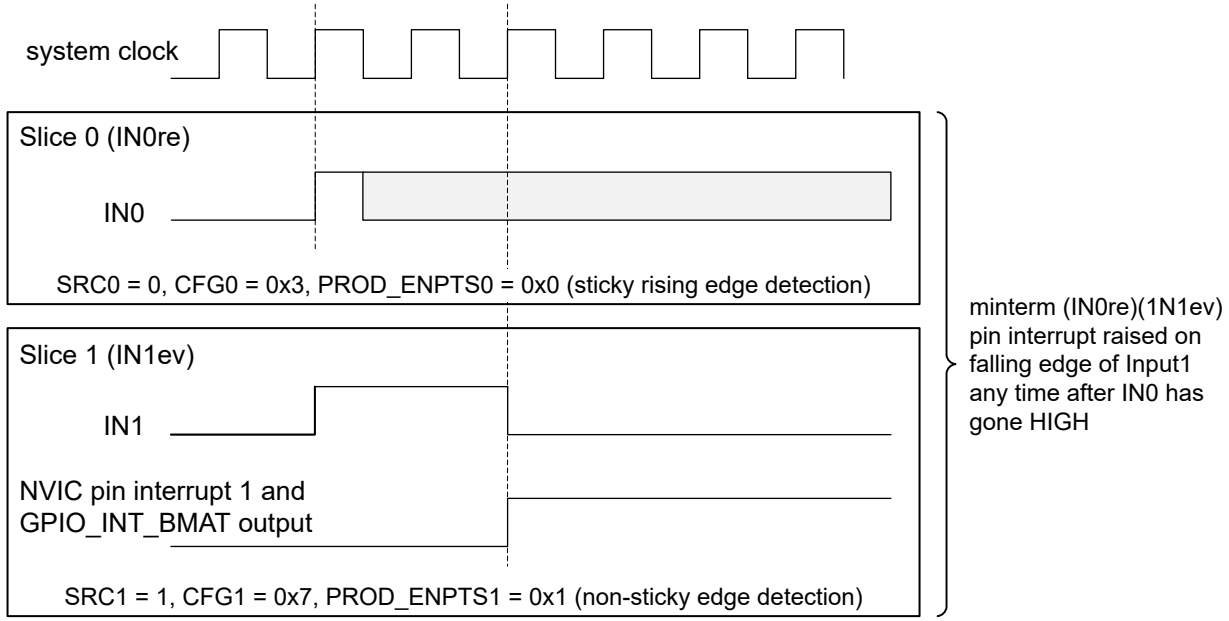


Figure 81. Pattern match engine examples: sticky edge detect

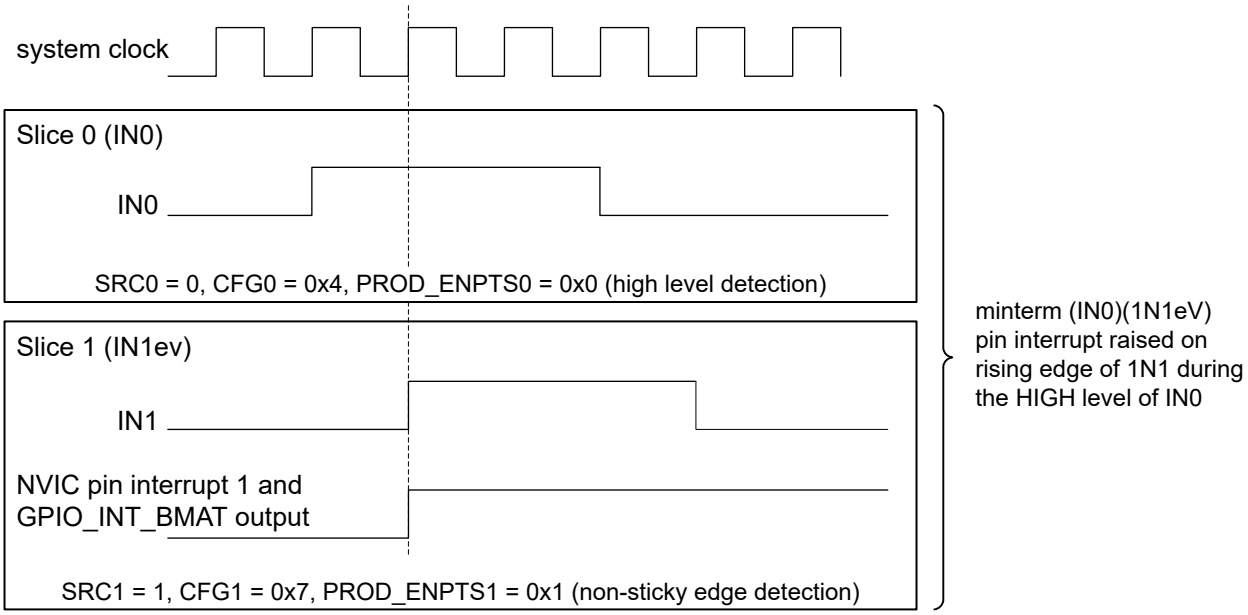


Figure 82. Pattern match engine examples: Windowed non-sticky edge detect evaluates as true

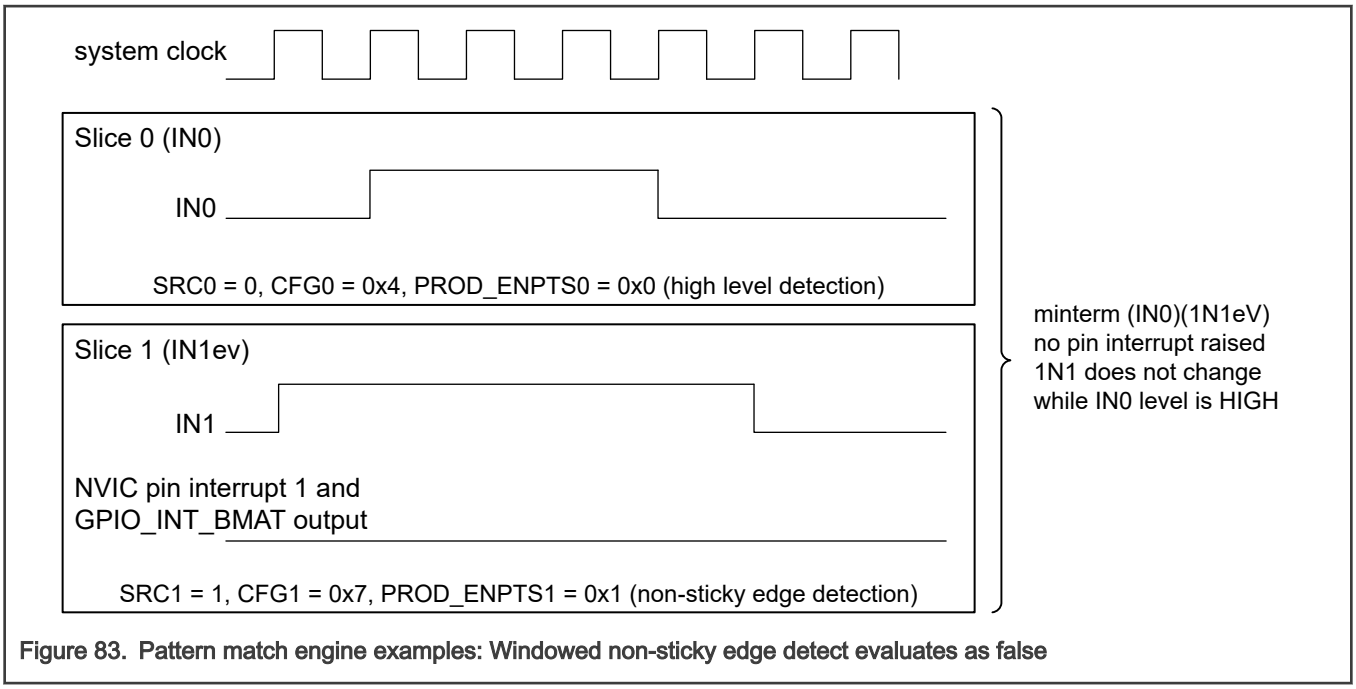


Figure 83. Pattern match engine examples: Windowed non-sticky edge detect evaluates as false

25.6 Memory Map and register definition

This section includes the PINT memory map and detailed descriptions of all registers.

The chip includes, up to 8 pins identified as interrupt sources by the Pin Interrupt Select registers (PINT_SELa). The registers have 8 bitfields that correspond to the pins called out by the PINT_SEL registers.

Table 115. Pin interrupt registers for edge- and level-sensitive pins

Name	Edge-sensitive function	Level-sensitive function
IENR	Enables rising-edge interrupts.	Enables level interrupts.
SIENR	Write to enable rising-edge interrupts.	Write to enable level interrupts.
CIENR	Write to disable rising-edge interrupts.	Write to disable level interrupts.
IENF	Enables falling-edge interrupts.	Selects active level.
SIENF	Write to enable falling-edge interrupts.	Write to select high-active.
CIENF	Write to disable falling-edge interrupts.	Write to select low-active.

25.6.1 Pin Interrupts and Pattern Match register descriptions

25.6.1.1 PINT memory map

PINT0 base address: 4000_4000h

PINT1 base address: 4000_5000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Pin Interrupt Mode (ISEL)	32	See section	See section
4	Pin Interrupt Level or Rising Edge Interrupt Enable (IENR)	32	See section	See section
8	Pin Interrupt Level or Rising Edge Interrupt Set (SIENR)	32	See section	See section
C	Pin Interrupt Level (Rising Edge Interrupt) Clear (CIENR)	32	See section	See section
10	Pin Interrupt Active Level or Falling Edge Interrupt Enable (IENF)	32	See section	See section
14	Pin Interrupt Active Level or Falling Edge Interrupt Set (SIENF)	32	See section	See section
18	Pin Interrupt Active Level or Falling Edge Interrupt Clear (CIENF)	32	See section	See section
1C	Pin Interrupt Rising Edge (RISE)	32	See section	See section
20	Pin Interrupt Falling Edge (FALL)	32	See section	See section
24	Pin Interrupt Status (IST)	32	See section	See section
28	Pattern Match Interrupt Control (PMCTRL)	32	See section	See section
2C	Pattern Match Interrupt Bit-Slice Source (PMSRC)	32	See section	See section
30	Pattern Match Interrupt Bit Slice Configuration (PMCFG)	32	See section	See section

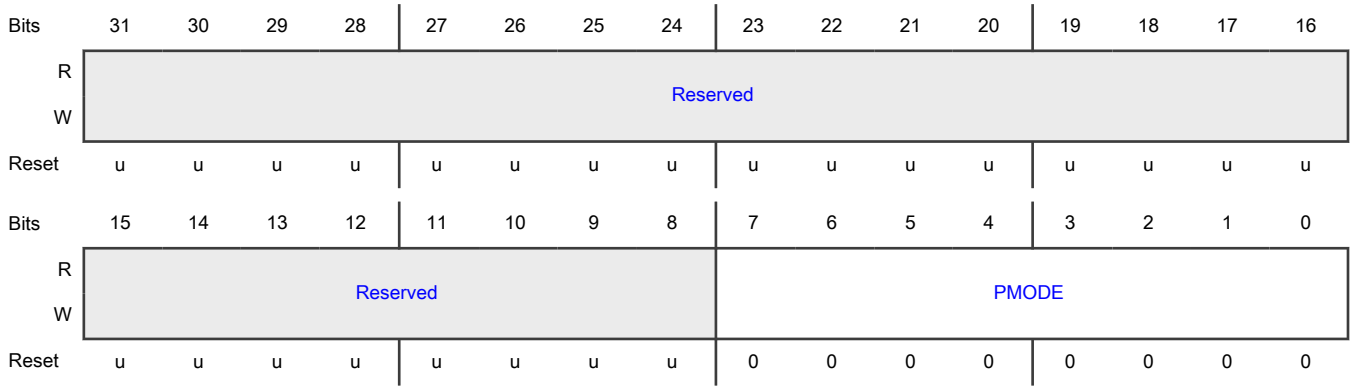
25.6.1.1.1 Pin Interrupt Mode (ISEL)

The ISEL register determines whether an interrupt is edge or level sensitive.

Offset

Register	Offset
ISEL	0h

Diagram



Fields

Field	Description
31-8 —	Read value is undefined, only zero should be written.
7-0 PMODE	Interrupt mode Selects the interrupt mode for each pin interrupt. Bit a configures the pin interrupt selected in PINT_SELa register in INPUTMUX. **NOTE** This field is not supported in every instance. The following table includes only supported registers. 0000_0000 - Edge-sensitive 0000_0001 - Level-sensitive

Instance	Field supported in	Field not supported in
PINT0	ISEL	—
PINT1	ISEL[1–0]	ISEL[7–2]

25.6.1.1.2 Pin Interrupt Level or Rising Edge Interrupt Enable (IENR)

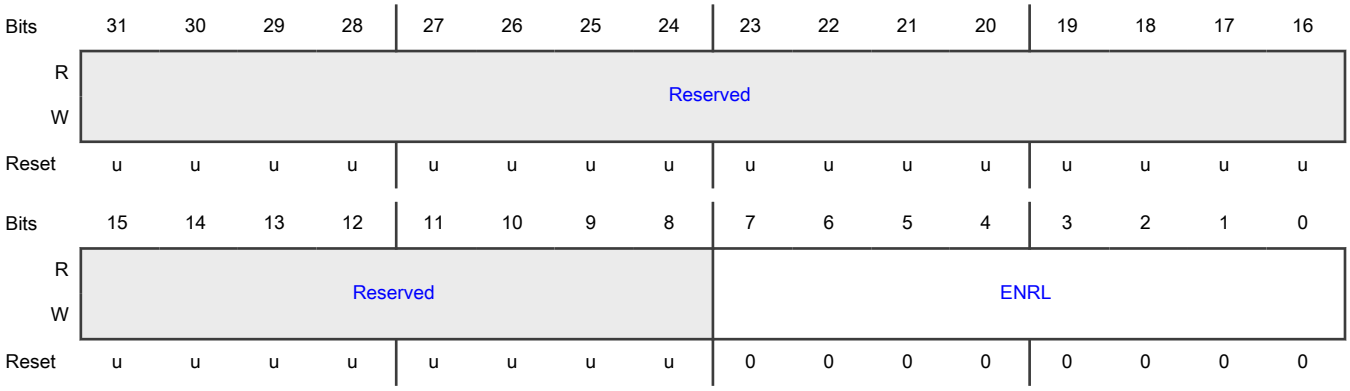
Each bit in the IENR register enables the interrupt depending on the PMODE configured in the ISEL register.

- If the PMODE is edge sensitive (PMODE = 0), the rising edge interrupt is enabled.
- If the PMODE is level sensitive (PMODE = 1), the level interrupt is enabled. The IENF register configures the active level (HIGH or LOW) for this interrupt.

Offset

Register	Offset
IENR	4h

Diagram



Fields

Field	Description
31-8 —	Read value is undefined, only zero should be written.
7-0 ENRL	Enable Interrupt Enables the rising edge or level interrupt for each pin interrupt. Bit n configures the pin interrupt selected in PINT_SELa. **NOTE** This field is not supported in every instance. The following table includes only supported registers.

Instance	Field supported in	Field not supported in
PINT0	IENR	—
PINT1	IENR[1–0]	IENR[7–2]

0000_0000 - Disable rising edge or level interrupt

0000_0001 - Enable rising edge or level interrupt

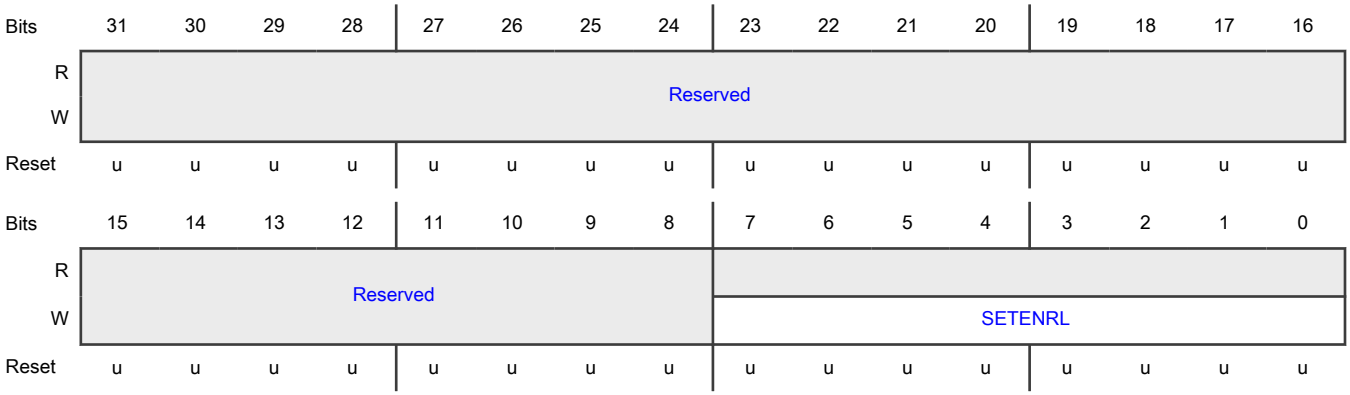
25.6.1.1.3 Pin Interrupt Level or Rising Edge Interrupt Set (SIENR)

Each bit in the SIENR register sets the corresponding bit in the IENR register depending on the PMODE configured in the ISEL register.

Offset

Register	Offset
SIENR	8h

Diagram



Fields

Field	Description
31-8 —	Reserved
7-0 SETENRL	Set bits in the IENR Ones written to this address set bits in the IENR, thus enabling interrupts. Bit a sets bit n in the IENR register. **NOTE** This field is not supported in every instance. The following table includes only supported registers. 0000_0000 - No operation 0000_0001 - Enable rising edge or level interrupt

Instance	Field supported in	Field not supported in
PINT0	SIENR	—
PINT1	SIENR[1–0]	SIENR[7–2]

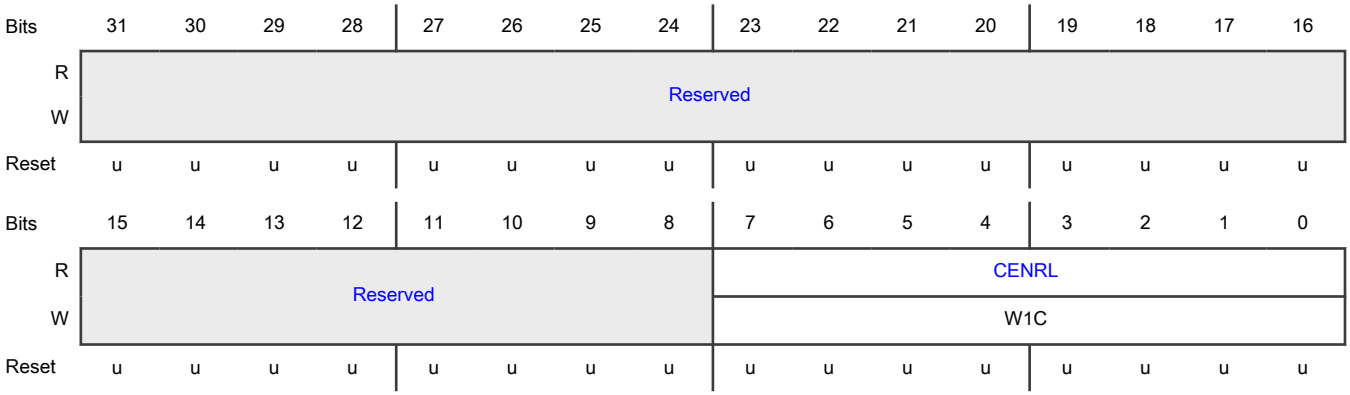
25.6.1.1.4 Pin Interrupt Level (Rising Edge Interrupt) Clear (CIENR)

Each bit in the CIENR register clears the corresponding bit in the IENR register depending on the PMODE configured in the ISEL register. Ones written to this address clear bits in the IENR, thus disabling the interrupts. Bit a clears bit n in the IENR register.

Offset

Register	Offset
CIENR	Ch

Diagram



Fields

Field	Description
31-8 —	Reserved
7-0 CENRL	Clear bits in the IENR **NOTE** This field is not supported in every instance. The following table includes only supported registers.

Instance	Field supported in	Field not supported in
PINT0	CIENR	—
PINT1	CIENR[1–0]	CIENR[7–2]

0000_0000 - No operation

0000_0001 - Disable rising edge or level interrupt

25.6.1.1.5 Pin Interrupt Active Level or Falling Edge Interrupt Enable (IENF)

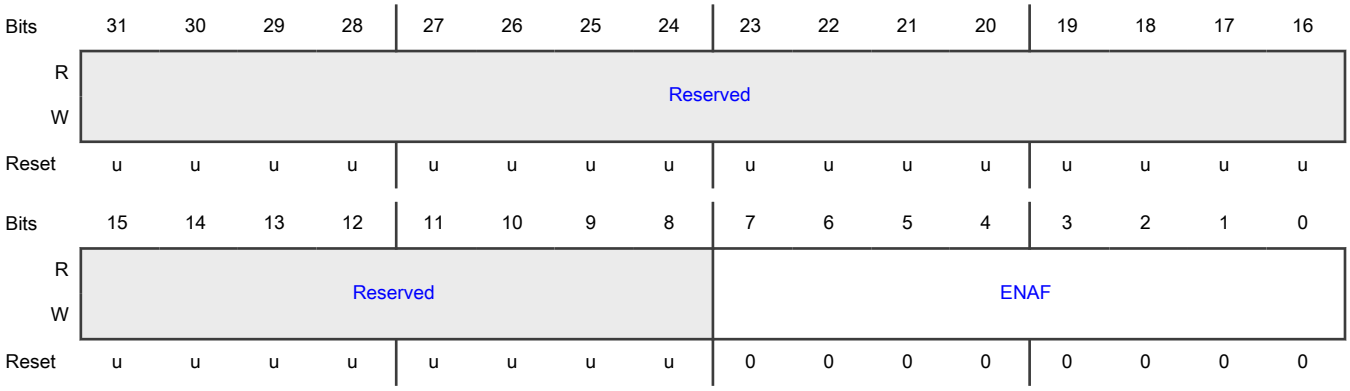
Each bit in the IENF register enables the falling edge interrupt or the configures the level sensitivity depending on the PMODE configured in the ISEL register.

- If the PMODE is edge sensitive (PMODE = 0), the falling edge interrupt is enabled.
- If the PMODE is level sensitive (PMODE = 1), the active level of the level interrupt (HIGH or LOW) is configured.

Offset

Register	Offset
IENF	10h

Diagram



Fields

Field	Description
31-8 —	Reserved
7-0 ENAF	Enable Interrupt Enables the falling edge or configures the active level interrupt for each pin interrupt. Bit a configures the pin interrupt selected in PINT_SELa. **NOTE** This field is not supported in every instance. The following table includes only supported registers.

Instance	Field supported in	Field not supported in
PINT0	IENF	—
PINT1	IENF[1–0]	IENF[7–2]

0000_0000 - Disable falling edge interrupt or set active interrupt level LOW

0000_0001 - Enable falling edge interrupt enabled or set active interrupt level HIGH

25.6.1.1.6 Pin Interrupt Active Level or Falling Edge Interrupt Set (SIENF)

Each bit in the SIENF register sets the corresponding bit in the IENF register depending on how PMODE configured in the ISEL register.

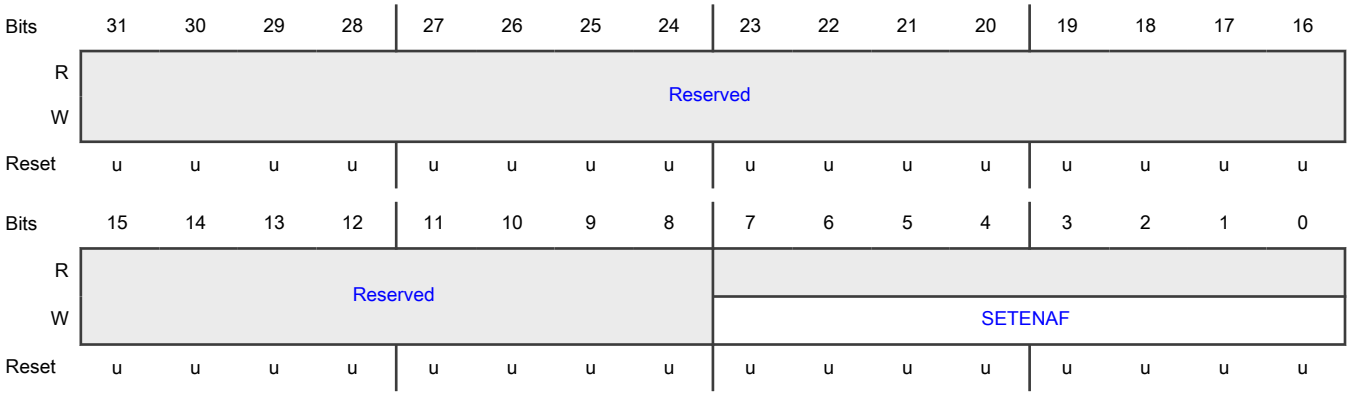
- If the PMODE is edge sensitive (PMODE = 0), the falling edge interrupt is set.

- If the PMODE is level sensitive (PMODE = 1), the HIGH-active interrupt is selected.

Offset

Register	Offset
SIENF	14h

Diagram



Fields

Field	Description
31-8 —	Reserved
7-0 SETENAF	Set bits in the IENF Ones written to this address set bits in the IENF, thus enabling interrupts. Bit a sets bit n in the IENF register. **NOTE** This field is not supported in every instance. The following table includes only supported registers.

Instance	Field supported in	Field not supported in
PINT0	SIENF	—
PINT1	SIENF[1–0]	SIENF[7–2]

0000_0000 - No operation

0000_0001 - Select HIGH-active interrupt or enable falling edge interrupt

25.6.1.1.7 Pin Interrupt Active Level or Falling Edge Interrupt Clear (CIENF)

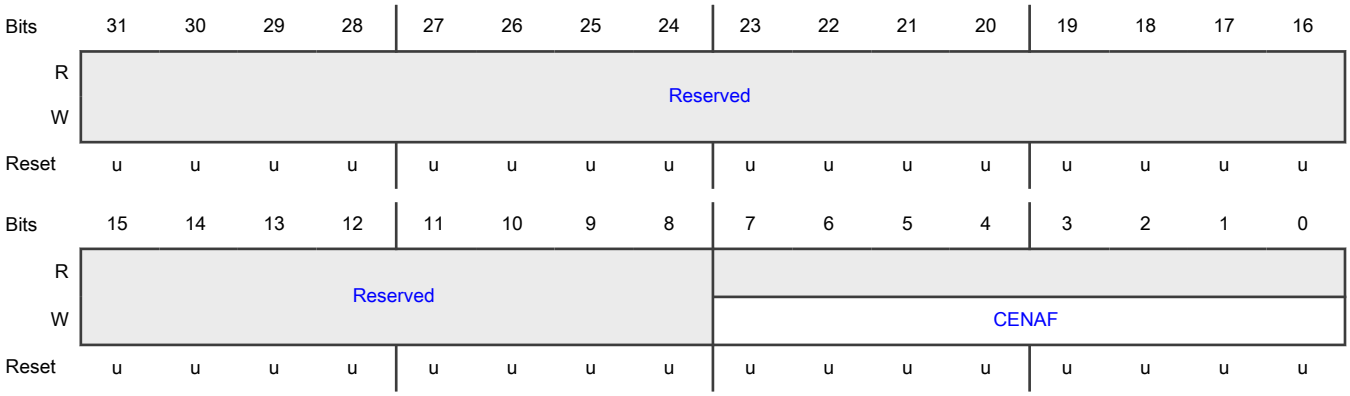
Each bit in the CIENF register sets the corresponding bit in the IENF register depending on the PMODE configured in the ISEL register.

- If the PMODE is edge sensitive (PMODE = 0), the falling edge interrupt is cleared.
- If the PMODE is level sensitive (PMODE = 1), the LOW-active interrupt is selected.

Offset

Register	Offset
CIENF	18h

Diagram



Fields

Field	Description
31-8 —	Reserved
7-0 CENAF	Clear bits in the IENF Ones written to this address clears bits in the IENF, thus disabling interrupts. Bit a clears bit n in the IENF register. **NOTE** This field is not supported in every instance. The following table includes only supported registers.

Instance	Field supported in	Field not supported in
PINT0	CIENF	—
PINT1	CIENF[1–0]	CIENF[7–2]

0000_0000 - No operation

0000_0001 - LOW-active interrupt selected or falling edge interrupt disabled

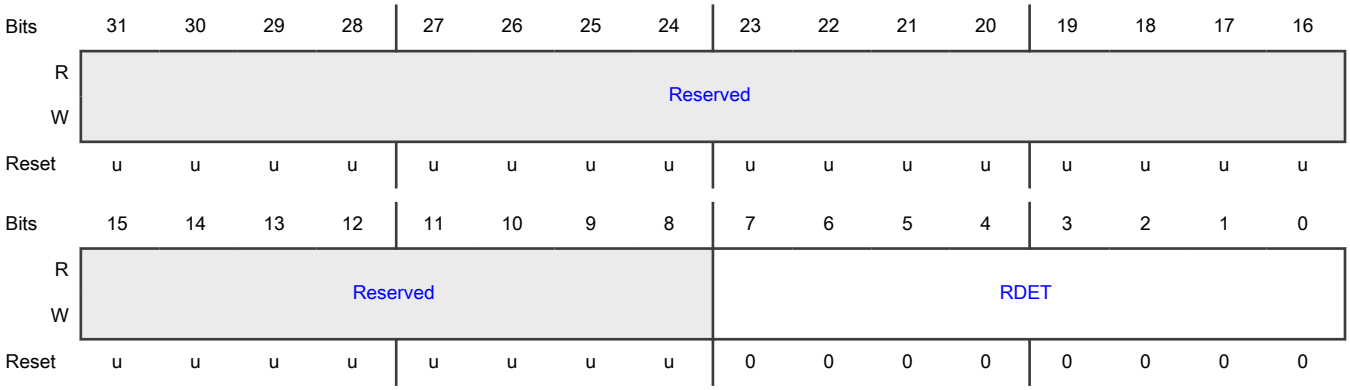
25.6.1.1.8 Pin Interrupt Rising Edge (RISE)

This register contains ones for pin interrupts selected in the PINT_SELa registers (see INPUTMUX) on which a rising edge has been detected. Writing ones to this register clears rising edge detection. Ones in this register assert an interrupt request for pins that are enabled for rising-edge interrupts. All edges are detected for all pins selected by the PINT_SEL registers, regardless of whether they are interrupt-enabled.

Offset

Register	Offset
RISE	1Ch

Diagram



Fields

Field	Description
31-8 —	Reserved
7-0 RDET	Rising edge detect Bit a detects the rising edge of the pin selected in PINT_SELa. **NOTE** This field is not supported in every instance. The following table includes only supported registers.

Instance	Field supported in	Field not supported in
PINT0	RISE	—
PINT1	RISE[1–0]	RISE[7–2]

0000_0000 - Read 0- No rising edge has been detected on this pin since Reset or the last time a one was written to this bit, Write 0- no operation

Table continued from the previous page...

Field	Description
	0000_0001 - Read 1- a rising edge has been detected since Reset or the last time a one was written to this bit, Write 1- clear rising edge detection for this pin

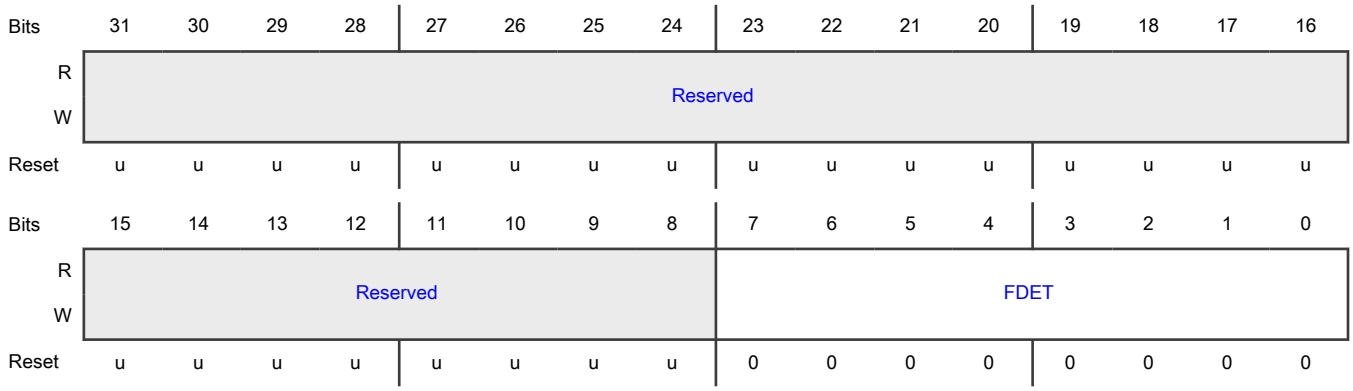
25.6.1.1.9 Pin Interrupt Falling Edge (FALL)

It contains ones for pin interrupts selected in the PINT_SELa registers on which a falling edge has been detected. Writing ones to this register clears falling edge detection. Ones in this register assert an interrupt request for pins that are enabled for falling-edge interrupts. All edges are detected for all pins selected by the PINT_SELa registers, regardless of whether they are interrupt-enabled.

Offset

Register	Offset
FALL	20h

Diagram



Fields

Field	Description
31-8 —	Reserved
7-0 FDET	Falling edge detect Bit a detects the falling edge of the pin selected in PINT_SELa. **NOTE** This field is not supported in every instance. The following table includes only supported registers.

Table continued from the previous page...

Field	Description
	<table><tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>PINT0</td><td>FALL</td><td>—</td></tr><tr><td>PINT1</td><td>FALL[1–0]</td><td>FALL[7–2]</td></tr></table> 0000_0000 - Read 0- No falling edge has been detected on this pin since Reset or the last time a one was written to this bit, Write 0- no operation 0000_0001 - Read 1- a falling edge has been detected since Reset or the last time a one was written to this bit, Write 1- clear falling edge detection for this bit

25.6.1.1.10 Pin Interrupt Status (IST)

Reading this register returns ones for pin interrupts that are currently requesting an interrupt. For pins identified as edge-sensitive in the Interrupt Select register, writing ones to this register clears both rising- and falling-edge detection for the pin. For level-sensitive pins, writing ones inverts the corresponding bit in the Active level register, thus switching the active level on the pin.

Offset

Register	Offset
IST	24h

Diagram

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

Fields

Field	Description
31-8 —	Reserved

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>7-0
PSTAT</td><td>Pin interrupt status
Bit a returns the status, clears the edge interrupt, or inverts the active level of the pin selected in PINT_SELa.
NOTE
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>PINT0</td><td>IST</td><td>—</td></tr>
<tr><td>PINT1</td><td>IST[1–0]</td><td>IST[7–2]</td></tr>
</table>
0000_0000 - Read 0- interrupt is not being requested for this pin, Write 0- no operation.
0000_0001 - Read 1- interrupt is being requested for this pin, Write 1 (edge-sensitive)- clear rising- and falling-edge detection for this pin, Write 1 (level-sensitive)- switch the active level for this pin (in the IENF register).</td></tr>
</table>

25.6.1.1.11 Pattern Match Interrupt Control (PMCTRL)

The pattern match control register contains one bit to select pattern-match interrupt generation (as opposed to pin interrupts which share the same interrupt request lines), and another to enable the RXEV output to the CPU. This register also allows the current state of any pattern matches to be read. If the pattern match feature is not used (either for interrupt generation or for RXEV assertion) bits SEL_PMATCH and ENA_RXEV of this register should be left at 0 to conserve power.

NOTE

Set up the pattern-match configuration in the PMSRC and PMCFG registers before writing to this register to enable (or re-enable) the pattern-match functionality. This eliminates the possibility of spurious interrupts as the feature is being enabled.

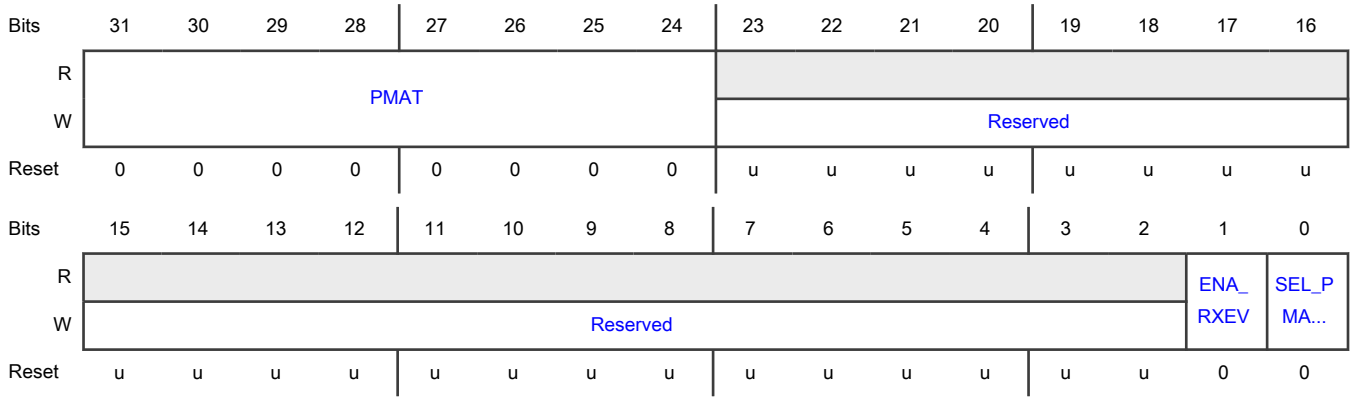
NOTE

The pattern match feature requires clocks in order to operate, and can thus not generate an interrupt or wake up the device during reduced power modes below sleep mode.

Offset

Register	Offset
PMCTRL	28h

Diagram



Fields

Field	Description
31-24 PMAT	Pattern Matches Displays the current state of pattern matches. 0000_0001 - The corresponding product term is matched by the current state of the appropriate inputs.
23-2 —	Do not write 1s to unused bits.
1 ENA_RXEV	Enables the RXEV output to the CPU and/or to a GPIO output, when the specified boolean expression evaluates to true. 0 - Disabled- RXEV output to the CPU is disabled. 1 - Enabled- RXEV output to the CPU is enabled.
0 SEL_PMATCH	Specifies whether the pin interrupts are controlled by the pin interrupt function or by the pattern match function. 0 - Pin interrupt- interrupts are driven in response to the standard pin interrupt function. 1 - Pattern match- interrupts are driven in response to pattern matches.

25.6.1.1.12 Pattern Match Interrupt Bit-Slice Source (PMSRC)

This register specifies the input source for each of the pattern match bit slices. Each of the possible 8 inputs is selected in the pin interrupt select registers in INPUTMUX. Input 0 corresponds to the pin selected in the PINT_SEL0 register, input 1 corresponds to the pin selected in the PINT_SEL1 register, and so on.

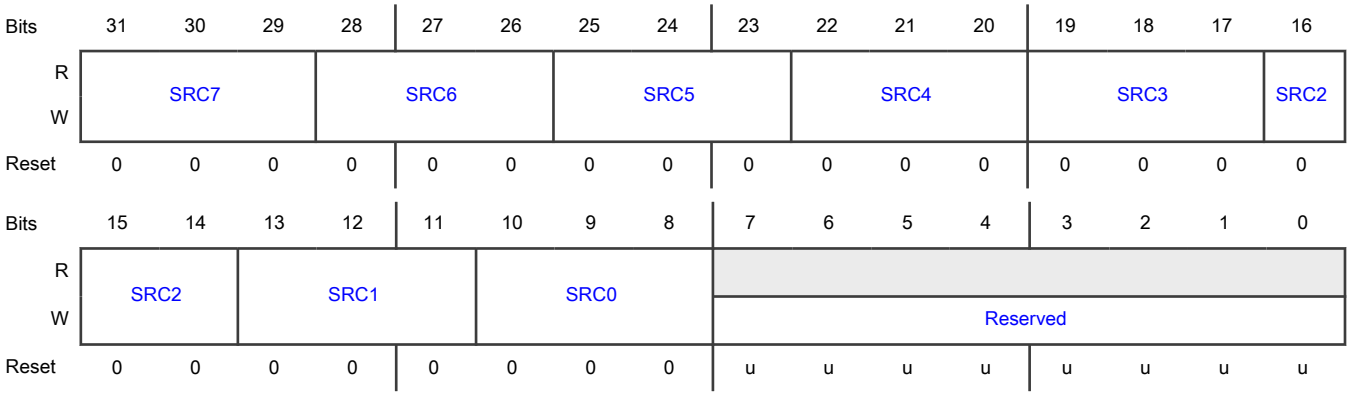
NOTE

Writing any value to either the PMCFG register or the PMSRC register, or disabling the pattern-match feature (by clearing both the SEL_PMATCH and ENA_RXEV bits in the PMCTRL register to zeros) will erase all edge-detect history.

Offset

Register	Offset
PMSRC	2Ch

Diagram



Fields

Field	Description
31-29: SRC7 28-26: SRC6 25-23: SRC5 22-20: SRC4 19-17: SRC3 16-14: SRC2 13-11: SRC1 10-8: SRC0	Selects the input source for bit slice n 000 - Input 0. Selects the pin selected in the PINT_SEL0 register as the source to bit slice n. 001 - Input 1. Selects the pin selected in the PINT_SEL1 register as the source to bit slice n. 010 - Input 2. Selects the pin selected in the PINT_SEL2 register as the source to bit slice n. 011 - Input 3. Selects the pin selected in the PINT_SEL3 register as the source to bit slice n. 100 - Input 4. Selects the pin selected in the PINT_SEL4 register as the source to bit slice n. 101 - Input 5. Selects the pin selected in the PINT_SEL5 register as the source to bit slice n. 110 - Input 6. Selects the pin selected in the PINT_SEL6 register as the source to bit slice n. 111 - Input 7. Selects the pin selected in the PINT_SEL7 register as the source to bit slice n.
7-0 —	Software should not write 1s to unused bits.

25.6.1.1.13 Pattern Match Interrupt Bit Slice Configuration (PMCFG)

The bit-slice configuration register configures the detect logic and contains bits to select from among eight alternative conditions for each bit slice that cause that bit slice to contribute to a pattern match. The seven LSBs of this register specify which bit-slices are the end-points of product terms in the boolean expression (where OR terms are to be inserted in the expression).

Two types of edge detection on each input are possible:

- Sticky: A rising edge, a falling edge, or a rising or falling edge that is detected at any time after the edge-detection mechanism has been cleared. The input qualifies as detected (the detect logic output remains HIGH) until the pattern match engine detect logic is cleared again.

- Non-sticky: Every time an edge (rising or falling) is detected, the detect logic output for this pin goes HIGH. This bit is cleared after one clock cycle, and the edge detect logic can detect another edge.

To clear the pattern match engine detect logic, write any value to either the PMCFG register or the PMSRC register, or disable the pattern-match feature (by clearing both the SEL_PMATCH and ENA_RXEV bits in the PMCTRL register to zeros). This will erase all edge-detect history.

To select whether a slice marks the final component in a minterm of the boolean expression, write a 1 in the corresponding PROD_ENPTSa bit. Setting a term as the final component has two effects:

1. The interrupt request associated with this bit slice will be asserted whenever a match to that product term is detected.
2. The next bit slice will start a new, independent product term in the boolean expression, an OR will be inserted in the boolean expression following the element controlled by this bit slice.

Offset

Register	Offset
PMCFG	30h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CFG7				CFG6				CFG5				CFG4			
Reset	0	0	0	0	0	0	0	0	0	0	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	CFG2			CFG1				CFG0	Reserved	PROD_EN...	PROD_EN...	PROD_EN...	PROD_EN...	PROD_EN...	PROD_EN...	PROD_EN...
Reset	0	0	0	0	0	0	0	0	u	0	0	0	0	0	0	0

Fields

Field	Description
31-29: CFG7 28-26: CFG6 25-23: CFG5 22-20: CFG4 19-17: CFG3 16-14: CFG2 13-11: CFG1 10-8: CFG0	Specifies the match contribution condition for bit slice n. 000 - Constant HIGH. This bit slice always contributes to a product term match. 001 - Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This match condition is only cleared when the PMCFG or the PMSRC registers are written to. 010 - Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This match condition is only cleared when the PMCFG or the PMSRC registers are written to. 011 - Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This match condition is only cleared when the PMCFG or the PMSRC registers are written to.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	100 - High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register. 101 - Low level. Match occurs when there is a low level on the specified input. 110 - Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices). 111 - Event. Non-sticky rising or falling edge. Match occurs on an event when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after 1 clock cycle.
7 ——	Bit slice 7 is automatically considered a product end point.
6-0 PROD_ENDPTSn	Determines whether slice n is an endpoint. 0 - No effect. Slice 0 is not an endpoint. 1 - Endpoint. Slice 0 is the endpoint of a product term (minterm). Pin interrupt 0 in the NVIC is raised if the minterm evaluates as true.

Chapter 26
Non-Secure Boot ROM

26.1 Overview

This device supports both on-chip FLASH image boot and an external NOR FLASH image boot via FlexSPI interface. User can use ROM to erase/program/read of the on-chip or external FLASH, that means ROM can be used to download the boot image into the on-chip or external FLASH via the ISP interfaces.

Also, ROM takes responsibility for the boot flow. ROM decides to boot from on-chip FLASH or external FLASH or ISP mode.

ROM also can be used to update those settings in ISP mode or using ROM API in application to do these updates.

There are four boot modes for ROM; ROM uses the ISP pins or CMPA configuration to select the boot mode for on-chip flash boot, FlexSPI boot or ISP boot, or Autoboot mode. For more details, see Boot mode and ISP download modes based on ISP pins.

26.1.1 Features

192 KB on-chip boot ROM with bootloader that allows various boot options and APIs:

- Based on ISP pins or CMPA setting in PFR region, supports automated booting from internal flash.

26.2 Functional description

The internal ROM memory is used to store the boot code. After a reset, the Arm processor starts its code execution from this memory. The bootloader code is executed every time the part is powered-on, reset, or wakes up from a deep power-down while in a low power mode.

Images must be stored in the internal flash because the LPC553x has an internal flash for code and data storage or in the Serial NOR connected to the FLEXSPI controller. The code is then validated, and the boot ROM vectors to on-chip flash or the off-chip FLASH.

Depending on the values of the CMPA bits, ISP pin, and the image header type definition, the bootloader decides whether to boot from internal flash, off-chip flash or run into ISP mode. The LPC553x reads the status of the ISP pins to determine the boot source. See the table below.

Table 116. Boot mode and ISP download modes based on ISP pins

Boot mode	ISP1 (PIO0_7 pin)	ISP0 (PIO0_5 pin)	Description
Internal FLASH boot	LOW	LOW	The LPC553x looks for a valid image from the internal flash; if no valid image, the LPC553x looks for a valid image in the recovery boot device if recovery boot is enabled(4:0, word 1 in CMPA) in CMPA, if still fail, the LPC553x will enter ISP boot mode based on DEFAULT_ISP_MODE bits defined in the table below.
ISP boot	LOW	HIGH	One of the serial interfaces (UART0, I^2C1, HS_SPI, USB0-FS, CAN) is used to download the image from the host into internal flash. The first valid probe message on USART, I^2C, SPI, CAN, or USB locks in that interface.
FLEXSPI boot	HIGH	LOW	The LPC553x looks for a valid image in the external flash; if no valid image, the LPC553x looks for a valid image in the recovery boot device if recovery boot is enabled(4:0, word 1 in CMPA) in CMPA, if still fail, the LPC553x will enter ISP boot mode based on DEFAULT_ISP_MODE bits defined in the table below.

Table continues on the next page...

Table 116. Boot mode and ISP download modes based on ISP pins (continued)

Boot mode	ISP1 (PIO0_7 pin)	ISP0 (PIO0_5 pin)	Description
AUTO boot	HIGH	HIGH	The LPC553x looks for a valid image in the internal flash; if no valid image, the LPC553x looks for a valid image in the external NOR flash; if no valid image, the LPC553x looks for a valid image in recovery boot device if recovery boot is enabled d(4:0, word 1 in CMPA) in CMPA, if still fail, the LPC553x will enter ISP boot mode based on DEFAULT_ISP_MODE bits.

Table 117. ISP download mode based on DEFAULT_ISP_MODE bits (6:4, word 0 in CMPA)

ISP Boot mode	ISP_MODE_2	ISP_MODE_1	ISP_MODE_0	Description
Auto ISP	0	0	0	The LPC553x probes the active peripheral from one of the below serial interfaces and download image from the probed peripherals: UART0, I^2C1, HS_SPI, USB0-FS, CAN
USB HID ISP	0	0	1	The USB HID class is used to download the image of the USB0 port.
UART ISP	0	1	0	The UART is used to download the image.
SPI Slave ISP	0	1	1	The SPI slave (HS-SPI) is used to download the image.
I^2C Slave ISP	1	0	0	The I^2C slave is used to download the image.
CAN Slave ISP1	1	0	1	The CAN slave is used to download the image
Disable ISP	1	1	1	Disable ISP mode.

Table below shows the ISP pin assignments and is the default pin assignment used by the ROM code that cannot be changed.

Table 118. ISP pin assignments

ISP pin	Port pin assignment
ISP0 ISP1	PIO0_5 PIO0_7
USART ISP mode	
FC0_TXD	PIO0_30
FC0_RXD	PIO0_29
I^2C ISP mode	
FC1_SDA	PIO0_13

Table continues on the next page...

Table 118. ISP pin assignments (continued)

ISP pin	Port pin assignment
FC1_SCL	PIO0_14
SPI Flash Recovery mode	
FC0_TXD_SCL_MISO_WS	PIO0_30
FC0_RXD_SDA_MOSI_DATA	PIO0_24
FC0_CTS_SDA_SSELN0	PIO0_31
FC0_SCK	PIO0_28
HS_SPI_SCK	PIO1_2
HS_SPI_SSEL1	PIO1_1
HS_SPI_MISO	PIO1_3
HS_SPI_MOSI	PIO0_26
FC3_TXD_SCL_MISO_WS	PIO0_2
FC3_RXD_SDA_MOSI_DATA	PIO0_3
FC3_CTS_SDA_SSELN0	PIO0_4
FC3_SCK	PIO0_6
SPI ISP mode	
HS_SPI_SCK	PIO1_2
HS_SPI_SSEL1	PIO1_1
HS_SPI_MISO	PIO1_3
HS_SPI_MOSI	PIO0_26
USB0-FS ISP mode	
USB0_VBUS	PIO1_31(if [bit_22] @0x3FC14 is set to 1, use P0_22)
USB0_DP	
USB0_DM	
CAN ISP mode	
CAN_TXD	PIO0_30

Table continues on the next page...

Table 118. ISP pin assignments (continued)

ISP pin	Port pin assignment
CAN_RXD	PIO1_22
FlexSPI Pins	
FLEXSPI_SSEL	PIO0_21
FLEXSPI_CLK	PIO0_19
FLEXSPI_DQS	PIO0_25
FLEXSPI_CLK_N	PIO0_22
FLEXSPI_D0	PIO0_6
FLEXSPI_D1	PIO0_4
FLEXSPI_D2	PIO0_3
FLEXSPI_D3	PIO0_2
FLEXSPI_D4	PIO1_16
FLEXSPI_D5	PIO1_15
FLEXSPI_D6	PIO1_27
FLEXSPI_D7	PIO1_29

High-level boot flow shows the top-level boot process. The boot starts after reset is released.

The CPU clock is 96 MHz based on the 192MHz FRO. When the Cortex-M33 starts the bootloader, the SWD access is disabled, and therefore, the debugger cannot connect to the CPU during this period. The boot ROM determines the boot mode based on the reset state of the ISP pins.

After finishing the boot mode detection, the bootloader starts validating the vector table and image header if the image is present in the boot media (internal FLASH or external FLASH).

The boot ROM checks the following for image validity check:

- Validate image using CRC32 if the CRC check is present in the image header.
- Validate the SP and PC if neither the integrity check nor authentication check is enabled.
- Validate the TZM image type if the above image checks pass.

The beginning of the image follows the format mentioned in Table 119. T he bootloader begins scanning for user images by examining the image type marker located at 0x0000_0024. If the value matches any supported image type markers, then the image header's validation will begin. After completing the image header's validation, the qualification continues by examining the TZM image type field.

If it is a CRC image, then the image length field value is used as the length to perform a CRC ON. See Table 119. The CRC is performed on the image in internal flash. The CRC calculation begins at offset 0x0 from the beginning of the image sector and continues up to the number of bytes specified by the length. The length does not include the *offsetToExtendedHeader* field that makes up the CRC value field, which means that the CRC calculated skips the CRC value field. The result is then compared to the *offsetToExtendedHeader* entry in the structure and the image is considered valid if a match exists; otherwise, the image is considered invalid. CRC is not performed if the image is not a CRC image.

Table 119. Image header

Offset	Size in bytes	Symbol	Description
0x00	4	Initial SP	Stack pointer.
0x04	4	Initial PC	The application first execution instruction.
0x08	24	Vector table	Cortex-M33 Vector table entries.
0x20	4	image length	The length of the current image (Total length, including the signature, etc.) Set to 0 if the image type is 0 as well Set to actual image length if the image type is other value.
0x24	4	imageType	Image Type Bit [7:0] 0x0: plain image 0x2: plain image with CRC 0x4: Xip plain signed 0x5: Xip plain with CRC Other values are reserved. Bit[10] - Image version included in ImageType[31:16] 0 - Image version is not in the ImageType 1 - Image version is included in the Image Type, applicable to internal FLASH XIP use case only Bit[13] Bits[31:16] Image version(for on chip flash) when bit[10] is set.
0x28	4	offsetToExtendedHeader	Offset to extended header It means the crcChecksum if the image type is 0x02 or 0x05.
0x2C	8	Vector table	Cortex-M33 Vector table entries
0x34	4	imageExecutionAddress	The execution address of the image Set to 0 if image type is XIP Set to actual image execution address if the image type is load to RAM
0x38	8	Vector table	Cortex-M33 Vector table entries

26.2.1 Top-level boot flow

Figure 84 shows the top-level boot process. The boot starts after Reset is released.

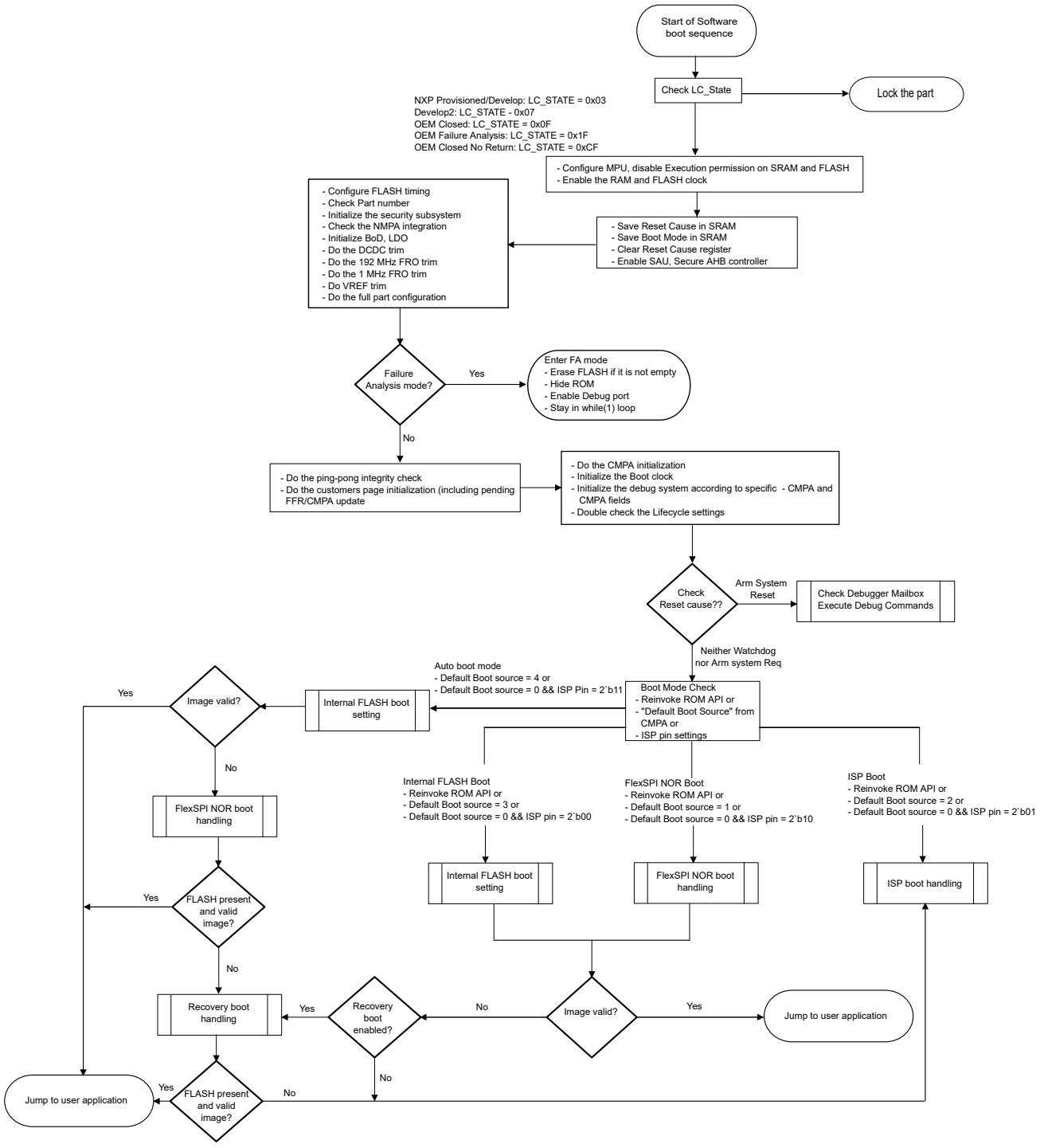


Figure 84. Top-level boot flow diagram

26.2.2 SPI flash recovery

Support is provided for a recovery boot from an external 1-bit SPI flash device

active="1'd0"When booting from internal flash or external flash using FlexSPI, if the flash image is deemed invalid, the device checks the SPI_RECOVERY_BOOT_EN (bits 1:0) in protected flash SPI_CAN_CFG (0x3E204) to determine if SPI flash

recovery is enabled. If SPI flash recovery is enabled, the boot ROM tests SEC_BOOT_EN (bits 31:30) in protected flash SECURE_BOOT_CFG (0x3E21C).

The following are the descriptions for the commands used in the above code:

- kSB3_CmdErase: call the memory interface erase to internal flash, IFR or external flash.
- kSB3_CmdLoad: call the memory interface program to internal flash, IFR or external flash.
- kSB3_CmdExecute:Execute the application with the transferred start address.
- kSB3_CmdProgramFuse: efuse program command.
- kSB3_CmdProgramIFR: Same as the kSB3_CmdLoad.
- kSB3_CmdCopy: Copy the data from src address to dest address(RAM or flash).
- kSB3_CmdLoadKeyBlob:Program the KeyStore data intoKeyStore region in the IFR.
- kSB3_CmdConfigMem: Call the memory interface to configure the internal or external flash.
- kSB3_CmdFillMem:Call the memory interface to fill the internal or external flash.
- kSB3_CmdFwVerCheck: Compare the CFPA offset 0x8 Secure_FW_Version or offset 0xc NS_FW_Version with the sbfile command version based on the command ID.

When the command ID is 0x1(kFwVerChk_Id_nonsecure) or 0x2(kFwVerChk_Id_secure), the sbfile command version will compare with the NS_FW_Version or Secure_FW_Version. If the SB file command version is smaller than the FW_Version, the sb file process will return the kStatusRomLdrRollbackBlocked.

The device boots a plain text image with the boot address and image length specified in the image header. For plain text SPI flash recovery, the image can only be booted into internal RAM in a non-reserved region. To determine reserved regions of RAM, use the following command in ISP mode:

```
blhost -p COM3 get-property 12
```

See 1-bit Serial NOR flash through SPI for more details.

26.3 Boot modes

The boot modes include:

- Master Boot mode
- ISP boot mode
- Recovery boot

26.3.1 Master Boot mode

The following boot devices are supported under Master boot mode:

• Internal flash boot

• FlexSPI NOR FLASH boot

• SPI 1-bit NOR recovery boot

Table 120. Image offset on different boot media

Boot media	Image Offset
Internal flash boot	0x0
FlexSPI NOR FLASH boot	0x1000

Table continues on the next page...

Table 120. Image offset on different boot media (continued)

SPI 1-bit NOR recovery Boot	0x0

The CPU clock is set to the boot speed specified in CMPA field and will boot directly from internal flash/or external FlexSPI NOR flash based on boot mode.

26.3.1.1 Internal flash boot

The CPU clock is set to the boot speed specified in CMPA field and will boot directly from the internal flash. Once the mode is selected as the internal flash boot, Rom tried to find the valid boot image from the internal flash.

26.3.1.1.1 Internal Flash dual image boot

ROM supports the dual image boot for internal flash, that means, in the flash region, two boot image can be placed there; ROM decides to boot which image based on the image version, boot the one with the newer image version first, if fail, boot the older one. The image version is the image header offset 0x24; bit 10 shows whether the image contains the image version for not; if bit10 is 0, that means the image has no image version; ROM will take the image version as 0.

Table 121. Internal FLASH boot image version(image header offset: 0x24)

image_type	Field Name	Field Description
bit 31 -- bit 16	Image Version	active when remap feature enable(remap offset and remap size not zero in cmpa)
bit 10	IMG_VER_INCLUDED_IN_IMG_TYPE	0: image has no version 1: image has a version in the dual image boot

26.3.1.1.1.1 Internal Flash controller remap feature

Below is a simple figure about the Internal FLASH remap function. When set the remap offset, Internal FLASH memory AHB access will change the access address adding the offset as the below figure shows. For example, when the offset is set to 128K(0x20000), the access to 0x0 will be remapped to 0x20000. Via this IP feature, ROM can implement a dual image boot with two images. The offset and the remap size of the image is set in CMPA region by the user.

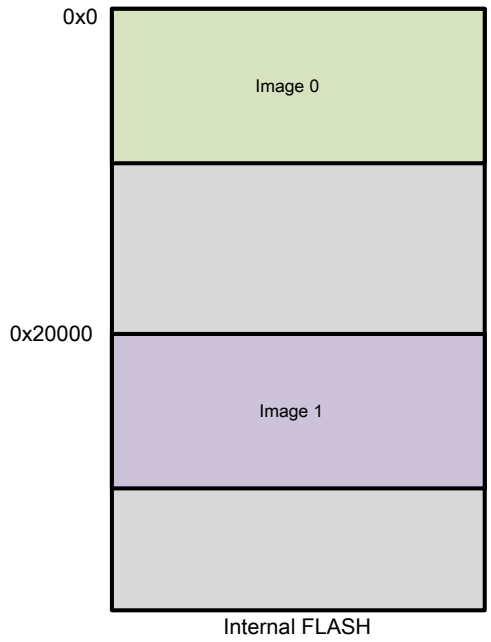


Figure 85. Internal FLASH dual image layout

26.3.1.1.1.2 Internal Flash dual image boot remap settings

The below table shows the dual image boot setting in CMPA. The user needs to update the below setting in CMPA to let ROM knows where the second image is put, and then the dual image boot can be enabled by setting the dual image boot image version in the image header as the above chapters describe.

Table 122. Boot image 1 remap offset and size (CMPA:0x3E238, 0x3E23C)

CMPA Word	FLASH Address	Word Name	Definition
Word14	3E238	FLASH_REMAP_SIZE	This is to tell ROM the flash remap size from the flash address 0x0
Word15	3E23C	FLASH_REMAP_OFFSET	This is to tell ROM the flash remap offset of image 1; ROM tried to find image 1 at this offset

26.3.1.2 FlexSPI NOR flash boot

The ROM supports access to different Quad/Octal SPI NOR Flash devices from various vendors via the FlexSPI interface using 1-bit, 2-bit (dual), 4-bit (quad), or 8-bit (octal) mode by employing the Flash configuration block(FCB) located at offset 0x400 on the Flash device or the Flash Auto-probe feature specified in CMPA.

- If FLEXSPI_AUTO_PROBE_EN is blown, the bootloader will perform Flash auto-probe sequence using parameters blown in FLEXSPI_FLASH_PROBE_TYPE

  field which defines the Flash Auto-probe type, FLEXSPI_FLASH_TYPE field which defines the Flash-type and FLEXSPI_FLASH_BOOT_FREQ which defines the Flash access speed.
- If not blown, the bootloader will look at offset 0x400 on the Flash device; if data at offset 0x400 equal to 0x42464346, the bootloader will read the whole 512-byte FCB into on-chip SRAM and configure the FlexSPI controller using this FCB accordingly.

After the above operation, the bootloader starts to perform the normal boot flow. See Figure 86 for more details.

Table 123. FlexSPI flash configuration block(FCB)

Field	Offset	Size (bytes)	Description
tag	0x000	4	Config Block tag, must be 0x42464346
version	0x004	4	Configuration block, used by the Boot ROM internally, fixed to 0x56020000
reserved	0x008	4	Reserved for future use
readSampleClkSrc	0x00c	1	Read Sampling Clock source option 0 - Internal sampling 1 - Loopback from DQS Pad 2 - Loopback from SCK Pad 3 - External DQS signal
csHoldTime	0x00d	1	CS Hold time in terms of Flash clock cycles Recommended value is 3
csSetupTIme	0x00e	1	CS setup time in terms of Flash clock cycles Recommended value is 3
columAddressWidth	0x00f	1	Column Address Width Set to 3 for HyperFLASH Set to 0 for other FLASH devices
deviceModeCfgEnable	0x010	1	Enable the Device Mode Configuration sequence
deviceModeType	0x011	1	Argument for the Device Mode Configuration. For example, Quad Enable setting in Status Register2 for some QSPI FLASH devices
waitTimeCfgCommands	0x012	2	Wait time is terms of 100 us for the Device Mode Config Command
deviceModeSeq	0x014	4	Device Mode Configuration Sequence byte 0: Number of required sequences byte 1: Sequence Index
deviceModeArg	0x018	4	Argument for the Device Mode Configuration.

Table continues on the next page...

Table 123. FlexSPI flash configuration block(FCB) (continued)

Field	Offset	Size (bytes)	Description
			For example, Quad Enable setting in Status Register2 for some QSPI FLASH devices
configCmdEnable	0x01c	1	Configuration Command Enable 0 - Ignore Configuration Command 1 - Enable Configuration Command
configModeType	0x01d	3	Config command types, support up to 3 types, each type is combined with a config command sequence
configCmdSeqs	0x020	12	Config command types Support up to 3 sequences
reserved	0x02c	4	Reserved for future use
configCmdArgs	0x030	12	Configure Command Argument Support up to 3 arguments
reserved	0x03c	4	Reserved for future use
controllerMiscOption	0x040	4	Miscellaneous Controller Configuration options bit 0 - Differential clock enable, set to 1 for HyperFLASH 1V8 device, set to 0 for other devices bit 3 - WordAddressableEnable, set to 1 for HyperFLASH, set to 0 for other devices bit 4 - SafeConfigFreqEnable, set to 1 if expecting to configure device with safe frequency bit 6 - DDR mode enable, set to 1 if DDR read is expected Other bits - Reserved, set to 0
deviceType	0x044	1	Device Type 1 - Serial NOR
sflashPadType	0x045	1	Data pad used in Read command 1 - Single pad 2 - Dual pads 4 - Quad pads 8 - Octal pads

Table continues on the next page...

Table 123. FlexSPI flash configuration block(FCB) (continued)

Field	Offset	Size (bytes)	Description
serialClkFreq	0x046	1	SDR: 1 - 30MHz 2 - 50MHz 3 - 60MHz 4 - 75MHz 5 - 100MHz DDR: 1 - 30MHz 2 - 50MHz
lutCustomSeqEnable	0x047	1	LUT customization Enable, it is required if the program/erase cannot be done using 1 LUT sequence, currently, only applicable to HyperFLASH
reserved	0x048	8	Reserved for future use
sflashA1Size	0x050	4	Size of Flash connected to A1
sflashA2Size	0x054	4	Size of Flash connected to A2
sflashB1Size	0x058	4	Size of Flash connected to B1
sflashB2Size	0x05c	4	Size of Flash connected to B2
csPadSettingOverride	0x060	4	Pad override value for CS pin Using this value to configure CS pin if it is non-zero, otherwise use default ROM setting
sclkPadSettingOverride	0x064	4	Sck pad setting override value
dataPadSettingOverride	0x068	4	Data pad setting override value
dqsPadSettingOverride	0x06c	4	Dqs pad setting override value
timeoutInMs	0x070	4	Timeout value in terms of ms to terminate the busy check
commandInterval	0x074	4	CS deselect interval between two commands
dataValidTime	0x078	4	CLK edge to data valid time
busyOffset	0x07c	2	Busy offset, valid value: 0-31

Table continues on the next page...

Table 123. FlexSPI flash configuration block(FCB) (continued)

Field	Offset	Size (bytes)	Description
busyBitPolarity	0x07e	2	Busy flag polarity, 0 - busy flag is 1 when flash device is busy, 1 -busy flag is 0 when flash device is busy
lookupTable	0x080	256	16 LUT sequences each sequence consists of 4 words, see FlexSPI chapter for more details.
lutCustomSeq	0x180	48	Customizable LUT Sequences
reserved	0x1b0	16	Reserved for future use
pageSize	0x1c0	4	Page size of Flash
sectorSize	0x1c4	4	Sector size of Flash
ipcmdSerialClkFreq	0x1c8	1	Serial Clock Frequency for IP command 0 - The same with Read command others - the same definition as serialClkFreq
isUniformBlockSize	0x1c9	1	If the sector size is the same with block size. 0 - No 1 - Yes
isDataOrderSwapped	0x1ca	1	Data order (D0, D1, D2, D3) is swapped (D1,D0, D3, D2)
reserved	0x1cb	1	
serialNorType	0x1cc	1	Serial NOR Flash type: 0/1/2/3
needExitNoCmdMode	0x1cd	1	Need to exit NoCmd mode before other IP command
halfClkForNonReadCmd	0x1ce	1	Half the Serial Clock for non-read command: true/false
needRestoreNoCmdMode	0x1cf	1	Need to Restore NoCmd mode after IP commmand execution
blockSize	0x1d0	4	Flash Block size
flashStateCtx	0x1d4	4	Flash State Context
reserved	0x1d8	40	Reserved for future use

26.3.1.2.1 FlexSPI Flash reset

During FlexSPI boot, the boot process requires the FlexSPI Flash device to be in a certain mode, for example, 1-bit SPI compatible mode. The Flash device will naturally be in this mode after a POR reset because the power-up sequence will reset it with the device together. However, the Flash device will not be in 1-bit SPI compatible mode if the Flash device is configured to DPI mode or QPI mode or Octal mode when any non-POR resets happen, for example, watchdog timer and external pin reset. In such a case, special processing is required by the boot process to restore the Flash device to 1-bit SPI compatible mode before continuing access to the Flash device. In general, this can be achieved by using a GPIO to assert a reset pin on the Flash device. The bootloader can perform the reset process and reset the Flash device to 1-bit SPI compatible mode if the FLEXSPI_RESET_PIN_EN is blown, using the GPIO specified by the combination of FLEXSPI_RESET_PIN_PORT and FLEXSPI_RESET_PIN_GPIO.

NOTE

For Octal flash, once ROM boots up, the FLASH will be configured to Octal mode. ROM cannot reset the NOR FLASH from Octal mode to stand SPI mode, if it is a pin/soft reset that causes reboot fail. The only way to solve this is to use a GPIO to reset the Octal NOR FLASH during reboot. There is no such issue for Quad FLASH.

Table 124. FlexSPI boot configurations in CMPA WORD32(0x3E280)(FLEXSPI_BOOT_CFG[0])

Field Name	Enum Name	Description	Offset	Width	Value
FLEXSPI_AUTO_PROBE_EN		Quad/Octal-SPI flash auto probe feature enable.	0	1	
FLEXSPI_PROBE_TYPE		Quad/Octal-SPI flash probe type.	1	3	
	QUADSPI_NOR	QuadSPI NOR			b'000
	MICRON_OCTAL	Micron Octal FLASH			b'001
	MACRONIX_OCTAL	Macronix Octal FLASH			b'010
	ADESTO_OCTAL	Adesto Octal FLASH			b'011
		Reserved			b'100
		Reserved			b'101
		Reserved			b'110
		Reserved			b'111
FLEXSPI_FLASH_TYPE		Define typical Serial NOR Flash types	4	3	
	QSPI_ADDR_3B	Device supports 3B read by default			b'000
		Reserved.			b'001
	HYPER_1V8	HyperFLASH 1V8			b'010
	HYPER_3V3	HyperFLASH 3V3			b'011

Table continues on the next page...

Table 124. FlexSPI boot configurations in CMPA WORD32(0x3E280)(FLEXSPI_BOOT_CFG[0]) (continued)

Field Name	Enum Name	Description	Offset	Width	Value
	OSPI_DDR_MXIC	MXIC Octal DDR			b'100
	OSPI_DDR_MICRON	Micron Octal DDR			b'101
		Reserved			b'110
		Reserved			b'111
FLEXSPI_DUMMY_CYCLES		Quad/Octal-SPI dummy cycles for read command.	7	4	
	PROBE_DUMMY_CYCLE	The dummy cycles are probed automatically			b’0000
		Number of dummy cycle is specified by this field			Other value
FLEXSPI_FREQUENCY		Q/O-SPI flash interface frequency.	11	3	
	FLEXSPI_75MHZ	75MHz			b'000
	FLEXSPI_60MHZ	60MHz			b'001
	FLEXSPI_50MHZ	50MHz			b'010
	FLEXSPI_100MHZ	100MHz			b'011
		Reserved			
		Reserved			
		Reserved			
		Reserved			
FLEXSPI_RESET_PIN_ENABLE		Use FLEXSPI_RESET_PIN to reset the flash device.	14	1	
	NO_RESET	O/QSPI device reset pin is not connected or available.			
	EN_RESET	O/QSPI device reset pin is connected to a GPIO (FLEXSPI_RESET_PIN).			
FLEXSPI_RESET_PIN		GPIO port and pin number to use for O/QSPI reset function.	15	8	

Table continues on the next page...

Table 124. FlexSPI boot configurations in CMPA WORD32(0x3E280)(FLEXSPI_BOOT_CFG[0]) (continued)

Field Name	Enum Name	Description	Offset	Width	Value
	GPIO_PORT	Defines GPIO port.	0	3	
	GPIO_PIN_NUM	Defines GPIO pin number.	3	5	
FLEXSPI_HOLD TIME		Wait time before access to Serial Flash.	23	2	
	WAIT_500_US	Wait for 500 micro seconds.			b'00
	WAIT_1_MS	Wait for 1 milli seconds.			b'01
	WAIT_3_MS	Wait for 3 milli seconds.			b'10
	WAIT_10_MS	Wait for 10 milli seconds.			b'11
FLEXSPI_PWR_HOLD_TIME		Delay after POR before accessing Quad/Octal-SPI flash devices in addition to delay defined by QSPI_HOLD TIME field.	25	4	
	NO_DELAY	No delay			b'0000
	100US_DELAY	Waits additional 100 micro-seconds.			b'0001
	500US_DELAY	Waits additional 500 micro-seconds.			b'0010
	1MS_DELAY	Waits additional 1 milli-seconds.			b'0011
	10MS_DELAY	Waits additional 10 milli-seconds.			b'0100
	20MS_DELAY	Waits additional 20 milli-seconds.			b'0101
	40MS_DELAY	Waits additional 40 milli-seconds.			b'0110
	60MS_DELAY	Waits additional 60 milli-seconds.			b'0111
	80MS_DELAY	Waits additional 80 milli-seconds.			b'1000
	100MS_DELAY	Waits additional 100 milli-seconds.			b'1001
	120MS_DELAY	Waits additional 120 milli-seconds.			b'1010
	140MS_DELAY	Waits additional 140 milli-seconds.			b'1011

Table continues on the next page...

Table 124. FlexSPI boot configurations in CMPA WORD32(0x3E280)(FLEXSPI_BOOT_CFG[0]) (continued)

Field Name	Enum Name	Description	Offset	Width	Value
	160MS_DELAY	Waits additional 160 milli-seconds.			b'1100
	180MS_DELAY	Waits additional 180 milli-seconds.			b'1101
	200MS_DELAY	Waits additional 200 milli-seconds.			b'1110
	220MS_DELAY	Waits additional 220 milli-seconds.			b'1111
RESERVED			29	3	

26.3.1.2.2 FlexSPI Boot flow

The below Figure illustrates the brief FLEXSPI Boot flow. It shows the details of normal image load and authentication flow.

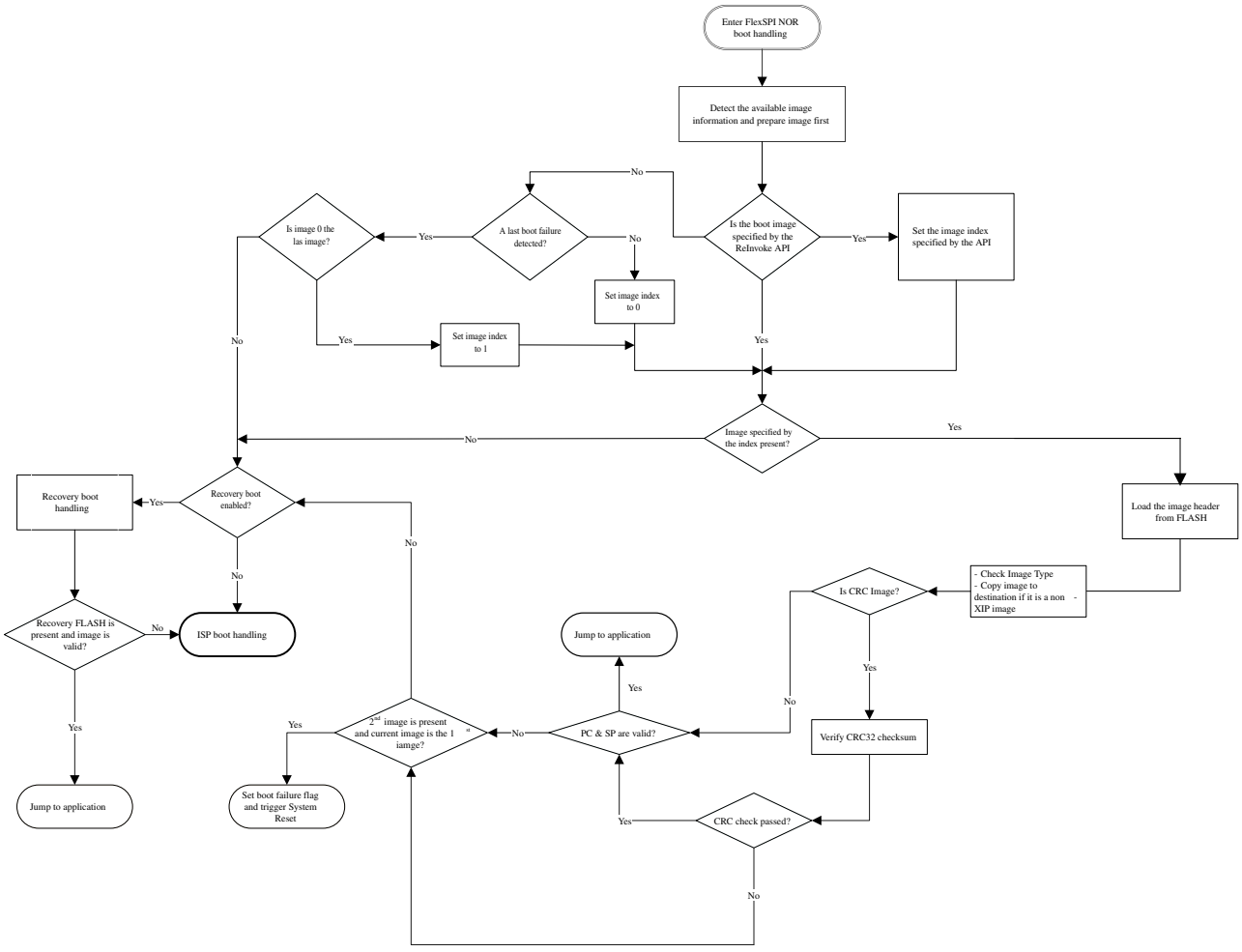


Figure 86. FLEXSPI NOR boot flow

NOTE

- The FLEXSPI_BOOT_CFG[0] and FLEXSPI_BOOT_CFG[1] fields in CMPA specifies the second image - offset and size, if both are 0, it means the 2nd image is not present.
- If the image version at each FLASH offset 0x600 is valid, use the image version, otherwise, the image version is treated as 0.
- In the image list, the newer one is image 0, and the older image is 1, if the image version information is missing or equal, the image starting from address 0 is treated as image 0, and the image specified by FLEXSPI_BOOT_CFG[0] and FLEXSPI_BOOT_CFG[1] fields are treated as image 1.

26.3.1.2.3 FLEXSPI Dual Image Ping-Pong Boot

FLEXSPI controller supports the remap function, which can remap the AHB accessing the address of the serial NOR flash to another address with the offset; in this way, FLEXSPI boot supports second image boot when the first image boots fail.

26.3.1.2.4 The basic function of the FLEXSPI remap

Below is a simple figure about the FLEXSPI remap function. When set the remap offset, FLEXSPI memory access will change the access address adding the offset as the below figure shows. For example, when the offset is set to 2MB(0x200000), the access

to 0x08000000 via FLEXSPI will be remapped to 0x08200000. Via this IP feature, ROM can implement a dual image boot with two images. The offset and the remap size of the image are set in CMPA region by the user.

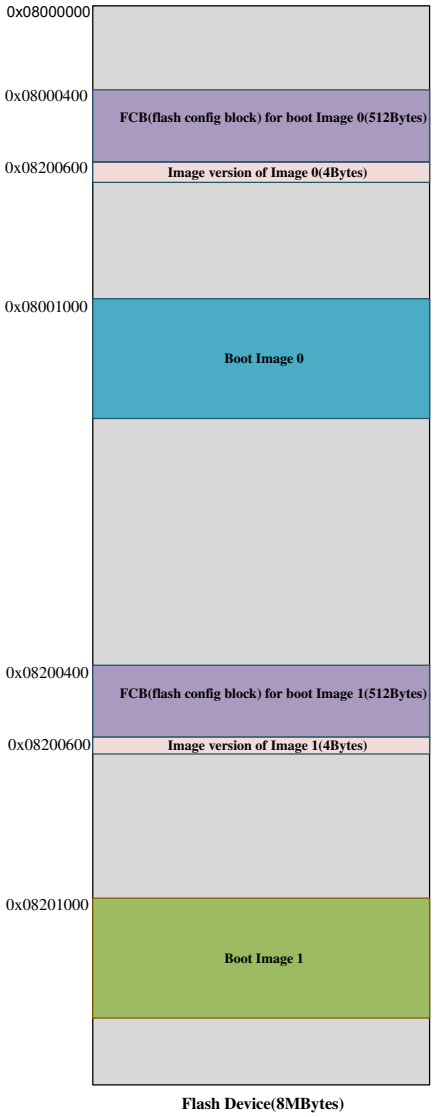


Figure 87. FLEXSPI image remap (Offset=0x200000)

26.3.1.2.5 FLEXSPI image remap offset and boot image remap size configuration in CMPA

The FLEXSPI dual image boot related configurations are allocated in CMPA word33(0x3E284) *FLEXSPI_BOOT_CFG [1]* word, more details please refer to below table

Table 125. Boot image 1 remap offset(CMPA:0x3E284)

Filed Name	Bit offset	Width	Value	Description
Second image offset	7	10	x	x * 256KB

Table 126. Boot Image size (FLEXSPI_BOOT_CFG [1])

Filed Name	Bit offset	Width	Value	Description
FlexSPI remap size	17	4	0	Same size of second image offset
			13	256KB
			14	512KB
			15	768KB
			x	x*1MB

NOTE

FlexSPI remap size cannot exceed the start address of boot Image 1.

26.3.1.2.6 Dual image ping-pong boot

For dual image ping-pong boot, each of the boot images has its image version, which is used for ROM to first select the latest boot image. If the latest image boot fails, the old image will be used to boot again.

26.3.1.2.7 Boot image version

Each boot image version is put at the offset *0x600* of the FLEXSPI boot device, as *Figure3* shows.

Image version is used 4 bytes to define, the lower 2 bytes are the real image version number, and the upper 2 bytes are the invert value of lower 2 bytes image version, all other values which do not follow this rule will be regarded as an invalid value, if the boot image version is not valid, ROM will not boot that image at all.

But if the image version is all 0xFFFFFFFF (for the condition that the user does not program the image version) it will also be considered as a valid value. When one image version is 0xFFFFFFFF and the other one is a valid image version, ROM always first boot the image with the valid value not the one with image version as 0xFFFFFFFF. If both image version is 0xFFFFFFFF, ROM always boots the first image resides at lower address in the FLEXSPI NOR flash device.

Below are some examples of the boot image version.

0xFFFE0001, valid boot image version, image version is 1.

0xFFFD0002, valid boot image version, image version is 2.

0x00000003, invalid boot image version, does not follow the image version rules.

0xFFFFFFFF, valid image version, but with lower boot sequence.

0x0000FFFF, valid image version, image version if the largest version 0xFFFF.

26.3.1.2.8 Ping-Pong boot flow

For FLEXSPI boot, ROM always tries to find the latest boot image and boot. If boot fails, ROM will try to find the old boot image. If the old image still boots fail, then ROM will enter ISP boot mode. Below is the boot flow chart for the Ping-Pong boot.

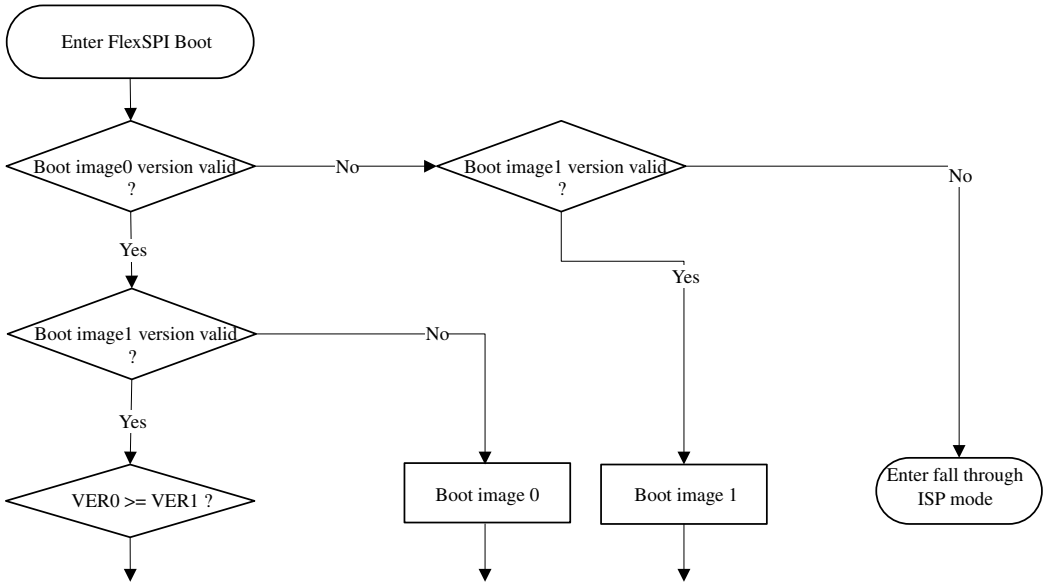


Figure 88. FLEXSPI boot image selection

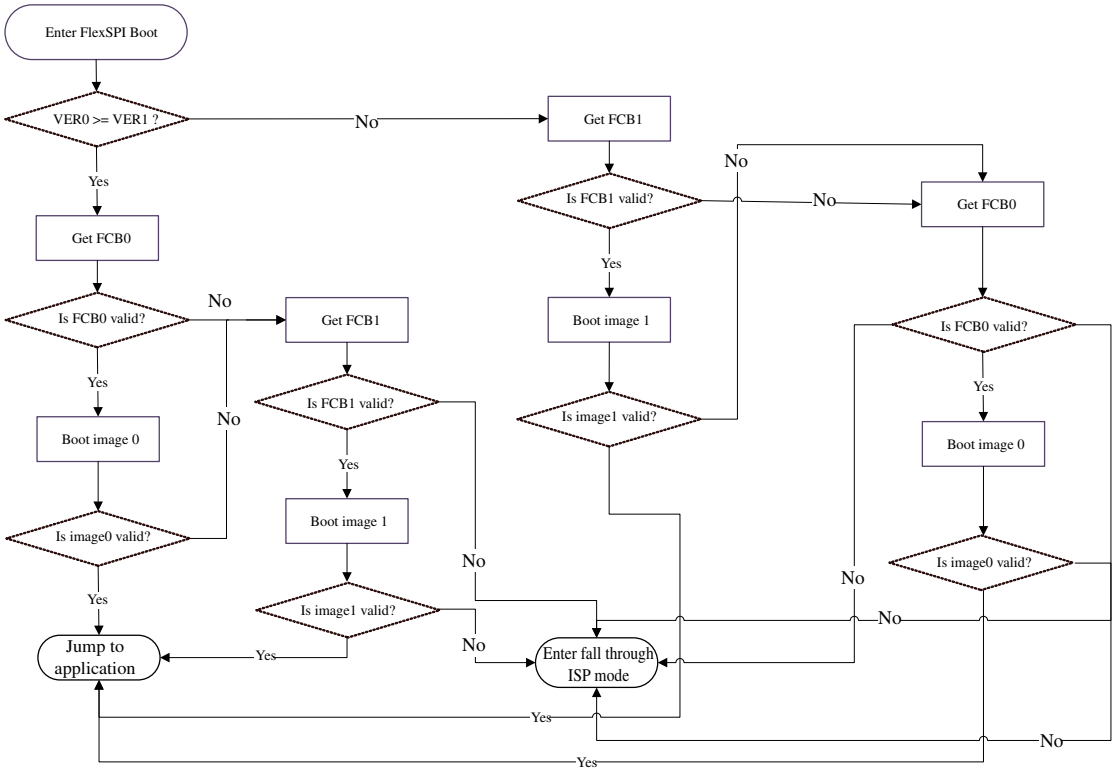


Figure 89. FLEXSPI Ping-Pong boot flow

26.3.1.2.9 Configure FLEXSPI NOR Device

The ROM supports the booting of the image from an external FLEXSPI NOR flash device. This section describes how to use *blhost* tool to program the image into external nonvolatile memory for booting. The *blhost* tool uses UART, SPI, I2C, USB HID to communicate with the ROM code via ROM ISP mode.

Table 127. FLEXSPI pin assignments for NOR flash connections

Boot interface	Pins	
FlexSPI	FLEXSPI_SSEL0	PIO0_21
	FLEXSPI_SSEL1	PIO0_22
	FLEXSPI_CLK	PIO0_19
	FLEXSPI_D0	PIO0_6
	FLEXSPI_D1	PIO0_4
	FLEXSPI_D2	PIO0_3
	FLEXSPI_D3	PIO0_2
	FLEXSPI_D4	PIO1_16
	FLEXSPI_D5	PIO1_15
	FLEXSPI_D6	PIO1_27
	FLEXSPI_D7	PIO1_29
	FLEXSPI_DQS	PIO0_25

26.3.1.2.9.1 FLEXSPI NOR FLASH boot image programming

ROM supports boot from different type of NOR flash such as Quad Flash, Octal Flash, Hyper Flash, which connects to FLEXSPI pins used by ROM. For FLEXSPI NOR boot, the FCB should be programmed at offset 0x08000400, and the boot image should be programmed at offset 0x08001000. The below section will provide how to config and program the boot image into NOR flash for 5 common NOR flash type

26.3.1.2.9.2 FLEXSPI NOR FLASH configuration parameters

```
typedef struct _serial_nor_config_option
{
union
{
struct
{
uint32_t max_freq : 4;                  // Maximum supported Frequency
uint32_t misc_mode : 4;               // miscellaneous mode
uint32_t quad_mode_setting : 4;  // Quad mode setting
uint32_t cmd_pads : 4;                 // Command pads
uint32_t query_pads : 4;               // SFDP read pads
uint32_t device_type : 4;            // Device type
uint32_t option_size : 4;             // Option size, in terms of uint32_t, size = (option_size
+ 1)
```

```
uint32_t tag : 4;                          // Tag, must be 0x0C
} B;
uint32_t U;
} option0;
union
{
struct
{
uint32_t dummy_cycles : 8;     // Dummy cycles before read
uint32_t status_override : 8;    // Override status register value during device
mode            configuration
uint32_t pinmux_group : 4;          // The pinmux group selection
uint32_t dqs_pinmux_group : 4;  // The DQS Pinmux Group Selection
uint32_t drive_strength : 4;          // The Drive Strength of FlexSPI Pads
uint32_t flash_connection : 4;     // Flash connection option: 0 - Single Flash connected to
port A, 1 - //! Parallel mode, 2 - Single Flash connected to Port B
} B;
uint32_t U;
} option1;
} serial_nor_config_option_t;
```

Table 128. serial_nor_config_option_t definition

Offset	Field	Description
0	Option0	See option0 definition for more details
4	Option1	Optional, effective only if the option Size field in Option0 is non-zero See option1 definition for more details.

Table 129. Option0 definition

Field	Bits	Description
tag	31:28	The tag of the config option, fixed to 0x0C
option_size	27:24	Size in bytes = (Option Size + 1) * 4 It is 0 if only option0 is required
device_type	23:20	Device Detection Type 0 - Read SFDP for SDR commands 1 - Read SFDP for DDR Read commands 2 - HyperFLASH 1V8 3 - HyperFLASH 3V 4 - Macronix Octal DDR 6 - Micron Octal DDR 8 - Adesto EcoXiP DDR
query_pad	19:16	Data pads during Query command

Table continues on the next page...

Table 129. Option0 definition (continued)

Field	Bits	Description
		(read SFDP or read MID) 0 - 1 2 - 4 3 – 8
cmd_pad	15:12	Data pads during Flash access command 0 - 1 2 - 4 3 – 8
quad_mode_setting	11:8	Quad Mode Enable Setting 0 - Not configured 1 - Set bit 6 in Status Register 1 2 - Set bit 1 in Status Register 2 3 - Set bit 7 in Status Register 2 4 - Set bit 1 in Status Register 2 vis 0x31 command This setting is just for user to configure the flash into QPI mode, when flash enters into QPI mode, ROM will not reset the flash into stand SPI mode after the device is reset, user need to reset the flash into stand SPI mode themselves. Note: This field will be effective only if device is compliant with JESD216 only (9 longword SFDP table)
misc_mode	7:4	Miscellaneous Mode 0 - Not enabled 1 - Enable 0-4-4 mode for High Random Read performance 3 - Data Order Swapped mode (for MXIC OctaFlash only) Note: Experimental feature, do not use in products, keep it as 0
max_freq	3:0	Max Flash Operation speed 0 - Don't change FlexSPI clock setting

Table 130. Option1 definition

Field	Bits	Description
reserved	31:8	Reserved for future use

Table continues on the next page...

Table 130. Option1 definition (continued)

Field	Bits	Description
dummy_cycles	7:0	Dummy cycles for read command 0 - Use detected dummy cycle Others - dummy cycles provided in flash data sheet

Typical Option configurations are as below:

- QuadSPI NOR - Quad SDR Read: option0 = 0xc0000002 (50MHz)
- QuadSPI NOR - Quad DDR Read: option0 = 0xc0100002 (50MHz)
- HyperFLASH 1V8: option0 = 0xc0233002 (50MHz)
- HyperFLASH 3V0: option0 = 0xc0333002 (50MHz)
- MXIC OPI DDR (OPI DDR enabled by default): option=0xc0433005(50MHz)
- Micron Octal DDR: option0=0xc0600002 (50MHz)
- Micron OPI DDR: option0=0xc0603002 (50MHz)
- Micron OPI DDR (DDR read enabled by default): option0 = 0xc0633002 (50MHz)
- Adesto OPI DDR: option0=0xc0803002(50MHz)

26.3.1.2.9.3 NOR FLASH Config, Erase and Program via blhost tool

Take Macronix MX25L25645G, which connects to FlexSPI interface, as an example. First, the config parameter should be stored in RAM, which will be used in configuring the FLEXSPI in the next step. From below Figure, the config parameter for FLEXSPI is 0xc0000001. The config parameter is selected according to the FLASH type

```
C:\blhost>blhost.exe  -p com4 fill-memory 0x2000f000 4 0xc0000001
Ping responded in 1 attempt(s)
Inject command 'fill-memory'
Successful generic response to command 'fill-memory'
Response status = 0 (0x0) Success.
```

Figure 90. Set the FLEXSPI config parameter in RAM

Use the config parameter stored in RAM in the previous step to config the FLEXSPI, as Figure 91 *shows. After that, you can read, erase, program the flash. For more details about the blhost tool command, you can refer to the help of the blhost tool.*

```
C:\blhost>blhost.exe  -p com4 configure-memory 0x9 0x2000f000
Ping responded in 1 attempt(s)
Inject command 'configure-memory'
Successful generic response to command 'configure-memory'
Response status = 0 (0x0) Success.
```

Figure 91. Config the FLEXSPI using the parameter stored in RAM

```
C:\blhost\blhost>blhost.exe -p com3 read-memory 0x08000400 0x100
Ping responded in 1 attempt(s)
Inject command 'read-memory'
Successful response to command 'read-memory'
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
(1/1)100% Completed!
Successful generic response to command 'read-memory'
Response status = 0 (0x0) Success.
Response word 1 = 256 (0x100)
Read 256 of 256 bytes.
```

Figure 92. Read the NOR flash via blhost tool

```
C:\blhost\blhost>blhost.exe -p com3 flash-erase-region 0x08000400 0x200
Ping responded in 1 attempt(s)
Inject command 'flash-erase-region'
Successful generic response to command 'flash-erase-region'
Response status = 0 (0x0) Success.
```

Figure 93. Erase the NOR flash via blhost tool

```
C:\blhost\blhost>blhost.exe -p com3 write-memory 0x08001000 boot_image.bin
Ping responded in 1 attempt(s)
Inject command 'write-memory'
Preparing to send 5796 (0x16a4) bytes to the target.
Successful generic response to command 'write-memory'
(1/1)100% Completed!
Successful generic response to command 'write-memory'
Response status = 0 (0x0) Success.
Wrote 5796 of 5796 bytes.
```

Figure 94. Program the boot image into NOR flash via blhost tool

```
C:\blhost\blhost>blhost.exe -p com3 read-memory 0x08001000 0x100
Ping responded in 1 attempt(s)
Inject command 'read-memory'
Successful response to command 'read-memory'
00 00 20 00 f9 d1 10 00 eb c8 10 00 5b cb 10 00
6f cc 10 00 bb cd 10 00 b7 d1 10 00 17 d3 10 00
a4 16 00 00 02 40 00 00 1d 26 cd c5 23 d3 10 00
9d d6 10 00 00 c0 10 00 9f d6 10 00 a1 d6 10 00
99 d4 10 00 a1 d4 10 00 a9 d4 10 00 b1 d4 10 00
b9 d4 10 00 c1 d4 10 00 c9 d4 10 00 d1 d4 10 00
d9 d4 10 00 e1 d4 10 00 e9 d4 10 00 f1 d4 10 00
f9 d4 10 00 01 d5 10 00 09 d5 10 00 11 d5 10 00
19 d5 10 00 21 d5 10 00 29 d5 10 00 31 d5 10 00
39 d5 10 00 41 d5 10 00 49 d5 10 00 51 d5 10 00
59 d5 10 00 61 d5 10 00 69 d5 10 00 71 d5 10 00
79 d5 10 00 81 d5 10 00 89 d5 10 00 91 d5 10 00
99 d5 10 00 a1 d5 10 00 a9 d5 10 00 b1 d5 10 00
b9 d5 10 00 c1 d5 10 00 c9 d5 10 00 d1 d5 10 00
d9 d5 10 00 e1 d5 10 00 e9 d5 10 00 f1 d5 10 00
f9 d5 10 00 01 d6 10 00 09 d6 10 00 11 d6 10 00
(1/1)100% Completed!
Successful generic response to command 'read-memory'
Response status = 0 (0x0) Success.
Response word 1 = 256 (0x100)
Read 256 of 256 bytes.
```

Figure 95. Read the boot image back via blhost tool

Generate and program the FLEXSPI NOR FCB(flash config block) into flash For FLEXSPI boot, and it needs an FCB (flash config block) at offset 0x08000400. This FCB is used to config the FLEXSPI interface when booting the image from external NOR flash via the FLEXSPI interface. The FCB be generated from the previous FLEXSPI config parameter(0xc0000002). Store the FCB generating and program parameter into the RAM, which will be used in the next step to generate and program the FCB into flash at 0x08000400.

```
C:\blhost>blhost.exe  -p com4 fill-memory 0x2000f000 4 0xf000000f
Ping responded in 1 attempt(s)
Inject command 'fill-memory'
Successful generic response to command 'fill-memory'
Response status = 0 (0x0) Success.
```

Figure 96. Store the config FCB parameter into the RAM

FCB generating and program into the flash after this step, the FCB is generated and programmed into flash at offset 0x08000400 automatically by ROM.

```
C:\blhost>blhost.exe  -p com4 configure-memory 0x9 0x2000f000
Ping responded in 1 attempt(s)
Inject command 'configure-memory'
Successful generic response to command 'configure-memory'
Response status = 0 (0x0) Success.
```

Figure 97. FCB generate and program into flash at offset 0x08000400

```
C:\blhost\blhost>blhost.exe -p com3 read-memory 0x08000400 0x100
Ping responded in 1 attempt(s)
Inject command 'read-memory'
Successful response to command 'read-memory'
46 43 46 42 00 04 01 56 00 00 00 00 01 03 03 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
10 00 00 00 01 04 02 00 00 00 00 00 00 00 00 00
00 00 00 02 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ec 04 20 0a 00 1e 04 32 04 26 00 00 00 00 00 00
05 04 04 24 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
06 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
21 04 20 08 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
(1/1)100% Completed!
Successful generic response to command 'read-memory'
Response status = 0 (0x0) Success.
Response word 1 = 256 (0x100)
Read 256 of 256 bytes.
```

Figure 98. Read the FCB back via blhost tool

26.3.1.2.9.4 Typical NOR flash config parameters

For different NOR flash type. It needs different config parameter, and the below section shows some typical flash config parameters

Table 131. Typical NOR flash config parameters

Device Vendor	Device Type	Flash Type	Config Parameter	Flash Mode	Comment
Adesto	Octal Flash	ATXP032	0xC0803001	DDR	The ATXP032 has 3 versions, REVA/B/C. For REV A, ROM only supports Auto probe boot, the OTP fuse word

Table continues on the next page...

Table 131. Typical NOR flash config parameters (continued)

Device Vendor	Device Type	Flash Type	Config Parameter	Flash Mode	Comment
					FLEXSPI_BOOT_CFG[0] should set to 0x7, and no need to config the FCB
Cypress	Hyper Flash	S26KS512S	0xC0233001	DDR	For Hyper flash boot, the FLEXSPI_BOOT_CFG[0] should set to 0x20 before boot
Macronix	Quad Flash	MX25L25645G	0xC0000001	SDR	
Macronix	Quad Flash	MX25L25645G	0xC0100001	DDR	
Macronix	Octal Flash	MX25UM51345G	0xC0403001	DDR	
Micron	Octal Flash	MT35XL512ABA	0xC0603001	DDR	

26.3.2 ISP boot mode

ISP boot mode can be entered as a result of failed internal flash or FlexSPI flash image verification or if ISP pin forces the device into ISP mode. The ISP mode is mainly used for:

- Downloading the image (initial or updated) into the internal flash/external flash from the Host.
- Provisioning the device into production life cycle (ISP fall-through mode, lock settings).

26.3.3 Recovery boot

The ROM supports a recovery boot from an external 1-bit SPI flash device.

In FLASH boot mode, if the internal flash image is deemed invalid, the device checks the SPI_RECOVERY_BOOT_EN bits (SPI_CAN_CFG <1:0>) at address 0x3E204 of CMPA in protected flash to determine if SPI flash recovery is enabled.

LPC55xx boots a plain text image with the boot address and image length specified in the image header. For plain text SPI flash recovery, the image can only be booted into internal RAM in a non-reserved region. To determine reserved regions of RAM, use the following command in ISP mode:

blhost -p COMx get-property 12 (take the UART peripheral as an example)

See 1-bit Serial NOR flash through SPI for more details.

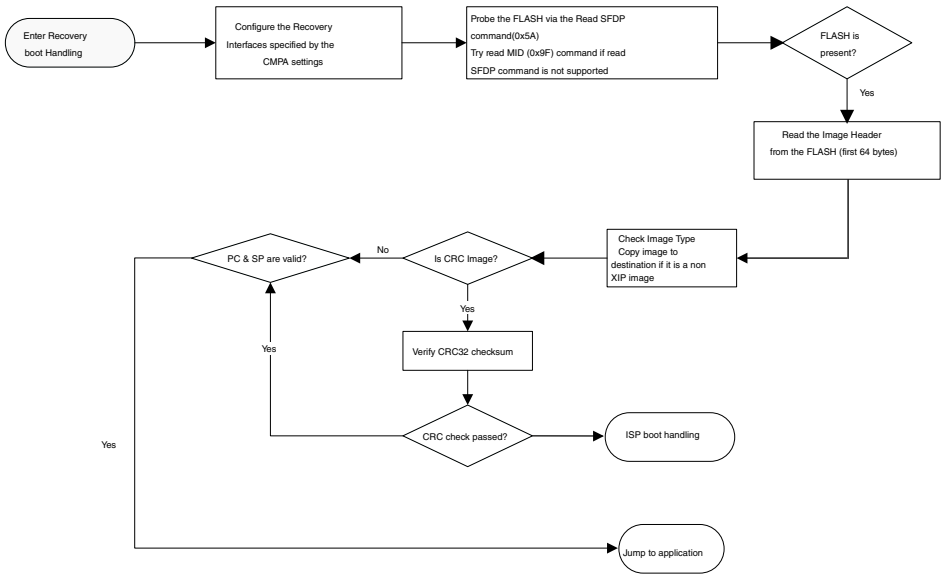


Figure 99. Recovery boot flow

26.4 External Memory Support

This section describes the external memory devices supported by the boot ROM ISP command. To use an external memory device correctly, the device must be enabled with the corresponding configuration profile. If the external memory device is not enabled, then it cannot be accessed by the ROM ISP command. The boot ROM enables specific external memory devices using a pre-assigned memory identifier, supported external memory devices, and memory identifiers are shown below.

Table 132. Memory ID for external memory devices

Memory Identifier	External Memory device
0x09	'Serial NOR over FlexSPI module'
0x110	'Serial NOR/EEPROM over SPI module.'

26.4.1 Serial NOR Flash through FlexSPI

The boot ROM supports read, write, and erase external Serial NOR Flash devices via the FlexSPI Module. Before accessing Serial NOR Flash devices, the FlexSPI module must be configured properly, using a simplified FlexSPI NOR Config block or a complete 512-byte FlexSPI NOR Configuration Block. Boot ROM can generate the 512byte FlexSPI NOR Configuration Block based on the simplified Flash Configuration Option Block for most Serial NOR Flash devices in the market.

Table 133. FlexSPI NOR Configuration Block(FCB)

Field	Offset	Size (bytes)	Description
tag	0x000	4	Config Block tag, must be 0x42464346

Table continues on the next page...

Table 133. FlexSPI NOR Configuration Block(FCB) (continued)

Field	Offset	Size (bytes)	Description
version	0x004	4	Configuration block, used by the Boot ROM internally, fixed to 0x56020000
reserved	0x008	4	Reserved for future use
readSampleClkSrc	0x00c	1	Read Sampling Clock source option 0 - Internal sampling 1 - Loopback from DQS Pad 2 - Loopback from SCK Pad 3 - External DQS signal
csHoldTime	0x00d	1	CS Hold time in terms of Flash clock cycles Recommended value is 3
csSetupTIme	0x00e	1	CS setup time in terms of Flash clock cycles Recommended value is 3
columAddressWidth	0x00f	1	Column Address Width Set to 3 for HyperFLASH Set to 0 for other FLASH devices
deviceModeCfgEnable	0x010	1	Enable the Device Mode Configuration sequence
deviceModeType	0x011	1	Argument for the Device Mode Configuration. For example, Quad Enable setting in Status Register2 for some QSPI FLASH devices
waitTimeCfgCommands	0x012	2	Wait time is terms of 100 us for the Device Mode Config Command
deviceModeSeq	0x014	4	Device Mode Configuration Sequence byte 0: Number of required sequences byte 1: Sequence Index
deviceModeArg	0x018	4	Argument for the Device Mode Configuration. For example, Quad Enable setting in Status Register2 for some QSPI FLASH devices

Table continues on the next page...

Table 133. FlexSPI NOR Configuration Block(FCB) (continued)

Field	Offset	Size (bytes)	Description
configCmdEnable	0x01c	1	Configuration Command Enable 0 - Ignore Configuration Command 1 - Enable Configuration Command
configModeType	0x01d	3	Config command types, support up to 3 types, each type is combined with a config command sequence
configCmdSeqs	0x020	12	Config command types Support up to 3 sequences
reserved	0x02c	4	Reserved for future use
configCmdArgs	0x030	12	Configure Command Argument Support up to 3 arguments
reserved	0x03c	4	Reserved for future use
controllerMiscOption	0x040	4	Miscellaneous Controller Configuration options bit 0 - Differential clock enable, set to 1 for HyperFLASH 1V8 device, set to 0 for other devices bit 3 - WordAddressableEnable, set to 1 for HyperFLASH, set to 0 for other devices bit 4 - SafeConfigFreqEnable, set to 1 if expecting to configure device with safe frequency bit 6 - DDR mode enable, set to 1 if DDR read is expected Other bits - Reserved, set to 0
deviceType	0x044	1	Device Type 1 - Serial NOR
sflashPadType	0x045	1	Data pad used in Read command 1 - Single pad 2 - Dual pads 4 - Quad pads 8 - Octal pads
serialClkFreq	0x046	1	SDR: 1 - 30MHz

Table continues on the next page...

Table 133. FlexSPI NOR Configuration Block(FCB) (continued)

Field	Offset	Size (bytes)	Description
			2 - 50MHz 3 - 60MHz 4 - 75MHz 5 - 100MHz DDR: 1 - 30MHz 2 - 50MHz
lutCustomSeqEnable	0x047	1	LUT customization Enable, it is required if the program/ erase cannot be done using 1 LUT sequence, currently, only applicable to HyperFLASH
reserved	0x048	8	Reserved for future use
sflashA1Size	0x050	4	Size of Flash connected to A1
sflashA2Size	0x054	4	Size of Flash connected to A2
sflashB1Size	0x058	4	Size of Flash connected to B1
sflashB2Size	0x05c	4	Size of Flash connected to B2
csPadSettingOverride	0x060	4	Pad override value for CS pin Using this value to configure CS pin if it is non-zero, otherwise use default ROM setting
sclkPadSettingOverride	0x064	4	Sck pad setting override value
dataPadSettingOverride	0x068	4	Data pad setting override value
dqsPadSettingOverride	0x06c	4	Dqs pad setting override value
timeoutInMs	0x070	4	Timeout value in terms of ms to terminate the busy check
commandInterval	0x074	4	CS deselect interval between two commands
dataValidTime	0x078	4	CLK edge to data valid time
busyOffset	0x07c	2	Busy offset, valid value: 0-31
busyBitPolarity	0x07e	2	Busy flag polarity, 0 - busy flag is 1 when flash device is busy, 1 -busy flag is 0

Table continues on the next page...

Table 133. FlexSPI NOR Configuration Block(FCB) (continued)

Field	Offset	Size (bytes)	Description
			when flash device is busy
lookupTable	0x080	256	16 LUT sequences each sequence consists of 4 words, see FlexSPI chapter for more details.
lutCustomSeq	0x180	48	Customizable LUT Sequences
reserved	0x1b0	16	Reserved for future use
pageSize	0x1c0	4	Page size of Flash
sectorSize	0x1c4	4	Sector size of Flash
ipcmdSerialClkFreq	0x1c8	1	Serial Clock Frequency for IP command 0 - The same with Read command others - the same definition as serialClkFreq
isUniformBlockSize	0x1c9	1	If the sector size is the same with block size. 0 - No 1 - Yes
isDataOrderSwapped	0x1ca	1	Data order (D0, D1, D2, D3) is swapped (D1,D0, D3, D2)
reserved	0x1cb	1	
serialNorType	0x1cc	1	Serial NOR Flash type: 0/1/2/3
needExitNoCmdMode	0x1cd	1	Need to exit NoCmd mode before other IP command
halfClkForNonReadCmd	0x1ce	1	Half the Serial Clock for non-read command: true/false
needRestoreNoCmdMode	0x1cf	1	Need to Restore NoCmd mode after IP commmand execution
blockSize	0x1d0	4	Flash Block size
flashStateCtx	0x1d4	4	Flash State Context
reserved	0x1d8	40	Reserved for future use

NOTE

To customize the LUT sequence for some specific device, users need to enable "**lutCustomSeqEnable**" and fill in the corresponding "**lutCustomSeq**" field specified by the command index below.

For Serial (SPI) NOR, the pre-defined LUT index is as following:

Table 134. Lookup Table index pre-assignment for FlexSPI NOR

Name	Index in lookup table	Description
Read	0	Read command Sequence
ReadStatus	1	Read Status command
ReadStatusXpi	2	Read Status command under OPI mode
WriteEnable	3	Write Enable command sequence
WriteEnableXpi	4	Write Enable command under OPI mode
EraseSector	5	Erase Sector Command
EraseBlock	8	Erase Block Command
PageProgram	9	Page Program Command
ChipErase	11	Full Chip Erase
ExitNoCmd	15	Exit No Command Mode as needed
Reserved	6,7,10,12,13,14	All reserved indexes can be freely used for other purposes

26.4.2 FlexSPI NOR Configuration Option Block

The FlexSPI NOR Configuration Option Block is organized by 4-bit unit, and it is expandable, the current definition of the block is as shown in the below table.

The ROM detects FCB via reading SFDP command, which is supported by most flash devices those are JESD216(A/B) compliant. However, JESD216A/B only defines the dummy cycles for Quad SDR read. In order to get the dummy cycles for DDR/DTR read mode, ROM supports auto probing by writing test patterns to offset 0x200 on the external memory devices. To get optimal timing, the readSampleClkSrc set to 1 for Flash devices that do not support external provided DQS pad input. It is set to 3 for flash devices that support external provided DQS pad input such as OctalFlash /HyperFLASH. FlexSPI_DQS pad is not used for other purposes.

Table 135. FlexSPI NOR Configuration Option Block

Offset	Field	Description							
0	Option0	**TAG [31:28]**	**Option size [27:24]**	**Device Detection Type [23:20]**	**Query CMD Pad(s) [19:16]**	**CMD Pad(s) [15:12]**	**Quad Enable Type[11:8]**	**Misc[7: 4]**	**Max Freq [3:0]**
		0xc	Size in bytes = (Option	0 - QuadSPI SDR	0 - 1 2 - 4 3 - 8	0 - 1 2 - 4 3 - 8	1 - QE bit is bit 6 in StatusReg1	Miscellaneous Mode 0 - Not enabled	Max Flash Operation speed

Table continues on the next page...

Table 135. FlexSPI NOR Configuration Option Block (continued)

Offset	Field	Description							
			Size + 1)* 4	1 - QuadSPI DDR 2 - HyperFLASH 1V8 3 - HyperFLASH 3V 4 - MXIC OPI DDR 6 -Micron OPI DDR 7 - Micron Octal SDR 8 -Adesto OPI DDR 9 - Adesto EcoXiP SDR			2 - QE bit is bit 1 in StatusReg2 3 - QE bit is in bit 7 in StatusReg2 4 - QE bit is bit 1 in StatusReg2, enable command is 0x31 **NOTE** This field will be effective only if device is compliant with JESD216 only (9 longword SDFP table)	1 - Enable 0-4-4 mode for High Random Read performance 3 - Data Order Swapped mode (for MXIC OctaFlash only) 5 - Select the FlexSPI data sample source as internal loop back, more details please refer FlexSPI usage 6 - Config the FlexSPI NOR flash running at stand SPI mode **NOTE** Experimental feature, do not use in products, keep it as 0.	0 - Don't change FlexSPI clock setting
4	Option1 Optional	**flash_connection [31:28]**		**Reserved [27:16]**		**status_override [15:8]**		**Dummy Cycle [7:0]**	
		Flash connection option: 0 - Single Flash connected to port A 1 - Parallel mode 2 - Single Flash connected to Port B				Override status register value during device mode configuration		Dummy cycles for read command 0 - Use detected dummy cycle Others - dummy cycles provided in flash data sheet	

- Tag - Fixed as 0x0C
- Option Size - Provide scalability for future use, the option block size equals to(Option size + 1) * 4 bytes
- Device Detection type - SW defined device types used for config block autodetection

- Query Command Pad(s) - Command pads (1/4/8) for the SFDP command
- CMD pad(s) - Commands pads for the Flash device (1/4/8), for device that works under 1-1-4,1-4-4,1-1-8 or 1-8-8 mode, CMD pad(s) value is always 0x0, for devices that only support 4-4-4 mode for high performance, CMD pads value is 2, for devices that only support 8-8-8 mode for high performance, CMD pads value is 3
- Quad Enable Type - Specify the Quad Enable sequence, only applicable for device that only JESD216 compliant, this field is ignored if device support JESD216A or later version
- Misc - Specify miscellaneous mode for selected flash type
- Max Frequency - The maximum work frequency for specified Flash device
- Dummy Cycle - User provided dummy cycles for SDR/DDR read command
- Status override - Override status register value during device mode configuration
- Flash connection – Select the FlexSPI Port A/B

26.4.3 Typical use cases for FlexSPI NOR Configuration Block

- QuadSPI NOR - Quad SDR Read: option0 = 0xc0000002 (50MHz)
- QuadSPI NOR - Quad DDR Read: option0 = 0xc0100002 (50MHz)
- Hyper FLASH 1V8: option0 = 0xc0233002 (50MHz)
- Hyper FLASH 3V0: option0 = 0xc0333002 (50MHz)
- MXIC OPI DDR (OPI DDR enabled by default): option=0xc0433002(50MHz)
- Micron Octal DDR: option0=0xc0600002 (50MHz)
- Micron OPI DDR: option0=0xc0603002 (50MHz)
- Micron OPI DDR (DDR read enabled by default): option0 = 0xc0633002 (50MHz)
- Adesto OPI DDR: option0=0xc0803002(50MHz)

26.4.4 Program Serial NOR Flash device using FlexSPI NOR Configuration Option

The boot ROM supports generating complete FCB using configure-memory command. It also supports programming the generated FCB to the start of the flash memory using a specific option "0xF000000F". Here is the example for configuring and accessing HyperFLASH (Assuming it is a blank HyperFLASH device). See Configure FLEXSPI NOR Device for more details.

```
blhost -u <PID,VID> - fill-memory 0x2000f000 0x04 0xc0233007 (write option block to SRAM
address 0x2000f000)

blhost -u <PID,VID> - configure-memory 0x09 0x2000f000 (configure HyperFLASH using option block)

blhost -u <PID,VID> - fill-memory 0x2000f000 0x04 0xf000000f (write specific option to SRAM
address 0x2000f000)

blhost -u <PID,VID> - configure-memory 0x09 0x2000f000 (program FCB to the offset 0x400 of
HyperFLASH device)

blhost -u <PID,VID> – flash-erase-region <addr> <size> (erase the HyperFLASH device from addr of size)

blhost -u <PID,VID> - write-memory <addr> image.bin (write the image to addr)
```

NOTE

Flash Block erase is not supported for FlxSPI memory, when flash-erase-region command fis issued from blhost.

26.4.5 1-bit Serial NOR flash through SPI

The boot ROM supports programming 1-bit SPI NOR FLASH devices (which supports a 3-byte address read 0x03 or 4-byte address read 0x13) via the Flash Configuration Option Block. The SPI clock frequency is set to 48Mhz by the ROM.

The boot ROM supports either a manually configured option or an auto-detected option. When using the manually configured option, you must specify all the Flash information (Flash size, sector size, page size) in the option block. When using the auto-detected option (which is only supported on devices that are JESD216 compliant), the boot ROM is able to detect the Flash information via the read SFDP (0x5A) command, where you can set all the Flash information to 0x0s. The below table shows the details required for configuring SPI NOR FLASH using either of these methods. The read Page (0x03) is used for the default read command, if the FLASH size is greater than 16MB (detected by the read SFDP command), the 0x13 command is used for the page read. If the SFDP command is not supported, ROM uses the read MID (0x9F) to detect whether there is a connected device to the boot ROM.

26.4.6 1-bit SPI NOR Configuration Option Block

Table 136. SPI NOR Configuration option Block Definition

<table>
<tr><th rowspan="2">Field</th><th>Tag</th><th>Reserved</th><th>Flash info set</th><th>Flash size</th><th>Sector size</th><th>Page size</th></tr>
<tr><th>[31:28]</th><th>[27:16]</th><th>[15:12]</th><th>[11:8]</th><th>[7:4]</th><th>[3:0]</th></tr>
<tr><td rowspan="11">option0</td><td rowspan="11">0xc</td><td rowspan="11"></td><td rowspan="3">0 - Manual
(Select Flash Parameter via Flash Datasheet)</td><td>0 - 512KB</td><td>0 - 4KB</td><td>0 - 256 Bytes</td></tr>
<tr><td>1 - 1MB</td><td>1 - 8KB</td><td>1 - 512 Bytes</td></tr>
<tr><td>2 - 2MB</td><td>2 - 32KB</td><td>2 - 1KB</td></tr>
<tr><td rowspan="8">2 - Auto
(Detect Flash Parameter via SFDP table)</td><td>3 - 4MB</td><td>3 - 64KB</td><td>3 - 128KB</td></tr>
<tr><td>4 - 8MB</td><td>4 - 128KB</td><td rowspan="7"></td></tr>
<tr><td>5 - 16MB</td><td>5 - 256KB</td></tr>
<tr><td>6 - 32MB</td><td rowspan="6"></td></tr>
<tr><td>7 - 64MB</td></tr>
<tr><td>8 - 64MB</td></tr>
<tr><td>9 - 128MB</td></tr>
<tr><td>10 - 256MB</td></tr>
</table>

- Tag - Fixed as 0xc
- SPI index - SPI interface used to access serial NOR flash
- Memory Type - Memory type used to configure and access the NOR flash
- Memory size - Memory capacity of the NOR flash device
- Sector size - Sector size of the NOR flash device
- Page size - Page size of the NOR flash device

26.4.7 Program 1-bit serial NOR flash device via SPI NOR Configuration Option Block

The boot ROM supports programing the serial NOR flash device via SPI interface using the SPI NOR configuration option block. Below are the example. Before writing the Configuration Option Block, the SPI peripheral index is selected by writing the value 0xCF90000<n>, where <n> is the index of the SPI peripheral.

```
blhost -p comxx - fill-memory 0x2000f000 0x04 0xcf900001 (write option block to SRAM address
0x2000f000,this is used to select the SPI index, 0xcf900003 to select SPI index 3)

blhost -p comxx - fill-memory 0x2000f000 0x04 0xc000001 (write option block to SRAM address 0x2000f000, the
spi instance is 1)

blhost -p comxx - configure-memory 0x110 0x2000f000 (configure serial NOR FLASH using option block)

blhost -p comxx - flash-erase-region <addr> <size> 0x110 (erase the serial NOR flash deice from addr
of size)

blhost -p comxx - write-memory <addr> image.bin 0x110
```

NOTE

Flash Block erase is not supported for FlxSPI memory, when flash-erase-region command is issued from blhost in ISP mode.

26.5 OTP-eFUSE definitions

The LPC553x supports the FUSE block, and ROM defines the following logical FUSE layout and exported the APIs for FUSE read and program. See the detailed FUSE layout in Table 137.

Table 137. FUSE definition in LPC553x

Fuse word index	Bit width	Field name	Description
0	8	LC_STATE	Life cycle state, only the lowest 8 bits are valid
1	32	CMPA_CRC	CMPA CRC check value
2	32	CMPA_MAN_DATA0	CMPA CMA check value word 0
3	32	CMPA_MAN_DATA1	CMPA CMA check value word 1
4	32	CMPA_MAN_DATA2	CMPA CMA check value word 2
5	32	CMPA_MAN_DATA3	CMPA CMA check value word 3
7	32	CUSTOMER_WORD0	Customer use word 0
8	8	CUSTOMER_WORD1	Customer use word 1, only the lowest 8 bits are valid
9	8	CUSTOMER_WORD2	Customer use word 2, only the lowest 8 bits are valid

NOTE

Flash Block erase is not supported for FlxSPI memory, when flash-erase-region command fis issued from blhost.

Chapter 27
ISP and IAP

27.1 Overview

This device include In-System Programming (ISP) functions to support image programming from the serial interface (UART, I2C, SPI, CAN) and USB HID. In-Application Programming (IAP) calls are available.

27.1.1 Features

In-System Programming supports:

- UART, I2C, SPI, CAN and USB peripheral interfaces.
  - UART boot from Flexcomm 0
  - I2C boot from Flexcomm 1
  - SPI boot from HS-SPI (Flexcomm 8)
- Automatic detection of the active peripheral.
- UART peripheral implements auto-baud detection.
- CAN peripheral implements auto-baud detection for the pre-defined baudrates(1Mbit/s, 500Kbit/s, 250Kbit/s, 125Kbit/s)
- Common packet-based protocol for all peripherals.
- Packet error detection and retransmit.
- Flash-resident configuration options (in CMPA)
- Protection of RAM used by the bootloader while it is running.
- Retrieval the properties of the device, such as Flash and RAM size.
- Multiple options for executing the bootloader either at system startup or under application control at runtime.
- Support for internal flash access.

27.2 Functional description

27.2.1 Bootloader

The internal ROM memory is used to store the boot code. After a reset, the Arm processor starts its code execution from this memory. The bootloader code is executed every time the part is powered-on, is reset, or is woken up from a deep power-down, low power mode.

The bootloader provides FLASH and OTP-eFuse programming utility that operates over a serial connection on the MCUs. It enables quick and easy programming of MCUs through the entire product lifecycle, including application development, final product manufacturing, and beyond. The host-side command line and GUI tools are available to communicate with the bootloader. Users can utilize host tools to upload/download application code and do manufacturing via the bootloader.

For the bootloader operation and boot pin, see Non-secure Boot ROM.

27.2.2 In-System Programming (ISP) and In-Application Programming (IAP)

Serial booting and other related functions, are supported in several different ways:

- For details of the ISP protocol, see In-System programming protocol.
- For details of the ISP packet, see Bootloader packet types.
- For details of the ISP commands, see The bootloader command set.

- For details of UART In-System Programming, see UART ISP.
- For details of I2C In-System Programming, see I2C In-System Programming
- For details of SPI In-System Programming, see SPI In-System programming.
- For details of USB In-System programming, see USB In-System Programming
- For details of CAN In-System Programming, see CAN ISP.
- For details of In-Application Programming, see In-Application-Programming.

27.2.3 Memory map after any reset

The boot ROM is located in the memory region, starting from the address 0x1300 0000. Both the ISP and IAP software use parts of the on-chip RAM. The RAM usage is

described in ISP interrupt and SRAM use.

Based on the DEFAULT_ISP_MODE bit settings or ISP pin settings, the ROM enters ISP mode and auto-detect activity on the I2C / SPI/ USART/CAN or USB-HID interface. The auto-detect looks for activity on the USART, I2C, SPI, CAN and USB-HID interfaces and selects the appropriate interface once a properly formed frame is received. If an invalid frame is received, the data is discarded and scanning resumes. USART, I2C, SPI, CANand USB-HID ISP communications are described in In-System programming protocol and Bootloader packet types.

27.2.4 ISP interrupt and SRAM use

27.2.4.1 Interrupts during IAP

When the user application code starts executing, the interrupt vectors from the SRAM are active. Before making any IAP call, disable the interrupts. The IAP code does not use or disable interrupts.

27.2.4.2 RAM used by the ISP command handler

The below regions are reserved for bootloader use when the bootloader is running.

Table 138. RAM used by ROM during ROM execution

RAM region	Purpose
0x1400_0000-0x1400_3FFF	BSS, RW, Stack and Heap
0x3000_1000-0x3000_7fff	0x3000_1000 - 0x3000_1FFF (For the recovery sb loader file handling only) 0x3000_4000 - 0x3000_7fff (For the USB HID RAM use)

27.2.5 ISP boot flow

ISP boot flow is shown below:

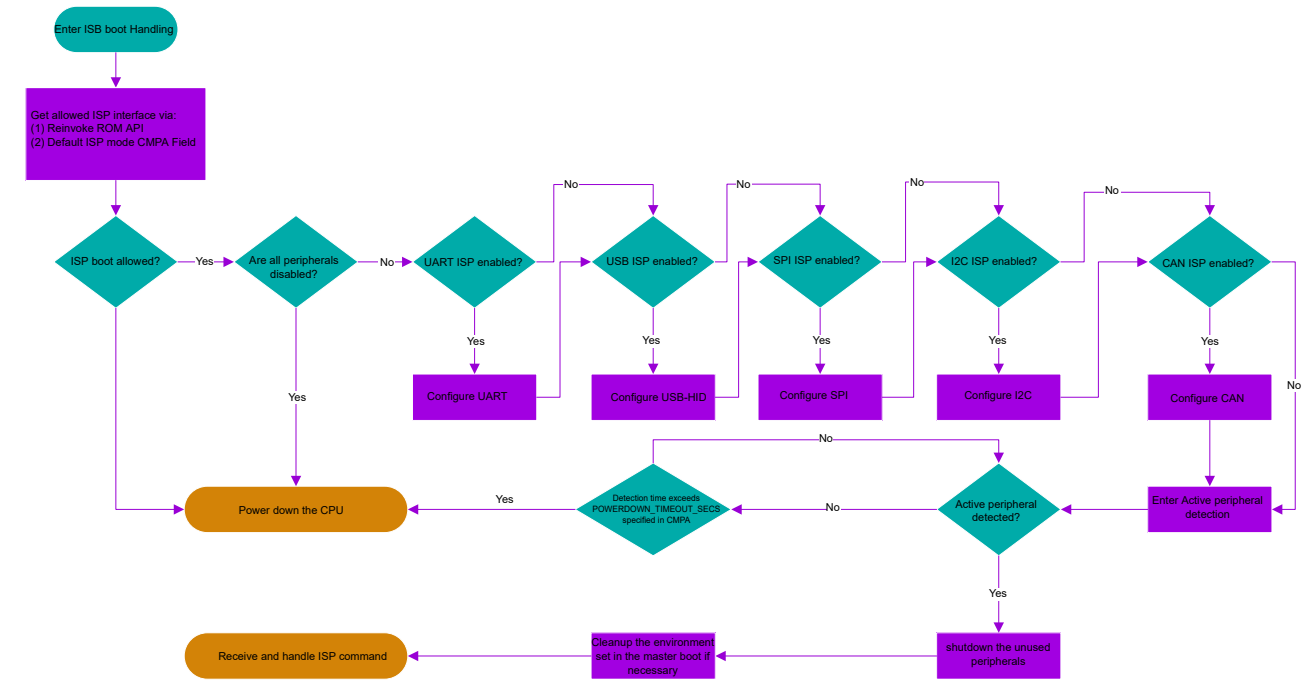


Figure 100. ISP boot flow

27.3 In-System programming protocol

This section explains the general protocol for the packet transfers between the host and the bootloader. The description includes the transfer of packets for different transactions, such as commands with no data phase and commands with incoming or outgoing data phase. The next section describes various packet types used in a transaction.

Each command sent from the host is replied to with a response command.

Commands may include an optional data phase.

- If the data phase is incoming (from the host to the bootloader), it is part of the original command.
- If the data phase is outgoing (from the bootloader to host), it is part of the response command.

27.3.1 Command with no data phase

The protocol for a command with no data phase contains:

- Command packet (from the host)
- Generic response command packet (to host)

Remark: In these diagrams, the ACK sent in response to a command or a data packet can arrive at any time before, during, or after the command or data packet has processed.

27.3.2 Command with the incoming data phase

The protocol for a command with incoming data phase contains:

- Command packet (from host) (kCommandFlag_HasDataPhase set).
- Generic response command packet (to host).
- Incoming data packets (from the host).

- Generic response command packet.

Note:

- The host may not send any further packets while it is waiting for the response to a command.
- The data phase is aborted if the Generic Response packet prior to the start of the Data phase does not have a status of kStatus_Success.
- Data phases may be aborted by the receiving side by sending the final
- GenericResponse early with a status of kStatus_AbortDataPhase. The host may abort the data phase early by sending a zero-length data packet.
- The final Generic Response packet sent after the data phase includes the status of the entire operation.

27.3.3 Command with outgoing data phase

The protocol for a command with an outgoing data phase contains:

- Command packet (from the host).
- ReadMemory Response command packet (to host) (kCommandFlag_HasDataPhase set).
- Outgoing data packets (to host).
- Generic response command packet (to host).
- The data phase is considered part of the response command for the outgoing data phase sequence.
- The host may not send any further packets while the host is waiting for the response to a command.
- The data phase is aborted if the ReadMemory Response command packet, prior to the start of the data phase, does not contain the kCommandFlag_HasDataPhase flag.
- Data phases may be aborted by the host sending the final Generic Response early with a status of kStatus_AbortDataPhase. The sending side may abort the data phase early by sending a zero-length data packet.
- The final Generic Response packet sent after the data phase includes the status of the entire operation.

27.4 Bootloader packet types

27.4.1 Introduction

The bootloader device works in slave mode. All data communications are initiated by a host, which is either a PC or an embedded host. The bootloader device is the target, which receives a command or data packet. All data communications between host and target are packetized.

There are six types of packets used:

- Ping packet.
- Ping Response packet.
- Framing packet.
- Command packet.
- Data packet.
- Response packet.

All fields in the packets are in little-endian byte order.

27.4.2 Ping packet

The Ping packet is the first packet sent from a host to the target to establish a connection on the selected peripheral in order to run autobaud detection. The Ping packet can be sent from host to target at any time that the target is expecting a command packet.

If the selected peripheral is UART, a Ping packet must be sent before any other communications. For other serial peripherals, it is optional.

In response to a Ping packet, the target sends a Ping response packet, discussed in the later sections.

Table 139. Ping packet format

Byte #	Value	Name
0	0x5A	Start byte
1	0xA6	Ping

27.4.3 Ping response packet

The target sends a Ping response packet back to the host after receiving a Ping packet. If communication is over a UART peripheral, the target uses the incoming Ping packet to determine the baud rate before replying with the Ping response packet. Once the Ping response packet is received by the host, the connection is established, and the host starts sending commands to the target.

Table 140. Ping response packet format

Byte	Value	Parameter
0	0x5A	Start byte
1	0xA7	Ping response code
2	0x00	Protocol bugfix
3	0x03	Protocol minor
4	0x01	Protocol major
5	0x50	Protocol name = 'P' (0x50)
6	0x00	Options low
7	0x00	Options high
8	0xfb	CRC16 low
9	0x40	CRC16 high

For the UART peripheral, it must be sent by the host when a connection is first established, in order to run outbound. For other serial peripherals, it is optional but recommended to determine the serial protocol version. The version number is in the same format as the bootloader version number returned by the GetProperty command.

27.4.4 Framing packet

The framing packet is used for flow control and error detection for the communications links that do not have such features built in. The framing packet structure sits between the link layer and the command layer. It wraps command and data packets as well.

Every framing packet containing data sent in one direction results in a synchronizing response framing packet in the opposite direction.

The framing packet described in this section is used for serial peripherals including the UART, I^2C, and SPI. The USB HID peripheral does not use framing packets. Instead, the packetization inherent in the USB protocol itself is used.

Table 141. Framing packet format

Byte	Value	Parameter	Description
0	0x5A	Start byte	
1		PacketType	
2		Length_low	Length is a 16-bit field that specifies the entire command or data packet size in bytes.
3		Length_high	
4		Crc16_low	This is a 16-bit field. The CRC16 value covers entire framing packet, including the start byte and command or data packets, but does not include the CRC bytes. See CRC16 algorithm.
5		Crc16_high	
6....n		Command or Data packet payload	

A special framing packet that contains only a start byte and a packet type is used for synchronization between the host and target.

Table 142. Special framing packet format

Byte	Value	Parameter
0	0x5A	Start byte
1	0xA*n*	packetType

The Packet Type field specifies the type of the packet from one of the defined types (below):

Table 143. Packet type field

Packet type	Name	Description
0xA1	kFramingPacketType_Ack	The previous packet was received successfully; the sending of more packets is allowed.
0xA2	kFramingPacketType_Nak	The previous packet was corrupted and must be re-sent.
0xA3	kFramingPacketType_AckAbort	Data phase is being aborted.

Table 144. Packet type field

Packet type	Name	Description
0xA4	kFramingPacketType_Command	The framing packet contains a command packet payload.
0xA5	kFramingPacketType_Data	The framing packet contains a data packet payload.

Table continues on the next page...

Table 144. Packet type field (continued)

Packet type	Name	Description
0xA6	kFramingPacketType_Ping	Sent to verify the other side is alive. Also used for UART autobaud.
0xA7	kFramingPacketType_PingResponse	A response to Ping. It contains the framing protocol version number and options.

27.4.5 CRC16 algorithm

The CRC is computed over each byte in the framing packet header, excluding the CRC16 field itself, and all of the payload bytes. The CRC algorithm is the XMODEM variant of

CRC16.

The characteristics of the XMODEM variants are:

Table 145. CRC16 algorithm

Width	16
Polynomial	0x1021
Init value	0x0000
Reflect in	False
Reflect out	False
Xor out	0x0000
Check result	0x31c3

```
The check result is computed by running the ASCII character sequence "123456789" through
the algorithm.
```

```
uint16_t crc16_update(const uint8_t * src, uint32_t lengthInBytes)
 { uint32_t crc = 0; uint32_t j; for (j=0; j < lengthInBytes; ++j)
 { uint32_t i; uint32_t byte = src[j];
crc ^= byte << 8; for (i = 0; i < 8; ++i)
 {
 uint32_t temp = crc << 1; if (crc & 0x8000)
 {
 temp ^= 0x1021;
 }
 crc = temp;
 } } return crc; }
```

27.4.6 Command packet

The command packet carries a 32-bit command header and a list of 32-bit parameters.

Table 146. Command packet format

Command Packet Format (32 bytes)										
Command Header (4 bytes)				28 bytes for Parameters (Max 7 parameters)						
Tag	Flags	Rsvd	Param Count	Param 1 (32-bit)	Param 2 (32-bit)	Param 3 (32-bit)	Param 4 (32-bit)	Param 5 (32-bit)	Param 6 (32-bit)	Param 7 (32-bit)

Table 147. Command header format

Byte #	Command header field	Reset value
0	Command or Response tag	The command header is 4 bytes long with these fields.
1	Flags	
2	Reserved. Should be 0x00.	
3	ParameterCount	

The header is followed by 32-bit parameters up to the value of the ParameterCount field specified in the header. Because a command packet is 32 bytes long, only seven parameters can fit into the command packet.

Command packets are also used by the target to send responses back to the host. As mentioned earlier, command packets and data packets are embedded into framing packets for all of the transfers.

Table 148. Command tags

Command tag	Name	Description
0x01	FlashEraseAll	The command tag specifies one of the commands supported by the bootloader. The valid command tags for the bootloader are listed here.[1]
0x02	FlashEraseRegion	
0x03	ReadMemory	
0x04	WriteMemory	
0x05	FillMemory	
0x06	Reserved	
0x07	GetProperty	
0x08	Reserved	
0x09	Execute	
0x0A	Call	
0x0B	Reset	

Table continues on the next page...

Table 148. Command tags (continued)

Command tag	Name	Description
0x0C	SetProperty	
0x0D	Reserved	
0x0E	eFuseProgram/FlashProgramOnce	
0x0F	eFuseRead/FlashReadOnce	
0x10	Reserved	
0x11	ConfigureMemory	
0x12	Reserved	
0x13	Reserved	
0x14	Reserved	
0x15	Reserved	
0x16	Reserved	

1. The GetProperty, Reset, and SetProperty are allowed in limited ISP mode (CMPA_DIGEST written).

Table 149. Response tags

Response tag	Name	Description
0xA0	GenericResponse	The response tag specifies one of the responses the bootloader (target) returns to the host. The valid response tags are listed here.
0xA3	ReadMemoryResponse	
0xA7	GetPropertyResponse (used for sending responses to GetProperty command only)	
0xA3	ReadMemoryResponse (used for sending responses to ReadMemory command only)	
0xAF	FlashReadOnceResponse (used for sending responses to FlashReadOnce command only)	

Flags: Each command packet contains a flag byte. Only bit 0 of the flag byte is used. If bit 0 of the flag byte is set to 1, then data packets follow the command sequence. The number of bytes that are transferred in the data phase is determined by a command specific parameter in the parameters array.

ParameterCount: The number of parameters included in the command packet.

Parameters: The parameters are word-length (32 bits). With the default maximum packet size of 32 bytes, a command packet can contain up to seven parameters.

27.4.7 Response packet

The responses are carried using the same command packet format wrapped with framing packet data. Types of responses include:

- GenericResponse.
- GetPropertyResponse.
- ReadMemoryResponse.
- FlashReadOnceResponse.

GenericResponse: After the bootloader has processed a command, the bootloader sends a generic response with status and command tag information to the host. The generic response is the last packet in the command protocol sequence. The generic response packet contains the framing packet data and the command packet data (with generic response tag = 0xA0) and a list of parameters (defined in the next section). The parameter count field in the header is always set to 2, for status code and command tag parameters.

Table 150. GenericResponse parameters

Byte #	Parameter	Description
0 - 3	Status code	The Status codes are errors encountered during the execution of a command by the target. If a command succeeds, then a kStatus_Success code is returned.
4 - 7	Command tag	The Command tag parameter identifies the response to the command sent by the host.

GetPropertyResponse: The GetPropertyResponse packet is sent by the target in response to the host query that uses the GetProperty command. The GetPropertyResponse packet contains the framing packet data and the command packet data, with the command/response tag set to a GetPropertyResponse tag value (0xA7).

The parameter count field in the header is set to greater than 1, to always include the status code and one or many property values.

Table 151. GetPropertyResponse parameters

Byte #	Value	Parameter
0 - 3		Status code
4 - 7		Property value
. . .		. . .
		Can be up to maximum 6 property values, limited to the size of the 32-bit command packet and property type.

ReadMemoryResponse: The ReadMemoryResponse packet is sent by the target in response to the host sending a ReadMemory command. The ReadMemoryResponse packet contains the framing packet data and the command packet data, with the command/response tag set to a ReadMemoryResponse tag value (0xA3), the flags field set to kCommandFlag_HasDataPhase (1).

The parameter count set to two for the status code and the data byte count parameters shown below.

Table 152. ReadMemoryResponse parameters

Byte #	Parameter	Description
0 - 3	Status code	The status of the associated Read Memory command.
4 - 7	Data byte count	The number of bytes sent in the data phase.

FlashReadOnceResponse: The FlashReadOnceResponse packet is sent by the target in response to the host sending a FlashReadOnce command. The FlashReadOnceResponse packet contains the framing packet data and the command packet data, with the command/response tag set to a FlashReadOnceResponse tag value (0xAF), and the flags field set to 0. The parameter count is set to 2 plus *the number of words* requested to be read in the FlashReadOnceCommand.

Table 153. FlashReadOnceResponse parameters

Byte #	Value	Parameter
0 - 3		Status code
4 - 7		Byte count to read
. . .		. . .
		Can be up to 20 bytes of requested read data.

27.5 The bootloader command set

27.5.1 Introduction

All bootloader commands follow the command packet format wrapped by the framing packet as explained in previous sections.

For a list of status codes returned by bootloader see Bootloader Status Error Codes.

27.5.2 GetProperty command

The GetProperty command is used to query the bootloader about various properties and settings. Each supported property has a unique 32-bit tag associated with it. The tag occupies the first parameter of the command packet. The target returns a

GetPropertyResponse packet with the property values for the property identified with the tag in the GetProperty command.

Properties are the defined units of data that can be accessed with the GetProperty or SetProperty commands. Properties may be read-only or read-write. All read-write properties are 32-bit integers, so they can easily be carried in a command parameter.

The 32-bit property tag is the only parameter required for GetProperty command.

Table 154. Parameters for GetProperty Command

Byte #	Parameter
0 - 3	Property tag
4 - 7	External Memory Identifier (only applies to get property for external memory, or status identifier if the property tag is equal to 8).

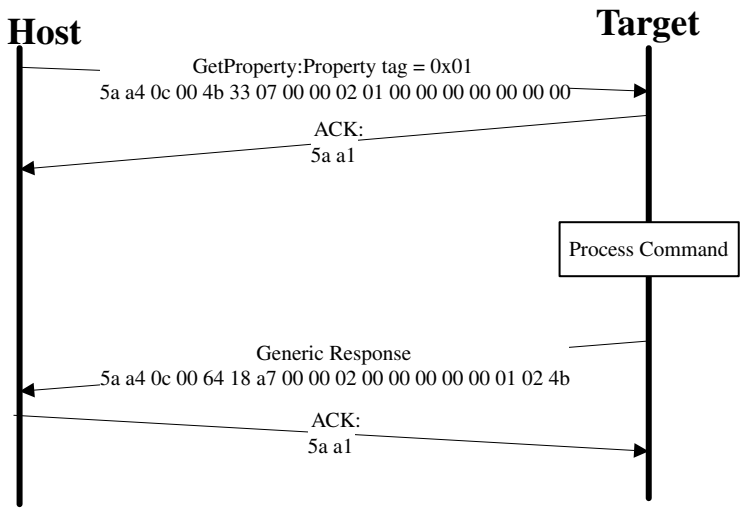


Figure 101. Protocol sequence for GetProperty command

Table 155. GetProperty command packet format (example)

GetProperty	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x0C 0x00
	CRC16	0x4B 0x33

Table 156. GetProperty command packet format (example)

GetProperty	Parameter	Value
Command packet	CommandTag	0x07 – GetProperty
	Flags	0x00
	Reserved	0x00
	ParameterCount	0x02
	PropertyTag	0x00000001 - CurrentVersion
	Memory ID	0x00000000 - Internal Flash

The GetProperty command has no data phase.

Response: In response to a GetProperty command, the target sends a

GetPropertyResponse packet with the response tag set to 0xA7. The parameter count indicates the number of parameters sent for the property values, with the first parameter showing status code 0, followed by the property value(s). Table 157 shows an example of a GetProperty response packet.

Table 157. GetProperty response packet format (example)

GetPropertyResponse	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x0c 0x00 (12 bytes)
	CRC16	0x07 0x7a
Command packet	ResponseTag	0xA7
	Flags	0x00
	Reserved	0x00
	ParameterCount	0x02
	Status	0x00000000
	PropertyValue	0x0000014b - CurrentVersion

27.5.3 SetProperty command

The SetProperty command is used to change or alter the values of the properties or options of the bootloader. The command accepts the same property tags used with the GetProperty command. However, only some properties are writable. If an attempt to write a read-only property is made, an error is returned indicating the property is read-only and cannot be changed.

The property tag and the new value to set are the two parameters required for the SetProperty command.

Table 158. Parameters for SetProperty command

Byte #	Command
0 - 3	Property tag
4 - 7	Property value

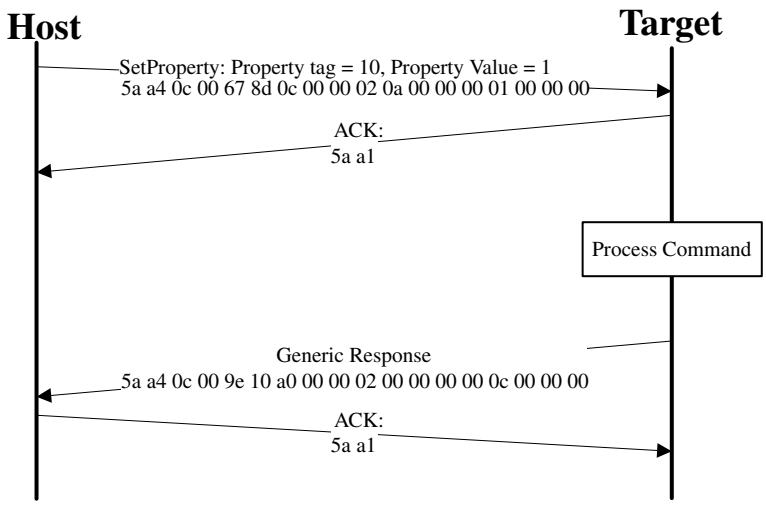


Figure 102. Protocol sequence for SetProperty Command

Table 159. SetProperty command packet format (example)

SetProperty	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x0C 0x00
	CRC16	0x67 0x8D
Command packet	CommandTag	0x0C – SetProperty with property tag 10
	Flags	0x00
	Reserved	0x00
	ParameterCount	0x02
	PropertyTag	0x0000000A - VerifyWrites
	PropertyValue	0x00000001

The SetProperty command has no data phase.

Response: The target returns a GenericResponse packet with one of following status codes:

Table 160. SetProperty response status codes

Status code
kStatus_Success
kStatus_ReadOnly

Table continues on the next page...

Table 160. SetProperty response status codes (continued)

Status code
kStatus_UnknownProperty
kStatus_InvalidArgument

27.5.4 FlashEraseAll command

The FlashEraseAll command performs an erase of the entire flash memory. If any flash regions are protected, then the FlashEraseAll command fails and returns an error status code. The Command tag for FlashEraseAll command is 0x01 set in the commandTag field of the command packet.

The FlashEraseAll command requires memory ID. If memory ID is not specified, the internal flash (memory ID =0) will be selected as default.

Table 161. Parameter for FlashEraseAll command

Byte #	Parameter	
0-3	Memory ID	
	0x000	Internal Flash
	0x09	FlexSPI NOR
	0x110	Serial NOR/EEPROM through SPI

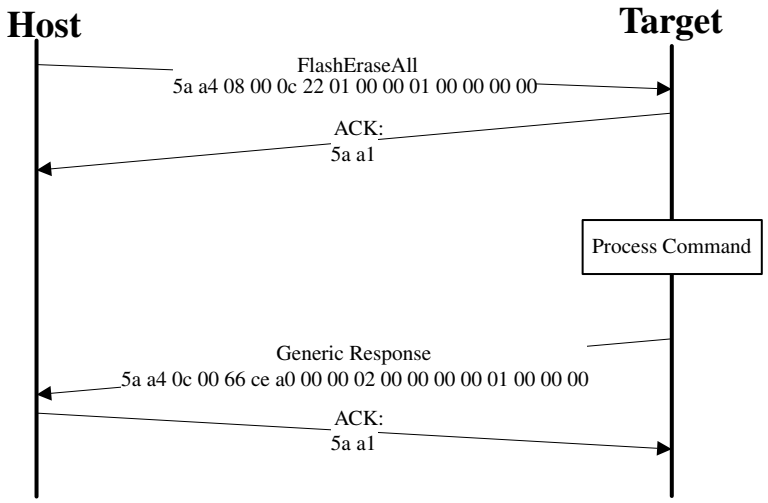


Figure 103. FlashEraseAll command

Table 162. FlashEraseAll command packet format (example)

FlashEraseAll	Parameter	Value
Framing packet	Start byte	0x5A

Table continues on the next page...

Table 162. FlashEraseAll command packet format (example) (continued)

FlashEraseAll	Parameter	Value
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x08 0x00
	Crc16	0x0C 0x22
Command packet	CommandTag	0x01 - FlashEraseAll
	Flags	0x00
	Reserved	0x00
	ParameterCount	0x01
	Memory ID	Refer the above table

The FlashEraseAll command has no data phase.

Response: The target returns a GenericResponse packet with status code either set to kStatus_Success for successful execution of the command or set to an appropriate error status code.

27.5.5 FlashEraseRegion command

The FlashEraseRegion command performs an erase of one or more sectors of the flash memory.

The start address, and number of bytes are the two parameters required for the FlashEraseRegion command. The start and byte count parameters must be 4-byte

aligned ([1:0] = 00), or the FlashEraseRegion command fails and returns

kStatus_FlashAlignmentError (101). If the region specified does not fit in the flash memory space, the FlashEraseRegion command fails and returns kStatus_FlashAddressError (102). If any part of the region specified is protected, the FlashEraseRegion command fails and returns kStatus_MemoryRangeInvalid (10200).

Table 163. Parameter for FlashEraseRegion command

Byte #	Parameter
0-3	Start address
4 - 7	Byte count
8 - 11	Memory ID

The FlashEraseRegion command has no data phase.

Response: The target returns a GenericResponse packet with one of the following error status codes.

Table 164. FlashEraseRegion response status codes

Status code
kStatus_Success (0).

Table continues on the next page...

Table 164. FlashEraseRegion response status codes (continued)

Status code
kStatus_MemoryRangeInvalid (10200).
kStatus_FlashAlignmentError (101).
kStatus_IFlashAddressError (102).

Table 165. FlashEraseRegion response status codes

Status code
kStatus_FlashAccessError (103).
kStatus_FlashProtectionViolation (104).
kStatus_FlashCommandFailure (105).

27.5.6 ReadMemory command

The ReadMemory command returns the contents of memory at the given address, for a specified number of bytes. This command can read any region of memory accessible by the CPU and not protected by security.

The start address, and number of bytes are the two parameters required for ReadMemory command. The memory ID is optional. Internal memory will be selected as default if memory ID is not specified.

Table 166. Parameter for read memory command

Byte #	Parameter	Description
0-3	Start address	Start address of memory to read from.
4-7	Byte count	Number of bytes to read and return to caller.
8-11	Memory ID	Internal or external memory Identifier.

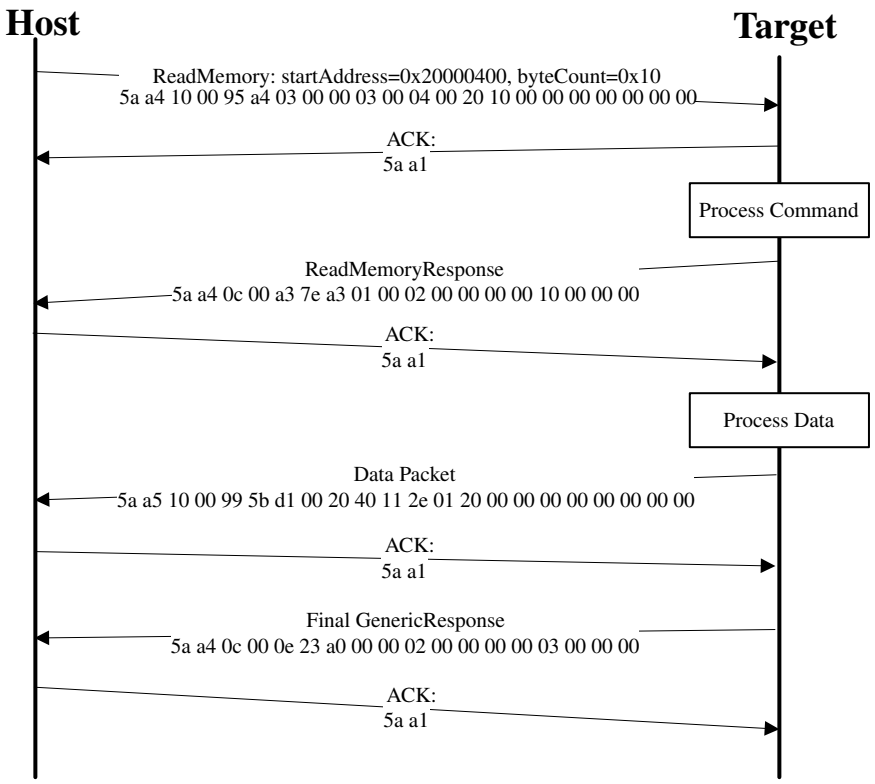


Figure 104. Command sequence for ReadMemory

Table 167. ReadMemory command packet format (example)

ReadMemory	Parameter	Value
Framing packet	Start byte	0x5A
	PpacketType	0xA4, kFramingPacketType_Command
	Length	0x10 0x00
	Crc16	0xF4 0x1B
Command packet	CommandTag	0x03 - ReadMemory
	Flags	0x00
	Reserved	0x00
	ParameterCount	0x03
	StartAddress	0x20000400
	ByteCount	0x00000064
	Memory ID	0x0

Data Phase: The ReadMemory command has a data phase. Because the target works in slave mode, the host needs to pull data packets until the number of bytes of data specified in the byteCount parameter of ReadMemory command are received by host.

Response: The target returns a GenericResponse packet with a status code either set to kStatus_Success upon successful execution of the command or set to an appropriate error status code

27.5.7 WriteMemory command

The WriteMemory command writes data provided in the data phase to a specified range of bytes in memory (flash or RAM). However, if flash protection is enabled, then writes to protected sectors fail.

Special care must be taken when writing to flash.

- First, any flash sector written to must have been previously erased with a FlashEraseAll or FlashEraseRegion command.
- Writing to flash requires the start address to be page aligned.
- The byte count is rounded up to a page size, and trailing bytes are filled with the flash erase pattern (0xff).
- If the VerifyWrites property is set to true, then writes to flash also performs a flash verify program operation.

When writing to RAM, the start address does not need to be aligned, and the data is not padded.

The start address and number of bytes are the two parameters required for WriteMemory command. The memory ID is optional. Internal memory will be selected as default if memory ID is not specified.

Table 168. Parameters for WriteMemory command

Byte #	Command
0-3	Start address
4-7	Byte count
8-11	Memory ID

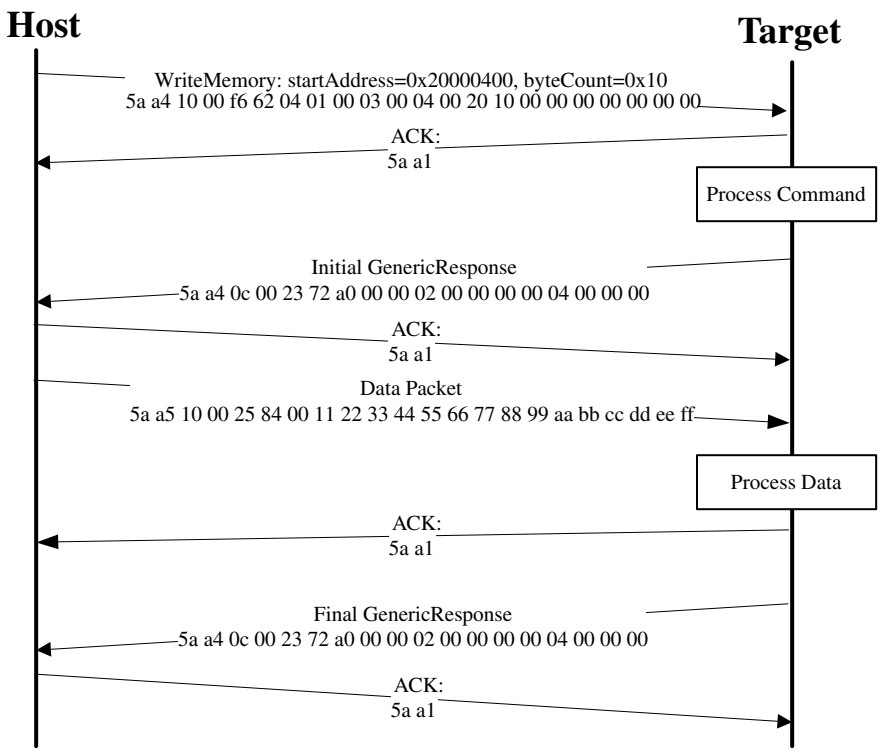


Figure 105. Protocol sequence for WriteMemory command

Table 169. WriteMemory command packet format (example)

WriteMemory	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x10 0x00
	Crc16	0x97 0xDD
Command packet	CommandTag	0x04 - WriteMemory
	Flags	0x01
	Reserved	0x00
	ParameterCount	0x03
	StartAddress	0x20000400
	ByteCount	0x00000064
	Memory ID	0x0

Data Phase: The WriteMemory command has a data phase; the host sends data packets until the number of bytes of data specified in the byteCount parameter of the WriteMemory command are received by the target.

Response: The target returns a GenericResponse packet with a status code set to kStatus_Success upon successful execution of the command, or to an appropriate error status code.

27.5.8 FillMemory command

The FillMemory command fills a range of bytes in memory with a data pattern. It follows the same rules as the WriteMemory command. The difference between FillMemory and WriteMemory is that a data pattern is included in FillMemory command parameter, and there is no data phase for the FillMemory command, while WriteMemory does have a data phase.

Table 170. Parameters for FillMemory command

Byte #	Command
0-3	Start address of memory to fill.
4-7	Number of bytes to write with the pattern • The start address should be 32-bit aligned. • The number of bytes must be evenly divisible by 4.
8-11	32-bit pattern.

- To fill with a byte pattern (8-bit), the byte must be replicated four times in the 32-bit pattern.
- To fill with a short pattern (16-bit), the short value must be replicated two times in the32-bit pattern.

For example, to fill a byte value with 0xFE, the word pattern is 0xFEFEFEFE; to fill a short value 0x5AFE, the word pattern is 0x5AFE5AFE.

Special care must be taken while writing to flash.

- First, any flash sector written to must have been previously erased with a FlashEraseAll, or FlashEraseRegion command.
- Writing to flash requires the start address to be 4-byte aligned ([1:0] = 00).
- If the VerifyWrites property is set to true, then writes to flash also performs a flash verify program operation.

When writing to RAM, the start address does not need to be aligned, and the data is not padded.

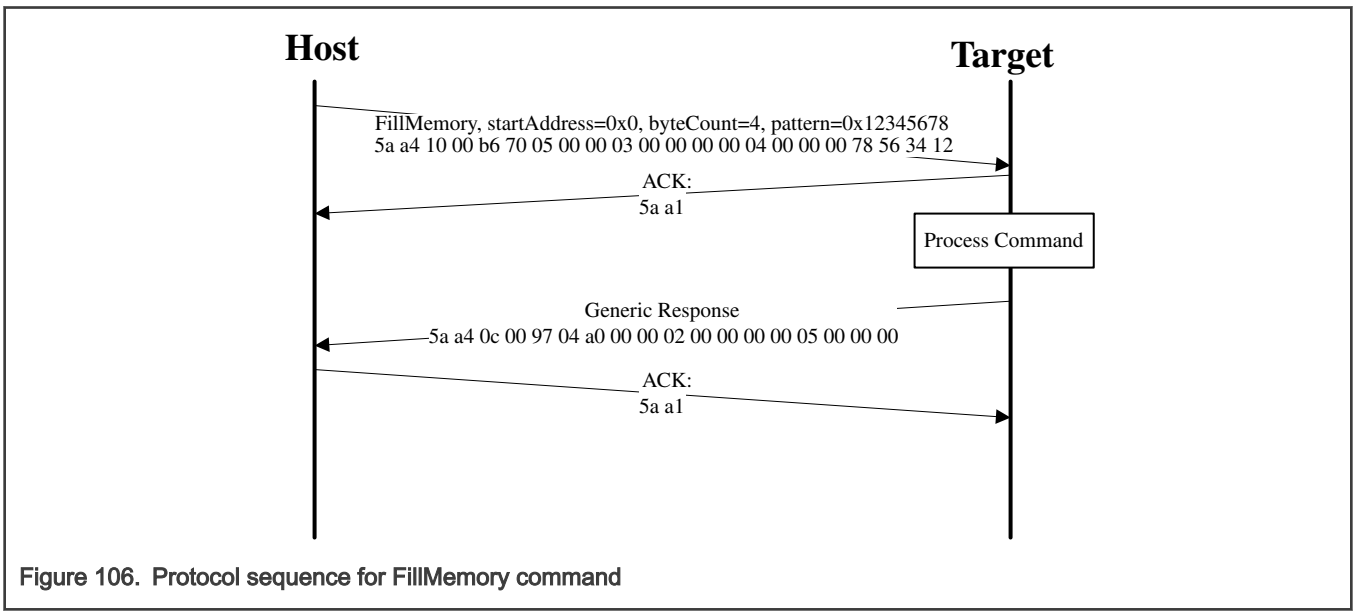


Figure 106. Protocol sequence for FillMemory command

Table 171. FillMemory command packet format (example)

FillMemory	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x10 0x00
	Crc16	0xE4 0x57
Command packet	CommandTag	0x05 – FillMemory
	Flags	0x01
	Reserved	0x00
	ParameterCount	0x03
	StartAddress	0x00007000
	ByteCount	0x00000800
	PatternWord	0x12345678

The FillMemory command has no data phase.

Response: upon successful execution of the command, the target (bootloader) returns a GenericResponse packet with a status code set to kStatus_Success, or to an appropriate error status code.

27.5.9 Execute command

The Execute command results in the bootloader setting the program counter to the code at the provided jump address, R0 to the provided argument, and a Stack pointer to the provided stack pointer address. Prior to the jump, the system is returned to the reset state.

The Jump address, function argument pointer, and stack pointer are the parameters required for the Execute command. If the stack pointer is set to zero, the called code is responsible for setting the processor stack pointer before using the stack.

Table 172. Parameters for Execute command

Byte #	Command
0-3	Jump address.
4-7	Argument word.
8-11	Stack pointer address.

The Execute command has no data phase.

Response: Before executing the Execute command, the target validates the parameters and return a GenericResponse packet with a status code either set to kStatus_Success or an appropriate error status code.

27.5.10 Reset command

The Reset command results in the bootloader resetting the chip.

The Reset command requires no parameters.

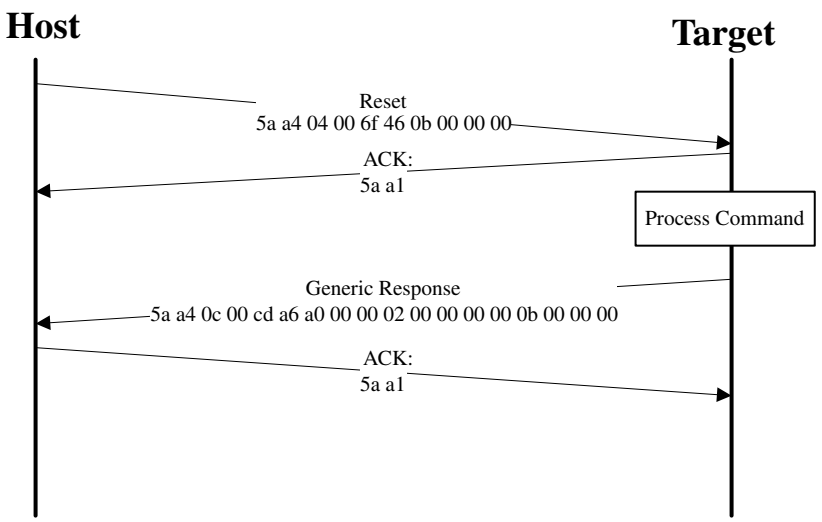


Figure 107. Protocol sequence for Reset command

Table 173. Reset command packet format (example)

Reset	Parameter	Value
Framing packet	Start byte	0x5A
	PacketType	0xA4, kFramingPacketType_Command
	Length	0x04 0x00
	Crc16	0x6F 0x46
Command packet	CommandTag	0x0B - reset
	Flags	0x00
	Reserved	0x00
	ParameterCount	0x03

The Reset command has no data phase.

Response: The target returns a GenericResponse packet with status code set to kStatus_Success, before resetting the chip.

The reset command can also be used to switch boot from flash after successful flash image provisioning via ROM bootloader. After issuing the reset command, allow five seconds for the user application to start running from Flash.

27.5.11 eFuseProgramOnce/FlashProgramOnce command

The FlashProgramOnce command writes data (that is provided in a command packet) to a specified range of bytes in the program once field. Special care must be taken when writing to the program once field.

• The program once field only supports programming once, so any attempted to reprogram a program-once field gets an error response.

• Writing to the program once field requires the byte count to be 4-byte aligned or 8-byte aligned. The FlashProgramOnce command uses three parameters: index 2, byteCount, data.

Table 174. Parameters for FlashProgramOnce Command

Byte #	Command
0 - 3	Index of program once field
4 - 7	Byte count (must be evenly divisible by 4)
8 - 11	Data

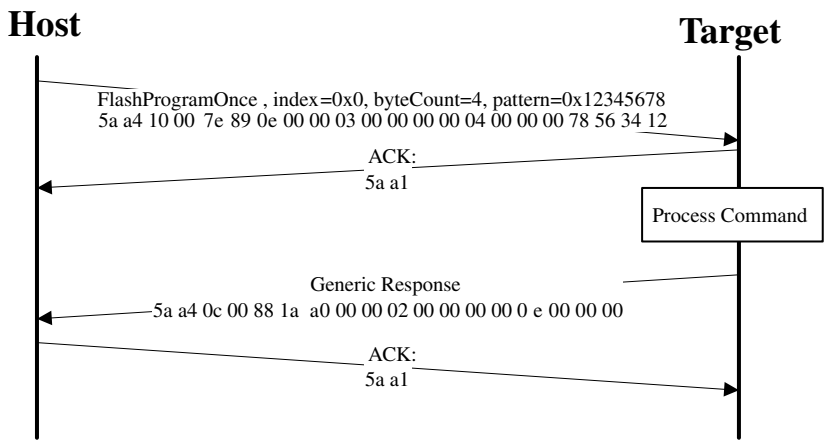


Figure 108. Protocol Sequence for FlashProgramOnce Command

Table 175. FlashProgramOnce Command Packet Format

FlashProgramOnce	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4, kFramingPacketType_Command
	length	0x10 0x00
	CRC16	0x7E4 0x89
Command packet	commandTag	0x0E – FlashProgramOnce
	flags	0
	reserved	0
	parameterCount	3

Table continues on the next page...

Table 175. FlashProgramOnce Command Packet Format (continued)

FlashProgramOnce	Parameter	Value
	index	0x0000_0000
	byteCount	0x0000_0004
	Data	0x1234_5678

Response: upon successful execution of the command, the target (MCU bootloader) returns a GenericResponse packet with a status code set to kStatus_Success, or to an appropriate error status code.

27.5.12 eFuseReadOnce / FlashReadOnce command

The FlashReadOnce command returns the contents of the program once field by given index and byte count. The FlashReadOnce command uses 2 parameters: index and byteCount. eFuseReadOnce command uses only one parameter index.

Table 176. Parameters for FlashReadOnce Command

Byte #	Parameter	Description
0 - 3	Index	Index of the program once field (to read from)
4 - 7	byteCount	Number of bytes to read and return to the caller

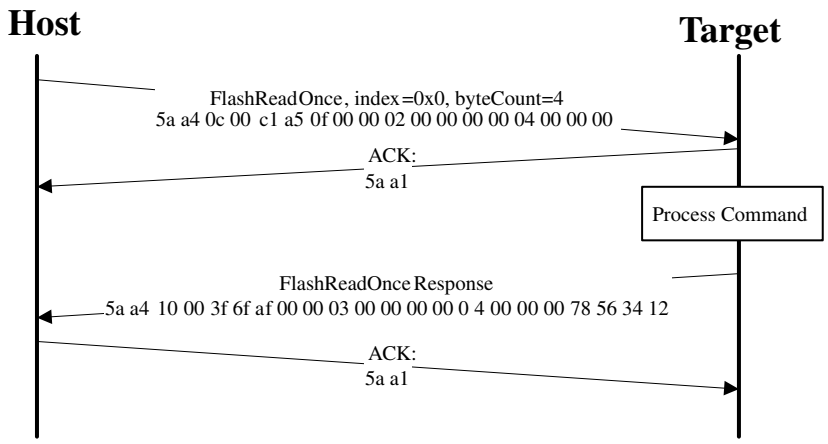


Figure 109. Protocol Sequence for FlashReadOnce Command

Table 177. FlashReadOnce Command Packet Format

FlashReadOnce	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4
	Length	0x0C 0x00

Table continues on the next page...

Table 177. FlashReadOnce Command Packet Format (continued)

FlashReadOnce	Parameter	Value
	CRC16	0xC1 0xA5
Command packet	commandTag	0x0F – FlashReadOnce
	Flags	0x00
	Reserved	0x00
	parameterCount	0x02
	Index	0x0000_0000
	byteCount	0x0000_0004

Table 178. FlashReadOnce Response Format

FlashReadOnce Response	Parameter	Value
Framing packet	start byte	0x5A
	packetType	0xA4
	Length	0x10 0x00
	CRC16	0x3F 0x6F
Command packet	commandTag	0xAF
	Flags	0x00
	Reserved	0x00
	parameterCount	0x03
	Status	0x0000_0000
	byteCount	0x0000_0004
	Data	0x1234_5678

Response: upon successful execution of the command, the target returns a

FlashReadOnceResponse packet with a status code set to kStatus_Success, a byte count and corresponding data read from Program Once Field upon successful execution of the command or returns with a status code set to an appropriate error status code and a byte count set to 0.

27.5.13 ConfigureMemory command

The ConfigureMemory command configures the internal/external memory device using a pre-programmed configuration block. The parameters passed in the command are the memory ID, and then the memory address from which the configuration data can

be loaded from. Options for loading the data can be a scenario where the configuration data is written to a RAM or flash location and then this command directs the bootloader to use the data at that location to configure the external memory devices.

Table 179. Parameters for ConfigureMemory command

Byte #	Command
0-3	Memory ID.
4-7	Configuration block address.

Response: The target (Bootloader) returns a GenericResponse packet with a status code either set to kStatus_Success upon successful execution of the command or set to an appropriate error code.

The following table shows the supported memory IDs.

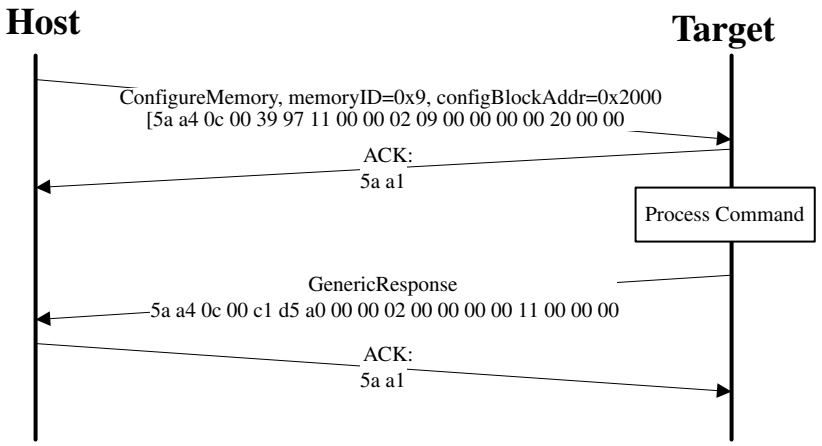


Figure 110. Protocol sequence for ConfigureMemory command

Table 180. Supported memory IDs

Memory ID	Description
0	Internal RAM/FLASH.
0x9	External FlexSPI device. See Serial NOR Flash through FlexSPI.
0x110	External 1-bit SPI NOR FLASH device. See 1-bit Serial NOR flash through SPI.

27.5.14 Get/SetProperty command properties

This section lists the properties of the GetProperty and SetProperty commands.

Table 181. Properties used by Get/SetProperty commands, sorted by values

Property	Writable	Tag Value	Size	Description
CurrentVersion	No	01h	4	Current bootloader version.
AvailablePeripherals	No	02h	4	The set of peripherals supported on this chip.

Table continues on the next page...

Table 181. Properties used by Get/SetProperty commands, sorted by values (continued)

FlashStartAddress	No	03h	4	Start address of program flash.
FlashSizeInBytes	No	04h	4	Size in bytes of program flash.
AvailableCommands	No	07h	4	The set of commands supported by the bootloader
Check Status	No	08h	4	Return the status based on specified status identifier 0 - CRC status 32-bit return value for CRC Check 10401 - Application CRC check failed 10402 - Application CRC check is inactive 10403 - Application CRC check is invalid 1 - Last Error See the details of last error in GetLastError property.
MaxPacketSize	No	0Bh	4	Maximum supported packet size for the currently active peripheral interface.
ReservedRegions	No	0Ch	8*n	List of memory regions reserved by the bootloader. Returned as value pairs (<start-address-of-region>,<end-address-of-region>. If HasDataPhase flag is not set, then the Response packet parameter count indicates the number of pairs. If HasDataPhase flag is set, then the second parameter is the number of bytes in the data phase. "n" indicates number of memory region pairs
SystemDeviceId	No	10h	4	Value of the DEVICD_ID and DIEID
LifeCycleState	No	11h	4	The life cycle of the device. 0x5aa55aa5 - Device is in development life cycle. 0xc33cc33c - Device is in deployment life cycle.
UniqueDevice/UUID	No	12h	16	Unique device identification
Target ROM Version	No	18h	4	The target build version number
ExternalMemoryAttributes	No	19h	24	List of attributes supported by the specified memory Id (0x110=SPI NOR FLASH). See description for the return value in the section ExternalMemoryAttribute s Property
IrqNotifierPin	Yes	1ch	4	IRQ Notifier Pin setting

Table continues on the next page...

Table 181. Properties used by Get/SetProperty commands, sorted by values (continued)

				bit[7:0] - gpio pin bit[15:8] - gpio port bit [30:16] - reserved bit [31] - enable flag, 0 - disable, 1 - enable

27.5.14.1 Property definitions

Get/Set property definitions are provided in this section.

27.5.14.1.1 CurrentVersion property

The value of this property is a 4-byte structure containing the current version of the bootloader.

Table 182. CurrentVersion property fields

Bit	[31:24]	[23:16]	[15:8]	[7:0]
Field	Name = 'K' (0x4B)	Major version	Major version	Bugfix version

27.5.14.1.2 AvailablePeripherals property

The value of this property is a bitfield that lists the peripherals supported by the bootloader and the hardware on which it is running.

Table 183. Peripheral bits

Bit	[31:7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
Field	Reserved	Reserved	Reserved	USB HID	CAN	SPI Slave	I2C Slave	LPUART

If the peripheral is available, then the corresponding bit will be set in the property value. All reserved bits must be set to 0.

27.5.14.1.3 Available Commands property

This property value is a bitfield with set bits indicating the commands enabled in the bootloader. Only commands that can be sent from the host to the target are listed in the bitfield. Response commands such as GenericResponse are excluded.

The bit number that identifies whether a command is present is the command's tag value minus 1. 1 is subtracted from the command tag because the lowest command tag value is 0x01. To get the bit mask for a given command, use this expression: mask = 1 << (tag - 1).

Table 184. Command bits

Bit	[Others]	[20]	[16]	[15]	[14]	[13]	[12]	[11]	[10]	[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
Command																			

27.5.14.1.4 ExternalMemoryAttributes property

The value returned by this property is a 24-byte data structure containing available external memory attributes: start address, total size in KB, page size, sector size, and block size. Below is the breakdown of 24-byte structure.

Table 185. Fields of ExternalMemoryAttributes property

Field offset	Field Description
0 - 3	The value returned is a bitmap showing the supported attributes for the external memory, with the corresponding bitfield set. 0x00000001 - start address 0x00000002 - total size 0x00000004 - page size 0x00000008 - sector size 0x00000010 - block size
4 - 7	Start address of external memory.
8 - 11	Total size of external memory in kilobytes.
12 - 15	Page size of external memory in bytes.
16 - 19	Sector size of external memory in bytes.
20 - 23	Block size of external memory in bytes.

27.5.14.1.5 GetLastError property

The following table lists the response words and corresponding error conditions.

Table 186. Response word and error description

Reponse word	Error	Description
0x0b35f300	kLog_Auth_ImageEntryCheck_Fail	Application entry point is invalid or stack address is invalid.
0x0b37f300	kLog_Auth_CrcCheck_Fail	CRC checksum is wrong.
0x0b37f301	kLog_Auth_CrcCheck_Fail_Time	The image CRC check failed and the cost time has been loaded into the log data.
0x0b37fc00	kLog_Auth_CrcCheck_Invalid	The image type or image length zero not meet the requirement of the crc image.
0x0c00f300	kLog_Jump_Fail	The image jump fail.

Table 187. Response word and error description

Reponse word	Error	Description
0x0c00f500	kLog_Jump_Fail_Fatal	Failed to jump to application.
0x0702f30<n>	kLog_Recoveryboot_Fail_Reason	Recovery boot failed.
0x0501f300	kLog_Masterboot_Init_Fail	The master boot init fail.
0x0801f300	kLog_Ispboot_Init_Fail	The isp boot init fail, means no available isp peripherals.

Table continues on the next page...

Table 187. Response word and error description (continued)

Reponse word	Error	Description
0x0901f300	kLog_BootDevice_Init_Fail	The boot device init fail.
0x0902f300	kLog_BootDevice_Read_Fail	The master boot read the initial image fail.
0x0a20f300	kLog_ImgLoad_InitLoad_Fail	The master boot load the image header part fail.
0x0a21f300	kLog_ImgLoad_RemainingLoad_Fail	Load the image remaining part except the image header and other configure part fail.

27.6 Bootloader Status Error Codes

This section describes the status error codes that the Bootloader returns to the host.

Table 188. Bootloader status error codes, sorted by value

Error Code	Value	Description
kStatus_Success	0	Operation succeeded without error.
kStatus_Fail	1	The operation failed with a generic error.
kStatus_ReadOnly	2	Requested value cannot be changed because it is read-only.
kStatus_OutOfRange	3	Requested value is out of range.
kStatus_InvalidArgument	4	The requested command's argument is undefined.
kStatus_Timeout	5	A timeout occurred.
kStatus_NoTransferInProgress	6	No send in progress
kStatus_FLASH_Success	0	API is executed successfully
kStatus_FLASH_InvalidArgument	4	An invalid argument is provided
kStatus_FlashSizeError	100	Not used.
kStatus_FlashAlignmentError	101	Address or length does not meet the required alignment.
kStatus_FlashAddressError	102	Address or length is outside addressable memory.
kStatus_FLASH_CommandFailure	105	The INT_STATUS[FAIL] bit is set.
kStatus_FlashUnknownProperty	106	Unknown Flash property.

Table continues on the next page...

Table 188. Bootloader status error codes, sorted by value (continued)

Error Code	Value	Description
kStatus_FlashEraseKeyError	107	The key provided does not match the programmed flash key.
kStatus_FlashRegionExecuteOnly	108	The area of flash is protected as execute-only.
kStatus_FLASH_ExecuteInRamFunctionNotReady	109	Execute-in-RAM function is not available
kStatus_FLASH_CommandNotSupported	111	Flash API is not supported
kStatus_FLASH_ReadOnlyProperty	112	The flash property is read-only
kStatus_FLASH_InvalidPropertyValue	113	The flash property value is out of range
kStatus_FLASH_InvalidSpeculationOption	114	The option of flash prefetch speculation is invalid
kStatus_FLASH_EccError	116	A correctable or uncorrectable error during command execution
kStatus_FLASH_CompareError	117	Destination and source memory contents do not match
kStatus_FLASH_RegulationLoss	118	A loss of regulation during read
kStatus_FLASH_InvalidWaitStateCycles	119	The wait state cycle set to r/w mode is invalid
kStatus_FLASH_OutOfDateCfpaPage	132	CFPA page version is out of date
kStatus_FLASH_BlankIfrPageData	133	Blank page cannnot be read
Reserved	-	-
kStatus_FLASH_ProgramVerificationNotAllowed	135	Program verification is not allowed when the encryption is enabled
kStatus_FLASH_HashCheckError	136	Hash check of page data is failed
kStatus_FLASH_SealedFfrRegion	137	The FFR region is sealed
kStatus_FLASH_FfrRegionWriteBroken	138	The FFR Spec region is not allowed to be written discontinuously
kStatus_FLASH_NmpaAccessNotAllowed	139	The NMPA region is not allowed to be read/written/erased
kStatus_FLASH_CmpaCfgDirectEraseNotAllowed	140	The CMPA Cfg region is not allowed to be erased directly
kStatus_FLASH_FfrBankIsLocked	141	The FFR bank region is locked

Table continues on the next page...

Table 188. Bootloader status error codes, sorted by value (continued)

Error Code	Value	Description
kStatus_FLASH_CfpaScratchPageInvalid	148	CFPA Scratch Page is invalid
kStatus_FLASH_CfpaVersionRollbackDisallowed	149	CFPA version rollback is not allowed
kStatus_FLASH_ReadHidingAreaDisallowed	150	Flash hiding read is not allowed
kStatus_FLASH_ModifyProtectedAreaDisallowed	151	Flash firewall page locked erase and program are not allowed
kStatus_FLASH_CommandOperationInProgress	152	The flash state is busy, indicate that a flash command in progress.
kStatus_UnknownCommand	10000	this command is not recognized
kStatus_SecurityViolation	10001	Security Violation happened when receiving disallowed commands
kStatus_AbortDataPhase	10002	Sender requested data phase abort
kStatus_Ping	10003	Ping Command Received from the host
kStatus_NoResponse	10004	No Response from the host
kStatus_NoResponseExpected	10005	Expected No Response from the host
kStatus_CommandUnsupported	10006	Unsupported command was received
kStatusRomLdrSectionOverrun	10100	means reached the end of the sb file processing
kStatusRomLdrSignature	10101	the signature or version are incorrect
kStatusRomLdrSectionLength	10102	the bootOffset/ new section count is out of range
kStatusRomLdrUnencryptedOnly	10103	the unencrypted image is disabled
kStatusRomLdrEOFReached	10104	the end of the image file is reached
kStatusRomLdrChecksum	10105	The checksum of a command tag block is invalid.
kStatusRomLdrCrc32Error	10106	The CRC-32 of the data for a load command is incorrect.
kStatusRomLdrJumpReturned	10110	The function that was jumped to by the SB file has returned.
kStatusRomLdrRollbackBlocked	10115	An Image version rollback event detected

Table continues on the next page...

Table 188. Bootloader status error codes, sorted by value (continued)

Error Code	Value	Description
kStatusRomLdrPendingJumpCommand	10119	Represent the jump command is pending and the actual jump is implemented in sbloader_finalize()
kStatusMemoryRangeInvalid	10200	The requested address range does not match an entry, or the length extends past the matching entry's end address.
kStatusMemoryReadFailed	10201	Memory Read Failed
kStatusMemoryWriteFailed	10202	Memory Write failed
kStatusMemoryCumulativeWrite	10203	Cumulative Write happened due to write to un-erased FLASH region
kStatusMemoryNotConfigured	10205	Memory is not configured yet before access
kStatusMemoryAlignmentError	10206	Alignment Error happened during access to memory
kStatusMemoryVerifyFailed	10207	Verifying operation failed after erasing/ programming FLASH
kStatusMemoryWriteProtected	10208	The memory to be written is protected
kStatusMemoryAddressError	10209	The Memory address is invalid/wrong
kStatusMemoryBlankCheckFailed	10210	Check the blank memory failled
kStatusMemoryBlankPageReadDisallowed	10211	The memory is blank, and read command is disallowed
kStatusMemoryProtectedPageReadDisallowed	10212	The memory is protected, and read command is disallowed
kStatusMemoryFfrSpecRegionWriteBroken	10213	The write operation to the FFR Region was broken
kStatusMemoryUnsupportedCommand	10214	The memory command is not supported
kStatus_UnknownProperty	10300	The requested property value is undefined.
kStatus_ReadOnlyProperty	10301	The requested property value cannot be written.
kStatus_InvalidPropertyValue	10302	The specified property value is invalid.
kStatus_AppCrcCheckPassed	10400	CRC check is valid and passed.
kStatus_AppCrcCheckFailed	10401	CRC check is valid but failed.

Table continues on the next page...

Table 188. Bootloader status error codes, sorted by value (continued)

Error Code	Value	Description
kStatus_AppCrcCheckInactive	10402	CRC check is inactive.
kStatus_AppCrcCheckInvalid	10403	CRC check is invalid because the BCA is invalid, or the CRC parameters are unset (all 0xFF bytes).
kStatus_AppCrcCheckOutOfRange	10404	CRC check is valid, but addresses are out of range.
kStatus_RomApiExecuteCompleted	0	ROM successfully process the whole sb file/ boot image
kStatus_RomApiNeedMoreData	10801	ROM needs more data to continue processing the boot image
kStatus_RomApiBufferSizeNotEnough	10802	The user buffer is not enough for use by Kboot during execution of the operation
kStatus_RomApiInvalidBuffer	10803	The user buffer is not ok for sbloader or authentication
kStatus_FLEXSPI_SequenceExecutionTimeout	6000	The FLEXSPI Sequence Execution timeout
kStatus_FLEXSPI_InvalidSequence	6001	The FLEXSPI LUT sequence invalid
kStatus_FLEXSPI_DeviceTimeout	6002	The FLEXSPI device timeout
kStatus_FLEXSPINOR_ProgramFail	20100	Page programming failure
kStatus_FLEXSPINOR_EraseSectorFail	20101	Sector Erase failure
kStatus_FLEXSPINOR_EraseAllFail	20102	Chip Erase failure
kStatus_FLEXSPINOR_WaitTimeout	20103	The execution time out
kStatus_FlexSPINOR_NotSupported	20104	Page size overflow
kStatus_FlexSPINOR_WriteAlignmentError	20105	Address alignment error
kStatus_FlexSPINOR_CommandFailure	20106	Erase/Program Verify Error
kStatus_FlexSPINOR_SFDP_NotFound	20107	Timeout occurs during the API call
kStatus_FLEXSPINOR_Unsupported_SFDP_Version	20108	Unrecognized SFDP version
kStatus_FLEXSPINOR_FLASH_NotFound	20109	Flash detection failure

Table continues on the next page...

Table 188. Bootloader status error codes, sorted by value (continued)

Error Code	Value	Description
kStatus_FLEXSPINOR_DTRRead_DummyProbeFailed	20110	DDR Read dummy probe failure
kStatus_IAP_Success	0	IAP API execution succedded
kStatus_IAP_Fail	1	IAP API exeution failed
kStatus_IAP_InvalidArgument	100001	Invalid argument detected during API execution
kStatus_IAP_OutOfMemory	100002	The Heap size is not enough during API execution
kStatus_IAP_ReadDisallowed	100003	The read memory operation is disallowed during API execution
kStatus_IAP_CumulativeWrite	100004	The FLASH region to be programmed is not empty
kStatus_IAP_EraseFailuire	100005	Erase operation failed
kStatus_IAP_CommandNotSupported	100006	The specific command is not supported
kStatus_IAP_MemoryAccessDisabled	100007	Memory access is disabled, typically occurred on the FLEXSPI NOR if it is not configured properly
kStatus_NBOOT_Success	0x5A5A5A5Au	Operation completed successfully in NBOOT functions.
kStatus_NBOOT_Fail	0x5A5AA5A5u	Operation failed in NBOOT functions.
kStatus_NBOOT_InvalidArgument	0x5A5AA5F0u	Invalid argument passed to the function in NBOOT functions.
kStatus_NBOOT_RequestTimeout	0x5A5AA5E1u	Operation timed out in NBOOT functions.
kStatus_NBOOT_KeyNotLoaded	0x5A5AA5E2u	The requested key is not loaded in NBOOT functions.
kStatus_NBOOT_AuthFail	0x5A5AA5E4u	Authentication failed in NBOOT functions.
kStatus_NBOOT_OperationNotAvaialable	0x5A5AA5E5u	Operation not available on this HW in NBOOT functions.
kStatus_NBOOT_KeyNotAvailable	0x5A5AA5E6u	Key is not avaialble in NBOOT functions.
kStatus_NBOOT_SelftestFail	0x5A5AA5E8u	FIPS self-test failure in NBOOT functions.
kStatus_NBOOT_InvalidDataFormat	0x5A5AA5E9u	Invalid data format for example antipole in NBOOT functions.
kStatus_NBOOT_IskCertUserDataTooBig	0x5A5AA5EAu	Size of User data in ISK certificate is greater than 96 bytes in NBOOT functions.

Table continues on the next page...

Table 188. Bootloader status error codes, sorted by value (continued)

Error Code	Value	Description
kStatus_NBOOT_IskCertSignatureOffsetTooSmall	0x5A5AA5EBu	Signature offset in ISK certificate is smaller than expected in NBOOT functions.
kStatus_NBOOT_MemcpyFail	0x5A5A845Au	Unexpected error detected during nboot_memcpy()in NBOOT functions.
kStatus_MKBOOT_Success	0x0000c35au	Operation completed successfully in Master KBOOT.
kStatus_MKBOOT_Fail	0x0000c3c3u	Operation failed in Master KBOOT.
kStatus_MKBOOT_InvalidArgument	0x00005ac3u	Return Invalid argument status in Master KBOOT.

NOTE

In UART, I2C, CANand SPI ISP modes, the LPC55xx expects posted responses to be read by the host within 20ms*number of bytes, otherwise the LPC55xx reports the abort command. Considering an ACK from the LPC55xx is only two bytes, the host must read those bytes within 40ms of the LPC55xx posting.

27.7 UART ISP

27.7.1 Introduction

The bootloader integrates an autobaud detection algorithm for the UART peripheral, thereby providing flexible baud rate choices.

Autobaud feature: If UART*n* is used to connect to the bootloader, then the UART*n*_RX pin must be kept high and not left floating during the detection phase in order to comply with the autobaud detection algorithm. After the bootloader detects the Ping packet (0x5A 0xA6) on UART*n*_RX, the bootloader firmware executes the autobaud sequence.

If the baudrate is successfully detected, then the bootloader sends a Ping packet response [(0x5A 0xA7), protocol version (4 bytes), protocol version options (2 bytes) and crc16 (2 bytes)] at the detected baudrate. The bootloader then enters a loop, waiting for bootloader commands via the UART peripheral.

NOTE: The data bytes of the ping packet must be sent continuously (with no more than 80 ms between bytes) in a fixed UART transmission mode (8-bit data, no parity bit and 1 stop bit). If the bytes of the ping packet are sent one-by-one with more than 80 ms delay between them, then the autobaud detection algorithm may calculate an incorrect baud rate. In this instance, the autobaud detection state machine should be reset.

Supported baud rates: The baud rate is closely related to the MCU core and system clock frequencies. Typical baud rates supported are 9600, 19200, 38400, 57600, 115200 ,230400, 460800 and 1000000.

Packet transfer: After autobaud detection succeeds, bootloader communications can take place over the UART peripheral. The following flow charts show:

- How the host detects an ACK from the target.
- How the host detects a ping response from the target.
- How the host detects a command response from the target.

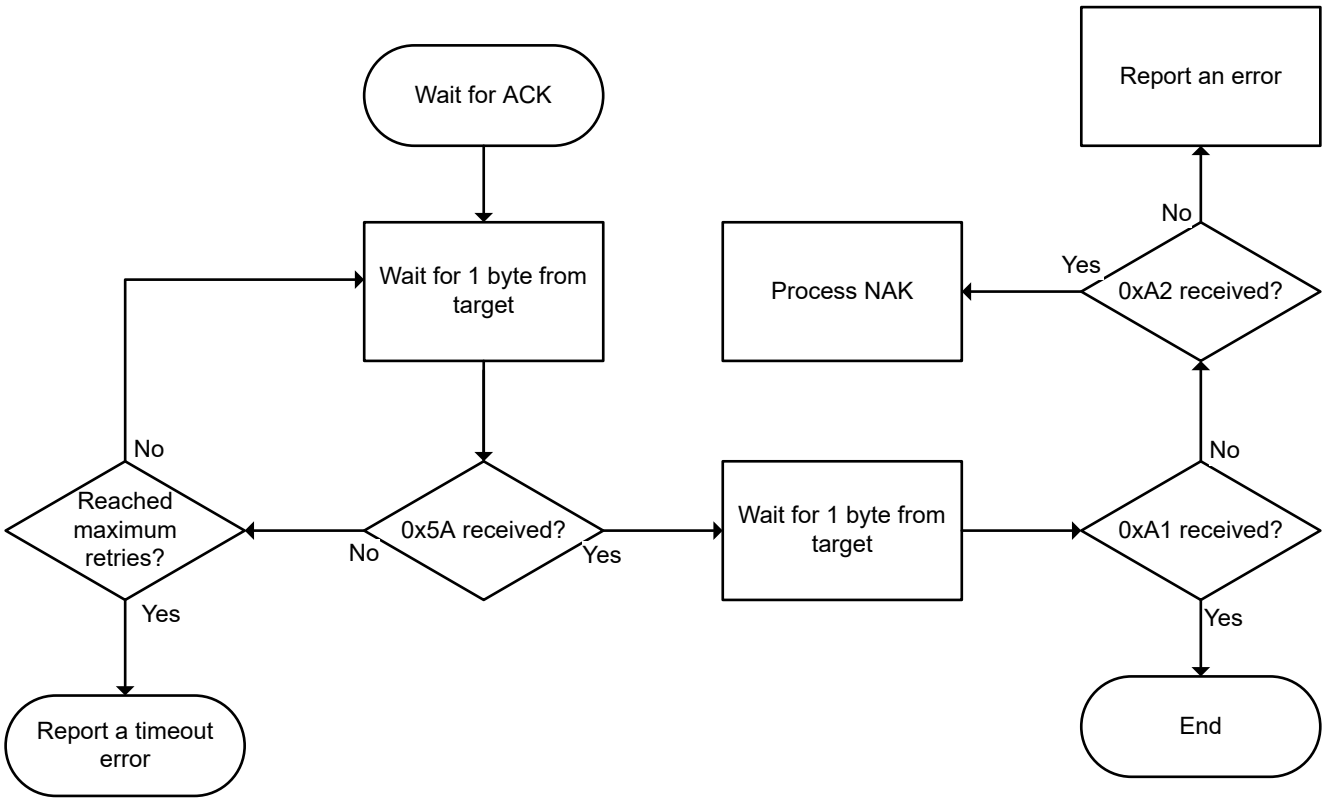


Figure 111. Host reads an ACK from target via UART

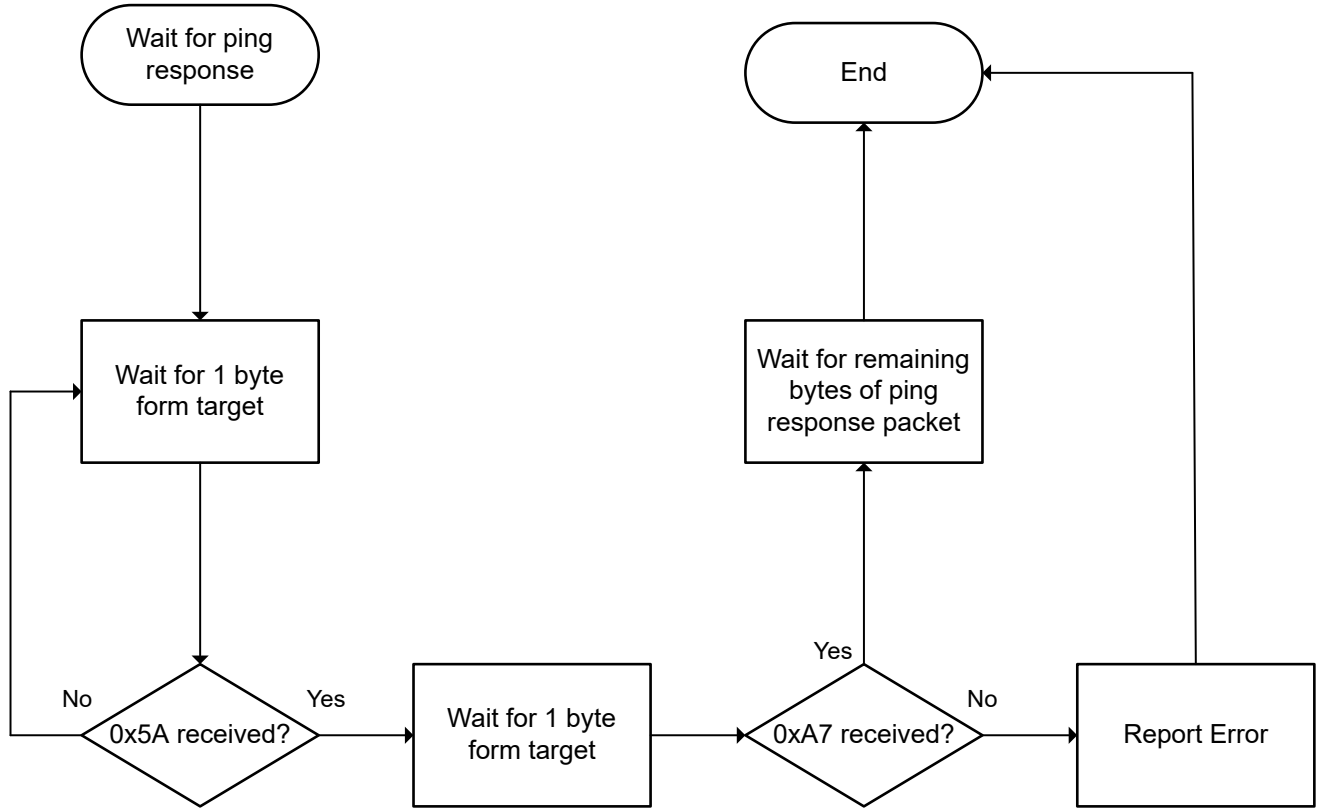


Figure 112. Host reads a ping response from target via UART

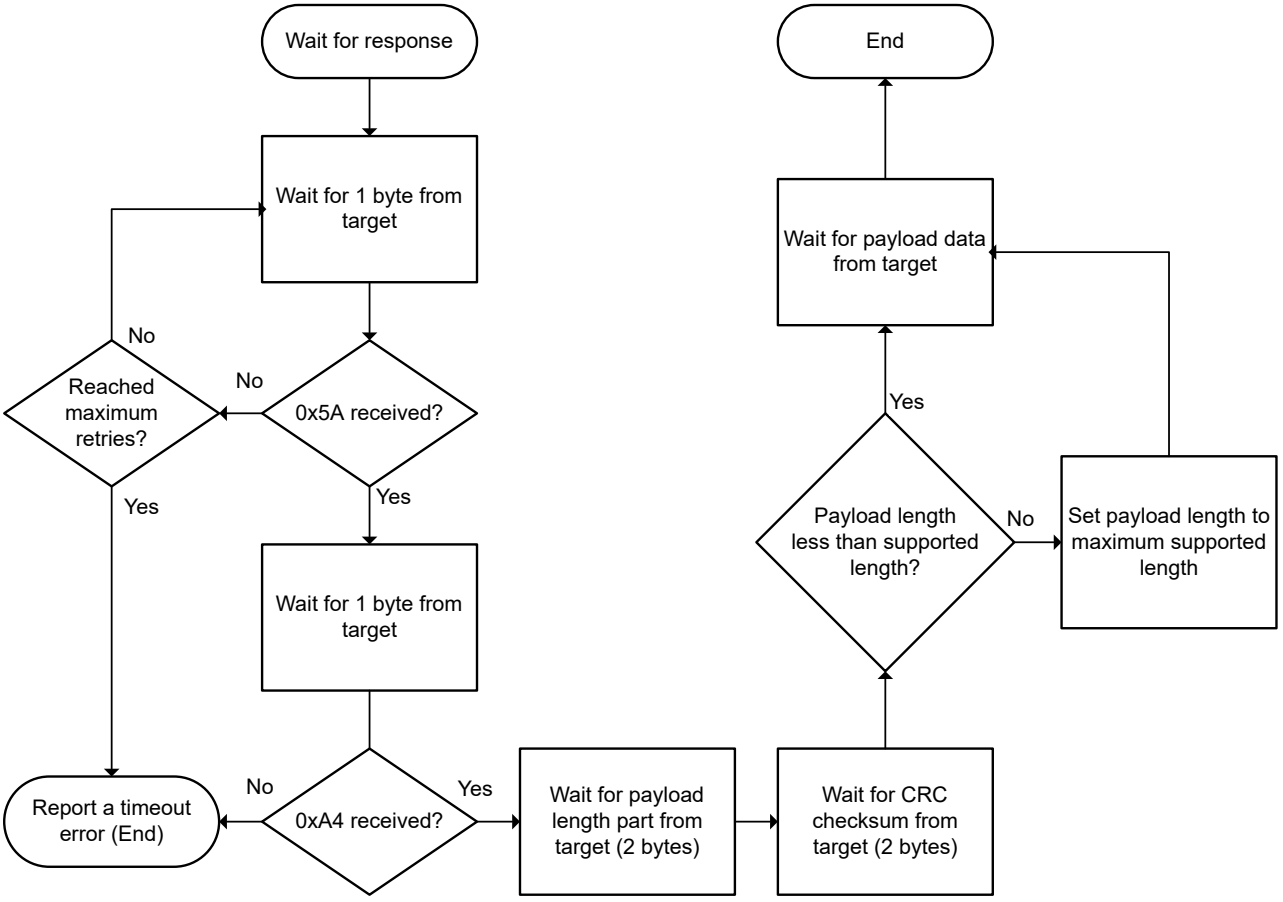


Figure 113. Host reads a command response from target via UART

See Bootloader packet types for more details on UART ISP command format, UART ISP response format, UART ISP data format, and UART ISP commands.

27.8 I2C In-System Programming

27.8.1 Introduction

The bootloader supports loading data into flash via the I^2C peripheral, where the I^2C peripheral serves as the I^2C slave. A 7-bit slave address is used during the transfer. The bootloader uses 0x10 as the I^2C slave address and supports up to 400 kbit/s as the I^2C baud rate.

The maximum supported I^2C baud rate depends on the core clock frequency when the bootloader is running. The typical baud rate is 400 kbit/s with factory settings.

Because the I^2C peripheral serves as an I^2C slave device, each transfer should be started by the host, and each outgoing packet should be fetched by the host.

- An incoming packet is sent by the host with a selected I^2C slave address and the direction bit is set to write.
- An outgoing packet is read by the host with a selected I^2C slave address and the direction bit is set as read.
- 0x00 is sent as the response to host if the target is busy with processing or preparing data.

The following charts show the communication flow of the host reading the ping and ACK packets, and the corresponding responses from the target.

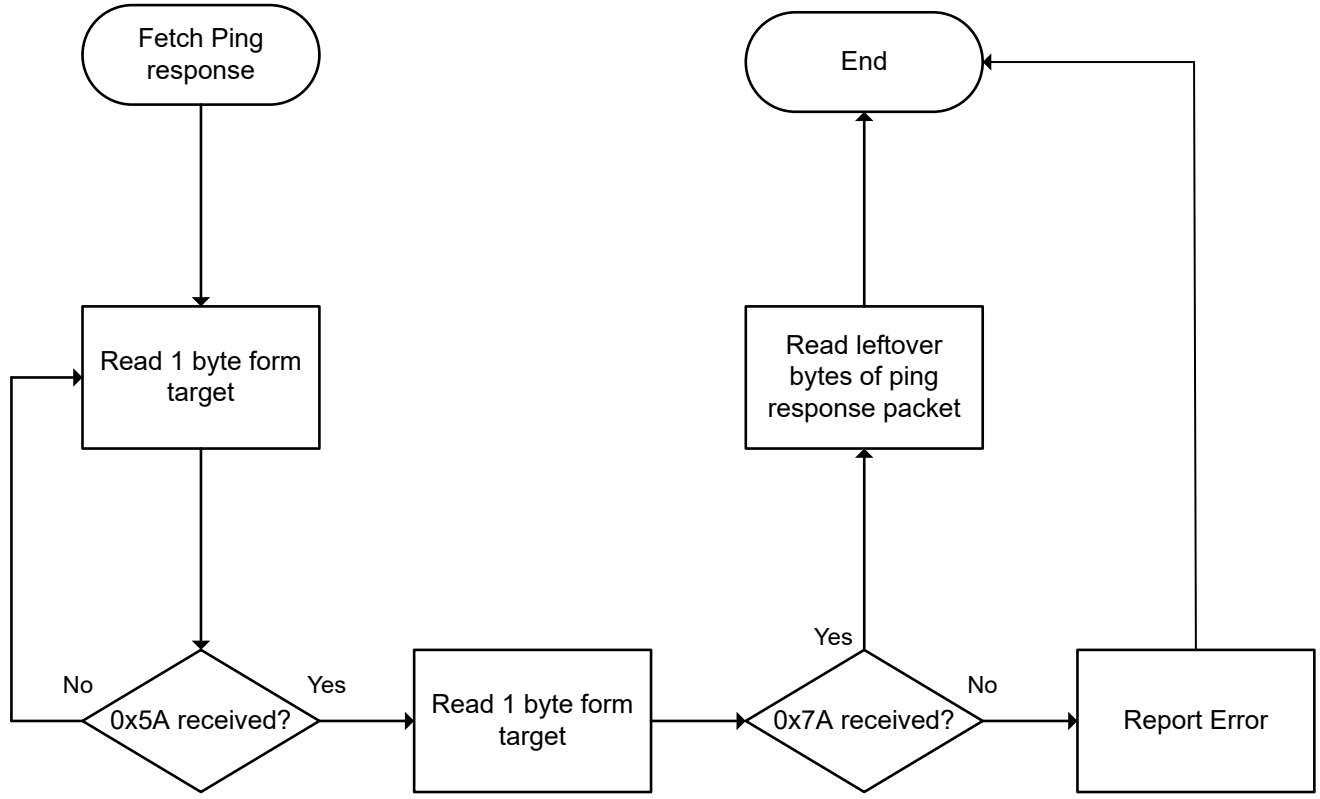


Figure 114. Host reads ping response from target via I²C

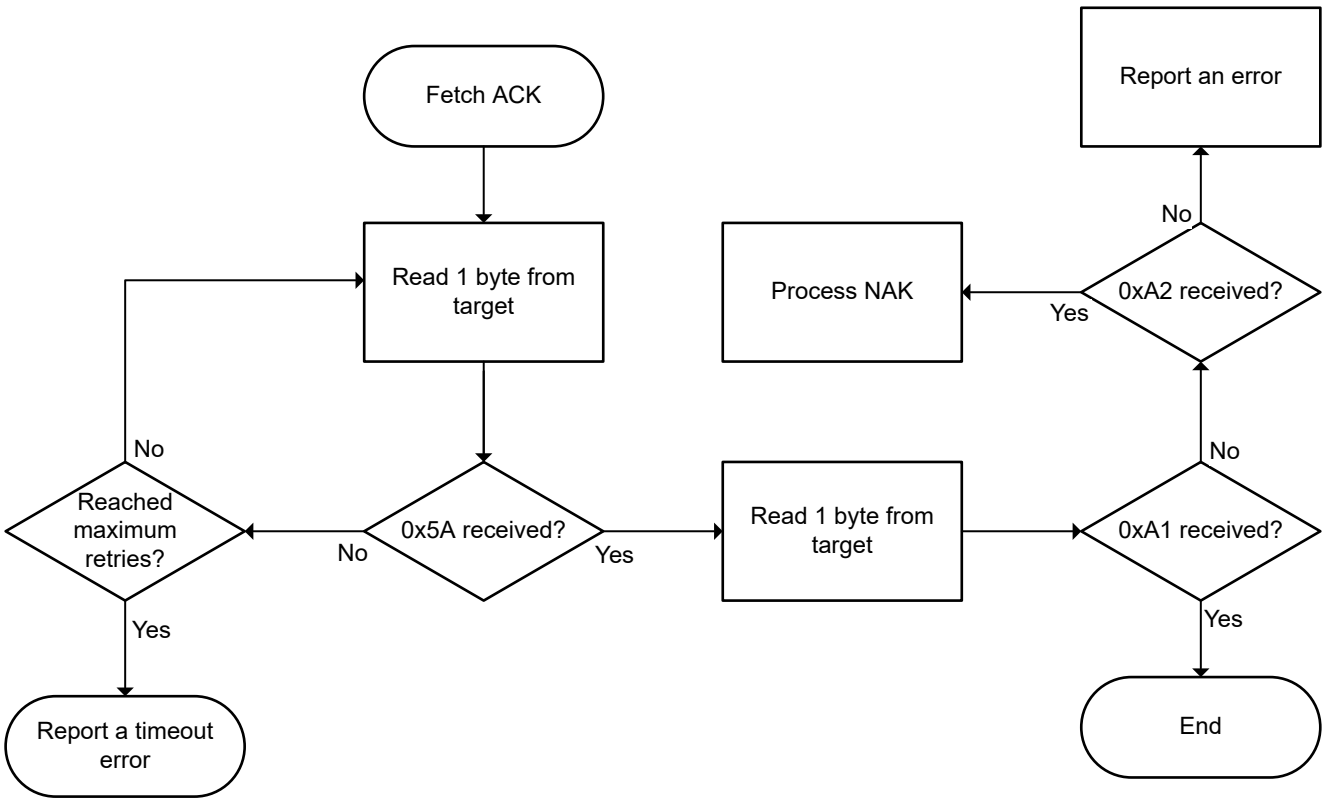


Figure 115. Host reads ACK packet from target via I²C

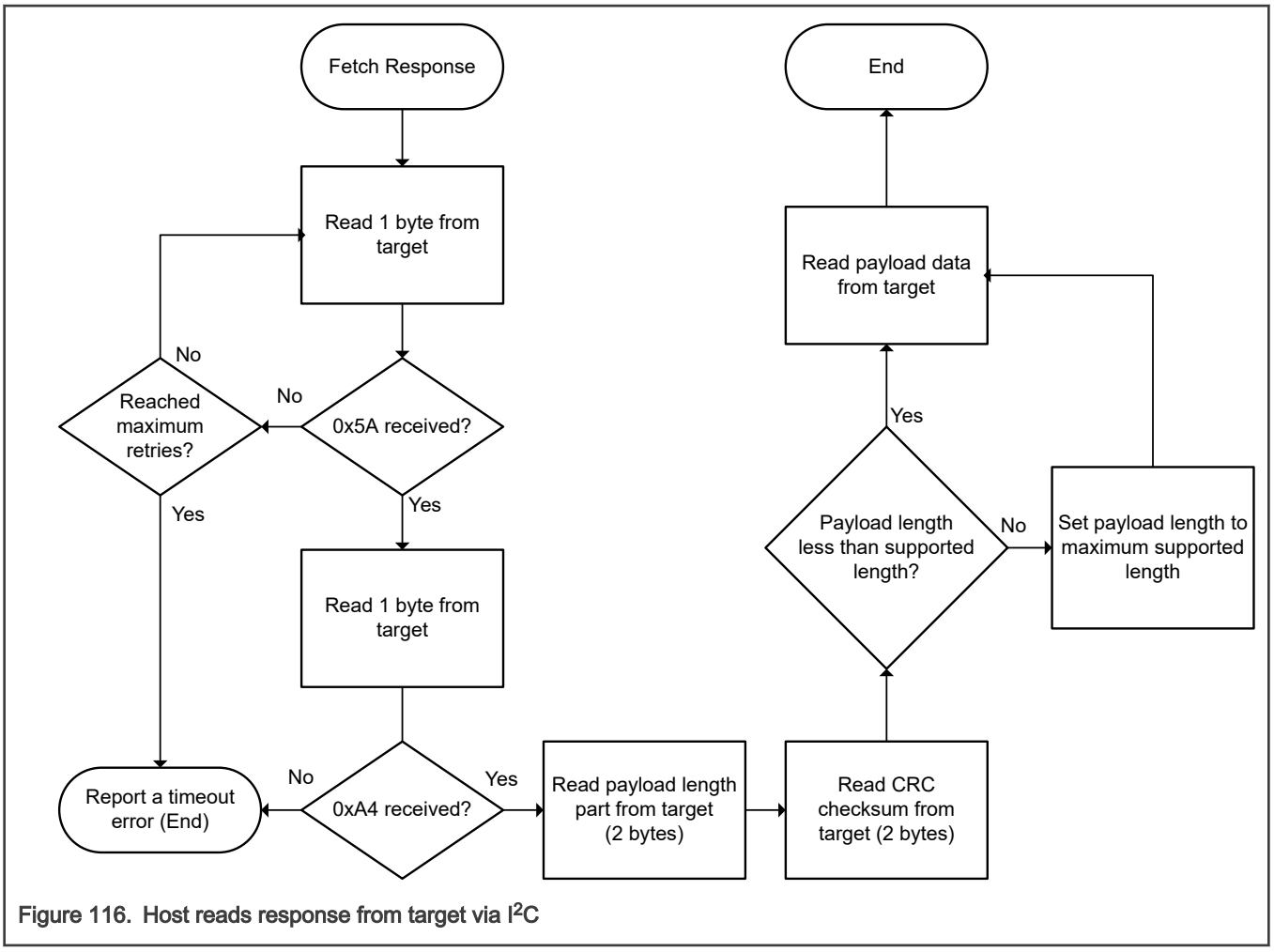


Figure 116. Host reads response from target via I²C

See Bootloader packet types for more details on I²C ISP command format, I²C ISP response format, and I²C ISP data format.

See The bootloader command set for more details on I²C ISP commands.

27.9 SPI In-System programming

27.9.1 Introduction

The bootloader supports loading data into flash via the SPI peripheral, where the SPI peripheral serves as an SPI slave. The SPI transfer should be SPI Mode 3 with 8 data bits.

The maximum supported baud rate of the SPI depends on the core clock frequency when the bootloader is running. The typical baud rate is 2000 kbit/s with the factory settings. The actual baud rate is lower or higher than 2000 kbit/s, depending on the actual value of the core clock.

Because the SPI peripheral serves as an SPI slave device, each transfer should be started by the host, and each outgoing packet should be fetched by the host.

- The transfer on SPI is slightly different from I²C:
- The host receives 1 byte after it sends out any byte.
- Received bytes should be ignored when the host is sending out bytes to the target
- The host starts reading bytes by sending 0x00s to target

The byte 0x00 is sent as a response to host if the target is under the following conditions:

- Processing incoming packet.
- Preparing outgoing data.
- Received invalid data.

The bootloader also supports the active notification pin (nIRQ pin) to notify the host processor it is busy or ready for new commands/data. See below figure for the typical physical connection between the host and the bootloader device.

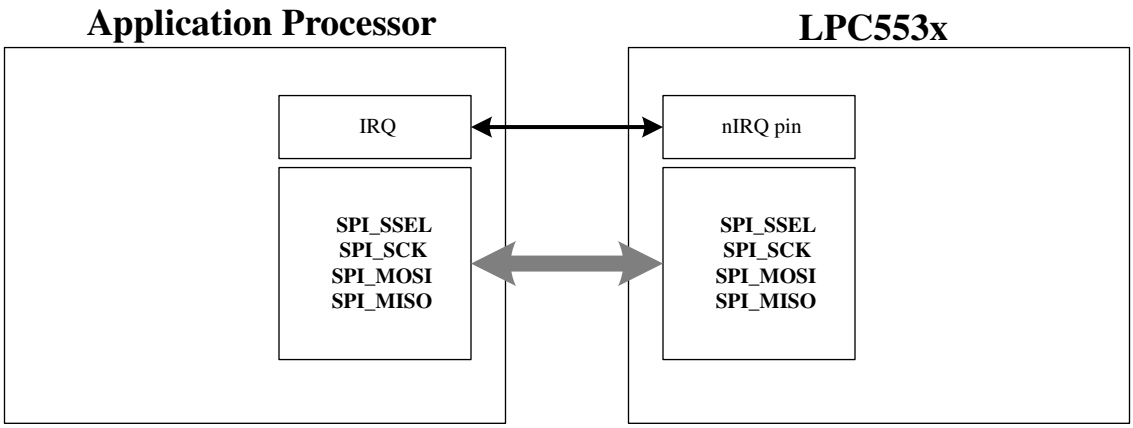


Figure 117. Physical interface for SPI ISP

The following flowcharts show how the host reads a ping response, an ACK and a command response from target via SPI without the nIRQ pin enabled.

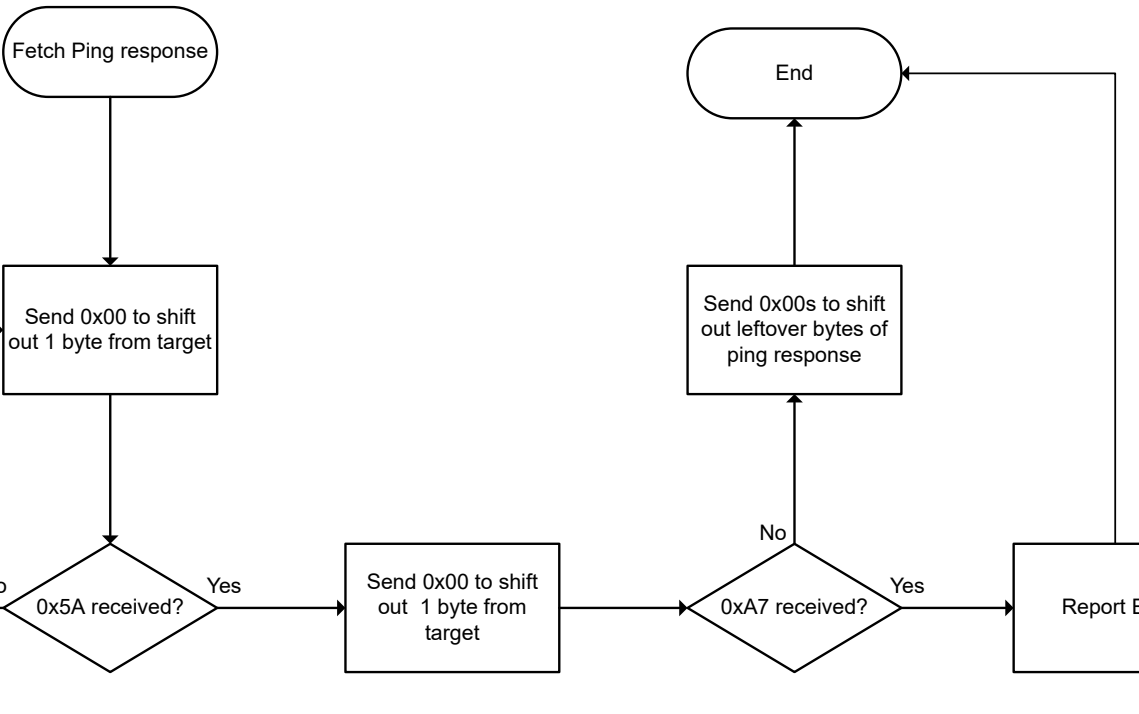


Figure 118. Host reads ping packet from target via SPI

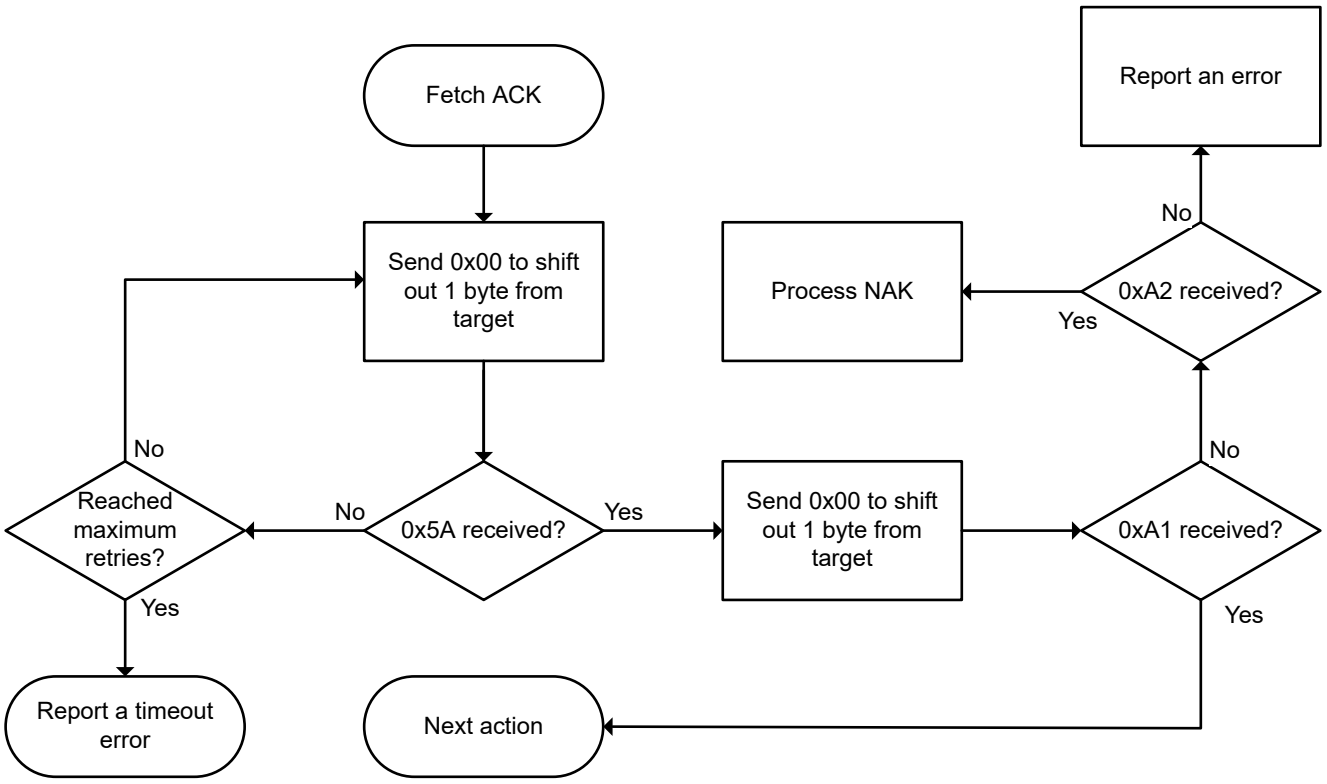


Figure 119. Host reads ACK from target via SPI

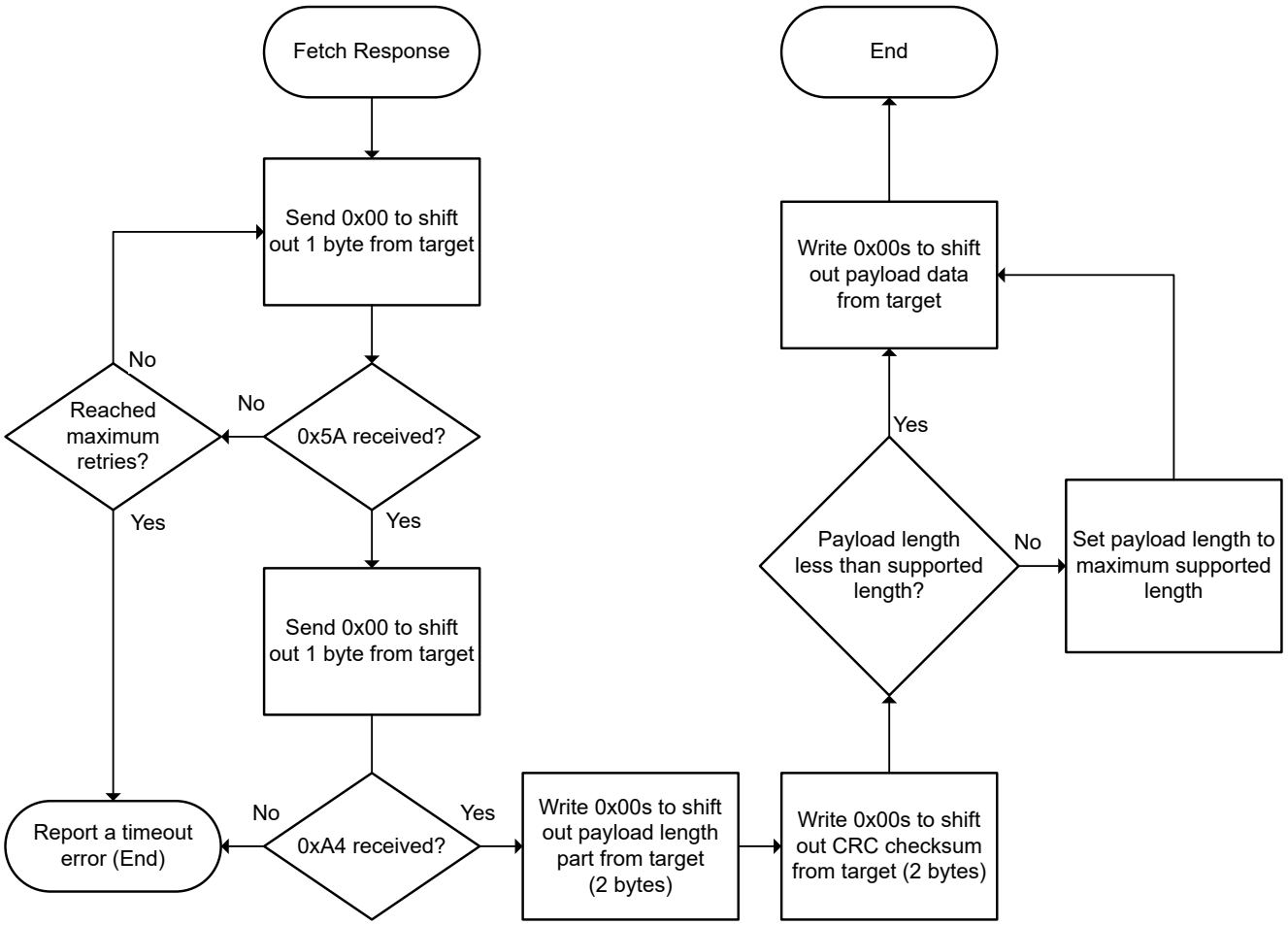


Figure 120. Host reads response from target via SPI

To accelerate the SPI transfer between the host and the bootloader, the bootloader provides an active notification pin known as the nIRQ pin, it can be enabled by the SetProperty command. Once being enabled, the host needs to wait until it sees a negative edge on the nIRQ pin before reading any data from the bootloader, and it needs to wait until the nIRQ pin is high before sending any data to the bootloader.

See Bootloader packet types for more details on SPI ISP command format, SPI ISP response format, and SPI ISP data format.

See The bootloader command set for more details on SPI ISP commands.

27.9.2 Host read ACK(5a,a1) packet timing restriction

If IRQ pin is not used for the In-System Programming protocol, the host needs to follow the below data fetching timing for ACK from the device as the below figures shows.

- After sending out the command, the host should wait for at least 100 us before polling the start byte 0x5A of the ACK packet sent out by the device; If fetching data is not 0x5A, delay 100us, and do polling to reread 0x5A.
- For the ACK packet 0x5A, 0xA1, after the host receives the 0x5A, a 50us delay must add before fetching the 0xA1 from the device.

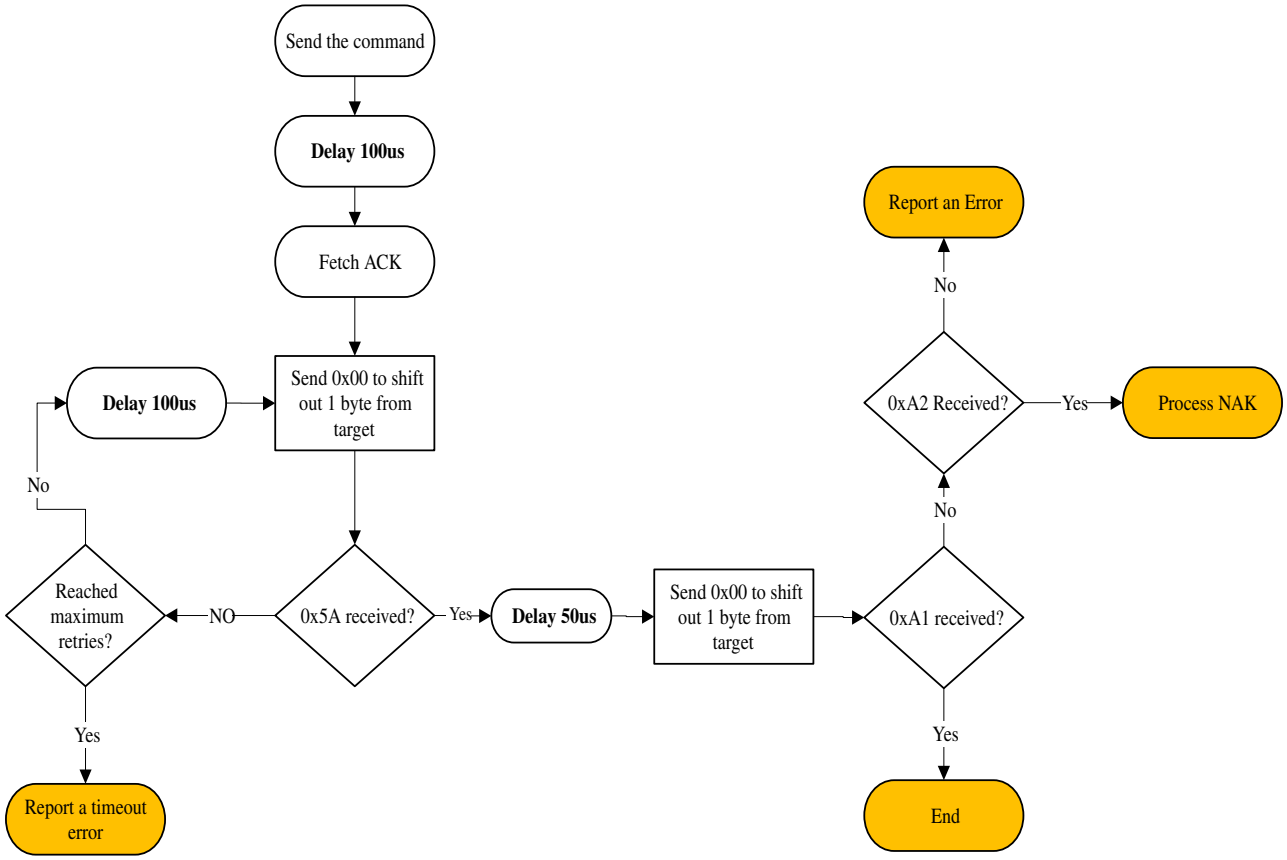


Figure 121. Host read ACK(5a,a1) packet timing restriction flow

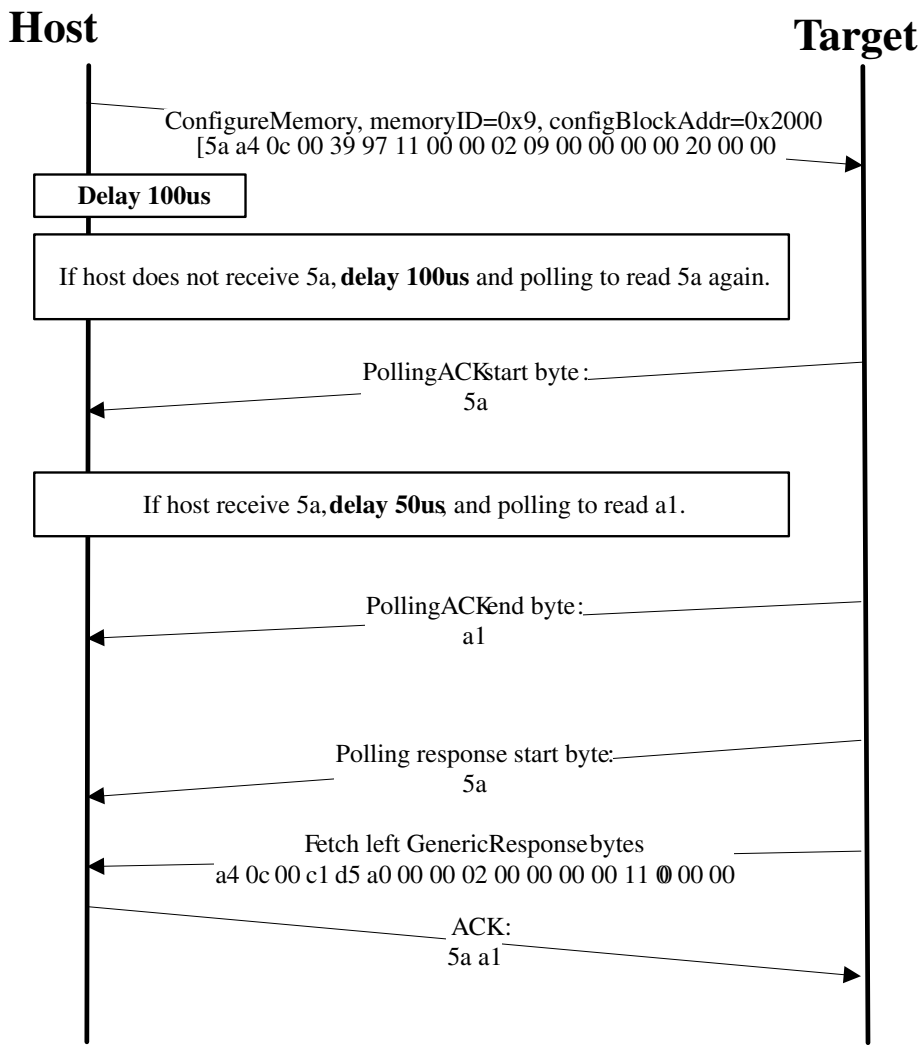


Figure 122. Host read ACK(5a,a1) packet timing restriction

27.10 USB In-System Programming

27.10.1 Introduction

The bootloader supports In-System Programming using the USB peripheral. The target is implemented as USB-HID device classes.

When transfer data through USB-HID device class, USB-HID does not use framing packets. Instead, the packetization, inherent in the USB protocol itself is used. The ability for the device to NAK Out transfers (until they can be received) provides the required flow control. The built-in CRC of each USB packet provides the required error detection

27.10.1.1 Device descriptor

The bootloader configures the default USB VID/PID/Strings as below:

Default VID/PID:

- VID = 0x1FC9.

- PID = 0x0025.

Default Strings:

- Manufacturer [1] = "NXP SEMICONDUCTOR INC".
- Product [2] = "USB COMPOSITE DEVICE".

The USB VID, PID, and Strings can be customized using the CMPA of the flash. For example, the USB VID and PID can be customized by writing the new VID to the usbVid field and the new PID to the usbPid field of the CMPA in flash.

27.10.1.2 Endpoints

The HID peripheral uses three endpoints:

- Control (0).
- Interrupt IN (1).
- Interrupt OUT (2).

The Interrupt OUT endpoint is optional for HID class devices, but the MCU bootloader uses it as a pipe, where the firmware can NAK send requests from the USB host.

27.10.1.3 HID Reports

There are four HID reports defined and used by the bootloader USB HID peripheral. The report ID determines the direction and type of packet sent in the report; otherwise, the contents of all reports are the same.

Table 189. HID reports assigned for the bootloader

Report ID	Packet Type	Direction
1	Command	OUT
2	Data	OUT
3	Command	IN
4	Data	IN

Each report has a maximum size of 60 bytes. The maximum payload size is 56 bytes. In addition, there is a 4-byte report header that indicates the length (in bytes) of the payload and report id sent the packet.

NOTE

In the future, the maximum report size may be increased, to support transfers of larger packets. Alternatively, additional reports may be added with larger maximum sizes.

The actual data sent in all of the reports looks like:

Table 190. Data format sent in USB HID packet

Report ID	Data
0	Report ID
1	Padding
2	Packet Length LSB

Table continues on the next page...

Table 190. Data format sent in USB HID packet (continued)

3	Packet Length MSB
4	Packet[0]
5	Packet[1]
6	Packet[2]
	...
N+4-1	Packet[N-1]

This data includes the Report ID, which is required if more than one report is defined in the HID report descriptor. The actual data sent and received has a maximum length of 35 bytes. The Packet Length header is written in little-endian format, and it is set to the size

(in bytes) of the packet sent in the report. This size does not include the Report ID or the Packet Length header itself. During a data phase, a packet size of 0 indicates a data phase abort request from the receiver.

See Command packet for more details on USB ISP command format, and USB ISP response format.

See Response packet for more details on USB ISP data format.

See The bootloader command set for more details on USB ISP commands.

27.11 CAN ISP

The Bootloader supports loading data into flash via the CAN peripheral. It supports four predefined auto detect speeds on CAN transferring:

• 125 KHz • 250 KHz • 500 KHz • 1 MHz

The current CAN IP can support up to 1 MHz speed, so the default speed is set to 1MHz.

In host applications, the user can specify the speed for CAN by providing the speed index as the following in the (19:16, word 2 in CMPA) in CMPA, which represents different speeds.

- 0000 - Auto baud detection (125K, 250K, 500K, 1M).
- 0001: 10 kbps
- 0010: 20 kbps
- 0011: 50 kbps
- 0100: 125 kbps
- 0101: 250 kbps
- 0110: 500 kbps
- 0111: 800 kbps
- 1000: 1,000 kbps

In bootloader, this supports the auto speed detection feature within supported speeds. In the beginning, the bootloader enters the listen mode with the initial speed (default speed 1 MHz). Once the host starts sending a ping to a specific node, it generates traffic on the CAN bus. Because the bootloader is in a listen mode. It can check if the local node speed is correct by detecting errors. If there is an error, some traffic will be visible, but it may not be on the right speed to see the real data. If this happens, the speed setting changes and checks for errors again. No error means the speed is correct. The settings change back to the normal receiving mode to see if there is a package for this node. It then stays in this speed until another host is using another speed and try to communicate with any node. It repeats the process to detect a right speed before sending host timeout and aborting the request.

The host side should have a reasonable time tolerance during the auto speed detection period. If it sends as timeout, it means there is no response from the specific node, or there is a real error and it needs to report the error to the application.

This flow chart demonstrates the communication flow for how the host reads the ping packet, ACK, and response from the target.

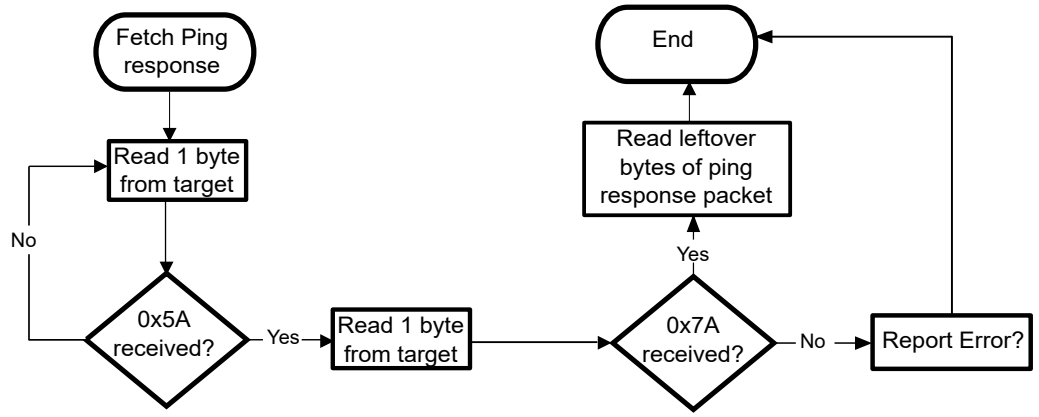


Figure 123. Host reads ping response from target via FCAN

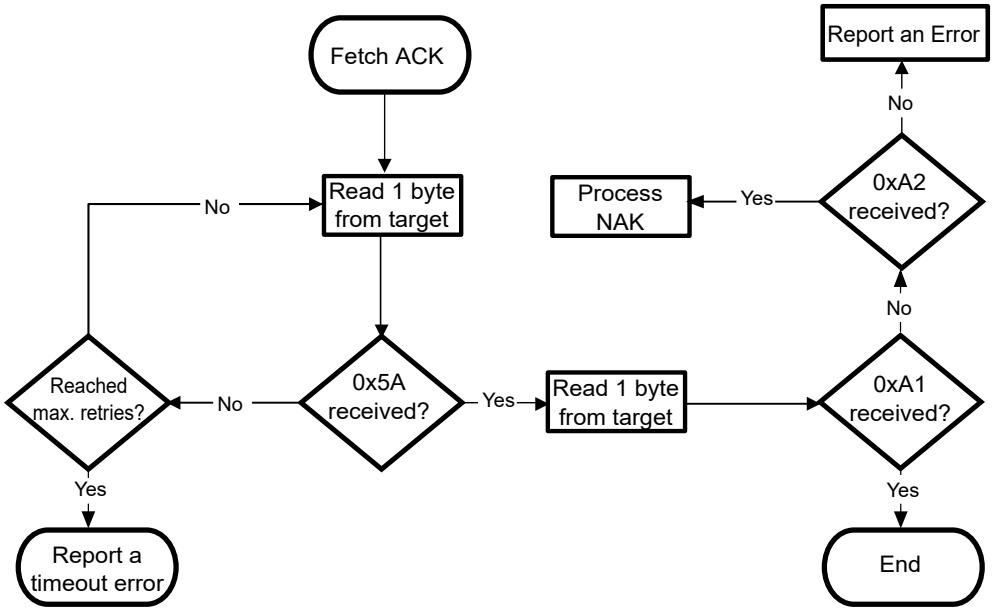


Figure 124. Host reads ACK packet from target via CAN

27.12 In-Application-Programming

See Flash API for details.

Chapter 28
ROM API

28.1 Overview

This sections provides an introduction to the ROM API in this device.

28.1.1 Features

- The ROM API supports programming both the Programmable FLASH region (store application code and data) and the FLASH Firewall Region (store device configurations, boot configuration, in-field programmable data).
- The ROM API enables the Serial NOR FLASH programming via the FlexSPI NOR API.
- The ROM API supports OTP-eFuse read and programming.
-
- The ROM API supports In-application-programming functionality by the IAP API, via the unified memory interface.

28.2 Functional description

This section introduces the ROM API structure first, then describes the FLASH API, OTP API, FLEXSPI NOR API, IAP API. This section also demonstrates the typical usage of each API.

- The primary purpose of the FLASH API is to update the programmable FLASH area and specify the parameters in the CMPA and the CFPA.
- The FLEXSPI FLASH API eases the support of various Serial NOR devices in the market.
- OTP API provides the functionality to advance lifecycle; program/read critical one-time programmable parameters and the general-purpose one-time programmable FUSE field.
- IAP API enables the flexibility of doing in-application-programming by the memory interface.

28.2.1 ROM API structure

The ROM API table locates at address 0x0302fc00. See the following figure.

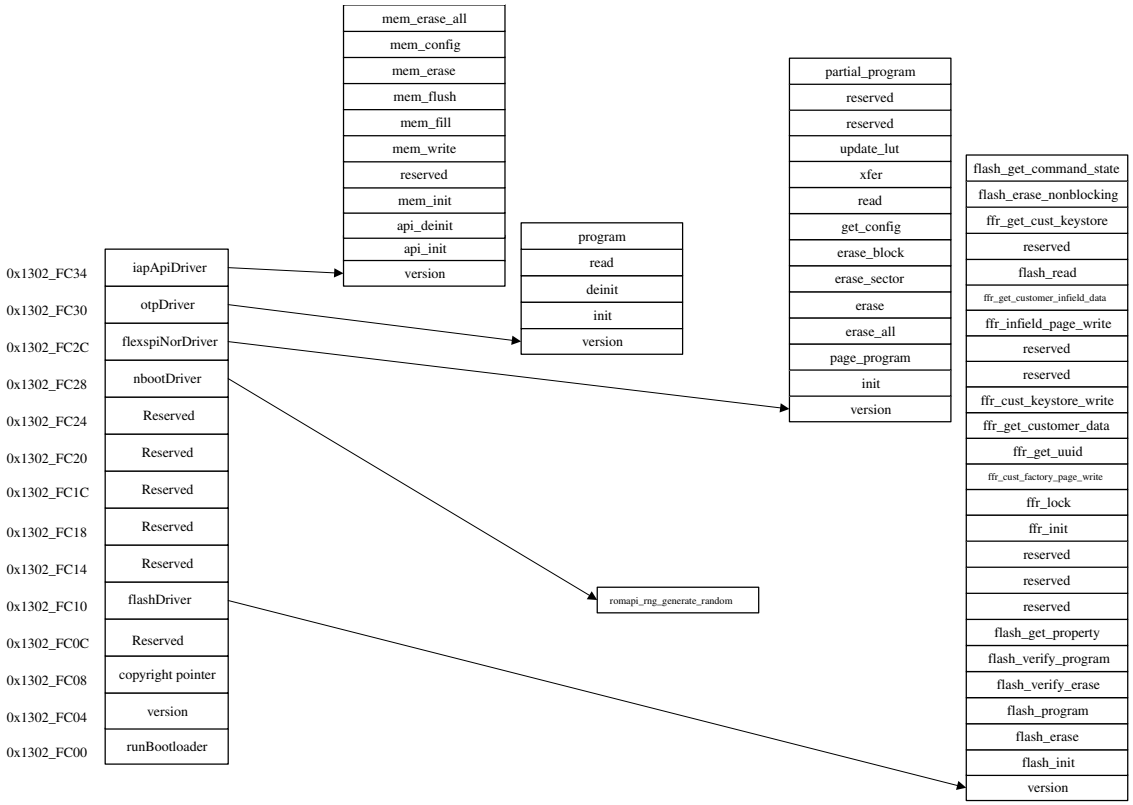


Figure 125. ROM API structure

28.2.2 FLASH APIs

The FLASH API set enables the following features:

- Initialize FLASH controller
- Erase and verify the specified FLASH area
- Program and verify the specified FLASH page
- Retrieve FLASH properties
- Initialize or Lock the FFR
- Program and Read CMPA
- Program and Read CFPA
- Non-blocking FLASH Erase/Status Check API for timing-critical use cases

The FLASH APIs are organized in the FLASH Driver API Interface structure. See the API layout and API prototypes from the following data structure. The whole FLASH API wrapper is available in the SDK release package.

The bootloader API prototypes are:

```
typedef struct FLASHDriverInterface
{
```

```
standard_version_t version; //!< flash driver API version number.
// FLASH driver
status_t (*flash_init)(flash_config_t *config);
status_t (*flash_erase)(flash_config_t *config, uint32_t start, uint32_t lengthInBytes, uint32_t key);
status_t (*flash_program)(flash_config_t *config, uint32_t start, uint8_t *src,
uint32_t lengthInBytes);
status_t (*flash_verify_erase)(flash_config_t *config, uint32_t start, uint32_t lengthInBytes);
status_t (*flash_verify_program)(flash_config_t *config,
uint32_t start,
uint32_t lengthInBytes,
const uint8_t *expectedData,
uint32_t *failedAddress,
uint32_t *failedData);
status_t (*flash_get_property)(flash_config_t *config, flash_property_tag_t whichProperty,
uint32_t *value);
uint32_t reserved0[3];
// FLASH FFR driver
status_t (*ffr_init)(flash_config_t *config);
status_t (*ffr_lock)(flash_config_t *config);
status_t (*ffr_cust_factory_page_write)(flash_config_t *config, uint8_t *page_data, bool seal_part);
status_t (*ffr_get_uuid)(flash_config_t *config, uint8_t *uuid);
status_t (*ffr_get_customer_data)(flash_config_t *config, uint8_t *pData, uint32_t offset,
uint32_t len);
status_t (*ffr_cust_keystore_write)(flash_config_t *config, ffr_key_store_t *pKeyStore);
status_t reserved1;
status_t reserved2;
status_t (*ffr_infield_page_write)(flash_config_t *config, uint8_t *page_data, uint32_t valid_len);
status_t (*ffr_get_customer_infield_data)(flash_config_t *config, uint8_t *pData, uint32_t offset,
uint32_t len);
status_t (*flash_read)(flash_config_t *config, uint32_t start, uint8_t *dest, uint32_t lengthInBytes);
status_t reserved3;
status_t (*flash_get_cust_keystore)(flash_config_t *config, uint8_t *pData, uint32_t offset,
uint32_t len);
status_t (*flash_erase_non_blocking)(flash_config_t *config, uint32_t start, uint32_t lengthInBytes,
uint32_t key);
status_t (*flash_get_command_state)(flash_config_t *config);
} flash_driver_interface_t;
```

Each FLASH API depends on a common FLASH context structure named *flash_config_t* to perform the proper FLASH operation. See the structure details below.

```
/*! @brief FLASH driver state information.
*
* An instance of this structure is allocated by the user of the flash driver and
* passed into each of the driver APIs.
*/
typedef struct
{
uint32_t PFlashBlockBase;  /*!< A base address of the first PFlash block */
uint32_t PFlashTotalSize;  /*!< The size of the combined PFlash block. */
uint32_t PFlashBlockCount; /*!< A number of PFlash blocks. */
uint32_t PFlashPageSize;   /*!< The size in bytes of a page of PFlash. */
uint32_t PFlashSectorSize; /*!< The size in bytes of a sector of PFlash. */
flash_ffr_config_t ffrConfig;
flash_mode_config_t modeConfig;
} flash_config_t;
```

The *flash_mode_config_t* is defined here:

```
/*! @brief FLASH controller paramter config. */
typedef struct
{
uint32_t sysFreqInMHz;
// ReadSingleWord parameter
struct
{
uint8_t readWithEccOff : 1;
uint8_t readMarginLevel : 2;
uint8_t readDmaccWord : 1;
uint8_t reserved0 : 4;
uint8_t reserved1[3];
} readSingleWord;
// SetWriteMode parameter
struct
{
uint8_t programRampControl;
uint8_t eraseRampControl;
uint8_t reserved[2];
} setWriteMode;
// SetReadMode parameter
struct
{
uint16_t readInterfaceTimingTrim;
uint16_t readControllerTimingTrim;
uint8_t readWaitStates;
uint8_t reserved[3];
} setReadMode;
} flash_mode_config_t;
```

This field is auto-managed by the FLASH_Init API call, the user application doesn't need to touch it.

The flash_ffr_config_t is defined as below:

```
/*! @brief FLASH controller paramter config. */
typedef struct
{
uint32_t ffrBlockBase;
uint32_t ffrTotalSize;
uint32_t ffrPageSize;
uint32_t cfpaPageVersion;
uint32_t cfpaPageOffset;
} flash_ffr_config_t;
```

This field is managed by the FFR_Init API call, the user application doesn't need to touch it.

28.2.2.1 Version

The "version" field in the FLASH API table indicates the current FLASH API version in the ROM bootloader.

Prototype :

```
standard_version_t version;
```

Table 191. Definition of the version field

Parameter	Description

standard_version_t	Pointer to version structure to store current FLASH driver version information
	uint8_t bugfix; //!< bugfix version [7:0]
	uint8_t minor; //!< minor version [15:8]
	uint8_t major; //!< major version [23:16]
	char name; //!< name [31:24]

Example:

```
#define ROM_API_TREE ((uint32_t *)0x1302FC00U)
#define FLASH_API_TREE ((flash_driver_interface_t*)ROM_API_TREE[4])
uint32_t FlashDriverVersion = FLASH_API_TREE-> version;
```

In this SoC, the FLASH Driver version is "0x46010100", it means the FLASH driver version is 1.1.0.

28.2.2.2 flash_init

This API initializes the FLASH default parameters and related FLASH clock for the FLASH and FMC. The *flash_init* API should be called before all the other FLASH APIs.

Prototype :

```
status_t (*flash_init)(flash_config_t *config);
```

Table 192. Parameters used for flash_init API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.

Example :

```
flash_config_t flashConfig;
uint32_t status = FLASH_API_TREE->flash_init (&flashConfig);
```

See the possible status codes in Table 210. If the status is kStatus_FLASH_Success return value, it means flash has been inited successfully.

NOTE

API FFR_Init API must be called right after Flash_Init before calling any other API which exercises Flash.

28.2.2.3 flash_erase

This API erases the internal FLASH region specified by the parameters. This API must be called after the flash_init.

Prototype :

```
status_t (*flash_erase)(flash_config_t *config, uint32_t start, uint32_t lengthInBytes, uint32_t key);
```

Table 193. Parameters used in flash_erase API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
start	The start address of the required flash memory to be erased. The start address must be page-aligned (that is, a multiple of 512).
lengthInBytes	The length, given in bytes (not words or long words) to be erased. Must be page-aligned.
key	Key is used to validate erase operation. Must be set to “kFLASH_ApiEraseKey”. kFLASH_ApiEraseKey = (((‘k’) << 24) | ((‘e’) << 16) | ((‘f’) << 8) | ((‘l’)))

Example :

```
uint32_t start = 0x0 ;
uint32_t lengthInBytes = 0x1000 ;
#define ERASE_KEY 0x6b65666b
uint32_t status = FLASH_API_TREE->flash_erase (&flashConfig, start, lengthInBytes, ERASE_KEY);
```

See the possible status codes in Table 210. If the status is kStatus_FLASH_Success return value, it means the FLASH region 0x0 – 0x1000 has been erased.

28.2.2.4 flash_program

This API programs the data into the internal flash page specified by input parameters. The required "start" and "lengthInBytes" parameters must be page-size aligned. This API must be called after the flash_init.

Prototype :

```
status_t (*flash_program)(flash_config_t *config, uint32_t start, uint8_t *src, uint32_t
lengthInBytes);
```

Table 194. Parameters used in flash_program API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
start	The start address of the required flash memory to be programmed. The start address must be 512bytes-aligned.
src	Pointer to the source buffer of data that is to be programmed into flash.
lengthInBytes	The length in bytes (not words or long words) to be erased; the length must also be 512bytes-aligned.

Example:

```
uint8_t programBuffer[512];
for (uint32_t i=0; i<sizeof(programBuffer); i++)
{
programBuffer[i] = (uint8_t)(i & 0xFF);
}
status = FLASH_API_TREE->flash_program(&flashConfig, 0x0, programBuffer,
sizeof(programBuffer));
```

See the possible status codes in Table 210. If the status is kStatus_FLASH_Success return value, it means the specified data has been programmed into the FLASH region.

28.2.2.5 flash_verify_erase

This API checks whether the specified FLASH area is in the erasure state.

This API must be called after the flash_init.

Prototype :

```
status_t (*flash_verify_erase)(flash_config_t *config, uint32_t start, uint32_t lengthInBytes);
```

Table 195. Parameters used in flash_verify_erase API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
start	The start address of the required flash memory to be verified. Must be page-aligned.
lengthInBytes	The length, given in bytes (not words or long words) to be verified. Must be page-aligned.

Example:

```
uint32_t start = 0x0 ;
uint32_t lengthInBytes = 0x1000 ;
uint32_t status = FLASH_API_TREE->flash_verify_erase (&flashConfig, start, lengthInBytes);
```

See the possible status codes in Table 210. If the status is kStatus_FLASH_Success return value, it means the internal FLASH region 0x0 – 0x1000 has been erased.

28.2.2.6 flash_verify_program

This API checks whether the data in the specified area is identical to the data supposed to be programmed. This API must be called after the flash_init.

Prototype :

```
status_t (*flash_verify_program)(flash_config_t *config,
uint32_t start,
uint32_t lengthInBytes,
const uint8_t *expectedData,
```

```
uint32_t *failedAddress,
uint32_t *failedData);
```

Table 196. Parameters used in flash_verify_program API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
start	The start address of the required flash memory to be verified.
lengthInBytes	The length, given in bytes (not words or long words) to be verified.
expectedData	Pointer to the expected data that is to be verified against.
failedAddress	Pointer to returned failing address.
failedData	Pointer to return failing data.

Example:

```
uint32_t start = 0x0 ;
uint32_t lengthInBytes = sizeof(expected_buffer) ;
uint32_t failedAddress, failedData;
uint32_t status = FLASH_API_TREE->flash_verify_erase (&flashConfig, start, lengthInBytes, (const
uint8_t *) expected_buffer, & failedAddress, & failedData);
```

See the possible status codes in Table 210. If the status is kStatus_FLASH_Success return value, it means the expected data same as the programed data in the specific FLASH region.

28.2.2.7 flash_get_property

This API returns the requested FLASH property. This API must be called after the flash_init and ffr_init.

Prototype :

```
status_t (*flash_get_property)(flash_config_t *config, flash_property_tag_t whichProperty, uint32_t
*value);
```

Table 197. Parameters used in flash_get_property API

Parameter	Value	Description
config		Pointer to flash_config_t data structure in memory to store driver runtime state.
whichProperty		Pointer to the which flash property need to get.
value		Pointer to the returned flash property value.
Property definition		
kFLASH_PropertyPflashTotalSize	0x01	Pflash total size property.

Table continues on the next page...

Table 197. Parameters used in flash_get_property API (continued)

Parameter	Value	Description
kFLASH_PropertyPflashBlockSize	0x02	Pflash Block size property.
kFLASH_PropertyPflashBlockCount	0x03	Pflash Block Count size property.
kFLASH_PropertyPflashBlockBaseAddr	0x04	flash block base address.
kFLASH_PropertyPflashPageSize	0x30	Pflash page size property.
kFLASH_PropertyFfrTotalSize	0x41	FFR total size property.
kFLASH_PropertyFfrBlockBaseAddr	0x42	FFR block base address property.
kFLASH_PropertyFfrPageSize	0x43	FFR page size property.

Example:

```
flash_property_tag_t whichProperty = kFLASH_PropertyPflashTotalSize;
uint32_t value = 0;
uint32_t status = FLASH_API_TREE->flash_get_property(&flashConfig, whichProperty, &value);
```

See the possible status codes in Table 210. If the status is kStatus_FLASH_Success return value, it means the flash property get successfully, and will return the property value if needed.

28.2.2.8 ffr_init

This API is used for initializing the FFR controller and the *flash_ffr_config* context. It must be called before calling other FFR APIs.

NOTE

flash_init must be called before calling the ffr_init API.

Prototype :

```
status_t (*ffr_init)(flash_config_t *config);
```

Table 198. Parameters used in ffr_init API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.

Example:

```
uint32_t status = FLASH_API_TREE-> ffr_init (&flashConfig);
```

See the possible status codes in Table 210. If the status is kStatus_FLASH_Success return value, it means the FFR parameters has been configured.

NOTE

API FFR_Init API must be called right after Flash_Init before calling any other API which exercises Flash.

28.2.2.9 ffr_lock

This API is used to enable the firewall for all FLASH banks, including enabling FLASH protection for three IFR banks and disabling write access to the FLASHBENKENABLE register. ffr_lock unconditionally locks writing to the CFPA, CMPA, and NMPA flash areas. Subsequent writes are inhibited unless a power-on reset (POR) or brown-out detect (BOD) reset occurs.

Prototype :

```
status_t (*ffr_lock)(flash_config_t *config);
```

Table 199. Parameters used in ffr_lock

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.

Example:

```
uint32_t status = FLASH_API_TREE-> ffr_lock (&flashConfig);
```

See the possible status codes in Table 210. If the status is kStatus_FLASH_Success return value, it means the FFR regions has been locked.

28.2.2.10 ffr_cust_factory_page_write

The API is used for writing the CMPA data into the CMPA region, and the API should be called after the flash_init and ffr_init.

Prototype :

```
status_t (*ffr_cust_factory_page_write)(flash_config_t *config, uint8_t *page_data, bool seal_part);
```

Table 200. Parameters used in ffr_cust_factory_page_write API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
page_data	Pointer to value address that will be written to the destination address.
seal_part	If set as ture or the page_data includes the non-zero CMAC data, the CMPA CMAC will be calculated and programed into the CMPA region.

Example:

```
uint32_t cmpa_buffer [ffr_page_size] = {0, 2, 3, 4};
uint32_t status = FLASH_API_TREE-> ffr_cust_factory_page_write (&flashConfig, (uint8_t *)cmpa_buffer,
false);
```

See the possible status codes in Table 210. If the status is kStatus_FLASH_Success return value, it means the cmpa_buffer data has been programed into the CMPA region.

28.2.2.11 ffr_get_uuid

The API is used for getting the UUID data of the device, and the API should be called after the flash_init and ffr_init.

Prototype :

```
status_t (*ffr_get_uuid)(flash_config_t *config, uint8_t *uuid);
```

Table 201. Parameters used in ffr_get_uuid API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
uuid	Pointer to value address, the value is read back from the nmpa configuration uuid

Example:

```
uint32_t uuid_buffer [4];
uint32_t status = FLASH_API_TREE-> ffr_get_uuid(&flashConfig, (uint8_t *)uuid_buffer);
```

See the possible status codes in Table 210. If the status is kStatus_FLASH_Success return value, it means the uuid data has been got from the UUID of the device.

28.2.2.12 ffr_get_customer_data

This API is used to read data stored in Customer Factory Page, and the API should be called after the flash_init and ffr_init.

Prototype :

```
status_t (*ffr_get_customer_data)(flash_config_t *config, uint8_t *pData, uint32_t offset, uint32_t
len);
```

Table 202. Parameters used in ffr_get_customer_data API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
pData	Point to the destination buffer of date that stores data read from Customer Factory Page.
offset	Point to the offset value based on the CMPA address(0x3e200) of the device.
len	The length in bytes to be read back, and the offset + len <= 512B.

Example:

```
uint32_t cmpa_buffer[4];
uint32_t offset = 0;
uint32_t status = FLASH_API_TREE->ffr_get_customer_data(&flashConfig, (uint8_t *)cmpa_buffer,
offset,      sizeof(cmpa_buffer));
```

See the possible status codes in Table 210. If the status is kStatus_FLASH_Success return value, it means the CMPA data has been got from the CMPA region and stored into the cmpa_buffer.

28.2.2.13 ffr_cust_keystore_write

The API is used for programing the customer key store data into the customer key store region(0x3e400 – 0x3e600), and the API should be called after the flash_init and ffr_init.

Prototype :

```
status_t (*ffr_cust_keystore_write)(flash_config_t *config, ffr_key_store_t *pKeyStore);
```

Table 203. . Parameters used in ffr_cust_keystore_write API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
pKeyStore	Pointer to the customer key store data buffer, which will be programed into the customer key store region.

Example:

```
uint32_t status = FLASH_API_TREE->ffr_cust_keystore_write(&flashConfig, (ffr_key_store_t *)
cust_keystore_buffer);
```

See the possible status codes in Table 210. If the status is kStatus_FLASH_Success return value, it means customer key store data has been programed into the customer key store region.

28.2.2.14 ffr_infield_page_write

The API is used for programing the CFPA data into the CFPA Scratch region(0x3dc00 - 0x3de00) and the API should be called after the flash_init and ffr_init.

Prototype :

```
status_t (*ffr_infield_page_write)(flash_config_t *config, uint8_t *page_data, uint32_t valid_len);
```

Table 204. . Parameters used in ffr_infield_page_write API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
page_data	Pointer to the source buffer of data that is to be programed into the in-field page.
valid_len	The length in bytes to be programmed, the length must equal the page size.

Example:

```
uint32_t cfpa_buffer [128];
uint32_t status = FLASH_API_TREE-> ffr_infield_page_write (&flashConfig, (uint8_t *) cfpa_buffer,
sizeof(cfpa_buffer));
```

See the possible status codes in Table 210. If the status is kStatus_FLASH_Success return value, it means the CFPA data has been programed into the CFPA Scratch region.

28.2.2.15 ffr_get_customer_infield_data

The API is used for getting the CFPA data from the FFR CFPA region, and the API should be called after the flash_init and ffr_init.

Prototype :

```
status_t (*ffr_get_customer_infield_data)(flash_config_t *config, uint8_t *pData, uint32_t offset,
uint32_t len);
```

Table 205. . Parameters used in ffr_get_customer_infield_data API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
pData	Point to the destination buffer of data that stores data read from Customer In-field page.
offset	Pointer to the offset value based on the CFPA address(0x3dc00) of the device.
len	Point to the length of data to read, and the offset + len <= 512B.

Example:

```
uint32_t cfpa_buffer[4];
uint32_t offset = 0;
uint32_t status = FLASH_API_TREE->ffr_get_customer_infield_data (&flashConfig, (uint8_t *)
cfpa_buffer, offset, sizeof(cfpa_buffer ));
```

See the possible status codes in Table 210. If the status is kStatus_FLASH_Success return value, it means the CFPA data has been got from the CFPA region and stored into the cfpa_buffer.

28.2.2.16 flash_read

The API is used for getting internal flash data, including the FLASH and FFR data, , and the API should be called after the flash_init.

Prototype :

```
status_t (*flash_read)(flash_config_t *config, uint32_t start, uint8_t *dest, uint32_t lengthInBytes);
```

Table 206. Parameters used in flash_read API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
start	Point to the start address where will be read.
dest	Pointer to the buffer used for storing the read data.
lengthInBytes	Point to the read data length

Example:

```
uint32_t start_addr = 0x1000;
```

```
uint8_t read_buffer[512] = {0};
uint32_t length = 512;
uint32_t status = FLASH_API_TREE-> flash_read(&flashConfig, start_addr, (uint8_t *) read_buffer,
length);
```

See the possible status codes in Table 210. If the status is kStatus_FLASH_Success return value, it means the expected read region has been loaded into the read_buffer.

28.2.2.17 flash_get_cust_keystore

The API is used for getting the customer key store data from the customer key store region(0x3e400 – 0x3e600), and the API should be called after the flash_init and ffr_init.

Prototype:

```
status_t (*flash_get_cust_keystore)(flash_config_t *config, uint8_t *pData, uint32_t offset, uint32_t
len);
```

Table 207. Parameters used in flash_get_cust_keystore API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
pData	Pointer to the customer key store data buffer, which got from the customer key store region.
offset	Point to the offset value based on the customer key store address(0x3e400) of the device.
len	Point to the length of the expected get customer key store data, and the offset + len <= 512B.

Example:

```
uint8_t cust_keystore_buffer[512] = {0};
uint32_t offset = 0;
uint32_t length = 512;
uint32_t status = FLASH_API_TREE-> flash_get_cust_keystore (&flashConfig, (uint8_t *)
cust_keystore_buffer, offset, length);
```

See the possible status codes in Table 210. If the status is kStatus_FLASH_Success return value, it means the customer key store data has been got from the customer key store region and stored into the cust_keystore_buffer.

28.2.2.18 flash_erase_non_blocking

The API is used for erasing the FLASH region with the non-blocking feature, the function should be call with the following the API flash_get_command_state, and the API should be called after the flash_init and ffr_init.

Prototype :

```
status_t (*flash_erase_non_blocking)(flash_config_t *config, uint32_t start, uint32_t lengthInBytes,
uint32_t key);
```

Table 208. Parameters used in flash_erase_non_blocking API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.
start	The start address of the required flash memory to be erased. The start address must be page-aligned (that is, a multiple of 512).
lengthInBytes	The length, given in bytes (not words or long words) to be erased. Must be page-aligned.
key	Key is used to validate erase operation. Must be set to “kFLASH_ApiEraseKey”. kFLASH_ApiEraseKey = (((‘k’) << 24) | ((‘e’) << 16) | ((‘f’) << 8) | ((‘l’)))

Example:

```
uint32_t start = 0x0 ;
uint32_t lengthInBytes = 0x1000 ;
#define ERASE_KEY 0x6b65666b
uint32_t status = FLASH_API_TREE-> flash_erase_non_blocking (&flashConfig, start, lengthInBytes,
ERASE_KEY);
if (status != kStatus_FLASH_Success)
{
return status;
}
uint32_t status = FLASH_API_TREE-> flash_get_command_state(&flashConfig);
```

See the possible status codes in Table 210. If the status is kStatus_FLASH_Success return value, it means the FLASH region 0x0 – 0x1000 has been erased.

28.2.2.19 flash_get_command_state

The API is used for getting the command state, and work with non-blocking erase feature.

Prototype :

```
status_t (*flash_get_command_state)(flash_config_t *config);
```

Table 209. Parameters used in flash_get_command_state API

Parameter	Description
config	Pointer to flash_config_t data structure in memory to store driver runtime state.

The Example can review to the above flash_erase_non_blocking API use.

28.2.2.20 Possible return codes of the FLASH APIs

Table 210. Return codes for FLASH APIs

Return code	Value	Description

Table continues on the next page...

Table 210. Return codes for FLASH APIs (continued)

kStatus_FLASH_Success	0	API is executed successfully
kStatus_FLASH_InvalidArgument	4	An invalid argument is provided
kStatus_FlashSizeError	100	Not used.
kStatus_FlashAlignmentError	101	Address or length does not meet the required alignment.
kStatus_FlashAddressError	102	Address or length is outside addressable memory.
kStatus_FLASH_CommandFailure	105	The INT_STATUS[FAIL] bit is set.
kStatus_FlashUnknownProperty	106	Unknown Flash property.
kStatus_FlashEraseKeyError	107	The key provided does not match the programmed flash key.
kStatus_FlashRegionExecuteOnly	108	The area of flash is protected as execute-only.
kStatus_FLASH_ExecuteInRamFunctionNotReady	109	Execute-in-RAM function is not available
kStatus_FLASH_CommandNotSupported	111	Flash API is not supported
kStatus_FLASH_ReadOnlyProperty	112	The flash property is read-only
kStatus_FLASH_InvalidPropertyValue	113	The flash property value is out of range
kStatus_FLASH_InvalidSpeculationOption	114	The option of flash prefetch speculation is invalid
kStatus_FLASH_EccError	116	A correctable or uncorrectable error during command execution
kStatus_FLASH_CompareError	117	Destination and source memory contents do not match
kStatus_FLASH_RegulationLoss	118	A loss of regulation during read
kStatus_FLASH_InvalidWaitStateCycles	119	The wait state cycle set to r/w mode is invalid
kStatus_FLASH_OutOfDateCfpaPage	132	CFPA page version is out of date
kStatus_FLASH_BlankIfrPageData	133	Blank page cannnot be read
kStatus_FLASH_EncryptedRegionsEraseNotDoneAtOnce	134	Encrypted flash subregions are not erased at once
kStatus_FLASH_ProgramVerificationNotAllowed	135	Program verification is not allowed when the encryption is enabled

Table continues on the next page...

Table 210. Return codes for FLASH APIs (continued)

kStatus_FLASH_HashCheckError	136	Hash check of page data is failed
kStatus_FLASH_SealedFfrRegion	137	The FFR region is sealed
kStatus_FLASH_FfrRegionWriteBroken	138	The FFR Spec region is not allowed to be written discontinuously
kStatus_FLASH_NmpaAccessNotAllowed	139	The NMPA region is not allowed to be read/ written/erased
kStatus_FLASH_CmpaCfgDirectEraseNotAllowed	140	The CMPA Cfg region is not allowed to be erased directly
kStatus_FLASH_FfrBankIsLocked	141	The FFR bank region is locked
kStatus_FLASH_CfpaScratchPageInvalid	148	CFPA Scratch Page is invalid
kStatus_FLASH_CfpaVersionRollbackDisallowed	149	CFPA version rollback is not allowed

28.2.2.21 Flash APIs Example

The section will provide the example to introduce the Flash APIs usage and help users understand and use more deeply.

Example1(program the image data or specific data into internal flash region):

```
#define ROM_API_TREE ((uint32_t *)0x1302fc00)
#define FLASH_API_TREE ((flash_driver_interface_t*)ROM_API_TREE[4]);
flash_config_t flashConfig;
status_t status = kStatus_Success;
uint32_t load_address = 0x10000;
uint32_t erase_size = 0x800;
uint32_t s_buffer[512];
for (uint32_t i = 0; i < 512; i++)
{
s_buffer[i] = i;
}
uint32_t FlashDriverVersion = FLASH_API_TREE->version;
debug_printf("Flash version= %x\r\n", FlashDriverVersion);
status = FLASH_API_TREE->flash_init(&flashConfig);
debug_printf("flash_init version= %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
return status
}
if (FFR_Init(&flashConfig) == kStatus_Success)
```

```
{
debug_printf("Flash init successfull!!. Halting...\r\n");
}
else
{
error_trap();
}
status = FLASH_API_TREE->flash_erase(&FlashConfig, load_address, erase_size, kFLASH_ApiEraseKey);
debug_printf("flash_erase status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
return status;
}
// call the flash_verify_erase api to check whether the flash region has been erased.
status = FLASH_API_TREE->flash_verify_erase(&FlashConfig, load_address, erase_size);
debug_printf("flash_verify_erase status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
return status;
}
// here s_buffer data can also is replaced by the image data that is used for dual image boot
status = FLASH_API_TREE->flash_program(&FlashConfig, load_address, (uint8_t *)s_buffer, sizeof(s_buffer);
debug_printf("flash_program status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
return status;
}
// call the flash_verify_program api to check whether the s_buffer data has been loaded into the
flash region.
uint32_t failedAddress, failedData;
status = FLASH_API_TREE->flash_verify_program (&FlashConfig, load_address, sizeof(s_buffer), (const
uint8_t *)s_buffer, &failedAddress, &failedData);
debug_printf("flash_verify_program status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
return status;
}
```

Example1(update the ffr data and get the data from the ffr region):

```
#define ROM_API_TREE ((uint32_t*)0x1302FC00U)
```

```
#define FLASH_API_TREE((flash_driver_interface_t*)ROM_API_TREE[4]);
flash_config_t flashConfig;
uint32_t pflashBlockBase = 0;
uint32_t cmpa_key_store_address = 0x3e400;
uint8_t pData[512];
uint8_t pData1[512] ;
uint8_t uuid[16];
uint32_t ffr_buffer_len = 512 ;
status_t status = kStatus_Success;
status = FLASH_API_TREE->flash_init(&flashConfig);
debug_printf("flash_init version= %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
return status
}
status = FLASH_API_TREE->ffr_init(&flashConfig);
debug_printf("ffr_init version= %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
return status
}
status = FLASH_API_TREE->flash_get_property(&FlashConfig,
kFLASH_PropertyPflashBlockBaseAddr, &pflashBlockBase);
debug_printf("kFLASH_PropertyPflashBlockBaseAddr = %x\r\n", pflashBlockBase);
if (status != kStatus_FLASH_Success)
{
return status;
}
// call the ffr_infield_page_write to load the CfpaData into the CFPA scratch region, and the CfpaData is
provided by user //and the version of the CfpaData should been larger than before or same as before , but
the CMPA_PROG_IN_PROGRESS has been //set.
status = FLASH_API_TREE->ffr_infield_page_write(&FlashConfig, (uint8_t *)&CfpaData, FFR_BUFFER_LEN);
debug_printf("ffr_infield_page_write status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
return status;
}
status = FLASH_API_TREE->ffr_get_customer_infield_data(&FlashConfig, (uint8_t *)pData, 0, ffr_buffer_len);
debug_printf("ffr_get_customer_infield_data status = %x\r\n", status);
```

```
if (status != kStatus_FLASH_Success)
{
return status;
}
// call the ffr_cust_factory_page_write to load the CmpaData into the CMPA region, and the CMpaData is
provided by user.
status = FLASH_API_TREE->ffr_cust_factory_page_write(&FlashConfig, (uint8_t *)CmpaData, false);
debug_printf("ffr_cust_factory_page_write status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
return status;
}
status = FLASH_API_TREE->ffr_get_customer_data(&FlashConfig, (uint8_t *)pData, 0, ffr_buffer_len);
debug_printf("ffr_get_customer_data status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
return status;
}
status = FLASH_API_TREE->ffr_get_uuid(&FlashConfig, uuid);
debug_printf("ffr_get_uuid status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
return status;
}
// call the ffr_keystore_write to load the CMPA_KeyStore into the CMPA key_store region, and the
CMPA_KeyStore is provided by //user.
status = FLASH_API_TREE->ffr_keystore_write(&FlashConfig, (ffr_key_store_t *)CMPA_KeyStore);
debug_printf("ffr_keystore_write status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
return status;
}
status = FLASH_API_TREE->flash_get_cust_keystore(&FlashConfig, (uint8_t *)pData, 0, ffr_buffer_len);
debug_printf("flash_get_cust_keystore status = %x\r\n", status);
if (status != kStatus_FLASH_Success)
{
return status;
}
```

```
status = FLASH_API_TREE->flash_read(&FlashConfig, cmpa_key_store_address, (uint8_t
*)pData1, ffr_buffer_len);

debug_printf("flash_read status = %x\r\n", status);

if (status != kStatus_FLASH_Success)

{

return status;

}

status = memcmp((const void*) pData, (const void*) pData1, ffr_buffer_len);

debug_printf("cmpa key_store read data compare status = %x\r\n", status);

if (status != 0)

{

return status;

}

// call the ffr_lock to lock the ffr region and the ffr region can not be accessed until to the next reset
and call the //ffr_init api

status = FLASH_API_TREE->ffr_lock(&FlashConfig);

debug_printf("ffr_lock status = %x\r\n", status);

if (status != kStatus_FLASH_Success)

{

return status;

}
```

28.2.3 runBootloader API

The ROM bootloader provides an API for the user application to enter the ISP mode based on the designated ISP interface mode.

Prototype

```
void (*runBootloader)(void *arg);
```

Table 211. API prototype fields

Field	Offset	Description
Tag	[31:24]	Fixed value: 0xEB (Enter Boot)
Boot Mode	[23:20]	0 - Enter passive mode
		1 - Enter ISP mode
ISP Interface	[19:16]	0 - Auto detection
		1 - USB-HID
		2 - UART
		3 - SPI

Table continues on the next page...

Table 211. API prototype fields (continued)

Field	Offset	Description
		4 - I2C
		5 - CAN
Reserved	[15:04]	
Image Index	[03:00]	Used for Boot Mode 0

Example:

In the application image boot process, regardless of whether the ISP pin is connected to the high level, the device will directly enter the ISP mode through the UART interface according to the parameter isp Interface in arg.

```
Change to
#define ROM_API_TREE ((uint32_t *)0x1302FC00U)
#define RUN_BOOTLOADER_API_TREE ((void (*)(void *)) ROM_API_TREE[0])
uint32_t arg = 0xEB120000;   //0xEB: represents Enter Boot; 0x12: represents enter ISP mode by UART
only
RUN_BOOTLOADER_API_TREE->runBootloader(&arg);
```

After the application image, which calls the above runBootloader API, has booted successfully, the device will only allow the UART interface to be connected to transfer the data with the host.

28.2.4 FLEXSPI FLASH Driver APIs

The FLEXSPI NOR Driver API set consists of several separate APIs to ease the external FLASH support. The API set provides the following features:

- Recognize an external Serial NOR FLASH device based on the JESD216 or above revision
- Probe the Octal Serial NOR FLASH via simplified option
- Program data to or read data from an external Serial NOR FLASH device
- Partially or fully erase an external Serial NOR FLASH device.
- Support for most Serial NOR FLASH devices
- API for feature extension

28.2.4.1 General description

The FLEXSPI FLASH APIs are organized in the flexspi_nor_flash_driver_t structure. See the details in data structure below. See the details of each API in the subsequent sections.

```
typedef struct
{
uint32_t version;
status_t (*init)(uint32_t instance, flexspi_nor_config_t *config);
status_t (*page_program)(uint32_t instance, flexspi_nor_config_t *config, uint32_t dstAddr,
const uint32_t *src);
status_t (*erase_all)(uint32_t instance, flexspi_nor_config_t *config);
status_t (*erase)(uint32_t instance, flexspi_nor_config_t *config, uint32_t start,
uint32_t length);
status_t (*erase_sector)(uint32_t instance, flexspi_nor_config_t *config, uint32_t address);
```

```
status_t (*erase_block)(uint32_t instance, flexspi_nor_config_t *config, uint32_t address);
status_t (*get_config)(uint32_t instance, flexspi_nor_config_t *config,
serial_nor_config_option_t *option);
status_t (*read)(uint32_t instance, flexspi_nor_config_t *config, uint32_t *dst,
uint32_t start, uint32_t bytes);
status_t (*xfer)(uint32_t instance, flexspi_xfer_t *xfer);
status_t (*update_lut)(uint32_t instance, uint32_t seqIndex, const uint32_t *lutBase,
uint32_t numberOfSeq);
status_t (*set_clock_source)(uint32_t clockSrc);
void (*config_clock)(uint32_t instance, uint32_t freqOption, uint32_t sampleClkMode);
status_t (*partial_program)(uint32_t instance, flexspi_nor_config_t *config, uint32_t dstAddr, const
uint32_t *src, uint32_t length);
} flexspi_nor_flash_driver_t;
```

Each FLEXSPI FLASH API depends on a common *flexspi_nor_config_t* parameter to perform the proper operation. The *flexspi_nor_config_t* data structure is defined as below; the explanation of each field is available in the ROM boot chapter.

```
/*
*  Serial NOR configuration block
*/
typedef struct _flexspi_nor_config
{
flexspi_mem_config_t memConfig; //!< Common memory configuration info via FlexSPI
uint32_t pageSize;              //!< Page size of Serial NOR
uint32_t sectorSize;            //!< Sector size of Serial NOR
uint8_t ipcmdSerialClkFreq;     //!< Clock frequency for IP command
uint8_t isUniformBlockSize;     //!< Sector/Block size is the same
uint8_t isDataOrderSwapped;     //!< Data order (D0, D1, D2, D3) is swapped (D1,D0, D3, D2)
uint8_t reserved0[1];           //!< Reserved for future use
uint8_t serialNorType;          //!< Serial NOR FLASH type: 0/1/2/3
uint8_t needExitNoCmdMode;      //!< Need to exit NoCmd mode before other IP command
uint8_t halfClkForNonReadCmd;   //!< Half the Serial Clock for non-read command: true/false
uint8_t needRestoreNoCmdMode;   //!< Need to Restore NoCmd mode after IP commmand execution
uint32_t blockSize;             //!< Block size
uint32_t flashStateCtx;         //!< FLASH State Context
uint32_t reserve2[10];          //!< Reserved for future use
} flexspi_nor_config_t;
```

28.2.4.2 version

This field indicates the FlexSPI FLASH driver version number.

Prototype :

```
uint32_t version;
```

Example:

```
#define ROM_API_TREE ((uint32_t*)0x1302FC00U)
#define FLEXSPI_FLASH_API_TREE ((flexspi_nor_flash_driver_t*)ROM_API_TREE[11])
uint32_t FlexSPIFLASHDriverVersion = FLEXSPI_FLASH_API_TREE -> version;
```

In this device, the FlexSPI FLASH Driver version is “0x00010802”, it means the FLEXSPI FLASH API version is 1.8.2.

28.2.4.3 flexspi_nor_flash_init

The API is used for initializing the FlexSPI controller based on the parameters. It requires a full flexspi_nor_config_t data structure to initialize the FLEXSPI controller, clocks, FLASH etc. The data structure can be generated via flexspi_nor_get_config API or in other manners.

Prototype :

```
status_t (*init)(uint32_t instance, flexspi_nor_config_t *config);
```

Table 212. Parameters used in the flexspi_nor_flash_init API

Parameter	Description
instance	Point to the FlexSPI FLASH controller instance. Only instance 0 is supported on this device.
config	This parameter points to the flash configuration block(FCB) which consists of arguments related to an external FLASH device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in FlexSPI FLASH Driver APIs.

Example:

```
#define FLEXSPI_INSTANCE (0)
flexspi_nor_config_t flashConfig;
// get the valid flashConfig data here
// Call the API with proper flashConfig parameter
uint32_t status = FLEXSPI_FLASH_API_TREE->init (FLEXSPI_INSTANCE, &flashConfig);
```

See the possible status codes in Table 226. If the status is kStatus_Success return value, it means FlexSPI FLASH driver initialization is completed successfully.

28.2.4.4 flexspi_nor_flash_page_program

The API is used for programming the page data into the Serial NOR FLASH, and the API should be called after the *flexspi_nor_flash_init* API call.

Prototype :

```
status_t (*page_program)(uint32_t instance, flexspi_nor_config_t *config, uint32_t dstAddr, const
uint32_t *src);
```

Table 213. Parameters used in flexspi_nor_flash_page_program API

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
config	This parameter points to the flash configuration block(FCB), consisting of arguments related to an external FLASH device, such as flash size, page size, and sector size. See the details of the flexspi_nor_config_t in FlexSPI FLASH Driver APIs.
dstAddr	The start address of the required FLASH memory to be programmed.

Table continues on the next page...

Table 213. Parameters used in flexspi_nor_flash_page_program API (continued)

Parameter	Description
src	Pointer to the source buffer of data that is to be programmed into FLASH. The src address must be 32-bit aligned

Example:

```
uint32_t dstAddr = 0x08001000 ;
uint8_t programBuffer[256];
for (uint32_t i=0; i<sizeof(programBuffer); i++)
{
programBuffer[i] = (uint8_t)(i & 0xFF);
}
status = FLEXSPI_FLASH_API_TREE->page_program(FLEXSPI_INSTANCE, &flashConfig, dstAddr,
&programBuffer);
```

See the possible status codes in Table 226. If the status is kStatus_Success return value, it means the data has been written into the specified region of the FLASH (Note: it cannot guarantee the data has been successfully programmed into the FLASH device)

28.2.4.5 flexspi_nor_flash_erase_all

The API is used for erasing the entire external FLASH region, and the API should be called after the *flexspi_nor_flash_init* API call.

Prototype :

```
status_t (*erase_all)(uint32_t instance, flexspi_nor_config_t *config);
```

Table 214. Parameters used in flexspi_nor_flash_erase_all API

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
config	This parameter points to the flash configuration block(FCB) which consists of arguments related to an external FLASH device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in FlexSPI FLASH Driver APIs.

Example:

```
status = FLEXSPI_FLASH_API_TREE-> erase_all (FLEXSPI_INSTANCE, &flashConfig);
```

See the possible status codes in Table 226. If the status is kStatus_Success return value, it means that the chip erase operation is completed on the FLASH side.

28.2.4.6 flexspi_nor_flash_erase

The API is used for erasing the specified region of the external FlexSPI FLASH region, and the API should be called after the *flexspi_nor_flash_init* API call.

Prototype :

```
status_t (*erase)(uint32_t instance, flexspi_nor_config_t *config, uint32_t start, uint32_t length);
```

Table 215. . Parameters used in flexspi_nor_flash_erase API

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
config	This parameter points to the flash configuration block(FCB) which consists of arguments related to an external FLASH device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in FlexSPI FLASH Driver APIs.
start	The start offset address of the required flash memory to be erased. The start address must be page-aligned (that is, a multiple of page of the specifical FlexSPI FLash).
lengthInBytes	The length, given in bytes (not words or long words) to be erased. Must be page-aligned.

Example:

```
uint32_t offset = 0x1000 ;
uint832_t lengthInBytes = 0x1000 ;
status = FLEXSPI_FLASH_API_TREE-> erase (FLEXSPI_INSTANCE, &flashConfig, offset, lengthInBytes);
```

See the possible status codes in Table 226. If the status is kStatus_Success return value, it means the external FlexSPI FLASH region (0x08001000 – 0x08002000) has been erased successfully.

28.2.4.7 flexspi_nor_flash_erase_sector

The API is used for erasing the specified sector of the external FlexSPI FLASH, and the API should be called after the *flexspi_nor_flash_init* API call.

Prototype :

```
status_t (*erase_sector)(uint32_t instance, flexspi_nor_config_t *config, uint32_t address);
```

Table 216. Parameters used in flexspi_nor_flash_erase_sector

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
config	This parameter points to the flash configuration block(FCB) which consists of arguments related to an external FLASH device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in FlexSPI FLASH Driver APIs.
start	The parameter specifies the offset address of the sector to be erased.

Example:

```
uint32_t offset = 0x0 ;
status = FLEXSPI_FLASH_API_TREE-> erase_sector (FLEXSPI_INSTANCE, &flashConfig, offset);
```

See the possible status codes in Table 226. If the status is kStatus_Success return value, it means the specified sector region of the external FlexSPI FLASH region has been erased successfully.

28.2.4.8 flexspi_nor_flash_erase_block

The API is used for erasing the specified block region of the external FlexSPI FLASH, and the API should be called after the *flexspi_nor_flash_init* API call.

Prototype :

```
status_t (*erase_block)(uint32_t instance, flexspi_nor_config_t *config, uint32_t address);
```

Table 217. Parameters used in flexspi_nor_flash_erase_block

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
config	This parameter points to the flash configuration block(FCB) which consists of arguments related to an external FLASH device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in FlexSPI FLASH Driver APIs.
start	The parameter specifies the offset address of the block to be erased.

Example:

```
uint32_t offset = 0x0 ;
status = FLEXSPI_FLASH_API_TREE-> erase_block (FLEXSPI_INSTANCE, &flashConfig, offset);
```

See the possible status codes in Table 226. If the status is kStatus_Success return value, it means the specified block region of the external FlexSPI FLASH has been erased successfully.

28.2.4.9 flexspi_nor_get_config

The API is used for getting the FlexSPI NOR configuration block based on specified option. It can probe the FLASH type and generate the flexspi_nor_config_t for later full initialization. Typically, the flexspi_nor_flash_init will be performed with the flexspi_nor_config_t parameter probed by the flexspi_nor_get_config API.

Note:

This FlexSPI Flash API only support the Quad FlexSPI Flash type.

Prototype :

```
status_t (*get_config)(uint32_t instance, flexspi_nor_config_t *config, serial_nor_config_option_t
*option);
```

Table 218. Parameters used in flexspi_nor_get_config API

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
config	This parameter points to the flash configuration block(FCB) which consists of arguments related to an external FLASH device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in FlexSPI FLASH Driver APIs.
option	This parameter provides a simplified option for the FlexSPI driver to probe the FLASH and get the FLASH parameters later. See the details of the flexspi_nor_config_t in FLEXSPI FLASH Driver APIs. Definitions of this block as below.

```
typedef struct _serial_nor_config_option
{
union
{
struct
{
uint32_t max_freq : 4;          //!< Maximum supported Frequency
uint32_t misc_mode : 4;         //!< miscellaneous mode
uint32_t quad_mode_setting : 4; //!< Quad mode setting
uint32_t cmd_pads : 4;          //!< Command pads
uint32_t query_pads : 4;        //!< SFDP read pads
uint32_t device_type : 4;       //!< Device type
uint32_t option_size : 4;       //!< Option size, in terms of uint32_t,size = (option_size + 1)* 4
uint32_t tag : 4;               //!< Tag, must be 0x0E
} B;
uint32_t U;
} option0;
union
{
struct
{
uint32_t dummy_cycles : 8;     //!< Dummy cycles before read
uint32_t status_override : 8;  //!< Override status register value during device mode
configuration
uint32_t pinmux_group : 4;     //!< The pinmux group selection
uint32_t dqs_pinmux_group : 4; //!< The DQS Pinmux Group Selection
uint32_t drive_strength : 4;   //!< The Drive Strength of FlexSPI Pads
uint32_t flash_connection : 4; //!< FLASH connection option: 0 - Single FLASH connected to port A,
1 - Parallel mode, 2 - Single FLASH connected to Port B
} B;
uint32_t U;
} option1;
} serial_nor_config_option_t;
```

Table 219. serial_nor_config_option_t option

Offset	Field	Description
0	Option0	See option0 definition for more details

Table continues on the next page...

Table 219. serial_nor_config_option_t option (continued)

Offset	Field	Description
4	Option1	Optional, effective only if the option Size field in Option0 is non-zero See option1 definition for more details.

Table 220. option0 definition

Field	Bits	Description
tag	31:28	The tag of the config option, fixed to 0x0C
option_size	27:24	Size in bytes = (Option Size + 1) * 4 It is 0 if only option0 is required
device_type	23:20	Device Detection Type 0 - Read SFDP for SDR commands 1 - Read SFDP for DDR Read commands 2 - HyperFLASH 1V8 3 - HyperFLASH 3V 4 - Macronix Octal DDR 6 - Micron Octal DDR 8 - Adesto EcoXiP DDR
query_pad	19:16	Data pads during Query command (read SFDP or read MID) 0 - 1 2 - 4 3 – 8
cmd_pad	15:12	Data pads during FLASH access command 0 - 1 2 - 4 3 – 8
quad_mode_setting	11:8	Quad Mode Enable Setting 0 - Not configure 1 - Set bit 6 in Status Register 1 2 - Set bit 1 in Status Register 2 3 - Set bit 7 in Status Register 2 4 - Set bit 1 in Status Register 2 via 0x31 command

Table continues on the next page...

Table 220. option0 definition (continued)

Field	Bits	Description
		NOTE This field will be effective only if the device is compliant with JESD216 only 5 - (9 longword SDFP table)
misc_mode	7:4	Miscellaneous Mode 0 - Not enabled 1 - Enable 0-4-4 mode for High Random Read performance 3 - Data Order Swapped mode (for MXIC OctaFLASH only) **NOTE** Experimental feature, do not use in products, keep it as 0
max_freq	3:0	Max FLASH Operation speed 0 - Don't change the FlexSPI clock setting

Table 221. option1 definition

Field	Bits	Description
flash_connection	31:28	Dummy cycles for a read command
		0 - Use detected dummy cycle
		Others - dummy cycles provided in flash datasheet
drive_strength	27:24	Override status register value during device mode configuration
dqs_pinmux_group	23:20	The pinmux group selection
pinmux_group	19:16	The DQS Pinmux Group Selection
status_override	15:08	The Drive Strength of FlexSPI Pads
dummy_cycles	7:00	FLASH connection option: 0 - Single FLASH connected to port A, 1 -Parallel mode, 2 - Single FLASH connected to Port B

Typical Option configurations are as below:

- QUAD NOR - Quad SDR Read: option0 = 0xc0000004 (75MHz)
- QUAD NOR - Quad DDR Read: option0 = 0xc0100003 (60MHz)

Example:

```
#define FLASH_OPTION_QSPI_SDR 0xc0000001
serial_nor_config_option_t flashConfigOption = { .option0 = { .U = FLASH_OPTION_QSPI_SDR } };
status = FLEXSPI_FLASH_API_TREE-> get_config(FLEXSPI_INSTANCE, &flashConfig, &flashConfigOption);
```

See the possible status codes in Table 226. If the status is kStatus_Success return value, it means the FlexSPI FLASH config block has been configured based on the provided option successfully.

28.2.4.10 flexspi_nor_flash_read

The API is used for getting the FlexSPI NOR configuration block based on specified option , and the API should be called only after the FLEXSPI IP has been properly configured.

Prototype :

```
status_t (*read)(uint32_t instance, flexspi_nor_config_t *config, uint32_t *dst, uint32_t start,
uint32_t bytes);
```

Table 222. Parameters used in flexspi_nor_flash_read API

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
config	This parameter points to the flash configuration block(FCB) which consists of arguments related to an external FLASH device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in FlexSPI FLASH Driver APIs.
dst	Pointer to the buffer used for storing the read data. The dst address must be 32-bit aligned
start	The offset address of the required flash memory to be read.
lengthInBytes	The length, given in bytes (not words or long words) to be erased. Must be page-aligned.

Example:

```
uint32_t start_offset = 0x1000;
uint32_t read_buffer[512] = {0};
status = FLEXSPI_FLASH_API_TREE->read(FLEXSPI_INSTANCE, &flashConfig, & read_buffer, start_offset,
sizeof(read_buffer));
```

See the possible status codes in Table 226. If the status is kStatus_Success return value, it means the FlexSPI FLASH (0x08001000 – 0x08001800) has been read and stored the data into the read_buffer successfully.

28.2.4.11 flexspi_command_xfer

The API is used for performing the FlexSPI command, this API should be called only when the FLEXSPI IP is properly configured.

Prototype :

```
status_t flexspi_command_xfer(uint32_t instance, flexspi_xfer_t *xfer);
```

Table 223. Parameters used in flexspi_command_xfer API

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
xfer	This parameter pointer to FlexSPI Transfer Context which consists of arguments related to FlexSPI operation, sequence, TX and RX related parameters.

Example:

```
uint32_t start_offset = 0x1000;
uint32_t read_buffer[512] = {0};
flexspi_xfer_t flashXfer;
flashXfer.operation = kFlexSpiOperation_Read;
flashXfer.seqNum = 1;
flashXfer.seqId = NOR_CMD_LUT_SEQ_IDX_READ;
while (sizeof(read_buffer))
{
uint32_t readLength = bytes > 65535 ? 65535 : bytes;
flashXfer.baseAddress = start_offset;
flashXfer.rxBuffer = read_buffer;
flashXfer.rxSize = readLength;
status = FLEXSPI_FLASH_API_TREE->flexspi_command_xfer (instance, &flashXfer);
if (status != kStatus_Success)
{
return status;
}
bytes -= readLength;
start_offset += readLength;
read_buffer += readLength / sizeof(uint32_t);
}
return status;
```

See the possible status codes in Table 226. If the status is kStatus_Success return value, it means the FlexSPI FLASH command_xfer has executed the read command successfully, and the read data (0x08001000 – 0x08001800) has been load into the read_buffer array.

NOTE

This API relies on the valid LUT to perform the supported operations. If the LUT is invalid, the result is unpredictable.

28.2.4.12 flexspi_update_lut

The API is used for configuring the FlexSPI Lookup table before calling the flexspi_command_xfer API. This API should be called only when the FLEXSPI IP is properly configured.

Prototype :

```
status_t (*update_lut)(uint32_t instance, uint32_t seqIndex, const uint32_t *lutBase, uint32_t
numberOfSeq);
```

Table 224. Parameters used in flexspi_update_lut API

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
seqIndex	This parameter pointer to FlexSPI Dedicated LUT Sequence Index for IP Command.
lutBase	This parameter pointer to FlexSPI Lookup table , which holds FLASH command sequences
numberOfSeq	This parameter points to FlexSPI Sequence Number

Example:

```
// Program Pattern
const uint32_t probe_pattern[4] = { 0x33221100, 0x77665544, 0xbbaa9988, 0xffeeddcc };
flexspi_xfer_t flashXfer;
flashXfer.baseAddress = sizeof(flexspi_nor_config_t);
flashXfer.seqId = NOR_CMD_LUT_SEQ_IDX_PAGEPROGRAM;
flashXfer.operation = kFlexSpiOperation_Write;
flashXfer.txBuffer = (uint32_t *)probe_pattern;
flashXfer.txSize = sizeof(probe_pattern);
flexspi_update_lut(instance,NOR_CMD_LUT_FOR_IP_CMD,&flashConfig-
>memConfig.lookupTable[4*flashXfer.seqId], flashXfer.seqNum);
flashXfer.seqId = NOR_CMD_LUT_FOR_IP_CMD;
status = flexspi_command_xfer(instance, &flashXfer);
```

See the possible status codes in Table 226. If the status is kStatus_Success return value, it means the flexspi_update_lut has execute successfully, and the the parode_pattern data has been load into the (baseAddress + 0x08000000) region..

28.2.4.13 flexspi_set_clock_source

The API is used for configuring the FlexSPI clock source. This API should be called only when the FlexSPI IP is properly configured.

Prototype:

```
status_t (*set_clock_source)(uint32_t clockSrc);
```

Parameter	Description
clockSrc	Point to the FlexSPI select clock source Flexspi clock select: • 0000 - Main clock • 0001 - PLL0 clock • 0010 - No clock • 0011 - FRO_HF • 0100 - No clock • 0101 - PLL1 clock • Others – No clock

Example:

```
uint32_t flexspi_clcok_source = 0x0;
status = FLEXSPI_FLASH_API_TREE-> set_clock_source (flexspi_clcok_source);
```

See the possible status codes in Table 160. If the status is kStatus_Success return value, it means the set_clock_source has execute successfully, and the FlexSPI has configure the clock source from the main clock.

28.2.4.14 flexspi_config_clock

The API is used for configuring the FlexSPI clock. This API should be called only when the FLEXSPI IP is properly configured.

Prototype:

```
void (*config_clock)(uint32_t instance, uint32_t freqOption, uint32_t sampleClkMode);
```

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
freqOption	This parameter pointer to FlexSPIFlexSPI flash serial clock frequency: kFlexSpiSerialClk_30MHz (1U) kFlexSpiSerialClk_50MHz (2U) kFlexSpiSerialClk_60MHz (3U) kFlexSpiSerialClk_75MHz (4U) kFlexSpiSerialClk_100MHz (5U)

sampleClkMode	This parameter pointer to configure the FlexSPI clock configuration type.
	kFlexSpiClk_SDR (0U) //!< Clock configure for SDR mode
	kFlexSpiClk_DDR (1U) //!< Clock configurat for DDR mode

Example:

```
uint32_t flexspi_freqOption = 0x1;
uint32_t flexspi_sampleClkMode = 0x0;
status = FLEXSPI_FLASH_API_TREE-> config_clock(FLEXSPI_INSTANCE, flexspi_freqOption,
flexspi_sampleClkMode);
```

See the possible status codes in Table 160. If the status is kStatus_Success return value, it means the config_clock has execute successfully, and the flexspi flash clock wiil be configured with the provided parameter.

28.2.4.15 flexspi_nor_flash_partial_program

The API is used for partially programming the data into the specified address of FlexSPI FLASH, and the API should be called after the *flexspi_nor_flash_init* API call.

Prototype :

```
status_t (*partial_program)(uint32_t instance, flexspi_nor_config_t *config, uint32_t dstAddr, const
uint32_t *src, uint32_t length);
```

Table 225. Parameters used in the flexspi_nor_flash_partial_program API

Parameter	Description
instance	Point to the FlexSPI controller instance, only support 0
config	This parameter points to the flash configuration block(FCB) which consists of arguments related to an external FLASH device, for example, flash size, page size, sector size. See the details of the flexspi_nor_config_t in FlexSPI FLASH Driver APIs.
dstAddr	The start address of the required FlexSPI flash memory to be programmed.
src	Pointer to the source buffer of data that is to be programmed into FlexSPI flash.
length	The program size of the source buffer.

Example:

```
uint32_t flashAhbAddr = 0x08001000UL;
uint32_t flashAhbBase = 0x08000000UL;
uint32_t flashOffset = flashAhbAddr - flashAhbBase;
uint32_t programBuffer[256 / sizeof(uint32_t)];
uint8_t *buf_u8 = (uint8_t*)programBuffer;
for (uint32_t i=0; i<sizeof(programBuffer); i++)
{
*buf_u8++ = (uint8_t)(i & 0xFFU);
}
status = FLEXSPI_FLASH_API_TREE->partial_program(FLEXSPI_INSTANCE, &flashConfig, flashOffset,
&programBuffer, sizeof(programBuffer));
```

See the possible status codes in Table 226. If the status is kStatus_Success return value, it means the data has been written into the FlexSPI FLASH.

28.2.4.16 Possible return codes for FLEXSPI FLASH driver APIs

Table 226. Return codes for FLEXSPI FLASH APIs

Error Code	Value	Description
kStatus_Success	0	Operation succeeded without error.
kStatus_Fail	1	The operation failed with a generic error.
kStatus_Invalidargument	4	The requested command's argument is undefined.
kStatus_FLEXSPI_SequenceExecutionTimeout	6000	The FLEXSPI Sequence Execution timeout
kStatus_FLEXSPI_InvalidSequence	6001	The FLEXSPI LUT sequence invalid
kStatus_FLEXSPI_DeviceTimeout	6002	The FLEXSPI device timeout

Table continues on the next page...

Table 226. Return codes for FLEXSPI FLASH APIs (continued)

kStatus_FLEXSPINOR_ProgramFail	20100	Page programming failure
kStatus_FLEXSPINOR_EraseSectorFail	20101	Sector Erase failure
kStatus_FLEXSPINOR_EraseAllFail	20102	Chip Erase failure
kStatus_FLEXSPINOR_WaitTimeout	20103	The execution time out
kStatus_FlexSPINOR_NotSupported	20104	Page size overflow
kStatus_FlexSPINOR_WriteAlignmentError	20105	Address alignment error
kStatus_FlexSPINOR_CommandFailure	20106	Erase/Program Verify Error
kStatus_FlexSPINOR_SFDP_NotFound	20107	Timeout occurs during the API call
kStatus_FLEXSPINOR_Unsupported_SFDP_Version	20108	Unrecognized SFDP version
kStatus_FLEXSPINOR_FLASH_NotFound	20109	Flash detection failure
kStatus_FLEXSPINOR_DTRRead_DummyProbeFailed	20110	DDR Read dummy probe failure

28.2.4.17 FlexSPI Flash APIs Example

The section will provide the example to introduce All the FlexSPI Flash APIs usage and help users understand and use more deeply.Example(Program the image or specific data into the FlexSPI Flash region):

```
#define ROM_API_TREE ((uint32_t *)0x1302FC00U)
#define FLEXSPI_FLASH_API_TREE ((flexspi_nor_flash_driver_t*)ROM_API_TREE[11]);
#define FLEXSPI_INSTANCE 0
#define FLEXSPI_PROGRAM_OFFSET (0x1000u)
#define FLEXSPI_ERASE_SIZE (0x2000u)
#define MAX_PAGE_SIZE 512
flexspi_nor_config_t flashConfig;
uint32_t FlexSPIFLASHDriverVersion = FLEXSPI_FLASH_API_TREE->version;
debug_printf("FlexSPIFLASHDriverVersion version= %x\r\n", FlexSPIFLASHDriverVersion);
// choose the Quad SDR FlexSPI Flash type as the example.
```

```
serial_nor_config_option_t flashConfigOption = { .option0 = { .U = 0xc0000001}};

status = FLEXSPI_FLASH_API_TREE->get_config(FLEXSPI_INSTANCE, &flashConfig, &flashConfigOption);

debug_printf("flexspi_nor_get_config API status=%d!\n",status);

if (status != kStatus_Success)

{

return status;

}

//Init the FlexSPI Flash base on the flashConfig, which is configured from the get_config API.

status = FLEXSPI_FLASH_API_TREE->init(FLEXSPI_INSTANCE, &flashConfig);
```

```
debug_printf("flexspi_nor_flash_init returned error:%d\n", status);
if (status != kStatus_Success)
{
return status;
}
status = FLEXSPI_FLASH_API_TREE->erase(FLEXSPI_INSTANCE, &flashConfig,
FLEXSPI_PROGRAM_OFFSET, FLEXSPI_ERASE_SIZE)
debug_printf ("flexspi_nor_flash_erase returned status:%d\n", status);;
if (status != kStatus_Success)
{
return status;
}
// Here created the specific data programBuffer to program the data into the FlexSPI Flash, also the data
can be replaced by // image data provided by user.
uint32_t programBuffer[MAX_PAGE_SIZE / sizeof(uint32_t)];
uint8_t *pBuffer = (uint8_t *)&programBuffer[0];
for (uint32_t i = 0; i < sizeof(programBuffer); i++)
{
*pBuffer++ = (uint8_t)(i & 0xFF);
}
status = FLEXSPI_FLASH_API_TREE->page_program(FLEXSPI_INSTANCE, &flashConfig,
FLEXSPI_PROGRAM_OFFSET, programBuffer);
debug_printf("flexspi_nor_flash_page_program returned error:%d\n", status);
if (status != kStatus_Success)
{
return status;
}
uint32_t readBuffer[MAX_PAGE_SIZE];
status = FLEXSPI_FLASH_API_TREE->read(FLEXSPI_INSTANCE, &flashConfig, readBuffer, FLEXSPI_PROGRAM_OFFSET,
flashConfig.pageSize);
debug_printf("flexspi_nor_flash_read status = %d\n", status);
if (status != kStatus_Success)
{
return status;
}
else
{
if (memcmp(readBuffer, programBuffer, flashConfig.pageSize) == 0)
{
```

```
debug_printf ("Verify programming by IP read: PASSED\n");
}
}
status = FLEXSPI_FLASH_API_TREE->erase(FLEXSPI_INSTANCE, &flashConfig,
FLEXSPI_PROGRAM_OFFSET, FLEXSPI_ERASE_SIZE)
debug_printf ("flexspi_nor_flash_erase returned status:%d\n", status);;
if (status != kStatus_Success)
{
return status;
}
```

28.2.5 OTP-eFuse APIs

28.2.5.1 Version

This fields indicates the version number of the OTP API in this device.

Prototype :

```
standard_version_t version;
```

Table 227. Definition of the version Field in OTP API tree

Parameter	Description

standard_version_t	Pointer to version structure to store current OTP-eFuse driver version information
	uint8_t bugfix; //!< bugfix version [7:0]
	uint8_t minor; //!< minor version [15:8]
	uint8_t major; //!< major version [23:16]
	char name; //!< name [31:24]

Example:

```
#define ROM_API_TREE ((uint32_t *)0x1302FC00U)
#define OTP_API_TREE ((efuse_driver_t*)ROM_API_TREE[12])
uint32_t otpDriverVersion = OTP_API_TREE-> version;
```

In this device, the OTP Driver version is "0x45010000", it means the OTP driver version is 1.0.0.

28.2.5.2 otp_init

The API is used for initializing the OTP clock. The *otp_init* API should be called before all the other OTP APIs.

Prototype :

```
status_t (*init)(void);
```

Example :

```
uint32_t status = OTP_API_TREE->init ();
```

See the possible status codes in Table 230. If the status is kStatus_Success return value, it means OTP driver initialization completed successfully.

28.2.5.3 otp_deinit

This API is used to disable OTP controller APB clock.

Prototype :

```
status_t (*deinit)(void);
```

Example :

```
uint32_t status = OTP_API_TREE->deinit ();
```

See the possible status codes in Table 230. If the status is kStatus_Success return value, it means OTP driver deinitialization completed successfully.

28.2.5.4 otp_read

This API is used to read data from OTP-eFuse controller.

Prototype :

```
status_t (*p_efuse_read)(uint32_t addr, uint32_t *data);
```

Table 228. Parameters used in otp_read API

Parameter	Description
addr	Point to the OTP-eFuse index that specifies the OTP-eFuse word to be read out.
data	Pointer to the buffer used for storing the read data.

Example :

```
#define EFUSE_INDEX (0x0)
uint32_t eFuseData;
uint32_t status = OTP_API_TREE-> p_efuse_read (EFUSE_INDEX, & eFuseData);
```

See the possible status codes in Table 230. If the status is kStatus_Success return value, it means the OTP data read completed successfully.

28.2.5.5 otp_program

This API is used to program data to specified OTP Word.

Prototype :

```
status_t (*p_efuse_program)(uint32_t addr, uint32_t data);
```

Table 229. Parameters used in the otp_program API

Parameter	Description
addr	Point to the OTP-eFuse index that specifies the OTP-eFuse word to be programmed
data	Point to the source data used for programming into the eFuse region.

Example :

```
#define EFUSE_INDEX (0x7)
uint32_t eFuseData = 0x01010101;
uint32_t status = OTP_API_TREE-> p_efuse_program (EFUSE_INDEX, & eFuseData);
```

See the possible status codes in Table 230. If the status is kStatus_Success return value, it means the data 0x01010101 has been programmed into the eFuse index 7(CUSTOMER_WORD0) region.

28.2.5.6 Possible return codes of OTP API

Table 230. Return codes for OTP APIs

Error Code	Value	Description
kStatus_Success	0	Operation succeeded without error.
kStatus_Fail	1	The operation failed with a generic error.
kStatus_ReadOnly	2	Requested value cannot be changed because it is read-only.
kStatus_OutOfRange	3	Requested value is out of range.
kStatus_InvalidArgument	4	The requested command's argument is undefined.
kStatus_Timeout	5	A timeout occurred.
kStatus_NoTransferInProgress	6	No send in progress

28.2.5.7 OTP-eFuse APIs Example

The section will provide the example to introduce All the OTP-eFuse APIs usage and help users understand and use more deeply.

NOTE

The OTP-eFuse program is only-program-once, user need to make sure the program data is valid value.

Example:

```
#define ROM_API_TREE ((uint32_t *)0x1302FC00U)
#define OTP_API_TREE ((efuse_driver_t*)ROM_API_TREE[12])
#define EFUSE_CUST_INDEX 7
```

```
#define EFUSE_CMPA_CRC_INDEX 1
uint32_t otpDriverVersion = OTP_API_TREE->version;
debug_printf("otpDriverVersion version= %x\r\n", otpDriverVersion);
uint32_t status = OTP_API_TREE->init();
debug_printf("efuse init status= %x\r\n", status);
if (status != kStatus_Success)
{
return status;
}
//the otp efuse value program should not program the invalid data into the efuse region, otherwise
the device will boot fail //with the invalid check of the CRC and CMAC of the CMPA.
uint32_t fuseLCVal = 0;
status_t status = OTP_API_TREE->efuse_program(EFUSE_CUST_INDEX, fuseLCVal);
debug_printf("efuse_program LC state status= %x\r\n", status);
if (status != kStatus_Success)
{
return status;
}
uint32_t fuseCMPACRCVal = 0x12345678; //here crc value of the cmpa is just the example data, user
need to                                  //set the valid crc data of the cmpa
status_t status = OTP_API_TREE->efuse_program(EFUSE_CMPA_CRC_INDEX, fuseCMPACRCVal);
debug_printf("efuse_program crc value of the CMPA status= %x\r\n", status);
if (status != kStatus_Success)
{
return status;
}
uint32_t fuseCustVal;
uint32_t fuseCMPACRCVal1;
status_t status = OTP_API_TREE->efuse_read(EFUSE_CUST_INDEX, &fuseCustVal);
debug_printf("efuse_read LC state status = %x \r\n", status);
if (status != kStatus_Success)
{
return status;
}
status_t status = OTP_API_TREE->efuse_read(EFUSE_CMPA_CRC_INDEX, & fuseCMPACRCVal1);
debug_printf("efuse_program crc value of the CMPA status= %x \r\n", status,);
if (status != kStatus_Success)
{
return status;
}
status_t status = OTP_API_TREE->deinit();
debug_printf("efuse deinit status= %x\r\n", status);
if (status != kStatus_Success)
{
return status;
}
```

28.2.6 IAP APIs

28.2.6.1 General description

The general IAP APIs call and execution flow is depicted on Figure 126.

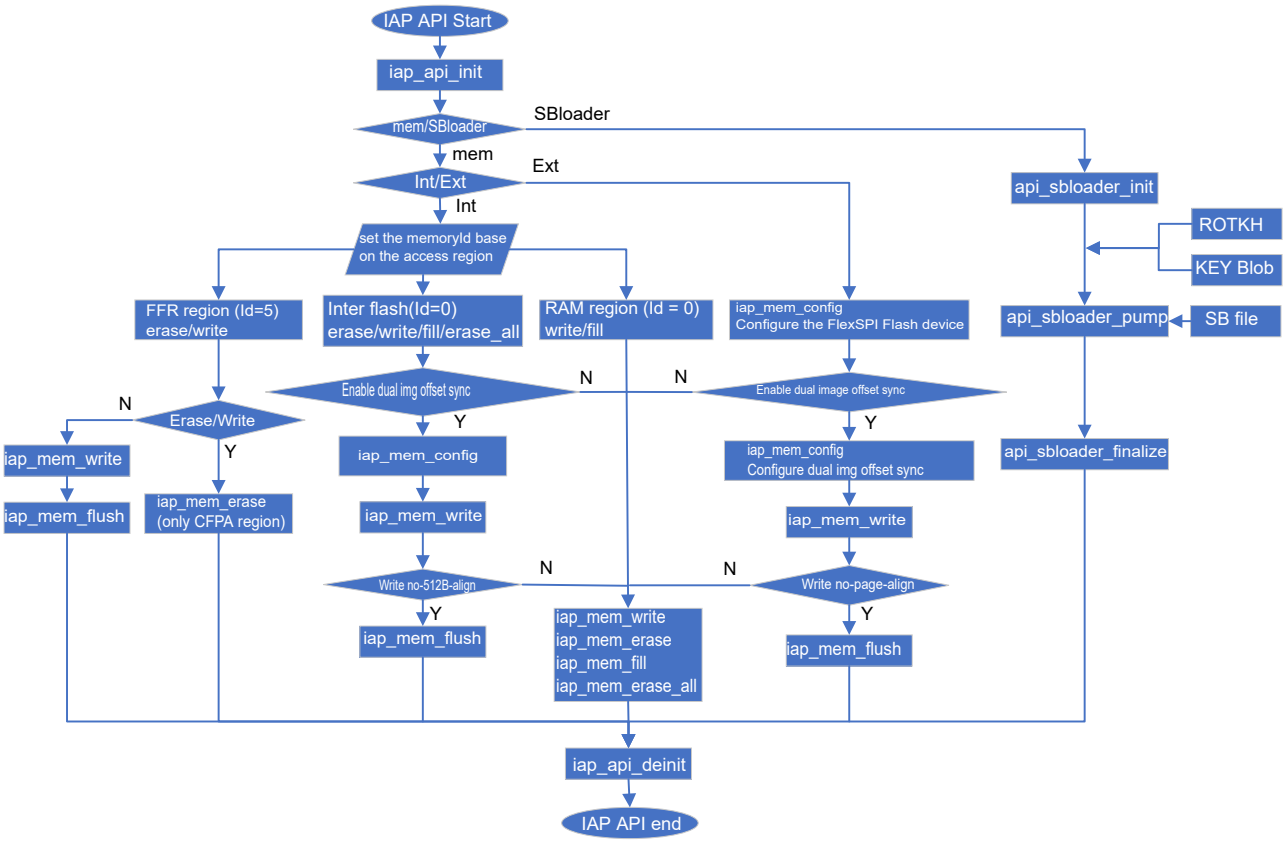


Figure 126. IAP API call flow

The bootloader IAP APIs are organized in the iap_api_interface_t structure, see the details from the below data structure. See the details of each API in the subsequent sections.

```
typedef struct iap_api_interface_struct
{
standard_version_t version; //!< IAP API version number.
status_t (*api_init)(api_core_context_t *coreCtx, const kp_api_init_param_t *param);
status_t (*api_deinit)(api_core_context_t *coreCtx);
status_t (*mem_init)(api_core_context_t *ctx);
status_t (*mem_read)(api_core_context_t *ctx, uint32_t addr, uint32_t len, uint8_t *buf,
uint32_t memoryId);
status_t (*mem_write)(api_core_context_t *ctx, uint32_t addr, uint32_t len, const uint8_t *buf,
uint32_t memoryId);
status_t (*mem_fill)(api_core_context_t *ctx, uint32_t addr, uint32_t len, uint32_t pattern,
uint32_t memoryId);
status_t (*mem_flush)(api_core_context_t *ctx);
status_t (*mem_erase)(api_core_context_t *ctx, uint32_t addr, uint32_t len, uint32_t memoryId);
status_t (*mem_config)(api_core_context_t *ctx, uint32_t *buf, uint32_t memoryId);
status_t (*mem_erase_all)(api_core_context_t *ctx, uint32_t memoryId);
} iap_api_interface_t;
```

Each IAP API depends on the common context named api_core_context_t. The detail of the context is defined as bellow:

```
//!@brief The API context structure
```

```
typedef struct api_core_context
{
    uint32_t reserved0[10];
    uint32_t reserved1[3];
    flash_config_t flashConfig;
    flexspi_nor_config_t flexspinorCfg;
    uint32_t reserved2[2];
    uint32_t reserved3[1];
    nboot_context_t *nbootCtx;
    uint8_t *sharedBuf;
    uint32_t reserved4[6];
} api_core_context_t;
```

The IAP APIs require a dedicated RAM region as the buffer for each operation. The user application needs to prepare an unused RAM region organized as the *kp_api_init_param_t* structure, and pass this structure to the iap_api_init API.

These fields are managed by api_core_context_t parameter, the user application doesn't need to touch it.

NOTE

Before IAP APIs are used, caller needs to make sure that ELS clocks are enabled. This is required for correct operation of sbloader and iap_mem API on devices with security subsystem.

28.2.6.2 version

The version field indicates the IAP API version number in the device.

Prototype :

```
standard_version_t version;
```

Table 231. Definition of the version in IAP API tree

Parameter	Description

standard_version_t	Pointer to version structure to store current FLASH driver version information
	uint8_t bugfix; //!< bugfix version [7:0]
	uint8_t minor; //!< minor version [15:8]
	uint8_t major; //!< major version [23:16]
	char name; //!< name [31:24]

Example:

```
#define ROM_API_TREE ((uint32_t *)0x1302FC00U)
#define IAP_API_TREE ((iap_api_interface_t*)ROM_API_TREE[13])
uint32_t IAPVersion = IAP_API_TREE-> version;
```

In this SoC, the version of IAP Driver is "0x4B030100", it means the API version is 1.0.0.

28.2.6.3 iap_api_init

The API is used for initializing the IAP API context for the other IAP APIs, this API must be called prior to other IAP APIs.

Prototype :

```
status_t (*api_init)(api_core_context_t *coreCtx, const kp_api_init_param_t *param);
```

Table 232. Parameters used in iap_api_init API

Parameter	Description
coreCtx	Pointer to IAP API core context structure.
param	Pointer to IAP API initialization data structure, used for storing the data during execute the IAP APIs.

Example:

```
api_core_context_t apiCoreCtx;
const kp_api_init_param_t apiInitParam = { .allocStart = 0x30010000, .allocSize = 0x6000 };
uint32_t status = IAP_API_TREE->api_init(&apiCoreCtx, &apiInitParam);
```

See the possible status codes in Table 241. If the status is kStatus_Success return value, it means the IAP API operating environment has been init successfully.

28.2.6.4 iap_api_deinit

The API is used for de-initialize the IAP API context.

Prototype :

```
status_t (*api_deinit)(api_core_context_t *coreCtx);
```

Table 233. Parameters used in iap_api_deinit API

Parameter	Description
coreCtx	Pointer to IAP API core context structure.

Example:

```
uint32_t status = IAP_API_TREE->api_deinit(&apiCoreCtx);
```

See the possible status codes in Table 241. If the status is kStatus_Success return value, it means the IAP API operating environment has been freed and can not call the other IAP APIs.

28.2.6.5 iap_mem_init

The API is used for initializing the memory interface, including the internal FLASH, RAM, FLEXSPI FLASH.

Prototype :

```
status_t (*iap_mem_init)(api_core_context_t *coreCtx);
```

Table 234. Parameters used in the iap_mem_init API

Parameter	Description
coreCtx	Pointer to IAP API core context structure.

Example:

```
uint32_t status = IAP_API_TREE->iap_mem_init(&apiCoreCtx);
```

See the possible status codes in Table 241. If the status is kStatus_Success return value, it means the IAP API memory interfaces have been init successfully.

NOTE

The mem_init will also be called in the api_init API, so the mem_init no need to be called again after calling the api_init.

28.2.6.6 iap_mem_write

The API is used for programing the data into the internal FLASH, RAM or the external FlexSPI FLASH base on the parameters, such as start address and memoryId, and this API should be called after the iap_api_init API.

Prototype :

```
status_t iap_mem_write(api_core_context_t *coreCtx, uint32_t start, uint32_t lengthInBytes, const
uint8_t *buf, uint32_t memoryId)
```

Table 235. Parameters used in iap_mem_write API

Parameter	Description
coreCtx	Pointer to IAP API core context structure.
start	Points to the program address.
lengthInBytes	Points to the program data size.
buf	Pointer to source buffer data that will be programed.

memoryId	Points to the memory id of the different memory regions.
	#define kMemoryID_Internal (0x0U) //!< Internal FLASH/RAM ID
	#define kMemoryID_FFR (0x5U) //!< FFR region ID
	#define kMemoryID_FlexspiNor (0x9U) //!< FlexSPI NOR ID

Example1:

```
uint32_t start1 = 0x10000 ;
uint8_t programBuffer[512];
for (uint32_t i=0; i<sizeof(programBuffer); i++)
{
programBuffer[i] = (uint8_t)(i & 0xFF);
}
```

```
uint32_t status = IAP_API_TREE->mem_write(&apiCoreCtx, start1, sizeof(programBuffer), (uint8_t *)
programBuffer, 0);
```

Example2:

```
uint32_t start2 = 0x08001000 ;
uint8_t programBuffer[512];
for (uint32_t i=0; i<sizeof(programBuffer); i++)
{
programBuffer[i] = (uint8_t)(i & 0xFF);
}
uint32_t status = IAP_API_TREE->mem_write(&apiCoreCtx, start2, sizeof(programBuffer), (uint8_t *)
programBuffer, 9);
```

See the possible status codes in Table 241. If the status is kStatus_Success return value, it means the programBuffer data has been programed into the FLASH region in the example1 and programed into the external FlexSPI FLASH in the example2.

NOTE

- Before calling the mem_write API, the internal flash should be erased first by calling the iap_mem_erase.
- Before calling the mem_write API to program the data into the external FlexSPI FLASH, the external flash should be configured first by calling the iap_mem_config.
- If the programBufferis not 512 alignment or program the data into the ffr region, need to call the iap_mem_flush to program the last less than 512 data into the memory region.

NOTE

If the iap_mem_write used for the FlexSPI Flash data program, the program data should be padded page-alignment and add additional few data(like one Byte) to use the iap_mem_flush API to load the expect program data.

28.2.6.7 iap_mem_fill

The API is used for filling the specified pattern data into the memory region with the specifical data length. This api is mainly for filling the FLASH configuration option related words into the internal RAM. This API should be called after the iap_api_init API.

Prototype :

```
status_t iap_mem_fill(api_core_context_t *coreCtx, uint32_t start, uint32_t lengthInBytes, uint32_t
pattern, uint32_t memoryId)
```

Table 236. Parameters used in iap_mem_fill API

Parameter	Description
coreCtx	Pointer to IAP API core context structure.
start	Points to the program address.
lengthInBytes	Points to the program data size.
pattern	Points to pattern data that will be programed.
memoryId	Points to the memory id of the different memory regions. #define kMemoryID_Internal (0x0U) //!< Internal FLASH/RAM ID

Table continues on the next page...

Table 236. Parameters used in iap_mem_fill API (continued)

Parameter	Description
	#define kMemoryID_FFR (0x5U) //!< FFR region ID #define kMemoryID_FlexspiNor (0x9U) //!< FlexSPI NOR ID

NOTE

The iap_mem_fill can not fill with the start address 0x0 (should be set non-zero address) and the last page region of the internal flash(0x3da00 - 0x3dc00).

Example:

```
uint32_t start = 0x10000 ;
uint32_t pattern = 0xa5a55a5a;
uint32_t status = IAP_API_TREE->mem_fill(&apiCoreCtx, start, 0x2000, pattern, 0);
```

See the possible status codes in Table 241. If the status is kStatus_Success return value, it means the pattern data has been programed into the FLASH region (0x10000 – 0x12000).

28.2.6.8 iap_mem_erase

The API is used for erasing the internal FLASH region, FLEXSPI FLASH region via the memory interface. This API should be called after the iap_api_init API.

Prototype :

```
status_t iap_mem_erase(api_core_context_t *coreCtx, uint32_t start, uint32_t lengthInBytes, uint32_t
memoryId)
```

Table 237. Parameters used in iap_mem_erase API

Parameter	Description
coreCtx	Pointer to IAP API core context structure.
start	Points to the erase address.
lengthInBytes	Points to the erase size.

memoryId	Points to the memory id of the different memory regions.
	#define kMemoryID_Internal (0x0U) //!< Internal FLASH/RAM ID
	#define kMemoryID_FFR (0x5U) //!< FFR region ID
	#define kMemoryID_FlexspiNor (0x9U) //!< FlexSPI NOR ID

NOTE

The iap_mem_erase can not erase the last page (0x3da00-0x3dc00) in the internal flash. The last page can be erased by the flash_erase API or iap_mem_erase_all.

Example:

```
uint32_t start = 0x10000 ;
uint32_t lengthInBytes = 0x2000;
uint32_t status = IAP_API_TREE->mem_erase(&apiCoreCtx, start, lengthInBytes, 0);
```

See the possible status codes in Table 241. If the status is kStatus_Success return value, it means the FLASH region (0x10000 – 0x12000) has be erased.

28.2.6.9 iap_mem_erase_all

The API is used for erasing the entire internal FLASH or FLEXSPI FLASH based on the memoryId parameter. This API should be called after the iap_api_init API.

Prototype :

*status_t iap_mem_erase_all(api_core_context_t *coreCtx, uint32_t memoryId)*

Table 238. Parameters used in iap_mem_erase_all API

Parameter	Description
coreCtx	Pointer to IAP API core context structure.

memoryId	Points to the memory id of the different memory regions.
	#define kMemoryID_Internal (0x0U) //!< Internal FLASH/RAM ID
	#define kMemoryID_FFR (0x5U) //!< FFR region ID
	#define kMemoryID_FlexspiNor (0x9U) //!< FlexSPI NOR ID

Example:

```
uint32_t status = IAP_API_TREE->mem_erase_all(&apiCoreCtx, 0);
```

See the possible status codes in Table 241. If the status is kStatus_Success return value, it means the FLASH region has be erased all.

28.2.6.10 iap_mem_config

The API is used for configuring the FLexSPI FLASH and enabling the dual image start address synchronization feature. This API should be called after the iap_api_init API.

Prototype :

```
status_t iap_mem_config(api_core_context_t *coreCtx, uint32_t *config, uint32_t memoryId)
```

Table 239. Parameters used in iap_mem_config API

Parameter	Description
coreCtx	Pointer to IAP API core context structure.

Table continues on the next page...

Table 239. Parameters used in iap_mem_config API (continued)

Parameter	Description
config	Pointer to the congifure data which will be used for configuring the memory region with the specifical encrypt feature bas on the memory id.

memoryId	Points to the memory id of the different memory regions.
	#define kMemoryID_Internal (0x0U) //!< Internal FLASH/RAM ID
	#define kMemoryID_FFR (0x5U) //!< FFR region ID
	#define kMemoryID_FlexspiNor (0x9U) //!< FlexSPI NOR ID

-

Example1:

```
typedef struct
{
uint32_t reserved : 24;
uint32_t tag : 8; // Fixed to 0x52 ('R')
} dual_image_update_option_t;
dual_image_update_option_t dualImgUpdateOption = { .tag = 0x52 };
uint32_t status = IAP_API_TREE->mem_config(&apiCoreCtx, (uint32_t *)& dualImgUpdateOption, 0);
```

See the possible status codes in Table 241. If the status is kStatus_Success return value, it means:

1. The external FlexSPI FLASH has been configured with the QSPI SDR mode and enable the region 0x08001000 – 0x08001800 .
2. The dual image start address synchronization feature has been enabled when using the dual image boot feature, the access address will be processed with the remap offset to access the actual address.

28.2.6.11 iap_mem_flush

The API is used for flushing the data stored in the memory buffer into the destination memory region. This API should be called after the iap_api_init API.

Prototype :

```
status_t iap_mem_flush(api_core_context_t *coreCtx);
```

Table 240. Parameters used in iap_mem_flush API

Parameter	Description
coreCtx	Pointer to IAP API core context structure.

Example:

```
uint32_t cfpastart = 0x3dc00 ;
uint8_t cfpaBuffer[512];
for (uint32_t i=0; i<sizeof(cfpaBuffer); i++)
{
cfpaBuffer [i] = (uint8_t)(i & 0xFF);
```

```
}
uint32_t status = IAP_API_TREE->mem_write(&apiCoreCtx, cfpastart, sizeof(cfpaBuffer), (uint8_t *)
cfpaBuffer, 5);
if (status != kStatus_Success)
{
return status;
}
uint32_t status = IAP_API_TREE->mem_flush(&apiCoreCtx);
```

See the possible status codes in Table 241. If the status is kStatus_Success return value, it means the cfpaBuffer data has been programed into the CFPA region. In the process of the FFR program and when the data of programed into the internal and external FLASH region is not page-align, user need to call the iap_mem_flush after the iap_mem_write to store the data into the actual memory region.

28.2.6.12 Possible return codes for IAP driver APIs

Table 241. Return and Error codes for IAP APIs

Error Code	Value	Description
kStatus_IAP_Success	0	IAP API execution succedded
kStatus_IAP_Fail	1	IAP API exeution failed
kStatus_IAP_InvalidArgument	100001	Invalid argument detected during API execution
kStatus_IAP_OutOfMemory	100002	The Heap size is not enough during API execution
kStatus_IAP_ReadDisallowed	100003	The read memory operation is disallowed during API execution
kStatus_IAP_CumulativeWrite	100004	The FLASH region to be programmed is not empty
kStatus_IAP_EraseFailuire	100005	Erase operation failed
kStatus_IAP_CommandNotSupported	100006	The specific command is not supported
kStatus_IAP_MemoryAccessDisabled	100007	Memory access is disabled, typically occurred on the FLEXSPI NOR if it is not configured properly
kStatus_Success	0	Operation succeeded without error.
kStatus_Fail	1	The operation failed with a generic error.
kStatus_Invalidargument	4	The requested command's argument is undefined.
kStatusRomLdrSectionOverrun	10100	means reached the end of the sb file processing
kStatusRomLdrSignature	10101	the signature or version are incorrect
kStatusRomLdrSectionLength	10102	the bootOffset/ new section count is out of range
kStatusRomLdrUnencryptedOnly	10103	the unencrypted image is disabled
kStatusRomLdrEOFReached	10104	the end of the image file is reached

Table continues on the next page...

Table 241. Return and Error codes for IAP APIs (continued)

Error Code	Value	Description
kStatusRomLdrChecksum	10105	The checksum of a command tag block is invalid.
kStatusRomLdrCrc32Error	10106	The CRC-32 of the data for a load command is incorrect.

28.2.6.13 IAP API examples

The section will provide the example to introduce All the IAP APIs usage and help users understand and use more deeply.

```
 active="1'd0">
#define ROM_API_TREE ((uint32_t *)0x1302FC00U)
#define IAP_API_TREE ((iap_api_interface_t*)ROM_API_TREE[13])
#define FLASH_BUFFER_LEN 0x500
#define CFPA_SCRATCH_ADDRESS 0x3dc00U
#define CMPA_ADDRESS 0x3e200U
#define CMPA_KEY_STORE_ADDRESS 0x3e400U
```

```
 active="1'd0">
status = IAP_API_TREE->mem_erase(&apiCoreCtx, FLASH_PROGRAM_ADDRESS, FLASH_ERASE_LEN, flashMemId);
```

Example1(update the dual-image into the internal flash):

```

#define ROM_API_TREE ((uint32_t *)0x1302FC00U)
#define IAP_API_TREE ((iap_api_interface_t*)ROM_API_TREE[13]);
```

```
#define FLASH_PROGRAM_ADDRESS 0x0;

#define FLASH_ERASE_LEN 0x2000;

#define flashMemId 0

typedef struct

{

uint32_t reserved : 24;

uint32_t tag : 8; // Fixed to 0x52 ('R')

} dual_image_update_option_t;

uint32_t status = kStatus_Success;
```

```

 api_core_context_t apiCoreCtx;
 // user need to provide the apiInitParam, that is used as the ram region for the IAP APIs call.
 const kp_api_init_param_t apiInitParam = { .allocStart = 0x30010000, .allocSize = 0x6000 };
 status = IAP_API_TREE->api_init(&apiCoreCtx, &apiInitParam);
```

```
debug_printf("IAP API api_init return status=%d!\n",status);

if (status != kStatus_Success)

{

return status;

}
```

```
// Configure the dual-image update into specific lower version image region
dual_image_update_option_t dualImgUpdateConfigOption = { .tag = 0x52 };
uint32_t flashMemId = 0x0U;
status = iap_mem_config(&apiCoreCtx, (uint32_t *)&dualImgUpdateConfigOption, flashMemId);
debug_printf("iap_mem_config status = %d\n", status);
if (status != kStatus_Success)
{
return status;
}
// The erase address will be processed with the remap offset to access the actual address, if the lower
version image is //stored in the 0x0, the erase address will be 0, if the lower version image is stored
in the remap offset region, the erase //address will be 0x0 + remap_offset. No need user to set the erase
address, which is just processed by API.
status = IAP_API_TREE->mem_erase(&apiCoreCtx, FLASH_PROGRAM_ADDRESS, FLASH_ERASE_LEN, flashMemId);
debug_printf("iap_mem_erase return status=%d!\n",status);
if (status != kStatus_Success)
{
return status;
}
// The write address will be processed with the remap offset to access the actual address, if the lower
version image is //stored in the 0x0, the write address will be 0, if the lower version image is stored
in the remap offset region, the erase //address will be 0x0 + remap_offset. No need user to set the write
address, which is just processed by API.
status = IAP_API_TREE->mem_write(&apiCoreCtx, FLASH_PROGRAM_ADDRESS, sizeof(s_buffer), (uint8_t
*)s_buffer, flashMemId);
debug_printf("iap_mem_write status = %d\n", status);
if (status != kStatus_Success)
{
return status;
}
// If the load data is not 512B-align, use need to call the mem_flush to program the last remaining no
512B-align data.
status = IAP_API_TREE->mem_flush(&apiCoreCtx);
debug_printf("iap_mem_flush status = %d\n", status);
if (status != kStatus_Success)
{
return status;
}
```

Chapter 29
Power Management Controller (PMC)

29.1 Chip-specific PKC information

Digital Core logic Brown Out Dectector control (BODCORE[TRIGLVL]) for device 0A:

Field	BoD Trigger Level			
2-0 TRIGLVL	000b-101b		Reserved	
	110b	-	0.9	When fclk<= 130MHz
	111b	-	0.95	When fclk >130MHz

Digital Core logic Brown Out Dectector control (BODCORE[TRIGLVL]) for device 1B:

Field	BoD Trigger Level			
2-0 TRIGLVL	000b-100b		Reserved	
	101b	-	0.929V	When fclk <100MHz
	110b	-	0.984V	When 101MHz<=fclk<= 130MHz
	111b	-	1.038V	When fclk >130MHz

29.2 Overview

This chapter provides an overview of power related information of the device.

The device includes a variety of features to fine tune power usage to match the requirements for different performance levels and reduced power modes.

The device power management infrastructure consists of:

- The Power Management Unit (PMU)
- The Power Management Controller (PMC)
- The Software Power API functions

29.2.1 Block diagram

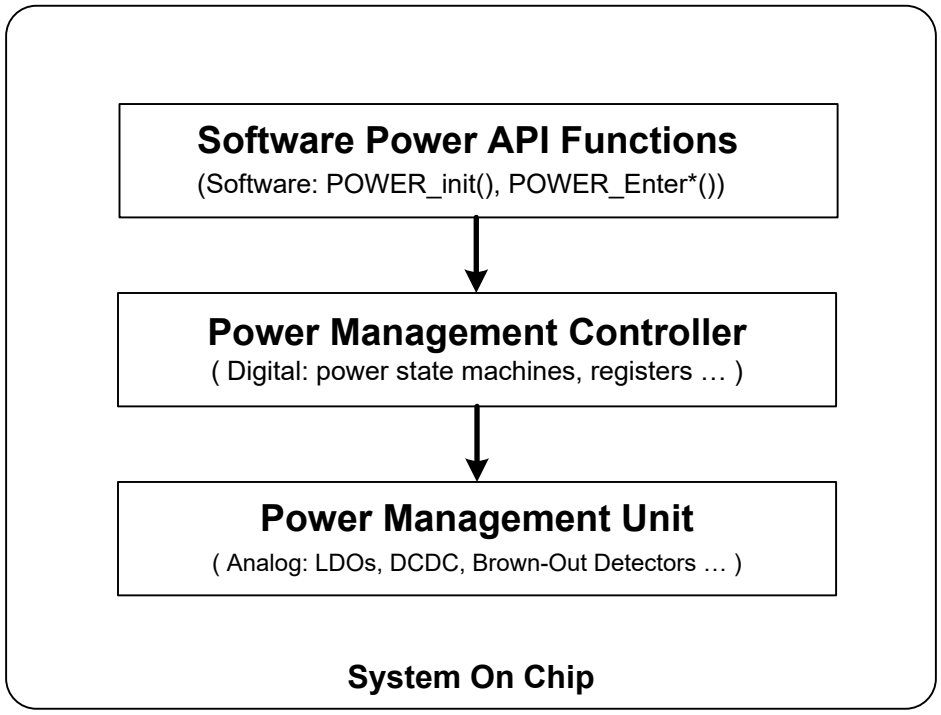


Figure 127. Power management overview

29.2.2 Features

The device power management includes the following features:

- On-chip power regulators.
- Rich set of low power modes.
- Efficient dynamic power control.

29.3 Functional description

The following sections describe functional details of the device power management.

29.3.1 The Power Management Unit (PMU)

The Device Power Management Unit (PMU) consists in several analog modules:

- a set of on-chip regulators: Low-drop output regulators and DC-DC convertor. See On-Chip Power Regulators.
- a set of voltage monitors modules: The Power on Reset (PoR) and The Brown-Out Detectors (BoD).

29.3.1.1 On-Chip Power Regulators

Power inside the chip is provided by six on-chip regulators:

- DC-DC Buck Converter (DCDC)
- Core Logic Low Drop-Out Regulator (LDO_CORE)
- Always-On Low Drop-Out Regulator (LDO_AO)

- SRAM Low Drop-Out Regulator (LDO_MEM)
- OTP-eFUSE Low Drop-Out Regulator (LDO_EFUSE_PROG)
- Flash Memory Low Drop-Out Regulator (LDO_FLASH)

All these six internal regulators are supplied by two independent main external supplies:

- VBAT (1.71 V to 3.6 V)
- VDD_MAIN (1.8 V to 3.6 V)
- On boot-up, the chip is powered by default using the internal LDO (default voltage is 1.1 V). The user can switch to the internal DCDC using power API call.

29.3.1.1.1 DC-DC Buck Converter (DCDC)

The on-chip DC-DC Buck converter converts an input supply voltage (ranging from1.8V to 3.6V via VDD_MAIN) to a fixed output voltage based on the frequency provided in the power API.

With the Core Logic Low Drop-Out Regulator (LDO_CORE), the DC-DC Buck converter can be the supply source for :

- the Core logic Power Domain (PD_CORE) (see Core Logic Power Domain (PD_CORE)),
- the System Power Domain (PD_SYSTEM) (see System Power Domain (PD_SYSTEM)) during ACTIVE (ACTIVE Mode) , SLEEP (SLEEP Mode) and DEEP-SLEEP (DEEP-SLEEP Mode) power modes.

NOTE

The switching between the DCDC and the LDO_CORE as the supply source for PD_CORE and PD_SYSTEM Power Domains is done using the POWER_SetCorePowerSource(...) function which is provided in the power API library (see power API).

NOTE

The optimum output voltage of the DCDC is automatically set up when calling the POWER_SetVoltageForFrequency(...) function which is provided in the power API library (see power API).

29.3.1.1.2 Core Logic Low Drop-Out Regulator (LDO_CORE)

The Core Logic Low Drop-Out Regulator converts input supply voltage (ranging from1.8V to 3.6V via VDD_MAIN) to a fixed output voltage.

With the DC-DC Buck converter (DCDC), the Core Logic Low Drop-Out Regulator (LDO_CORE) can be the supply source for :

- the Core logic Power Domain (PD_CORE) (see Core Logic Power Domain (PD_CORE)),
- the System Power Domain (PD_SYSTEM) (see System Power Domain (PD_SYSTEM)) during ACTIVE (ACTIVE Mode) , SLEEP (SLEEP Mode) and DEEP-SLEEP (DEEP-SLEEP Mode) power modes.

NOTE

The switching between the LDO_CORE and the DCDC as the supply source for PD_CORE and PD_SYSTEM Power Domains is done using the POWER_SetCorePowerSource(...) function which is provided in the power API library (see power API).

NOTE

The optimum output voltage of the LDO_CORE is automatically set up when calling the POWER_SetVoltageForFrequency(...) and POWER_EnterDeepSleep(...) functions in ACTIVE mode. Both functions are provided in the power API library (see power API).

29.3.1.1.3 OTP-eFUSE Low Drop-Out Regulator (LDO_EFUSE_PROG)

The OTP-eFUSE Low Drop-Out Regulator (LDO_EFUSE_PROG) converts input supply voltage (ranging from 1.8 V to 3.6 V via VDD_MAIN) to a fixed output voltage.

The LDO_EFUSE_PROG provides the high voltage required for OTP-eFUSE programming.

Before programming the OTP-eFUSE, the LDO_EFUSE_PROG must be enabled first. This is done via registers PDRUNCFG0 or PDRUNCFGCLR0. Once enabled, the user must check that the LDO_EFUSE_PROG is ready for operation by making sure that the status bit LDOEFUSEPROGPWROK (in STATUSPWR register) is high.

NOTE

The optimum output voltage of the LDO_EFUSE_PROG during OTP-eFUSE programming is 1.8V. This is the default value and it shall not be modified.

IMPORTANT

Before reading the OTP-eFUSE, the LDO_EFUSE_PROG must be shut down first (see registers PDRUNCFG0 or PDRUNCFGSET0). Once disabled, to make sure that the LDO_EFUSE_PROG is completly shut down, the user must wait at least 10 µs before reading the OTP-eFUSE. Reading the OTP-eFUSE while the LDO_EFUSE_PROG is enabled (or not completely shut down) will return wrong/unreliable values.

29.3.1.1.4 Flash Memory Low Drop-Out Regulator (LDO_FLASH)

The Flash Memory Low Drop-Out Regulator (LDO_FLASH) converts input supply voltage (ranging from 1.8V to 3.6V via VDD_MAIN) to a fixed output voltage.

The LDO_FLASH provides the high voltage required for all Flash operations (Read, Program and Erase).

NOTE

The LDO_FLASH power state (enabled or disabled) management does not require any user action: the LDO_FLASH is enabled during ACTIVE, and SLEEP power modes, it is disabled during DEEP-SLEEP, POWER-DOWN and DEEP-POWER-DOWN low power modes.

29.3.1.1.5 SRAM Memories Low Drop-Out Regulator (LDO_MEM)

The SRAM Memories Low Drop-Out Regulator (LDO_MEM) converts input supply voltage (ranging from 1.8V to 3.6V via VDD_MAIN) to a fixed output voltage.

The LDO_MEM is the supply source for SRAM memories instances located in the power domains PD_MEM_0 (First SRAM memories Power Domain (PD_MEM_0)) and PD_MEM_1(Second SRAM memories Power Domain (PD_MEM_1)) during DEEP-SLEEP, POWER-DOWN and DEEP-POWER-DOWN low power modes. in ACTIVE mode, all SRAM instances are supplied by either the DCDC or the LDO_CORE (High Power mode).

NOTE

The LDO_MEM power state (enabled or disabled) management, as well as its optimum output voltage, do not require any user action.

29.3.1.1.6 Always-On Low Drop-Out Regulator (LDO_AO)

The Always-On Low Drop-Out Regulator (LDO_AO) converts the supply voltage VDD_MAX (ranging from 1.8V to 3.6V) to a fixed output voltage.

VDD_MAX is the output voltage of the VMAX module; The VMAX module compares the chip supply inputs VBAT and VDDMAIN and it outputs the highest: VDD_MAX = maximum(VBAT,VDDMAIN).

The Always-On Low Drop-Out Regulator (LDO_AO) is the supply source for:

- The Always-On power domain PD_AON (Always-On Power Domain (PD_AON)),
- The System power domain PD_SYSTEM(System Power Domain (PD_SYSTEM)) during POWER-DOWN mode.
- The third SRAM memories power domain PD_MEM_2 (Third SRAM memories Power Domain (PD_MEM_2)) during DEEP-SLEEP, POWER-DOWN and DEEP-POWER-DOWN modes.

NOTE

The management of the LDO_AO output voltage does not require any user action.

29.3.1.2 Voltage Monitors

29.3.1.2.1 The Power on Reset (PoR)

The Power on Reset module (PoR) provides under-voltage detection of the internal always-on supply VDD_AO.

The PoR has a built-in hysteresis: If VDD_AO goes below the trip-low level, a whole chip reset will be asserted, and the chip reset will be released only when VDD_AO goes above the trip-high level.

The PoR is configured with a trip low level of 0.705V and a trip high level of 0.87V. These two values are hard-coded inside the PMU and cannot be modified in anyway.

29.3.1.2.2 The Brown-Out Detectors (BoD)

They are two brown-out detectors (BoD) modules which provides under-voltage detection. This functionality is mainly useful for detecting the supply going below a programmable threshold voltage after powering up. Both BoD have a programmable hysteresis. One BoD monitors the external VDDMAIN supply (BOD_VDDMAIN) and the other BoD monitors the internal core logic supply (BOD_CORE).

Figure 128 shows the response of brown-out detector to voltage which is monitored. If the bod_out output is in state ‘0’, the voltage that is monitored has to fall below the programmed ‘threshold’ for the bod_out pin to transition from ‘0’ to ‘1’. If the bod_out output is in state ‘1’, the voltage that is monitor has to exceed the programmed ‘threshold’ value plus the programmed ‘hysteresis’ for the bod_out to transition from ‘1’ to ‘0’ again.

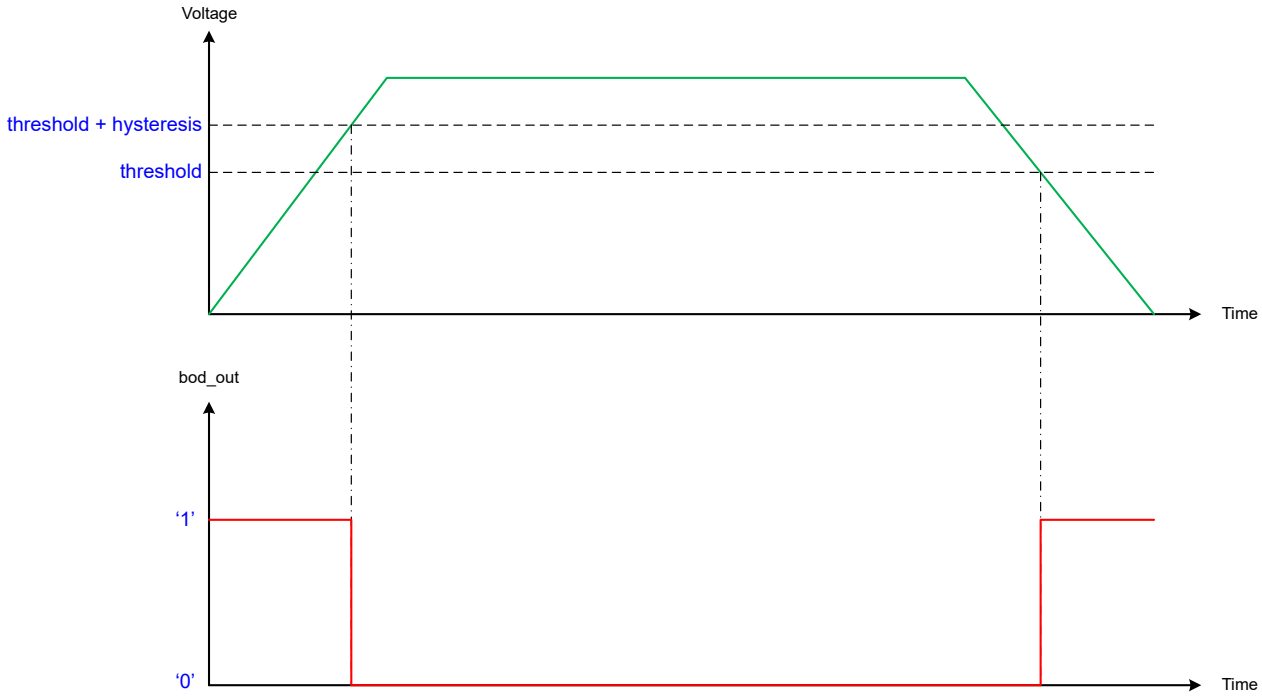


Figure 128. Brown-out detector response

29.3.1.2.2.1 VDDMAIN Brown-Out Detector (BOD_VDDMAIN)

The BOD_VDDMAIN provides under-voltage detection of the external VDDMAIN supply. The low voltage detection threshold can be programmed with a value between 1.75 V and 3.3V, in steps of 100 mV, with 2% maximum error (variation from the selected threshold value). Also, the BOD_VDDMAIN has a programmable hyteresis; possible values are: 25 mV, 50 mV, 75 mV and 100 mV, with 25% maximum error (variation from the selected hysteresis value). See register BODVDDMAIN.

The BOD_VDDMAIN can be configured to (when a low voltage is detected) :

- Reset the whole chip (see register RESETCTRL) and/or
- Generate an interrupt to the CPU (see register in Analog Control module).

NOTE

In case both the chip reset triggering and interrupt generation to CPU are enabled, only the chip reset will occur.

After any PoR reset, Pin reset, BoD or Software resets, the BOD_VDDMAIN is enabled and configured to trigger a chip reset when a low voltage is detected. The default low voltage detection threshold is 1.75 V, with an hysteresis of 50 mV.

After any PoR reset, Pin reset, BoD or Software resets, the chip boot process is blocked until VDDMAIN is above 1.8 V (1.75 V threshold + 0.05V hysteresis).

29.3.1.2.2.2 Core Logic Brown-Out Detector (BOD_CORE)

The BOD_CORE provides under-voltage detection of the internal core supply VDD_CORE. The low voltage detection threshold can be programmed with a value between 0.6V and 0.95V, in steps of 50 mV, with 2% maximum error (variation from the selected threshold value) . Also, the BOD_CORE has a programmable hyteresis; possible values are: 25 mV, 50 mV, 75 mV and 100 mV, with 25% maximum error (variation from the selected hysteresis value). See register BODCORE.

The BOD_CORE can be configured to (when a low voltage is detected) :

- Reset the whole chip (see register RESETCTRL) and/or
- Generate an interrupt to the CPU (see register in Analog Control module).

NOTE

In case both the chip reset triggering and interrupt generation to CPU are enabled, only the chip reset will occur.

After any PoR reset, Pin reset, BoD or Software resets, the BOD_CORE is enabled and configured to trigger a chip reset when a low voltage is detected. The default low voltage detection threshold is 0.8V, with an hysteresis of 50 mV.

29.3.2 The Power Management Controller (PMC)

The Device Power Management Controller (PMC) is the digital control unit of the Power Management Unit (PMU). It also acts as the interface between the PMU and the Software Power API.

29.3.2.1 Power Domains

The chip is partitioned into six power domains:

- Always-On Power Domain (PD_AON)
- Core Logic Power Domain (PD_CORE)
- System Power Domain (PD_SYSTEM)
- First SRAM memories Power Domain (PD_MEM_0)
- Second SRAM memories Power Domain (PD_MEM_1)
- Third SRAM memories Power Domain (PD_MEM_2)

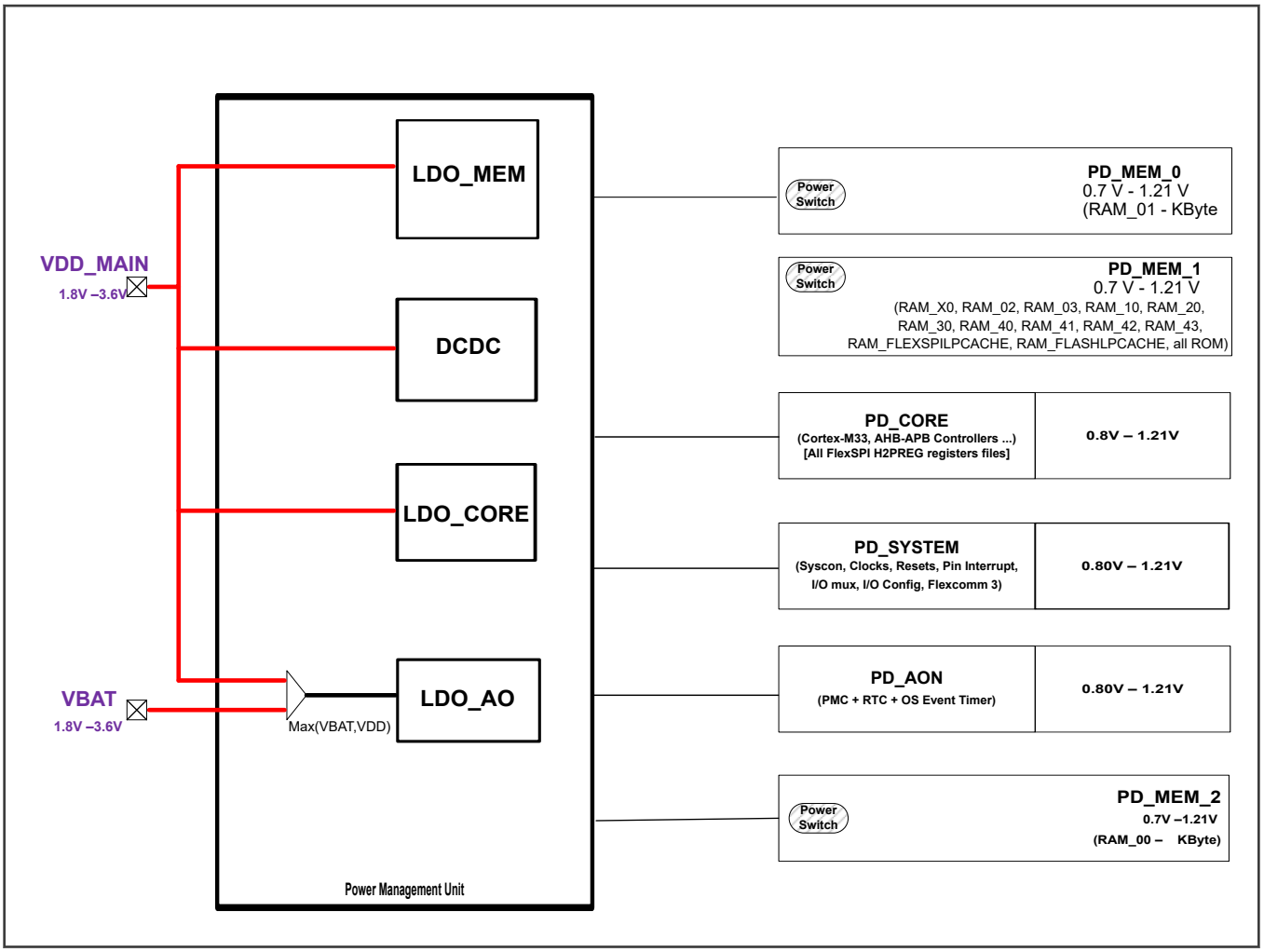


Table 242 shows the detailed list of all modules per power domain.

Table 242. Modules per Power Domains

	Core Logic Power Domain (PD_CORE)	System Power Domain (PD_SYSTEM)	Always-On Power Domain (PD_AON)	First memory Power Domain (PD_MEM_0)	Second memory Power Domain (PD_MEM_1)	Third memory Power Domain (PD_MEM_2)
Input / Output		GPIOs other than the five wake-up GPIO pins.	The five wake-up GPIO pins.			
Analog components			32-KHz Free Running Oscillator (FRO32K)			
			32-KHz Crystal Oscillator (XTAL32K)			

Table continues on the next page...

Table 242. Modules per Power Domains (continued)

	Core Logic Power Domain (PD_CORE)	System Power Domain (PD_SYSTEM)	Always-On Power Domain (PD_AON)	First memory Power Domain (PD_MEM_0)	Second memory Power Domain (PD_MEM_1)	Third memory Power Domain (PD_MEM_2)
			1-MHz Free Running OScillator (FRO1M)			
			Analog Comparator			
	FRO192M (12 MHz, 48 MHz and 96 MHz clocks)					
			VDDMAIN Brown-Out Detector VDDMAIN (BOD_VDDMAIN)			
	Core Logic Brown-Out Detector Core (BOD_CORE)					
	PLL0					
	PLL1					
	High Frequency Crystal Oscillator (XTALHF)					
	Analog to Digital Converters and Temperature Sensors (ADC0, ADC1)					
	High Speed Comparators (HSCMP0, HSCMP1, HSCMP2)					
	Operational Amplifiers (OPAMP0, OPAMP1, OPAMP2)					

Table continues on the next page...

Table 242. Modules per Power Domains (continued)

	Core Logic Power Domain (PD_CORE)	System Power Domain (PD_SYSTEM)	Always-On Power Domain (PD_AON)	First memory Power Domain (PD_MEM_0)	Second memory Power Domain (PD_MEM_1)	Third memory Power Domain (PD_MEM_2)
	Digitital to Analog Converters (DAC0, DAC1, DAC2)					
Digital components	CPU (Cortex-M33)					
	System DMA (SDMA0, SDMA1)					
		Group GPIO Input Interrupt (GINT0, GINT1)				
	General Purpose I/O Controller					
		I/O Configuration (IOCON)				
		I/O Functional Multiplexers				
	Pin Interrupt & Pattern Matching (PINT0 to PINT7)					
	Mirco-Tick Timer(UTICK)					
	Multi-Rate Timer (MRT)					
	Standard Counter/Timers (CTIMER0, CTIMER1,CTIMER2, CTIMER3, CTIMER4)					
	SCTimer/PWM (SCT)					
	CRC Engine					
	FLEXCOMM0,FLEXCOMM1,FL					

Table continues on the next page...

Table 242. Modules per Power Domains (continued)

	Core Logic Power Domain (PD_CORE)	System Power Domain (PD_SYSTEM)	Always-On Power Domain (PD_AON)	First memory Power Domain (PD_MEM_0)	Second memory Power Domain (PD_MEM_1)	Third memory Power Domain (PD_MEM_2)
	EXCOMM2, FLEXCOMM4,FLEXCOMM5,FLEXCOMM6,FLEXCOMM7					
		FLEXCOMM3				
	DMIC					
	USB Full Speed (USBFS)					
			RTC			
	Analog Control					
	Mailbox & Debugger Mailbox					
			OS Event Timer (OSTIMER)			
	FlexSPI					
	CAN					
Digital components	Power Quad (PQ)					
	High Speed SPI					
	I3C					
	All FlexPWM					
	Windowed-Watchdog Timer (WWDT)					
	Quadrature Encoders/ Decoders					
Digital components	Resets Controller					
		Clocks Controller				
		System Configuration (SYSCON)				

Table continues on the next page...

Table 242. Modules per Power Domains (continued)

	Core Logic Power Domain (PD_CORE)	System Power Domain (PD_SYSTEM)	Always-On Power Domain (PD_AON)	First memory Power Domain (PD_MEM_0)	Second memory Power Domain (PD_MEM_1)	Third memory Power Domain (PD_MEM_2)
			Power Management Controller (PMC)			
	ADC & Temperature Sensors Controllers (ADC0, ADC1)					
	DAC Controllers (DAC0, DAC1, DAC2)					
Memories	System ROM (128 Kbytes)					
	FLASH (256 KBytes)					
						RAM_00
				RAM_01		
					RAM_X0	
					RAM_02	
					RAM_03	
					RAM_10	
					RAM_20	
					RAM_30	
					RAM_40	
					RAM_41	
					RAM_42	
					RAM_43	
					Flash Cache SRAM (8 KBytes)	
					FlexSPI Cache SRAM (8 KBytes)	

29.3.2.1.1 Always-On Power Domain (PD_AON)

The Always-On Power domain consists in :

- A set of digital units: The Power Management Controller (PMC), the RTC and OS Event Timer.
- Several analog modules located in the Power Management Unit (PMU) : FRO32K, XTAL32K, FRO1M, Analog Comparator.
- 5 wake-up GPIOs.

This power domain always has power as long as VBAT or VDDMAIN is available with sufficient voltage [1.8V - 3.6V].

29.3.2.1.2 Core Logic Power Domain (PD_CORE)

The Core Logic Power Domain consists of nearly all digital components, for example the Cortex-M33 CPU, System DMA, most of the Flexcomms, etc. All digital units not located in PD_AO or PD_SYSTEM power domains are part of the Core Logic Power Domain.

The Core Logic Power Domain is supplied by:

- LDO_CORE (configured in High Power Mode) or the DCDC in ACTIVE.
- LDO_CORE (configured in Low Power Mode) during DEEP-SLEEP.

The Core Logic Power Domain is shut down during POWER-DOWN and DEEP-POWER-DOWN low power modes.

NOTE

Though part of the Core Logic Power Domain - which is entirely shut down during POWER-DOWN, the following 2 modules will have their states saved during POWER-DOWN (and automatically restored back after wake up from POWER-DOWN) : the Cortex-M33 CPU and Analog Controller.

29.3.2.1.3 System Power Domain (PD_SYSTEM)

The System Power Domain consists of:

- Some system critical digital components: Reset and Clock controllers, System Configuration (SYSCON).
- All GPIOs with their associated configuration and multiplexing (IOCON).
- The Flexcomm3 (UART, SPI, I2C) and both Group GPIO Input Interrupt (GINT0, GINT1).

The System Power Domain is supplied by:

- the LDO_CORE (configured in High Power Mode) or the DCDC in ACTIVE.
- the LDO_CORE (configured in Low Power Mode) during DEEP-SLEEP.
- the LDO_AO during POWER-DOWN.

The System Power Domain is shut down during DEEP-POWER-DOWN low power mode.

29.3.2.1.4 First SRAM memories Power Domain (PD_MEM_0)

The first SRAM Power Domain (PD_MEM0) consists of a single SRAM instance : RAM_01 (4 Kbytes)

PD_MEM0 is supplied by:

- the LDO_CORE (configured in High Power Mode) or the DCDC in ACTIVE.
- the LDO_MEM during DEEP-SLEEP, POWER-DOWN and DEEP-POWER-DOWN low power modes.

NOTE

During DEEP-POWER-DOWN, PD_MEM0 (RAM_01) is shut down if VDDMAIN is NOT available.

29.3.2.1.5 Second SRAM memories Power Domain (PD_MEM_1)

The second SRAM Power Domain (PD_MEM1) consists in these SRAM instances : RAM_X0 (16 Kbytes), RAM_02 (4 Kbytes), RAM_03 (4 Kbytes), RAM_10 (32 Kbytes), RAM_20 (32 Kbytes) , RAM_30 (16 Kbytes), RAM_40 (4 Kbytes), RAM_41 (4 Kbytes), RAM_42 (4 Kbytes), RAM_43 (4 Kbytes).

PD_MEM1 is supplied by:

- the LDO_CORE (configured in High Power Mode) or the DCDC in ACTIVE.
- the LDO_MEM during DEEP-SLEEP, POWER-DOWN and DEEP-POWER-DOWN low power modes.

NOTE

During DEEP-POWER-DOWN, PD_MEM1 is shut down if VDDMAIN is NOT available.

29.3.2.1.6 Third SRAM memories Power Domain (PD_MEM_2)

The third SRAM Power Domain (PD_MEM2) consists in a single SRAM instance : RAM_00 (4 Kbytes)

PD_MEM2 is supplied by:

- the LDO_CORE (configured in High Power Mode) or the DCDC in ACTIVE.
- the LDO_AO during DEEP-SLEEP, POWER-DOWN and DEEP-POWER-DOWN low power modes.

NOTE

Unlike RAM_01 (PD_MEM0), RAM_00 can always be used as a retention memory in DEEP-POWER-DOWN mode, wether or not VDDMAIN is present.

29.3.2.2 Power Modes

The Device supports a variety of power modes.

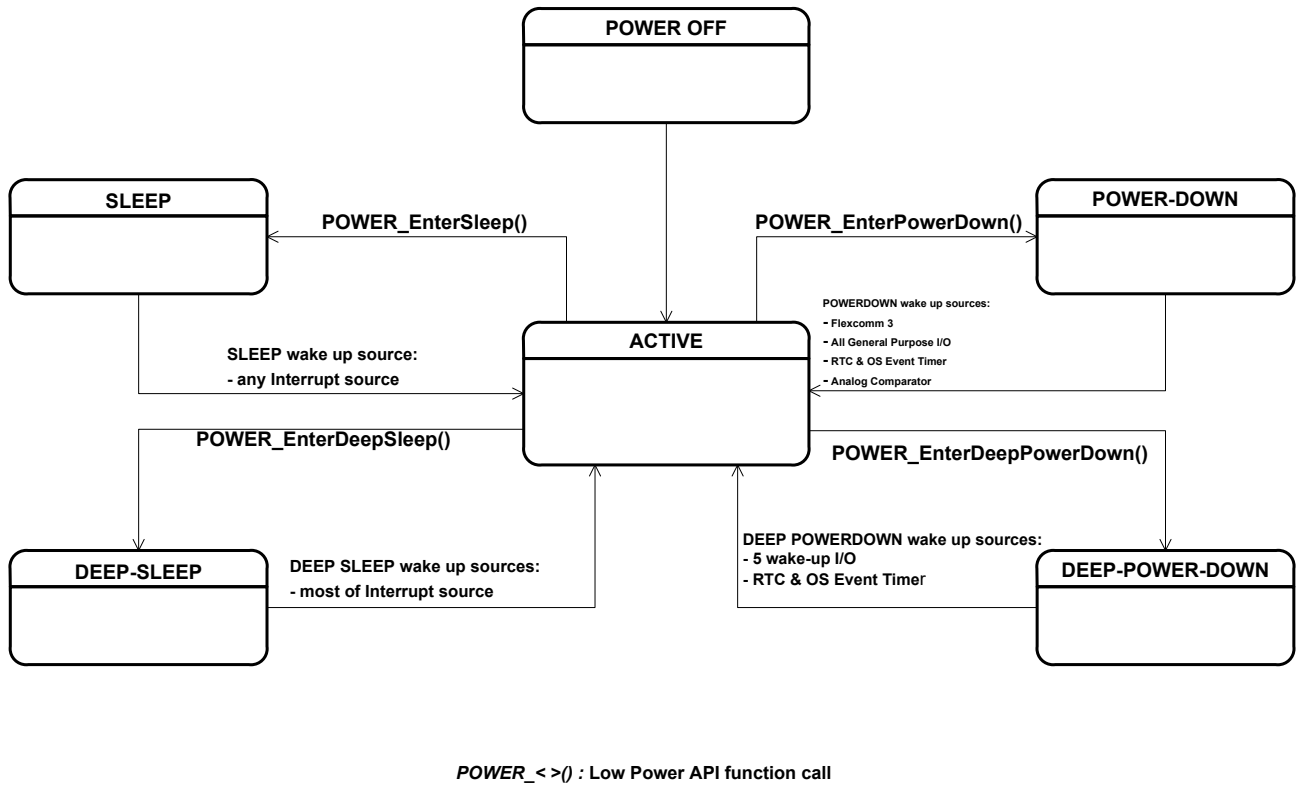


Figure 129. Power Modes

After any chip reset, whatever the reset cause, the device is in ACTIVE mode.

In addition to the ACTIVE mode, there are four special power reduction modes with different peripherals running:

- SLEEP mode,
- DEEP-SLEEP mode,
- POWER-DOWN mode,
- DEEP-POWER-DOWN mode.

Each of these four power reduction mode is activated via a set of power functions, which are all fully described in see :

- POWER_EnterSleep(...), to enter in SLEEP low power mode,
- POWER_EnterDeepSleep(...), to enter in DEEP-SLEEP low power mode,
- POWER_EnterPowerDown(...), to enter in POWER-DOWN low power mode,
- POWER_EnterDeepPowerDown(...), to enter in DEEP-POWER-DOWN low power mode.

NOTE

ARM CORTEX-M33 debug (via the SWD interface) is not possible during all low power modes (SLEEP, DEEP-SLEEP, POWER-DOWN and DEEP-POWER-DOWN).

NOTE

It is not possible to go directly from one low power mode (SLEEP, DEEP-SLEEP, POWER-DOWN or DEEP-POWER-DOWN) to another low power mode (for example from SLEEP to POWER-DOWN): the ACTIVE state must be entered first.

Table 243 summarizes the power state of the different power domains according to the power modes.

Table 243. Power Modes vs Power Domains

	PD_CORE	PD_SYSTEM	PD_AON	PD_MEM_0	PD_MEM_1	PD_MEM_2
ACTIVE	ON	ON	ON	ON	ON	ON
SLEEP	ON	ON	ON	ON	ON	ON
DEEP-SLEEP	ON	ON	ON	Configurable[1]	Configurable[2]	Configurable[3]
POWER-DOWN	OFF	ON	ON	Configurable	Configurable	Configurable
DEEP-POWER-DOWN	OFF	OFF	ON	Configurable	Configurable	Configurable

1. Depending on whether RAM_01 Deep Sleep power mode (data retention) is required by the user via the relevent Power API function, this power domain will be ON or OFF.
2. Depending on whether Deep Sleep power mode (data retention) is required for one SRAM instance in this power domain by the user via the relevent Power API function, this power domain will be ON or OFF.
3. Depending on whether RAM_00 Deep Sleep power mode (data retention) is required by the user via the relevent Power API function, this power domain will be ON or OFF.

Table 244. Peripherals reduced power modes

Peripherals		Device Low Power Modes			
Name	Description	SLEEP	DEEP-SLEEP	POWER-DOWN	DEEP-POWER-DOWN
DCDC	Bulk DC-DC Converter	Same state as in ACTIVE	OFF	OFF	OFF
BIAS	Analog references	ON	ON	Software configured	OFF
BODCORE	Core Logic supply Brownout Detector	Same state as in ACTIVE	Software configured	OFF	OFF
BODVDDMAIN	VDD_MAIN supply Brownout Detector	Same state as in ACTIVE	Software configured	OFF	OFF
FRO1M	1-MHz Free Running Oscillator	ON	Software configured	Software configured	Software configured
FRO192M	High Speed Free Running Oscillator	ON	Software configured	OFF	OFF
FRO32K	32-KHz Free Running Oscillator	Same state as in ACTIVE	Software configured	Software configured	Software configured
XTAL32K	32-KHz Crystal Oscillator	Same state as in ACTIVE	Software configured	Software configured	Software configured
XTALHF	High Frequency Crystal Oscillator	Same state as in ACTIVE	Software configured	OFF	OFF
PLL0	Phase-Locked Loop module 0	Same state as in ACTIVE	Software configured	OFF	OFF

Table continues on the next page...

Table 244. Peripherals reduced power modes (continued)

Peripherals		Device Low Power Modes			
Name	Description	SLEEP	DEEP-SLEEP	POWER-DOWN	DEEP-POWER-DOWN
PLL1	Phase-Locked Loop module 1	Same state as in ACTIVE	Software configured	OFF	OFF
USBFSPHY	USB Full Speed Physical interface	Same state as in ACTIVE	Software configured	OFF	OFF
COMP	Analog Comparator	Same state as in ACTIVE	Software configured	Software configured	OFF
LDOEFUSEPROG	OTP-eFUSE Programming Low Drop-Out Regulator	Same state as in ACTIVE	Software configured	OFF	OFF
LDOXTALHF	High Frequency Crystal Oscillator Low Drop-Out Regulator	Same state as in ACTIVE	Software configured	OFF	OFF
LDOFLASHNV	Non Volatile FLash Macro Low Drop-Out Regulator	ON	OFF	OFF	OFF
PLL0_SSCG	PLL0 Spread Spectrum Clock Generator	Same state as in ACTIVE	Software configured	OFF	OFF
HSCMP0	High Speed Comparator 0	Same state as in ACTIVE	Software configured	OFF	OFF
HSCMP1	High Speed Comparator 1	Same state as in ACTIVE	Software configured	OFF	OFF
HSCMP2	High Speed Comparator 2	Same state as in ACTIVE	Software configured	OFF	OFF
OPAMP0	Operational Amplifier 0	Same state as in ACTIVE	Software configured	OFF	OFF
OPAMP1	Operational Amplifier 1	Same state as in ACTIVE	Software configured	OFF	OFF
OPAMP2	Operational Amplifier 2	Same state as in ACTIVE	Software configured	OFF	OFF
VREF	ADCs/DACs Analog Reference module	Same state as in ACTIVE	Software configured	Software configured	OFF
CMPBIAS	High Speed Comparators Biasing	Same state as in ACTIVE	Software configured	OFF	OFF

Table continues on the next page...

Table 244. Peripherals reduced power modes (continued)

Peripherals		Device Low Power Modes			
Name	Description	SLEEP	DEEP-SLEEP	POWER-DOWN	DEEP-POWER-DOWN
HSCMP0_DAC	HSCMP0 (internal) DAC	Same state as in ACTIVE	Software configured	OFF	OFF
HSCMP1_DAC	HSCMP1 (internal) DAC	Same state as in ACTIVE	Software configured	OFF	OFF
HSCMP2_DAC	HSCMP2 (internal) DAC	Same state as in ACTIVE	Software configured	OFF	OFF
DAC0	Digital to Analog Converter 0	Same state as in ACTIVE	Software configured	OFF	OFF
DAC1	Digital to Analog Converter 1	Same state as in ACTIVE	Software configured	OFF	OFF
DAC2	Digital to Analog Converter 2	Same state as in ACTIVE	Software configured	OFF	OFF
STOP_DAC0	DAC0 Stop Mode	Same state as in ACTIVE	Software configured	OFF	OFF
STOP_DAC1	DAC1 Stop Mode	Same state as in ACTIVE	Software configured	OFF	OFF
STOP_DAC2	DAC2 Stop Mode	Same state as in ACTIVE	Software configured	OFF	OFF

29.3.2.2.1 ACTIVE Mode

The device is in ACTIVE mode after any chip level reset and the default dynamic power consumption is determined by the reset values of PDRUNCFG0/1, SYSCON registers AHBCTRLCLK0/1/2/3 and AUTOCLKGATEOVERRIDE.

In ACTIVE mode, by default, the core logic supply source is the Low Drop-Out Regulator (LDO_CORE). The user can switch to the DC-DC Converter - to gain better dynamic power efficiency - via a set of functions provided in the power library: POWER_SetCorePowerSource(...) and POWER_GetCorePowerSource(...). See power API for more details.

In ACTIVE mode, it is possible to individually control all SRAM instances power mode via the function POWER_SRAMPowerModeControl(...) provided in the power library. See SRAM Power Down Modes for more details.

In ACTIVE mode, when the device is in operation, it is possible to reduce the power consumption of the various peripherals which are not used (temporarily or permanently) by :

- switching off their functional clock; see SYSCON registers AHBCTRLCLK0/1/2/3.
- activating the so-called "automatic clock gating" feature; see SYSCON register AUTOCLKGATEOVERRIDE
- switching off the entire module when possible: see registers PDRUNCFG0 and PDRUNCFG1.

29.3.2.2.1.1 Power Configuration in ACTIVE mode

Dynamic Power consumption in active mode is determined by the following configuration choices:

- The AHBCLKCTRL0/1/2/3, AHBCLKCTRLSET0/1/2/3 and AHBCLKCTRLCLR0/1/2/3 registers control which memories and peripherals are enabled. See SYSCON chapter for more details . To save power, functions that are not needed by the

application should be turned off. If specific times are known when certain functions will not be needed, they can be turned off temporarily and turned back on again as needed.

- The power to various analog blocks (PLLs, oscillators, ADCs, DACs High Speed Comparators, ...) can be controlled individually through the PDRUNCFG0/1, PDRUNCFGSET0/1, PDRUNCFGCLR0/1 register, see . As with clock controls, these blocks should generally be turned off if not needed by the application. If turned off, time will be needed before these blocks can be used again after being turned on.
- The clock source for the AHB System Bus clock and the CPU clock can be selected from the 192-MHz FRO (either 12 MHz or 96 MHz), the 32-KHz FRO, the 1-MHz FRO, the PLL0/1, the High Speed Crystal oscillator or an external clock, see Figure 4 and related registers.
- If the 96-MHz output of the 192-MHz FRO is not needed, it should be turned off (see FRO192M_CTRL[30] in Analog Control module).
- Generally speaking, everything uses less power at lower frequencies, so running the CPU and other device features at a frequency sufficient for the application (plus some margin) will save power. If the PLL is not needed, it should be turned off to save power. Also, running the PLL at a lower CCO frequency saves power.
- Several peripherals use individual peripheral clocks with their own clock dividers. The peripheral clocks can be shut down through the corresponding clock divider registers if the base clock is still needed for another function.
- The power library provides an easy way to optimize power consumption depending on CPU load and performance requirements, especially the function POWER_SetVoltageForFrequency(...). See power API.

29.3.2.2.2 SLEEP Mode

SLEEP mode saves some power by stopping Cortex-M33 CPU execution without affecting peripherals or requiring significant wake-up time. The clock to the CPU is shut off. Peripherals and memories are active and operational. CPU is stopped and execution of instructions is suspended until either a reset or an interrupt occurs.

Peripheral functions, if selected to be clocked in the AHBCLKCTRL registers, continue operation during SLEEP mode and may generate interrupts to cause the processor to resume execution. SLEEP mode only eliminates dynamic power used by the processor itself, memory systems and related controllers, and internal buses. The processor state and registers, peripheral registers, and internal SRAM values are maintained.

29.3.2.2.2.1 Power configuration in SLEEP mode

Power consumption in SLEEP mode is configured by the same settings as in ACTIVE mode:

- Enabled clocks remain running.
- The system clock frequency remains the same as in ACTIVE mode, but the CPU is not clocked.
- Analog and digital peripherals are powered and selected as in ACTIVE mode through the PDRUNCFG0/1, AHBCLKCTRL0/1/2/3 registers.

29.3.2.2.2.2 Programming SLEEP mode

The following steps must be performed to enter SLEEP mode:

1. In the NVIC, enable all interrupts that are needed to wake-up the CPU
2. Alter PDRUNCFG0 and PDRUNCFG1 if needed to reflect any functions that should be ON or OFF during SLEEP.
3. Call the power API POWER_EnterSleep(). See

29.3.2.2.2.3 Wake up from SLEEP mode

SLEEP mode is exited automatically when an interrupt enabled in the NVIC arrives at the processor or a reset occurs. After a wake-up caused by an interrupt, the device returns to its original power configuration defined by the contents of the PDRUNCFG0/

PDRUNCFG1 and the AHBCLKCTRL registers. If a reset occurs, the device enters the default configuration in ACTIVE mode after the reset cycle.

29.3.2.2.3 DEEP-SLEEP Mode

In DEEP-SLEEP mode, the full device remained powered, but flash and ROM are shut down, with the cost of a longer wake-up time compared to the SLEEP mode. The system clock to the CPU is disabled as in sleep-mode. Analog blocks are powered down by default but can be selected to keep running through the power API if needed as wake-up sources. The main clock and all peripheral clocks are disabled.

DEEP-SLEEP mode eliminates power used by analog peripherals and all dynamic power used by the CPU, its memory systems and related controllers, and internal buses. The CPU state and registers, peripheral registers, and internal SRAM values are maintained, and the GPIO logic levels of the pins remain static. All SRAM instances that are not configured to enter in SRAM Deep Sleep low power mode will keep the state they had before entering DEEP-SLEEP, meaning that if a SRAM was in Active state before entering in DEEP-SLEEP, it will stay in Active state during DEEP-SLEEP and therefore it will consume more power.

Selected peripherals such as GPIO group interrupts (GINT), USB Full Speed, Flexcomms (SPI, I2C, USART), Windowed-Watchdog Timer, RTC, standard Counter/Timers, Micro-tick, RTC alarm, a watchdog timer interrupt/reset, BOD interrupt/reset and comparator can be left running in DEEP-SLEEP mode. The oscillators like the FRO1M, FRO32K, XTAL32K but also the FRO192M (which delivers the 12-MHz and 96 MHz clocks), can also be left running.

Some peripherals can have DMA service during DEEP-SLEEP mode without waking up entire device.

29.3.2.2.3.1 Power configuration in DEEP-SLEEP mode

Power consumption in DEEP-SLEEP mode is determined primarily by which analog wake-up sources remain enabled (see Table 244). Serial peripherals and pin interrupts configured to wake-up the device, contribute to the power consumption only to the extent that they are clocked by external sources. All wake-up events (other than reset) must be enabled through the POWER_EnterDeepSleep() power API. In addition, any related analog block, for example the RTC clock sources (FRO32K, XTAL32K) must be explicitly enabled through the POWER_EnterDeepSleep() power API. See

29.3.2.2.3.2 Programming DEEP-SLEEP mode

The following step must be performed to enter DEEP-SLEEP mode:

1. Call the POWER_EnterDeepSleep() power API with the peripheral parameter to enable the analog peripherals and wake-up sources/events, See

29.3.2.2.3.3 Wake-up from DEEP-SLEEP mode

The device can wake-up from DEEP-SLEEP mode in the following ways:

- Using a signal on one of the eight pin interrupts selected (PINT). See Pin Interrupt and Pattern Match (PINT).
- Using an interrupt from a block such as the Windowed-Watchdog Timer, Standard Counter/Timers (CTIMER).
- Using the RESET pin
- Using a wake-up signal from any of the serial peripherals that are operating in DEEP-SLEEP mode.
- GPIO group interrupt signal (GINT0 & GINT1).
- RTC alarm. See Real Time Clock (RTC).
- OS Event Timer. See OS Event Timer.

For all (except for the RESET pin), interrupts must also be enabled via the POWER_EnterDeepSleep() power API.

29.3.2.2.4 POWER-DOWN Mode

POWER-DOWN mode turns off nearly all on-chip power consumption by eliminating power used by almost all analog modules and by shutting down the DCDC and the LDO_CORE, thus eliminating almost all digital peripherals power, with the cost of a longer wake-up time compared to DEEP-SLEEP mode. FRO192M is disabled. The flash memory is also disabled. The clock to CPU and peripherals is shut down and, if not configured, the peripherals in power domains PD_SYSTEM and PD_AO receive no internal clocks. All SRAM can be configured to maintain their internal state and all registers lose their internal states except those located in the power domains PD_SYSTEM and PD_AO. Any SRAM instance that is not configured to maintain its internal state will lose it.The Cortex-M33 CPU state and some critical peripherals like the Analog Controller are retained. The GPIO logic level does not remain static in power-down mode. All GPIO pin state will be logic '0' in power-down mode. All IOCON registers and peripheral registers related ONLY to Flexcomm3 (SPI,I2C, I2S, USART), GINTO0, RTC, OS Event timer and analog comparator will maintain state in power-down mode.. GPIO group interrupts (GINT0 & GINT1), selected serial peripherals in Flexcomm3 (SPI, I2C, USART), RTC, OS Event Timers and Analog Comparator can be left running to wake up the device.

When a wake-up event occurs, the Cortex-M33 CPU code execution will resume from where it has stopped. It is the responsibility of the customer application to re-configure all modules in the power domain core PD_CORE (whose states have not been retained, for example all Flexcomm -except Flexcomm3-, SDMA, Power Quad, DMIC, ...).

29.3.2.2.4.1 Power configuration in POWER-DOWN mode

Power consumption in POWER-DOWN mode is determined primarily by the number of SRAM instances which remain enabled in SRAM Deep Sleep power mode (retention mode). Serial peripherals in Flexcomm3 and pin interrupts configured to wake-up contribute to the dynamic power consumption only to the extent that they are clocked by external sources. All wake-up events (other than RESET pin) must be enabled via the POWER_EnterPowerDown() power API. In addition, any analog block (the Analog Comparator, FRO32K, XTAL32K, FRO1M) must be explicitly enabled through the POWER_EnterPowerDown() power API function. See .

29.3.2.2.4.2 Programming POWER-DOWN mode

The following steps must be performed to enter POWER-DOWN mode:

1. Call the POWER_EnterPowerDown() power API with the peripheral parameter to enable the analog peripherals and select the wake-up sources/events. See

29.3.2.2.4.3 Wake-up from POWER-DOWN mode

The device can wake-up from POWER-DOWN mode in the following ways:

- Using a reset from the RESET pin.
- Using a wake-up signal from any of the serial peripherals in Flexcomm3.
- Using the analog comparator.
- GPIO group interrupt signals (GINT0 & GINT1).
- RTC alarm. See Real Time Clock (RTC).
- OS Event Timer. See OS Event Timer.

For all (except for the RESET pin), also enable the wake-up sources via the POWER_EnterPowerDown() power API.

29.3.2.2.5 DEEP-POWER-DOWN Mode

DEEP-POWER-DOWN mode shuts down virtually all on-chip power consumption but requires a significantly longer wake-up time (compared to POWER-DOWN mode). For maximal power savings, the entire power domains PD_CORE (CPU, DSP, Flexcomms, SDMA ...) and PD_SYSTEM (Reset, Clocks, SYSCON, IOCON, ...) are shut down. Only the Always-on power domain PD_AO (PMU, PMC, the RTC and the OS Event Timer) stays powered. Clocks are shut off to the entire chip device with the exception of the RTC and the OS Event Timer if they are needed.On wake-up, the device reboots.

During DEEP-POWER-DOWN mode, the contents of some SRAM can be retained (software configured via the POWER_EnterDeepPowerDown() low power API) but registers (other than those in the PMC, the RTC and OS Event Timer) are not retained. All functional pins are tri-stated in DEEP-POWER-DOWN mode, except the 5 wake-up pins and the RESET pin.

29.3.2.2.5.1 Power configuration in DEEP-POWER-DOWN mode

DEEP-POWER-DOWN mode has the following configuration options (via the POWER_EnterDeepPowerDown() low power API):

- Some SRAM instances to be retained.
- Wake-up pins.
- Clock sources for RTC (FRO32K and XTAL32K) and OS Event Timer (FRO32K, XTAL32K and FRO1M).

29.3.2.2.5.2 Wake-up from DEEP-POWER-DOWN mode

Wake-up from DEEP-POWER-DOWN can be accomplished via:

- RESET pin
- RTC
- OS Event Timer
- Five wake-up pins

29.3.2.2.5.3 Programming DEEP-POWER-DOWN mode using the RTC for wake-up

The following steps must be performed to enter DEEP-POWER-DOWN mode when using the RTC for waking up:

1. Set up the RTC alarm.
2. Call the POWER_EnterDeepPowerDown() power API function.

29.3.2.2.5.4 Programming DEEP-POWER-DOWN mode using the OS Event Timer for wake-up

The following steps must be performed to enter DEEP-POWER-DOWN mode when using the OS Event Timer for waking up:

1. Configure the OS Event Timer clock source in PMC OSTIMER register.
2. Configure OS Event Timer (OSTIMER MATCHN, EVENT_CTRL …).
3. Call the POWER_EnterDeepPowerDown() power API function. See

29.3.2.2.5.5 Programming DEEP-POWER-DOWN mode using the wake-up pins for wake-up

The following steps must be performed to enter DEEP-POWER-DOWN mode when using the wake-up pins for waking up:

1. Call the POWER_EnterDeepPowerDown() power API function. See

29.3.2.2.5.6 Wake-up from DEEP-POWER-DOWN mode

The device goes through the entire reset process when a DEEP-POWER-DOWN wake-up event occurs:

- The PMU/PMC will turn on the device on-chip voltage regulators. When the core voltage reaches the required level, a system reset will be triggered and the Cortex-M33 CPU boots.
- All device registers will be reset, except some key status registers inside the PMC like the device reset cause ...

29.3.2.2.6 SRAM Power Down Modes

After any chip level reset, all SRAM instances are in Active mode. In Active mode, Read and Write operations on the SRAM are allowed.

Each SRAM instance supports two power down modes:

- Deep sleep mode,
- Shutdown mode.

To configure any of the SRAM instances in one of the two power down mode or to exit any SRAM instance from any power down modes, use the function POWER_SRAMPowerModeControl(...) provided in the Power API library. See Power API for more details.

NOTE

The SRAM instances power down modes (Deep Sleep and Shutdown) shall not be confused with the chip low power modes (SLEEP, DEEP-SLEEP, POWER-DOWN and DEEP-POWER-DOWN).

29.3.2.2.6.1 SRAM Deep Sleep mode

Deep sleep mode can be used to reduce the standby supply current (leakage) of the SRAM instance without losing the content.

When a SRAM instance is in Deep Sleep power down mode, it is not possible to access (Read or Write) its content.

NOTE

Once a SRAM instance has been set in Deep Sleep power down mode, it can only be transitioned to Active mode: it is not possible to go directly from Deep Sleep mode to Shutdown mode.

29.3.2.2.6.2 SRAM Shutdown mode

Shutdown mode can be used to reduce the standby supply current (leakage) of the SRAM instance. The content of the memory is lost in this mode.

When a SRAM instance is in Shutdown power down mode, it is not possible to access (Read or Write) its content.

Compared Deep Sleep mode, Shutdown allows further standby current (leakage) reduction; however, unlike Deep Sleep mode, the content of the memory is lost in Shutdown mode. Exiting a SRAM from Shutdown mode is as long as exiting from Deep Sleep mode (about 7 µs).

NOTE

Once a SRAM instance has been set in Shutdown power down mode, it can only be transitioned to Active mode: it is not possible to go directly from Shutdown mode to Deep Sleep mode.

29.3.2.2.6.3 SRAM Power down modes versus Chip level Power modes

The SRAM instances power down modes (Deep Sleep and Shutdown) shall not be confused with the chip low power modes (SLEEP, DEEP-SLEEP, POWER-DOWN and DEEP-POWER-DOWN).

In DEEP-SLEEP mode, each SRAM instance can be in one of the three following SRAM power modes (depending on the value of the parameter "sram_retention_ctrl" passed to POWER_EnterDeepSleep(...) API. See power API for more details):

- Active (Memory content retained, Read/Write possible)
- Deep Sleep (Memory content retained, Read/Write not possible)
- Shutdown (Memory content not retained, Read/Write not possible)

SRAM instances that are not required to be put in retention mode during DEEP-SLEEP (as specified by the parameter "sram_retention_ctrl") will keep the state they had before calling POWER_EnterDeepSleep(...). SRAM instances that must be put in retention mode during DEEP-SLEEP (as specified by the parameter "sram_retention_ctrl") will be put in SRAM Deep Sleep power down mode, and, when waking up from DEEP-SLEEP, these SRAM instances will be put in SRAM Active mode.

In POWER-DOWN and DEEP-POWER-DOWN mode, each SRAM instance can in one of the two following SRAM power down modes (depending on the value of the parameter "sram_retention_ctrl" passed to POWER_EnterPowerDown(...) and POWER_EnterDeepPowerDown(...) API. See power API for more details):

- Deep Sleep (Memory content retained)
- Shutdown (Memory content not retained)

NOTE

To minimize leakage current during POWER-DOWN and DEEP-POWER-DOWN modes, depending on the amount of data retention required, it is recommended to use the Deep Sleep mode of SRAM instances in this order of preference:

1. RAM_01
2. RAM_00
3. All others SRAM instances, knowing that the bigger the SRAM instance, the higher the standby leakage current will be.

NOTE

During DEEP-POWER-DOWN with VDD_MAIN power supply swithed off, only RAM_00 instance can be put in SRAM Deep Sleep power down mode. All other SRAM instances are in SRAM Shutdown power down mode (memory content not retained).

29.4 Memory Map and register definition

This section includes the PMC module memory map and detailed descriptions of all registers.

29.4.1 PMC register descriptions

29.4.1.1 PMC memory map

PMC base address: 4002_0000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Power Management Control (CTRL)	32	See section	See section
4	PMC status (STATUS)	32	See section	See section
8	Reset Control (RESETCTRL)	32	See section	See section
C	Reset Cause (RESETCAUSE)	32	See section	See section
10	DCDC (first) control (DCDC0)	32	See section	See section
14	DCDC (second) control register (DCDC1)	32	RW	0180_3A98
1C	LDO control (LDOPMU)	32	See section	See section

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
20	Memory LDO control (LDOMEM)	32	See section	See section
24	LDO CORE control 0 (LDOCORE0)	32	See section	See section
28	LDO Flash High Voltage control (LDOFLASHNV)	32	See section	See section
2C	OTP-eFUSE (One Time Programmable Memory) Programming LDO control (LDOEFUSEPROG)	32	See section	See section
30	VDDMAIN Brown Out Dectector control (BODVDDMAIN)	32	See section	See section
38	Digital Core logic Brown Out Dectector control (BODCORE)	32	See section	See section
40	Analog References fast wake-up Control (REFFASTWKUP)	32	See section	See section
4C	32 KHz Crystal oscillator (XTAL) control (XTAL32K)	32	See section	See section
50	Analog Comparator control (COMP)	32	See section	See section
60	DCDC and LDOCORE power state (enable/disable) control. (CMD)	32	WO	0000_0000
64	Wake-up IO control (WAKEUPIOCTRL)	32	See section	See section
68	Wake-up I/O source (WAKEIOCAUSE)	32	See section	See section
6C	Life Cycle State (LIFECYCLESTATE)	32	See section	See section
70	Power status (STATUSPWR)	32	See section	See section
74	Clock status (STATUSCLK)	32	See section	See section
80	Always-on 0 (AOREG0)	32	See section	0000_0000
84	Always-on 1 (AOREG1)	32	See section	0000_0000
90	Miscellaneous control (MISCCTRL)	32	See section	See section
98	32 KHz clocks source control (RTCOSC32K)	32	See section	See section

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
9C	OS Event Timer control (OSEVENTTIMER)	32	See section	See section
A0	Power down sleep configuration 1 (PDSLEEPCFG1)	32	See section	See section
A4	Time-out events (TIMEOUTEVENTS)	32	See section	See section
B0	Power down sleep configuration 0 (PDSLEEPCFG0)	32	RW	See section
B4	SRAM control (SRAMRETCTRL)	32	RW	See section
B8	Power down run configuration 0 (PDRUNCFG0)	32	RW	See section
BC	Power down run configuration 1 (PDRUNCFG1)	32	See section	See section
C0	Power down run configuration set 0 (PDRUNCFGSET0)	32	WO	0000_0000
C4	Power down run configuration set 1 (PDRUNCFGSET1)	32	WO	See section
C8	Power down run configuration clear 0 (PDRUNCFGCLR0)	32	WO	0000_0000
CC	Power down run configuration clear 1 (PDRUNCFGCLR1)	32	WO	See section
D4	SRAM control (SRAMCTRL)	32	See section	See section
D8	SRAM control 0 (SRAMCTRL0)	32	RW	6666_6666
DC	SRAM control 1 (SRAMCTRL1)	32	See section	See section

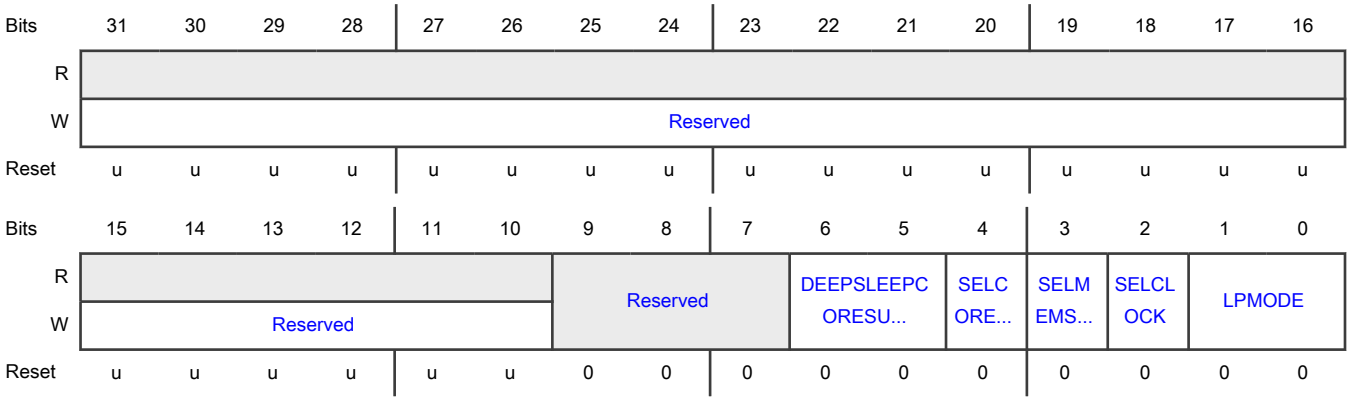
29.4.1.1.1 Power Management Control (CTRL)

Reset by: PoR, Pin Reset, Software Reset and BoDs reset.

Offset

Register	Offset
CTRL	0h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9-7 —	Reserved
6-5 DEEPSLEEPCORESUPPLY	Select Core Logic supply source during DEEP-SLEEP low power mode : 00 - LDO CORE in Low Power Mode. 01 - LDO CORE in High Power Mode. 10 - DCDC Converter.
4 SELCORESUPPLYWK	Select Core Logic supply source when waking up from DEEP-SLEEP and POWER-DOWN low power modes : 0 - Core Logic is supplied by DCDC Converter. 1 - Core Logic is supplied by LDO CORE (configured in High Power mode).
3 SELMEMSUPPLY	Select Memories supply source in DEEP-SLEEP low power mode: Note: in POWER-DOWN and DEEP-POWER-DOWN, memories are always supplied by LDO_MEM. 0 - Memories are supplied by LDO_MEM in 'DEEP-SLEEP' low power mode. 1 - Memories are supplied by DCDC/LDO_CORE in 'DEEP-SLEEP' low power mode.
2 SELCLOCK	Select the Power Management Controller (PMC) functional clock : 0 - 1 MHz Free Running Oscillator. 1 - 12 MHz Free Running Oscillator.
1-0 LPMODE	Power Mode Control. 00 - ACTIVE power mode. 01 - DEEP-SLEEP low power mode. 10 - POWER-DOWN low power mode.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	11 - DEEP-POWER-DOWN low power mode.

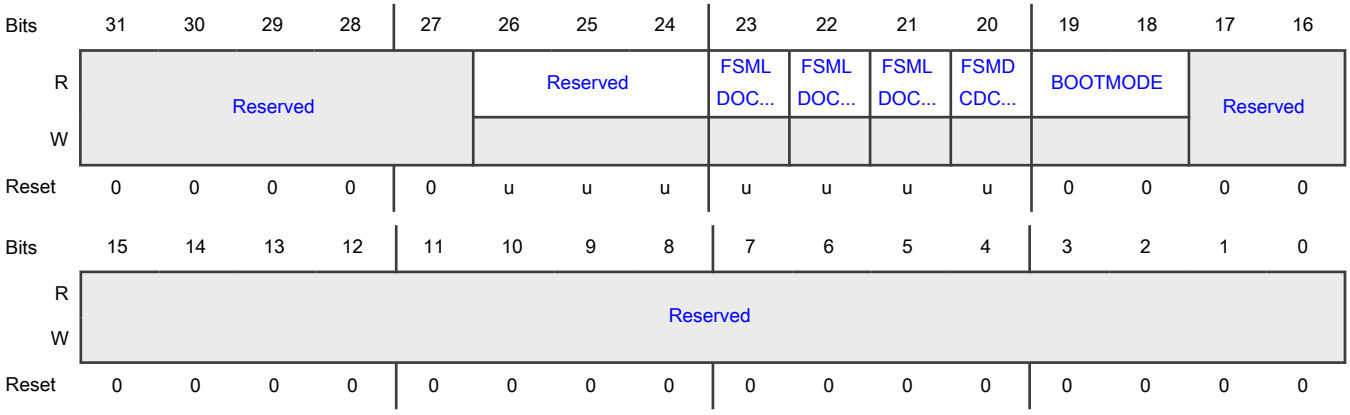
29.4.1.1.2 PMC status (STATUS)

This register controls Power Management Controller FSM (Finite State Machines) status.

Offset

Register	Offset
STATUS	4h

Diagram



Fields

Field	Description
31-27 —	Reserved
26-24 —	Reserved Read value is undefined, only zero should be written.
23 FSMLDOCOREEXPTMRENABLE	Indicates the status of the LDO CORE Exponential Timer (enabled or disabled) as driven by the Hardware Finite State Machines (FSM).
22 FSMLDOCORELPENABLE	Indicates the power status of the LDO CORE Low Power Mode (enabled or disabled) as driven by the Hardware Finite State Machines (FSM). 0 - LDO CORE Low Power Mode is currently disabled by the Hardware Finite State Machine (FSM).

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - LDO CORE Low Power Mode is currently enabled by the Hardware Finite State Machine (FSM).
21 FSMLDOCOREHPENABLE	Indicates the power status of the LDO CORE High Power Mode (enabled or disabled) as driven by the Hardware Finite State Machines (FSM). 0 - LDO CORE High Power Mode is currently disabled by the Hardware Finite State Machine (FSM). 1 - LDO CORE High Power Mode is currently enabled by the Hardware Finite State Machine (FSM).
20 FSMDCDCENABLE	Indicates the power status of the DCDC (enabled or disabled) as driven by the Hardware Finite State Machines (FSM). 0 - DCDC is currently disabled by the Hardware Finite State Machine (FSM). 1 - DCDC is currently enabled by the Hardware Finite State Machine (FSM).
19-18 BOOTMODE	Latest IC Boot cause:. 00 - Latest IC boot was a Full power cycle boot sequence (PoR, Pin Reset, Brown Out Detectors Reset, Software Reset). 01 - Latest IC boot was from DEEP-SLEEP low power mode. 10 - Latest IC boot was from POWER-DOWN low power mode. 11 - Latest IC boot was from DEEP-POWER-DOWN low power mode.
17-0 —	Reserved

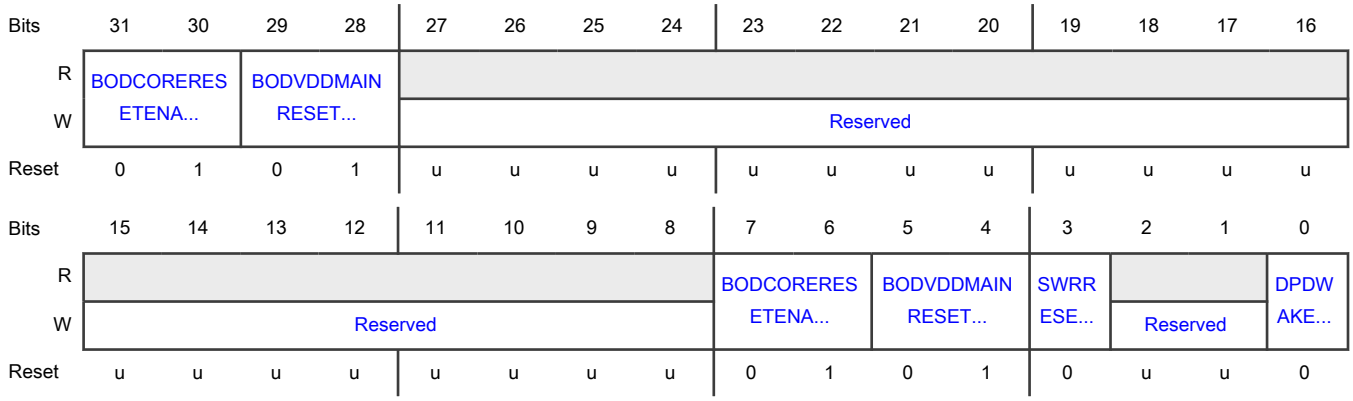
29.4.1.1.3 Reset Control (RESETCTRL)

Reset by PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset.

Offset

Register	Offset
RESETCTRL	8h

Diagram



Fields

Field	Description
31-30 BODCORERESETENA_SECURE_DP	BOD_CORE reset enable. 01 - And any other value than b10: BOD_CORE reset is enabled. 10 - BOD_CORE reset is disabled.
29-28 BODVDDMAINRESETENA_SECURE_DP	BOD_VDDMAIN reset enabled. 01 - And any other value than b10: BOD_VDDMAIN reset is enabled. 10 - BOD_VDDMAIN reset is disabled.
27-8 —	Reserved Read value is undefined, only zero should be written.
7-6 BODCORERESETENA_SECURE	BOD_CORE reset enabled. 01 - And any other value than b10: BOD_CORE reset is enabled. 10 - BODCORE reset is disabled.
5-4 BODVDDMAINRESETENA_SECURE	BOD_VDDMAIN reset enabled. 01 - And any other value than b10: BOD_VDDMAIN reset is enabled. 10 - BOD_VDDMAIN reset is disabled.
3 SWRRESETENABLE	Software reset enable. 0 - Software reset is disabled. 1 - Software reset is enabled.
2-1 —	Reserved Read value is undefined, only zero should be written.
0 DPDWAKEUPRESETENABLE	Wake-up from DEEP-POWER-DOWN reset event (either from wake up I/O or RTC or OS Event Timer). 0 - Reset event from DEEP-POWER-DOWN mode is disabled. 1 - Reset event from DEEP-POWER-DOWN mode is enabled.

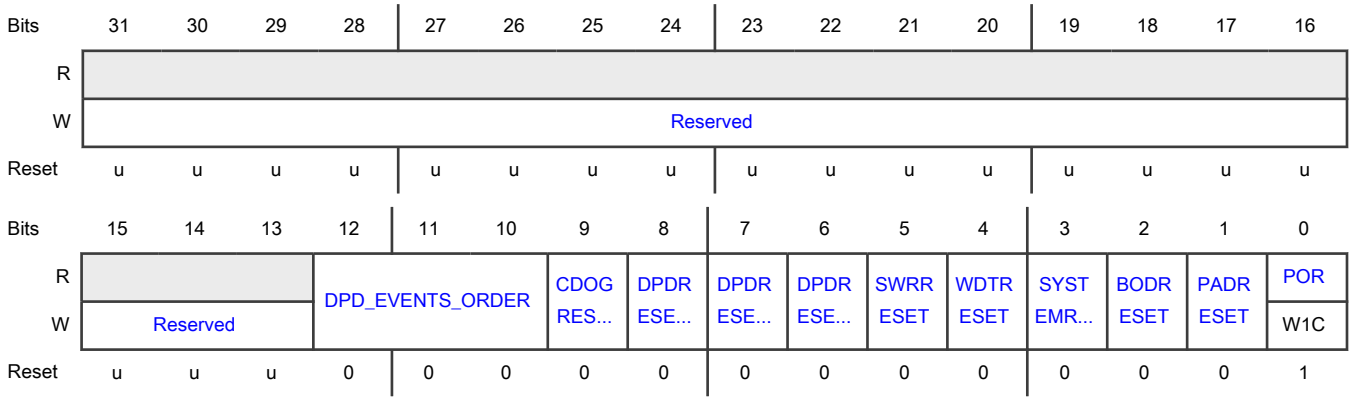
29.4.1.1.4 Reset Cause (RESETCAUSE)

Reset by PoR.

Offset

Register	Offset
RESETCAUSE	Ch

Diagram



Fields

Field	Description
31-13 —	Reserved Read value is undefined, only zero should be written.
12-10 DPD_EVENTS_ORDER	In DEEP-POWER-DOWN mode, indicates which reset event occured first between DPDRESET_WAKEUPIO, DPDRESET_RTC and DPDRESET_OSTIMER. Write 'b001' to clear these bit field. Indicates which reset event occured first, between a wake up I/O event (in DEEP-POWER-DOWN), a RTC event (in DEEP-POWER-DOWN) and a OS Timer event (in DEEP-POWER-DOWN). May be usefull when several reset events are enabled during DEEP-POWER-DOWN. 000 - No event 001 - WAKEUPIO 010 - RTC 011 - Both WAKEUPIO and RTC events occured at the same time (less than 1 nano-second from each other) 100 - OSTIMER 101 - Both WAKEUPIO and OSTIMER events occured at the same time (less than 1 nano-second from each other) 110 - Both RTC and OSTIMER events occured at the same time (less than 1 nano-second from each other) 111 - WAKEUPIO, RTC and OSTIMER events occured at the same time (less than 1 nano-second from each other)
9 CDOGRESET	1 : The last chip reset was caused by the code Watchdog. Write '1' to clear this bit.
8 DPDRESET_OSTIMER	1 : A OS Event Timer reset event occured during DEEP-POWER-DOWN mode. Write '1' to clear this bit. See the related field DPD_EVENTS_ORDER in this register

Table continues on the next page...

Table continued from the previous page...

Field	Description
7 DPDRESET_RTC	1 : A RTC (either RTC Alarm or RTC wake up) reset event occured during DEEP-POWER-DOWN mode. Write '1' to clear this bit. See the related field DPD_EVENTS_ORDER in this register
6 DPDRESET_WAKEUPIO	1 : A Wake-up I/O reset event occured during DEEP-POWER-DOWN mode. Write '1' to clear this bit. See the related field DPD_EVENTS_ORDER in this register
5 SWRRESET	1 : The last chip reset was caused by a Software. Write '1' to clear this bit.
4 WDTRESET	1 : The last chip reset was caused by the Watchdog Timer. Write '1' to clear this bit.
3 SYSTEMRESET	1 : The last chip reset was caused by a System Reset requested by the ARM CPU. Write '1' to clear this bit.
2 BODRESET	1 : The last chip reset was caused by a Brown Out Detector (BoD), either BOD_VDDMAIN or BOD_CORE. Write '1' to clear this bit.
1 PADRESET	1 : The last chip reset was caused by a Pin Reset. Write '1' to clear this bit.
0 POR	1 : The last chip reset was caused by a Power On Reset. Write '1' to clear this bit.

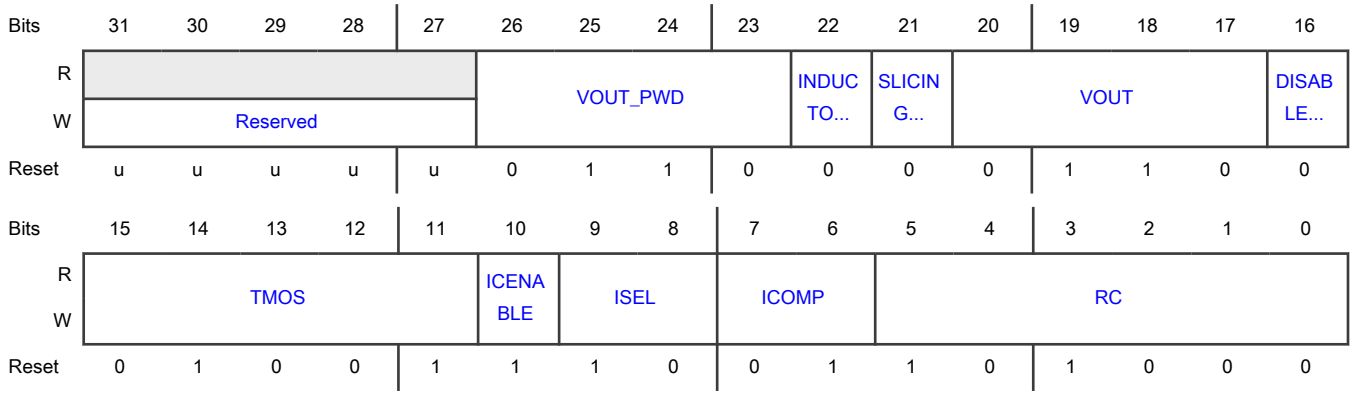
29.4.1.1.5 DCDC (first) control (DCDC0)

Reset by PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset.

Offset

Register	Offset
DCDC0	10h

Diagram



Fields

Field	Description
31-27 —	Reserved Read value is undefined, only zero should be written.
26-23 VOUT_PWD	Set output regulation voltage during Deep Sleep.
22 INDUCTORCLAMPENABLE	Enable shorting of Inductor during PFM idle time.
21 SLICINGENABLE	Enable staggered switching of power switches.
20-17 VOUT	Set output regulation voltage. 0000 - 0.95 V. 0001 - 0.975 V. 0010 - 1 V. 0011 - 1.025 V. 0100 - 1.05 V. 0101 - 1.075 V. 0110 - 1.1 V. 0111 - 1.125 V. 1000 - 1.15 V. 1001 - 1.175 V. 1010 - 1.2 V.
16	Disable Current sensing.

Table continues on the next page...

Table continued from the previous page...

Field	Description
DISABLEISENSE	
15-11 TMOS	One-shot generator reference current trimming signal.
10 ICENABLE	Selection of auto scaling of COT period with variations in VDD.
9-8 ISEL	Alter Internal biasing currents.
7-6 ICOMP	Select the type of ZCD comparator.
5-0 RC	Constant On-Time calibration.

29.4.1.1.6 DCDC (second) control register (DCDC1)

Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset.

Offset

Register	Offset
DCDC1	14h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TOFFENA...	TOFF					LCENABLE	FORCEFU...	TRIMAUTOCOT				FORCEBY...	ISCALEE...	DTESTSEL	
Reset	0	0	0	0	0	0	0	1	1	0	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	DTESTSEL	SETDC			SETCURVE			DTESTEN...	RSENSETRIM				RTRIMOFFET			
Reset	0	0	1	1	1	0	1	0	1	0	0	1	1	0	0	0

Fields

Field	Description
31 TOFFENABLE	Enable Constant Off-Time feature.
30-26 TOFF	Constant Off-Time calibration input.
25 LCENABLE	Change the range of the peak detector of current inside the inductor.
24 FORCEFULLCYCLE	Force full PFM PMOS and NMOS cycle.
23-20 TRIMAUTOCOT	Change the scaling ratio of the feedforward compensation.
19 FORCEBYPASS	Force bypass mode.
18 ISCALEENABLE	Modify COT behavior.
17-15 DTESTSEL	Select the output signal for test.
14-11 SETDC	Bandgap calibration parameter.
10-9 SETCURVE	Bandgap calibration parameter.
8 DTESTENABLE	Enable Digital test signals.
7-4 RSENSETRIM	Adjust Max inductor peak current limiting.
3-0 RTRIMOFFET	Adjust the offset voltage of BJT based comparator.

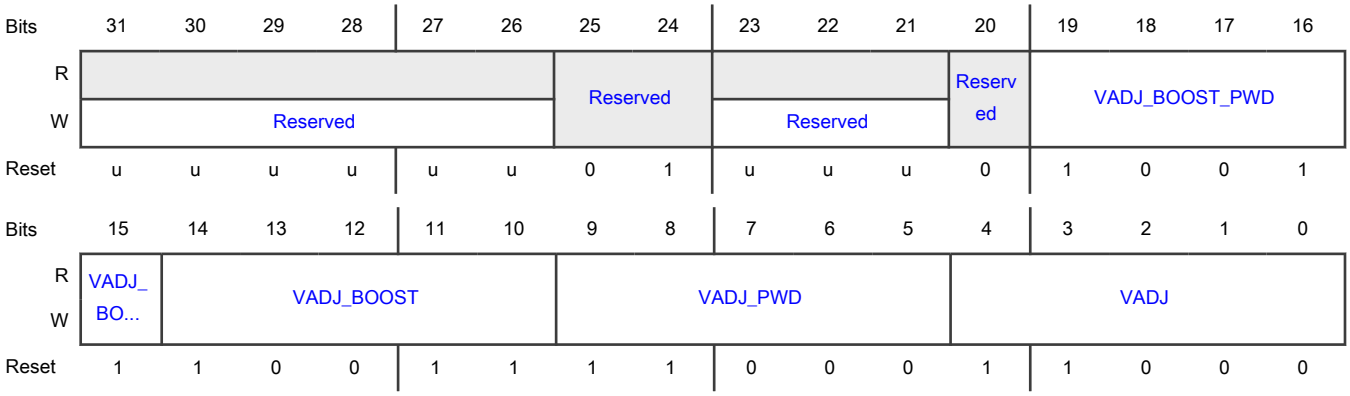
29.4.1.1.7 LDO control (LDOPMU)

Power Management Unit (PMU) and Always-On domains LDO control [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset]

Offset

Register	Offset
LDOPMU	1Ch

Diagram



Fields

Field	Description
31-26 —	Reserved Read value is undefined, only zero should be written.
25-24 —	Reserved
23-21 —	Reserved Read value is undefined, only zero should be written.
20 —	Reserved
19-15 VADJ_BOOST_PWD	Sets the Always-On domain LDO Boost output level in all power down modes.
14-10 VADJ_BOOST	Sets the Always-On domain LDO Boost output level.
9-5	Sets the Always-On domain LDO output level in all power down modes.

Table continues on the next page...

Table continued from the previous page...

Field	Description
VADJ_PWD	
4-0 VADJ	Sets the Always-On domain LDO output level. 0_0000 - 1.22 V. 0_0001 - 0.7 V. 0_0010 - 0.725 V. 0_0011 - 0.75 V. 0_0100 - 0.775 V. 0_0101 - 0.8 V. 0_0110 - 0.825 V. 0_0111 - 0.85 V. 0_1000 - 0.875 V. 0_1001 - 0.9 V. 0_1010 - 0.96 V. 0_1011 - 0.97 V. 0_1100 - 0.98 V. 0_1101 - 0.99 V. 0_1110 - 1 V. 0_1111 - 1.01 V. 1_0000 - 1.02 V. 1_0001 - 1.03 V. 1_0010 - 1.04 V. 1_0011 - 1.05 V. 1_0100 - 1.06 V. 1_0101 - 1.07 V. 1_0110 - 1.08 V. 1_0111 - 1.09 V. 1_1000 - 1.1 V. 1_1001 - 1.11 V. 1_1010 - 1.12 V. 1_1011 - 1.13 V. 1_1100 - 1.14 V. 1_1101 - 1.15 V. 1_1110 - 1.16 V.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1_1111 - 1.22 V.

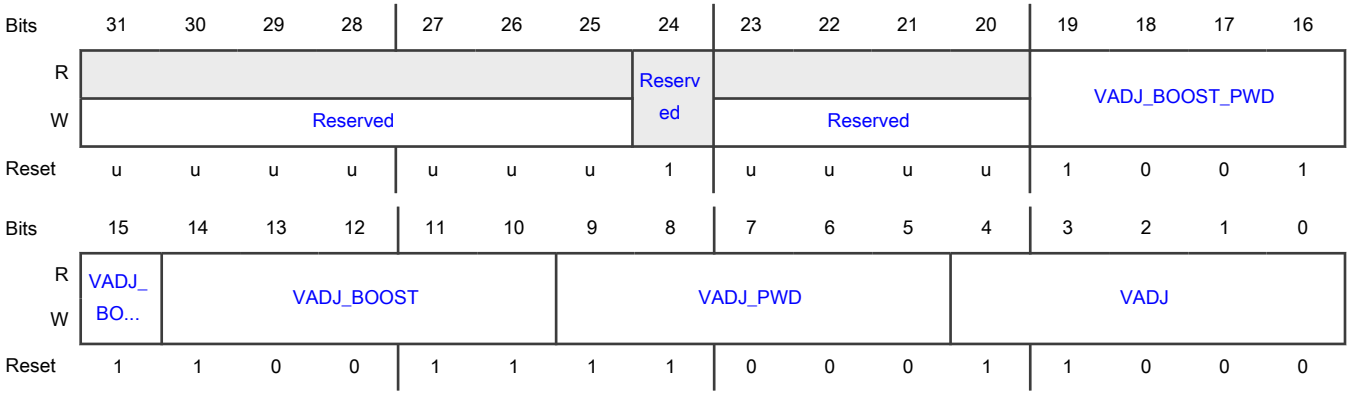
29.4.1.1.8 Memory LDO control (LDOMEM)

Memories LDO control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset]

Offset

Register	Offset
LDOMEM	20h

Diagram



Fields

Field	Description
31-25 —	Reserved Read value is undefined, only zero should be written.
24 —	Reserved
23-20 —	Reserved Read value is undefined, only zero should be written.
19-15 VADJ_BOOST_PWD	Sets the Memories LDO Boost output level in all power down modes.

Table continues on the next page...

Table continued from the previous page...

Field	Description
14-10 VADJ_BOOST	Sets the Memories LDO Boost output level.
9-5 VADJ_PWD	Sets the Memories LDO output level in all power down modes.
4-0 VADJ	Sets the Memories LDO output level. 0_0000 - 1.22 V. 0_0001 - 0.7 V. 0_0010 - 0.725 V. 0_0011 - 0.75 V. 0_0100 - 0.775 V. 0_0101 - 0.8 V. 0_0110 - 0.825 V. 0_0111 - 0.85 V. 0_1000 - 0.875 V. 0_1001 - 0.9 V. 0_1010 - 0.96 V. 0_1011 - 0.97 V. 0_1100 - 0.98 V. 0_1101 - 0.99 V. 0_1110 - 1 V. 0_1111 - 1.01 V. 1_0000 - 1.02 V. 1_0001 - 1.03 V. 1_0010 - 1.04 V. 1_0011 - 1.05 V. 1_0100 - 1.06 V. 1_0101 - 1.07 V. 1_0110 - 1.08 V. 1_0111 - 1.09 V. 1_1000 - 1.1 V. 1_1001 - 1.11 V. 1_1010 - 1.12 V. 1_1011 - 1.13 V.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1_1100 - 1.14 V. 1_1101 - 1.15 V. 1_1110 - 1.16 V. 1_1111 - 1.22 V.

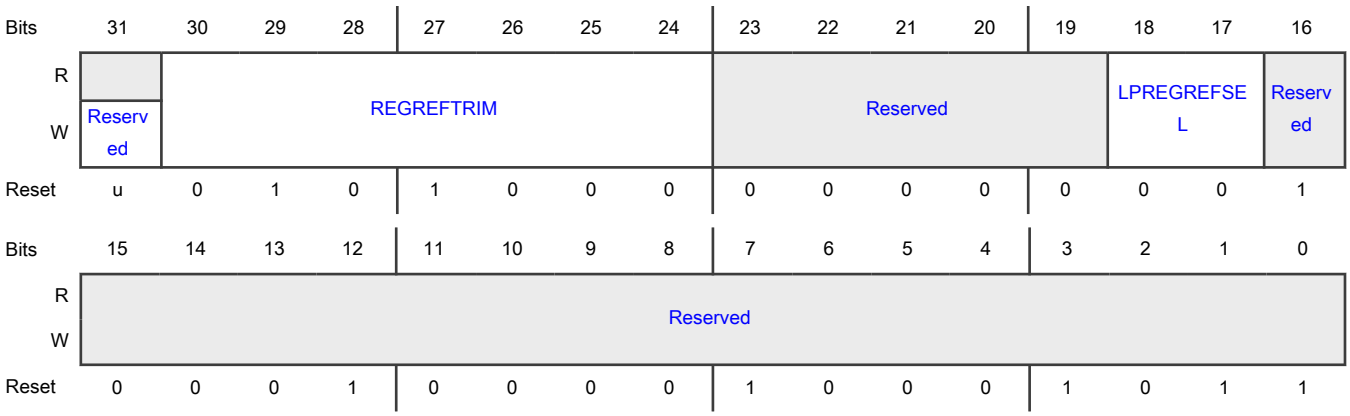
29.4.1.1.9 LDO CORE control 0 (LDOCORE0)

LDO CORE control 0 register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset]

Offset

Register	Offset
LDOCORE0	24h

Diagram



Fields

Field	Description
31 —	Reserved Read value is undefined, only zero should be written.
30-24 REGREFTRIM	High Power regulation point select.
23-19 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
18-17 LPREGREFSEL	Low Power regulation point select. 00 - 900 mV. 01 - 850 mV. 10 - 800 mV. 11 - 750 mV.
16-0 —	Reserved

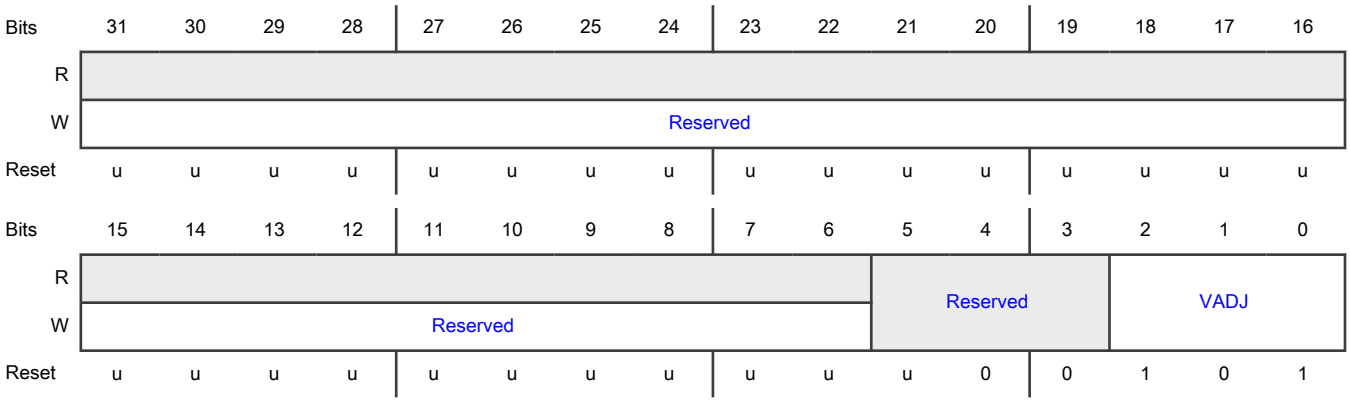
29.4.1.1.10 LDO Flash High Voltage control (LDOFLASHNV)

Flash High Voltage LDO control register [Reset by: PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset]

Offset

Register	Offset
LDOFLASHNV	28h

Diagram



Fields

Field	Description
31-6 —	Reserved Read value is undefined, only zero should be written.
5-3 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
2-0 VADJ	Sets the LDO output level. 000 - 1.650 V. 001 - 1.700 V. 010 - 1.750 V. 011 - 1.800 V. 100 - 1.850 V. 101 - 1.900 V. 110 - 1.950 V. 111 - 2.0 V.

29.4.1.1.11 OTP-eFUSE (One Time Programmable Memory) Programming LDO control (LDOEFUSEPROG)

This register is reset by PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset.

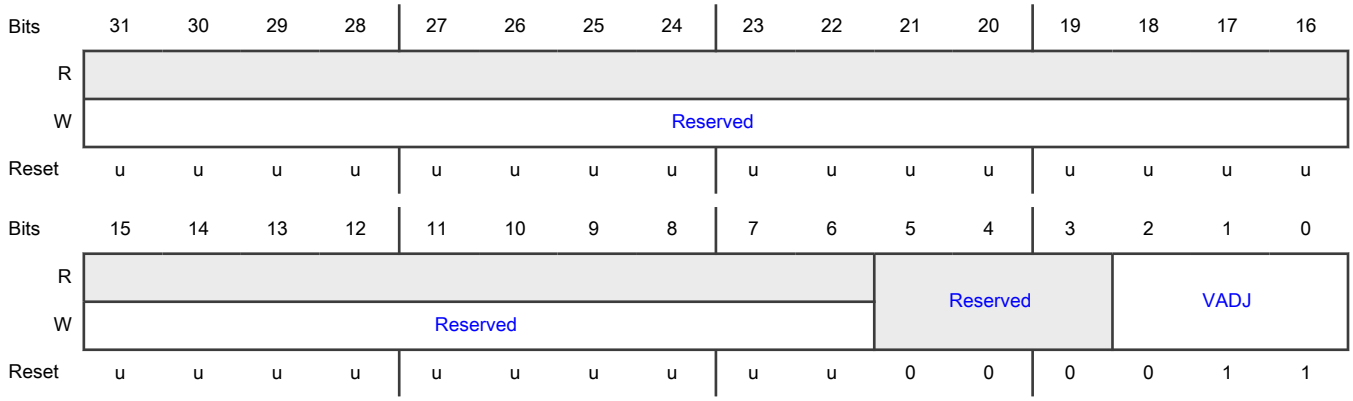
NOTE

The LDO MUST be disabled when reading the OTP-eFUSE. Otherwise, read will fail.

Offset

Register	Offset
LDOEFUSEPROG	2Ch

Diagram



Fields

Field	Description
31-6	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
5-3 —	Reserved
2-0 VADJ	Sets the LDO output level. 000 - 1.650 V. 001 - 1.700 V. 010 - 1.750 V. 011 - 1.800 V. 100 - 1.850 V. 101 - 1.900 V. 110 - 1.950 V. 111 - 2.0 V.

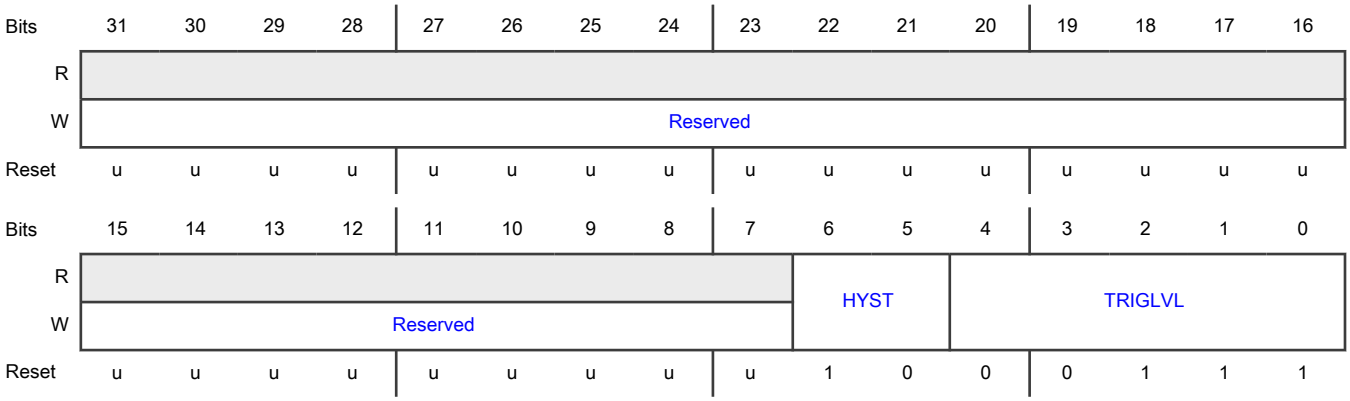
29.4.1.1.12 VDDMAIN Brown Out Dectector control (BODVDDMAIN)

This register is reset by PoR, Pin Reset, Software Reset.

Offset

Register	Offset
BODVDDMAIN	30h

Diagram



Fields

Field	Description
31-7 —	Reserved Read value is undefined, only zero should be written.
6-5 HYST	BoD Hysteresis control. 00 - 25 mV. 01 - 50 mV. 10 - 75 mV. 11 - 100 mV.
4-0 TRIGLVL	BoD trigger level. 0_0000-0_1000 - Reserved 0_1001 - 1.75 V. 0_1010 - 1.80 V. 0_1011 - 1.90 V. 0_1100 - 2.00 V. 0_1101 - 2.10 V. 0_1110 - 2.20 V. 0_1111 - 2.30 V. 1_0000 - 2.40 V. 1_0001 - 2.50 V. 1_0010 - 2.60 V. 1_0011 - 2.70 V. 1_0100 - 2.80 V. 1_0101 - 2.90 V. 1_0110 - 3.00 V. 1_0111 - 3.10 V. 1_1000 - 3.20 V. 1_1001 - 3.30 V. 1_1010 - 3.30 V. 1_1011 - 3.30 V. 1_1100 - 3.30 V. 1_1101 - 3.30 V. 1_1110 - 3.30 V. 1_1111 - 3.30 V.

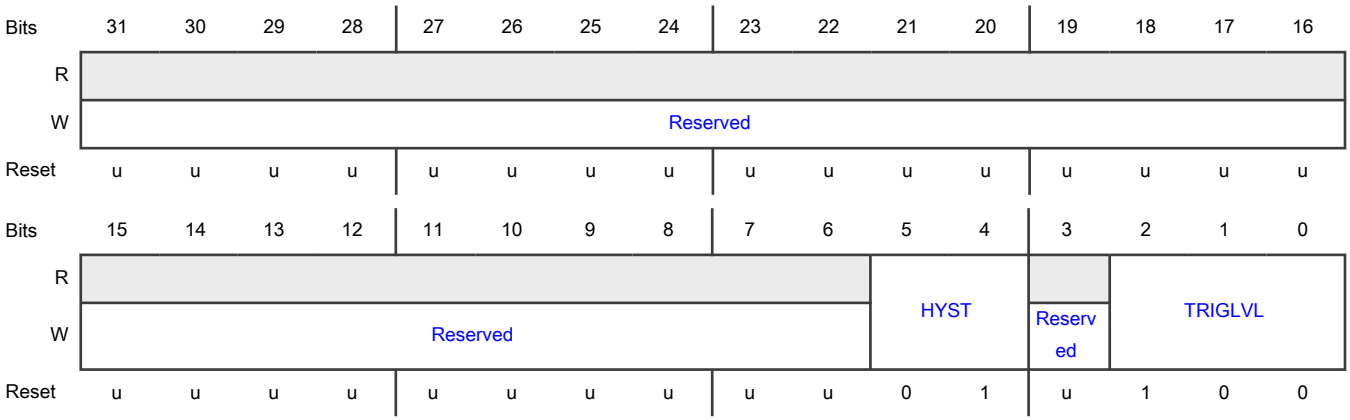
29.4.1.1.13 Digital Core logic Brown Out Dectector control (BODCORE)

This register is reset by PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset.

Offset

Register	Offset
BODCORE	38h

Diagram



Fields

Field	Description
31-6 —	Reserved Read value is undefined, only zero should be written.
5-4 HYST	BOD_CORE Hysteresis control. 00 - 25 mV. 01 - 50 mV. 10 - 75 mV. 11 - 100 mV.
3 —	Reserved Read value is undefined, only zero should be written.
2-0 TRIGLVL	BoD trigger level. 000-101 - Reserved 110 - 0.90 V. 111 - 0.95 V.

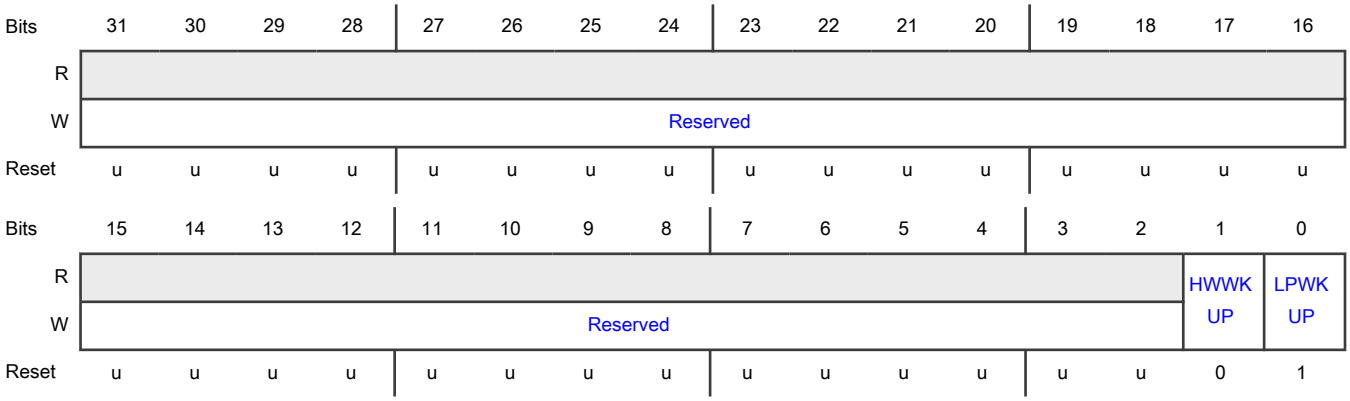
29.4.1.1.14 Analog References fast wake-up Control (REFFASTWKUP)

This register is reset by PoR.

Offset

Register	Offset
REFFASTWKUP	40h

Diagram



Fields

Field	Description
31-2 —	Reserved Read value is undefined, only zero should be written.
1 HWWKUP	Analog References fast wake-up in case of Hardware Pin reset: 0 - Analog References fast wake-up feature is disabled in case of Hardware Pin reset. 1 - Analog References fast wake-up feature is enabled in case of Hardware Pin reset.
0 LPWKUP	Analog References fast wake-up in case of wake-up from a low power mode (DEEP-SLEEP, POWER-DOWN and DEEP-POWER-DOWN): 0 - Analog References fast wake-up feature is disabled in case of wake-up from any Low power mode. 1 - Analog References fast wake-up feature is enabled in case of wake-up from any Low power mode.

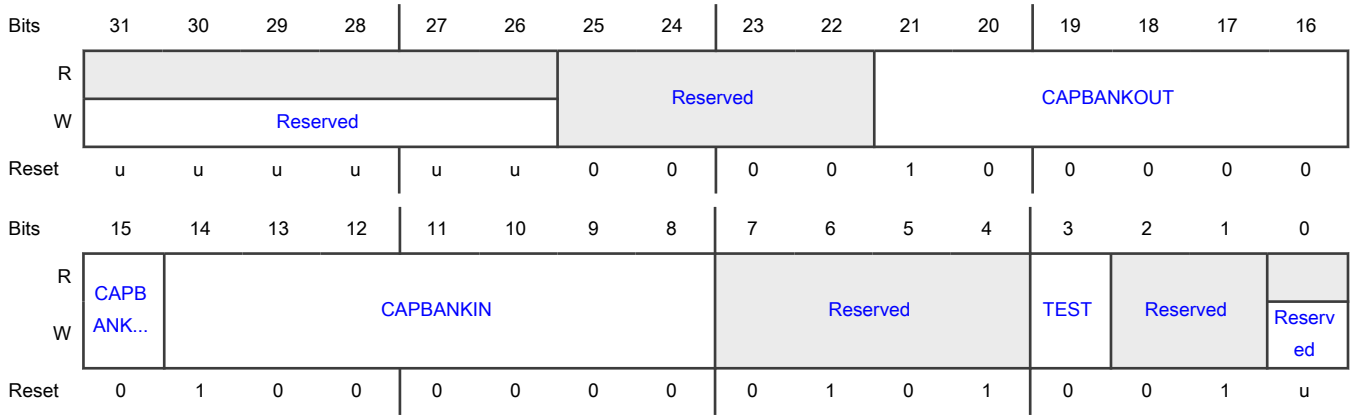
29.4.1.1.15 32 KHz Crystal oscillator (XTAL) control (XTAL32K)

This register is reset by PoR, Brown Out Detectors Reset.

Offset

Register	Offset
XTAL32K	4Ch

Diagram



Fields

Field	Description
31-26 —	Reserved Read value is undefined, only zero should be written.
25-22 —	Reserved
21-15 CAPBANKOUT	Capa bank setting output.
14-8 CAPBANKIN	Capa bank setting input.
7-4 —	Reserved
3 TEST	Oscillator Bypass Test Mode control. 0 - Oscillation mode. In the oscillation mode, a crystal, together with 2 load capacitances on XTAL32K_P and XTAL32K_N pins, are used to produce a 32768 Hz clock which is delivered to the rest of the system (RTC, OS Event Timer ...). 1 - Bypass test mode is enabled. In the bypass test mode, all XTAL-32K internal blocks are in power down. An external clock can be applied on the XTAL32K_P pin. The same clock will then be delivered to the rest of the system (RTC, OS Event Timer ...).
2-1 —	Reserved
0 —	Reserved Read value is undefined, only zero should be written.

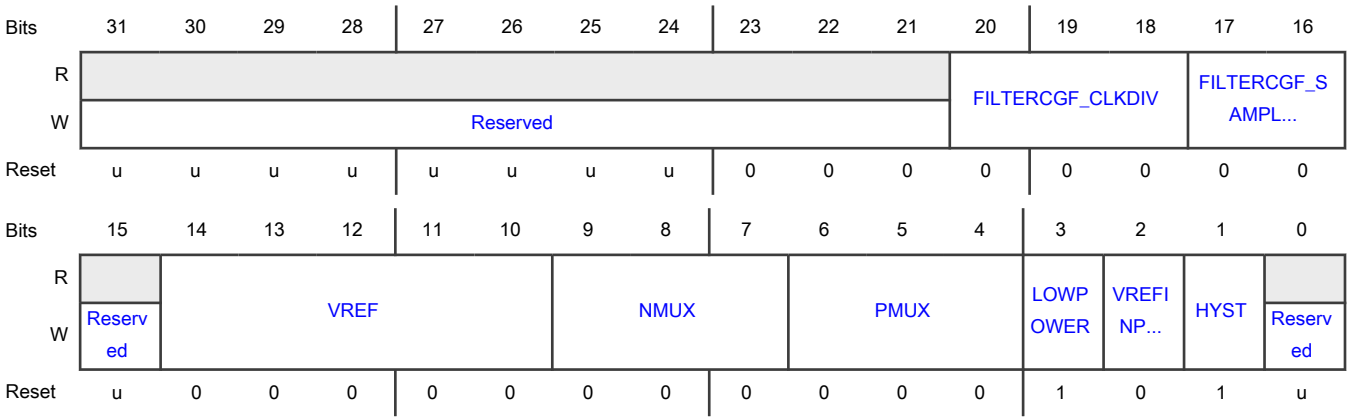
29.4.1.1.16 Analog Comparator control (COMP)

This register is reset by PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset.

Offset

Register	Offset
COMP	50h

Diagram



Fields

Field	Description
31-21 —	Reserved Read value is undefined, only zero should be written.
20-18 FILTERCGF_CLKDIV	Filter Clock divider. Filter clock equals the Analog Comparator clock divided by 2^FILTERCGF_CLKDIV. 000 - Filter clock period duration equals 1 Analog Comparator clock period. 001 - Filter clock period duration equals 2 Analog Comparator clock period. 010 - Filter clock period duration equals 4 Analog Comparator clock period. 011 - Filter clock period duration equals 8 Analog Comparator clock period. 100 - Filter clock period duration equals 16 Analog Comparator clock period. 101 - Filter clock period duration equals 32 Analog Comparator clock period. 110 - Filter clock period duration equals 64 Analog Comparator clock period. 111 - Filter clock period duration equals 128 Analog Comparator clock period.
17-16 FILTERCGF_SAMPLEMODE	Control the filtering of the Analog Comparator output. 00 - Bypass mode. Filtering is disabled. The raw Analog Comparator output will be passed-through.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01 - Filter 1 clock period. Any pulse duration shorter than one cycle of the designated filter clock (see FILTERCGF_CLKDIV) will be filtered-out. Pulse widths up to two cycles long may be filtered. 10 - Filter 2 clock period. Any pulse duration shorter than two cycles of the designated filter clock will be filtered-out. Pulse widths up to three cycles long may be filtered. 11 - Filter 3 clock period. Any pulse duration shorter than three cycles of the designated filter clock will be filtered-out. Pulse widths up to four cycles long may be filtered.
15 —	Reserved Read value is undefined, only zero should be written.
14-10 VREF	Control reference voltage step, per steps of (VREFINPUT/31).
9-7 NMUX	Control word for N multiplexer:. 000 - VREF (See field VREFINPUT). 001 - Pin P0_0. 010 - Pin P0_9. 011 - Pin P0_18. 100 - Pin P1_14. 101 - Reserved.
6-4 PMUX	Control word for P multiplexer:. 000 - VREF (See field VREFINPUT). 001 - Pin P0_0. 010 - Pin P0_9. 011 - Pin P0_18. 100 - Pin P1_14. 101 - Reserved.
3 LOWPOWER	Low power mode. 0 - High speed mode. 1 - Low power mode (Low speed).
2 VREFINPUT	Dedicated control bit to select between internal VREF and VDDA (for the resistive ladder). 0 - Select internal VREF. 1 - Select VDDA.
1 HYST	Hysteris when hyst = '1'. 0 - Hysteresis is disabled. 1 - Hysteresis is enabled.

Table continues on the next page...

Table continued from the previous page...

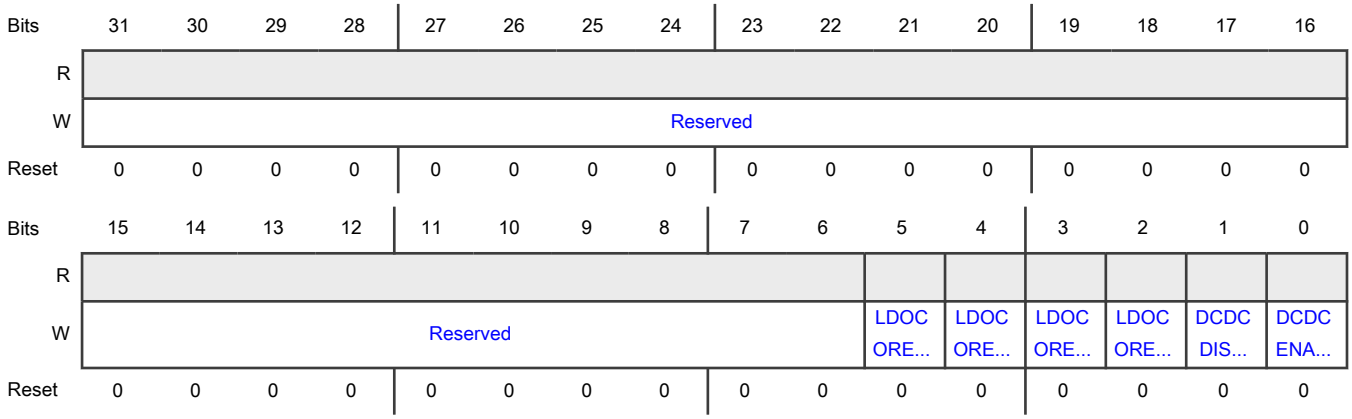
Field	Description
0 —	Reserved Read value is undefined, only zero should be written.

29.4.1.1.17 DCDC and LDOCORE power state (enable/disable) control. (CMD)

Offset

Register	Offset
CMD	60h

Diagram



Fields

Field	Description
31-6 —	Reserved
5 LDOCORELOWPWRDISABLE	Disable LDO CORE Low Power Mode (self clearing bit). Note: If both LDOCORELOWPWRENABLE and LDOCORELOWPWRDISBALE are set during the same command, the LDO CORE Low Power Mode will be enabled. 0 - No effect. 1 - Disable LDO CORE Low Power Mode. Automatically reset to '0' by the Hardware.
4 LDOCORELOWPWRENABLE	Enable LDO CORE Low Power Mode (self clearing bit). Note: If both LDOCORELOWPWRENABLE and LDOCORELOWPWRDISBALE are set during the same command, the LDO CORE Low Power Mode will be enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - No effect. 1 - Enable LDO CORE Low Power Mode. Automatically reset to '0' by the Hardware.
3 LDOCOREHIGHPWRDISABLE	Disable LDO CORE High Power Mode (self clearing bit). Note: If both LDOCOREHIGHPWRENABLE and LDOCOREHIGHPWRDISBALE are set during the same command, the LDO CORE High Power Mode will be enabled. 0 - No effect. 1 - Disable LDO CORE High Power Mode. Automatically reset to '0' by the Hardware.
2 LDOCOREHIGHPWRENABLE	Enable LDO CORE High Power Mode (self clearing bit). Note: If both LDOCOREHIGHPWRENABLE and LDOCOREHIGHPWRDISBALE are set during the same command, the LDO CORE High Power Mode will be enabled. 0 - No effect. 1 - Enable LDO CORE High Power Mode. Automatically reset to '0' by the Hardware.
1 DCDCDISABLE	Disable DCDC (self clearing bit). Note: If both DCDCENABLE and DCDCDISABLE are set during the same command, the DCDC will be enabled. 0 - No effect. 1 - Disbale DCDC. Automatically reset to '0' by the Hardware.
0 DCDCENABLE	Enable DCDC (self clearing bit). Note: If both DCDCENABLE and DCDCDISABLE are set during the same command, the DCDC will be enabled. 0 - No effect. 1 - Enable DCDC. Automatically reset to '0' by the Hardware.

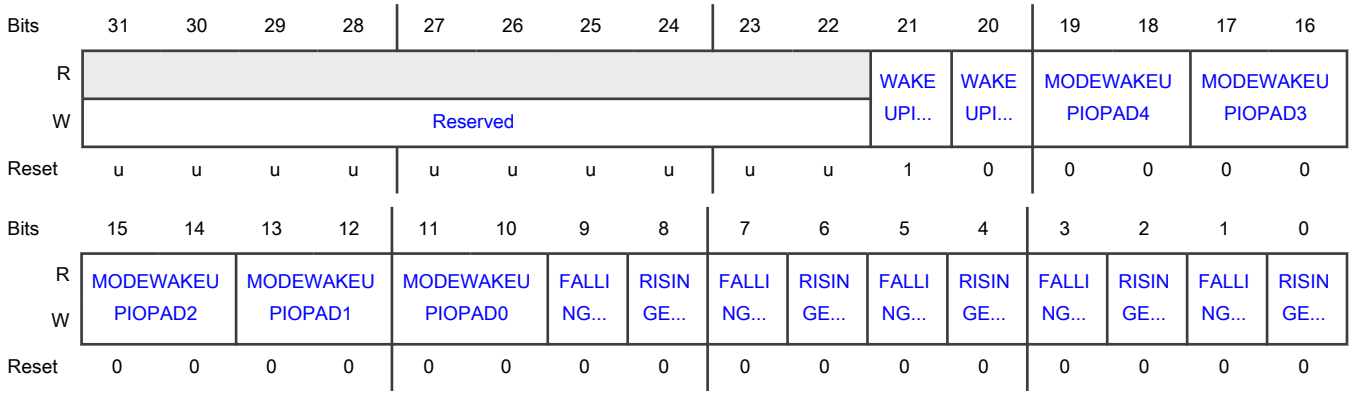
29.4.1.1.18 Wake-up IO control (WAKEUPIOCTRL)

Deep Power Down wake-up source is reset by PoR, Pin Reset, Software Reset.

Offset

Register	Offset
WAKEUPIOCTRL	64h

Diagram



Fields

Field	Description
31-22 —	Reserved Read value is undefined, only zero should be written.
21 WAKEUPIO_RSTN	WAKEUP IO event detector reset control. 0 - Bloc is reset. 1 - Bloc is not reset.
20 WAKEUPIO_ENABLE_CTRL	Enable WAKEUP IO PAD control from MODEWAKEUPIOPAD (bits 10 to 19). 0 - WAKEUP IO PAD mode control comes from IOCON. 1 - WAKEUP IO PAD mode control comes from MODEWAKEUPIOPAD (bits 10 to 19).
19-18 MODEWAKEUPIOPAD4	Selects function mode (on-chip pull-up/pull-down resistor control). 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
17-16 MODEWAKEUPIOPAD3	Selects function mode (on-chip pull-up/pull-down resistor control). 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
15-14 MODEWAKEUPIOPAD2	Selects function mode (on-chip pull-up/pull-down resistor control). 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	11 - Repeater. Repeater mode.
13-12 MODEWAKEUPIOPAD1	Selects function mode (on-chip pull-up/pull-down resistor control). 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
11-10 MODEWAKEUPIOPAD0	Selects function mode (on-chip pull-up/pull-down resistor control). 00 - Inactive. Inactive (no pull-down/pull-up resistor enabled). 01 - Pull-down. Pull-down resistor enabled. 10 - Pull-up. Pull-up resistor enabled. 11 - Repeater. Repeater mode.
9 FALLINGEDGEWAKEUP4	Enable / disable detection of falling edge events on Wake Up 4 pin in Deep Power Down modes: 0 - Falling edge detection is disabled. 1 - Falling edge detection is enabled.
8 RISINGEDGEWAKEUP4	Enable / disable detection of rising edge events on Wake Up 4 pin in Deep Power Down modes: 0 - Rising edge detection is disabled. 1 - Rising edge detection is enabled.
7 FALLINGEDGEWAKEUP3	Enable / disable detection of falling edge events on Wake Up 3 pin in Deep Power Down modes: 0 - Falling edge detection is disabled. 1 - Falling edge detection is enabled.
6 RISINGEDGEWAKEUP3	Enable / disable detection of rising edge events on Wake Up 3 pin in Deep Power Down modes: 0 - Rising edge detection is disabled. 1 - Rising edge detection is enabled.
5 FALLINGEDGEWAKEUP2	Enable / disable detection of falling edge events on Wake Up 2 pin in Deep Power Down modes:. 0 - Falling edge detection is disabled. 1 - Falling edge detection is enabled.
4 RISINGEDGEWAKEUP2	Enable / disable detection of rising edge events on Wake Up 2 pin in Deep Power Down modes:. 0 - Rising edge detection is disabled. 1 - Rising edge detection is enabled.
3 FALLINGEDGEWAKEUP1	Enable / disable detection of falling edge events on Wake Up 1 pin in Deep Power Down modes:. 0 - Falling edge detection is disabled. 1 - Falling edge detection is enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Description
2 RISINGEDGEWAKEUP1	Enable / disable detection of rising edge events on Wake Up 1 pin in Deep Power Down modes:. 0 - Rising edge detection is disabled. 1 - Rising edge detection is enabled.
1 FALLINGEDGEWAKEUP0	Enable / disable detection of falling edge events on Wake Up 0 pin in Deep Power Down modes:. 0 - Falling edge detection is disabled. 1 - Falling edge detection is enabled.
0 RISINGEDGEWAKEUP0	Enable / disable detection of rising edge events on Wake Up 0 pin in Deep Power Down modes:. 0 - Rising edge detection is disabled. 1 - Rising edge detection is enabled.

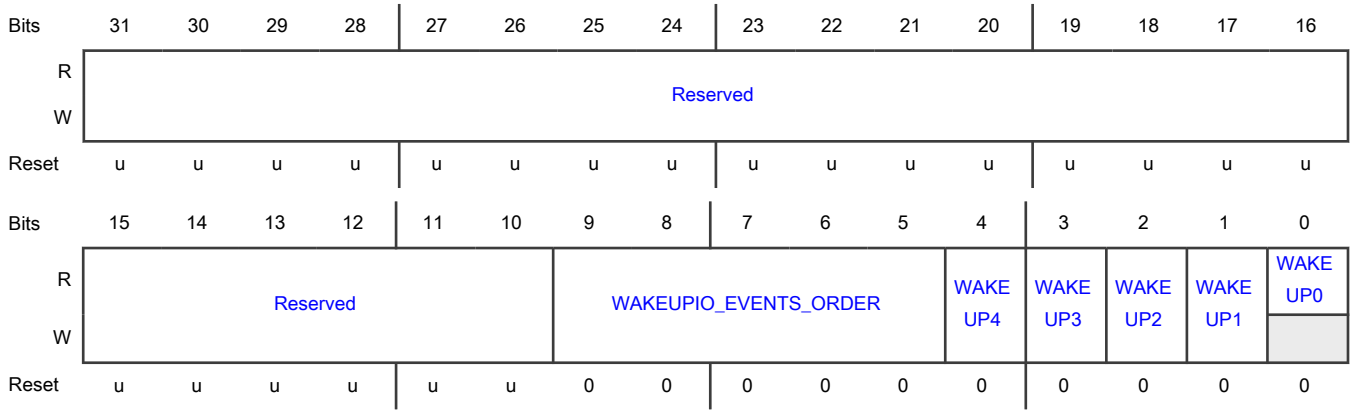
29.4.1.1.19 Wake-up I/O source (WAKEIOCAUSE)

Allows to identify the Wake-up I/O source from Deep Power Down mode. This register is relevant when RESETCAUSE[DPDRESET_WAKEUPIO] is set to '1'.

Offset

Register	Offset
WAKEIOCAUSE	68h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
9-5 WAKEUPIO_EVENTS_ORDER	In DEEP-POWER-DOWN mode, indicates which wake up I/O event occured first when several wake up I/Os are enabled. For any (not null) value not listed below, it indicates that several wake up I/O events occured at the same time (within less than 1 nano-second of each other), with: WAKEUPIO_EVENTS_ORDER[0] for WAKEUP0 WAKEUPIO_EVENTS_ORDER[1] for WAKEUP1 WAKEUPIO_EVENTS_ORDER[2] for WAKEUP2 WAKEUPIO_EVENTS_ORDER[3] for WAKEUP3 WAKEUPIO_EVENTS_ORDER[4] for WAKEUP4 For example, the value b10110 (0x16) indicates WAKEUP1, WAKEUP2 and WAKEUP4 events occured at the same time. 0_0000 - None 0_0001 - Wake up I/O 0 0_0010 - Wake up I/O 1 0_0100 - Wake up I/O 2 0_1000 - Wake up I/O 3 1_0000 - Wake up I/O 4
4 WAKEUP4	Allows to identify Wake up I/O 4 as the wake-up source from Deep Power Down mode. 0 - Last wake up from Deep Power down mode was NOT triggred by wake up I/O 4. 1 - Last wake up from Deep Power down mode was triggred by wake up I/O 4.
3 WAKEUP3	Allows to identify Wake up I/O 3 as the wake-up source from Deep Power Down mode. 0 - Last wake up from Deep Power down mode was NOT triggred by wake up I/O 3. 1 - Last wake up from Deep Power down mode was triggred by wake up I/O 3.
2 WAKEUP2	Allows to identify Wake up I/O 2 as the wake-up source from Deep Power Down mode. 0 - Last wake up from Deep Power down mode was NOT triggred by wake up I/O 2. 1 - Last wake up from Deep Power down mode was triggred by wake up I/O 2.
1 WAKEUP1	Allows to identify Wake up I/O 1 as the wake-up source from Deep Power Down mode. 0 - Last wake up from Deep Power down mode was NOT triggred by wake up I/O 1. 1 - Last wake up from Deep Power down mode was triggred by wake up I/O 1.
0 WAKEUP0	Allows to identify Wake up I/O 0 as the wake-up source from Deep Power Down mode. 0 - Last wake up from Deep Power down mode was NOT triggred by wake up I/O 0. 1 - Last wake up from Deep Power down mode was triggred by wake up I/O 0.

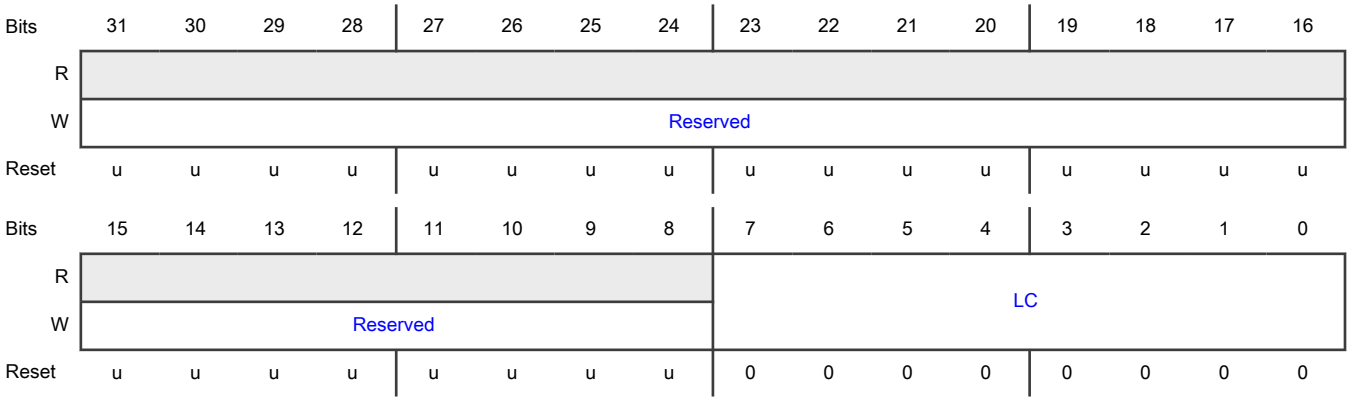
29.4.1.1.20 Life Cycle State (LIFECYCLESTATE)

Life Cycle State as configured in the OTP

Offset

Register	Offset
LIFECYCLESTATE	6Ch

Diagram



Fields

Field	Description
31-8 —	Reserved
7-0 LC	Life Cycle state

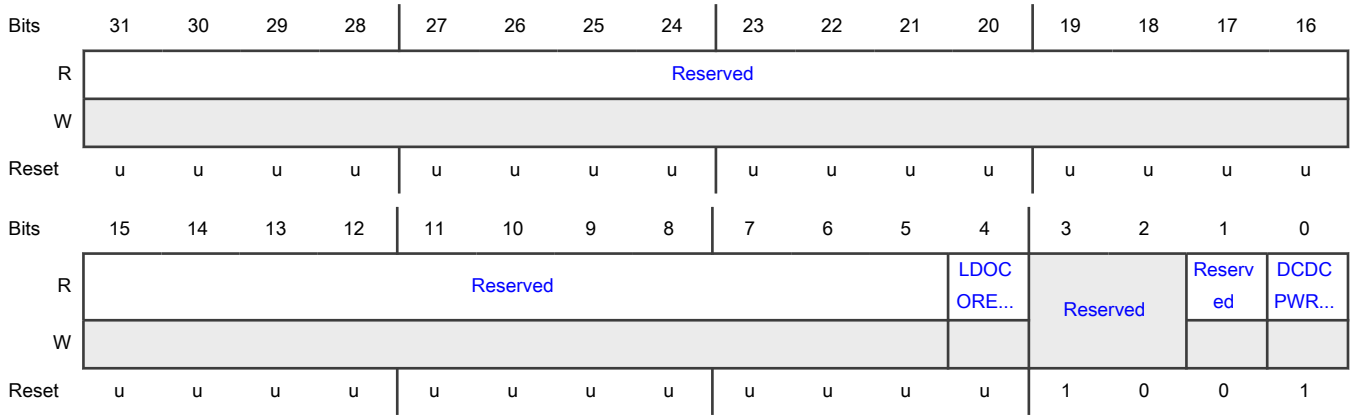
29.4.1.1.21 Power status (STATUSPWR)

Power status from various analog modules (DCDC, LDO, etc)

Offset

Register	Offset
STATUSPWR	70h

Diagram



Fields

Field	Description
31-5 —	Reserved Read value is undefined, only zero should be written.
4 LDOCOREPWROK	CORE LDO power OK.
3-2 —	Reserved
1 —	Reserved
0 DCDCPWROK	DCDC converter power OK.

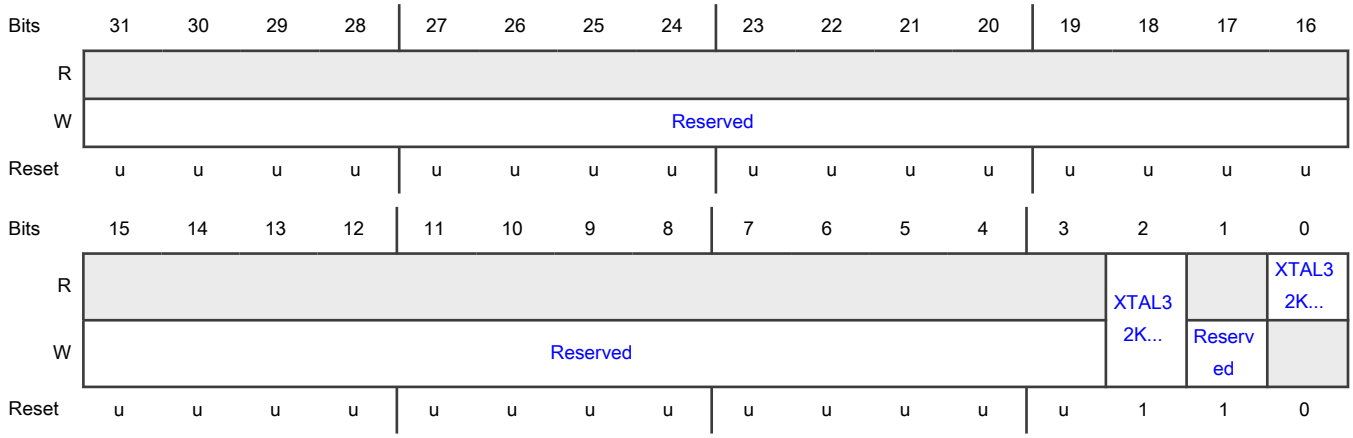
29.4.1.1.22 Clock status (STATUSCLK)

FRO and XTAL status register is reset by: PoR, Brown Out Detectors Reset.

Offset

Register	Offset
STATUSCLK	74h

Diagram



Fields

Field	Description
31-3 —	Reserved Read value is undefined, only zero should be written.
2 XTAL32KOSCFAILURE	XTAL32 KHZ oscillator oscillation failure detection indicator. 0 - No oscillation failure has been detetced since the last time this bit has been cleared. 1 - At least one oscillation failure has been detetced since the last time this bit has been cleared.
1 —	Reserved Read value is undefined, only zero should be written.
0 XTAL32KOK	XTAL oscillator 32 K OK signal.

29.4.1.1.23 Always-on 0 (AOREG0)

General purpose always-on domain data storage, 16 bits wide [Reset by: PoR].

Offset

Register	Offset
AOREG0	80h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
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	DATA_15_0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-16 —	Reserved
15-0 DATA_15_0	General purpose always on domain data storage.

29.4.1.1.24 Always-on 1 (AOREG1)

General purpose always on domain data storage [Reset by: PoR, Brown Out Detectors Reset].

This register is managed and updated by the ROM Boot Code. It gathers some important System Status information like the last System reset cause and the number of fatal errors that occurred during the ROM boot. Though it is readable and writable, it MUST NOT be modified by any application.

Offset

Register	Offset
AOREG1	84h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R					BOOTERRORCOUNTER											DPD_EVE...
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	DPD_EVENTS_ORDER	CDOGRES...	DPDRESE...	DPDRESE...	DPDRESE...	DPDRESET	SWRRESET	WDTRESET	SYSTEMR...	BODRESET	PADRESET	POR				
W													Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-28 —	Reserved Read value is undefined, only zero should be written.
27-24 BOOTERRORCOUNTER	ROM Boot Fatal Error Counter.
23-17 —	Reserved Read value is undefined, only zero should be written.
16-14 DPD_EVENTS_ORDER	In DEEP-POWER-DOWN mode, indicates which reset event occured first, between a wake up I/O event (in DEEP-POWER-DOWN), a RTC event (in DEEP-POWER-DOWN) and a OS Timer event (in DEEP-POWER-DOWN). May be usefull when several reset events are enabled during DEEP-POWER-DOWN. 000 - No event 001 - WAKEUPIO 010 - RTC 011 - Both WAKEUPIO and RTC events occured at the same time (the 2 events occured within 1 nano-second of each other) 100 - OSTIMER 101 - Both WAKEUPIO and OSTIMER events occured at the same time (the 2 events occured within 1 nano-second of each other) 110 - Both RTC and OSTIMER events occured at the same time (the 2 events occured within 1 nano-second of each other) 111 - WAKEUPIO, RTC and OSTIMER events occured at the same time (the 3 events occured within 1 nano-second of each other)
13 CDOGRESET	The last chip reset was caused by the code Watchdog.
12 DPDRESET_OSTIMER	An OS Timer event occured during a DEEP-POWER-DOWN mode. See the related field DPD_EVENTS_ORDER in this register
11 DPDRESET_RTC	A RTC event occured during DEEP-POWER-DOWN mode. See the related field DPD_EVENTS_ORDER in this register
10 DPDRESET_WAKEUPIO	A Wake-up I/O reset event occured during DEEP-POWER-DOWN mode. See the related field DPD_EVENTS_ORDER in this register
9	The last chip reset was caused by a Software event.

Table continues on the next page...

Table continued from the previous page...

Field	Description
SWRRESET	
8 WDTRESET	The last chip reset was caused by the Watchdog Timer.
7 SYSTEMRESET	The last chip reset was caused by a System Reset requested by the ARM CPU.
6 BODRESET	The last chip reset was caused by a Brown Out Detector (BoD), either BOD_VDDMAIN or BOD_CORE.
5 PADRESET	The last chip reset was caused by a Pin Reset.
4 POR	The last chip reset was caused by a Power On Reset.
3-0 —	Reserved Read value is undefined, only zero should be written.

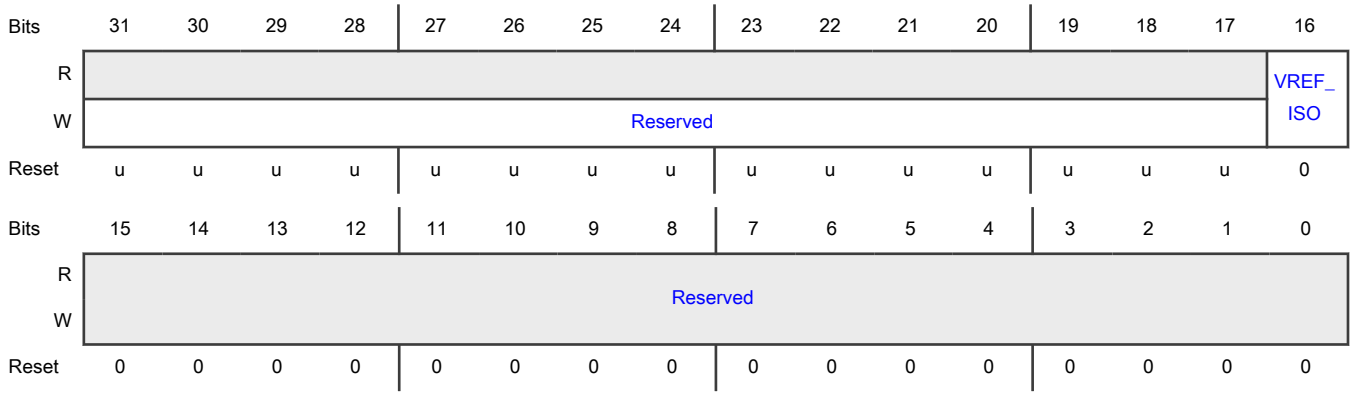
29.4.1.1.25 Miscellaneous control (MISCCTRL)

Miscellaneous control Register for PMU is reset by PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset.

Offset

Register	Offset
MISCCTRL	90h

Diagram



Fields

Field	Description
31-17 —	Reserved Read value is undefined, only zero should be written.
16 VREF_ISO	VREF isolation control. 0 - VREF module isolation is disabled. 1 - VREF module isolation is enabled.
15-0 —	Reserved

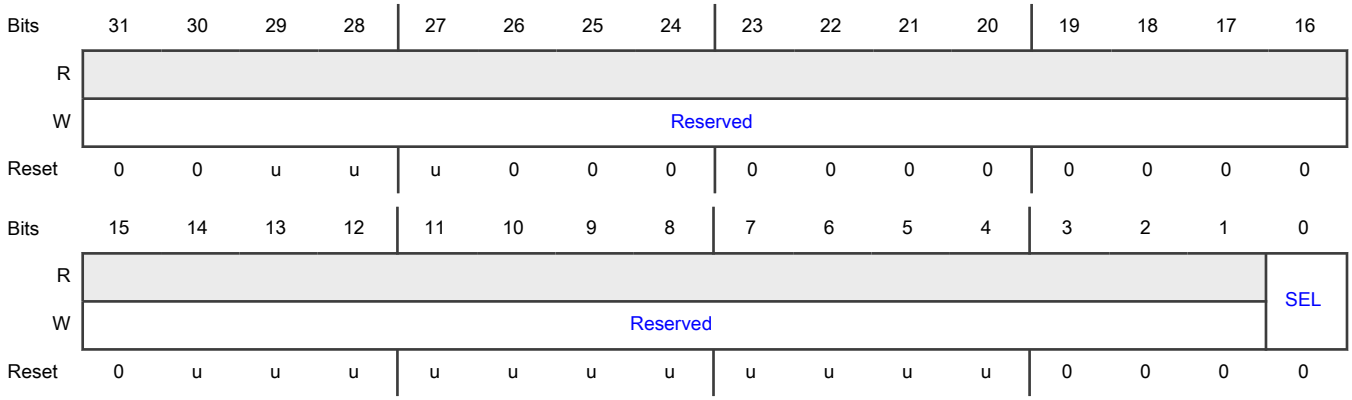
29.4.1.1.26 32 KHz clocks source control (RTCOSC32K)

This register is reset by PoR, Brown Out Detectors Reset.

Offset

Register	Offset
RTCOSC32K	98h

Diagram



Fields

Field	Description
31-1 —	Reserved Read value is undefined, only zero should be written.
0 SEL	Select the 32K oscillator to be used in for the RTC, the OS Event Timer and the rest of the SoC (either XTAL32KHz or FRO32KHz) . 0 - FRO 32 KHz.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - XTAL 32KHz.

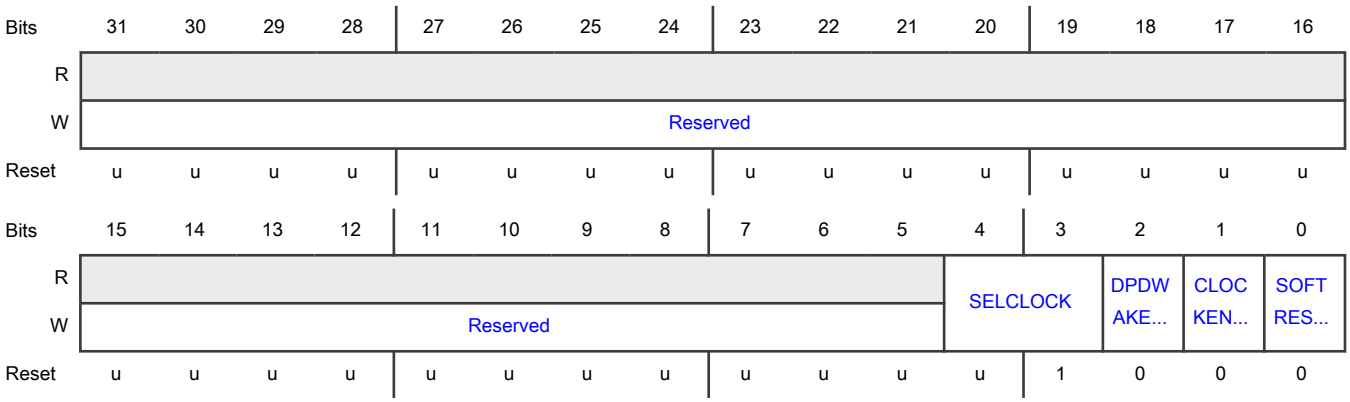
29.4.1.1.27 OS Event Timer control (OSEVENTTIMER)

This register is reset by PoR, Brown Out Detectors Reset.

Offset

Register	Offset
OSEVENTTIMER	9Ch

Diagram



Fields

Field	Description
31-5 —	Reserved Read value is undefined, only zero should be written.
4-3 SELCLOCK	Select OS Event Timer Clock source 00 - 32-KHz Free Running Oscillator (FRO) 01 - 32-KHz Crystal Oscillator (XTAL) 10 - 1-MHz FRO 11 - System Bus clock
2 DPDWAKEUPENABLE	Wake up enable in Deep Power Down mode (To be used in Enable Deep Power Down mode).
1	Enable OSTIMER 32 KHz clock.

Table continues on the next page...

Table continued from the previous page...

Field	Description
CLOCKENABLE	
0 SOFTRESET	Active high reset.

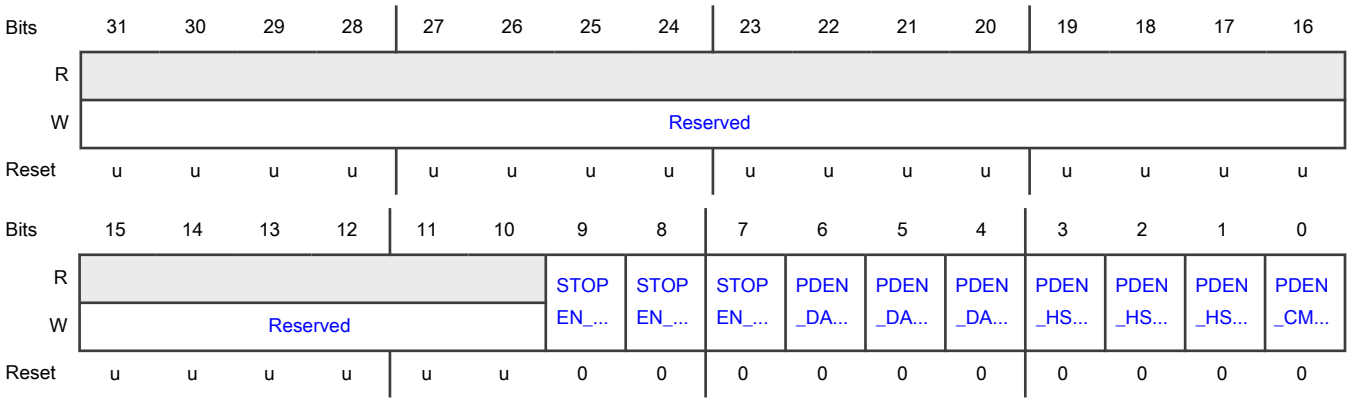
29.4.1.1.28 Power down sleep configuration 1 (PDSLEEPCFG1)

Controls the power to various modules during Low Power modes - DEEP-SLEEP, POWER-DOWN and DEEP-POWER-DOWN. This register is reset by PoR, Pin Reset, Brown Out Detectors Reset, Software Reset.

Offset

Register	Offset
PDSLEEPCFG1	A0h

Diagram



Fields

Field	Description
31-10 —	Reserved
9 STOPEN_DAC2	Controls DAC2 Stop mode during DEEP-SLEEP & POWER-DOWN (DAC stop mode is always disabled in DEEP-POWER-DOWN). 0 - DAC Stop Mode is disabled. 1 - DAC Stop Mode is enabled.
8	Controls DAC1 Stop mode during DEEP-SLEEP & POWER-DOWN (DAC stop mode is always disabled in DEEP-POWER-DOWN).

Table continues on the next page...

Table continued from the previous page...

Field	Description
STOPEN_DAC1	0 - DAC Stop Mode is disabled. 1 - DAC Stop Mode is enabled.
7 STOPEN_DAC0	Controls DAC0 Stop mode during DEEP-SLEEP & POWER-DOWN (DAC stop mode is always disabled in DEEP-POWER-DOWN). 0 - DAC Stop Mode is disabled. 1 - DAC Stop Mode is enabled.
6 PDEN_DAC2	Controls DAC2 power during DEEP-SLEEP & POWER-DOWN (always shut down during DEEP-POWER-DOWN). 0 - DAC2 is powered on during low power mode. 1 - DAC2 is powered off during low power mode.
5 PDEN_DAC1	Controls DAC1 power during DEEP-SLEEP & POWER-DOWN (always shut down during DEEP-POWER-DOWN). 0 - DAC1 is powered on during low power mode. 1 - DAC1 is powered off during low power mode.
4 PDEN_DAC0	Controls DAC0 power during DEEP-SLEEP & POWER-DOWN (always shut down during DEEP-POWER-DOWN). 0 - DAC0 is powered on during low power mode. 1 - DAC0 is powered off during low power mode.
3 PDEN_HSCMP2_DAC	Controls High Speed Comparator2 DAC power during DEEP-SLEEP (always shut down during POWER-DOWN & DEEP-POWER-DOWN). 0 - High Speed Comparator2 DAC is powered on during low power mode. 1 - High Speed Comparator2 DAC is powered off during low power mode.
2 PDEN_HSCMP1_DAC	Controls High Speed Comparator1 DAC power during DEEP-SLEEP (always shut down during POWER-DOWN & DEEP-POWER-DOWN). 0 - High Speed Comparator1 DAC is powered on during low power mode. 1 - High Speed Comparator1 DAC is powered off during low power mode.
1 PDEN_HSCMP0_DAC	Controls High Speed Comparator0 DAC power during DEEP-SLEEP (always shut down during POWER-DOWN & DEEP-POWER-DOWN). 0 - High Speed Comparator0 DAC is powered on during low power mode. 1 - High Speed Comparator0 DAC is powered off during low power mode.
0 PDEN_CMPBIAS	Controls Comparators 1/2/3 Bias power during DEEP-SLEEP (always shut down during POWER-DOWN & DEEP-POWER-DOWN). 0 - Analog Bias is powered on during low power mode. 1 - Analog Bias is powered off during low power mode.

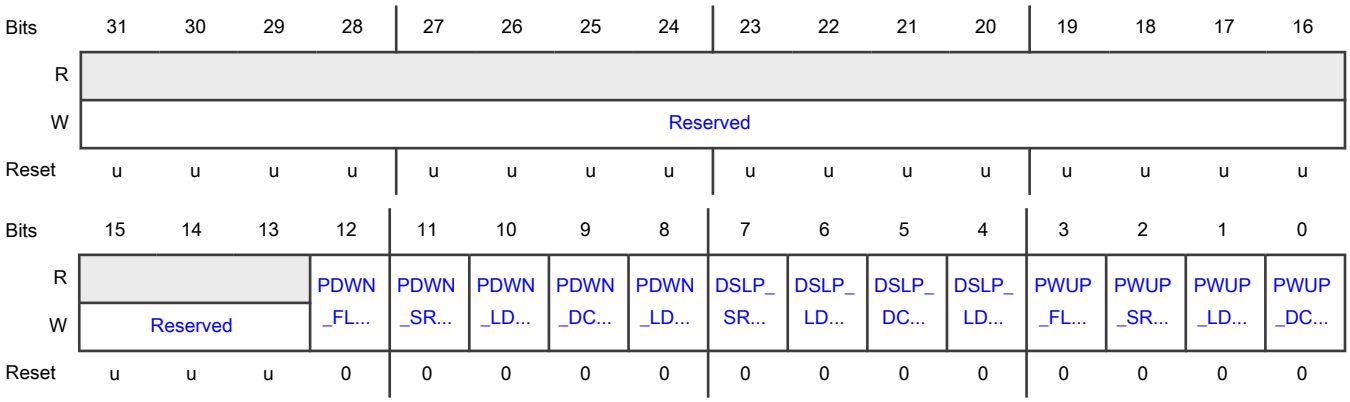
29.4.1.1.29 Time-out events (TIMEOUTEVENTS)

Record time-out errors that might occur at different stages during IC power up.

Offset

Register	Offset
TIMEOUTEVENTS	A4h

Diagram



Fields

Field	Description
31-13 —	Reserved Read value is undefined, only zero should be written.
12 PDWN_FLASHINIT_DONE	1: a time out event occured during power down when waiting for Flash initialization.
11 PDWN_SRAM_WAKEUP	1: a time out event occured during power down when waiting for SRAM to become functional.
10 PDWN_LDOFLASHNV_OK	1: a time out event occured during power down when waiting for LDO Flash NV to become functional.
9 PDWN_DCDC_BODVDDMAIN_OK	1: a time out event occured during power down when waiting for DCDC or BOD_VDDMAIN to become functional.
8	1: a time out event occured during power down when waiting for for LDO Flash NV or SRAM shut off.

Table continues on the next page...

Table continued from the previous page...

Field	Description
PDWN_LDOFLASH_SRAM_OFF	
7 DSLP_SRAM_WAKEUP	1: a time out event occured during deep sleep when waiting for SRAM to become functional.
6 DSLP_LDOFLASHNV_OK	1: a time out event occured during deep sleep when waiting for LDO Flash NV to become functional.
5 DSLP_DCDC_OK	1: a time out event occured during deep sleep when waiting for DCDC to become functional.
4 DSLP_LDOFLASH_SRAM_OFF	1: a time out event occured during deep sleep when waiting for LDO Flash NV or SRAM shut off.
3 PWUP_FLASHINIT_DONE	1: a time out event occured during power up when waiting for Flash initialization.
2 PWUP_SRAM_WAKEUP	1: a time out event occured during power up when waiting for SRAM to become functional.
1 PWUP_LDOFLASHNV_OK	1: a time out event occured during power up when waiting for LDO Flash NV to become functional.
0 PWUP_DCDC_OK	1: a time out event occured during power up when waiting for DCDC to become functional.

29.4.1.1.30 Power down sleep configuration 0 (PDSLEEPCFG0)

Controls the power to various modules during Low Power modes - DEEP-SLEEP, POWER-DOWN and DEEP-POWER-DOWN. This register is reset by PoR, Pin Reset, Brown Out Detectors Reset, Software Reset.

Offset

Register	Offset
PDSLEEPCFG0	B0h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PDEN_VR...	PDEN_OP...	PDEN_OP...	PDEN_OP...	PDEN_HS...	PDEN_HS...	PDEN_HS...	PDEN_ROM	PDEN_PL...	Reserved	PDEN_LD...	PDEN_LD...	Reserved	PDEN_LD...	Reserved	PDEN_LD...
Reset	u	u	u	u	u	u	u	0	0	0	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	Reserved		PDEN_CO...	Reserved	PDEN_US...	PDEN_PL...	PDEN_PL...	PDEN_XT...	PDEN_XT...	PDEN_FR...	PDEN_FR...	PDEN_FR...	PDEN_BO...	PDEN_BO...	PDEN_BI...	Reserved
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

Fields

Field	Description
31 PDEN_VREF	Controls VREF power during DEEP-SLEEP (always shut down during POWER-DOWN and DEEP-POWER-DOWN). 0 - VREF is powered on during low power mode. 1 - VREF is powered off during low power mode.
30 PDEN_OPAMP2	Controls Operational Amplifier2 power during DEEP-SLEEP (always shut down during POWER-DOWN and DEEP-POWER-DOWN). 0 - Operational Amplifier is powered on during low power mode. 1 - Operational Amplifier is powered off during low power mode.
29 PDEN_OPAMP1	Controls Operational Amplifier1 power during DEEP-SLEEP (always shut down during POWER-DOWN and DEEP-POWER-DOWN). 0 - Operational Amplifier is powered on during low power mode. 1 - Operational Amplifier is powered off during low power mode.
28 PDEN_OPAMP0	Controls Operational Amplifier0 power during DEEP-SLEEP (always shut down during POWER-DOWN and DEEP-POWER-DOWN). 0 - Operational Amplifier is powered on during low power mode. 1 - Operational Amplifier is powered off during low power mode.
27 PDEN_HSCMP2	Controls High Speed Comparator2 power during DEEP-SLEEP (always shut down during POWER-DOWN and DEEP-POWER-DOWN). 0 - High Speed Comparator is powered on during low power mode. 1 - High Speed Comparator is powered off during low power mode.

Table continues on the next page...

Table continued from the previous page...

Field	Description
26 PDEN_HSCMP1	Controls High Speed Comparator1 power during DEEP-SLEEP (always shut down during POWER-DOWN and DEEP-POWER-DOWN). 0 - High Speed Comparator is powered on during low power mode. 1 - High Speed Comparator is powered off during low power mode.
25 PDEN_HSCMP0	Controls High Speed Comparator0 power during DEEP-SLEEP (always shut down during POWER-DOWN and DEEP-POWER-DOWN). 0 - High Speed Comparator is powered on during low power mode. 1 - High Speed Comparator is powered off during low power mode.
24 PDEN_ROM	Controls ROM power during DEEP-SLEEP (ROM is always shut down during POWER-DOWN and DEEP-POWER-DOWN). 0 - ROM is powered on during low power mode. 1 - ROM is powered off during low power mode.
23 PDEN_PLL0_SSCG	Controls PLL0 Spread Sprectrum module power during DEEP-SLEEP (PLL0 Spread Spectrum is always shut down during POWER-DOWN and DEEP-POWER-DOWN). 0 - PLL0 Spread Sprectrum module is powered on during low power mode. 1 - PLL0 Spread Sprectrum module is powered off during low power mode.
22 —	Reserved
21 PDEN_LDOFLASHNV	Controls Flash NV (high voltage) LDO power during DEEP-SLEEP (always shut down during POWER-DOWN and DEEP-POWER-DOWN). 0 - Flash NV (high voltage) is powered on during low power mode. 1 - Flash NV (high voltage) is powered off during low power mode.
20 PDEN_LDOXTALHF	Controls High speed crystal LDO power during DEEP-SLEEP (always shut down during POWER-DOWN and DEEP-POWER-DOWN). 0 - High speed crystal LDO is powered on during low power mode. 1 - High speed crystal LDO is powered off during low power mode.
19 —	Reserved
18 PDEN_LDOEFUSEPROG	Controls USB high speed LDO power during DEEP-SLEEP (always shut down during POWER-DOWN and DEEP-POWER-DOWN). 0 - USB high speed LDO is powered on during low power mode. 1 - USB high speed LDO is powered off during low power mode.
17 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
16 PDEN_LDOMEM	Controls Memories LDO power during DEEP-SLEEP, POWER-DOWN and DEEP-POWER-DOWN. 0 - Memories LDO is powered on during low power mode. 1 - Memories LDO is powered off during low power mode.
15-14 —	Reserved
13 PDEN_COMP	Controls Analog Comparator power during DEEP-SLEEP and POWER-DOWN (always shut down during DEEP-POWER-DOWN). When the Analog Comparator is powered and used as a wake-up source in Power Down mode, the BIAS analog references must also be powered. See bit PDSLEEPCFG0[PDEN_BIAS]. 0 - Analog Comparator is powered on during low power mode. 1 - Analog Comparator is powered off during low power mode.
12 —	Reserved
11 PDEN_USBFSPHY	Controls USB Full Speed phy power during DEEP-SLEEP (always shut down during POWER-DOWN and DEEP-POWER-DOWN). 0 - USB Full Speed phy is powered on during low power mode. 1 - USB Full Speed phy is powered off during low power mode.
10 PDEN_PLL1	Controls USB PLL (also refered as PLL1) power during DEEP-SLEEP (always shut down during POWER-DOWN and DEEP-POWER-DOWN). 0 - USB PLL (also refered as PLL1) is powered on during low power mode. 1 - USB PLL (also refered as PLL1) is powered off during low power mode.
9 PDEN_PLL0	Controls System PLL (also refered as PLL0) power during DEEP-SLEEP (always shut down during POWER-DOWN and DEEP-POWER-DOWN). 0 - System PLL (also refered as PLL0) is powered on during low power mode. 1 - System PLL (also refered as PLL0) is powered off during low power mode.
8 PDEN_XTALHF	Controls high speed crystal power during DEEP-SLEEP (always shut down during POWER-DOWN and DEEP-POWER-DOWN). 0 - High speed crystal is powered on during low power mode. 1 - High speed crystal is powered off during low power mode.
7 PDEN_XTAL32K	Controls crystal 32 KHz power during DEEP-SLEEP, POWER-DOWN and DEEP-POWER-DOWN. 0 - crystal 32 KHz is powered on during low power mode. 1 - crystal 32 KHz is powered off during low power mode.
6	Controls power during DEEP-SLEEP, POWER-DOWN and DEEP-POWER-DOWN.

Table continues on the next page...

Table continued from the previous page...

Field	Description
PDEN_FRO32K	0 - FRO 32 KHz is powered on during low power mode. 1 - FRO 32 KHz is powered off during low power mode.
5 PDEN_FRO192M	Controls 192MHz Free Running Oscillator power during DEEP-SLEEP (always shut down during POWER-DOWN and DEEP-POWER-DOWN). 0 - FRO 192 MHz is powered on during low power mode. 1 - FRO 192 MHz is powered off during low power mode.
4 PDEN_FRO1M	Controls 1 MHz Free Running Oscillator power during DEEP-SLEEP, POWER-DOWN and DEEP-POWER-DOWN. 0 - FRO 1MHz is powered on during low power mode. 1 - FRO 1MHz is powered off during low power mode.
3 PDEN_BODVDDMAIN	Controls BOD_VDDMAIN power during DEEP-SLEEP and POWER-DOWN (always shut down during DEEP-POWER-DOWN). 0 - BOD_VDDMAIN is powered on during low power mode. 1 - BOD_VDDMAIN is powered off during low power mode.
2 PDEN_BODCORE	Controls Core Logic BoD power during DEEP-SLEEP and POWER-DOWN (always shut down during DEEP-POWER-DOWN). 0 - BOD_CORE is powered on during low power mode. 1 - BOD_CORE is powered off during low power mode.
1 PDEN_BIAS	Controls Analog Bias power during DEEP-SLEEP and POWER-DOWN (always shut down during DEEP-POWER-DOWN). 0 - Analog Bias is powered on during low power mode. 1 - Analog Bias is powered off during low power mode.
0 —	Reserved

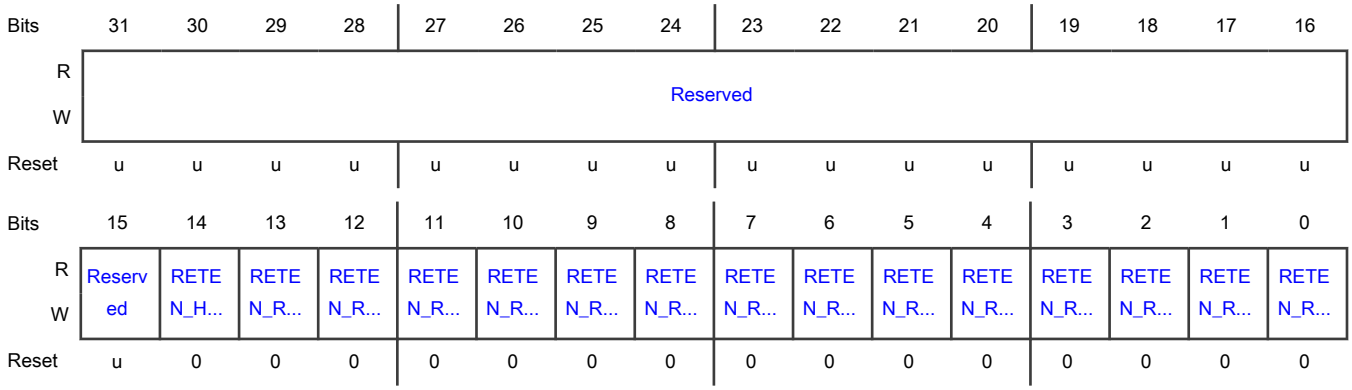
29.4.1.1.31 SRAM control (SRAMRETCTRL)

Controls all SRAM instances power down modes during Low Power modes. This register is reset by PoR, Pin Reset, Brown Out Detectors Reset, Software Reset.

Offset

Register	Offset
SRAMRETCTRL	B4h

Diagram



Fields

Field	Description
31-15 —	Reserved Read value is undefined, only zero should be written.
14 RETEN_H2PREG_FLEXSPI	Controls FlexSPI Dual Port Register Files power down modes during deep sleep. In power-down and deep power-down modes, FlexSPI Dual Port Register Files are always shutoff. 0 - DEEP-SLEEP: all FlexSPI dual port register files keep the configuration they had before entering DEEP-SLEEP. POWER-DOWN and DEEP-POWER-DOWN: all FlexSPI dual port register instances are shut off (In this mode there is no data retention). 1 - DEEP-SLEEP: all FlexSPI Dual Port egister files are in 'Power Down' mode (In this mode there is data retention). POWER-DOWN and DEEP-POWER-DOWN: all FlexSPI dual port register instances are shut off (In this mode there is no data retention).
13 RETEN_RAM_FLEXSPILPCACHE	Controls FlexSPI Cache SRAM power down modes during low power modes.
12 RETEN_RAM_FLASHLPCACHE	Controls Embedded Flash Cache SRAM power down modes during low power modes.
11 RETEN_RAM_43	Controls RAM_43 power down modes during low power modes. 0 - DEEP-SLEEP: the SRAM instance keeps the configuration it has before entering DEEP-SLEEP, what evere it is (Normal, Light Sleep, Deep-Sleep mode and Shut down modes) POWER-DOWN and DEEP-POWER-DOWN: the SRAM instance is in 'Shutdown mode' (In this mode there is no data retention). 1 - The SRAM is in 'Deep Sleep' mode (In this mode there is data retention).
10 RETEN_RAM_42	Controls RAM_42 power down modes during low power modes.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - DEEP-SLEEP: the SRAM instance keeps the configuration it has before entering DEEP-SLEEP, what evere it is (Normal, Light Sleep, Deep-Sleep mode and Shut down modes) POWER-DOWN and DEEP-POWER-DOWN: the SRAM instance is in 'Shutdown mode' (In this mode there is no data retention). 1 - The SRAM is in 'Deep Sleep' mode (In this mode there is data retention).
9 RETEN_RAM_41	Controls RAM_41 power down modes during low power modes. 0 - DEEP-SLEEP: the SRAM instance keeps the configuration it has before entering DEEP-SLEEP, what evere it is (Normal, Light Sleep, Deep-Sleep mode and Shut down modes) POWER-DOWN and DEEP-POWER-DOWN: the SRAM instance is in 'Shutdown mode' (In this mode there is no data retention). 1 - The SRAM is in 'Deep Sleep' mode (In this mode there is data retention).
8 RETEN_RAM_40	Controls RAM_40 power down modes during low power modes. 0 - DEEP-SLEEP: the SRAM instance keeps the configuration it has before entering DEEP-SLEEP, what evere it is (Normal, Light Sleep, Deep-Sleep mode and Shut down modes) POWER-DOWN and DEEP-POWER-DOWN: the SRAM instance is in 'Shutdown mode' (In this mode there is no data retention). 1 - The SRAM is in 'Deep Sleep' mode (In this mode there is data retention).
7 RETEN_RAM_30	Controls RAM_30 power down modes during low power modes. 0 - DEEP-SLEEP: the SRAM instance keeps the configuration it has before entering DEEP-SLEEP, whatever it is (Normal, Light Sleep, Deep-Sleep mode and Shut down modes) POWER-DOWN and DEEP-POWER-DOWN: the SRAM instance is in 'Shutdown mode' (In this mode there is no data retention). 1 - The SRAM is in 'Deep Sleep' mode (In this mode there is data retention).
6 RETEN_RAM_20	Controls RAM_20 power down modes during low power modes. 0 - DEEP-SLEEP: the SRAM instance keeps the configuration it has before entering DEEP-SLEEP, whatever it is (Normal, Light Sleep, Deep-Sleep mode and Shut down modes) POWER-DOWN and DEEP-POWER-DOWN: the SRAM instance is in 'Shutdown mode' (In this mode there is no data retention). 1 - The SRAM is in 'Deep Sleep' mode (In this mode there is data retention).
5 RETEN_RAM_10	Controls RAM_10 power down modes during low power modes. 0 - DEEP-SLEEP: the SRAM instance keeps the configuration it has before entering DEEP-SLEEP, whatever it is (Normal, Light Sleep, Deep-Sleep mode and Shut down modes) POWER-DOWN and DEEP-POWER-DOWN: the SRAM instance is in 'Shutdown mode' (In this mode there is no data retention). 1 - The SRAM is in 'Deep Sleep' mode (In this mode there is data retention).
4 RETEN_RAM_03	Controls RAM_03 power down modes during low power modes. 0 - DEEP-SLEEP: the SRAM instance keeps the configuration it has before entering DEEP-SLEEP, what evere it is (Normal, Light Sleep, Deep-Sleep mode and Shut down modes) POWER-DOWN and DEEP-POWER-DOWN: the SRAM instance is in 'Shutdown mode' (In this mode there is no data retention). 1 - The SRAM is in 'Deep Sleep' mode (In this mode there is data retention).
3 RETEN_RAM_02	Controls RAM_02 power down modes during low power modes.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - DEEP-SLEEP: the SRAM instance keeps the configuration it has before entering DEEP-SLEEP, what evere it is (Normal, Light Sleep, Deep-Sleep mode and Shut down modes) POWER-DOWN and DEEP-POWER-DOWN: the SRAM instance is in 'Shutdown mode' (In this mode there is no data retention). 1 - The SRAM is in 'Deep Sleep' mode (In this mode there is data retention).
2 RETEN_RAM_01	Controls RAM_01 power down modes during low power modes. 0 - DEEP-SLEEP: the SRAM instance keeps the configuration it has before entering DEEP-SLEEP, what evere it is (Normal, Light Sleep, Deep-Sleep mode and Shut down modes) POWER-DOWN and DEEP-POWER-DOWN: the SRAM instance is in 'Shutdown mode' (In this mode there is no data retention). 1 - The SRAM is in 'Deep Sleep' mode (In this mode there is data retention).
1 RETEN_RAM_00	Controls RAM_00 power down modes during low power modes. 0 - DEEP-SLEEP: the SRAM instance keeps the configuration it has before entering DEEP-SLEEP, what evere it is (Normal, Light Sleep, Deep-Sleep mode and Shut down modes) POWER-DOWN and DEEP-POWER-DOWN: the SRAM instance is in 'Shutdown mode' (In this mode there is no data retention). 1 - The SRAM is in 'Deep Sleep' mode (In this mode there is data retention).
0 RETEN_RAM_X0	Controls RAM_X0 power down modes during low power modes. 0 - DEEP-SLEEP: the SRAM instance keeps the configuration it has before entering DEEP-SLEEP, what evere it is (Normal, Light Sleep, Deep-Sleep mode and Shut down modes) POWER-DOWN and DEEP-POWER-DOWN: the SRAM instance is in 'Shutdown mode' (In this mode there is no data retention). 1 - The SRAM is in 'Deep Sleep' mode (In this mode there is data retention).

29.4.1.1.32 Power down run configuration 0 (PDRUNCFG0)

Controls the power to various analog blocks. This register is reset by (for all bit fields except PDEN_FRO32K and PDEN_XTAL32K) PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset. PDEN_FRO32K and PDEN_XTAL32K are reset by PoR and Brown Out Detectors.

Offset

Register	Offset
PDRUNCFG0	B8h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PDEN_VR...	PDEN_OP...	PDEN_OP...	PDEN_OP...	PDEN_HS...	PDEN_HS...	PDEN_HS...	Reserved	PDEN_PL...	Reserved	PDEN_LD...	PDEN_LD...	Reserved	PDEN_LD...	Reserved	PDEN_LD...
Reset	u	u	u	u	u	u	u	u	1	1	0	1	1	1	1	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved		PDEN_CO...	Reserved	PDEN_US...	PDEN_PL...	PDEN_PL...	PDEN_XT...	PDEN_XT...	PDEN_FR...	PDEN_FR...	Reserved	PDEN_BO...	PDEN_BO...	PDEN_BI...	PDEN_DC...
Reset	1	1	1	1	1	1	1	1	1	1	0	u	0	0	0	0

Fields

Field	Description
31 PDEN_VREF	Controls power to VREF module 0 - VREF is powered on. 1 - VREF is powered down.
30 PDEN_OPAMP2	Controls power to Operational Amplifier2 0 - Operational Amplifier2 is powered on 1 - Operational Amplifier2 is powered down
29 PDEN_OPAMP1	Controls power to Operational Amplifier1 0 - Operational Amplifier1 is powered on 1 - Operational Amplifier1 is powered down
28 PDEN_OPAMP0	Controls power to Operational Amplifier0 0 - Operational Amplifier0 is powered on. 1 - Operational Amplifier0 is powered down.
27 PDEN_HSCMP2	Controls power to High Speed Comparator2 0 - High Speed Comparator2 is powered on 1 - High Speed Comparator2 is powered down
26 PDEN_HSCMP1	Controls power to High Speed Comparator1 0 - High Speed Comparator1 is powered on 1 - High Speed Comparator1 is powered down
25 PDEN_HSCMP0	Controls power to High Speed Comparator0 0 - High Speed Comparator0 is powered on. 1 - High Speed Comparator0 is powered down.
24	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
23 PDEN_PLL0_SSCG	Controls power to System PLL (PLL0) Spread Spectrum module. 0 - PLL0 Sread spectrum module is powered. 1 - PLL0 Sread spectrum module is powered down.
22 —	Reserved
21 PDEN_LDOFLASHNV	Controls power to Flasn NV (high voltage) LDO. This bit has effects only when LDOFLASHNVPWRCTRL (in CTRL register) is set to '1'. 0 - Flash NV LDO is powered. 1 - Flash NV LDO is powered down.
20 PDEN_LDOXTALHF	Controls power to high speed crystal LDO. 0 - High speed crystal LDO is powered. 1 - High speed crystal LDO is powered down.
19 —	Reserved
18 PDEN_LDOEFUSEPROG	Controls power to OTP-eFUSE Programming LDO. 0 - USB high speed LDO is powered. 1 - USB high speed LDO is powered down.
17 —	Reserved Only zero should be written.
16 PDEN_LDOMEM	Controls power to Memories LDO. 0 - Memories LDO is powered. 1 - Memories LDO is powered down.
15-14 —	Reserved Read value is undefined, only zero should be written.
13 PDEN_COMP	Controls power to Analog Comparator. 0 - Analog Comparator is powered. 1 - Analog Comparator is powered down.
12 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
11 PDEN_USBFS PHY	Controls power to USB Full Speed phy. 0 - USB Full Speed phy is powered. 1 - USB Full Speed phy is powered down.
10 PDEN_PLL1	Controls power to USB PLL (also refered as PLL1). 0 - PLL1 is powered. 1 - PLL1 is powered down.
9 PDEN_PLL0	Controls power to System PLL (also refered as PLL0). 0 - PLL0 is powered. 1 - PLL0 is powered down.
8 PDEN_XTALHF	Controls power to high speed crystal. 0 - High speed crystal is powered. 1 - High speed crystal is powered down.
7 PDEN_XTAL32 K	Controls power to crystal 32 KHz. 0 - Crystal 32KHz is powered. 1 - Crystal 32KHz is powered down.
6 PDEN_FRO32K	Controls power to the Free Running Oscillator (FRO) 32 KHz. 0 - FRO32KHz is powered. 1 - FRO32KHz is powered down.
5 PDEN_FRO192 M	Controls power to the Free Running Oscillator (FRO) 192 MHz; The 12MHz, 48 MHz and 96 MHz clocks are derived from this FRO. 0 - FRO 192MHz is powered. 1 - FRO 192MHz is powered down.
4 —	Reserved Read value is undefined, only zero should be written.
3 PDEN_BODVD DMAIN	Controls power to VDDMAIN Brown Out Detector (BOD_VDDMAIN). 0 - BOD_VDDMAIN is powered. 1 - BOD_VDDMAIN is powered down.
2 PDEN_BODCO RE	Controls power to Core Brown Out Detector (BOD_CORE). 0 - BOD_CORE is powered. 1 - BOD_CORE is powered down.
1 PDEN_BIAS	Controls power to . 0 - Analog Bias is powered.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Analog Bias is powered down.
0 PDEN_DCDC	Controls power to Bulk DCDC Converter. This bit has effects only when DCDCPWRCTRL (in CTRL register) is set to '1'. 0 - DCDC is powered. 1 - DCDC is powered down.

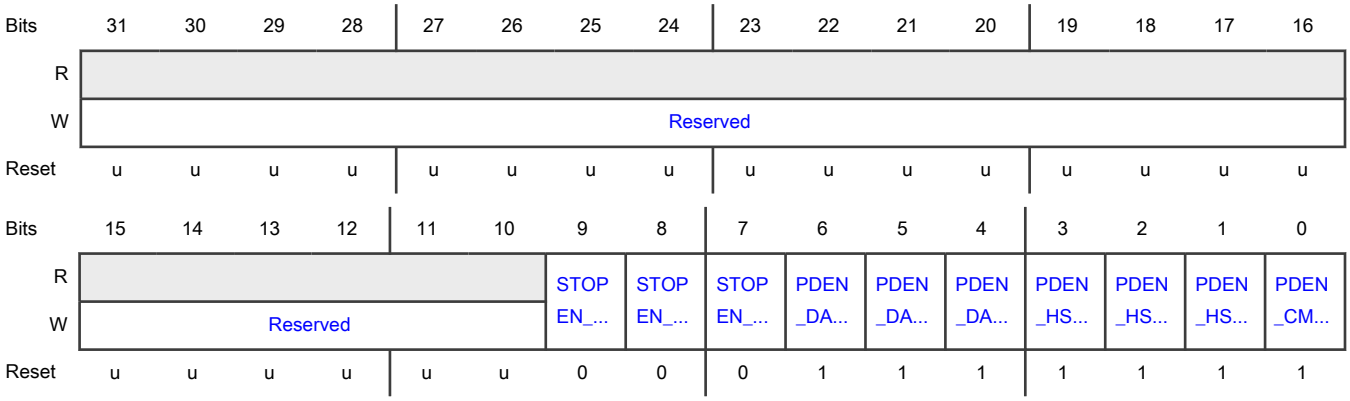
29.4.1.1.33 Power down run configuration 1 (PDRUNCFG1)

Controls the power to various analog blocks. This register is reset by PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset.

Offset

Register	Offset
PDRUNCFG1	BCh

Diagram



Fields

Field	Description
31-10 —	Reserved
9 STOPEN_DAC2	Controls DAC2 Stop mode. 0 - DAC2 Stop mode is disabled. 1 - DAC2 Stop mode is enabled.
8	Controls DAC1 Stop mode.

Table continues on the next page...

Table continued from the previous page...

Field	Description
STOPEN_DAC1	0 - DAC1 Stop mode is disabled. 1 - DAC1 Stop mode is enabled.
7 STOPEN_DAC0	Controls DAC0 Stop mode. 0 - DAC0 Stop mode is disabled. 1 - DAC0 Stop mode is enabled.
6 PDEN_DAC2	Controls power to DAC2. 0 - DAC2 is powered. 1 - DAC2 is powered down.
5 PDEN_DAC1	Controls power to DAC1. 0 - DAC1 is powered. 1 - DAC1 is powered down.
4 PDEN_DAC0	Controls power to DAC0. 0 - DAC0 is powered. 1 - DAC0 is powered down.
3 PDEN_HSCMP2_DAC	Controls power to High Speed Comparator2 DAC. 0 - High Speed Comparator2 DAC is powered. 1 - High Speed Comparator2 DAC is powered down.
2 PDEN_HSCMP1_DAC	Controls power to High Speed Comparator1 DAC. 0 - High Speed Comparator1 DAC is powered. 1 - High Speed Comparator1 DAC is powered down.
1 PDEN_HSCMP0_DAC	Controls power to High Speed Comparator0 DAC. 0 - High Speed Comparator0 DAC is powered. 1 - High Speed Comparator0 DAC is powered down.
0 PDEN_CMPBIAS	Controls power of Comparators 1/2/3 bias. 0 - Comparators 1/2/3 bias is powered. 1 - Comparators 1/2/3 bias is powered down.

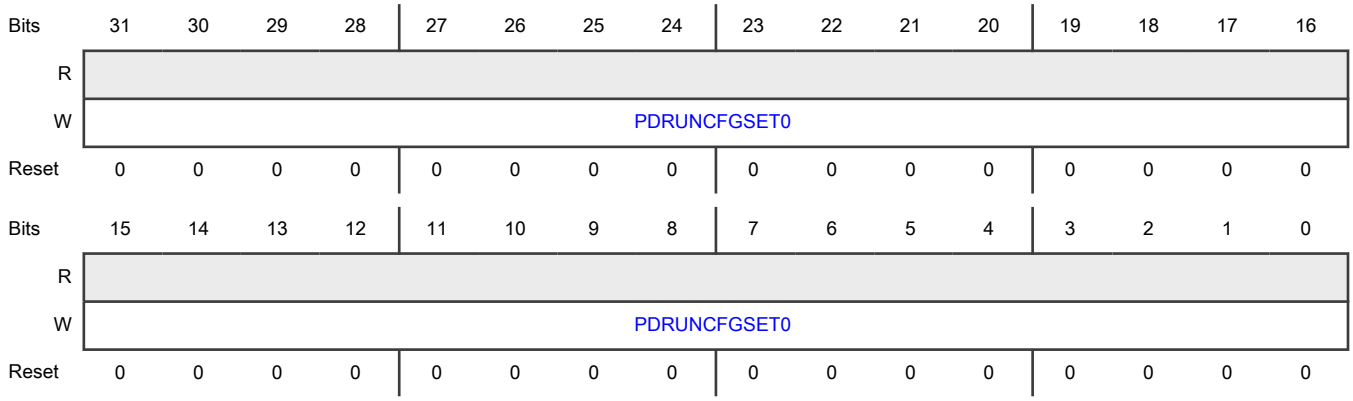
29.4.1.1.34 Power down run configuration set 0 (PDRUNCFGSET0)

Sets the power configuration 0 register [PDRUNCFG0].

Offset

Register	Offset
PDRUNCFGSET0	C0h

Diagram



Fields

Field	Description
31-0 PDRUNCFGSET0	Writing ones to this register sets the corresponding bit or bits in the PDRUNCFG0 register, if they are implemented.

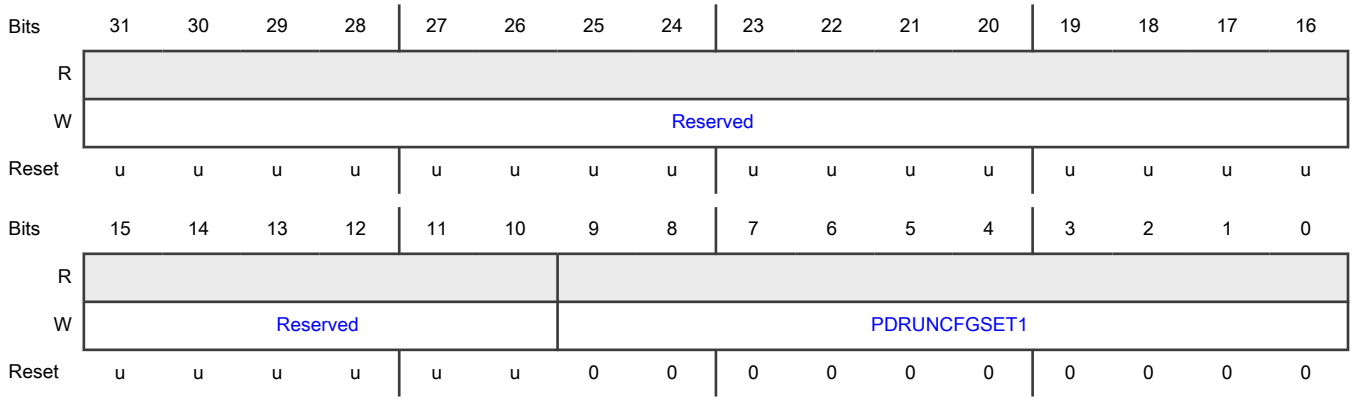
29.4.1.1.35 Power down run configuration set 1 (PDRUNCFGSET1)

Sets the power configuration 1 [PDRUNCFG1].

Offset

Register	Offset
PDRUNCFGSET1	C4h

Diagram



Fields

Field	Description
31-10 —	Reserved
9-0 PDRUNCFGSET1	Writing ones to this register sets the corresponding bit or bits in the PDRUNCFG0 register, if they are implemented.

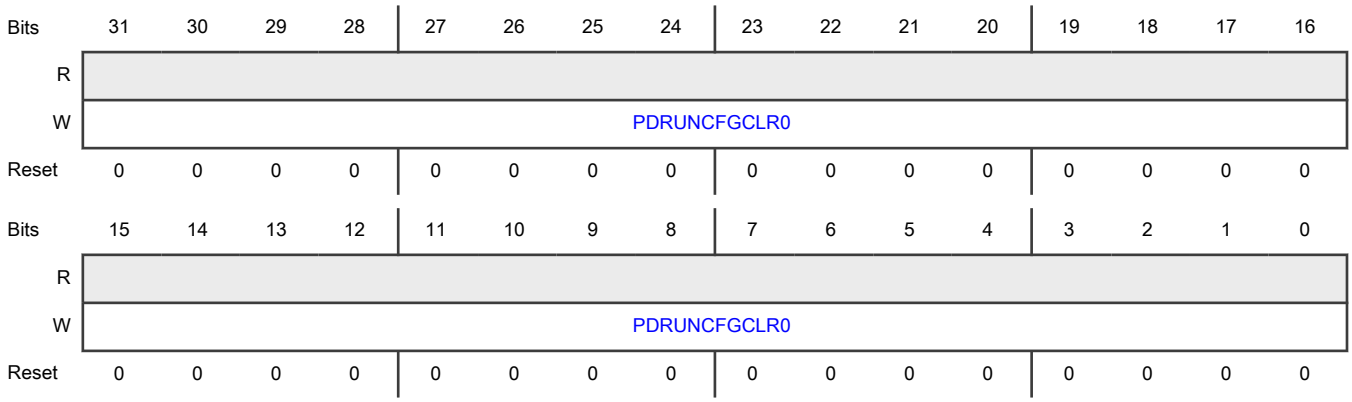
29.4.1.1.36 Power down run configuration clear 0 (PDRUNCFGCLR0)

Clears the power configuration register 0 [PDRUNCFG0].

Offset

Register	Offset
PDRUNCFGCLR0	C8h

Diagram



Fields

Field	Description
31-0 PDRUNCFGCLR0	Writing ones to this register clears the corresponding bit or bits in the PDRUNCFG0 register, if they are implemented.

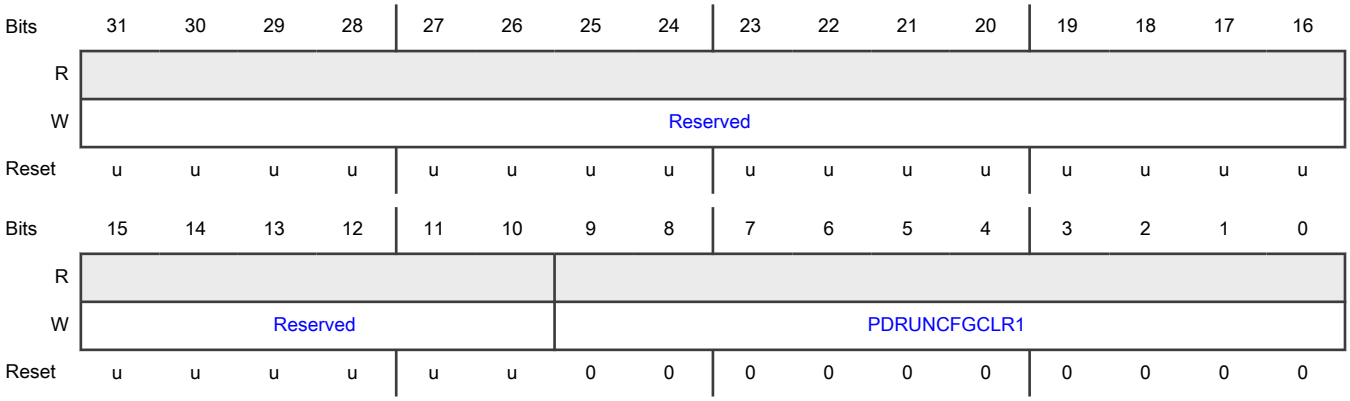
29.4.1.1.37 Power down run configuration clear 1 (PDRUNCFGCLR1)

Clears the power configuration register 1 [PDRUNCFG1].

Offset

Register	Offset
PDRUNCFGCLR1	CCh

Diagram



Fields

Field	Description
31-10 —	Reserved
9-0 PDRUNCFGCLR1	Writing ones to this register clears the corresponding bit or bits in the PDRUNCFG0 register, if they are implemented.

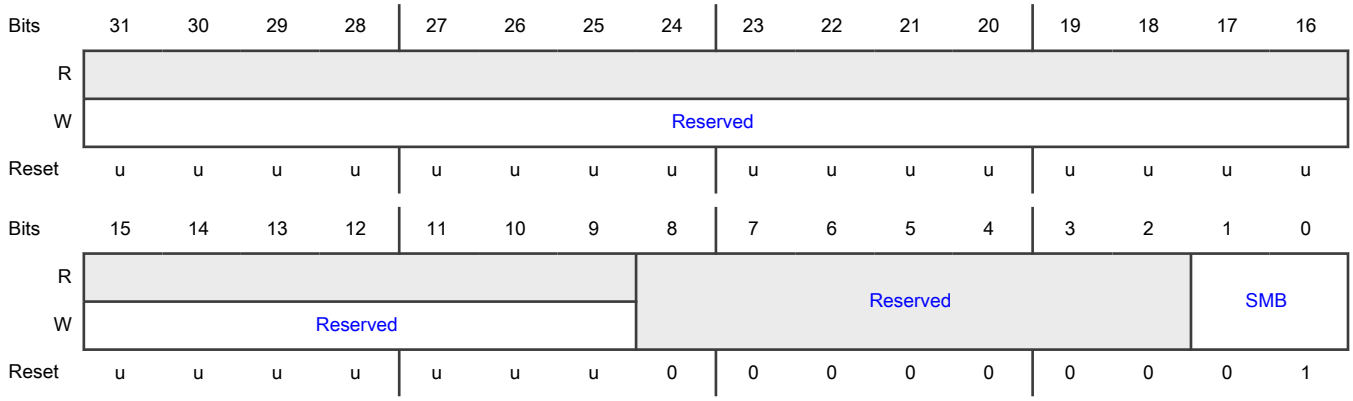
29.4.1.1.38 SRAM control (SRAMCTRL)

All SRAMs common control signals. This register is reset by PoR, Pin Reset, Brown Out Detectors Reset, Software Reset.

Offset

Register	Offset
SRAMCTRL	D4h

Diagram



Fields

Field	Description
31-9 —	Reserved Read value is undefined, only zero should be written.
8-2 —	Reserved
1-0 SMB	Source Biasing voltage. 00 - Low leakage. 01 - Medium leakage. 10 - Highest leakage. 11 - Disable.

29.4.1.1.39 SRAM control 0 (SRAMCTRL0)

RAM_X0, and RAM_00 to RAM_30 power modes controls. This register is reset by PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset.

When [LS, LSDEL, DSB, DSBDEL] is:

- 0011 : Normal Mode
- 1111 : Light sleep mode
- 0100 : Deep-sleep mode
- 1100 : Shut down Mode

Offset

Register	Offset
SRAMCTRL0	D8h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RAM_30_...	RAM_30_...	RAM_30_...	RAM_30_...	RAM_20_...	RAM_20_...	RAM_20_...	RAM_20_...	RAM_10_...	RAM_10_...	RAM_10_...	RAM_10_...	RAM_03_...	RAM_03_...	RAM_03_...	RAM_03_...
Reset	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RAM_02_...	RAM_02_...	RAM_02_...	RAM_02_...	RAM_01_...	RAM_01_...	RAM_01_...	RAM_01_...	RAM_00_...	RAM_00_...	RAM_00_...	RAM_00_...	RAM_X0_...	RAM_X0_...	RAM_X0_...	RAM_X0_...
Reset	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

Fields

Field	Description
31 RAM_30_LSDEL	RAM_30 Light Sleep mode delayed.
30 RAM_30_DSBDEL	RAM_30 Deep sleep delayed.
29 RAM_30_DSB	RAM_30 Deep sleep mode.
28 RAM_30_LS	RAM_30 Light Sleep mode.
27 RAM_20_LSDEL	RAM_20 Sleep mode disable.
26 RAM_20_DSBDEL	RAM_20 Deep sleep delayed.
25 RAM_20_DSB	RAM_20 Deep sleep mode.
24 RAM_20_LS	RAM_20 Light Sleep mode.
23	RAM_10 Sleep mode disable.

Table continues on the next page...

Table continued from the previous page...

Field	Description
RAM_10_LSDEL	
22 RAM_10_DSBDEL	RAM_10 Deep sleep delayed.
21 RAM_10_DSB	RAM_10 Deep sleep mode.
20 RAM_10_LS	RAM_10 Light Sleep mode.
19 RAM_03_LSDEL	RAM_03 Sleep mode disable.
18 RAM_03_DSBDEL	RAM_03 Deep sleep delayed.
17 RAM_03_DSB	RAM_03 Deep sleep mode.
16 RAM_03_LS	RAM_03 Light Sleep mode.
15 RAM_02_LSDEL	RAM_02 Sleep mode disable.
14 RAM_02_DSBDEL	RAM_02 Deep sleep delayed.
13 RAM_02_DSB	RAM_02 Deep sleep mode.
12 RAM_02_LS	RAM_02 Light Sleep mode.
11 RAM_01_LSDEL	RAM_01 Sleep mode disable.

Table continues on the next page...

Table continued from the previous page...

Field	Description
10 RAM_01_DSBDEL	RAM_01 Deep sleep delayed.
9 RAM_01_DSB	RAM_01 Deep sleep mode.
8 RAM_01_LS	RAM_01 Light Sleep mode.
7 RAM_00_LSDEL	RAM_00 Sleep mode disable.
6 RAM_00_DSBDEL	RAM_00 Deep sleep delayed.
5 RAM_00_DSB	RAM_00 Deep sleep mode.
4 RAM_00_LS	RAM_00 Light Sleep mode.
3 RAM_X0_LSDEL	RAM_X0 Sleep mode disable.
2 RAM_X0_DSBDEL	RAM_X0 Deep sleep delayed.
1 RAM_X0_DSB	RAM_X0 Deep sleep mode.
0 RAM_X0_LS	RAM_X0 Light Sleep mode.

29.4.1.1.40 SRAM control 1 (SRAMCTRL1)

RAM_40 to RAM_43 power modes controls. This register is reset by PoR, Pin Reset, Brown Out Detectors Reset, Deep Power Down Reset, Software Reset.

When [LS, LSDEL, DSB, DSBDEL] is:

- >0011 : Normal Mode

- 1111 : Light sleep mode
- 0100 : Deep-sleep mode
- 1100 : Shut down Mode

Offset

Register	Offset
SRAMCTRL1	DCh

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								RAM_FLE...	RAM_FLE...	RAM_FLE...	RAM_FLE...	RAM_FLA...	RAM_FLA...	RAM_FLA...	RAM_FLA...
W																
Reset	u	u	u	u	u	u	u	u	0	1	1	0	0	1	1	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RAM_43_...	RAM_43_...	RAM_43_...	RAM_43_...	RAM_42_...	RAM_42_...	RAM_42_...	RAM_42_...	RAM_41_...	RAM_41_...	RAM_41_...	RAM_41_...	RAM_40_...	RAM_40_...	RAM_40_...	RAM_40_...
W																
Reset	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

Fields

Field	Description
31-24 —	Reserved
23 RAM_FLEXSPILPCACHE_LSDEL	Flex SPI Cache RAM Sleep mode disable.
22 RAM_FLEXSPILPCACHE_DSBDEL	Flex SPI Cache RAM Deep sleep delayed.
21 RAM_FLEXSPILPCACHE_DSB	Flex SPI Cache RAM Deep sleep mode.
20	Flex SPI Cache RAM Light Sleep mode.

Table continues on the next page...

Table continued from the previous page...

Field	Description
RAM_FLEXSPILPCACHE_LS	
19 RAM_FLASHLPCACHE_LSDEL	Flash Cache RAM Sleep mode disable.
18 RAM_FLASHLPCACHE_DSBDEL	Flash Cache RAM Deep sleep delayed.
17 RAM_FLASHLPCACHE_DSB	Flash Cache RAM Deep sleep mode.
16 RAM_FLASHLPCACHE_LS	Flash Cache RAM Light Sleep mode.
15 RAM_43_LSDEL	RAM_43 Sleep mode disable.
14 RAM_43_DSBDEL	RAM_43 Deep sleep delayed.
13 RAM_43_DSB	RAM_43 Deep sleep mode.
12 RAM_43_LS	RAM_43 Light Sleep mode.
11 RAM_42_LSDEL	RAM_42 Sleep mode disable.
10 RAM_42_DSBDEL	RAM_42 Deep sleep delayed.
9 RAM_42_DSB	RAM_42 Deep sleep mode.

Table continues on the next page...

Table continued from the previous page...

Field	Description
8 RAM_42_LS	RAM_42 Light Sleep mode.
7 RAM_41_LSDEL	RAM_41 Sleep mode disable.
6 RAM_41_DSBDEL	RAM_41 Deep sleep delayed.
5 RAM_41_DSB	RAM_41 Deep sleep mode.
4 RAM_41_LS	RAM_41 Light Sleep mode.
3 RAM_40_LSDEL	RAM_40 Sleep mode disable.
2 RAM_40_DSBDEL	RAM_40 Deep sleep delayed.
1 RAM_40_DSB	RAM_40 Deep sleep mode.
0 RAM_40_LS	RAM_40 Light Sleep mode.

Chapter 30
Power Control API

30.1 Introduction

The power API functions are available in the SDK software package available on LPC553x-LPC55S3x|ARM Cortex-M33 32-bit Microcontrollers (MCUs)|NXP.

Control of device power consumption or entry to low power modes can be configured through simple calls to the low power profile API.

API sets up:

- reduced power modes
- on-chip power based on the expected operating frequency

30.1.1 Features

The power API functions fall into five categories:

- Power library initialization function & Device wake up Cause function
- Prepares the device to enter low power modes: SLEEP, DEEP-SLEEP, POWER-DOWN, DEEP-POWER-DOWN. See Low power entry functions.
- Core power domain supply source functions
- CPU and System frequency configuration
- Manages power consumption of device SRAM instances. See SRAM power modes control functions.

30.2 Functional description

The following sections describe functional details of the Power API functions.

30.2.1 Power library initialization function & Device wake up Cause function

30.2.1.1 POWER_PowerInit

30.2.1.1.1 Detailed Description

Table 245. POWER_PowerInit API

power_status_t	**POWER_PowerInit**	(	
			void
		)	

The POWER_PowerInit API :

- checks the status of the hardware boot sequence, then
- performs some power-related initialization of the system (like configuring some internal voltage regulators)

IMPORTANT

The POWER_PowerInit API must be called only once at the beginning of any application before calling any other *POWER_** API (like POWER_EnterSleep, POWER_SRAMPowerModeControl) ...

Table 246. Parameters

Parameters	Description
	This API does not take any parameter.

Table 247. Returned values

Return	Description
power_status_t	This API returns one of the two following values: • kPOWER_Status_Success: on success. • kPOWER_Status_Fail: on failure : at least one serious issue occured during the hardware boot sequence. It is recommended to re-initiate a full system software reset. See

30.2.1.1.2 Examples

```
/* Performs some power related initialization of the system ... */
if ( POWER_PowerInit() == kPOWER_Status_Success )
{
   /*
    * On success :
    * - Continue normal processing.
    * - Others POWER_* API functions can be called.
    */
}
else
{
   /* On Failure :
    * - stop processing: the system should be reintialized
    */
}
```

30.2.1.2 POWER_GetWakeUpCause

The POWER_GetWakeUpCause API returns some key information related to device events like the most recent reset cause, the most recent low power mode, and the most recent wake-up pins events.

30.2.1.2.1 Detailed Description

Table 248. POWER_GetWakeUpCause API

void	**POWER_GetWakeUpCause**	(		
			power_reset_cause_t *	**reset_cause**
			power_boot_mode_t *	**boot_mode**
			power_wakeup_pin_t *	**wakeup_pin_cause**
		)		

The POWER_GetWakeUpCause API returns some key information related to the device reset wake-up cause, for all power modes.

Table 249. Parameters

Parameters	Description
reset_cause	The device most recent reset event. The definition of this parameter is aligned with power_reset_cause_t type definition. **NOTE** In DEEP-POWER-DOWN mode, if several wake up sources have been enabled (for example, the RTC and a wake-up pin), the parameter contains the first event that occured. it is possible to know if the other enabled wake up event have occured by the reading directly PMC->AOREG1[DPDRESET_*] (DPDRESET_WAKEUPIO, DPDRESET_RTC and DPDRESET_OSTIMER).
boot_mode	The device most recent boot mode. The definition of this parameter is aligned with power_boot_mode_t type definition.
wakeup_pin_cause	The device most recent wake-up pin source. The definition of this parameter is aligned with power_wakeup_pin_t type definition. **NOTE** The value returned in this parameter is relevant *only if* the wake-up pins have been configured as wake-up source before to calling this function (via either POWER_SetWakeUpPins API or the parameter *wakeup_io_ctrl* of the POWER_EnterDeepPowerDown API. **NOTE** If several wake up pins have been enabled (for example wake-up pin 0, 3 and 4), the parameter contains the first wake up pin that triggered the device wake up. It is possible to know if the other enabled wake up pins event have occured by the reading directly PMC->WAKEIOCAUSE.

Table 250. Returned values

Return	Description
	This API does not return any value.

30.2.1.2.2 Examples

```
int main()
{
   power_reset_cause_t reset_cause;
   power_boot_mode_t boot_mode;
   power_wakeup_pin_t wakeup_pin_cause;

   if ( POWER_PowerInit() != kPOWER_Status_Success )
   {
      /* On failure, stop processing */
   }

   /* On success, continue normal processing ... */

   /* Get device wake up cause */
   POWER_GetWakeUpCause(&reset_cause, &boot_mode, &wakeup_pin_cause);
```

```
    if ( boot_mode == kBOOT_MODE_POWER_UP )
    { /* All wake up from non low power mode */

       if ( reset_cause == kRESET_CAUSE_POR )
       { /* Power On Reset */
          /* ... */

       }
       else
       {
          /* ... */
       }
    }
    else
    { /* Wake up from all low power modes */
       /* ... */

    }
}
```

30.2.2 Low power entry functions

Low power API provide functions to configure the system into the different low power modes: SLEEP, DEEP-SLEEP, POWER-DOWN and DEEP-POWER-DOWN.

30.2.2.1 POWER_EnterSleep

This routine puts the device in SLEEP low power mode: only the activity of the ARM CORTEX-M33 is stopped.

30.2.2.1.1 Detailed Description

Table 251. POWER_EnterSleep API

void	**POWER_EnterSleep**	(	
			void
		)	

SLEEP mode saves power by stopping CORTEX-M33 execution without affecting peripherals or requiring significant wake-up time.

The part wakes up from SLEEP mode after:

- any interrupt enabled in the CORTEX-M33 Nested Vector Interrupt Controller (NVIC) arrives at the processor.
- any chip reset occurs (Power on Reset, Brown-out detector reset, pin reset, ...)

Table 252. Parameters

Parameters	Description
	This API does not take any parameter.

Table 253. Returned values

Return	Description
	This API does not return any value.

Below is the implementation of the POWER_EnterSleep, along with some explanations for each line of code.

```
/**
 * @brief   Configures and enters in SLEEP low power mode
 * @return  Nothing
 */
void POWER_EnterSleep(void)
{
   uint32_t pmsk;
   pmsk = __get_PRIMASK();             /* Save CORTEX-M33 interrupt configuration */
   __disable_irq();                    /* Disable all interrupts */
   SCB->SCR &= ~SCB_SCR_SLEEPDEEP_Msk; /* CORTEX-M33 uses "Sleep" mode */
   __WFI();                            /* CORTEX-M33 enters "Sleep" mode */
   __set_PRIMASK(pmsk);                /* Restore CORTEX-M33 interrupt configuration (after wake up)
*/
}
```

30.2.2.1.2 Examples

```
   /* First, do some power related initialization of the system : */
   if ( POWER_PowerInit() != kPOWER_Status_Success )
   {
      /* On failure, stop processing */
   }

   /* On success, continue normal processing ... */

   /* ... then enter SLEEP low power mode. */
   POWER_EnterSleep();

   /* Processing continues after any (enabled) interrupt event, after the interrupt handler(s) has
been executed */
```

30.2.2.2 POWER_EnterDeepSleep

This routine prepares the part, then enter DEEP-SLEEP low power mode. the API function configures which analog/digital components remain running, so that an interrupt from one of these analog/digital peripherals can wake up the part.

30.2.2.2.1 Detailed Description

Table 254. POWER_EnterDeepSleep API

void	**POWER_EnterDeepSleep**	(		
			uint32_t	**exclude_from_pd[2]**
			uint32_t	**sram_retention_ctrl**
			uint32_t	**wakeup_interrupts[4]**
			uint32_t	**hardware_wake_ctrl**
		)		

The POWER_EnterDeepSleep API configures the DEEP-SLEEP low power mode: allows controlling which peripherals are powered up and and the SRAM instances low power mode during DEEP-SLEEP.

IMPORTANT

The POWER_EnterDeepSleep API switches the CPU & System Bus clock to 12 MHz. It is the responsibility of the user to reconfigure the CPU and System Bus clock after the function has returned.

Table 255. Parameters

Parameters	Description
exclude_from_pd[2]	Defines which analog peripherals shall NOT be powered down. It is a 2 x 32-bit vectors, with each bit of the vectors corresponding to one module. The definition of the vectors is given in Table 257 and Table 258, and is aligned with power_pd_bit_t type definition. For each bit field of the vector: • 0: the module is powered down during DEEP-SLEEP. • 1: the module is running during DEEP-SLEEP.
sram_retention_ctrl	Defines which SRAM instances will be in put SRAM Deep Sleep low power mode during DEEP-SLEEP. SRAM instances in SRAM Deep Sleep low power mode do not lose their content but they cannot be involved in any DMA transfer during DEEP-SLEEP. SRAM instances that are not required to be put in SRAM Deep Sleep low power mode during DEEP-SLEEP will keep the state they had before calling the API, meaning: • If the SRAM instance was in Active mode, it will stay in Active mode during DEEP-SLEEP and after wake up from DEEP-SLEEP. Such an SRAM instance can be involved in DMA transfer during DEEP-SLEEP • If the SRAM instance was in Deep Sleep mode, it will stay in SRAM Deep Sleep low power mode during DEEP-SLEEP and after wake up from DEEP-SLEEP. Such an SRAM instance cannot be involved in DMA transfer during DEEP-SLEEP. • If the SRAM instance was in Shutdown mode, it will stay in SRAM Shutdown low power mode during DEEP-SLEEP and after wake up from DEEP-SLEEP. Such an SRAM instance cannot be involved in DMA transfer during DEEP-SLEEP. The sram_retention_ctrl parameter is a 32-bit vector, with each bit of the vector corresponding to one SRAM instance. The definition of the vector is given in Table 259 and is aligned with power_sram_bit_t type definition. For each bit field of the vector: • 0: during DEEP-SLEEP, the SRAM instance keeps the state it has before entering DEEP-SLEEP. • 1: during DEEP-SLEEP, the SRAM instance will be in SRAM Deep Sleep low power mode (content preserved). **NOTE** When SDMA transfers are expected to occur during DEEP-SLEEP, make sure that the SRAM instances in which the *SDMA descriptors* and the Periphal's data are stored will stay in Active mode during DEEP-SLEEP.
wakeup_interrupts[4]	Defines which peripheral interrupts can be a wake-up source during DEEP-SLEEP. It is a 4 x 32-bit vectors, with each bit inside the vectors corresponding to one interrupt source. The definition of the vectors is given in Table 260,Table 261,Table 262, Table 263 and is aligned with Low Power Modes Wake up sources (#define WAKEUP_*) #defines definitions. For each bit field of the vectors:

Table continues on the next page...

Table 255. Parameters (continued)

	• 0: the associated peripheral cannot be a wake up source during DEEP-SLEEP • 1: the associated peripheral can be a wake up source during DEEP-SLEEP
hardware_wake_ctrl	Provides the possibility for all Flexcomm, High-Speed SPI, all DAC and the DMIC to have DMA service during DEEP-SLEEP without waking up entire device. The detailed mapping of the parameter is given in Table 264 and is aligned with the definition of #define LOWPOWER_HWWAKE_.

Table 256. Returned values

Return	Description
	This API does not return any value.

Table 257. Parameter exclude_from_pd[0] definition (1st vector)

Bit	Symbol	Description	Value
1:0	-	Reserved	Must be set to 0
2	BODCORE	Core Logic supply Brown-out Detector	• 0: Powered down • 1: Running
3	BODVDDMAIN	VDD_MAIN supply Brown-out Detector	• 0: Powered down • 1: Running
4	FRO1M	1-MHz Free Running Oscillator	• 0: Powered down • 1: Running
5	FRO192M	High Speed Free Running Oscillator	• 0: Powered down • 1: Running
6	FRO32K	32-KHz Free Running Oscillator	• 0: Powered down • 1: Running
7	XTAL32K	32-KHz Crystal Oscillator	• 0: Powered down • 1: Running
8	XTALHF	High Frequency Crystal Oscillator	• 0: Powered down • 1: Running
9	PLL0	Phase-Locked Loop module 0	• 0: Powered down • 1: Running
10	PLL1	Phase-Locked Loop module 1	• 0: Powered down • 1: Running
11	USBFSPHY	USB Full Speed Physical interface	• 0: Powered down

Table continues on the next page...

Table 257. Parameter exclude_from_pd[0] definition (1st vector) (continued)

Bit	Symbol	Description	Value
			• 1: Running
12	-	Reserved	Must be set to 0
13	COMP	Analog Comparator	• 0: Powered down • 1: Running
17:14	-	Reserved	Must be set to 0
18	LDOEFUSEPROG	OTP-eFUSE Programming Low Drop-Out Regulator	• 0: Powered down • 1: Running
19	-	Reserved	Must be set to 0
20	LDOXTALHF	High Frequency Crystal Oscillator Low Drop-Out Regulator	• 0: Powered down • 1: Running
22:21	-	Reserved	Must be set to 0
23	PLL0_SSCG	PLL0 Spread Spectrum Clock Generator	• 0: Powered down • 1: Running
24	-	Reserved	Must be set to 0
25	HSCMP0	High Speed Comparator 0	• 0: Powered down • 1: Running
26	HSCMP1	High Speed Comparator 1	• 0: Powered down • 1: Running
27	HSCMP2	High Speed Comparator 2	• 0: Powered down • 1: Running
28	OPAMP0	Operational Amplifier 0	• 0: Powered down • 1: Running
29	OPAMP1	Operational Amplifier 1	• 0: Powered down • 1: Running
30	OPAMP2	Operational Amplifier 2	• 0: Powered down • 1: Running
31	VREF	ADCs/DACs Analog Reference module.	• 0: Powered down • 1: Running

Table 258. Parameter exclude_from_pd[1] definition (2nd vector)

Bit	Symbol	Description	Value
0	CMPBIAS	High Speed Comparators Biasing	• 0: Powered down • 1: Running
1	HSCMP0_DAC	HSCMP0 (internal) DAC	• 0: Powered down • 1: Running
2	HSCMP1_DAC	HSCMP1 (internal) DAC	• 0: Powered down • 1: Running
3	HSCMP2_DAC	HSCMP2 (internal) DAC	• 0: Powered down • 1: Running
4	DAC0	Digital to Analog Converter 0	• 0: Powered down • 1: Running
5	DAC1	Digital to Analog Converter 1	• 0: Powered down • 1: Running
6	DAC2	Digital to Analog Converter 2	• 0: Powered down • 1: Running
7	STOP_DAC0	DAC0 Stop Mode	• 0: Stop Mode enabled • 1: Stop Mode disabled
8	STOP_DAC1	DAC1 Stop Mode	• 0: Stop Mode enabled • 1: Stop Mode disabled
9	STOP_DAC2	DAC2 Stop Mode	• 0: Stop Mode enabled • 1: Stop Mode disabled
31:10	-	Reserved	Must be set to 0

Table 259. Parameter sram_retention_ctrl

Bit	SRAM instance (SRAM size)	Value
0	RAM_X0 (16 KBytes)	• 0: Keeps state prior entering DEEP-SLEEP • 1: Deep Sleep (content preserved)
1	RAM_00 (4 KBytes)	• 0: Keeps state prior entering DEEP-SLEEP • 1: Deep Sleep (content preserved)

Table continues on the next page...

Table 259. Parameter sram_retention_ctrl (continued)

Bit	SRAM instance (SRAM size)	Value
2	RAM_01 (4 KBytes)	• 0: Keeps state prior entering DEEP-SLEEP • 1: Deep Sleep (content preserved)
3	RAM_02 (4 KBytes)	• 0: Keeps state prior entering DEEP-SLEEP • 1: Deep Sleep (content preserved)
4	RAM_03 (4 KBytes)	• 0: Keeps state prior entering DEEP-SLEEP • 1: Deep Sleep (content preserved)
5	RAM_10 (16 KBytes)	• 0: Keeps state prior entering DEEP-SLEEP • 1: Deep Sleep (content preserved)
6	RAM_20 (32 KBytes)	• 0: Keeps state prior entering DEEP-SLEEP • 1: Deep Sleep (content preserved)
7	RAM_30 (32 KBytes)	• 0: Keeps state prior entering DEEP-SLEEP • 1: Deep Sleep (content preserved)
8	RAM_40 (4 KBytes)	• 0: Keeps state prior entering DEEP-SLEEP • 1: Deep Sleep (content preserved)
9	RAM_41 (4 KBytes)	• 0: Keeps state prior entering DEEP-SLEEP • 1: Deep Sleep (content preserved)
10	RAM_42 (4 KBytes)	• 0: Keeps state prior entering DEEP-SLEEP • 1: Deep Sleep (content preserved)
11	RAM_43 (4 KBytes)	• 0: Keeps state prior entering DEEP-SLEEP • 1: Deep Sleep (content preserved)
31:12	Reserved	Must be set to 0

Table 260. Parameter wakeup_interrupts[0] (1st vector)

Bit	Symbol	Description	Value
0	SYS	Watchdog, Brown-out Detectors (BOD VDDMAIN and BOD Core)	• 0: interrupt disabled • 1: interrupt enabled
1	SDMA0	SDMA0 controller	• 0: interrupt disabled • 1: interrupt enabled
2	GLOBALINT0	Group GPIO Input Interrupt 0 (GINT0)	• 0: interrupt disabled • 1: interrupt enabled
3	GLOBALINT1	Group GPIO Input Interrupt 1 (GINT1)	• 0: interrupt disabled • 1: interrupt enabled
4	GPIO_INT0_0	Pin interrupt 0 or pattern match engine slice 0 (PINT0)	• 0: interrupt disabled • 1: interrupt enabled
5	GPIO_INT0_1	Pin interrupt 1 or pattern match engine slice 1 (PINT1)	• 0: interrupt disabled • 1: interrupt enabled
6	GPIO_INT0_2	Pin interrupt 2 or pattern match engine slice 2 (PINT2)	• 0: interrupt disabled • 1: interrupt enabled
7	GPIO_INT0_3	Pin interrupt 3 or pattern match engine slice 3 (PINT3)	• 0: interrupt disabled • 1: interrupt enabled
8	UTICK	Micro-tick timer	• 0: interrupt disabled • 1: interrupt enabled
9	-	Reserved	Must be set to 0
10	CTIMER0	Standard counter/timer CTIMER0	• 0: interrupt disabled • 1: interrupt enabled
11	CTIMER1	Standard counter/timer CTIMER1	• 0: interrupt disabled • 1: interrupt enabled
12	-	Reserved	Must be set to 0
13	CTIMER3	Standard counter/timer CTIMER3	• 0: interrupt disabled • 1: interrupt enabled
14	FLEXCOMM0	Flexcomm Interface 0 (USART, SPI, I2C)	• 0: interrupt disabled • 1: interrupt enabled
15	FLEXCOMM1	Flexcomm Interface 1 (USART, SPI, I2C)	• 0: interrupt disabled

Table continues on the next page...

Table 260. Parameter wakeup_interrupts[0] (1st vector) (continued)

Bit	Symbol	Description	Value
			• 1: interrupt enabled
16	FLEXCOMM2	Flexcomm Interface 2 (USART, SPI, I2C)	• 0: interrupt disabled • 1: interrupt enabled
17	FLEXCOMM3	Flexcomm Interface 3 (USART, SPI, I2C)	• 0: interrupt disabled • 1: interrupt enabled
18	FLEXCOMM4	Flexcomm Interface 4 (USART, SPI, I2C)	• 0: interrupt disabled • 1: interrupt enabled
19	FLEXCOMM5	Flexcomm Interface 5 (USART, SPI, I2C)	• 0: interrupt disabled • 1: interrupt enabled
20	FLEXCOMM6	Flexcomm Interface 6 (USART, SPI, I2C)	• 0: interrupt disabled • 1: interrupt enabled
21	FLEXCOMM7	Flexcomm Interface 7 (USART, SPI, I2C)	• 0: interrupt disabled • 1: interrupt enabled
23:22	-	Reserved	Must be set to 0
24	ACMP	Analog Comparator	• 0: interrupt disabled • 1: interrupt enabled
25	-	Reserved	Must be set to 0
26	HWVAD	Hardware Voice Activity Detector	• 0: interrupt disabled • 1: interrupt enabled
27	USB0_NEEDCLK	USB Full Speed	• 0: interrupt disabled • 1: interrupt enabled
28	-	Reserved	Must be set to 0
29	RTC	Real Time Clock (counter event and tamper event)	• 0: interrupt disabled • 1: interrupt enabled
31:30	-	Reserved	Must be set to 0

Table 261. Parameter wakeup_interrupts[1] (2nd vector)

Bit	Symbol	Description	Value
0	GPIO_INT0_4	Pin interrupt 4 or pattern match engine slice 4 (PINT4).	• 0: interrupt disabled • 1: interrupt enabled

Table continues on the next page...

Table 261. Parameter wakeup_interrupts[1] (2nd vector) (continued)

Bit	Symbol	Description	Value
1	GPIO_INT0_5	Pin interrupt 5 or pattern match engine slice 5 (PINT5).	• 0: interrupt disabled • 1: interrupt enabled
2	GPIO_INT0_6	Pin interrupt 6 or pattern match engine slice 6 (PINT6).	• 0: interrupt disabled • 1: interrupt enabled
3	GPIO_INT0_7	Pin interrupt 7 or pattern match engine slice 7 (PINT7).	• 0: interrupt disabled • 1: interrupt enabled
4	CTIMER2	Standard counter/timer CTIMER2.	• 0: interrupt disabled • 1: interrupt enabled
5	CTIMER4	Standard counter/timer CTIMER4.	• 0: interrupt disabled • 1: interrupt enabled
6	OS_EVENT_TIMER	OS Event Timer	• 0: interrupt disabled • 1: interrupt enabled
12:7	-	Reserved	Must be set to 0
13	-	Reserved	Must be set to 0
17:14	-	Reserved	Must be set to 0
18	-	Reserved	Must be set to 0
19	-	Reserved	Must be set to 0
25:20	-	Reserved	Must be set to 0
26	SDMA1	SDMA1 controller	• 0: interrupt disabled • 1: interrupt enabled
27	LSPI_HS	High Speed SPI	• 0: interrupt disabled • 1: interrupt enabled
29:28	-	Reserved	Must be set to 0
30	I3C	MIPI I3C	• 0: interrupt disabled • 1: interrupt enabled
31	-	Reserved	Must be set to 0

Table 262. Parameter wakeup_interrupts[2] (3rd vector)

Bit	Symbol	Description	Value
1:0	-	Reserved	Must be set to 0
2	-	Reserved	Must be set to 0

Table continues on the next page...

Table 262. Parameter wakeup_interrupts[2] (3rd vector) (continued)

Bit	Symbol	Description	Value
9:3	-	Reserved	Must be set to 0
10	DAC0	Digital to Analog Converter 0	• 0: interrupt disabled • 1: interrupt enabled
11	DAC1	Digital to Analog Converter 1	• 0: interrupt disabled • 1: interrupt enabled
12	DAC2	Digital to Analog Converter 2	• 0: interrupt disabled • 1: interrupt enabled
31:12	-	Reserved	Must be set to 0

Table 263. Parameter wakeup_interrupts[3] (4th vector)

Bit	Symbol	Description	Value
31:0	-	Reserved	Must be set to 0

Table 264. Parameter hardware_wake_ctrl

Bit	Symbol	Description	Value
0	-	Reserved	Must be set to 0
1	HWWAKE_PERIPHERALS	Wake for Flexcomms. Any Flexcomm FIFO reaching the level specified by its own TXLVL will cause peripheral clocking to wake up temporarily while the related status is asserted.	• 0: Disabled • 1: Enabled
2	HWWAKE_DMIC	Wake for Digital Microphone (DMIC). When 1, the DMIC input FIFO reaching the level specified by TRIGLVL of either channel will cause peripheral clocking to be enabled temporarily while the DMIC FIFO level is at or above TRIGLVL. This allows DMA to become active to move data out of the DMIC FIFO. Used in conjonction with LOWPOWER_HWWAKE_PERIPHERALS	• 0: Disabled • 1: Enabled
3	HWWAKE_SDMA0	Wake for SDMA0. SDMA0 being busy will cause	• 0: Disabled

Table continues on the next page...

Table 264. Parameter hardware_wake_ctrl (continued)

Bit	Symbol	Description	Value
		peripheral clocking to remain running until DMA completes. Used in conjunction with HWWAKE_PERIPHERALS.	• 1: Enabled
4	-	Reserved	Must be set to 0
5	HWWAKE_SDMA1	Wake for SDMA1. SDMA1 being busy will cause peripheral clocking to remain running until DMA completes. Used in conjunction with HWWAKE_PERIPHERALS.	• 0: Disabled • 1: Enabled
6	HWWAKE_DAC	Wake for DAC0, DAC1, DAC2. Any DAC0/1/2 FIFO reaching the level specified by the configuration will generate an asynchronous SDMA0 request, and SDMA0 will wake up the bus clock temporarily to transfer data to DAC0/1/2	• 0: Disabled • 1: Enabled
31:7	-	Reserved	Must be set to 0

30.2.2.2.2 Examples

In all examples presented herafter, we define the following variables :

```
uint32_t exclude_from_pd[2];   /* */
uint32_t sram_retention_ctrl;  /* */
uint32_t wakeup_interrupts[4]; /* */
uint32_t hardware_wake_ctrl;   /* */
```

30.2.2.2.2.1 Enter DEEP-SLEEP mode with wake up by RTC, using FRO32K as clock source, all SRAM instances in SRAM Deep Sleep power mode

```
void RTC_ALARM_IRQHandler(void); /* The RTC Alarm Interrupt Handler */

int main()
{
   /*
    * 1 - Do some power related initialization of the system :
    */
   if ( POWER_PowerInit() != kPOWER_Status_Success )
   {
      /* On failure, stop processing */
   }
```

```
    /* On success, continue normal processing ... */

    /*
     * 2 - Enable and configure the RTC Alarm module
     *   (Sleep duration, RTC Interrupt Handler ...)
     */
    /* RTC-> ... */

    /*
     * 3 - Set up DEEP-SLEEP low power parameters
     */

    /* The RTC uses the FRO32K as clock source ... */
    /* All others modules are disabled during DEEP-SLEEP */
    exclude_from_pd[0]   = kPDRUNCFG_PD_FRO32K;
    exclude_from_pd[1]   = 0UL;

    /* All SRAM in Deep Sleep low power mode. */
    sram_retention_ctrl  = kPOWER_SRAM_DSLP_MASK;

    /* Wake up upon RTC alarm interrupt ... */
    /* All other interrupts sources are disabled during DEEP-SLEEP */
    wakeup_interrupts[0] = WAKEUP_RTC_ALARM_WAKEUP;
    wakeup_interrupts[1] = 0UL;
    wakeup_interrupts[2] = 0UL;
    wakeup_interrupts[3] = 0UL;

    /* No SDMA transfer during DEEP-SLEEP */
    hardware_wake_ctrl   = 0UL;

    /*
     * 4 - Enter DEEP-SLEEP low power mode
     */
    POWER_EnterDeepSleep(exclude_from_pd, sram_retention_ctrl, wakeup_interrupts, hardware_wake_ctrl);

    /*
     * When the RTC interrupt raises, the system wakes up, then:
     * 1 - First, the RTC Interrupt Handler (RTC_ALARM_IRQHandler) is executed
     * 2 - Next, normal processing continues right after POWER_EnterDeepSleep call
     */

    /* ... */
}
```

30.2.2.2.2.2 Enter DEEP-SLEEP mode with wake up by System DMA0 (SDMA0)

```

void SDMA0_IRQHandler(void); /* The SDMA0 Interrupt Handler */

int main()
{
    /*
     * 1 - Do some power related initialization of the system :
     */
    if ( POWER_PowerInit() != kPOWER_Status_Success )
    {
```

```
      /* On failure, stop processing */
   }

   /* On success, continue normal processing ... */

   /*
    * 2 - Configure :
    *   --> any Flexcomm to receive some data via a serial interface (SPI, I2C)
    *   --> the SDMA0 to read the data from the flexcomm FIFO and store them in RAM_X0.
    *       A wake up event will be triggered when the required number of data has been
    *       stored in RAM_X0 by the SDMA0 engine.
    */
   /* SDMA0-> ... ; FLEXCOMM0-> */

   /*
    * 3 - Set up DEEP-SLEEP low power parameters
    */

   /* All analog modules are disabled during DEEP-SLEEP */
   exclude_from_pd[0]   = 0UL;
   exclude_from_pd[1]   = 0UL;

   /* All SRAM instances in Deep Sleep low power mode during DEEP-SLEEP,
    * except RAM_X0 which is involved in SDMA transfers:
    * (RAM_X0 is used to store both SDMA Descriptors and data from the Peripheral)
    * RAM_X0 will stay in Active mode (which is the state it has before calling
    * POWER_EnterDeepSleep).
    */
   sram_retention_ctrl  = kPOWER_SRAM_DSLP_MASK & (~kPOWER_SRAM_RAM_X0);

   /* Wake up upon SDMA0 interrupt ... */
   /* All other interrupt sources are disabled during DEEP-SLEEP */
   wakeup_interrupts[0] = WAKEUP_SDMA0;
   wakeup_interrupts[1] = 0UL;
   wakeup_interrupts[2] = 0UL;
   wakeup_interrupts[3] = 0UL;

   /* Allows DMA transfer during DEEP-SLEEP without leaving DEEP-SLEEP mode */
   hardware_wake_ctrl   = LOWPOWER_HWWAKE_SDMA0 | LOWPOWER_HWWAKE_PERIPHERALS;

   /*
    * 4 - Enter DEEP-SLEEP low power mode
    */
   POWER_EnterDeepSleep(exclude_from_pd, sram_retention_ctrl, wakeup_interrupts, hardware_wake_ctrl);

   /*
    * When the SDMA0 interrupt raises, the system wakes up, then:
    * 1 - First, the SDMA0 Interrupt Handler (SDMA0_IRQHandler) is executed
    * 2 - Next, normal processing continues right after POWER_EnterDeepSleep call
    */

   /* ... */
}
```

30.2.2.2.2.3 Enter DEEP-SLEEP mode with wake up by OS Event timer (FRO1M as clock source) or any General Purpose I/O (via GINT0 or GINT1), only RAM_20, RAM_30 and RAM_43 in SRAM Deep Sleep power mode

```
 void OSEVTIMER0_IRQHandler(void);      /* The OS Timer Interrupt Handler */
void GPIO_GLOBALINT0_IRQHandler(void); /* GINT0 Interrupt Handler */
void GPIO_GLOBALINT1_IRQHandler(void); /* GINT1 Interrupt Handler */

int main()
{
   /*
    * 1 - Do some power related initialization of the system :
    */
   if ( POWER_PowerInit() != kPOWER_Status_Success )
   {
      /* On failure, stop processing */
   }

   /* On success, continue normal processing ... */

   /*
    * 2 - Configure the OS Timer, GINT0 and GINT1 :
    *   --> Select the FRO1M as the clock source for the OS Timer
    *   --> Enable OS Timer clock, ...
    *   --> Set OS Timer Match Low and Match High registers ...
    *   --> ...
    */
   /* PMC->PDRUNCFGCLR0 = PMC_PDRUNCFG0_PDEN_XTAL32K_MASK;*/
   /* PMC->OSEVENTTIMER = ... ; */
   /* OSTIMER-> ... ; */
   /* GINT0->PORT_IPOL ... ; */
   /* GINT1->PORT_IPOL ... ; */

   /*
    * 3 - Set up DEEP-SLEEP low power parameters
    */

   /* All analog modules are shut down during DEEP-SLEEP,
    * except FRO1M as it is the clock source of the OS Timer.
    */
   exclude_from_pd[0]   = kPDRUNCFG_PD_FRO1M;
   exclude_from_pd[1]   = 0UL;

   /* All SRAM instances keep the power mode they had before
    * entering DEEP-SLEEP except RAM_20, RAM_30 and RAM_43 which
    * are in Deep Sleep SRAM low power mode (data retention)
    * during DEEP-SLEEP.
    */
   sram_retention_ctrl  = kPOWER_SRAM_RAM_20 | kPOWER_SRAM_RAM_30 | kPOWER_SRAM_RAM_43;

   /* Wake up upon OS Timer, GINT0 or GINT1 interrupts ... */
   /* All other interrupts sources are disabled during DEEP-SLEEP */
   wakeup_interrupts[0] = WAKEUP_GPIO_GLOBALINT0 | WAKEUP_GPIO_GLOBALINT1;
   wakeup_interrupts[1] = WAKEUP_OS_EVENT_TIMER;
   wakeup_interrupts[2] = 0UL;
   wakeup_interrupts[3] = 0UL;

   /* No SDMA transfer during DEEP-SLEEP */
```

```
    hardware_wake_ctrl   = 0UL;

    /*
     * 4 - Enter DEEP-SLEEP low power mode
     */
    POWER_EnterDeepSleep(exclude_from_pd, sram_retention_ctrl, wakeup_interrupts, hardware_wake_ctrl);

    /*
     * When the OS Timer or GINT0 or GINT1 interrupt raises, the system wakes up, then:
     * 1 - First, the relevant Interrupt Handler (OSEVTIMER0_IRQHandler or
           GPIO_GLOBALINT0_IRQHandler or GPIO_GLOBALINT1_IRQHandler) is executed.
     * 2 - Next, normal processing continues right after POWER_EnterDeepSleep call
     */

    /* ... */
}
```

30.2.2.3 POWER_EnterPowerDown

This routine prepares the part, then enter POWER-DOWN low power mode. the API function configures which analog/digital components remain running, so that an interrupt from one of these analog/digital peripherals can wake up the part.

30.2.2.3.1 Detailed Description

Table 265. POWER_EnterPowerDown API

void	**POWER_EnterPowerDown**	(		
			uint32_t	**exclude_from_pd[1]**
			uint32_t	**sram_retention_ctrl**
			uint32_t	**wakeup_interrupts[2]**
			uint32_t	**cpu_retention_addr**
		)		

The POWER_EnterPowerDown API configures the POWER-DOWN low power mode: allows controlling which peripherals are powered up and which SRAM instances are in retention state during POWER-DOWN.

When a wake-up event occurs, CPU will resume code execution after the call to the low power API function. However, during POWER-DOWN, a couple of modules will lose their states and therefore, they must be reconfigured by the user application. See .

IMPORTANT

It is the responsibility of the user to make sure that the SRAM instance(s) containing the application software stacks and variables will be preserved during POWER-DOWN (see parameter "sram_retention_ctrl").

IMPORTANT

The POWER_EnterPowerDown API switches the CPU & System Bus clock to 12 MHz. It is the responsibility of the user to reconfigure the CPU and System Bus clock after the function has returned.

IMPORTANT

During POWER-DOWN, a couple of modules will lose their state (see) . It is the responsibility of the user application to reconfigure them if the application requires them.

Table 266. Parameters

Parameters	Description
exclude_from_pd[1]	Defines which analog peripherals shall NOT be powered down. It is a 1 x 32-bit vector, with each bit of the vector corresponding to one module. The definition of the vector is given in Table 268, and is aligned with power_pd_bit_t type definition. For each bit field of the vector: • 0: the module is powered down during POWER-DOWN. • 1: the module is running during POWER-DOWN.
sram_retention_ctrl	Defines which SRAM instances will preserve their content during deep POWER-DOWN. The sram_retention_ctrl parameter is a 32-bit vector, with each bit of the vector corresponding to one SRAM instance. The definition of the vector is given in Table 269, and is aligned with power_sram_bit_t type definition. For each bit field of the vector: • 0: during POWER-DOWN, the SRAM instance will be in SRAM Shut Down low power mode (content not preserved). • 1: during POWER-DOWN, the SRAM instance will be in SRAM Deep Sleep low power mode (content preserved). **NOTE** The SRAM instance which is used to save CORTEX-M33 will be automatically put in Deep Sleep low power mode (content preserved) by the Low Power API (see parameter *cpu_retention_addr* just below.)
wakeup_interrupts[2]	Defines which peripheral interrupts can be a wake-up source during POWER-DOWN. It is a 2 x 32-bit vectors, with each bit inside the vectors corresponding to one interrupt source. The definition of the vectors is given in Table 270 and Table 271, and is aligned with Low Power Modes Wake up sources (#define WAKEUP_*) #defines definitions. For each bit field of the vectors: • 0: the associated peripheral cannot be a wake up source during POWER-DOWN • 1: the associated peripheral can be a wake up source during POWER-DOWN **IMPORTANT** In case Flexcomm3 UART is used as wake up source, a 32-KHz clock source (either FRO32K or XTAL32K) need to be enabled during POWER-DOWN. The unique baudrate supported is 9600 Baud.
cpu_retention_addr	Defines the start address of the area inside SRAM region where the CORTEX-M33 state will be saved during POWER-DOWN: • Must be word-aligned (address ending by 0x0, 0x4, 0x8 and 0xC) • Can be any value: — Between 0x20000000 and 0x200009FC (inside RAM_00) or — Between 0x20001000 and 0x200019FC (inside RAM_01) or — Between 0x20002000 and 0x200029FC (inside RAM_02) or — Between 0x20003000 and 0x200039FC (inside RAM_03). If the parameter does not meet the boundaries mentioned above, it will be set to 0x20000000 (start of RAM_00) by the Low Power API.

Table continues on the next page...

Table 266. Parameters (continued)

IMPORTANT

The states of the CORTEX-M33 will be stored in SRAM from *cpu_retention_addr* to *cpu_retention_addr + 1540 Bytes - 1*. Therefore, any user data present in this area before calling the function will be overwritten and definitely lost.

Table 267. Returned values

Return	Description
	This API does not return any value.

Table 268. Parameter exclude_from_pd[0] definition

Bit	Symbol	Description	Value
3:0	-	Reserved	Must be set to 0
4	FRO1M	1-MHz Free Running Oscillator	• 0: Powered down • 1: Running
5	-	Reserved	Must be set to 0
6	FRO32K	32-kHz Free Running Oscillator	• 0: Powered down • 1: Running
7	XTAL32K	32-kHz Crystal Oscillator	• 0: Powered down • 1: Running
12:8	-	Reserved	Must be set to 0
13	COMP	Analog Comparator	• 0: Powered down • 1: Running
30:14	-	Reserved	Must be set to 0
31	VREF	ADCs/DACs Analog Reference module.	• 0: Powered down • 1: Running

Table 269. Parameter sram_retention_ctrl

Bit	SRAM instance (SRAM size)	Value
0	RAM_X0 (16 KBytes)	• 0: Shut Down (content not preserved) • 1: Deep Sleep (content preserved)
1	RAM_00 (4 KBytes)	• 0: Shut Down (content not preserved) • 1: Deep Sleep (content preserved)
2	RAM_01 (4 KBytes)	• 0: Shut Down (content not preserved)

Table continues on the next page...

Table 269. Parameter sram_retention_ctrl (continued)

Bit	SRAM instance (SRAM size)	Value
		• 1: Deep Sleep (content preserved)
3	RAM_02 (4 KBytes)	• 0: Shut Down (content not preserved) • 1: Deep Sleep (content preserved)
4	RAM_03 (4 KBytes)	• 0: Shut Down (content not preserved) • 1: Deep Sleep (content preserved)
5	RAM_10 (16 KBytes)	• 0: Shut Down (content not preserved) • 1: Deep Sleep (content preserved)
6	RAM_20 (32 KBytes)	• 0: Shut Down (content not preserved) • 1: Deep Sleep (content preserved)
7	RAM_30 (32 KBytes)	• 0: Shut Down (content not preserved) • 1: Deep Sleep (content preserved)
8	RAM_40 (4 KBytes)	• 0: Shut Down (content not preserved) • 1: Deep Sleep (content preserved)
9	RAM_41 (4 KBytes)	• 0: Shut Down (content not preserved) • 1: Deep Sleep (content preserved)
10	RAM_42 (4 KBytes)	• 0: Shut Down (content not preserved) • 1: Deep Sleep (content preserved)
11	RAM_43 (4 KBytes)	• 0: Shut Down (content not preserved) • 1: Deep Sleep (content preserved)
31:12	Reserved	Must be set to 0

Table 270. Parameter wakeup_interrupts[0] (1st vector)

Bit	Symbol	Description	Value
1:0	-	Reserved	Must be set to 0
2	GLOBALINT0	Group GPIO Input Interreupt 0 (GINT0)	• 0: interrupt disabled • 1: interrupt enabled
3	GLOBALINT1	Group GPIO Input Interrupt 1 (GINT1)	• 0: interrupt disabled • 1: interrupt enabled
16:4	-	Reserved	Must be set to 0
17	FLEXCOMM3	Any module inside Flexcomm3 (UART, SPI, I2C)	• 0: interrupt disabled

Table continues on the next page...

Table 270. Parameter wakeup_interrupts[0] (1st vector) (continued)

Bit	Symbol	Description	Value
			• 1: interrupt enabled
23:18	-	Reserved	Must be set to 0
24	ACMP	Analog Comparator	• 0: interrupt disabled • 1: interrupt enabled
28:25	-	Reserved	Must be set to 0
29	RTC	Real Time Clock (counter event and tamper event)	• 0: interrupt disabled • 1: interrupt enabled
31:30	-	Reserved	Must be set to 0

Table 271. Parameter wakeup_interrupts[1] (2nd vector)

Bit	Symbol	Description	Value
5:0	-	Reserved	Must be set to 0
6	OSTIMER	OS Event Timer	• 0: interrupt disabled • 1: interrupt enabled
31:7	-	Reserved	Must be set to 0

30.2.2.3.2 Examples

In all examples presented herafter, we define the following variables :

```
uint32_t exclude_from_pd[1];   /* */
uint32_t sram_retention_ctrl;  /* */
uint32_t wakeup_interrupts[2]; /* */
uint32_t cpu_retention_addr;   /* */
```

30.2.2.3.2.1 Enter POWER-DOWN with wake up by RTC, using FRO32K as clock source, RAM_01 in Deep Sleep mode (data retention), all other SRAM instances in Shut down mode (no data retention)

```
 void RTC_ALARM_IRQHandler(void); /* The RTC Alarm Interrupt Handler */

int main()
{
   /*
    * 1 - Do some power related initialization of the system :
    */
   if ( POWER_PowerInit() != kPOWER_Status_Success )
   {
      /* On failure, stop processing */
   }
```

```
    /* On success, continue normal processing ... */

    /*
     * 2 - Enable and configure the RTC Alarm module
     *   (Sleep duration, RTC Interrupt Handler ...)
     */
    /* RTC-> ... */

    /*
     * 3 - Set up POWER-DOWN low power parameters
     */

    /* All analog modules are shut down during POWER-DOWN,
     * except FRO32K as it is the clock source of the OS Timer.
     */
    exclude_from_pd[0]   = kPDRUNCFG_PD_FRO32K;

    /* All SRAM instances in Shut down low power mode (no data
     * retention, low leakage) except RAM_01 which
     * is in Deep Sleep mode (data retention) during POWER-DOWN
     * because in this example, it is used to store application
     * stack, heap, variables ...
     * Note: RAM_00 will automically be put in Deep Sleep mode
     * (data retention) by the Low Power API because it is used
     * to save CORTEX-M33 state (see parameter cpu_retention_addr).
     */
    sram_retention_ctrl  = kPOWER_SRAM_RAM_01;

    /* Wake up upon RTC Alarm interrupts.
     * All other interrupts sources are disabled during POWER-DOWN
     */
    wakeup_interrupts[0] = WAKEUP_RTC_ALARM_WAKEUP;
    wakeup_interrupts[1] = 0UL;

    /*
     * CPU retention Area in SRAM:
     * The CORTEX-M33 state will be stored from 0x2000_0000
     * to 0x2000_0603; therefore, no application data (stack, heap,
     * variables ...) shall be stored in this address range.
     *
     */
    cpu_retention_addr  = 0x20000000; /* Start of RAM_00 */

    /*
     * 4 - Enter POWER-DOWN low power mode
     */
    POWER_EnterPowerDown(exclude_from_pd, sram_retention_ctrl, wakeup_interrupts, cpu_retention_addr);

    /*
     * When the RTC interrupt raises, the system wakes up, then:
     * 1 - First, the RTC Interrupt Handler (RTC_ALARM_IRQHandler) is executed
     * 2 - Next, normal processing continues right after POWER_EnterPowerDown call
     */

    /* ... */
}
```

30.2.2.3.2.2 Enter POWER-DOWN with wake up by any GPIO via GINT0/GINT1, RAM_00 and RAM_01 in Deep Sleep mode (data retention), all other SRAM instances in Shut down mode (no data retention)

```
void GPIO_GLOBALINT0_IRQHandler(void); /* GINT0 Interrupt Handler */
void GPIO_GLOBALINT1_IRQHandler(void); /* GINT1 Interrupt Handler */

int main()
{
   /*
    * 1 - Do some power related initialization of the system :
    */
   if ( POWER_PowerInit() != kPOWER_Status_Success )
   {
      /* On failure, stop processing */
   }

   /* On success, continue normal processing ... */

   /*
    * 2 - Configure the GINT0 and GINT1 :

   /* GINT0->PORT_IPOL ... ; */
   /* GINT1->PORT_IPOL ... ; */

   /*
    * 3 - Set up POWER-DOWN low power parameters
    */

   /*
    * All analog modules are shut down during POWER-DOWN
    */
   exclude_from_pd[0]   = 0UL;

   /* All SRAM instances in Shut down low power mode (no data
    * retention, low leakage) except RAM_00 and RAM_01 which
    * are in Deep Sleep (data retention) during POWER-DOWN.
    * because in this example, RAM_00/RAM_01 are used for
    * application stack, heap, variables ...
    * Note: RAM_03 will automically be put in Deep Sleep mode
    * (data retention) by the Low Power API because it is used
    * to save CORTEX-M33 state (see parameter cpu_retention_addr).
    */
   sram_retention_ctrl  = kPOWER_SRAM_RAM_00 | kPOWER_SRAM_RAM_01;

   /* Wake up upon GINT0/GINT1 interrupts.
    * All other interrupts sources are disabled during POWER-DOWN
    */
   wakeup_interrupts[0] = WAKEUP_GPIO_GLOBALINT0 | WAKEUP_GPIO_GLOBALINT1;
   wakeup_interrupts[1] = 0UL;

   /*
    * CPU retention Area in SRAM:
    * The CORTEX-M33 state will be stored from 0x2000_3800
    * to 0x2000_3E03; therefore, no application data (stack, heap,
    * variables ...) shall be stored in this address range.
    *
    */
   cpu_retention_addr  = 0x20003800; /* Bottom of RAM_03 */
```

```
    /*
     * 4 - Enter POWER-DOWN low power mode
     */
    POWER_EnterPowerDown(exclude_from_pd, sram_retention_ctrl, wakeup_interrupts, cpu_retention_addr);

    /*
     * When the GINT0 or GINT1 interrupt raises, the system wakes up, then:
     * 1 - First, the GPIO_GLOBALINT0_IRQHandler or GPIO_GLOBALINT1_IRQHandler is executed
     * 2 - Next, normal processing continues right after POWER_EnterPowerDown call
     */

    /* ... */
}
```

30.2.2.3.2.3 Enter POWER-DOWN with wake up by Flexcomm3 (SPI or I²C), all SRAM instances in Deep Sleep mode (data retention)

```
void FC3_IRQHandler(void); /* Flexcomm3 SPI Interrupt Handler (SPI or I2C) */

int main()
{
    /*
     * 1 - Do some power related initialization of the system :
     */
    if ( POWER_PowerInit() != kPOWER_Status_Success )
    {
        /* On failure, stop processing */
    }

    /* On success, continue normal processing ... */

    /*
     * 2 - Configure the Flexcomm3 (as either SPI or I2C) :

    /* ... */

    /*
     * 3 - Set up POWER-DOWN low power parameters
     */

    /*
     * All analog modules are shut down during POWER-DOWN
     */
    exclude_from_pd[0]   = 0UL;

    /* All SRAM instances are in Deep Sleep (data retention) during POWER-DOWN.
     * In this example, all other SRAM are used for application stack,
     * heap, variables ... except RAM_01.
     * Note: RAM_01 will automically be put in Deep Sleep mode
     * (data retention) by the Low Power API because it is used
     * to save CORTEX-M33 state (see parameter cpu_retention_addr).
     */
    sram_retention_ctrl  = kPOWER_SRAM_PDWN_MASK;
```

```
    /* Wake up upon Flexcomm3 interrupt.
     * All other interrupts sources are disabled during POWER-DOWN
     */
    wakeup_interrupts[0] = WAKEUP_FLEXCOMM3;
    wakeup_interrupts[1] = 0UL;

    /*
     * CPU retention Area in SRAM:
     * The CORTEX-M33 state will be stored from 0x2000_1000
     * to 0x2000_1603; therefore, no application data (stack, heap,
     * variables ...) shall be stored in this address range.
     *
     */
    cpu_retention_addr  = 0x20001000; /* Start of RAM_01 */

    /*
     * 4 - Enter POWER-DOWN low power mode
     */
    POWER_EnterPowerDown(exclude_from_pd, sram_retention_ctrl, wakeup_interrupts, cpu_retention_addr);

    /*
     * When the FLEXCOMM3 interrupt raises, the system wakes up, then:
     * 1 - First, the FC3_IRQHandler is executed
     * 2 - Next, normal processing continues right after POWER_EnterPowerDown call
     */

    /* ... */
}
```

30.2.2.3.2.4 Enter POWER-DOWN with wake up by either the Analog Comparator or the OS Event Timer (using FRO1M as clock source), RAM_00, RAM_01, RAM_20 and RAM_30 in Deep Sleep mode (data retention), all other SRAM instances in Shut down mode (no data retention)

```
void ACMP_IRQHandler(void);       /* Analog Comparator Interrupt Handler */
void OSEVTIMER0_IRQHandler(void); /* OS Timer Interrupt Handler */

int main()
{
    /*
     * 1 - Do some power related initialization of the system :
     */
    if ( POWER_PowerInit() != kPOWER_Status_Success )
    {
        /* On failure, stop processing */
    }

    /* On success, continue normal processing ... */

    /*
     * 2 - Configure the Analog Comparator and the OS Timer :

    /* PMC->COMP =    */
    /* SYSCON->COMP_INT_CTRL  */

    /* PMC->OSEVENTTIMER = ... ; */
```

```
    /* OSTIMER-> ... ; */

    /*
     * 3 - Set up POWER-DOWN low power parameters
     */

    /* All analog modules are shut down during POWER-DOWN,
     * except FRO1M (as it is the clock source of the OS Timer)
     * the Analog Comparator.
     */
    exclude_from_pd[0]   = kPDRUNCFG_PD_COMP | kPDRUNCFG_PD_FRO1M;


    /* All SRAM instances in Shut down low power mode (no data
     * retention, low leakage) except RAM_00,RAM_01,
     * RAM_20 and RAM_30, which are in Deep Sleep (data retention)
     * during POWER-DOWN because in this example, they are used
     * for application stack, heap, variables ...
     * Note: RAM_02 will automically be put in Deep Sleep mode
     * (data retention) by the Low Power API because it is used
     * to save CORTEX-M33 state (see parameter cpu_retention_addr).
     */
    sram_retention_ctrl  = kPOWER_SRAM_RAM_00 |
                           kPOWER_SRAM_RAM_01 |
                           kPOWER_SRAM_RAM_20 |
                           kPOWER_SRAM_RAM_30;

    /* Wake up upon Analog Comparator or OS Timer interrupt.
     * All other interrupts sources are disabled during POWER-DOWN
     */
    wakeup_interrupts[0] = WAKEUP_ACMP;
    wakeup_interrupts[1] = WAKEUP_OS_EVENT_TIMER;

    /*
     * CPU retention Area in SRAM:
     * The CORTEX-M33 state will be stored from 0x2000_2000
     * to 0x2000_2603; therefore, no application data (stack, heap,
     * variables ...) shall be stored in this address range.
     *
     */
    cpu_retention_addr  = 0x20002000; /* Start of RAM_02 */

    /*
     * 4 - Enter POWER-DOWN low power mode
     */
    POWER_EnterPowerDown(exclude_from_pd, sram_retention_ctrl, wakeup_interrupts, cpu_retention_addr);

    /*
     * When either Analog Comparator or OS Timer interrupt raises,
     * the system wakes up, then :
     * 1 - First, appropriate IRQ Handler is executed (ACMP_IRQHandler or OSEVTIMER0_IRQHandler)
     * 2 - Next, normal processing continues right after POWER_EnterPowerDown call
     */

    /* ... */
}
```

30.2.2.4 POWER_EnterDeepPowerDown

This routine prepares the part, then enter DEEP-POWER-DOWN low power mode. the API function configures which analog/digital components remain running, so that an interrupt from one of these analog/digital peripherals can wake up the part.

30.2.2.4.1 Detailed Description

Table 272. POWER_EnterDeepPowerDown API

void	**POWER_EnterDeepPowerDown**	(		
			uint32_t	**exclude_from_pd[1]**
			uint32_t	**sram_retention_ctrl**
			uint32_t	**wakeup_interrupts[2]**
			uint32_t	**wakeup_io_ctrl**
		)		

The POWER_EnterDeepPowerDown API configures the DEEP-POWER-DOWN low power mode: allows controlling which peripherals are powered up and which SRAM instances are in retention state during DEEP-POWER-DOWN.

NOTE

The POWER_EnterDeepPowerDown API switches the CPU & System Bus clock to 12 MHz. It is the responsibility of the user to reconfigure the CPU and System Bus clock after the function has returned (this happens when the DEEP-POWER-DOWN state is not entered because a RTC or OS Event Timer interrupt was pending when the API was called. Otherwise, this API never returns).

Table 273. Parameters

Parameters	Description
exclude_from_pd[1]	Defines which analog peripherals shall NOT be powered down. It is a 1 x 32-bit vector, with each bit of the vector corresponding to one module. The definition of the vector is given in Table 275, and is aligned with power_pd_bit_t type definition. For each bit field of the vector: • 0: the module is powered down during DEEP-POWER-DOWN. • 1: the module is running during DEEP-POWER-DOWN.
sram_retention_ctrl	Defines which SRAM instances will preserve their content during DEEP-POWER-DOWN. The sram_retention_ctrl parameter is a 32-bit vector, with each bit of the vector corresponding to one SRAM instance. The definition of the vector is given in Table 276, and is aligned with power_sram_bit_t type definition. For each bit field of the vector: • 0: during DEEP-POWER-DOWN, the SRAM instance will be in SRAM Shut Down low power mode (content not preserved). • 1: during DEEP-POWER-DOWN, the SRAM instance will be in SRAM Deep Sleep low power mode (content preserved).

Table continues on the next page...

Table 273. Parameters (continued)

	NOTE During SoC boot time, If the FlexSPI or a 1-bit SPI are used to load an executable code image from an external flash to internal SRAM, then only the following SRAM instances can be in SRAM Deep Sleep low power mode (content preserved) during DEEP-POWER-DOWN: RAM_00 and RAM_01. Otherwise, the following SRAM instances can be in SRAM Deep Sleep low power mode (content preserved) during DEEP-POWER-DOWN: RAM_00, RAM_01, RAM_10, RAM_20, RAM_30, RAM_40, RAM_41, RAM_42 and RAM_43. **NOTE** If VDDMAIN power is removed during DEEP-POWER-DOWN mode, only RAM_00 SRAM instance can be in SRAM Deep Sleep low power mode (content preserved) during DEEP-POWER-DOWN mode.
wakeup_interrupts[2]	Defines which peripheral interrupts can be executed if they are pending before entering DEEP-POWER-DOWN. It is a 2 x 32-bit vectors, with each bit inside the vectors corresponding to one interrupt source. The definition of the vectors is given in Table 277 and Table 278, and is aligned with Low Power Modes Wake up sources (#define WAKEUP_*) #defines definitions. For each bit field of the vectors: • 0: the peripheral interrupt handler will NOT be executed if the interrupt is pending before entering DEEP-POWER-DOWN • 1: the peripheral interrupt handler will be executed if the interrupt is pending before entering DEEP-POWER-DOWN Unlike DEEP-SLEEP and POWER-DOWN modes, in DEEP-POWER-DOWN mode, the "wakeup_interrupts" parameter has no influence on the ability of the peripheral (RTC or OS Event Timer) to wake up the device from DEEP-POWER-DOWN mode; in other words, even if a peripheral interrupt is NOT enabled via the "wakeup_interrupts" parameter (meaning that wakeup_interrupts[0] = 0 and wakeup_interrupts[1] = 0), the device will wake up from DEEP-POWER-DOWN when the peripheral's wake up event occurs (RTC countdown or tamper events, OS Event Timer countdown). **IMPORTANT** *After wake up from DEEP-POWER-DOWN*, the RTC interrupt (if the RTC module is enabled and was the wake up source) is always pending on the CORTEX-M33 Nested Vector Interrupt Controller, but it will be executed *only if* the user enables it directly in the CORTEX-M33 Nested Vector Interrupt Controller (atfer the wake up). **IMPORTANT** *After wake up from DEEP-POWER-DOWN*, the OS Event Timer interrupt is never pending on to the CORTEX-M33 Nested Vector Interrupt Controller even if it has been enabled via this "wakeup_interrupts" prior entering DEEP-POWER-DOWN mode.
wakeup_io_ctrl	Configures the five wake-up pins that can wake up the part from DEEP-POWER-DOWN mode. The definition of the parameter is given in Table 279, and it is aligned with Wake up I/O sources (#define LOWPOWER_WAKEUPIO*) #define definitions.

Table continues on the next page...

Table 273. Parameters (continued)

IMPORTANT

During DEEP-POWER-DOWN mode, the **five wake-up I/Os** are **all** configured as **Input**. The user can choose to enable or disable the internal pull-up and pull-down for each wake-up I/O individually. After wake up from DEEP-POWER-DOWN, the **five wake-up I/Os** retrieve their default configuration (Input and output disabled, both internal pull-up and pull-down disabled) when the POWER_PowerInit API is called.

Table 274. Returned values

Return	Description
	This API does not return any value.

Table 275. Parameter exclude_from_pd[0] definition

Bit	Symbol	Description	Value
3:0	-	Reserved	Must be set to 0
4	FRO1M	1-MHz Free Running Oscillator	• 0: Powered down • 1: Running
5	-	Reserved	
6	FRO32K	32-kHz Free Running Oscillator	• 0: Powered down • 1: Running
7	XTAL32K	32-kHz Crystal Oscillator	• 0: Powered down • 1: Running
31:8	-	Reserved	Must be set to 0

Table 276. Parameter sram_retention_ctrl

Bit	SRAM instance (SRAM size)	Value
0	Reserved	Must be set to 0
1	RAM_00 (4 KBytes)	• 0: Shut Down (content not preserved) • 1: Deep Sleep (content preserved)
2	RAM_01 (4 KBytes)	• 0: Shut Down (content not preserved) • 1: Deep Sleep (content preserved)
4:3	Reserved	Must be set to 0
5	RAM_10 (16 KBytes)[1]	• 0: Shut Down (content not preserved) • 1: Deep Sleep (content preserved)

Table continues on the next page...

Table 276. Parameter sram_retention_ctrl (continued)

Bit	SRAM instance (SRAM size)	Value
6	RAM_20 (32 KBytes)[1,2]	• 0: Shut Down (content not preserved) • 1: Deep Sleep (content preserved)
7	RAM_30 (32 KBytes)[1,2]	• 0: Shut Down (content not preserved) • 1: Deep Sleep (content preserved)
8	RAM_40 (4 KBytes)[1,2]	• 0: Shut Down (content not preserved) • 1: Deep Sleep (content preserved)
9	RAM_41 (4 KBytes)[1,2]	• 0: Shut Down (content not preserved) • 1: Deep Sleep (content preserved)
10	RAM_42 (4 KBytes)[1,2]	• 0: Shut Down (content not preserved) • 1: Deep Sleep (content preserved)
11	RAM_43 (4 KBytes)[1,2]	• 0: Shut Down (content not preserved) • 1: Deep Sleep (content preserved)
31:12	Reserved	Must be set to 0

1. If VDDMAIN power is removed during DEEP-POWER-DOWN mode, this SRAM instance cannot be in SRAM Deep Sleep low power mode (content preserved) during DEEP-POWER-DOWN mode.
2. During SoC Boot time, If the FlexSPI or a 1-bit SPI are used to load an executable code image from an external flash to internal SRAM, then this SRAM instance cannot be in SRAM Deep Sleep low power mode (content preserved) during DEEP-POWER-DOWN.

Table 277. Parameter wakeup_interrupts[0] (1st vector)

Bit	Symbol	Description	Value
28:0	-	Reserved	Must be set to 0
29	RTC	Real Time Clock (counter event and tamper event)	• 0: interrupt disabled • 1: interrupt enabled
31:30	-	Reserved	Must be set to 0

Table 278. Parameter wakeup_interrupts[1] (2nd vector)

Bit	Symbol	Description	Value
5:0	-	Reserved	Must be set to 0
6	OSTIMER	OS Event Timer	• 0: interrupt disabled • 1: interrupt enabled
31:7	-	Reserved	Must be set to 0

Table 279. Parameter wakeup_io_ctrl

Bit	Symbol	Description	Value
1:0	MODEWAKEUP0	Select wake-up pin 0 operating mode during DEEP-POWER-DOWN.	• 0: Disabled • 1: Rising edge • 2: Falling edge • 3: Rising and falling edges
3:2	MODEWAKEUP1	Select wake-up pin 1 operating mode during DEEP-POWER-DOWN.	• 0: Disabled • 1: Rising edge • 2: Falling edge • 3: Rising and falling edges
5:4	MODEWAKEUP2	Select wake-up pin 2 operating mode during DEEP-POWER-DOWN.	• 0: Disabled • 1: Rising edge • 2: Falling edge • 3: Rising and falling edges
7:6	MODEWAKEUP3	Select wake-up pin 3 operating mode during DEEP-POWER-DOWN.	• 0: Disabled • 1: Rising edge • 2: Falling edge • 3: Rising and falling edges
9:8	MODEWAKEUP4	Select wake-up pin 4 operating mode during DEEP-POWER-DOWN.	• 0: Disabled • 1: Rising edge • 2: Falling edge • 3: Rising and falling edges
19:10	-	Reserved	Must be set to 0
20	DISABLEPULLUPDOWNWAKEUP0	Disable both pull-up and pull-down for wake-up pin 0 during DEEP-POWER-DOWN. When wake-up pin 0: • is enabled (MODEWAKEUP0 is not 0) AND • pull-up or pull-down is enabled (DISABLEPULLUPDOWNWAKEUP0=0) then the state of the pull-up and pull-down is determined by the wake-up pin mode (and not by the field PULLUPDOWNWAKEUP0) as described in Table 280.	• 0: Enable pull-up or pull-down (see PULLUPDOWNWAKEUP0) • 1: Disable both pull-up and pull-down

Table continues on the next page...

Table 279. Parameter wakeup_io_ctrl (continued)

Bit	Symbol	Description	Value
21	DISABLEPULLUPDOWNWAKEUP1	Disable both pull-up and pull-down for wake-up pin 1 during DEEP-POWER-DOWN. When wake-up pin 1: • is enabled (MODEWAKEUP1 is not 0) AND • pull-up or pull-down is enabled (DISABLEPULLUPDOWNWAKEUP1=0) then the state of the pull-up and pull-down is determined by the wake-up pin mode (and not by the field PULLUPDOWNWAKEUP1) as described in Table 280.	• 0: Enable pull-up or pull-down (see PULLUPDOWNWAKEUP1) • 1: Disable both pull-up and pull-down
22	DISABLEPULLUPDOWNWAKEUP2	Disable both pull-up and pull-down for wake-up pin 2 during DEEP-POWER-DOWN. When wake-up pin 2: • is enabled (MODEWAKEUP2 is not 0) AND • pull-up or pull-down is enabled (DISABLEPULLUPDOWNWAKEUP2=0) then the state of the pull-up and pull-down is determined by the wake-up pin mode (and not by the field PULLUPDOWNWAKEUP2) as described in Table 280.	• 0: Enable pull-up or pull-down (see PULLUPDOWNWAKEUP2) • 1: Disable both pull-up and pull-down
23	DISABLEPULLUPDOWNWAKEUP3	Disable both pull-up and pull-down for wake-up pin 3 during DEEP-POWER-DOWN. When wake-up pin 3: • is enabled (MODEWAKEUP3 is not 0) AND • pull-up or pull-down is enabled (DISABLEPULLUPDOWNWAKEUP3=0) then the state of the pull-up and pull-down is determined by the wake-up pin mode (and not by the field PULLUPDOWNWAKEUP3) as described in Table 280.	• 0: Enable pull-up or pull-down (see PULLUPDOWNWAKEUP3) • 1: Disable both pull-up and pull-down
24	DISABLEPULLUPDOWNWAKEUP4	Disable both pull-up and pull-down for wake-up pin 4 during DEEP-POWER-DOWN. When wake-up pin 4: • is enabled (MODEWAKEUP4 is not 0) AND • pull-up or pull-down is enabled (DISABLEPULLUPDOWNWAKEUP4=0) then the state of the pull-up and pull-down is determined by the wake-up pin mode (and not by the field PULLUPDOWNWAKEUP4) as described in Table 280.	• 0: Enable pull-up or pull-down (see PULLUPDOWNWAKEUP4) • 1: Disable both pull-up and pull-down
25	PULLUPDOWNWAKEUP0	Select pull-up or pull-down for wake-up pin 0 during DEEP-POWER-DOWN. This	• 0: Pull-down

Table continues on the next page...

Table 279. Parameter wakeup_io_ctrl (continued)

Bit	Symbol	Description	Value
		field has effect only when wake-up pin 0 is disabled (MODEWAKEUP0=0) and when wake-up pin 0 pull-up/pull-down are enabled (DISABLEPULLUPDOWNWAKEUP0=0)	• 1: Pull-up
26	PULLUPDOWNWAKEUP1	Select pull-up or pull-down for wake-up pin 1 during DEEP-POWER-DOWN. This field has effect only when wake-up pin 1 is disabled (MODEWAKEUP1=0) and when wake-up pin 1 pull-up/pull-down are enabled (DISABLEPULLUPDOWNWAKEUP1=0)	• 0: Pull-down • 1: Pull-up
27	PULLUPDOWNWAKEUP2	Select pull-up or pull-down for wake-up pin 2 during DEEP-POWER-DOWN. This field has effect only when wake-up pin 2 is disabled (MODEWAKEUP2=0) and when wake-up pin 2 pull-up/pull-down are enabled (DISABLEPULLUPDOWNWAKEUP2=0)	• 0: Pull-down • 1: Pull-up
28	PULLUPDOWNWAKEUP3	Select pull-up or pull-down for wake-up pin 3 during DEEP-POWER-DOWN. This field has effect only when wake-up pin 3 is disabled (MODEWAKEUP3=0) and when wake-up pin 3 pull-up/pull-down are enabled (DISABLEPULLUPDOWNWAKEUP3=0)	• 0: Pull-down • 1: Pull-up
29	PULLUPDOWNWAKEUP4	Select pull-up or pull-down for wake-up pin 4 during DEEP-POWER-DOWN. This field has effect only when wake-up pin 4 is disabled (MODEWAKEUP4=0) and when wake-up pin 4 pull-up/pull-down are enabled (DISABLEPULLUPDOWNWAKEUP4=0)	• 0: Pull-down • 1: Pull-up
31:30	-	Reserved	Must be set to 0

Table 280. pull-up/pull-down configuration

User Configuration				Resulting Pull-up/pull-down state		Comments
DISABLEPULLUPDOWNWAKEUPx	MODEWAKEUPx	PULLUPDOWNWAKEUPx		Pull-down State	Pull-up state	
0	0(disabled)	0 (pull-down)		Enabled	Disabled	The wake-up pin is disabled (MODEWAKEUP = 0) and and both the pull-up/pull-down are not disabled (DISABLEPULLUPDOWNWAKEUP=0): in that case, the pull-up and pull-down states depend
0	0 (disabled)	1 (pull-up)		Disabled	Enabled	

Table continues on the next page...

Table 280. pull-up/pull-down configuration (continued)

User Configuration				Resulting Pull-up/pull-down state		Comments
DISABLEPULLUPDOWNWAKEUPx	MODEWAKEUPx	PULLUPDOWNWAKEUPx		Pull-down State	Pull-up state	
						on the value of PULLUPDOWNWAKEUPx.
0	1 (Rising edge)	-		Enabled	Disabled	The wake-up pin is enabled (MODEWAKEUP != 0) and both the pull-up/ pull-down are not disabled (DISABLEPULLUPDOWNWAKEUP=0): in that case, the pull-up and pull-down states depend only on the value of MODEWAKEUPx.
0	2 (Falling edge)	-		Disabled	Enabled	
0	3 (Rising and falling edges)	-		Enabled	Disabled	
1	-	-		Disabled	Disabled	Both the pull-up and the pull-down are disabled, whatever the values of MODEWAKEUP and PULLUPDOWNWAKEUP.Use this configuration when an external pull-up or pull-down resistor is connected on the wake-up pin.

30.2.2.4.2 Examples

In all examples presented herafter, we define the following variables :

```
uint32_t exclude_from_pd[1];   /* */
uint32_t sram_retention_ctrl;  /* */
uint32_t wakeup_interrupts[2]; /* */
uint32_t wakeup_io_ctrl;   /* */
```

30.2.2.4.2.1 Enter DEEP-POWER-DOWN mode with wake up by RTC (FRO32K as clock source) or WAKE-UP I/O 2 (falling edge), all SRAM instances in SRAM Shut down power mode (no data retention)

```
void RTC_ALARM_IRQHandler(void); /* The RTC Alarm Interrupt Handler */

int main()
{
   /*
    * 1 - Do some power related initialization of the system :
```

```
 */
if ( POWER_PowerInit() != kPOWER_Status_Success )
{
   /* On failure, stop processing */
}

/* On success, continue normal processing ... */
/*
 * 2 - Configure the RTC :
 *   --> Enable the FRO32K
 *   --> Select the FRO32K as the clock source for the RTC
 *   --> Enable the Alarm interrupt in RTC
 *   --> ...
 */
/* RTC-> ... ; */

/*
 * 3 - Set up DEEP-POWER-DOWN low power parameters
 */

/* All analog modules are shut down during DEEP-POWER-DOWN,
 * except FRO32K as it is the clock source of the RTC.
 */
exclude_from_pd[0]   =kPDRUNCFG_PD_FRO32K;

/* All SRAM instances in Shut down low power mode (no data
 * retention, low leakage) during DEEP-POWER-DOWN.
 */
sram_retention_ctrl  = 0UL;

/*
 * For DEEP-POWER-DOWN, enabling the RTC Interrupt in the
 * "wakeup_interrupts" parameter below is usefull only
 * in case the interrupt is pending before actually enter the
 * DEEP-POWER-DOWN mode: in such a situation, the interrupt handler
 * will be executed before entering the DEEP-POWER-DOWN mode.
 *
 * Unlike DEEP-SLEEP and POWER-DOWN modes, in DEEP-POWER-DOWN mode,
 * the "wakeup_interrupts" parameter has no influence on the ability
 * of the RTC to wake up the device from DEEP-POWER-DOWN mode;
 * in other words, even if the RTC interrupt is NOT enabled via the
 * "wakeup_interrupts" parameter (meaning that wakeup_interrupts[0] = 0),
 * the device will wake up from DEEP-POWER-DOWN at the end of the
 * RTC countdown.
 *
 * After wake up from DEEP-POWER-DOWN, the RTC interrupt handler
 * will be executed ONLY IF the user enables it in the CORTEX-M33
 * NVIC (after wake up).
*/
wakeup_interrupts[0] = WAKEUP_RTC_ALARM_WAKEUP;
wakeup_interrupts[1] = 0UL;

/* Wake up I/O configuration:
 * wake-up 0 : Edge detectors disabled (by default), internal pull-up and pull-down disabled
 * wake-up 1 : Edge detectors disabled (by default), internal pull-up and pull-down disabled
 * wake-up 2 : Falling edge detection enabled (and internal pull-up enabled)
 * wake-up 3 : Edge detectors disabled (by default), internal pull-up and pull-down disabled
 * wake-up 4 : Edge detectors disabled (by default), internal pull-up and pull-down disabled
 */
wakeup_io_ctrl       = LOWPOWER_WAKEUPIO_PIO0_DISABLEPULLUPDOWN_MASK |
```

```
                            LOWPOWER_WAKEUPIO_PIO1_DISABLEPULLUPDOWN_MASK |
                            (LOWPOWER_WAKEUPIOSRC_FALLING << LOWPOWER_WAKEUPIOSRC_PIO2_INDEX) |
                            LOWPOWER_WAKEUPIO_PIO3_DISABLEPULLUPDOWN_MASK |
                            LOWPOWER_WAKEUPIO_PIO4_DISABLEPULLUPDOWN_MASK;

    /*
     * 4 - Enter DEEP-POWER-DOWN low power mode
     */
    POWER_EnterDeepPowerDown(exclude_from_pd, sram_retention_ctrl, wakeup_interrupts, wakeup_io_ctrl);

    /* This point shall never be reached.  */
}
```

30.2.2.4.2.2 Enter DEEP-POWER-DOWN mode with wake up by OS Event Timer (FRO1M as clock source), all WAKE-UP I/Os disabled with internal pull-down enabled, RAM_00 in Deep Sleep mode (data retention), all other SRAM instances in Shut down mode (no data retention)

```

    /*
     * 1 - Do some power related initialization of the system :
     */
    if ( POWER_PowerInit() != kPOWER_Status_Success )
    {
       /* On failure, stop processing */
    }

    /* On success, continue normal processing ... */

    /*
     * 2 - Configure the OS Timer :
     *   --> Select the FRO1M as the clock source for the OS Timer
     *   --> Enable OS Timer clock, ...
     *   --> Set OS Timer Match Low and Match High registers ...
     *   --> ...
     */
    /* PMC->OSEVENTTIMER = ... ; */
    /* OSTIMER-> ... ; */

    /*
     * 3 - Set up DEEP-POWER-DOWN low power parameters
     */

    /* All analog modules are shut down during DEEP-POWER-DOWN,
     * except FRO1M as it is the clock source of the OS Timer.
     */
    exclude_from_pd[0]   = kPDRUNCFG_PD_FRO1M;

    /* All SRAM instances in Shut down low power mode (no data
     * retention, low leakage) except RAM_00 which is in Deep Sleep
     * (data retention) during DEEP-POWER-DOWN.
     */
    sram_retention_ctrl  = kPOWER_SRAM_RAM_00;

    /*
     * For DEEP-POWER-DOWN, enabling the OS Timer Interrupt in the
     * "wakeup_interrupts" parameter below is usefull only
     * in case the interrupt is pending before actually enter the
```

```
     * DEEP-POWER-DOWN mode: in such a situation, the interrupt handler
     * will be executed before entering the DEEP-POWER-DOWN mode.
     *
     * Unlike DEEP-SLEEP and POWER-DOWN modes, in DEEP-POWER-DOWN mode,
     * the "wakeup_interrupts" parameter has no influence on the ability
     * of the OS Timer to wake up the device from DEEP-POWER-DOWN mode;
     * in other words, even if the OS Timer interrupt is NOT enabled via the
     * "wakeup_interrupts" parameter (meaning that wakeup_interrupts[1] = 0),
     * the device will wake up from DEEP-POWER-DOWN at the end of the
     * RTC countdown.
     *
     * After wake up from DEEP-POWER-DOWN, the OS Timer interrupt is never
     * pending at CORTEX-M33 at NVIC (after wake up).
     */
    wakeup_interrupts[0] = 0UL;
    wakeup_interrupts[1] = WAKEUP_OS_EVENT_TIMER;

    /* Wake up I/O configuration:
     * wake-up 0 : Edge detectors disabled (by default), internal pull-down enabled (by default)
     * wake-up 1 : Edge detectors disabled (by default), internal pull-down enabled (by default)
     * wake-up 2 : Edge detectors disabled (by default), internal pull-down enabled (by default)
     * wake-up 3 : Edge detectors disabled (by default), internal pull-down enabled (by default)
     * wake-up 4 : Edge detectors disabled (by default), internal pull-down enabled (by default)
     */
    wakeup_io_ctrl       = 0UL;

    /*
     * 4 - Enter DEEP-POWER-DOWN low power mode
     */
    POWER_EnterDeepPowerDown(exclude_from_pd, sram_retention_ctrl, wakeup_interrupts, wakeup_io_ctrl);

    /* This point shall never be reached. */
```

30.2.2.4.2.3 Enter DEEP-POWER-DOWN mode with wake up by OS Event Timer (XTAL32K as clock source), all WAKE-UP I/Os disabled with internal pull-down enabled, RAM_00 and RAM01 in Deep Sleep mode (data retention), all other SRAM instances in Shut down mode (no data retention)

```
    /*
     * 1 - Do some power related initialization of the system :
     */
    if ( POWER_PowerInit() != kPOWER_Status_Success )
    {
       /* On failure, stop processing */
    }

    /* On success, continue normal processing ... */

    /*
     * 2 - Configure the OS Timer :
     *   --> Enable XTAL32K
     *   --> Select the XTAL32K as the clock source for the OS Timer
     *   --> Enable OS Timer clock, ...
     *   --> Set OS Timer Match Low and Match High registers ...
     *   --> ...
     */
    /* PMC->PDRUNCFGCLR0 = PMC_PDRUNCFG0_PDEN_XTAL32K_MASK;*/
```

```
    /* PMC->OSEVENTTIMER = ... ; */
    /* OSTIMER-> ... ; */

    /*
     * 3 - Set up DEEP-POWER-DOWN low power parameters
     */

    /* All analog modules are shut down during DEEP-POWER-DOWN,
     * except XTAL32K as it is the clock source of the OS Timer.
     */
    exclude_from_pd[0]   = kPDRUNCFG_PD_XTAL32K;

    /* All SRAM instances in Shut down low power mode (no data
     * retention, low leakage) except RAM_00 and RAM_01 which
     * are in Deep Sleep (data retention) during DEEP-POWER-DOWN.
     */
    sram_retention_ctrl  = kPOWER_SRAM_RAM_00 | kPOWER_SRAM_RAM_01;

    /*
     * For DEEP-POWER-DOWN, enabling the OS Timer Interrupt in the
     * "wakeup_interrupts" parameter below is usefull only
     * in case the interrupt is pending before actually enter the
     * DEEP-POWER-DOWN mode: in such a situation, the interrupt handler
     * will be executed before entering the DEEP-POWER-DOWN mode.
     *
     * Unlike DEEP-SLEEP and POWER-DOWN modes, in DEEP-POWER-DOWN mode,
     * the "wakeup_interrupts" parameter has no influence on the ability
     * of the OS Timer to wake up the device from DEEP-POWER-DOWN mode;
     * in other words, even if the OS Timer interrupt is NOT enabled via the
     * "wakeup_interrupts" parameter (meaning that wakeup_interrupts[1] = 0),
     * the device will wake up from DEEP-POWER-DOWN at the end of the
     * RTC countdown.
     *
     * After wake up from DEEP-POWER-DOWN, the OS Timer interrupt is never
     * pending at CORTEX-M33 at NVIC (after wake up).
     */
    wakeup_interrupts[0] = 0UL;
    wakeup_interrupts[1] = WAKEUP_OS_EVENT_TIMER;

    /* Wake up I/O configuration:
     * wake-up 0 : Edge detectors disabled (by default), internal pull-down enabled (by default)
     * wake-up 1 : Edge detectors disabled (by default), internal pull-down enabled (by default)
     * wake-up 2 : Edge detectors disabled (by default), internal pull-down enabled (by default)
     * wake-up 3 : Edge detectors disabled (by default), internal pull-down enabled (by default)
     * wake-up 4 : Edge detectors disabled (by default), internal pull-down enabled (by default)
     */
    wakeup_io_ctrl       = 0UL;

    /*
     * 4 - Enter DEEP-POWER-DOWN low power mode
     */
    POWER_EnterDeepPowerDown(exclude_from_pd, sram_retention_ctrl, wakeup_interrupts, wakeup_io_ctrl);

    /* This point shall never be reached */
```

30.2.2.4.2.4 More examples of wake up I/Os configuration

```
   /* Wake up I/O configuration:
    * wake-up 0 : Edge detectors disabled (by default), internal pull-down enabled
(by default)
    * wake-up 1 : Edge detectors disabled (by default), internal pull-down enabled
(by default)
    * wake-up 2 : Edge detectors disabled (by default), internal pull-down enabled
(by default)
    * wake-up 3 : Edge detectors disabled (by default), internal pull-down enabled
(by default)
    * wake-up 4 : Edge detectors disabled (by default), internal pull-down enabled
(by default)
    */
   wakeup_io_ctrl       = 0UL;
```

```
   /* Wake up I/O configuration:
    * wake-up 0 : Rising edge detection enabled (and internal pull-down enabled)
    * wake-up 1 : Edge detectors disabled (by default), internal pull-up and pull-
down disabled
    * wake-up 2 : Rising and Falling edges detection enabled (and internal pull-
down enabled)
    * wake-up 3 : Edge detectors disabled (by default), internal pull-down enabled
(by default)
    * wake-up 4 : Falling edge detection enabled (and internal pull-up enabled)
    */
   wakeup_io_ctrl       = (LOWPOWER_WAKEUPIOSRC_RISING <<
LOWPOWER_WAKEUPIOSRC_PIO0_INDEX) |
                          LOWPOWER_WAKEUPIO_PIO1_DISABLEPULLUPDOWN_MASK |
                          (LOWPOWER_WAKEUPIOSRC_RISING_FALLING <<
LOWPOWER_WAKEUPIOSRC_PIO2_INDEX) |

                          (LOWPOWER_WAKEUPIOSRC_FALLING
<< LOWPOWER_WAKEUPIOSRC_PIO4_INDEX);
```

```
   /* Wake up I/O configuration:
    * wake-up 0 : Rising edge detection enabled (and internal pull-down enabled)
    * wake-up 1 : Rising edge detection enabled, internal pull-up and pull-
down disabled
    * wake-up 2 : Rising and Falling edges detection enabled (and internal pull-
down enabled)
    * wake-up 3 : Rising and Falling edges detection enabled, internal pull-up and
pull-down disabled
    * wake-up 4 : Edge detectors disabled (by default), internal pull-down enabled
(by default)
    */
   wakeup_io_ctrl       = (LOWPOWER_WAKEUPIOSRC_RISING <<
LOWPOWER_WAKEUPIOSRC_PIO0_INDEX) |
```

```
                            (LOWPOWER_WAKEUPIOSRC_RISING <<
LOWPOWER_WAKEUPIOSRC_PIO1_INDEX) | LOWPOWER_WAKEUPIO_PIO1_DISABLEPULLUPDOWN_MASK |
                            (LOWPOWER_WAKEUPIOSRC_RISING_FALLING <<
LOWPOWER_WAKEUPIOSRC_PIO2_INDEX) |
                            (LOWPOWER_WAKEUPIOSRC_RISING_FALLING <<
LOWPOWER_WAKEUPIOSRC_PIO3_INDEX) | LOWPOWER_WAKEUPIO_PIO3_DISABLEPULLUPDOWN_MASK
                            ;
```

30.2.2.5 POWER_SetWakeUpPins

This routine configures the 5 wake-up pins before entering DEEP-SLEEP or POWER-DOWN modes. It is expected to be called before calling POWER_EnterDeepSleep or POWER_EnterPowerDown. The API function configures which wake-up pins can wake up the device from low power mode, and sets up some other pin parameters like the internal pull-up/pull-down, rising or falling edge detections.

NOTE

For DEEP-POWER-DOWN mode, the 5 wake-up pins are set up via the *wakeup_io_ctrl* parameter of the POWER_EnterDeepPowerDown API.

30.2.2.5.1 Detailed Description

Table 281. POWER_SetWakeUpPins API

void	**POWER_SetWakeUpPins**	(		
			uint32_t	**wakeup_io_cfg_src**
			uint32_t	**wakeup_io_ctrl**
		)		

The POWER_SetWakeUpPins API configures the 5 wake-up pins before entering DEEP-SLEEP or POWER-DOWN modes.

IMPORTANT

Wake-up pins interrupt management during DEEP-SLEEP and POWER-DOWN: for wake-up pins to wake up the device during DEEP-SLEEP and POWER-DOWN, the interrupt vector 31 (#define WAKEUP_WAKEUP_MAILBOX, same interrupt slot as the CPU MAILBOX) MUST be enabled via the *wakeup_interrupts[0]* parameter of POWER_EnterDeepSleep and POWER_EnterPowerDown API respectively . After wake-up from DEEP-SLEEP/POWER-DOWN, the wake-up pins interrupt handler will be automatically executed. See examples of wake-up pins interrupt handler implementation (Wake-up pins Interrupt Handler implementation example).

Table 282. Parameters

Parameters	Description
wakeup_io_cfg_src	Defines the source for some parameters of **the whole 5** wake-up pins, especially the control of the internal pull-up/pull-down and wether the pins are configured as input or output. The definition of the parameter is aligned with Wake up I/O sources (#define LOWPOWER_WAKEUPIO*) (LOWPOWER_WAKEUPIO_CFG_SRC*) #define definitions: • 0: the **5** wake-up pins configuration (in/out, pull-up/pull-down/external pull-up/pull-down ...) is set up in the IOCON module (see). In that case, the **wakeup_io_ctrl** parameter is used only to enable/disable rising/falling edge detections.

Table continues on the next page...

Table 282. Parameters (continued)

	• 1: the **5** wake-up pins configuration (in/out, pull-up/pull-down/external pull-up/pull-down ...) is set up via the **wakeup_io_ctrl**, as well as the rising/falling edge detections.
wakeup_io_ctrl	Configures the 5 wake-up pins that can wake up the part from DEEP-SLEEP and POWER-DOWN modes. The definition of the parameter is given in Table 284, and it is aligned with Wake up I/O sources (#define LOWPOWER_WAKEUPIO*) #define definitions. **IMPORTANT** When *wakeup_io_cfg_src = 0*, bit fields [31:20] are not relevent (they have no effect), only bit fields [9:0] have effect. When *wakeup_io_cfg_src = 1*, the whole [31:0] vector is used.

Table 283. Returned values

Return	Description
	This API does not return any value.

Table 284. Parameter wakeup_io_ctrl

Bit	Symbol	Description	Value
1:0	MODEWAKEUP0	Select wake-up pin 0 operating mode during DEEP-SLEEP or POWER-DOWN.	• 0: Disabled • 1: Rising edge • 2: Falling edge • 3: Rising and falling edges
3:2	MODEWAKEUP1	Select wake-up pin 1 operating mode during DEEP-SLEEP or POWER-DOWN.	• 0: Disabled • 1: Rising edge • 2: Falling edge • 3: Rising and falling edges
5:4	MODEWAKEUP2	Select wake-up pin 2 operating mode during DEEP-SLEEP or POWER-DOWN.	• 0: Disabled • 1: Rising edge • 2: Falling edge • 3: Rising and falling edges
7:6	MODEWAKEUP3	Select wake-up pin 3 operating mode during DEEP-SLEEP or POWER-DOWN.	• 0: Disabled • 1: Rising edge • 2: Falling edge • 3: Rising and falling edges

Table continues on the next page...

Table 284. Parameter wakeup_io_ctrl (continued)

Bit	Symbol	Description	Value
9:8	MODEWAKEUP4	Select wake-up pin 4 operating mode during DEEP-SLEEP or POWER-DOWN.	• 0: Disabled • 1: Rising edge • 2: Falling edge • 3: Rising and falling edges
19:10	-	Reserved	Must be set to 0
20	DISABLEPULLUPDOWNWAKEUP0[1]	Disable both pull-up and pull-down for wake-up pin 0 during DEEP-SLEEP or POWER-DOWN. When wake-up pin 0: • is enabled (MODEWAKEUP0 is not 0) AND • pull-up or pull-down is enabled (DISABLEPULLUPDOWNWAKEUP0=0) then the state of the pull-up and pull-down is determined by the wake-up pin mode (and not by the field PULLUPDOWNWAKEUP0) as described in Table 280.	• 0: Enable pull-up or pull-down (see PULLUPDOWNWAKEUP0) • 1: Disable both pull-up and pull-down
21	DISABLEPULLUPDOWNWAKEUP1[1]	Disable both pull-up and pull-down for wake-up pin 1 during DEEP-SLEEP or POWER-DOWN. When wake-up pin 1: • is enabled (MODEWAKEUP1 is not 0) AND • pull-up or pull-down is enabled (DISABLEPULLUPDOWNWAKEUP1=0) then the state of the pull-up and pull-down is determined by the wake-up pin mode (and not by the field PULLUPDOWNWAKEUP1) as described in Table 280.	• 0: Enable pull-up or pull-down (see PULLUPDOWNWAKEUP1) • 1: Disable both pull-up and pull-down
22	DISABLEPULLUPDOWNWAKEUP2[1]	Disable both pull-up and pull-down for wake-up pin 2 during DEEP-SLEEP or POWER-DOWN. When wake-up pin 2: • is enabled (MODEWAKEUP2 is not 0) AND • pull-up or pull-down is enabled (DISABLEPULLUPDOWNWAKEUP2=0) then the state of the pull-up and pull-down is determined by the wake-up pin mode (and not by the field PULLUPDOWNWAKEUP2) as described in Table 280.	• 0: Enable pull-up or pull-down (see PULLUPDOWNWAKEUP2) • 1: Disable both pull-up and pull-down
23	DISABLEPULLUPDOWNWAKEUP3[1]	Disable both pull-up and pull-down for wake-up pin 3 during DEEP-SLEEP or POWER-DOWN. When wake-up pin 3: • is enabled (MODEWAKEUP3 is not 0) AND	• 0: Enable pull-up or pull-down (see PULLUPDOWNWAKEUP3)

Table continues on the next page...

Table 284. Parameter wakeup_io_ctrl (continued)

Bit	Symbol	Description	Value
		• pull-up or pull-down is enabled (DISABLEPULLUPDOWNWAKEUP3=0) then the state of the pull-up and pull-down is determined by the wake-up pin mode (and not by the field PULLUPDOWNWAKEUP3) as described in Table 280.	• 1: Disable both pull-up and pull-down
24	DISABLEPULLUPDOWNWAKEUP4[1]	Disable both pull-up and pull-down for wake-up pin 4 during DEEP-SLEEP or POWER-DOWN. When wake-up pin 4: • is enabled (MODEWAKEUP4 is not 0) AND • pull-up or pull-down is enabled (DISABLEPULLUPDOWNWAKEUP4=0) then the state of the pull-up and pull-down is determined by the wake-up pin mode (and not by the field PULLUPDOWNWAKEUP4) as described in Table 280.	• 0: Enable pull-up or pull-down (see PULLUPDOWNWAKEUP4) • 1: Disable both pull-up and pull-down
25	PULLUPDOWNWAKEUP0[1]	Select pull-up or pull-down for wake-up pin 0 during DEEP-SLEEP or POWER-DOWN. This field has effect only when wake-up pin 0 is disabled (MODEWAKEUP0=0) and when wake-up pin 0 pull-up/pull-down are enabled (DISABLEPULLUPDOWNWAKEUP0=0)	• 0: Pull-down • 1: Pull-up
26	PULLUPDOWNWAKEUP1[1]	Select pull-up or pull-down for wake-up pin 1 during DEEP-SLEEP or POWER-DOWN. This field has effect only when wake-up pin 1 is disabled (MODEWAKEUP1=0) and when wake-up pin 1 pull-up/pull-down are enabled (DISABLEPULLUPDOWNWAKEUP1=0)	• 0: Pull-down • 1: Pull-up
27	PULLUPDOWNWAKEUP2[1]	Select pull-up or pull-down for wake-up pin 2 during DEEP-SLEEP or POWER-DOWN. This field has effect only when wake-up pin 2 is disabled (MODEWAKEUP2=0) and when wake-up pin 2 pull-up/pull-down are enabled (DISABLEPULLUPDOWNWAKEUP2=0)	• 0: Pull-down • 1: Pull-up
28	PULLUPDOWNWAKEUP3[1]	Select pull-up or pull-down for wake-up pin 3 during DEEP-SLEEP or POWER-DOWN. This field has effect only when wake-up pin 3 is disabled (MODEWAKEUP3=0) and when wake-up pin 3 pull-up/pull-down are enabled (DISABLEPULLUPDOWNWAKEUP3=0)	• 0: Pull-down • 1: Pull-up
29	PULLUPDOWNWAKEUP4[1]	Select pull-up or pull-down for wake-up pin 4 during DEEP-SLEEP or POWER-DOWN. This field has effect only when wake-up pin 4 is disabled (MODEWAKEUP4=0) and when	• 0: Pull-down • 1: Pull-up

Table continues on the next page...

Table 284. Parameter wakeup_io_ctrl (continued)

Bit	Symbol	Description	Value
		wake-up pin 4 pull-up/pull-down are enabled (DISABLEPULLUPDOWNWAKEUP4=0)	
31:30	-[1]	Reserved	Must be set to 0

1. Meaningfull only when *wakeup_io_cfg_src = 0*, ignored for any other values.

30.2.2.5.2 Examples

30.2.2.5.2.1 Wake-up pins Interrupt Handler implementation example

```
void WAKEUP_PINS_IRQHandler(void)
{
   uint32_t wake_up_pins_cause;

   /* Get the wake-up pins status, for example
    * to know which wake-up pin triggered the device
    * wake up (when several wake-up pins are enabled
    * to wake-up the device).
    */
   wake_up_pins_cause = PMC->WAKEIOCAUSE;

   /* Force wake-up pins controls to come from IOCON (not mandatory) */
   PMC->WAKEUPIOCTRL = PMC->WAKEUPIOCTRL & (~PMC_WAKEUPIOCTRL_WAKEUPIO_ENABLE_CTRL_MASK);

   /*
    * Do some useful processing with "wake_up_pins_cause"
    * according the application needs.
    */
   /* ... */

   /* ... and finally : */
   /* Reset the wake-up pins status (WAKEUPIO_RSTN = 0) */
   PMC->WAKEUPIOCTRL = PMC->WAKEUPIOCTRL & (~PMC_WAKEUPIOCTRL_WAKEUPIO_RSTN_MASK);
}
```

30.2.2.5.2.2 Enter DEEP-SLEEP with wake up by wake-up pin 2 (rising edge) or wake-up pin 4 (falling edge), with all 5 wake-up pins settings (pull-up/pull-down/plain input) set up in the IOCON module

```
   uint32_t exclude_from_pd[2];   /* */
   uint32_t sram_retention_ctrl;  /* */
   uint32_t wakeup_interrupts[4]; /* */
   uint32_t hardware_wake_ctrl;   /* */

   /*
    * 0 - Do some power related initialization of the system :
    */
   if ( POWER_PowerInit() != kPOWER_Status_Success )
   {
```

```
    /* On failure, stop processing */
}

/* On success, continue normal processing ... */

/*
 * 1 - Set up DEEP-SLEEP low power parameters
 */

/* All analog modules are shut down during POWER-DOWN */
exclude_from_pd[0] = exclude_from_pd[1] = 0UL;

/* All SRAM in retention mode.*/
sram_retention_ctrl = kPOWER_SRAM_DSLP_MASK;

/* Wake up upon Wake-up pins interrupt.
 * All other interrupts sources are disabled during POWER-DOWN
 */
wakeup_interrupts[0] = WAKEUP_WAKEUP_MAILBOX;
wakeup_interrupts[1] = wakeup_interrupts[2] = wakeup_interrupts[3] = 0UL;

/* No SDMA transfer during DEEP-SLEEP */
hardware_wake_ctrl = 0UL;

/*
 * 2 - Set up wake-up pins:
 * - All 5 wake-up pins configuration (in/out, pull-up/pull-down/plain input)
 *    are set up in the IOCON module.
 * - The 5 wake-up pins event detection are configured as following:
 *     wake-up 0 : disabled
 *     wake-up 1 : disabled
 *     wake-up 2 : enabled, rising edge
 *     wake-up 3 : disabled
 *     wake-up 4 : enabled, falling edge
 */

/* Configure the 5 wake-up pins (pull-up/pull-down/plain input) in IOCON:
 *  IOCON->PIO[1][ 1] = ... ; // wake-up pin 0
 *  IOCON->PIO[0][28] = ... ; // wake-up pin 1
 *  IOCON->PIO[1][18] = ... ; // wake-up pin 2
 *  IOCON->PIO[1][30] = ... ; // wake-up pin 3
 *  IOCON->PIO[0][26] = ... ; // wake-up pin 4
 */

/* Enable edge detectors via "wakeup_io_ctrl" parameter of the POWER_SetWakeUpPins API */
wakeup_io_ctrl = (LOWPOWER_WAKEUPIOSRC_RISING  << LOWPOWER_WAKEUPIOSRC_PIO2_INDEX) |
                 (LOWPOWER_WAKEUPIOSRC_FALLING << LOWPOWER_WAKEUPIOSRC_PIO4_INDEX);

/* Set up wake-up pins */
POWER_SetWakeUpPins(LOWPOWER_WAKEUPIO_CFG_SRC_IOCON, wakeup_io_ctrl);

/*
 * 3 - Enter DEEP-SLEEP low power mode
 */
POWER_EnterDeepSleep(exclude_from_pd, sram_retention_ctrl, wakeup_interrupts, hardware_wake_ctrl);

/*
 * When the wake-up pins interrupt raises, the device wakes up, then:
 * 1 - First, the WAKEUP_PINS_IRQHandler is executed.
 * 2 - Next, normal processing continues right after POWER_EnterDeepSleep call
```

```
 */
 /* ... */
```

30.2.2.5.2.3 Enter POWER-DOWN with wake up by wake-up pin 0 (rising edge), with all 5 wake-up pins settings (pull-up/pull-down/plain input) set up via the *wakeup_io_ctrl* parameter of the POWER_SetWakeUpPins API

```
uint32_t exclude_from_pd[1];   /* */
uint32_t sram_retention_ctrl;  /* */
uint32_t wakeup_interrupts[2]; /* */
uint32_t cpu_retention_addr;   /* */

/*
 * 0 - Do some power related initialization of the system :
 */
if ( POWER_PowerInit() != kPOWER_Status_Success )
{
   /* On failure, stop processing */
}

/* On success, continue normal processing ... */

/*
 * 1 - Set up POWER-DOWN low power parameters
 */

/*
 * All analog modules are shut down during POWER-DOWN
 */
exclude_from_pd[0]   = 0UL;

/* All SRAM instances in Shut down low power mode (no data
 * retention, low leakage) except RAM_00,RAM_01 and RAM_03 which
 * are in Deep Sleep (data retention) during POWER-DOWN.
 * In this example, RAM_03 is used to save CORTEX-M33 state
 * and RAM_00/RAM_01 are used for application stack, heap, variables ...
 */
sram_retention_ctrl  = kPOWER_SRAM_RAM_00 | kPOWER_SRAM_RAM_01 | kPOWER_SRAM_RAM_03;

/* Wake up upon Wake-up pins interrupt.
 * All other interrupts sources are disabled during POWER-DOWN
 */
wakeup_interrupts[0] = WAKEUP_WAKEUP_MAILBOX;
wakeup_interrupts[1] = 0UL;

/*
 * CPU retention Area in SRAM: bottom of RAM_03 (for example)
 */
cpu_retention_addr  = 0x20003800;

/*
 * 2 - Set up wake-up pins:
 * - All 5 wake-up pins configuration (in/out, pull-up/pull-down/) are set up via the
 *   "wakeup_io_ctrl" parameter of the POWER_SetWakeUpPins API.
 * - The 5 wake-up pins are configured as following:
```

```
     *     wake-up 0 : enabled, rising edge
     *     wake-up 1 : disabled, pull-up
     *     wake-up 2 : disabled, pull-down
     *     wake-up 3 : disabled, plain input
     *     wake-up 4 : disabled, plain input
     */

    wakeup_io_ctrl = (LOWPOWER_WAKEUPIOSRC_RISING << LOWPOWER_WAKEUPIOSRC_PIO0_INDEX) |
                     (LOWPOWER_WAKEUPIO_PULLUP << LOWPOWER_WAKEUPIO_PIO1_PULLUPDOWN_INDEX) |
                     (LOWPOWER_WAKEUPIO_PULLDOWN << LOWPOWER_WAKEUPIO_PIO2_PULLUPDOWN_INDEX) |
                    LOWPOWER_WAKEUPIO_PIO3_DISABLEPULLUPDOWN_MASK |
                    LOWPOWER_WAKEUPIO_PIO4_DISABLEPULLUPDOWN_MASK;

    POWER_SetWakeUpPins(LOWPOWER_WAKEUPIO_CFG_SRC_PMC, wakeup_io_ctrl);

    /*
     * 3 - Enter POWER-DOWN low power mode
     */
    POWER_EnterPowerDown(exclude_from_pd, sram_retention_ctrl, wakeup_interrupts, cpu_retention_addr);

    /*
     * When the wake-up pins interrupt raises, the device wakes up, then:
     * 1 - First, the WAKEUP_PINS_IRQHandler is executed.
     * 2 - Next, normal processing continues right after POWER_EnterPowerDown call
     */
     /* ... */
```

30.2.3 CPU and System frequency configuration

30.2.3.1 POWER_SetVoltageForFreq

Disposition: / Status:
All numbers given in this section MUST be reviewed once the device characterization is completed.

By default, the on-chip power regulators (DC-DC Converter, Core and Always-on Low Drop-Out regulators) output voltages are set to 1.05 volts to accommodate frequencies of 100 MHz and below. However, the voltage can be modified to accommodate higher frequency ranges (thanks to the POWER_SetVoltageForFreq API) as described in Table 285.

Table 285. On-chip power regulators output voltage settings

Range	Fequency Ranges (MHz)	On-chip Power Regulators Output (V)
1	CPU / System Frequency ≤ 100 MHz	1.0V - 1.1V
2	CPU / System Frequency >100 MHz and ≤ 130 MHz	1.025V - 1.15V
3	CPU / System Frequency > 130 MHz and ≤ 150 MHz	1.05V - 1.2V

The POWER_SetVoltageForFreq API call must always be used before initially setting the frequency and when changing the frequency from one range to another. The sequence of steps for switching from one frequency to another depends on whether the range is higher or lower.

When switching from a lower range to a higher range, the following series of steps must be followed:

1. Call the POWER_SetVoltageForFreq API.
2. Call the SDK API CLOCK_SetFLASHAccessCyclesForFreq(uint32_t iFreq) function which will set up all the necessary flash timings (both the FMC and Flash Controller).

3. Use the SDK to update the CPU/System clock frequency to the new frequency.

When switching from a higher range to a lower range, the following series of steps must be followed:

1. Use the SDK to update the system clock frequency to the new frequency.
2. Call the SDK API CLOCK_SetFLASHAccessCyclesForFreq(uint32_t iFreq) function which will set up all flash timings (both the FMC and Flash Controller).
3. Call the POWER_SetVoltageForFreq API call.

30.2.3.1.1 Detailed Description

Table 286. POWER_SetVoltageForFreq API

void	**POWER_SetVoltageForFreq**	(		
			uint32_t	**system_freq_hz**
		)		

The POWER_SetVoltageForFreq API configures the device's internal power control settings according to the calling arguments. The goal is to prepare on-chip power regulators (DC-DC Converter / Core and Always-on Low Drop-Out regulators) to deliver the amount of power needed for the requested performance level, as defined by the CPU operating frequency.

IMPORTANT

Disable all interrupts before making calls to the POWER_SetVoltageForFreq API.

Table 287. Parameters

Parameters	Description
system_freq_hz	The clock rate in Hertz (Hz) the CPU and System Bus will be using. This operand must represent an integer from 1 to 150000000 inclusive (meaning from 1 Hz to 150 MHz inclusive).

Table 288. Returned values

Return	Description
	This API does not return any value.

30.2.3.1.2 Examples

```
/*
 * 1 - Do some power related initialization of the system :
 */
if ( POWER_PowerInit() != kPOWER_Status_Success )
{
   /* On failure, stop processing */
   return 1;
}

/* On success, continue normal processing ... */

/*
 * 2 - Assumption: current CPU/System frequency is 12 MHz and
 *    We want to switch the System to a higher frequency: 150 MHz.
 */
```

```
    /*
     * a - Call the POWER_SetVoltageForFreq API
     */
    __disable_irq(); /* Disable all interrupts */
    POWER_SetVoltageForFreq(150000000);
    __enable_irq();  /* Re-enable all interrupts */

    /*
     * b - Call the SDK CLOCK_SetFLASHAccessCyclesForFreq API
     */
    /* CLOCK_SetFLASHAccessCyclesForFreq ( ...); */

    /*
     * c - Use the SDK to update the CPU/System clock to the new frequency (150 MHz)
     */

    /*
     * 3 - Assumption: current CPU/System frequency is 150 MHz and
     *    We want to switch the System to a lower frequency: 48 MHz.
     */

    /*
     * a - Use the SDK to update the CPU/System clock to the new frequency (48 MHz)
     */

    /*
     * b - Call the SDK CLOCK_SetFLASHAccessCyclesForFreq API
     */
    /* CLOCK_SetFLASHAccessCyclesForFreq ( ...); */

    /*
     * c - Call the POWER_SetVoltageForFreq API
     */
    __disable_irq(); /* Disable all interrupts */
    POWER_SetVoltageForFreq(48000000);
    __enable_irq();  /* Re-enable all interrupts */
}
```

30.2.4 Core power domain supply source functions

30.2.4.1 POWER_SetCorePowerSource

30.2.4.1.1 Detailed Description

Table 289. POWER_SetCorePowerSource API

power_status_t	**POWER_SetCorePowerSource**	(		
			power_core_pwr_source_t	pwr_source
		)		

The POWER_SetCorePowerSource API selects the core logic supply source among the DC-DC Buck Converter and the Core logic Low Drop-Out Regulator (LDO_CORE). Once a supply source is selected, the other internal sources are disabled. For example, if you select the DC-DC Buck Converter, the LDO_CORE.

Table 290. Parameters

Parameters	Description
pwr_source	Defines which regulator will be used to power the part core logic (internally). This parameter is defined according to power_core_pwr_source_t: • *kPOWER_CoreSrcDCDC*: DC-DC Buck Converter (DCDC). • *kPOWER_CoreSrcLDOCoreHP*: Core logic Low Drop-Out Regulator (LDO_CORE). This is the default core logic supply source after Power On Reset, Brown-out Detectors Resets, pin reset, Software Reset and when waking up from DEEP-POWER-DOWN.

Table 291. Returned values

Return	Description
power_status_t	This API returns one of the two following values: • kPOWER_Status_Success: on success. • kPOWER_Status_Fail: on failure : a serious issue occured during the configuration of the power regulators. It is recommended to re-call this API once again. If the error persists, it is recommended to initiate a full system software reset. See

30.2.4.1.2 Examples

```
int main()
{
   /*
    * 1 - Do some power related initialization of the system :
    */
   if ( POWER_PowerInit() != kPOWER_Status_Success )
   {
      /* On failure, stop processing */
   }

   /* On success, continue normal processing ... */

   /*
    * Set DCDC as Core power supply source
    */
   if ( POWER_SetCorePowerSource(kPOWER_CoreSrcDCDC) == kPOWER_Status_Fail )
   {
      /* On failure, stop processing */
      return 1;
   }

   /* Do some application tasks ... */

   /*
    * Set LDO_CORE as Core power supply source
    */
   if ( POWER_SetCorePowerSource(kPOWER_CoreSrcLDOCoreHP) == kPOWER_Status_Fail )
   {
      /* On failure, stop processing */
      return 1;
   }
```

```
    /* Do some application tasks ... */

}
```

30.2.4.2 POWER_GetCorePowerSource

30.2.4.2.1 Detailed Description

Table 292. POWER_GetCorePowerSource API

power_core_pwr_source_t	**POWER_GetCorePowerSource**	(
		void
		)

The POWER_GetCorePowerSource API returns the current core logic supply source.

Table 293. Parameters

Parameters	Description
	This API does not take any parameter.

Table 294. Returned values

Return	Description
power_core_pwr_source_t	This API returns one of the four following values: • *kPOWER_CoreSrcDCDC*: DC-DC Buck Converter(DCDC). • *kPOWER_CoreSrcLDOCoreHP*: Core logic Low Drop-Out Regulator (LDO_CORE).

30.2.4.2.2 Examples

```
int main()
{
   power_core_pwr_source_t core_current_power_source;

   /*
    * 1 - Do some power related initialization of the system :
    */
   if ( POWER_PowerInit() != kPOWER_Status_Success )
   {
      /* On failure, stop processing */
      return 1;
   }

   /* On success, continue normal processing ... */

   /*
    * Set DCDC as Core power supply source
    */
   if ( POWER_SetCorePowerSource(kPOWER_CoreSrcDCDC) == kPOWER_Status_Fail )
   {
```

```
        /* On failure, stop processing */
        return 1;
    }

    /* Do some application tasks ... */

    /*
     * Get current Core power supply source.
     * In this example, POWER_GetCorePowerSource()
     * will return "kPOWER_CoreSrcDCDC"
     */
    core_current_power_source = POWER_GetCorePowerSource();

    /* Do some application tasks ... */
}
```

30.2.4.3 POWER_CorePowerSourceControl

30.2.4.3.1 Detailed Description

Table 295. POWER_CorePowerSourceControl

power_status_t	**POWER_CorePowerSourceControl**	(		
			power_core_pwr_source_t	pwr_source
			power_core_pwr_state_t	pwr_state
		)		

The POWER_GetCorePowerSource API allows to control the state (enabled or disabled) of the core logic internal regulators (DCDC, LDO_CORE).

Table 296. Parameters

Parameters	Description
pwr_source	Defines which regulator will be enabled or disabled. This parameter is defined according to power_core_pwr_source_t, and it can take one of the following values: • *kPOWER_CoreSrcDCDC*: DC-DC Buck Converter (DCDC). • *kPOWER_CoreSrcLDOCoreHP*: Core logic Low Drop-Out Regulator (LDO_CORE).
pwr_state	Defines the state of the internal regulator indicated by pwr_source. This parameter is defined according to power_core_pwr_state_t, and it can take one of the two following values: • kPOWER_CorePwrDisable: disabled. • kPOWER_CorePwrEnable: enabled.

Table 297. Returned values

Return	Description
power_status_t	This API returns one of the two following values: • kPOWER_Status_Success: on success.

Table continues on the next page...

Table 297. Returned values (continued)

	• kPOWER_Status_Fail: on failure : a serious issue occured during the configuration of the power regulators. It is recommended to re-call this API once again. If the error persists, it is recommended to initiate a full system software reset. See

30.2.4.3.2 Examples

```
int main()
{
   /*
    * 1 - Do some power related initialization of the system :
    */
   if ( POWER_PowerInit() != kPOWER_Status_Success )
   {
      /* On failure, stop processing */
      return 1;
   }

   /* On success, continue normal processing ... */

   /*
    * Set DCDC as Core power supply source
    */
   if ( POWER_SetCorePowerSource(kPOWER_CoreSrcDCDC) == kPOWER_Status_Fail )
   {
      /* On failure, stop processing */
      return 1;
   }

   /* Do some application tasks ... */

   /*
    * Set LDO_CORE as Core power supply source
    */

   /* First, enable LDO_CORE ... */
   if ( POWER_CorePowerSourceControl(kPOWER_CoreSrcLDOCoreHP, kPOWER_CorePwrEnable)
== kPOWER_Status_Fail )
   {
      /* On failure, stop processing */
      return 1;
   }

   /* Next, disable DCDC ... */
   if ( POWER_CorePowerSourceControl(kPOWER_CoreSrcDCDC, kPOWER_CorePwrDisable)
== kPOWER_Status_Fail )
   {
      /* On failure, stop processing */
      return 1;
   }

   /* Do some application tasks ... */

}
```

30.2.5 SRAM power modes control functions

SRAM power state control API provide functions to configure all SRAM instances into the different low power down: Active, Deep Sleep and Shutdown modes.

30.2.5.1 POWER_SRAMPowerModeControl

30.2.5.1.1 Detailed Description

Table 298. POWER_SRAMPowerModeControl

power_status_t	**POWER_SRAMPowerModeControl**	(		
			power_sram_bit_t	sram_inst
			power_sram_pwr_mode_t	pwr_mode
		)		

The POWER_SRAMPowerModeControl API allows to configure SRAM instances (low) power modes when the part is in ACTIVE mode.

IMPORTANT

Disable all interrupts before making calls to the POWER_SRAMPowerModeControl API.

Table 299. Parameters

Parameters	Description
sram_inst	Defines the SRAM instance(s) to be configured. This parameter is defined according to power_sram_bit_t.
pwr_mode	Defines the SRAM low power mode to be applied to all SRAM instances given by sram_inst parameter. This parameter is defined according to power_sram_pwr_mode_t , and it can take one of the three following values: • *kPOWER_SRAMPwrActive*: Active (full functional mode: Read/Write accesses possible). • *kPOWER_SRAMPwrDeepSleep*: Deep Sleep (SRAM content retained, Read/Write accesses not possible). • *kPOWER_SRAMPwrShutDown*: Shutdown (SRAM content lost, Read/Write accesses not possible).

Table 300. Returned values

Return	Description
power_status_t	This API returns one of the two following values: • kPOWER_Status_Success: on success. • kPOWER_Status_Fail: on failure : non-supported transition encountered (like for example transitioning from Shuntdown low power mode to Deep Sleep low power mode directly).

30.2.5.1.2 Examples

```
int main()
{
   /*
```

```
     * 1 - Do some power related initialization of the system :
     */
    if ( POWER_PowerInit() != kPOWER_Status_Success )
    {
       /* On failure, stop processing */
       return 1;
    }

    /* On success, continue normal processing ... */


    /* Switch RAM_30 to SRAM Deep Sleep low power mode
     * (data retained, read/write operations not allowed)
     */
    __disable_irq(); /* Disable all interrupts */
    if ( POWER_SRAMPowerModeControl(kPOWER_SRAM_RAM_30, kPOWER_SRAMPwrDeepSleep) !
= kPOWER_Status_Success )
    {
       /* On failure, take any appropriate actions ... */
       return 1;
    }

    /* Switch RAM_X0 to SRAM Shutdown low power mode
     * (data lost, read/write operations not allowed)
     */
    if ( POWER_SRAMPowerModeControl(kPOWER_SRAM_RAM_X0, kPOWER_SRAMPwrShutDown) !
= kPOWER_Status_Success )
    {
       /* On failure, take any appropriate actions ... */
       return 1;
    }
    __enable_irq();  /* Re-enable all interrupts */

    /* Do some application tasks ... */

    /* ... then switch back RAM_X0 & and RAM_30 in Active Mode */
    __disable_irq(); /* Disable all interrupts */
    if (
        POWER_SRAMPowerModeControl(
          (power_sram_bit_t)((uint32_t)kPOWER_SRAM_RAM_X0 |
(uint32_t)kPOWER_SRAM_RAM_30), kPOWER_SRAMPwrActive)
          != kPOWER_Status_Success
       )
    {
       /* On failure, take any appropriate actions ... */
       return 1;
    }
    __enable_irq();  /* Re-enable all interrupts */

    /* Do some application tasks ... */

}
```

30.2.5.2 POWER_GetSRAMPowerMode

30.2.5.2.1 Detailed Description

Table 301. POWER_SRAMPowerModeControl

power_sram_pwr_mode_t	**POWER_GetSRAMPowerMode**	(		
			power_sram_index_t	sram_index
		)		

The POWER_GetSRAMPowerMode API allows to determine the power mode of a single SRAM instance when the part is in ACTIVE mode.

Table 302. Parameters

Parameters	Description
sram_index	Defines the SRAM instance index for which you want to now the (current) power mode. This parameter is defined according to power_sram_index_t.

Table 303. Returned values

Return	Description
power_sram_pwr_mode_t	This API returns one of the three following values: • *kPOWER_SRAMPwrActive*: Active (full functional mode: Read/Write accesses possible). • *kPOWER_SRAMPwrDeepSleep*: Deep Sleep (SRAM content retained, Read/Write accesses not possible). • *kPOWER_SRAMPwrShutDown*: Shutdown (SRAM content lost, Read/Write accesses not possible).

30.2.5.2.2 Examples

```
int main()
{
   /*
    * 1 - Do some power related initialization of the system :
    */
   if ( POWER_PowerInit() != kPOWER_Status_Success )
   {
      /* On failure, stop processing */
      return 1;
   }

   /* On success, continue normal processing ... */

   /* Switch RAM_30 to SRAM Deep Sleep low power mode
    * (data retained, read/write operations not allowed)
    */
   if ( POWER_SRAMPowerModeControl(kPOWER_SRAM_RAM_30, kPOWER_SRAMPwrDeepSleep) !
= kPOWER_Status_Success )
   {
      /* On failure, take any appropriate actions ... */
      return 1;
   }

   /* Switch RAM_X0 to SRAM Shutdown low power mode
```

```
     * (data lost, read/write operations not allowed)
     */
    if ( POWER_SRAMPowerModeControl(kPOWER_SRAM_RAM_X0, kPOWER_SRAMPwrShutDown) !
= kPOWER_Status_Success )
    {
       /* On failure, take any appropriate actions ... */
       return 1;
    }

    /* Do some application tasks ... */

    /* Check RAM_30 current power mode
     * (Expecting "kPOWER_SRAMPwrDeepSleep")
     */
    if ( POWER_GetSRAMPowerMode(kPOWER_SRAM_IDX_RAM_30) != kPOWER_SRAMPwrDeepSleep )
    {
       /* On failure, take any appropriate actions ... */
       return 1;
    }

    /* Check RAM_X0 current power mode
     * (Expecting "kPOWER_SRAMPwrShutDown")
     */
    if ( POWER_GetSRAMPowerMode(kPOWER_SRAM_IDX_RAM_X0) != kPOWER_SRAMPwrShutDown )
    {
       /* On failure, take any appropriate actions ... */
       return 1;
    }

}
```

30.3 Power Library Types and Defines

30.3.1 power_pd_bit_t

```
/**
 * @brief analog components power modes control during low power modes
 */
typedef enum _power_pd_bit
{
    /* Power Down Vector 0 */
    kPDRUNCFG_PD_DCDC          = (1UL << 0),
    kPDRUNCFG_PD_BIAS          = (1UL << 1),
    kPDRUNCFG_PD_BODCORE       = (1UL << 2),
    kPDRUNCFG_PD_BODVDDMAIN    = (1UL << 3),
    kPDRUNCFG_PD_FRO1M         = (1UL << 4),
    kPDRUNCFG_PD_FRO192M       = (1UL << 5),
    kPDRUNCFG_PD_FRO32K        = (1UL << 6),
    kPDRUNCFG_PD_XTAL32K       = (1UL << 7),
    kPDRUNCFG_PD_XTALHS        = (1UL << 8),
    kPDRUNCFG_PD_PLL0          = (1UL << 9),
    kPDRUNCFG_PD_PLL1          = (1UL << 10),
    kPDRUNCFG_PD_USBFSPHY      = (1UL << 11),
    // kPDRUNCFG_PD_           = (1UL << 12), /*!< RESERVED */
    kPDRUNCFG_PD_COMP          = (1UL << 13),
    // kPDRUNCFG_PD_           = (1UL << 14), /*!< RESERVED */
```

```
    // kPDRUNCFG_PD_          = (1UL << 15), /*!< RESERVED */
    kPDRUNCFG_PD_LDOMEM       = (1UL << 16),
    // kPDRUNCFG_PD_          = (1UL << 17), /*!< RESERVED */
    kPDRUNCFG_PD_LDOEFUSEPROG = (1UL << 18),
    // kPDRUNCFG_PD_          = (1UL << 19), /*!< RESERVED */
    kPDRUNCFG_PD_LDOXTALHF    = (1UL << 20),
    kPDRUNCFG_PD_LDOFLASHNV   = (1UL << 21),
    // kPDRUNCFG_PD_          = (1UL << 22), /*!< RESERVED */
    kPDRUNCFG_PD_PLL0_SSCG    = (1UL << 23),
    kPDRUNCFG_PD_ROM          = (1UL << 24),
    kPDRUNCFG_PD_HSCMP0       = (1UL << 25),
    kPDRUNCFG_PD_HSCMP1       = (1UL << 26),
    kPDRUNCFG_PD_HSCMP2       = (1UL << 27),
    kPDRUNCFG_PD_OPAMP0       = (1UL << 28),
    kPDRUNCFG_PD_OPAMP1       = (1UL << 29),
    kPDRUNCFG_PD_OPAMP2       = (1UL << 30),
    kPDRUNCFG_PD_VREF         = (1UL << 31),

    /* Power Down Vector 1 */
    kPDRUNCFG_PD_CMPBIAS      = (1UL << 0),
    kPDRUNCFG_PD_HSCMP0_DAC   = (1UL << 1),
    kPDRUNCFG_PD_HSCMP1_DAC   = (1UL << 2),
    kPDRUNCFG_PD_HSCMP2_DAC   = (1UL << 3),
    kPDRUNCFG_PD_DAC0         = (1UL << 4),
    kPDRUNCFG_PD_DAC1         = (1UL << 5),
    kPDRUNCFG_PD_DAC2         = (1UL << 6),
    kPDRUNCFG_STOP_DAC0       = (1UL << 7),
    kPDRUNCFG_STOP_DAC1       = (1UL << 8),
    kPDRUNCFG_STOP_DAC2       = (1UL << 9),
} power_pd_bit_t;
```

30.3.2 Low Power Modes Wake up sources (#define WAKEUP_*)

```
/**
 * @brief Low Power Modes Wake up sources
 */
/* Wake up source vector 0 */
#define WAKEUP_SYS                 (1UL << 0)
#define WAKEUP_SDMA0               (1UL << 1)
#define WAKEUP_GPIO_GLOBALINT0      (1UL << 2)
#define WAKEUP_GPIO_GLOBALINT1      (1UL << 3)
#define WAKEUP_GPIO_INT0_0         (1UL << 4)
#define WAKEUP_GPIO_INT0_1         (1UL << 5)
#define WAKEUP_GPIO_INT0_2         (1UL << 6)
#define WAKEUP_GPIO_INT0_3         (1UL << 7)
#define WAKEUP_UTICK               (1UL << 8)
#define WAKEUP_MRT                 (1UL << 9)
#define WAKEUP_CTIMER0             (1UL << 10)
#define WAKEUP_CTIMER1             (1UL << 11)
#define WAKEUP_SCT                 (1UL << 12)
#define WAKEUP_CTIMER3             (1UL << 13)
#define WAKEUP_FLEXCOMM0           (1UL << 14)
#define WAKEUP_FLEXCOMM1           (1UL << 15)
#define WAKEUP_FLEXCOMM2           (1UL << 16)
#define WAKEUP_FLEXCOMM3           (1UL << 17)
#define WAKEUP_FLEXCOMM4           (1UL << 18)
```

```
#define WAKEUP_FLEXCOMM5           (1UL << 19)
#define WAKEUP_FLEXCOMM6           (1UL << 20)
#define WAKEUP_FLEXCOMM7           (1UL << 21)
#define WAKEUP_ADC0                (1UL << 22)
#define WAKEUP_ADC1                (1UL << 23)
#define WAKEUP_ACMP                (1UL << 24)
#define WAKEUP_DMIC                (1UL << 25)
#define WAKEUP_HWVAD               (1UL << 26)
#define WAKEUP_USB0_NEEDCLK        (1UL << 27)
#define WAKEUP_USB0                (1UL << 28)
#define WAKEUP_RTC_ALARM_WAKEUP    (1UL << 29)

// reserved                        (1UL << 30)

#define WAKEUP_WAKEUP_MAILBOX      (1UL << 31)

/* Wake up source vector 1 */
#define WAKEUP_GPIO_INT0_4         (1UL << 0)
#define WAKEUP_GPIO_INT0_5         (1UL << 1)
#define WAKEUP_GPIO_INT0_6         (1UL << 2)
#define WAKEUP_GPIO_INT0_7         (1UL << 3)
#define WAKEUP_CTIMER2             (1UL << 4)
#define WAKEUP_CTIMER4             (1UL << 5)
#define WAKEUP_OS_EVENT_TIMER      (1UL << 6)
#define WAKEUP_FLEXSPI             (1UL << 7)
// reserved                        (1UL << 8)
// reserved                        (1UL << 9)
// reserved                        (1UL << 10)
#define WAKEUP_CAN0_0              (1UL << 11)
#define WAKEUP_CAN0_1              (1UL << 12)
#define WAKEUP_SPIFILTER           (1UL << 13)
// reserved                        (1UL << 14)
// reserved                        (1UL << 15)
// reserved                        (1UL << 16)

// reserved                        (1UL << 17)

// reserved                        (1UL << 18)

// reserved                        (1UL << 19)
// reserved                        (1UL << 20)

// reserved                        (1UL << 21)

// reserved                        (1UL << 22)

// reserved                        (1UL << 23)

// reserved                        (1UL << 24)
#define WAKEUP_PQ                  (1UL << 25)
#define WAKEUP_SDMA1               (1UL << 26)
#define WAKEUP_LSPI_HS             (1UL << 27)

// reserved                        (1UL << 28)
// reserved                        (1UL << 29)
#define WAKEUP_I3C                 (1UL << 30)
// reserved                        (1UL << 31)

/* Wake up source vector 2 */
// reserved                        (1UL << 0)
```

```
// reserved                        (1UL << 1)

// reserved                        (1UL << 2)

// reserved                        (1UL << 3)
// reserved                        (1UL << 4)
// reserved                        (1UL << 5)
// reserved                        (1UL << 6)
// reserved                        (1UL << 7)
// reserved                        (1UL << 8)
// reserved                        (1UL << 9)
#define WAKEUP_DAC0                (1UL << 10)
#define WAKEUP_DAC1                (1UL << 11)
#define WAKEUP_DAC2                (1UL << 12)
#define WAKEUP_HS_COMP0            (1UL << 13)
#define WAKEUP_HS_COMP1            (1UL << 14)
#define WAKEUP_HS_COMP2            (1UL << 15)
#define WAKEUP_FLEXPWM0_CAPTURE    (1UL << 16)
#define WAKEUP_FLEXPWM0_FAULT      (1UL << 17)
#define WAKEUP_FLEXPWM0_RELOAD_ERROR (1UL << 18)
#define WAKEUP_FLEXPWM0_COMPARE0   (1UL << 19)
#define WAKEUP_FLEXPWM0_RELOAD0    (1UL << 20)
#define WAKEUP_FLEXPWM0_COMPARE1   (1UL << 21)
#define WAKEUP_FLEXPWM0_RELOAD1    (1UL << 22)
#define WAKEUP_FLEXPWM0_COMPARE2   (1UL << 23)
#define WAKEUP_FLEXPWM0_RELOAD2    (1UL << 24)
#define WAKEUP_FLEXPWM0_COMPARE3   (1UL << 25)
#define WAKEUP_FLEXPWM0_RELOAD3    (1UL << 26)
#define WAKEUP_FLEXPWM1_CAPTURE    (1UL << 27)
#define WAKEUP_FLEXPWM1_FAULT      (1UL << 28)
#define WAKEUP_FLEXPWM1_RELOAD_ERROR (1UL << 29)
#define WAKEUP_FLEXPWM1_COMPARE0   (1UL << 30)
#define WAKEUP_FLEXPWM1_RELOAD0    (1UL << 31)

/* Wake up source vector 3 */
#define WAKEUP_FLEXPWM1_COMPARE1   (1UL << 0)
#define WAKEUP_FLEXPWM1_RELOAD1    (1UL << 1)
#define WAKEUP_FLEXPWM1_COMPARE2   (1UL << 2)
#define WAKEUP_FLEXPWM1_RELOAD2    (1UL << 3)
#define WAKEUP_FLEXPWM1_COMPARE3   (1UL << 4)
#define WAKEUP_FLEXPWM1_RELOAD3    (1UL << 5)
#define WAKEUP_ENC0_COMPARE        (1UL << 6)
#define WAKEUP_ENC0_HOME           (1UL << 7)
#define WAKEUP_ENC0_WDG            (1UL << 8)
#define WAKEUP_ENC0_IDX            (1UL << 9)
#define WAKEUP_ENC1_COMPARE        (1UL << 10)
#define WAKEUP_ENC1_HOME           (1UL << 11)
#define WAKEUP_ENC1_WDG            (1UL << 12)
#define WAKEUP_ENC1_IDX            (1UL << 13)
// reserved                        (1UL << 14)
#define WAKEUP_CF_DSP24L_IRQ0      (1UL << 15)
#define WAKEUP_CF_DSP24L_IRQ1      (1UL << 16)
#define WAKEUP_FTM0                (1UL << 17)
// reserved                        (1UL << 18)
// reserved                        (1UL << 19)
// reserved                        (1UL << 20)
// reserved                        (1UL << 21)
// reserved                        (1UL << 22)
// reserved                        (1UL << 23)
```

```
// reserved                          (1UL << 24)
// reserved                          (1UL << 25)
// reserved                          (1UL << 26)
// reserved                          (1UL << 27)
// reserved                          (1UL << 28)
// reserved                          (1UL << 29)
// reserved                          (1UL << 30)
// reserved                          (1UL << 31)
```

30.3.3 Wake up I/O sources (#define LOWPOWER_WAKEUPIO*)

```
/**
 * @brief Wake up I/O sources (DEEP POWER-DOWN)
 */
#define LOWPOWER_WAKEUPIOSRC_PIO0_INDEX 0 /*!< Pin P1( 1) */
#define LOWPOWER_WAKEUPIOSRC_PIO1_INDEX 2 /*!< Pin P0(28) */
#define LOWPOWER_WAKEUPIOSRC_PIO2_INDEX 4 /*!< Pin P1(18) */
#define LOWPOWER_WAKEUPIOSRC_PIO3_INDEX 6 /*!< Pin P1(30) */
#define LOWPOWER_WAKEUPIOSRC_PIO4_INDEX 8 /*!< Pin P0(26) */

#define LOWPOWER_WAKEUPIOSRC_DISABLE        0 /*!< Wake up is disable                      */
#define LOWPOWER_WAKEUPIOSRC_RISING         1 /*!< Wake up on rising edge                  */
#define LOWPOWER_WAKEUPIOSRC_FALLING        2 /*!< Wake up on falling edge                 */
#define LOWPOWER_WAKEUPIOSRC_RISING_FALLING 3 /*!< Wake up on both rising or falling edges */

#define LOWPOWER_WAKEUPIO_PIO0_DISABLEPULLUPDOWN_INDEX \
    20 /*!< Wake-up I/O 0 pull-up/down disable/enable control index */
#define LOWPOWER_WAKEUPIO_PIO1_DISABLEPULLUPDOWN_INDEX \
    21 /*!< Wake-up I/O 1 pull-up/down disable/enable control index */
#define LOWPOWER_WAKEUPIO_PIO2_DISABLEPULLUPDOWN_INDEX \
    22 /*!< Wake-up I/O 2 pull-up/down disable/enable control index */
#define LOWPOWER_WAKEUPIO_PIO3_DISABLEPULLUPDOWN_INDEX \
    23 /*!< Wake-up I/O 3 pull-up/down disable/enable control index */
#define LOWPOWER_WAKEUPIO_PIO4_DISABLEPULLUPDOWN_INDEX \
    24 /*!< Wake-up I/O 4 pull-up/down disable/enable control index */

#define LOWPOWER_WAKEUPIO_PIO0_DISABLEPULLUPDOWN_MASK \
    (1UL << LOWPOWER_WAKEUPIO_PIO0_DISABLEPULLUPDOWN_INDEX) /*!< Wake-up I/O 0 pull-up/down disable/
enable mask */
#define LOWPOWER_WAKEUPIO_PIO1_DISABLEPULLUPDOWN_MASK \
    (1UL << LOWPOWER_WAKEUPIO_PIO1_DISABLEPULLUPDOWN_INDEX) /*!< Wake-up I/O 1 pull-up/down disable/
enable mask */
#define LOWPOWER_WAKEUPIO_PIO2_DISABLEPULLUPDOWN_MASK \
    (1UL << LOWPOWER_WAKEUPIO_PIO2_DISABLEPULLUPDOWN_INDEX) /*!< Wake-up I/O 2 pull-up/down disable/
enable mask */
#define LOWPOWER_WAKEUPIO_PIO3_DISABLEPULLUPDOWN_MASK \
    (1UL << LOWPOWER_WAKEUPIO_PIO3_DISABLEPULLUPDOWN_INDEX) /*!< Wake-up I/O 3 pull-up/down disable/
enable mask */
#define LOWPOWER_WAKEUPIO_PIO4_DISABLEPULLUPDOWN_MASK \
    (1UL << LOWPOWER_WAKEUPIO_PIO4_DISABLEPULLUPDOWN_INDEX) /*!< Wake-up I/O 4 pull-up/down disable/
enable mask */

#define LOWPOWER_WAKEUPIO_PIO0_PULLUPDOWN_INDEX 25 /*!< Wake-up I/O 0 pull-up/down configuration
index */
#define LOWPOWER_WAKEUPIO_PIO1_PULLUPDOWN_INDEX 26 /*!< Wake-up I/O 1 pull-up/down configuration
index */
```

```
#define LOWPOWER_WAKEUPIO_PIO2_PULLUPDOWN_INDEX 27 /*!< Wake-up I/O 2 pull-up/down configuration
index */
#define LOWPOWER_WAKEUPIO_PIO3_PULLUPDOWN_INDEX 28 /*!< Wake-up I/O 3 pull-up/down configuration
index */
#define LOWPOWER_WAKEUPIO_PIO4_PULLUPDOWN_INDEX 29 /*!< Wake-up I/O 4 pull-up/down configuration
index */

#define LOWPOWER_WAKEUPIO_PIO0_PULLUPDOWN_MASK \
    (1UL << LOWPOWER_WAKEUPIO_PIO0_PULLUPDOWN_INDEX) /*!< Wake-up I/O 0 pull-up/down mask */
#define LOWPOWER_WAKEUPIO_PIO1_PULLUPDOWN_MASK \
    (1UL << LOWPOWER_WAKEUPIO_PIO1_PULLUPDOWN_INDEX) /*!< Wake-up I/O 1 pull-up/down mask */
#define LOWPOWER_WAKEUPIO_PIO2_PULLUPDOWN_MASK \
    (1UL << LOWPOWER_WAKEUPIO_PIO2_PULLUPDOWN_INDEX) /*!< Wake-up I/O 2 pull-up/down mask */
#define LOWPOWER_WAKEUPIO_PIO3_PULLUPDOWN_MASK \
    (1UL << LOWPOWER_WAKEUPIO_PIO3_PULLUPDOWN_INDEX) /*!< Wake-up I/O 3 pull-up/down mask */
#define LOWPOWER_WAKEUPIO_PIO4_PULLUPDOWN_MASK \
    (1UL << LOWPOWER_WAKEUPIO_PIO4_PULLUPDOWN_INDEX) /*!< Wake-up I/O 4 pull-up/down mask */

#define LOWPOWER_WAKEUPIO_PULLDOWN 0 /*!< Select pull-down */
#define LOWPOWER_WAKEUPIO_PULLUP   1 /*!< Select pull-up   */

#define LOWPOWER_WAKEUPIO_CFG_SRC_IOCON 0 /*!< Wake-up pins configuration (in/out, pull up/down plain
input ...) is coming from IOCON (valid for DEEP-SLEEP and POWER-DOWN) */
#define LOWPOWER_WAKEUPIO_CFG_SRC_PMC   1 /*!< Wake-up pins configuration (in/out, pull up/down plain
input ...) is coming from PMC and set up via
                                            the second parameter (wakeup_io_ctrl) of
POWER_SetWakeUpPins API (valid for DEEP-SLEEP and POWER-DOWN) */
```

30.3.4 power_status_t

```
/*@brief Generic Power Library APIs Status codes */
typedef enum _power_status
{
   kPOWER_Status_Success = 0U, /*!< OK */
   kPOWER_Status_Fail    = 1U, /*!< Generic error code */
} power_status_t;
```

30.3.5 power_core_pwr_source_t

```
/*@brief Core Power Source */
typedef enum _power_core_pwr_source
{
   kPOWER_CoreSrcDCDC      = 0U, /*!< DCDC */
   kPOWER_CoreSrcLDOCoreHP = 1U, /*!< LDO Core
} power_core_pwr_source_t;
```

30.3.6 power_core_pwr_state_t

```
/*@brief Core Regulators Power State */
typedef enum _power_core_pwr_state
{
   kPOWER_CorePwrDisable = 0U, /*!< Disable */
   kPOWER_CorePwrEnable  = 1U, /*!< Enable */
} power_core_pwr_state_t;
```

30.3.7 power_sram_bit_t

```
/**
 * @brief SRAM instances bit masks
 */
typedef enum _power_sram_bit
{
   kPOWER_SRAM_RAM_X0        = (1UL << 0),  /*!< RAM_X0 */
   kPOWER_SRAM_RAM_00        = (1UL << 1),  /*!< RAM_00 */
   kPOWER_SRAM_RAM_01        = (1UL << 2),  /*!< RAM_01 */
   kPOWER_SRAM_RAM_02        = (1UL << 3),  /*!< RAM_02 */
   kPOWER_SRAM_RAM_03        = (1UL << 4),  /*!< RAM_03 */
   kPOWER_SRAM_RAM_10        = (1UL << 5),  /*!< RAM_10 */
   kPOWER_SRAM_RAM_20        = (1UL << 6),  /*!< RAM_20 */
   kPOWER_SRAM_RAM_30        = (1UL << 7),  /*!< RAM_30 */
   kPOWER_SRAM_RAM_40        = (1UL << 8),  /*!< RAM_40 */
   kPOWER_SRAM_RAM_41        = (1UL << 9),  /*!< RAM_41 */
   kPOWER_SRAM_RAM_42        = (1UL << 10), /*!< RAM_42 */
   kPOWER_SRAM_RAM_43        = (1UL << 11), /*!< RAM_43 */
} power_sram_bit_t;
```

30.3.8 power_sram_index_t

```
/**
 * @brief SRAM instances indexes
 */
typedef enum _power_sram_index
{
   kPOWER_SRAM_IDX_RAM_X0        = 0UL,  /*!< RAM_X0 */
   kPOWER_SRAM_IDX_RAM_00        = 1UL,  /*!< RAM_00 */
   kPOWER_SRAM_IDX_RAM_01        = 2UL,  /*!< RAM_01 */
   kPOWER_SRAM_IDX_RAM_02        = 3UL,  /*!< RAM_02 */
   kPOWER_SRAM_IDX_RAM_03        = 4UL,  /*!< RAM_03 */
   kPOWER_SRAM_IDX_RAM_10        = 5UL,  /*!< RAM_10 */
   kPOWER_SRAM_IDX_RAM_20        = 6UL,  /*!< RAM_20 */
   kPOWER_SRAM_IDX_RAM_30        = 7UL,  /*!< RAM_30 */
   kPOWER_SRAM_IDX_RAM_40        = 8UL,  /*!< RAM_40 */
   kPOWER_SRAM_IDX_RAM_41        = 9UL,  /*!< RAM_41 */
   kPOWER_SRAM_IDX_RAM_42        = 10UL, /*!< RAM_42 */
   kPOWER_SRAM_IDX_RAM_43        = 11UL, /*!< RAM_43 */
} power_sram_index_t;
```

30.3.9 power_sram_pwr_mode_t

```
/*@brief SRAM Power Mode */
typedef enum _power_sram_pwr_mode
{
   kPOWER_SRAMPwrActive      = 0U, /*!< Active */
   kPOWER_SRAMPwrDeepSleep   = 2U, /*!< Deep Sleep : SRAM content retained */
   kPOWER_SRAMPwrShutDown    = 3U, /*!< Shutdown: SRAM content lost */
} power_sram_pwr_mode_t;
```

30.3.10 power_reset_cause_t

```
/**
 * @brief Device Reset Causes
 */
typedef enum _power_reset_cause
{
   kRESET_CAUSE_POR                          = 0UL,  /*!< Power On Reset */
   kRESET_CAUSE_PADRESET                     = 1UL,  /*!< Hardware Pin Reset */
   kRESET_CAUSE_BODRESET                     = 2UL,  /*!< Brown-out Detector reset (either
BOD_VDDMAIN or BODCORE) */
   kRESET_CAUSE_ARMSYSTEMRESET               = 3UL,  /*!< ARM System Reset */
   kRESET_CAUSE_WDTRESET                     = 4UL,  /*!< Watchdog Timer Reset */
   kRESET_CAUSE_SWRRESET                     = 5UL,  /*!< Software Reset */

   /* Reset causes in DEEP-POWER-DOWN low power mode */
   kRESET_CAUSE_DPDRESET_WAKEUPIO            = 7UL,  /*!< Any of the 5 wake-up pins */
   kRESET_CAUSE_DPDRESET_RTC                 = 8UL,  /*!< Real Time Clock (RTC) */
   kRESET_CAUSE_DPDRESET_OSTIMER             = 9UL,  /*!< OS Event Timer (OSTIMER) */
   kRESET_CAUSE_DPDRESET_WAKEUPIO_RTC        = 10UL, /*!< Any of the 5 wake-up pins and RTC (the 2
events occured within 1 nano-second of each other) */
   kRESET_CAUSE_DPDRESET_WAKEUPIO_OSTIMER    = 11UL, /*!< Any of the 5 wake-up pins and OSTIMER (the
2 events occured within 1 nano-second of each other) */
   kRESET_CAUSE_DPDRESET_RTC_OSTIMER         = 12UL, /*!< Real Time Clock or OS Event Timer (the 2
events occured within 1 nano-second of each other) */
   kRESET_CAUSE_DPDRESET_WAKEUPIO_RTC_OSTIMER = 13UL, /*!< Any of the 5 wake-up pins or RTC or OS
Event Timer (the 3 events occured within 1 nano-second of each other) */
   /* Miscallenous */
   kRESET_CAUSE_NOT_RELEVANT                 = 14UL, /*!< No reset cause (for example, this code is
used when waking up from DEEP-SLEEP low power mode) */
   kRESET_CAUSE_NOT_DETERMINISTIC            = 15UL, /*!< Unknown Reset Cause. Should be treated
like "Hardware Pin Reset" from an application point of view. */
} power_reset_cause_t;
```

30.3.11 power_boot_mode_t

```
/**
 * @brief Device Boot Modes
 */
typedef enum _power_boot_mode
{
```

```
   kBOOT_MODE_POWER_UP           = 0UL,  /*!< All non Low Power Mode wake up (Power On Reset, Pin
Reset, BoD Reset, ARM System Reset ... ) */
   kBOOT_MODE_LP_DEEP_SLEEP      = 1UL,  /*!< Wake up from DEEP-SLEEP Low Power mode */
   kBOOT_MODE_LP_POWER_DOWN      = 2UL,  /*!< Wake up from POWER-DOWN Low Power mode */
   kBOOT_MODE_LP_DEEP_POWER_DOWN = 4UL,  /*!< Wake up from DEEP-POWER-DOWN Low Power mode */
} power_boot_mode_t;
```

30.3.12 power_wakeup_pin_t

```
/**
 * @brief Device wake up pins events
 */
typedef enum _power_wakeup_pin_t
{
   kWAKEUP_PIN_NONE     = 0UL,         /*!< No wake up pin event */
   kWAKEUP_PIN_0        = (1UL << 0),  /*!< Wake up pin 0 event */
   kWAKEUP_PIN_1        = (1UL << 1),  /*!< Wake up pin 1 event */
   kWAKEUP_PIN_2        = (1UL << 2),  /*!< Wake up pin 2 event */
   kWAKEUP_PIN_3        = (1UL << 3),  /*!< Wake up pin 3 event */
   kWAKEUP_PIN_4        = (1UL << 4),  /*!< Wake up pin 4 event */
   kWAKEUP_PIN_MULTIPLE = 0x1FUL,      /*!< More than 1 wake up pins events occured (within 1 nano-
second of each other) */
} power_wakeup_pin_t;
```

Chapter 31
Life Cycle States

31.1 Customer Life Cycle States for LPC553x devices

31.1.1 Overview

LPC553x devices support a life cycle state model to protect code from reading from the device internal flash. This feature is called code read protection feature. Code Read Protection is a mechanism that allows the user to enable different levels of protections in the system, so that access to the on-chip flash and use of the ISP can be restricted. The current life cycle state of the device determines the debug access and ISP command availability. The life cycle state is controlled by the LC_STATE fuse value, and state values are selected so that additional fuse bits are burned to advance the state. Because fuses control the life cycle state, moving to a more advanced state is an irreversible and permanent process. The life cycle can only be advanced and can't return to a previous state. The Boot ROM is responsible for checking the life cycle state. Based on the life cycle state the Boot ROM will determine what boot flow is used, including if control will be passed to application code or not. The ROM also handles the opening of ISP and debug ports based on the life cycle state. If the part is in the Bricked state or any invalid life cycle state, then the ROM will lock the part.

Code read protection has 5 levels:

- CRP_LEVEL0
- CRP_LEVEL1
- CRP_LEVEL2
- CRP_LEVEL3
- CRP_LEVEL4

Below table summarizes the code read protection levels, corresponding LC_STATE fuse value, ISP commands and Debug mailbox commands.

Table 304. Code read protection levels

CRP_LEVEL	LC_STATE(7:0) Fuses	ISP commands	Debug Access
CRP_LEVEL0 (LC_STATE = 0x03)	0000_0011	• Full ISP Command Set[1] available • ISP entry points: — ISP pin assert — ISP fall through — runBootloader ROM API	• CM33 debug disabled by default . • ROM enables debug access and locks debug control registers on rom code exit. • Available debug mail box commands: — GET_CRP_LEVEL (0x0002) — BULK_ERASE_FLASH (0x0003) — ENTER_ISP_MODE (0x0005) — START_DEBUG_SESSION (0x0007)

Table continues on the next page...

Table 304. Code read protection levels (continued)

CRP_LEVEL	LC_STATE(7:0) Fuses	ISP commands	Debug Access
CRP_LEVEL1 (LC_STATE = 0x07)	0000_0111	• Reduced ISP Command Set [2] available. • ISP entry points: — ISP pin assert — ISP fall through — runBootloader ROM API	• CM33 debug disabled by default. • ROM doesn't lock debug control registers by writing 0xA to DEBUG_LOCK_EN register on rom code exit. • User code can enable debug access by writing to unlocked debug control registers. • Available debug mail box commands: — GET_CRP_LEVEL (0x0002) — BULK_ERASE_FLASH (0x0003) — ENTER_ISP_MODE (0x0005)
CRP_LEVEL2 (LC_STATE = 0x0F)	0000_1111	• ISP is disabled if flash is not blank. ISP entry through runBootloader ROM_API call is still possible. • If flash is blank, Reduced ISP Command Set[2] is available through following entry point: — ISP pin assert — ISP fall through — runBootloader ROM API	• CM33 debug disabled by default. • ROM disables debug access and locks debug control register on user code exit. • Available debug mail box commands: — GET_CRP_LEVEL (0x0002) — BULK_ERASE_FLASH (0x0003).
CRP_LEVEL3 (LC_STATE = 0x0F) and CMPA.ISP_mode= Disable	0000_1111	• Reduced ISP Command Set[2] is available. • ISP is disabled except through runBootloader API call.	• CM33 debug disabled by default. • ROM disables debug access and locks debug control register on user code exit. • Available debug mail box commands:

Table continues on the next page...

Table 304. Code read protection levels (continued)

CRP_LEVEL	LC_STATE(7:0) Fuses	ISP commands	Debug Access
			— GET_CRP_LEVEL (0x0002) — BULK_ERASE_FLASH (0x0003)
CRP_LEVEL4 (LC_STATE = 0xCF) and CMPA.ISP_mode= Disable	1100_1111	• Reduced ISP Command Set[2] is available. • ISP is disabled except through runBootloader API call.	• CM33 debug disabled by default. • ROM disables debug access and locks debug control register on user code exit. • Available debug mail box commands: — GET_CRP_LEVEL (0x0002)
(Field Return OEM) LC_STATE = 0x1F		ISP is disabled.	CM33 debug is enabled and ROM stays in while(1) loop
(Failure Analysis NXP) LC_STATE = 0x7F		ISP is disabled.	CM33 debug is enabled. But user flash is erased before ROM enters in while(1) loop
All other LC_STATE values (Should not be used)		• Reduced ISP Command Set[2] is available. • ISP entry points: — ISP pin assert ISP fall through — runBootloader ROM API	• CM33 debug disabled by default. • ROM disables debug access and locks debug control register on user code exit. • Available debug mail box commands: — GET_CRP_LEVEL (0x0002)

1. Full ISP Command Set contains FlashEraseAll, FlashEraseRegion, Reset, WriteMemory, SetProperty, GetProperty, ConfigureMemory, FillMemory, eFuseProgram/FlashProgramOnce, eFuseRead/FlashReadOnce, Execute, ReadMemory.
2. Reduced ISP command set: FlashEraseAll, Reset, WriteMemory, SetProperty, GetProperty, ConfigureMemory, FillMemory.

Chapter 32
Cyclic Redundancy Check (CRC)

32.1 Chip-specific CRC information

Table 305. Reference links to related information

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

32.1.1 Module instances

This device has one instance of the CRC module, CRC.

32.2 Overview

The Cyclic Redundancy Check (CRC) module generates 16/32-bit CRC code for error detection.

The CRC module provides a programmable polynomial and other parameters required to implement a 16-bit or 32-bit CRC standard.

The 16/32-bit code is calculated for 32 bits of data at a time.

32.2.1 Block diagram

The following is a block diagram of the CRC.

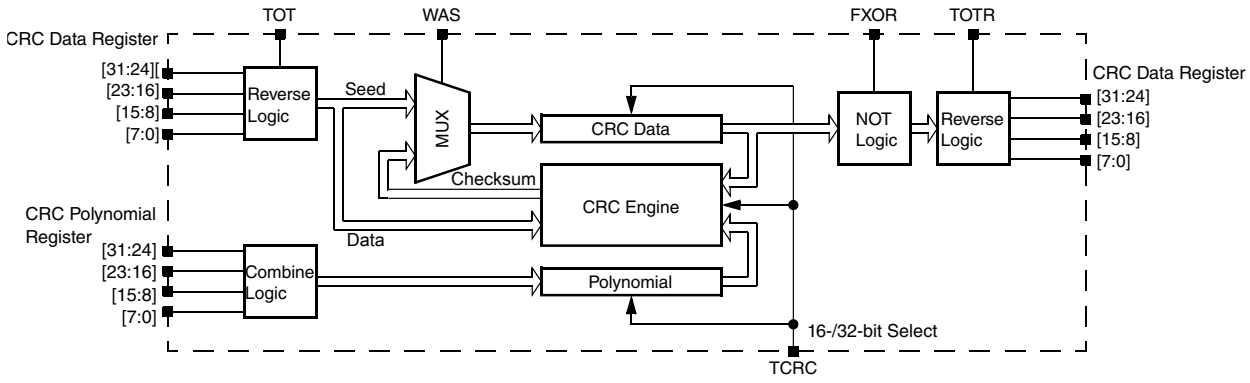


Figure 130. Programmable cyclic redundancy check (CRC) block diagram

32.2.2 Features

Following are the features of the CRC module.

- Hardware CRC generator circuit using a 16-bit or 32-bit programmable shift register

- Programmable initial seed value and polynomial
- Option to transpose input data or output data (the CRC result) bitwise or bytewise. This option is required for certain CRC standards. A bytewise transpose operation is not possible when accessing the CRC data register via 8-bit accesses. In this case, the user's software must perform the bytewise transpose function.
- Option for inversion of final CRC result
- 32-bit CPU register programming interface

32.3 Functional description

32.3.1 Modes of operation

Following sections provide a brief description of various modes which affect the functionality of the CRC module.

32.3.1.1 Run mode

This is the basic mode of operation.

32.3.1.2 Low-power modes

Any CRC calculation in progress stops when the device enters a low-power mode that disables the module clock. It resumes after the clock is enabled or via the system reset for exiting the low-power mode. Clock gating for this module is dependent on the device.

32.3.2 CRC calculations

In 16-bit and 32-bit CRC modes, data values can be programmed 8 bits, 16 bits, or 32 bits at a time, provided all bytes are contiguous. Noncontiguous bytes can lead to an incorrect CRC computation.

32.3.2.1 16-bit CRC

The steps to compute a 16-bit CRC are listed as follows.

1. Clear CRC_CTRL[TCRC] to enable 16-bit CRC mode.
2. Program the transpose and complement options in the CTRL register as required for the CRC calculation. See Transpose feature and CRC result complement for details.
3. Write a 16-bit polynomial to the CRC_GPOLY[LOW] field. The CRC_GPOLY[HIGH] field is not usable in 16-bit CRC mode.
4. Set CRC_CTRL[WAS] to program the seed value.
5. Write a 16-bit seed to CRC_DATA[LU:LL]. CRC_DATA[HU:HL] are not used.
6. Clear CRC_CTRL[WAS] to start writing data values.
7. Write data values into CRC_DATA[HU:HL:LU:LL]. A CRC is computed on every data value write, and the intermediate CRC result is stored back into CRC_DATA[LU:LL].
8. When all values have been written, read the final CRC result from CRC_DATA[LU:LL].

Transpose and complement operations are performed on-the-fly while reading or writing values. See Transpose feature and CRC result complement for details.

32.3.2.2 32-bit CRC

The steps to compute a 32-bit CRC are listed as follows.

1. Set CRC_CTRL[TCRC] to enable 32-bit CRC mode.

2. Program the transpose and complement options in the CTRL register as required for the CRC calculation. See Transpose feature and CRC result complement for details.
3. Write a 32-bit polynomial to CRC_GPOLY[HIGH:LOW].
4. Set CRC_CTRL[WAS] to program the seed value.
5. Write a 32-bit seed to CRC_DATA[HU:HL:LU:LL].
6. Clear CRC_CTRL[WAS] to start writing data values.
7. Write data values into CRC_DATA[HU:HL:LU:LL]. A CRC is computed on every data value write, and the intermediate CRC result is stored back into CRC_DATA[HU:HL:LU:LL].
8. When all values have been written, read the final CRC result from CRC_DATA[HU:HL:LU:LL]. The CRC is calculated bytewise, and two clocks are required to complete one CRC calculation.

Transpose and complement operations are performed on-the-fly while reading or writing values. See Transpose feature and CRC result complement for details.

32.3.3 Transpose feature

By default, the transpose feature is not enabled. However, some CRC standards require the input data and/or the final checksum to be transposed. The user software has the option to configure each transpose operation separately, as desired by the CRC standard. The data is transposed on-the-fly while being read or written.

Some protocols use little-endian format for the data stream to calculate a CRC. In this case, the transpose feature usefully flips the bits. This transpose option is one of the types supported by the CRC module.

32.3.3.1 Types of transpose

The CRC module provides several types of transpose functions to flip the bits and/or bytes, for both writing input data and reading the CRC result, separately using the CTRL[TOT] or CTRL[TOTR] fields, according to the CRC calculation being used.

The following types of transpose functions are available for writing to and reading from the CRC Data register.

1. CTRL[TOT] or CTRL[TOTR] is 0b00.

   No transposition occurs.

2. CTRL[TOT] or CTRL[TOTR] is 0b01

   Bits in a byte are transposed, while bytes are not transposed.

   reg[31:0] becomes {reg[24:31], reg[16:23], reg[8:15], reg[0:7]}. See the following figure.

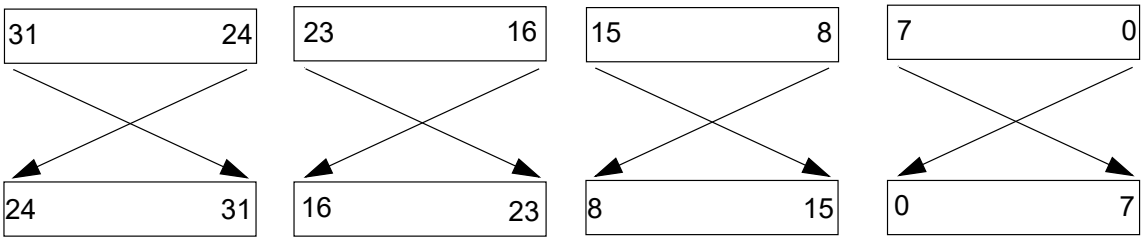


Figure 131. Transpose type 0b01

3. CTRL[TOT] or CTRL[TOTR] is 0b10.

   Both bits in bytes and bytes are transposed.

   reg[31:0] becomes = {reg[0:7], reg[8:15],reg[16:23], reg[24:31]}. See the following figure.

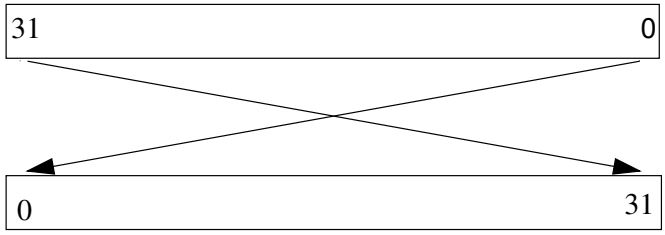


Figure 132. Transpose type 0b10

4. CTRL[TOT] or CTRL[TOTR] is 0b11.

   Bytes are transposed, but bits are not transposed.

   reg[31:0] becomes {reg[7:0], reg[15:8], reg[23:16], reg[31:24]}. See the following figure.

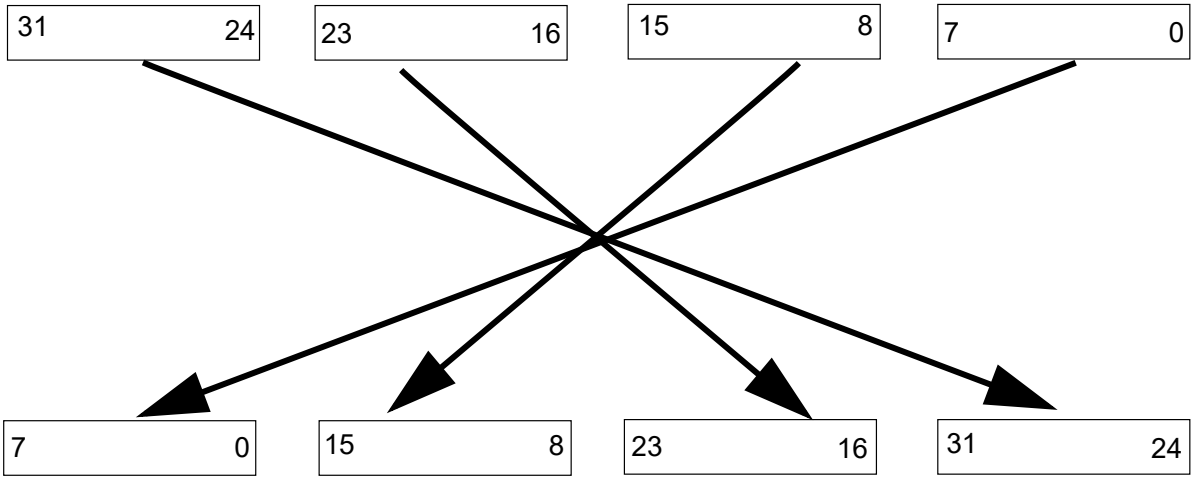


Figure 133. Transpose type 0b11

NOTE

For 8-bit and 16-bit write accesses to the CRC data register, the data is transposed with zeros on the unused byte or bytes (taking 32 bits as a whole), but the CRC is calculated on the valid byte(s) only. When reading the CRC data register for a 16-bit CRC result and using transpose options 10 and 11, the resulting value after transposition resides in the CRC[HU:HL] fields. The user software must account for this situation when reading the 16-bit CRC result, so reading 32 bits is preferred.

32.3.4 CRC result complement

When CTRL[FXOR] is set, the checksum is complemented. The CRC result complement function outputs the complement of the checksum value stored in the CRC Data (DATA) Register every time the CRC Data (DATA) register is read. When CTRL[FXOR] is cleared, reading the CRC Data (DATA) Register accesses the raw checksum value.

32.4 Use cases

NOTE

The following tables apply to little-endian format.

32.4.1 CRC Control Register programming

The following table shows the CRC Control Register programming for 16-bit CRC.

Table 306. Control Register (CTRL) programming for 16-bit CRC

Algorithm	Poly	Seed	Ref In	Ref Out	XOR Out	CTRL[TOT]	CTRL[TOTR]	CTRL[FXOR]
CRC-16/ CCITT_FALSE	0x1021	0xFFFF	0	0	0x0000	0x0	0x0	0x0
CRC-16 / ARC	0x8005	0x0000	1	1	0x0000	0x1	0x2	0x0
CRC16_AUG_CCITT	0x1021	0x1D0F	0	0	0x0000	0x0	0x0	0x0
CRC16_BUYPASS	0x8005	0x0000	0	0	0x0000	0x0	0x0	0x0
CRC16_CCITT_ZERO	0x1021	0x0000	0	0	0x0000	0x0	0x0	0x0
CRC16_CDMA2000	0xC867	0xFFFF	0	0	0x0000	0x0	0x0	0x0
CRC16_DDS_110	0x8005	0x800D	0	0	0x0000	0x0	0x0	0x0
CRC16_DECT_R	0x589	0x0000	0	0	0xFFFF	0x1	0x2	0x1
CRC16_DECT_X	0x589	0x0000	0	0	0x0000	0x0	0x0	0x0
CRC16_DNP	0x3D65	0x0000	1	1	0xFFFF	0x1	0x2	0x1
CRC16_EN_13757	0x3D65	0x0000	0	0	0xFFFF	0x1	0x2	0x1
CRC16_GENIBUS	0x1021	0xFFFF	0	0	0xFFFF	0x1	0x2	0x1
CRC16_MAXIM	0x8005	0x0000	1	1	0xFFFF	0x1	0x2	0x1
CRC16_MCRF4XX	0x1021	0xFFFF	1	1	0x0000	0x1	0x2	0x0
CRC16_RIELLO	0x1021	0xB2AA	1	1	0x0000	0x1	0x2	0x0
CRC16_T10_DIF	0x8BB7	0x0000	0	0	0x0000	0x0	0x0	0x0
CRC16_TELEDISK	0xA097	0x0000	0	0	0x0000	0x0	0x0	0x0
CRC16_TMS37157	0x1021	0x89EC	1	1	0x0000	0x1	0x2	0x0
CRC16_USB	0x8005	0xFFFF	1	1	0xFFFF	0x1	0x2	0x1
CRC16_A	0x1021	0xC6C6	1	1	0x0000	0x1	0x2	0x0
CRC16_KERMIT	0x1021	0x0000	1	1	0x0000	0x1	0x2	0x0
CRC16_MODBUS	0x8005	0xFFFF	1	1	0x0000	0x1	0x2	0x0
CRC16_X_25	0x1021	0xFFFF	1	1	0xFFFF	0x1	0x2	0x1
CRC16_XMODEM	0x1021	0x0000	0	0	0x0000	0x0	0x0	0x0

The following table shows the CRC Control Register programming for 32-bit CRC.

Table 307. Control Register (CTRL) programming for 32-bit CRC

Algorithm	Poly	Seed	Ref In	Ref Out	Xor Out	CTRL[TOT]	CTRL[TOTR]	CTRL[FXOR]
CRC - 32	0x04C11DB7	0xFFFFFFFF	1	1	0xFFFF_FFFF	0x1	0x2	0x1
CRC-32/BZIP2	0x04C11DB7	0xFFFFFFFF	0	0	0xFFFF_FFFF	0x0	0x0	0x1
CRC-32C	0x1EDC6F41	0xFFFFFFFF	1	1	0xFFFF_FFFF	0x1	0x2	0x1
CRC-32D	0xA833982B	0xFFFFFFFF	1	1	0xFFFF_FFFF	0x1	0x2	0x1
CRC-32/MPEG-2	0x04C11DB7	0xFFFFFFFF	0	0	0x0000_0000	0x0	0x0	0x0
CRC-32/POSIX	0x04C11DB7	0x00000000	0	0	0xFFFF_FFFF	0x0	0x0	0x1
CRC-32Q	0x814141AB	0x00000000	0	0	0x0000_0000	0x0	0x0	0x0
CRC-32/JAMCRC	0x04C11DB7	0xFFFFFFFF	1	1	0x0000_0000	0x1	0x2	0x0
CRC-32/XFER	0x000000AF	0x00000000	0	0	0x0000_0000	0x0	0x0	0x0

32.4.2 Expected read data fields

The following table shows the expected read data fields for 16-bit CRC.

Table 308. Expected read data fields for 16-bit CRC

Algorithm	CRC DATA Register (DATA)
CRC-16/CCITT_FALSE	[31 : 16] = Unknown [15 : 0] = Valid Data
CRC-16 / 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_R	[31 : 16] = Valid Data [15 : 0] = Unknown
CRC16_DECT_X	[31 : 16] = Unknown [15 : 0] = Valid Data
CRC16_DNP	[31 : 16] = Valid Data [15 : 0] = Unknown
CRC16_EN_13757	[31 : 16] = Valid Data [15 : 0] = Unknown
CRC16_GENIBUS	[31 : 16] = Valid Data [15 : 0] = Unknown
CRC16_MAXIM	[31 : 16] = Valid Data [15 : 0] = Unknown

Table continues on the next page...

Table 308. Expected read data fields for 16-bit CRC (continued)

Algorithm	CRC DATA Register (DATA)
CRC16_MCRF4XX	[31 : 16] = Valid Data [15 : 0] = Unknown
CRC16_RIELLO	[31 : 16] = Valid Data [15 : 0] = Unknown
CRC16_T10_DIF	[31 : 16] = Unknown [15 : 0] = Valid Data
CRC16_TELEDISK	[31 : 16] = Unknown [15 : 0] = Valid Data
CRC16_TMS37157	[31 : 16] = Valid Data [15 : 0] = Unknown
CRC16_USB	[31 : 16] = Valid Data [15 : 0] = Unknown
CRC16_A	[31 : 16] = Valid Data [15 : 0] = Unknown
CRC16_KERMIT	[31 : 16] = Valid Data [15 : 0] = Unknown
CRC16_MODBUS	[31 : 16] = Valid Data [15 : 0] = Unknown
CRC16_X_25	[31 : 16] = Valid Data [15 : 0] = Unknown
CRC16_XMODEM	[31 : 16] = Unknown [15 : 0] = Valid Data

The following table shows the expected read data fields for 32-bit CRC.

Table 309. Expected read data fields for 32-bit CRC

Algorithm	CRC DATA Register (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

32.5 External signals

There is no CRC signal that connects off chip.

32.6 CRC initialization/reinitialization

To enable the CRC calculation, the user must program CRC_CTRL[WAS], CRC_GPOLY, necessary parameters for transposition and CRC result inversion in the applicable registers. Asserting CRC_CTRL[WAS] enables the programming of the seed value into the CRC_DATA register.

After a completed CRC calculation, the module can be reinitialized for a new CRC computation by reasserting CRC_CTRL[WAS] and programming a new, or previously used, seed value. All other parameters must be set before programming the seed value and subsequent data values.

32.7 Memory map and register descriptions

NOTE

Write access to the register addresses not mapped to the peripheral but included in the address space of the peripheral may result in unpredictable functionality. The CRC module should be reconfigured if a transfer error occurs.

32.7.1 CRC register descriptions

32.7.1.1 CRC memory map

CRC base address: 4009_5000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	CRC DATA register (DATA)	32	RW	FFFF_FFFF
4	CRC Polynomial register (GPOLY)	32	RW	0000_1021
8	CRC Control register (CTRL)	32	See section	0000_0000

32.7.1.1.1 CRC DATA register (DATA)

The CRC DATA register contains the value of the seed, data, and checksum. When CTRL[WAS] is set, any write to the data register is regarded as the seed value. When CTRL[WAS] is cleared, any write to the data register is regarded as data for general CRC computation.

In 16-bit CRC mode, the HU and HL fields 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.

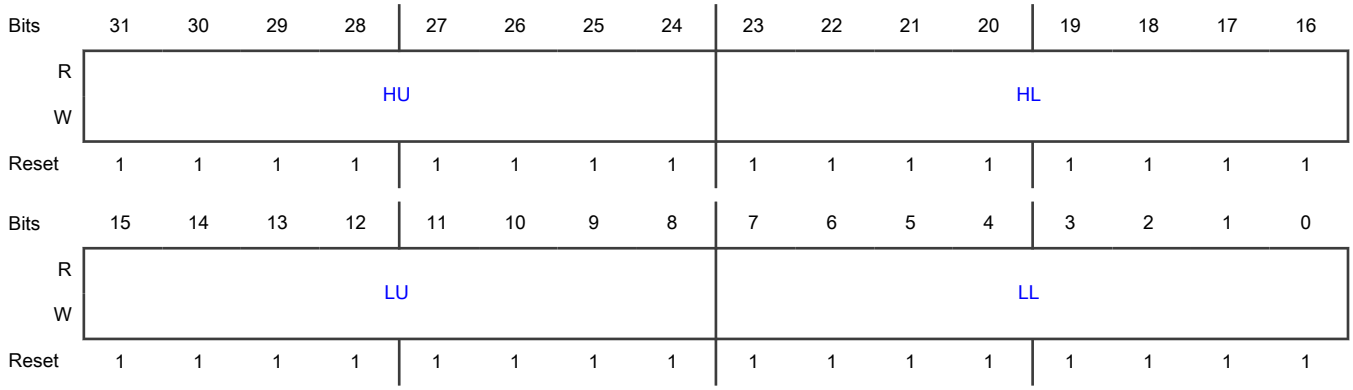
When programming data values, the values can be written 8 bits, 16 bits, or 32 bits at a time, provided all bytes are contiguous; with MSB of data value written first.

After all data values are written, the CRC result can be read from this data register. In 16-bit CRC mode, the CRC result is available in the LU and LL fields. In 32-bit CRC mode, all fields contain the result. Reads of this register at any time return the intermediate CRC value, provided the CRC module is configured.

Offset

Register	Offset
DATA	0h

Diagram



Fields

Field	Description
31-24 HU	CRC High Upper Byte In 16-bit CRC mode (CTRL[TCRC] is 0), this field is not used for programming a seed value. In 32-bit CRC mode (CTRL[TCRC] is 1), values written to this field are part of the seed value when CTRL[WAS] is 1. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation in both 16-bit and 32-bit CRC modes.
23-16 HL	CRC High Lower Byte In 16-bit CRC mode (CTRL[TCRC] is 0), this field is not used for programming a seed value. In 32-bit CRC mode (CTRL[TCRC] is 1), values written to this field are part of the seed value when CTRL[WAS] is 1. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation in both 16-bit and 32-bit CRC modes.
15-8 LU	CRC Low Upper Byte When CTRL[WAS] is 1, values written to this field are part of the seed value. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation.
7-0 LL	CRC Low Lower Byte When CTRL[WAS] is 1, values written to this field are part of the seed value. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation.

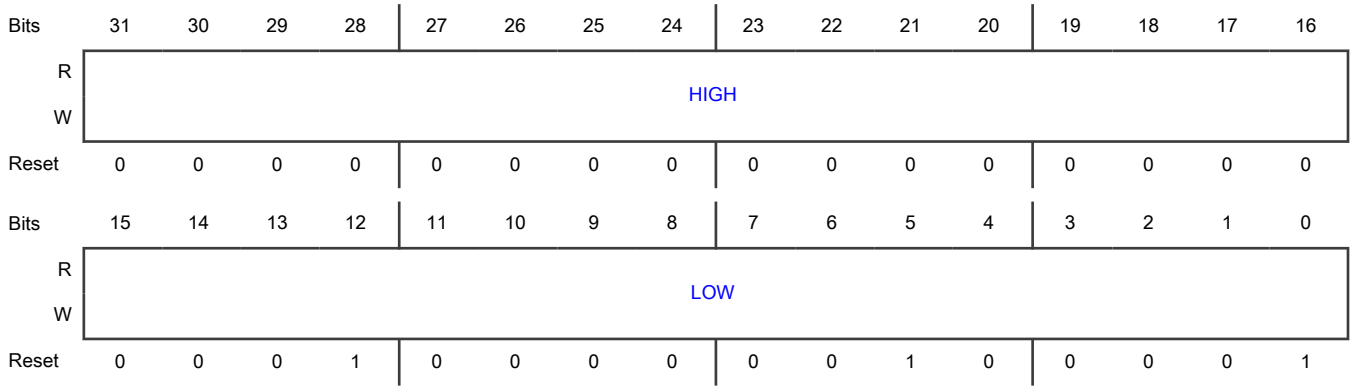
32.7.1.1.2 CRC Polynomial register (GPOLY)

This register contains the value of the polynomial for the CRC calculation. The HIGH field contains the upper 16 bits of the CRC polynomial, which are used only in 32-bit CRC mode. Writes to the HIGH field are ignored in 16-bit CRC mode. The LOW field contains the lower 16 bits of the CRC polynomial, which are used in both 16- and 32-bit CRC modes.

Offset

Register	Offset
GPOLY	4h

Diagram



Fields

Field	Description
31-16 HIGH	High Polynominal Half-word Writable and readable in 32-bit CRC mode (CTRL[TCRC] is 1). This field is not writable in 16-bit CRC mode (CTRL[TCRC] is 0).
15-0 LOW	Low Polynominal Half-word Writable and readable in both 32-bit and 16-bit CRC modes.

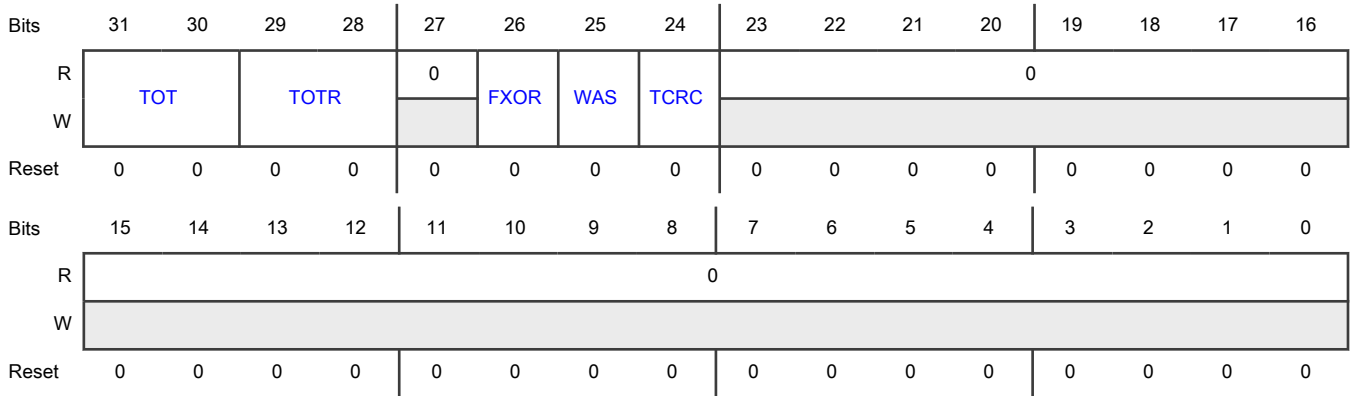
32.7.1.1.3 CRC Control register (CTRL)

This register controls the configuration and working of the CRC module. Appropriate bits must be set before starting a new CRC calculation. A new CRC calculation is initialized by asserting CTRL[WAS] and then writing the seed into the CRC DATA register.

Offset

Register	Offset
CTRL	8h

Diagram



Fields

Field	Description
31-30 TOT	Type Of Transpose For Writes Defines the transpose configuration of the data written to the CRC DATA register. See Transpose feature for the available transpose options. 00 - No transposition. 01 - Bits in bytes are transposed; bytes are not transposed. 10 - Both bits in bytes and bytes are transposed. 11 - Only bytes are transposed; no bits in a byte are transposed.
29-28 TOTR	Type Of Transpose For Read Identifies the transpose configuration of the value read from the CRC DATA register. See Transpose feature for the available transpose options. 00 - No transposition. 01 - Bits in bytes are transposed; bytes are not transposed. 10 - Both bits in bytes and bytes are transposed. 11 - Only bytes are transposed; no bits in a byte are transposed.
27 —	Reserved
26 FXOR	Complement Read Of CRC DATA register Some CRC protocols require the final checksum to be XORed with 0xFFFFFFFF or 0xFFFF. Asserting this bit enables on-the-fly complementing of read data. 0 - No XOR on reading. 1 - Invert or complement the read value of the CRC DATA register.
25 WAS	Write CRC DATA register As Seed When asserted, a value written to the CRC DATA register is considered a seed value. When deasserted, a value written to the CRC DATA register is taken as data for CRC computation. 0 - Writes to the CRC DATA register are data values. 1 - Writes to the CRC DATA register are seed values.
24 TCRC	TCRC Width of CRC protocol. 0 - 16-bit CRC protocol. 1 - 32-bit CRC protocol.
23-0 —	Reserved

Chapter 33
Standard counter/timers (CTIMER)

33.1 Chip-specific CTIMER information

Table 310. Reference links to related information

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

NOTE

The chip does not include timer-specific capture input pins. Instead there are 16 shared ct_inp[0:15] functions, each of which may be brought in from a selection of pins. These 16 inputs go to all 5 CTIMERS via PMUXes (Peripheral input Muxes) in front of each CTIMER. So, user can elect to use any of these inputs for any of the timers. The inputs can be used for capture or clock.

33.1.1 Module instances

This device has five instances of the CTIMER module, CTIMER0, CTIMER1, CTIMER2, CTIMER3, CTIMER4.

33.1.2 Initialization

- Enable the clock to the CTIMER in the AHBCLKCTRL1 and AHBCLKCTRL2 registers (See SYSCON). This enables the register interface and the peripheral function clock.
- Select a clock source for the CTIMER using the appropriate CTIMERCLKSEL register.
- Clear the CTIMER peripheral reset in the PRESETCTRL1/2 register by writing to the PRESETCTRLCLR1/2 register (See SYSCON).
- Each CTIMER provides interrupts to the NVIC. Please refer MCR and CCR in CTIMER register section for match and capture events. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register. For interrupt connections, see Nested Vectored Interrupt Controller (NVIC).
- Select timer pins and pin modes as needed through the relevant IOPCTL registers.
- The CTIMER DMA request lines are connected to the DMA trigger inputs via the DMAC0_ITRIG_SEL registers (See INPUTMUX). Note that timer DMA request outputs are connected to DMA trigger inputs.

33.1.3 Peripheral Input Multiplexers

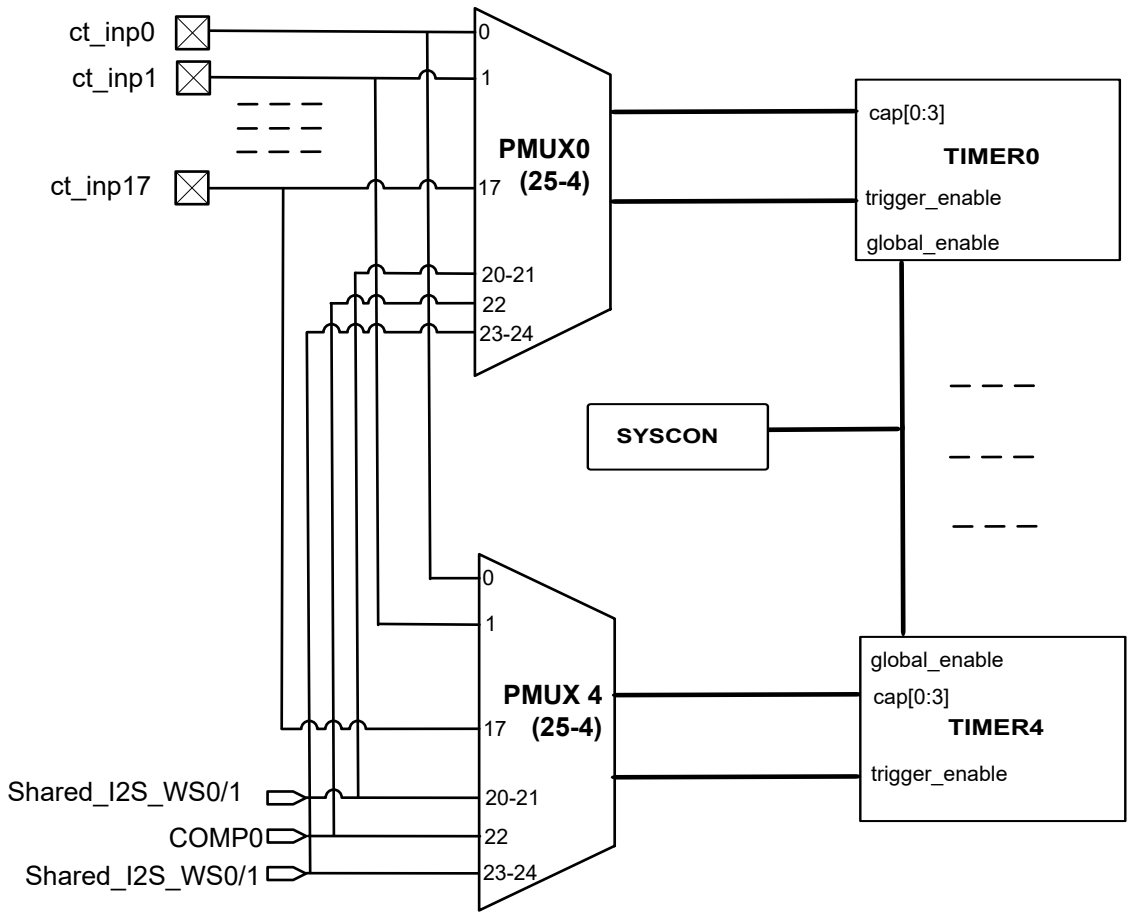


Figure 134. Peripheral Input Multiplexers for CTimers

33.1.4 External global enable

When the timer is enabled by the external global start register, it is equivalent to writing the a 1 to the CEN bit. The external global start register is used to enable multiple timers at the same time. This operation is allowed by writing a 1 to each Timer's AGCEN bit. If the Timer control register's CEN bit is already 1, the external global start enable register has no effect (See SYSCON[CTIMERGLOBALSTARTEN]).

For more information refer CTIMER global start enable register (See SYSCON).

33.2 Overview

Each Counter/timer is designed to count cycles of the CTIMER function clock or an externally supplied clock and can optionally generate interrupts or perform other actions at specified timer values based on four match registers. Each counter/timer also includes capture inputs to capture the timer value when an input signal transitions, optionally generating an interrupt.

In PWM mode, three match registers can be used to provide a single-edge controlled PWM output on the match output pins. One match register is used to control the PWM cycle length. All match registers can optionally be auto-reloaded from a companion shadow register whenever the counter is reset to zero. This permits modifying the match values for the next counter cycle without risk of disrupting the PWM waveforms during the current cycle. When enabled, match reload will occur whenever the counter is reset either due to a match event or a write to bit 1 of the Timer Control Register (TCR).

33.2.1 Features

- Each is a 32-bit counter/timer with a programmable 32-bit prescaler. The timers include external capture and match pin connections.
- Each timer can be enabled by its timer control register or by an external trigger (trigger_enable). One or more timers can also be globally enabled by an external global start enable register.
- Counter or timer operation.
- Each CTIMER has a selection of function clocks that may be asynchronous to other system clocks.
- Up to four 32-bit captures can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt. The number of capture inputs for each timer that are actually available on device pins may vary by device.
- The timer and prescaler may be configured to be cleared on a designated capture event. This feature permits easy pulse-width measurement by clearing the timer on the leading edge of an input pulse and capturing the timer value on the trailing edge.
- Four 32-bit match registers that allow:
  - Continuous operation with optional interrupt generation on match.
  - Optional auto-reload from match shadow registers when counter is reset.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- For each timer, up to 4 external outputs corresponding to match registers with the following capabilities (the number of match outputs for each timer that are actually available on device pins may vary by device):
  - Set LOW on match.
  - Set HIGH on match.
  - Toggle on match.
  - Do nothing on match.
- Up to 4 match registers can be configured for PWM operation, allowing up to 3 single edged controlled PWM outputs. (The number of match outputs for each timer that are actually available on device pins may vary by device.)
- Up to 2 match registers can be used to generate DMA requests.

33.2.1.1 Block Diagram

The block diagram for the timers is shown below.

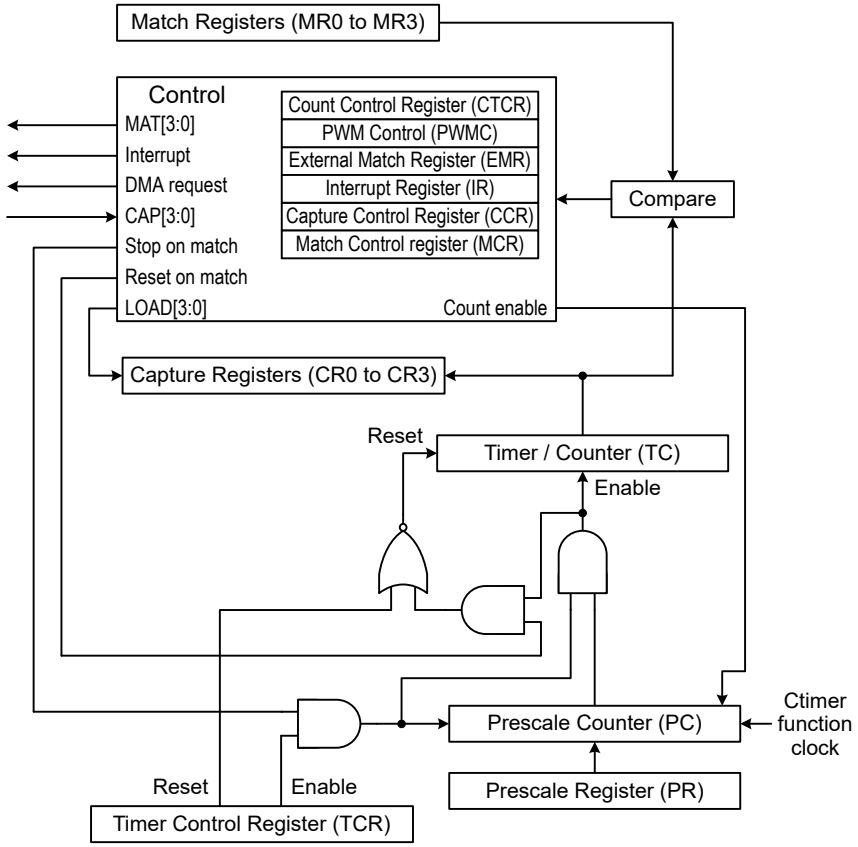


Figure 135. 32-bit counter/timer block diagram

33.3 Functional description

The figure below shows a timer configured to reset the count and generate an interrupt on match. The prescaler is set to 2 and the match register set to 6. At the end of the timer cycle where the match occurs, the timer count is reset. This gives a full length cycle to the match value. The interrupt indicating that a match occurred is generated in the next clock after the timer reached the match value.

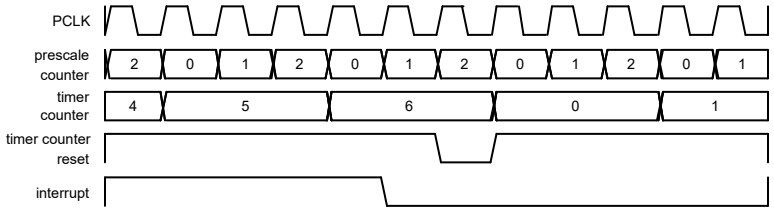


Figure 136. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled

The figure below shows a timer configured to stop and generate an interrupt on match. The prescaler is again set to 2 and the match register set to 6. In the next clock after the timer reaches the match value, the timer enable bit in TCR is cleared, and the interrupt indicating that a match occurred is generated.

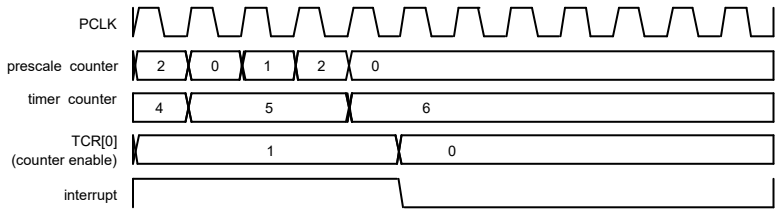


Figure 137. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled

33.3.1 Using the CTIMER to generate single edge controlled PWM outputs

1. All single edge controlled PWM outputs go LOW at the beginning of each PWM cycle (timer is set to zero) unless their match value is equal to zero.
2. Each PWM output will go HIGH when its match value is reached. If no match occurs (i.e. the match value is greater than the PWM cycle length), the PWM output remains continuously LOW.
3. If a match value larger than the PWM cycle length is written to the match register, and the PWM signal is HIGH already, then the PWM signal will be cleared with the start of the next PWM cycle.
4. If a match register contains the same value as the timer reset value (the PWM cycle length), then the PWM output will be reset to LOW on the next clock tick after the timer reaches the match value. Therefore, the PWM output will always consist of a one clock tick wide positive pulse with a period determined by the PWM cycle length (i.e. the timer reload value).
5. If a match register is set to zero, then the PWM output will go to HIGH the first time the timer goes back to zero and will stay HIGH continuously.

NOTE

When the match outputs are selected to perform as PWM outputs, the timer reset (MRnR) and timer stop (MRnS) bits in the Match Control Register MCR must be set to zero except for the match register setting the PWM cycle length. For this register, set the MRnR bit to one to enable the timer reset when the timer value matches the value of the corresponding match register.

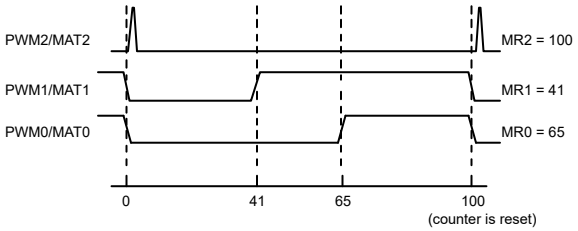


Figure 138. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.

33.3.2 DMA operation

DMA requests are generated by a match of the Timer Counter (TC) register value to either Match Register 0 (MR0) or Match Register 1 (MR1). This is not connected to the operation of the Match outputs controlled by the EMR register. Each match sets a DMA request flag, which is connected to the DMA controller. In order to have an effect, the DMA controller must be configured correctly.

When a timer is initially set up to generate a DMA request, the request may already be asserted before a match condition occurs. An initial DMA request may be avoided by having software write a one to the interrupt flag location, as if clearing a timer interrupt. See Interrupt Register (IR). A DMA request will be cleared automatically when it is handled by the DMA controller.

NOTE

Timer DMA requests are generated whenever the timer value is equal to the related Match Register value, DMA requests are always generated when the timer is running, unless the Match Register value is higher than the upper count limit of the timer. It is important not to select and enable timer DMA requests in the DMA block unless the timer is correctly configured to generate valid DMA requests.

33.3.3 External trigger enable

When the external trigger enable is asserted (rising edge), it is equivalent to writing 1 to Timer Control Register (TCR)[CEN] bit. This operation is allowed by writing a 1 to the Timer Control Register (TCR)[TRIGEN] bit. If the Timer control register's CEN bit is already 1, external trigger enable rising edge events have no effect.

33.4 Signals

The tables below give a summary of the Timer/Counter pins.

Table 311. Timer/Counter pin description

Pin	Type	Description
CTIMER0_CAPn CTIMER1_CAPn CTIMER2_CAPn CTIMER3_CAPn CTIMER4_CAPn	Input	Capture Signals- A transition on a capture pin can be configured to load one of the Capture Registers with the value in the Timer Counter and optionally generate an interrupt. Capture functionality can be selected from a number of pins. Timer/Counter block can select a capture signal as a clock source instead of the APB bus clock. For more details see Count Control Register (CTCR).
CTIMER0_MATn CTIMER1_MATn CTIMER2_MATn CTIMER3_MATn CTIMER4_MATn	Output	External Match Output - When a match register (MRn) equals the timer counter (TC) this output can either toggle, go low, go high, or do nothing. The External Match Register (EMR) controls the functionality of this output. Match Output functionality can be selected on a number of pins in parallel.

33.4.1 Multiple Capture (CAP) and Match (MAT) pins

Software can select from multiple pins for the CAP or MAT signals. Note that match conditions may be used internally without the use of a device pin.

33.4.1.1 Capture inputs

The capture signal can be configured to load the Capture Register with the value in the counter/timer and optionally generate an interrupt. The capture signal is generated by one of the pins with a capture function. Each capture signal is connected to one capture channel of the timer.

The Counter/Timer block can select a capture signal as a clock source instead of the APB CTIMER function clock. For more details see Count Control Register (CTCR).

33.4.1.2 Match outputs

When a match register equals the timer counter (TC), the corresponding match output can either toggle, go LOW, go HIGH, or do nothing. The External Match Register (EMR) and PWM Control Register (PWMC) control the functionality of this output.

33.5 Application information

- Interval Timer for counting internal events.
- PWM outputs
- Pulse Width Demodulator via Capture inputs.
- Free running timer.

33.6 Memory map and Register definition

This section includes the CTIMER module memory map and detailed descriptions of all registers.

33.6.1 Counter/Timer register descriptions

33.6.1.1 CTIMER memory map

CTIMER0 base address: 4000_8000h

CTIMER1 base address: 4000_9000h

CTIMER2 base address: 4002_8000h

CTIMER3 base address: 4002_9000h

CTIMER4 base address: 4002_A000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Interrupt Register (IR)	32	See section	See section
4	Timer Control Register (TCR)	32	See section	See section
8	Timer Counter (TC)	32	RW	0000_0000
C	Prescale Register (PR)	32	RW	0000_0000
10	Prescale Counter (PC)	32	RW	0000_0000
14	Match Control Register (MCR)	32	See section	See section
18 - 24	Match Register (MR0 - MR3)	32	RW	0000_0000
28	Capture Control Register (CCR)	32	See section	See section
2C - 38	Capture Register (CR0 - CR3)	32	RO	0000_0000
3C	External Match Register (EMR)	32	See section	See section
70	Count Control Register (CTCR)	32	See section	See section
74	PWM Control Register (PWMC)	32	See section	See section
78 - 84	Match Shadow Register (MSR0 - MSR3)	32	RW	0000_0000

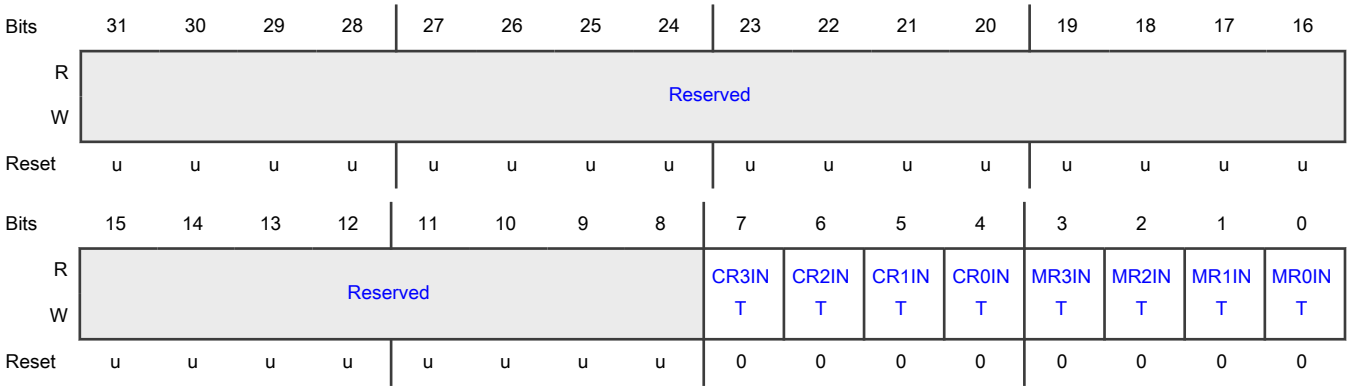
33.6.1.1.1 Interrupt Register (IR)

The Interrupt Register consists of 4 bits for the match interrupts and 4 bits for the capture interrupts. If an interrupt is generated then the corresponding bit in the IR will be high. Otherwise, the bit will be low. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect. The act of clearing an interrupt for a timer match also clears any corresponding DMA request. Writing a zero has no effect. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.

Offset

Register	Offset
IR	0h

Diagram



Fields

Field	Description
31-8 —	Reserved Read value is undefined, only zero should be written
7-4 CRnINT	Interrupt flag for capture channel n event
3-0 MRnINT	Interrupt flag for match channel n

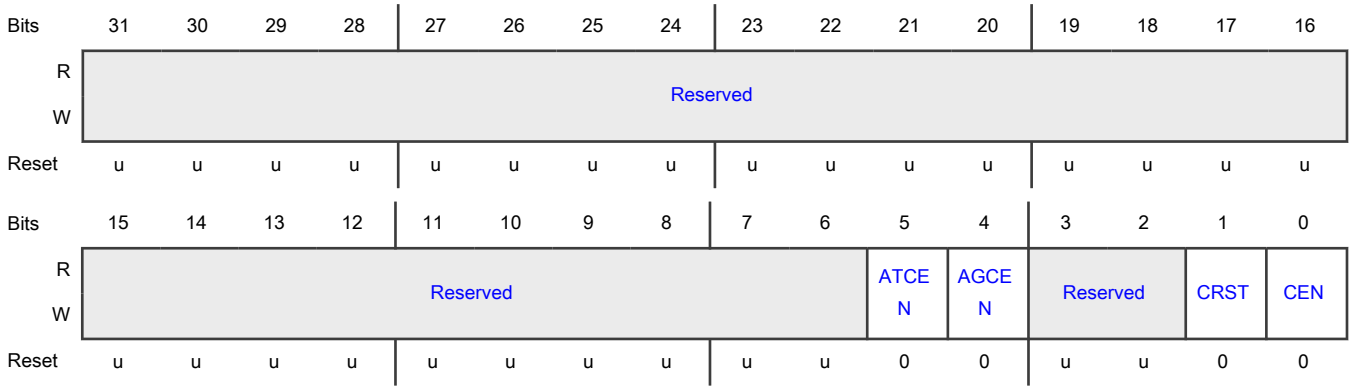
33.6.1.1.2 Timer Control Register (TCR)

The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.

Offset

Register	Offset
TCR	4h

Diagram



Fields

Field	Description
31-6 —	Reserved Read value is undefined, only zero should be written
5 ATCEN	Allow Trigger Count Enable This bit enables and disables input trigger. 0 - Not allowed 1 - Allow input trigger_enable=1 action to take effect
4 AGCEN	Allow Global Count Enable This bit enables and disables global enable. 0 - Not allowed 1 - Allow input global_enable=1 action to take effect
3-2 —	Reserved
1 CRST	Counter reset This bit enables and disables counter reset. 0 - Disabled. Do nothing 1 - Enabled. The Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of the CTIMER function clock. The counters remain reset until TCR[1] is returned to zero.
0 CEN	Counter enable This bit enables and disables counters. 0 - Disabled. The counters are disabled 1 - Enabled. The Timer Counter and Prescale Counter are enabled. When the timer is enabled by an external trigger or globally enabled by the external global start enable register, the CEN bit will automatically be set to 1

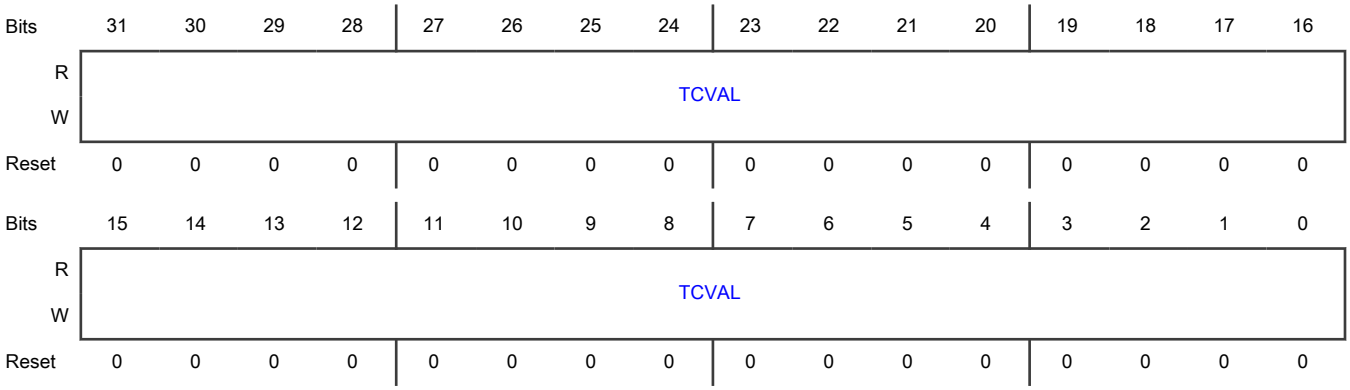
33.6.1.1.3 Timer Counter (TC)

The 32-bit Timer Counter register is incremented when the prescale counter reaches its terminal count. Unless it is reset before reaching its upper limit, the Timer Counter will count up through the value 0xFFFF FFFF and then wrap back to the value 0x0000 0000. This event does not cause an interrupt, but a match register can be used to detect an overflow if needed. The 32 bit TC is incremented every PR+1 cycles of the CTIMER function clock. The TC is controlled through the TCR.

Offset

Register	Offset
TC	8h

Diagram



Fields

Field	Description
31-0 TCVAL	Timer counter value

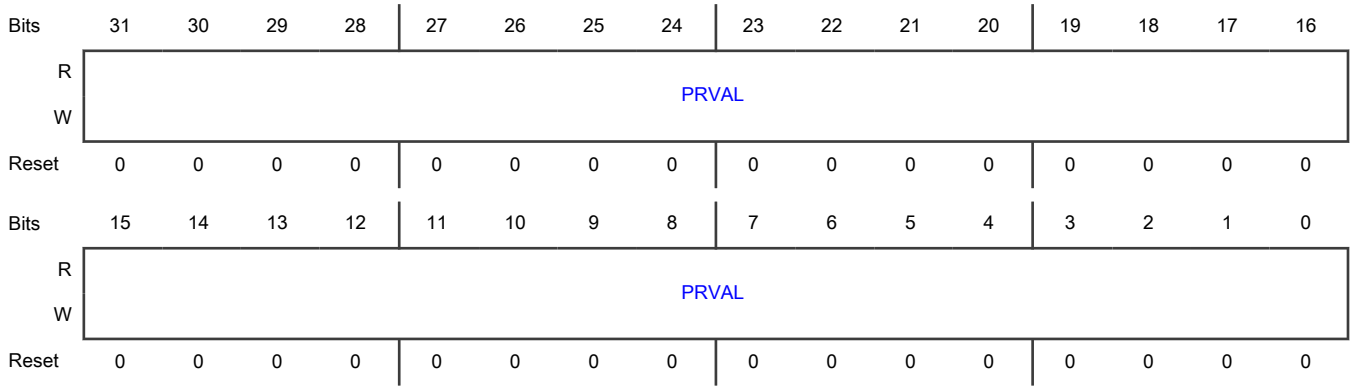
33.6.1.1.4 Prescale Register (PR)

When the Prescale Counter (PC) is equal to this value, the next clock increments the TC and clears the PC.

Offset

Register	Offset
PR	Ch

Diagram



Fields

Field	Description
31-0 PRVAL	Prescale reload value

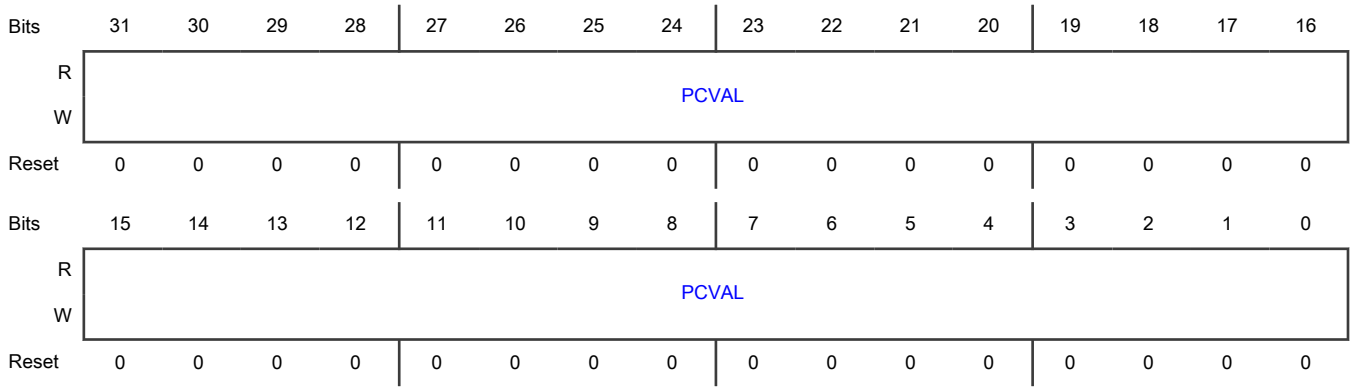
33.6.1.1.5 Prescale Counter (PC)

The 32-bit Prescale Counter (PC) controls division of the CTIMER function clock by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every CTIMER function clock. When it reaches the value stored in the Prescale register, the Timer Counter is incremented and the Prescale Counter is reset on the next CTIMER function clock. This causes the Timer Counter to increment on every CTIMER function clock when PR = 0, every 2 CTIMER function clocks when PR = 1, and so on. The 32 bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.

Offset

Register	Offset
PC	10h

Diagram



Fields

Field	Description
31-0 PCVAL	Prescale counter value

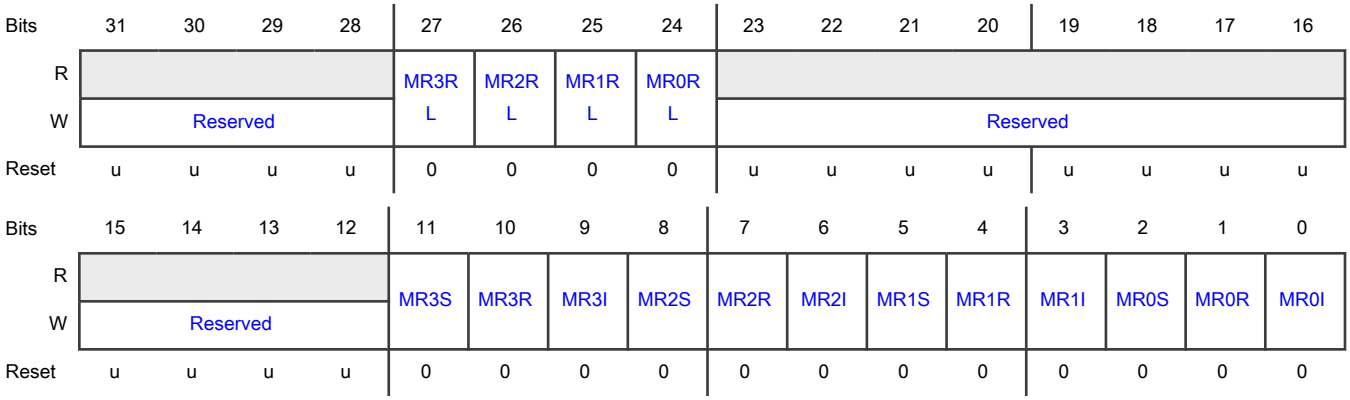
33.6.1.1.6 Match Control Register (MCR)

The Match Control Register is used to control what operations are performed when one of the Match Registers matches the Timer Counter (TC).

Offset

Register	Offset
MCR	14h

Diagram



Fields

Field	Description
31-28 —	Reserved Read value is undefined, only zero should be written.
27-24 MRnRL	Reload MRn Reload with the contents of the Match 0 Shadow Register when the TC is reset to zero (either via a match event or a write to bit 1 of the TCR). 0 = disabled. 1 = enabled. 0 - Disabled 1 - Enabled
23-12 —	Reserved Read value is undefined, only zero should be written

Table continues on the next page...

Table continued from the previous page...

Field	Description
11 MR3S	Stop on MR3 TC and PC will be stopped and TCR[0] will be set to 0 if MR3 matches the TC. 0 - Disabled 1 - Enabled
10 MR3R	Reset on MR3 TC will be reset if MR3 matches it. 0 - Disabled 1 - Enabled
9 MR3I	Interrupt on MR3 An interrupt is generated when MR3 matches the value in the TC. 0 - Disabled 1 - Enabled
8 MR2S	Stop on MR2 TC and PC will be stopped and TCR[0] will be set to 0 if MR2 matches the TC. 0 - Disabled 1 - Enabled
7 MR2R	Reset on MR2 TC will be reset if MR2 matches it. 0 - Disabled 1 - Enabled
6 MR2I	Interrupt on MR2 An interrupt is generated when MR2 matches the value in the TC. 0 - Disabled 1 - Enabled
5 MR1S	Stop on MR1 TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC. 0 - Disabled 1 - Enabled
4 MR1R	Reset on MR1 TC will be reset if MR1 matches it. 0 - Disabled

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Enabled
3 MR1I	Interrupt on MR1 An interrupt is generated when MR1 matches the value in the TC. 0 - Disabled 1 - Enabled
2 MR0S	Stop on MR0 TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC. 0 - Disabled 1 - Enabled
1 MR0R	Reset on MR0 TC will be reset if MR0 matches it. 0 - Disabled 1 - Enabled
0 MR0I	Interrupt on MR0 An interrupt is generated when MR0 matches the value in the TC. 0 - Disabled 1 - Enabled

33.6.1.1.7 Match Register (MR0 - MR3)

The Match register values are continuously compared to the Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the Timer Counter, or stop the timer. Actions are controlled by the settings in the MCR register.

If the associated MRxRL bit in the Match Control Register is set, the Match Register will be automatically reloaded with the current contents of its corresponding Match Shadow register whenever the TC is cleared to zero. This transfer will take place on the same clock edge that advances the TC to zero.

NOTE

The TC is typically reset in response to an occurrence of a match on the Match Register being used to set the cycle counter rate. A reset can also occur due to software writing a 1 to bit 1 of the Timer Control Register.

Offset

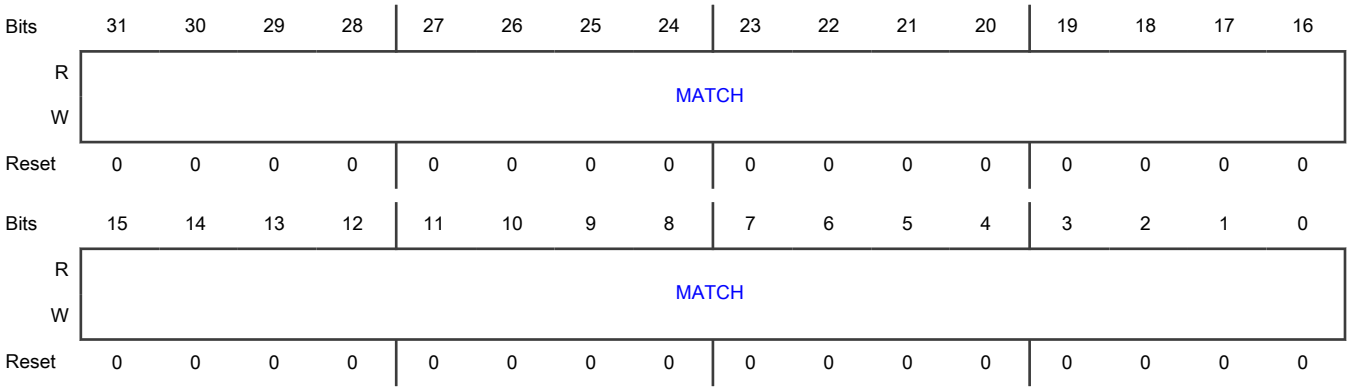
Register	Offset
MR0	18h
MR1	1Ch

Table continues on the next page...

Table continued from the previous page...

Register	Offset
MR2	20h
MR3	24h

Diagram



Fields

Field	Description
31-0 MATCH	Timer counter match value When the timer counter match value is same as timer counter value actions are triggered automatically.

33.6.1.1.8 Capture Control Register (CCR)

The Capture Control Register is used to control whether one of the four Capture Registers is loaded with the value in the Timer Counter when the capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges. In the description below, "n" represents the timer number, 0 or 1.

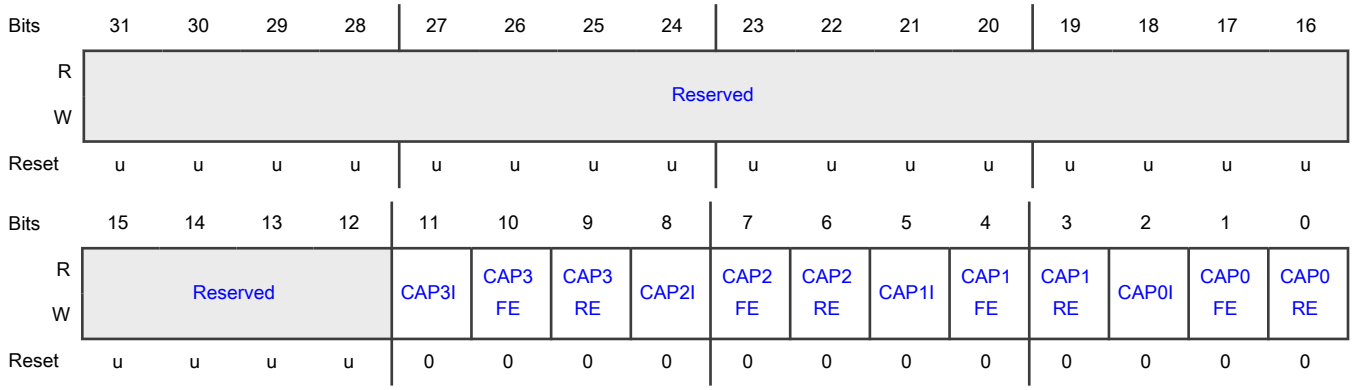
NOTE

If Counter mode is selected for a particular CAP input in the CTCR, the 3 bits for that input in this register should be programmed as 000, but capture and/or interrupt can be selected for the other 3 CAP inputs.

Offset

Register	Offset
CCR	28h

Diagram



Fields

Field	Description
31-12 —	Reserved Read value is undefined, only zero should be written
11 CAP3I	Generate interrupt on channel 3 capture event: a CR3 load generates an interrupt 0 - Disabled 1 - Enabled
10 CAP3FE	Falling edge of capture channel 3: a sequence of 1 then 0 causes CR3 to be loaded with the contents of TC 0 - Disabled 1 - Enabled
9 CAP3RE	Rising edge of capture channel 3: a sequence of 0 then 1 causes CR3 to be loaded with the contents of TC 0 - Disabled 1 - Enabled
8 CAP2I	Generate interrupt on channel 2 capture event: a CR2 load generates an interrupt 0 - Disabled 1 - Enabled
7 CAP2FE	Falling edge of capture channel 2: a sequence of 1 then 0 causes CR2 to be loaded with the contents of TC 0 - Disabled 1 - Enabled
6 CAP2RE	Rising edge of capture channel 2: a sequence of 0 then 1 causes CR2 to be loaded with the contents of TC 0 - Disabled

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Enabled
5 CAP1I	Generate interrupt on channel 1 capture event: a CR1 load generates an interrupt 0 - Disabled 1 - Enabled
4 CAP1FE	Falling edge of capture channel 1: a sequence of 1 then 0 causes CR1 to be loaded with the contents of TC 0 - Disabled 1 - Enabled
3 CAP1RE	Rising edge of capture channel 1: a sequence of 0 then 1 causes CR1 to be loaded with the contents of TC 0 - Disabled 1 - Enabled
2 CAP0I	Generate interrupt on channel 0 capture event: a CR0 load generates an interrupt 0 - Disabled 1 - Enabled
1 CAP0FE	Falling edge of capture channel 0: a sequence of 1 then 0 causes CR0 to be loaded with the contents of TC 0 - Disabled 1 - Enabled
0 CAP0RE	Rising edge of capture channel 0: a sequence of 0 then 1 causes CR0 to be loaded with the contents of TC 0 - Disabled 1 - Enabled

33.6.1.1.9 Capture Register (CR0 - CR3)

Each Capture register is associated with one capture channel and may be loaded with the counter/timer value when a specified event occurs on the signal defined for that capture channel. The signal could originate from an external pin or from an internal source. The settings in the Capture Control Register register determine whether the capture function is enabled, and whether a capture event happens on the rising edge of the associated signal, the falling edge, or on both edges. The register is loaded with the value of TC when there is an event on the CAPn. input.

Offset

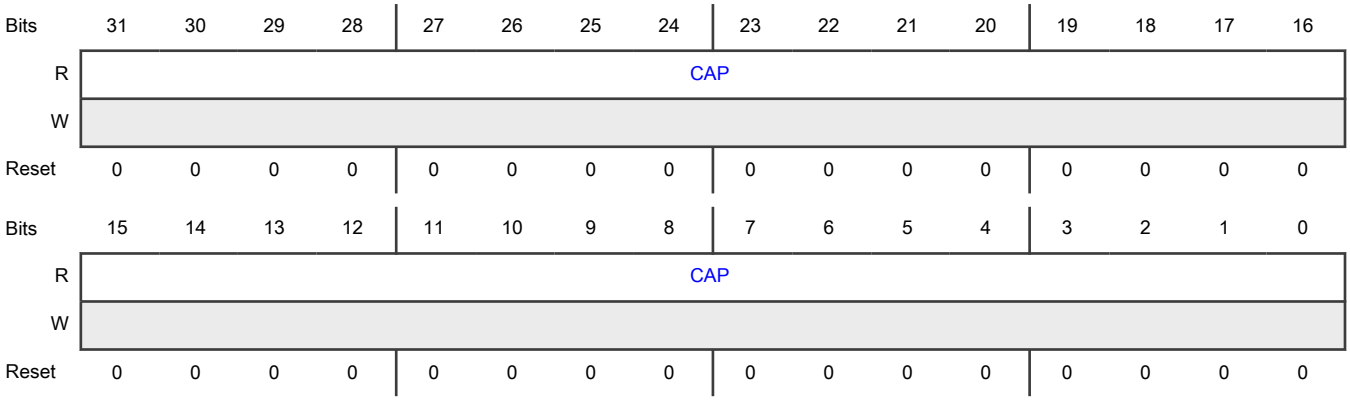
Register	Offset
CR0	2Ch

Table continues on the next page...

Table continued from the previous page...

Register	Offset
CR1	30h
CR2	34h
CR3	38h

Diagram



Fields

Field	Description
31-0 CAP	Timer counter capture value The value of timer counter capture determines if the capture function is enabled and the edge on which the capture function happens.

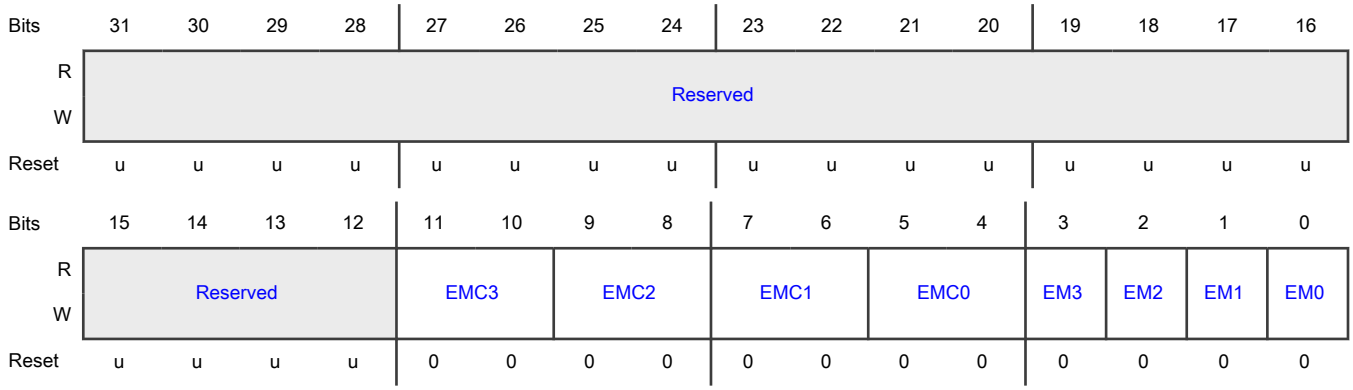
33.6.1.1.10 External Match Register (EMR)

The External Match Register provides both control and status of the external match pins. If the match outputs are configured as PWM output, the function of the external match registers is determined by the PWM rules.

Offset

Register	Offset
EMR	3Ch

Diagram



Fields

Field	Description
31-12 —	Reserved Read value is undefined, only zero should be written
11-10 EMC3	External Match Control 3 Determines the functionality of External Match 3. 00 - Do Nothing 01 - Clear. Clear the corresponding External Match bit/output to 0 (MAT3 pin is LOW if pinned out) 10 - Set. Set the corresponding External Match bit/output to 1 (MAT3 pin is HIGH if pinned out) 11 - Toggle. Toggle the corresponding External Match bit/output
9-8 EMC2	External Match Control 2 Determines the functionality of External Match 2. 00 - Do Nothing 01 - Clear. Clear the corresponding External Match bit/output to 0 (MAT2 pin is LOW if pinned out) 10 - Set. Set the corresponding External Match bit/output to 1 (MAT2 pin is HIGH if pinned out) 11 - Toggle. Toggle the corresponding External Match bit/output
7-6 EMC1	External Match Control 1 Determines the functionality of External Match 1. 00 - Do Nothing 01 - Clear. Clear the corresponding External Match bit/output to 0 (MAT1 pin is LOW if pinned out) 10 - Set. Set the corresponding External Match bit/output to 1 (MAT1 pin is HIGH if pinned out) 11 - Toggle. Toggle the corresponding External Match bit/output
5-4 EMC0	External Match Control 0 Determines the functionality of External Match 0.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00 - Do Nothing 01 - Clear. Clear the corresponding External Match bit/output to 0 (MAT0 pin is LOW if pinned out) 10 - Set. Set the corresponding External Match bit/output to 1 (MAT0 pin is HIGH if pinned out) 11 - Toggle. Toggle the corresponding External Match bit/output
3 EM3	External Match 3 This bit reflects the state of output MAT3, whether or not this output is connected to a pin. When a match occurs between the TC and MR3, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by MR[11:10]. This bit is driven to the MAT pins if the match function is selected via IOPCTL. 0 = LOW. 1 = HIGH.
2 EM2	External Match 2 This bit reflects the state of output MAT2, whether or not this output is connected to a pin. When a match occurs between the TC and MR2, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[9:8]. This bit is driven to the MAT pins if the match function is selected via IOPCTL. 0 = LOW. 1 = HIGH.
1 EM1	External Match 1 This bit reflects the state of output MAT1, whether or not this output is connected to a pin. When a match occurs between the TC and MR1, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[7:6]. This bit is driven to the MAT pins if the match function is selected via IOPCTL. 0 = LOW. 1 = HIGH.
0 EM0	External Match 0 This bit reflects the state of output MAT0, whether or not this output is connected to a pin. When a match occurs between the TC and MR0, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[5:4]. This bit is driven to the MAT pins if the match function is selected via IOPCTL. 0 = LOW. 1 = HIGH.

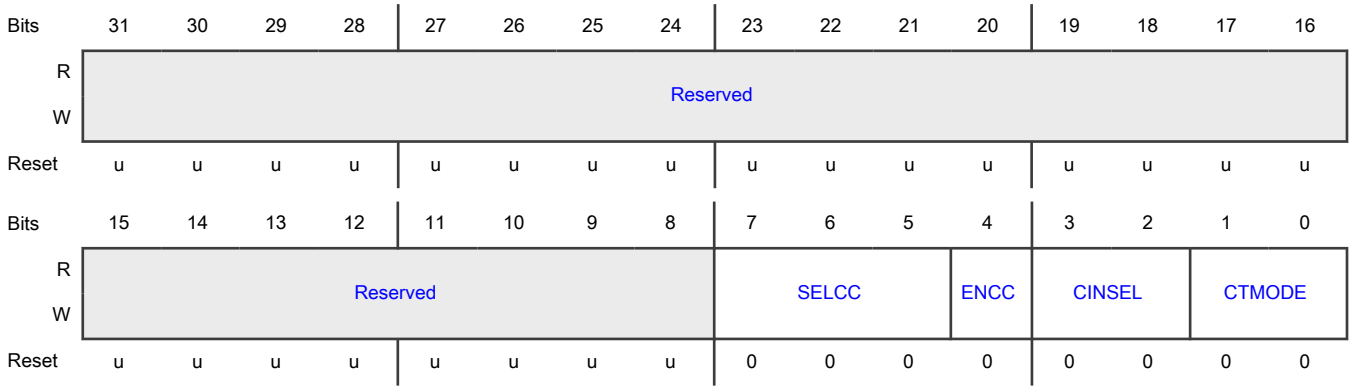
33.6.1.1.11 Count Control Register (CTCR)

The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.

Offset

Register	Offset
CTCR	70h

Diagram



Fields

Field	Description
31-8 —	Reserved Read value is undefined, only zero should be written
7-5 SELCC	Edge select When bit 4 is 1, these bits select which capture input edge will cause the timer and prescaler to be cleared. These bits have no effect when bit 4 is low. Values 0x2 to 0x3 and 0x6 to 0x7 are reserved. 000 - Channel 0 Rising Edge. Rising edge of the signal on capture channel 0 clears the timer (if bit 4 is set) 001 - Channel 0 Falling Edge. Falling edge of the signal on capture channel 0 clears the timer (if bit 4 is set) 010 - Channel 1 Rising Edge. Rising edge of the signal on capture channel 1 clears the timer (if bit 4 is set) 011 - Channel 1 Falling Edge. Falling edge of the signal on capture channel 1 clears the timer (if bit 4 is set) 100 - Channel 2 Rising Edge. Rising edge of the signal on capture channel 2 clears the timer (if bit 4 is set) 101 - Channel 2 Falling Edge. Falling edge of the signal on capture channel 2 clears the timer (if bit 4 is set)
4 ENCC	Setting this bit to 1 enables clearing of the timer and the prescaler when the capture-edge event specified in bits 7:5 occurs.
3-2 CINSEL	Count Input Select When bits 1:0 in this register are not 00, these bits select which CAP pin is sampled for clocking. Note: If Counter mode is selected for a particular CAPn input in the CTCR, the 3 bits for that input in the Capture Control Register (CCR) must be programmed as 000. However, capture and/or interrupt can be selected for the other 3 CAPn inputs in the same timer. 00 - Channel 0. CAPn[0] for CTIMERn 01 - Channel 1. CAPn[1] for CTIMERn 10 - Channel 2. CAPn[2] for CTIMERn 11 - Channel 3. CAPn[3] for CTIMERn
1-0	The Count Control Register (CTCR) is used to select between Timer and Counter mode, and in Counter mode to select the pin and edge(s) for counting.

Table continues on the next page...

Table continued from the previous page...

Field	Description
CTMODE	When Counter Mode is chosen as a mode of operation, the CAP input (selected by the CTCR bits 3:2) is sampled on every rising edge of the CTIMER function clock. After comparing two consecutive samples of this CAP input, one of the following four events is recognized: rising edge, falling edge, either of edges or no changes in the level of the selected CAP input. Only if the identified event occurs and the event corresponds to the one selected by bits 1:0 in the CTCR register, will the Timer Counter register be incremented. Effective processing of the externally supplied clock to the counter has some limitations. Since two successive rising edges of the CTIMER function clock are used to identify only one edge on the CAP selected input, the frequency of the CAP input cannot exceed one half of the CTIMER function clock. Consequently, duration of the HIGH/LOWLOW levels on the same CAP input in this case cannot be shorter than a CTIMER function clock. Bits 7:4 of this register are also used to enable and configure the capture-clears-timer feature. This feature allows for a designated edge on a particular CAP input to reset the timer to all zeros. Using this mechanism to clear the timer on the leading edge of an input pulse and performing a capture on the trailing edge, permits direct pulse-width measurement using a single capture input without the need to perform a subtraction operation in software. 00 - Timer Mode. Incremented every rising CTIMER function clock edge. 01 - Counter Mode rising edge. TC is incremented on rising edges on the CAP input selected by bits 3:2. 10 - Counter Mode falling edge. TC is incremented on falling edges on the CAP input selected by bits 3:2. 11 - Counter Mode dual edge. TC is incremented on both edges on the CAP input selected by bits 3:2.

33.6.1.1.12 PWM Control Register (PWMC)

The PWM Control Register is used to configure the match outputs as PWM outputs. Each match output can be independently set to perform either as PWM output or as match output whose function is controlled by the External Match Register (EMR). Note that different part number and package variations may provide different match output pin functions.

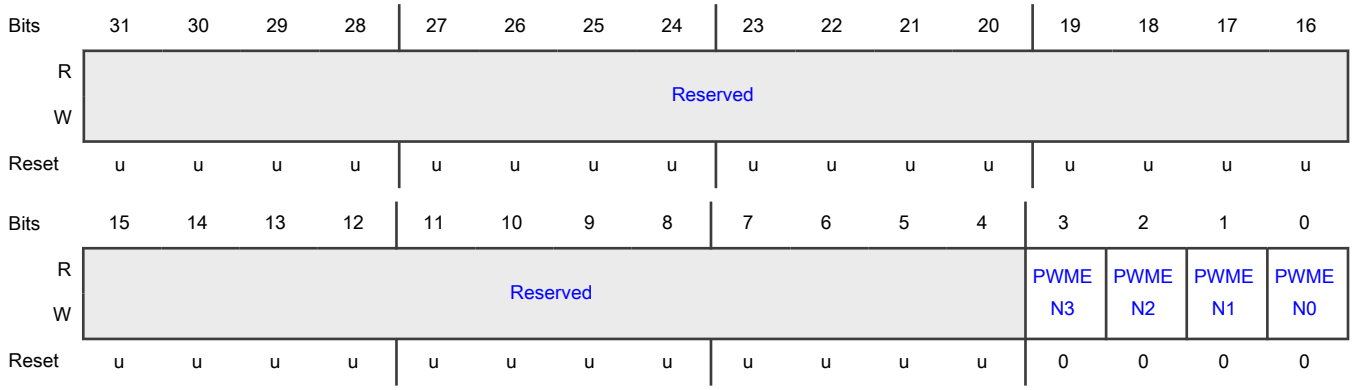
For each timer, a maximum of three single edge controlled PWM outputs can be selected on the MATn[2:0] outputs. One additional match register determines the PWM cycle length. When a match occurs in any of the other match registers, the PWM output is set to HIGH. The timer is reset by the match register that is configured to set the PWM cycle length. When the timer is reset to zero, all currently HIGH match outputs configured as PWM outputs are cleared.

The PWMCON enables PWM mode for the external match pins.

Offset

Register	Offset
PWMC	74h

Diagram



Fields

Field	Description
31-4 —	Reserved Read value is undefined, only zero should be written
3 PWMEN3	PWM mode enable for channel 3. It is recommended to use match channel 3 to set the PWM cycle 0 - Match. CTIMERn_MAT3 is controlled by EM3 1 - PWM. PWM mode is enabled for CT132Bn_MAT3
2 PWMEN2	PWM mode enable for channel 2 0 - Match. CTIMERn_MAT2 is controlled by EM2 1 - PWM. PWM mode is enabled for CTIMERn_MAT2
1 PWMEN1	PWM mode enable for channel 1 0 - Match. CTIMERn_MAT01 is controlled by EM1 1 - PWM. PWM mode is enabled for CTIMERn_MAT1
0 PWMEN0	PWM mode enable for channel 0 0 - Match. CTIMERn_MAT0 is controlled by EM0 1 - PWM. PWM mode is enabled for CTIMERn_MAT0

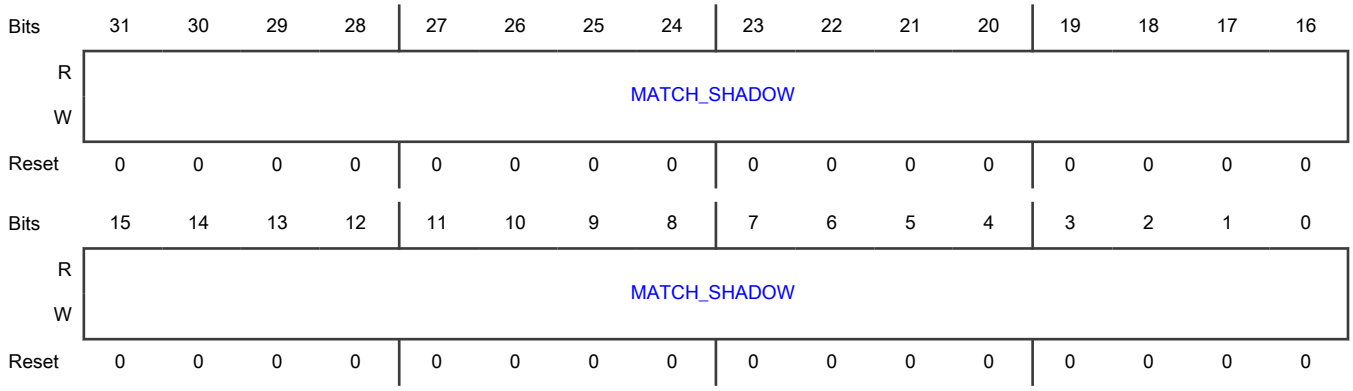
33.6.1.1.13 Match Shadow Register (MSR0 - MSR3)

The Match Shadow registers contain the values that the corresponding Match Registers are (optionally) reloaded with at the start of each new counter cycle. Typically, the match that causes the counter to be reset (and instigates the match reload) will also be programmed to generate an interrupt or DMA request. Software or the DMA engine will then have one full counter cycle to modify the contents of the Match Shadow Register(s) before the next reload occurs.

Offset

Register	Offset
MSR0	78h
MSR1	7Ch
MSR2	80h
MSR3	84h

Diagram



Fields

Field	Description
31-0 MATCH_SHADOW	Timer counter match shadow value If these bits are enabled, the register will be automatically reloaded with the contents of this register whenever the TC is reset to zero.

Chapter 34
SCTIMER

34.1 Chip-specific SCTimer information

Table 312. Reference links to related information

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

34.1.1 Module instances

This device contains one instance of the SCTIMER, SCT0.

34.1.2 Chip-specific signals

See the IOPCTL registers to assign the SCT functions to external pins.

SCT input can come from numerous places and are selected by the SCT Input Mux (see the Peripheral Input Mux Controller (INPUTMUX) chapter). SCT outputs can be routed to multiple places and can be connected (for example) to both a pin and an ADC trigger at the same time.

Table 313. SCT0 signals (inputs)

Type	Function	Connect to	Use register
external from pin	SCT0_GPI0	PIO0_5, PIO0_11, PIO0_14, PIO0_31, PIO1_18, PIO1_30	IOPCTL register for the related pin
	SCT0_GPI1	PIO0_6, PIO0_15, PIO1_0, PIO1_20, PIO1_31	
	SCT0_GPI2	PIO0_12, PIO0_16, PIO0_24, PIO1_1, PIO1_22, PIO2_0	
	SCT0_GPI3	PIO0_13, PIO0_17, PIO1_2, PIO1_24, PIO2_1	
	SCT0_GPI4	PIO0_7, PIO0_19, PIO1_6, PIO1_26	
	SCT0_GPI5	PIO0_8, PIO0_20, PIO0_21, PIO1_7, PIO1_28	
	SCT0_GPI6	PIO0_9, PIO0_18, PIO0_22, PIO0_26, PIO1_8, PIO2_10	
	SCT0_GPI7	PIO0_10, PIO0_23, PIO0_27, PIO1_9	

Table 314. SCT0 signals (outputs)

Type	Function	Connect to	Use register
external to pin	SCT0_OUT0	PIO0_5, PIO0_14, PIO1_19, PIO2_2	IOPCTL register for the related pin
	SCT0_OUT1	PIO0_6, PIO0_15, PIO1_21	
	SCT0_OUT2	PIO0_12, PIO0_16, PIO1_23	
	SCT0_OUT3	PIO0_13, PIO0_17, PIO1_25, PIO2_30	
	SCT0_OUT4	PIO0_7, PIO0_19, PIO1_6, PIO1_27	
	SCT0_OUT5	PIO0_8, PIO0_21, PIO1_7, PIO1_29, PIO2_9	
	SCT0_OUT6	PIO0_9, PIO0_18, PIO0_22, PIO0_31	
	SCT0_OUT7	PIO0_10, PIO1_0, PIO2_31	
	SCT0_OUT8	PIO0_11, PIO0_23	
	SCT0_OUT9	PIO0_24, PIO2_15	

Table 315. Suggested SCT input pin settings

IOPCTL bit(s)	Type D pin
10	I2CFILTER: Set to 1
9	I2CDRIVE: Set to 0.
8	FILTEROFF: Generally set to 1.
7	DIGIMODE: Set to 1.
6	INVERT: Set to 0.
5	I2CSLEW: Set to 1.
4:3	Not used, set to 0.
2:0	FUNC: not used, set to 0. Specific pin inputs are directly connected to the SCT.

Table 316. Suggested SCT output pin settings

IOPCTL bit(s)	Type D pin	Type A pin	Type I pin
10	OD: Set to 0 unless open-drain output is desired.	Same as type D.	I2CFILTER: Set to 1
9	SLEW: Set to 0.	Not used, set to 0	I2CDRIVE: Set to 0.

Table continues on the next page...

Table 316. Suggested SCT output pin settings (continued)

IOPCTL bit(s)	Type D pin	Type A pin	Type I pin
8	FILTEROFF: Set to 1.	Same as type D.	Same as type D.
7	DIGIMODE: Set to 1.	Same as type D.	Same as type D.
6	INVERT: Set to 0.	Same as type D.	Same as type D.
5	Not used, set to 0.	Same as type D.	I2CSLEW: Set to 1.
4:3	MODE: set to 0.	Same as type D.	Not used, set to 0.
2:0	FUNC: Must select the correct function for this peripheral.	Same as type D.	Same as type D.
General comment	A good choice for an SCT output.	A reasonable choice for an SCT output.	Not recommended for SCT outputs.

34.1.3 Register reset values after boot

The reset values of the following registers are affected by boot options and are indeterminate:

- CONFIG
- INPUT

34.2 Overview

The SCTimer/PWM is a powerful, flexible timer module capable of creating complex PWM waveforms and performing other advanced timing and control operations with minimal or no CPU intervention.

The SCT can operate as a single 32-bit counter or as two independent, 16-bit counters in uni-directional or bi-directional mode. Software access to 16-bit registers when the SCT is configured as two 16-bit counters has some limitations, see Important notes on using the SCT as two 16-bit counters.

As with most timers, the SCT supports a selection of match registers against which the count value can be compared, and capture registers where the current count value can be recorded when some pre-defined condition is detected.

An additional feature contributing to the versatility of the SCT is the concept of events. The SCT module supports multiple separate events that can be defined by the user based on some combination of parameters including:

- A match on one of the match registers, and/or
- A transition on one of the SCT inputs or outputs
- The direction of count

Every action that the SCT block performs occurs in direct response to one of these user-defined events; without any software overhead. Any event can be enabled to:

- Start, stop, or halt the counter
- Limit the counter which means to clear the counter in unidirectional mode or change the counter direction in bi-directional mode
- Set, clear, or toggle any SCT output
- Force a capture of the count value into any Capture registers
- Generate an interrupt or DMA request

The SCT allows the user to group and filter events. A group of enabled and disabled events can be described as a state, and several states with different sets of enabled and disabled events are allowed. Changing from one state to another is event driven as well and can therefore happen without software intervention. By defining these states, the SCTimer/PWM provides the means to run entire state machines in hardware to accomplish complex waveform and timing tasks.

In a simple system such as a classical timer/counter with capture and match capabilities, all events that could cause the timer to capture the timer value or toggle a match output are enabled and remain enabled at all times while the counter is running. In this case, no events are filtered and the system is described by one state that does not change. This is the default configuration of the SCT.

In a more complex system, two states could be set up that allow some events in one state and not in the other. An event itself, enabled in both states, can then be used, to move from one state to the other and back while filtering out events in either state. In such a two-state system different waveforms at the SCT output can be created depending on the event history. Changing between states is event-driven and happens without any intervention by the CPU.

Formally, the SCTimer/PWM can be programmed as state machine generator. The ability to perform switching between groups of events provides the SCT the unique capability to be utilized as a highly complex State Machine engine. Events identify the occurrence of conditions that warrant state changes and determine the next state to move to. This provides an extremely powerful control tool - particularly when the SCT inputs and outputs are connected to other on-chip resources.

In addition to events and states, the SCTimer/PWM provides other enhanced features:

- Four alternative clocking modes including a fully asynchronous mode
- Selection of any SCT input as a clock source or a clock gate
- Ability to define a fractional portion to certain match registers. Fractional matching uses a “dither” approach to achieve average resolutions of up to 1/16th of a clock period when generating PWM edges.
- Capability of selecting a greater-than-or-equal-to match condition for the purpose of event generation.

NOTE

In this chapter, the term bus error indicates an SCT response that makes the processor take an exception.

34.2.1 Block diagram

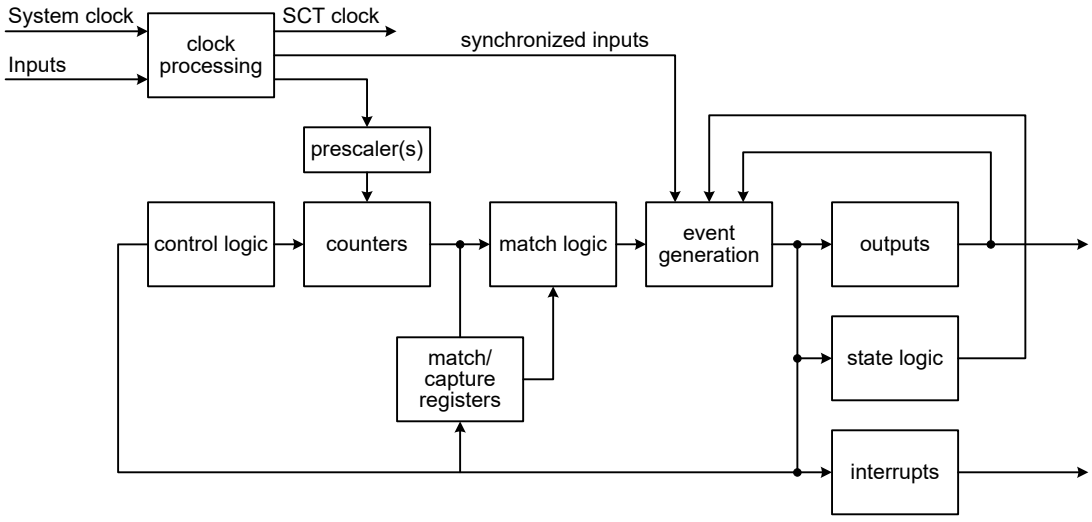


Figure 139. SCTimer block diagram

34.2.2 Features

- The SCTimer/PWM supports:

  - 8 inputs
  - 10 outputs
  - 16 Match/Capture registers
  - 16 events
  - 16 states
  - 6 dithers
- Counter/Timer features:
  - Each SCTimer/PWM is configurable as two 16-bit counters or one 32-bit counter.
  - Counters clocked by system clock or selected input
  - Configurable as up counters or up-down counters
  - Configurable number of Match and Capture registers
  - For match and/or input/output transitions, create the following events for the timer:
    - Interrupt
    - Stop
    - Limit
    - Halt
    - Change counting direction
    - Toggle outputs
    - Change the state
  - Counter value may load into capture register triggered by a match or input/output toggle.
- PWM features:
  - Use Counters with Match registers to toggle outputs and create time-proportioned PWM signals. PWM waveforms can change based on the current state.
  - Up to 8 single-edge or 4 dual-edge PWM outputs with independent duty cycle and common PWM cycle length
- Event creation features:
  - In Bidirectional mode, enable events based on the count direction.
  - The following conditions define an event:
    - A counter match condition
    - An input (or output) condition such as a rising or falling edge or level
    - A combination of match and/or input/output conditions
  - Selected events can limit, halt, start, or stop a counter or change direction.
  - Events trigger state changes, output transitions, timer captures, interrupts, and DMA transactions.
  - Use Match register 0 as an automatic limit.
  - In Bidirectional mode, enable events based on the count direction
  - Match events can be held until another qualifying event occurs
- State control features:
  - A state is defined by events that can happen in the state while the counter is running.
  - A state transitions into another state as a result of an event.

— Each event can be assigned to one or more states.

— State variables allow sequencing across multiple counter cycles.

34.3 Functional description

34.3.1 SCT operation

In its simplest, single-state configuration, the SCT operates as an event-controlled one- or bidirectional counter.

Events can be configured to be

- Counter match events
- An input or output level
- Transitions on an input or output pin
- Or a combination of match and input/output behavior

In response to an event, the SCT output or outputs can transition, or the SCT can perform other actions such as creating an interrupt or starting, stopping, or resetting the counter. Multiple simultaneous actions are allowed for each event. Furthermore, any number of events can trigger one specific action of the SCT.

An action or multiple actions of the SCT uniquely define an event. A state is defined by which events are enabled to trigger an SCT action or actions in any stage of the counter. Events not selected for this state are ignored.

In a multi-state configuration, states change in response to events. A state change is an additional action that the SCT can perform when the event occurs. When an event is configured to change the state, the new state defines a new set of events resulting in different actions of the SCT. Through multiple cycles of the counter, events can change the state multiple times and thus create a large variety of event controlled transitions on the SCT outputs and/or interrupts.

Once configured, the SCT can run continuously without software intervention and can generate multiple output patterns entirely under the control of events.

- To configure the SCT, see Configure the SCT for multiple events and states.
- To start, run, and stop the SCT, see Run the SCT.
- To configure the SCT as simple event controlled counter/timer, see Configure the SCT without using states.

34.3.2 Important notes on using the SCT as two 16-bit counters

The implementation of the SCT on this chip has a limitation. It is not possible to do half-word writes to the upper half of registers, such as (for example) CTRL_H. For the most part, this can be dealt with in software by reading the entire word register, making changes to the upper half-word, and writing back the entire value. Note that for registers CTRL_H, STATE_H, and MATCH_H, this can only be done if both halves of the timer are in HALT mode (halted). Also see the BUSERRH flag in the CONFLAG register.

34.3.3 Configure the SCT without using states

The SCT can function as standard counter/timer with external capture inputs and match outputs without using the state logic. To operate the SCT without states, configure the SCT as follows:

- Write zero to the STATE register (zero is the default).
- Write zero to the EVn_CTRL[STATELD] and EVn_CTRL[STATEV] fields for each event.
- Write 0x1 to the EVn_STATE register of each event. Writing 0x1 enables the event.

In effect, the event occurs in a single state which never changes while the counter is running.

34.3.4 Configure the SCT for multiple events and states

To set up the SCT for multiple events and states, perform the following configuration steps:

34.3.4.1 Configure the counter

1. Configure the L and H counters in the CONFIG register by selecting two independent 16-bit counters (L counter and H counter) or one combined 32-bit counter in the UNIFY field.
2. Select the SCT clock source in the CONFIG register (fields CLKMODE and CLKSEL) from any of the inputs or an internal clock.

34.3.4.2 Configure the match and capture registers

1. Select how many match and capture registers the application uses:
   - In the REGMODE register, select for each of the match/capture register pairs whether the register is used as a match register or capture register.
2. Define match conditions for each match register selected:
   - Each match register MATCH sets one match value, if a 32-bit counter is used, or two match values, when the L and H 16-bit counters are used.
   - Each match reload register MATCHRELOAD sets a reload value that is loaded into the match register when the counter reaches a limit condition or the value 0.

34.3.4.3 Configure events and event responses

State variables allow control of the SCT across more than one cycle of the counter. Counter matches, input/output edges, and state values are combined into a set of general-purpose events that can switch outputs, request interrupts, and change state values.

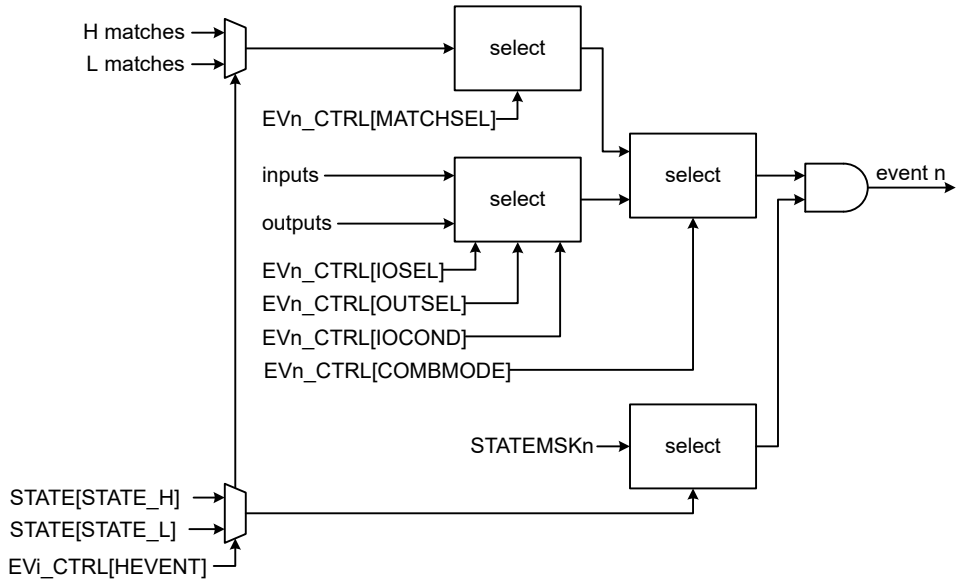


Figure 140. Event selection

1. Define what will trigger each event in the EVn_CTRL registers, one register per event):
   - Select in EVn_CTRL[COMBMODE]field whether the event occurs:
     - On an input or output changing
     - On an input or output level
     - On a match condition of the counter
     - Or a combination of match and input/output conditions in field.

- For a match condition:

  Select the match register that contains the match condition for the event to occur. Enter the number of the selected match register in field EVn_CTRL[MATCHSEL].

  If using L and H counters, define whether the event occurs on matching the L or the H counter in field EVn_CTRL[HEVENT].

- For an SCT input or output level or transition:

  Select the input number or the output number that is associated with this event in fields EVn_CTRL[IOSEL] and EVn_CTRL[OUTSEL].

  Define how the selected input or output triggers the event (edge or level sensitive) in field EVn_CTRL[IOCOND].

2. Define what the effect of each event is on the SCT outputs in the OUTn_SET or OUTn_CLR registers (up to the maximum number of outputs on this device, one register per output):
   - For each SCT output, select which events set or clear this output. More than one event can change the output, and each event can change multiple outputs.
3. Define how each event affects the counter:
   - Set the corresponding event bit in the LIMIT register for the event to set an upper limit for the counter.

     When a limit event occurs in unidirectional mode, the counter is cleared to zero and begins counting up on the next clock edge.

     When a limit event occurs in bidirectional mode, the counter begins to count down from the current value on the next clock edge.
   - Set the corresponding event bit in the HALT register for the event to halt the counter. If the counter is halted, it stops counting and no new events can occur. The counter operation can only be restored by clearing the HALT_L and/or the HALT_H bits in the CTRL register.
   - Set the corresponding event bit in the STOP register for the event to stop the counter. If the counter is stopped, it stops counting. However, an event that is configured as a transition on an input/output can restart the counter.
   - Set the corresponding event bit in the START register for the event to restart the counting. Only events that are defined by an input changing can be used to restart the counter.
4. Define which events contribute to the SCT interrupt:
   - Set the corresponding event bit in the EVEN and the EVFLAG registers to enable the event to contribute to the SCT interrupt.

34.3.4.4 Configure multiple states

1. In the EVn_STATE register for each event (up to the maximum number of events on this device, one register per event), select the state or states (up to 2) in which this event is allowed to occur. Each state can be selected for more than one event.
2. Determine how the event affects the system state:

In the EVn_CTRL registers (up to the maximum number of events on this device, one register per event), set the new state value in the EVn_CTRL[STATEV} field for this event. If the event is the highest numbered in the current state, this value is either added to the existing state value or replaces the existing state value, depending on the field EVn_CTRL[STATELD].

NOTE

If there are higher numbered events in the current state, this event cannot change the state.

If the EVn_CTRL[STATEV] and EVn_CTRL[STATELD] values are set to zero, the state does not change.

34.3.4.5 Miscellaneous options

- There are a certain (selectable) number of capture registers. Each capture register can be programmed to capture the counter contents when one or more events occur.
- If the counter is in bidirectional mode, the effect of set and clear of an output can be made to depend on whether the counter is counting up or down by writing to the OUTPUTDIRCTRL register.

34.3.5 State logic

The SCT can be configured as a timer/counter with multiple programmable states. The states are user-defined through the events that can be captured in each particular state. In a multi-state SCT, the SCT can change from one state to another state when a user-defined event triggers a state change. The state change is triggered through each event's EV_CTRL register in one of the following ways:

- The event can increment the current state number by a new value.
- The event can write a new state value.

If an event increments the state number beyond the number of available states, the SCT enters a locked state where all further events are ignored while the counter is still running. Software must interfere to change out of this state.

Software can capture the counter value (and potentially create an interrupt and write to all outputs) when the event moving the SCT into a locked state occurs. Later, while the SCT is in the locked state, software can read the counter again to record the time passed since the locking event and can also read the state variable to obtain the current state number.

If the SCT registers an event that forces an abort, putting the SCT in a locked state can be a safe way to record the time that has passed since the abort event while no new events are allowed to occur. Since multiple states (any state number between the maximum implemented state and 31) are locked states, multiple abort or error events can be defined each incrementing the state number by a different value.

34.3.6 Clearing the prescaler

When enabled by a non-zero CTRL[PRE_L] and/or CTRL[PRE_H] field, the prescaler acts as a clock divider for the counter, like a fractional part of the counter value. The prescaler is cleared whenever the counter is cleared or loaded for any of the following reasons:

- Hardware reset
- Software writing to the counter register
- Software writing a 1 to a CTRL[CLRCTR] bit
- An event selected by a 1 in the counter limit register when CTRL[BIDIR] = 0

When CTRL[BIDIR] is 0, a limit event caused by an I/O signal can clear a non-zero prescaler. However, a limit event caused by a Match only clears a non-zero prescaler in one special case as described in Match vs. I/O events.

A limit event when CTRL[BIDIR] is 1 does not clear the prescaler. Instead, it clears the CTRL[DOWN] bit, and decrements the counter on the same clock if the counter is enabled in that clock.

34.3.7 Match vs. I/O events

Counter operation is complicated by the prescaler and by clock mode 01 in which the SCT clock is the bus clock. However, the prescaler and counter are enabled to count only when a selected edge is detected on a clock input.

- The prescaler is enabled when the clock mode is not 01, or when the input edge selected by the CONFIG[CLKSEL] field is detected.
- The counter is enabled when the prescaler is enabled, and (PRELIM=0 or the prescaler is equal to the value in PRELIM).

An I/O component of an event can occur in any SCT clock when its counter CTRL[HALT] bit is 0. In general, a Match component of an event can only occur in a SCT clock when its counter CTRL[HALT] and CTRL[STOP] bits are both 0 and the counter is enabled.

The Event conditions table shows when the various kinds of events can occur.

Table 317. Event conditions

EVn_CTRL[COMBMODE]	EVn_CTRL[IOCOND]	Event can occur on clock:
IO	Any	Event can occur whenever CTRL[HALT] = 0.
MATCH	Any	Event can occur when CTRL[HALT] = 0 and CTRL[STOP] = 0 and the counter is enabled.
OR	Any	From the I/O component: Event can occur whenever CTRL[HALT] = 0. From the match component: Event can occur when CTRL[HALT] = 0 and CTRL[STOP] = 0 and the counter is enabled.
AND	LOW or HIGH	Event can occur when CTRL[HALT] = 0 and CTRL[STOP] = 0 and the counter is enabled.
AND	RISE or FALL	Event can occur whenever CTRL[HALT] = 0.

34.3.8 Fractional matches

The first 6 match registers may be configured to have a fractional portion to their match values. Higher average resolution is achievable on the match registers with associated fractional match register by using a dithering mechanism. The dither engine delays the assertion of a match by one counter clock every n (0 to 15) out of 16 counter cycles. The value of n is specified in the 4-bit FRACMAT register associated with each of the first six match registers.

Dithering can be disabled on any of the match registers by loading all zeroes (the default value) into its FRACMAT register.

34.3.8.1 Dithering

At the start of each new SCT counter cycle (that is, when the counter counts-down to zero in bi-directional mode or is cleared to zero by a limit event), the dither engine determines which matches are to be delayed by one clock during the coming counter cycle. Delaying the match effectively adds 1 to the designated match value when up-counting or subtracts 1 when down-counting, during that particular counter cycle.

For each dither-enabled match register, the value programmed in its associated FRACMAT register specifies how many out of every 16 counter cycles its match is to be delayed. An algorithm applied to the FRACMAT value distributes this number as evenly as possible across the 16 counter cycles. This results in a unique dither pattern for each match register. See Table 318.

Additional control over the dithering process is provided to the user through a Dither Condition (event-mask) register. Typically, the dither engine advances though the match dither patterns at the start of every new SCT counter cycle. The Dither Condition register allows the user to specify that advancement to the next element in the dither patterns will only occur if one or more designated events occurred during the previous cycle of the counter.

The dither algorithm is designed to spread out the cycles in which the matches are delayed as evenly as possible across the 16 counter cycles. The following table shows the dither pattern that is applied for each value of FRACMAT. A ‘D’ indicates the counter cycles where a match on the relevant match register is delayed.

Table 318. Dither pattern

	Counter cycle															
FRACMAT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x1	-	-	-	-	-	-	-	-	D	-	-	-	-	-	-	-
0x2	-	-	-	-	D	-	-	-	-	-	-	-	D	-	-	-

Table continues on the next page...

Table 318. Dither pattern (continued)

	Counter cycle															
FRACMAT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0x3	-	-	-	-	D	-	-	-	D	-	-	-	D	-	-	-
0x4	-	-	D	-	-	-	D	-	-	-	D	-	-	-	D	-
0x5	-	-	D	-	-	-	D	-	D	-	D	-	-	-	D	-
0x6	-	-	D	-	D	-	D	-	-	-	D	-	D	-	D	-
0x7	-	-	D	-	D	-	D	-	D	-	D	-	D	-	D	-
0x8	-	D	-	D	-	D	-	D	-	D	-	D	-	D	-	D
0x9	-	D	-	D	-	D	-	D	D	D	-	D	-	D	-	D
0xA	-	D	-	D	D	D	-	D	-	D	-	D	D	D	-	D
0xB	-	D	-	D	D	D	-	D	D	D	-	D	D	D	-	D
0xC	-	D	D	D	-	D	D	D	-	D	D	D	-	D	D	D
0xD	-	D	D	D	-	D	D	D	D	D	D	D	-	D	D	D
0xE	-	D	D	D	D	D	D	D	-	D	D	D	D	D	D	D
0xF	-	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D

34.4 Signals

SCT inputs can come from numerous places and are selected by the SCT input mux.

SCT outputs can be routed to multiple places and can be connected to both a pin and an ADC trigger at the same time.

Table 319. SCT signals

Signal	I/O	Description
SCT_INn	I	SCT Input. Signals come from GPIO sources, each from a variety of pins selected via IOPCTL muxing.
SCT_OUTn	O	SCT Output. Signals go to IOPCTL muxing where they can be (optionally) output onto a selection of GPIO pins.

34.5 Initialization

To initialize the SCT:

- Enable the clock to the SCTimer/PWM (SCT)
- Clear the SCT peripheral reset
- Configure the interrupts
- Connect the SCT outputs to external pins
- Configure the DMA request lines to connect to the DMA trigger inputs

34.6 Application information

This section describes applications supported by the SCTimer module.

34.6.1 Run the SCT

1. Initialize the SCT (see Initialization).
2. Write to the STATE register to define the initial state. By default the initial state is state 0.
3. To start the SCT, write to the CTRL register:
   - Clear the counters.
   - Clear or set the CTRL[STOP_L] and/or CTRL[STOP_H] bits.

     NOTE

     The counter starts counting once the CTRL[STOP] bit is cleared as well. If the CTRL[STOP] bit is set, the SCT waits instead for an event to occur that is configured to start the counter.

   - For each counter, select unidirectional or bidirectional counting mode (field CTRL[BIDIR_L] and/or CTRL[BIDIR_H]).
   - Select the prescale factor for the counter clock (CTRL[PRE_L or CTRL[PRE_H]).
   - Clear the CTRL[HALT_L] and/or CTRL[HALT_H] bit. By default, the counters halt and no events can occur.
4. Use software to stop the counters at any time, stop or halt the counter (write to STOP_L and/or STOP_H bits or HALT_L and/or HALT_H bits in the CTRL register).
   - When the counters stop, both an event configured to clear the CTRL[STOP_L] and/or CTRL[STOP_H] bit or software writing a zero to the CTRL[STOP_L] and/or CTRL[STOP_H] bit can start the counter again.
   - When the counter halts, only a software write to clear the CTRL[HALT_L] and/or CTRL[HALT_H] bit can start the counter again. No events can occur.
   - When the counters halt, software can set any SCT output HIGH or LOW directly by writing to the OUT register.

Read the current state at any time in the STATE register.

Use software to change the current state (that is, independently of any event occurring), set the CTRL[HALT] bit and write to the STATE register to change the state value. Writing to the STATE register is only allowed when the counter halts (the HALT_L and/or HALT_H bits are set) and no events can occur.

34.6.2 SCT PWM Example

The SCT configuration example shows a simple application of the SCT using two sets of match events (EV0/1 and EV3/4) to set/clear SCT output 0. The timer is automatically reset whenever it reaches the MAT0 match value.

In the initial state 0, match event EV0 sets output 0 to HIGH and match event EV1 clears output 0.

While monitoring the SCT input 0:

If input 0 is LOW at the next timer reset (EV2), the state changes to state 1, enabling events EV3 and EV4, which creates the same output but with different match value triggers.

If input 0 is HIGH at the next timer reset, the associated event (EV5) causes the state to change back to the state enabling events EV0 and EV1.

The example uses the following SCT configuration:

- 1 input
- 1 output
- 5 match registers
- 6 events and match 0 used with autolimit function
- 2 states

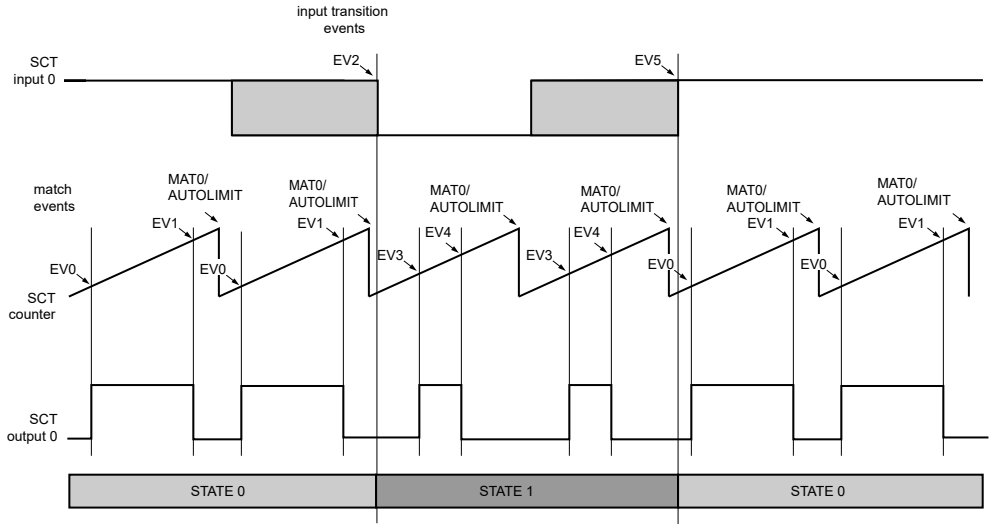


Figure 141. SCT configuration example

This application of the SCT uses the following configuration (all register values not listed are their default values):

Table 320. SCT configuration example

Configuration	Registers	Setting
Counter	CONFIG	Uses one counter (UNIFY = 1).
	CONFIG	Enable the autolimit for MAT0. (AUTOLIMIT_L = 1.)
	CTRL	Uses unidirectional counter (BIDIR_L = 0).
Clock base	CONFIG	Uses default values for clock configuration.
Match/Capture registers	REGMODE	Configure one match register for each match event by setting REGMODE_L bits 0, 1, 2, 3, 4 to 0, which is the default.
Define match values	MATCH 0/1/2/3/4	Set a match value MATCH0/1/2/4_L in each register. The match 0 register serves as an automatic limit event that resets the counter, without using an event. To enable the automatic limit, set the AUTOLIMIT bit in the CONFIG register.
Define match reload values	MATCHREL 0/1/2/3/4	Set a match reload value RELOAD0/1/2/3/4_L in each register (same as the match value in this example).
Define when event 0 occurs	EV0_CTRL	• Set COMBMODE = 0x1. Event 0 uses match condition only. • Set MATCHSEL = 1. Select match value of match register 1. The match value of MAT1 is associated with event 0.
Define when event 1 occurs	EV1_CTRL	• Set COMBMODE = 0x1. Event 1 uses match condition only. • Set MATCHSEL = 2 Select match value of match register 2. The match value of MAT2 is associated with event 1.

Table continues on the next page...

Table 320. SCT configuration example (continued)

Configuration	Registers	Setting
Define when event 2 occurs	EV2_CTRL	• Set COMBMODE = 0x3. Event 2 uses match condition and I/O condition. • Set IOSEL = 0. Select input 0. • Set IOCOND = 0x0. Input 0 is LOW. • Set MATCHSEL = 0. Chooses match register 0 to qualify the event.
Define how event 2 changes the state	EV2_CTRL	Set STATEV bits to 1 and the STATED bit to 1. Event 2 changes the state to state 1.
Define when event 3 occurs	EV3_CTRL	• Set COMBMODE = 0x1. Event 3 uses match condition only. • Set MATCHSEL = 0x3. Select match value of match register 3. The match value of MAT3 is associated with event 3.
Define when event 4 occurs	EV4_CTRL	• Set COMBMODE = 0x1. Event 4 uses match condition only. • Set MATCHSEL = 0x4. Select match value of match register 4.The match value of MAT4 is associated with event 4.
Define when event 5 occurs	EV5_CTRL	• Set COMBMODE = 0x3. Event 5 uses match condition and I/O condition. • Set IOSEL = 0. Select input 0. • Set IOCOND = 0x3. Input 0 is HIGH. • Set MATCHSEL = 0. Chooses match register 0 to qualify the event.
Define how event 5 changes the state	EV5_CTRL	Set STATEV bits to 0 and the STATED bit to 1. Event 5 changes the state to state 0.
Define by which events output 0 is set	OUT0_SET	Set SET0 bits 0 (for event 0) and 3 (for event 3) to one to set the output when these events 0 and 3 occur.
Define by which events output 0 is cleared	OUT0_CLR	Set CLR0 bits 1 (for events 1) and 4 (for event 4) to one to clear the output when events 1 and 4 occur.
Configure states to enable event 0	EV0_STATE	Set STATEMSK0 bit 0 to 1, enabling Event 0 in state 0. Set all other bits to 0.
Configure states to enable event 1	EV1_STATE	Set STATEMSK1 bit 0 to 1, enabling Event 1 in state 0. Set all other bits to 0.
Configure states to enable event 2	EV2_STATE	Set STATEMSK2 bit 0 to 1, enabling Event 2 in state 0. Set all other bits to 0.
Configure states to enable event 3	EV3_STATE	Set STATEMSK3 bit 1 to 1, enabling Event 3 in state 1. Set all other bits to 0.
Configure states to enable event 4	EV4_STATE	Set STATEMSK4 bit 1 to 1, enabling Event 4 in state 1. Set all other bits to 0.

Table continues on the next page...

Table 320. SCT configuration example (continued)

Configuration	Registers	Setting
Configure states to enable event 5	EV5_STATE	Set STATEMSK5 bit 1 to 1, enabling Event 5 in state 1. Set all other bits to 0.

34.7 Memory map and register definition

This section includes the SCTIMER module memory map and detailed descriptions of all registers.

34.7.1 Register description

For most of the SCT registers, the register function depends on the setting of certain other register bits:

1. The CONFIG[UNIFY] bit determines whether the SCT is used as one 32-bit register (for operation as one 32-bit counter/timer) or as two 16-bit counter/timers named L and H.
   - CONFIG[UNIFY] = 1: Only one register is used (for operation as one 32-bit counter/timer).
   - CONFIG[UNIFY] = 0: Access the L and H registers by a 32-bit read or write operation or can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

Typically, the CONFIG[UNIFY] bit is configured by writing to the CONFIG register before any other registers are accessed.

1. The REGMODEn bits in the REGMODE register determine whether each set of Match/Capture registers uses the match or capture functionality:
   - REGMODEn = 0: Registers operate as match and reload registers.
   - REGMODEn = 1: Registers operate as capture and capture control registers.

34.7.2 Register functional grouping

Most SCT registers either configure an event or select an event for a specific action of the counter (or counters) and outputs. The diagram shows the registers and register bits that need to be configured for each event.

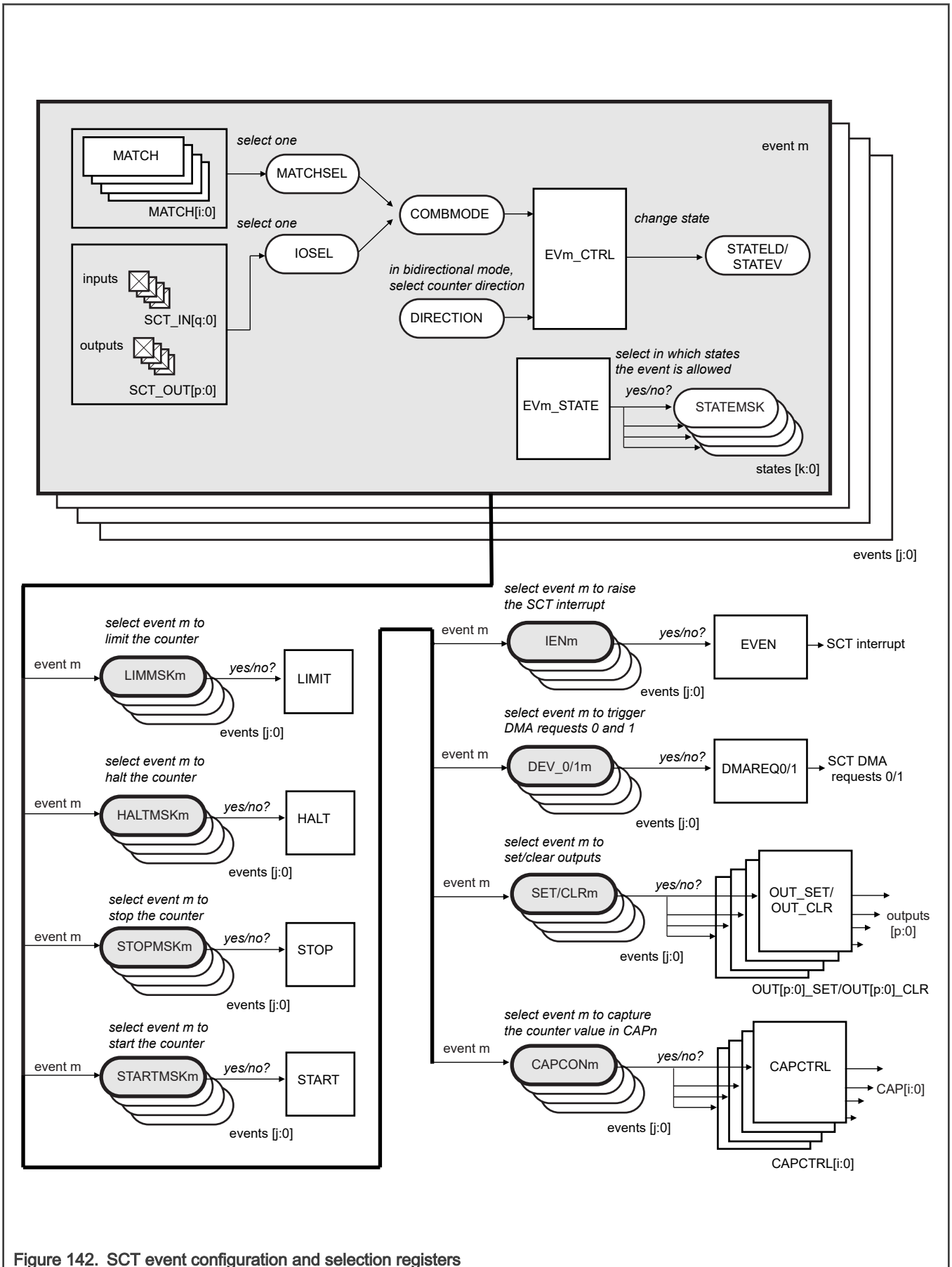


Figure 142. SCT event configuration and selection registers

NOTE

In this figure, letters are used to represent the maximum quantity of certain SCTimer/PWM features as noted below.

i = match/captures, j = events, k = states, p = outputs, q = inputs

34.7.2.1 Counter configuration and control registers

The SCT contains two registers for configuring the SCT and to monitor and control its operation by software.

- The Configuration register (CONFIG) configures the SCT in single, 32-bit counter mode or in dual, 16-bit counter mode, configures the clocking and clock synchronization, and configures automatic limits and the use of reload registers.
- The Control register (CTRL) allows monitoring and setting the counter direction, and to clear, start, stop, or halt the 32-bit counter or each individual 16-bit counter if in dual-counter mode.

34.7.2.2 Event configuration registers

Each event is associated with two registers:

- One EVn_STATE register per event to enable the event.
- One EVn_CTRL register per event to define what triggers the event.

34.7.2.3 Match and capture registers

The SCT includes a set of registers to store the SCT match or capture values. Each Match register is associated with a Match Reload register which automatically reloads the Match register at the beginning of each counter cycle. This register group includes the following registers:

- One REGMODE register to configure each Match/Capture register for either storing a match value or a capture value.
- A set of Match/Capture registers with each register, depending on the setting of REGMODE, either storing a match value or a counter capture value.
- One Match Reload register for each Match register.
- One Capture Control register for each Capture register.

This diagram shows the match logic for the SCT.

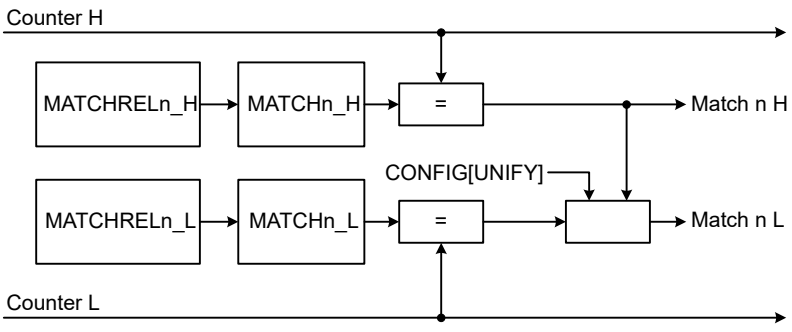


Figure 143. Match logic

This diagram shows the capture logic for the SCT.

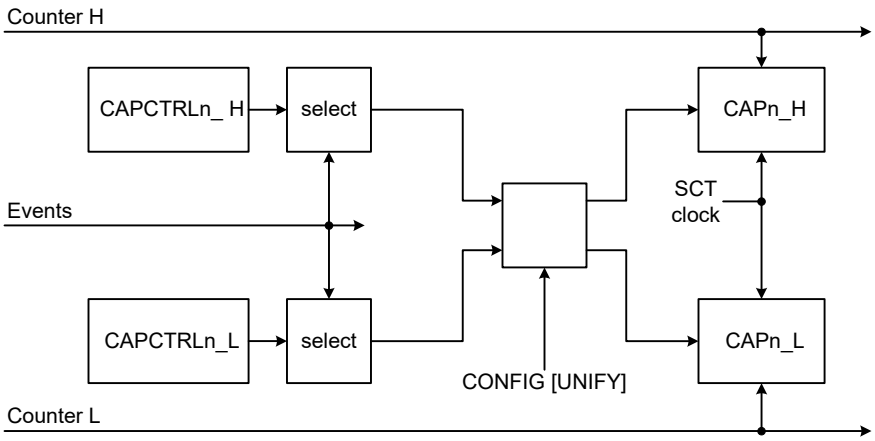


Figure 144. Capture logic

34.7.2.4 Event select registers for the counter operations

This group contains the registers that select which events affect the counter. Counter actions are limit, halt, and start or stop. The counter actions apply to the unified counter or to the two 16-bit counters. Also included is the Counter register with the counter value, or values in the dual-counter set-up. This register group includes the following registers:

- LIMIT selects the events that limit the counter.
- START and STOP select events that start or stop the counter.
- HALT selects events that halt the counter
- COUNT contains the counter value.

The LIMIT, START, STOP, and HALT registers each contain one bit per event that selects for each event whether the event limits, stops, starts, or halts the counter, or counters in dual-counter mode.

In the dual-counter mode, the events can be selected independently for each counter.

34.7.2.5 Event select registers for setting or clearing the outputs

This group contains the registers that select the events which affect the level of each SCT output. Also included are registers to manage conflicts that occur when events try to set or clear the same output. This register group includes the following registers:

- One Output n Set register (OUTn_SET) for each output to select the events which set the output.
- One Output n Clear register (OUTn_CLR) for each output to select the events which clear the output.
- The Conflict Resolution register (RES) which defines an action when more than one event try to control an output at the same time.
- The Conflict Flag (CONFLAG) and Conflict Interrupt Enable (CONEN) registers that monitor interrupts arising from output set and clear conflicts.
- The Output Counter Direction Control register (OUTPUTDIRCTRL) that interchanges the set and clear output operation caused by an event in bidirectional mode.

The OUTn_SET and OUTn_CLR registers each contain one bit per event that selects whether the event changes the state for a given output n.

In the dual-counter mode, the events can be selected independently for each output.

Output slice n shows one output slice of the SCT.

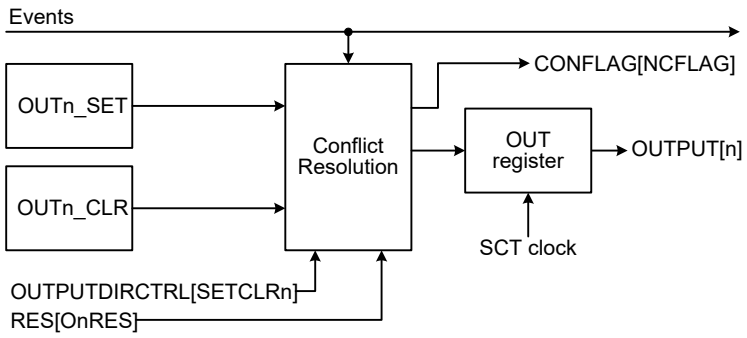


Figure 145. Output slice n

34.7.2.6 Event select registers for capturing a counter value

This group contains registers that select events which capture the counter value and store it in one of the CAPn registers. Each Capture register n has one associated CAPCTRLn register which in turn selects the events to capture the counter value.

34.7.2.7 Event select register for initiating DMA transfers

One register is provided for each of the two DMA requests to select the events that can generate a DMA request.

The DMAREQn register contain one bit for each event that selects whether this event generates a DMA request. An additional bit enables the DMA trigger when the match registers are reloaded.

34.7.2.8 Interrupt handling registers

The following registers provide flags that are set by events and select the events that request an interrupt.

- The Event Flag register (EVFLAG) provides one flag for each event that is set when the event occurs.
- The Event Interrupt Enable register (EVEN) provides one bit for each event to be enabled for the SCT interrupt.

This diagram shows an event triggering the SCT to generate one interrupt.

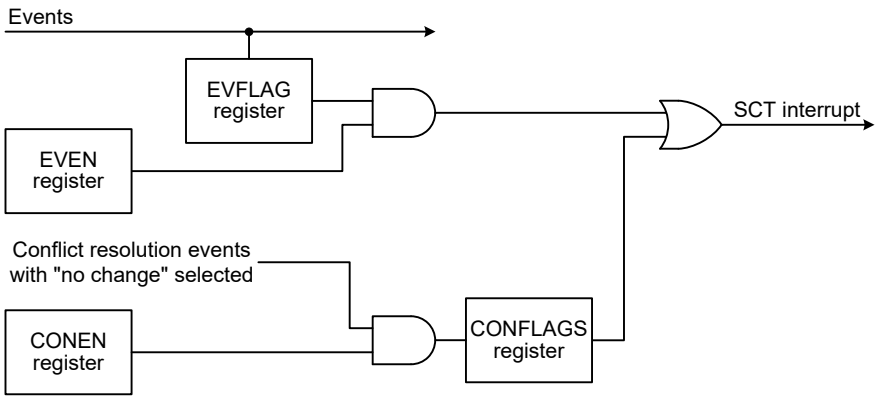


Figure 146. SCT interrupt generation

34.7.2.9 Registers for controlling SCT inputs and outputs by software

Two registers are provided that allow software (as opposed to events) to set input and outputs of the SCT:

- The SCT Input register (INPUT) to read the state of any of the SCT inputs
- The SCT Output register (OUTPUT) to set or clear any of the SCT outputs or to read the state of the outputs

34.7.3 SCTimer register descriptions

34.7.3.1 SCT memory map

SCT0 base address: 4008_5000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	SCTimer Configuration (CONFIG)	32	See section	See section
4	SCT Control (CTRL)	32	See section	See section
8	SCT Limit Event Select (LIMIT)	32	RW	0000_0000
C	Halt Event Select (HALT)	32	RW	0000_0000
10	Stop Event Select (STOP)	32	RW	0000_0000
14	Start Event Select (START)	32	RW	0000_0000
18	Dither Condition (DITHER)	32	RW	0000_0000
40	Counter (COUNT)	32	RW	0000_0000
44	State (STATE)	32	See section	See section
48	Input (INPUT)	32	RO	0000_0000
4C	Match/Capture Mode (REGMODE)	32	RW	0000_0000
50	Output (OUTPUT)	32	See section	See section
54	Output Counter Direction Control (OUTPUTDIRCTRL)	32	See section	0000_0000
58	Output Conflict Resolution (RES)	32	See section	0000_0000
5C	DMA Request 0 (DMAREQ0)	32	See section	See section
60	DMA Request 1 (DMAREQ1)	32	See section	See section
F0	Event Interrupt Enable (EVEN)	32	See section	See section
F4	Event Flag (EVFLAG)	32	See section	See section
F8	Conflict Interrupt Enable (CONEN)	32	See section	See section
FC	Conflict Flag (CONFLAG)	32	See section	See section

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
100 - 13C	Capture Value (CAP0 - CAP15)	32	RW	0000_0000
100 - 13C	Match Value (MATCH0 - MATCH15)	32	RW	0000_0000
140 - 154	Fractional Match (FRACMAT0 - FRACMAT5)	32	See section	0000_0000
200 - 23C	Capture Control (CAPCTRL0 - CAPCTRL15)	32	RW	0000_0000
200 - 23C	Match Reload Value (MATCHREL0 - MATCHREL15)	32	RW	0000_0000
240 - 254	Fractional Match Reload (FRACMATREL0 - FRACMATREL5)	32	See section	0000_0000
300	Event n State (EV0_STATE)	32	RW	See section
304	Event n Control (EV0_CTRL)	32	See section	See section
308	Event n State (EV1_STATE)	32	RW	See section
30C	Event n Control (EV1_CTRL)	32	See section	See section
310	Event n State (EV2_STATE)	32	RW	See section
314	Event n Control (EV2_CTRL)	32	See section	See section
318	Event n State (EV3_STATE)	32	RW	See section
31C	Event n Control (EV3_CTRL)	32	See section	See section
320	Event n State (EV4_STATE)	32	RW	See section
324	Event n Control (EV4_CTRL)	32	See section	See section
328	Event n State (EV5_STATE)	32	RW	See section
32C	Event n Control (EV5_CTRL)	32	See section	See section
330	Event n State (EV6_STATE)	32	RW	See section
334	Event n Control (EV6_CTRL)	32	See section	See section
338	Event n State (EV7_STATE)	32	RW	See section
33C	Event n Control (EV7_CTRL)	32	See section	See section
340	Event n State (EV8_STATE)	32	RW	See section

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
344	Event n Control (EV8_CTRL)	32	See section	See section
348	Event n State (EV9_STATE)	32	RW	See section
34C	Event n Control (EV9_CTRL)	32	See section	See section
350	Event n State (EV10_STATE)	32	RW	See section
354	Event n Control (EV10_CTRL)	32	See section	See section
358	Event n State (EV11_STATE)	32	RW	See section
35C	Event n Control (EV11_CTRL)	32	See section	See section
360	Event n State (EV12_STATE)	32	RW	See section
364	Event n Control (EV12_CTRL)	32	See section	See section
368	Event n State (EV13_STATE)	32	RW	See section
36C	Event n Control (EV13_CTRL)	32	See section	See section
370	Event n State (EV14_STATE)	32	RW	See section
374	Event n Control (EV14_CTRL)	32	See section	See section
378	Event n State (EV15_STATE)	32	RW	See section
37C	Event n Control (EV15_CTRL)	32	See section	See section
500	Output n Set (OUT0_SET)	32	See section	See section
504	Output n Clear (OUT0_CLR)	32	See section	See section
508	Output n Set (OUT1_SET)	32	See section	See section
50C	Output n Clear (OUT1_CLR)	32	See section	See section
510	Output n Set (OUT2_SET)	32	See section	See section
514	Output n Clear (OUT2_CLR)	32	See section	See section

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
518	Output n Set (OUT3_SET)	32	See section	See section
51C	Output n Clear (OUT3_CLR)	32	See section	See section
520	Output n Set (OUT4_SET)	32	See section	See section
524	Output n Clear (OUT4_CLR)	32	See section	See section
528	Output n Set (OUT5_SET)	32	See section	See section
52C	Output n Clear (OUT5_CLR)	32	See section	See section
530	Output n Set (OUT6_SET)	32	See section	See section
534	Output n Clear (OUT6_CLR)	32	See section	See section
538	Output n Set (OUT7_SET)	32	See section	See section
53C	Output n Clear (OUT7_CLR)	32	See section	See section
540	Output n Set (OUT8_SET)	32	See section	See section
544	Output n Clear (OUT8_CLR)	32	See section	See section
548	Output n Set (OUT9_SET)	32	See section	See section
54C	Output n Clear (OUT9_CLR)	32	See section	See section

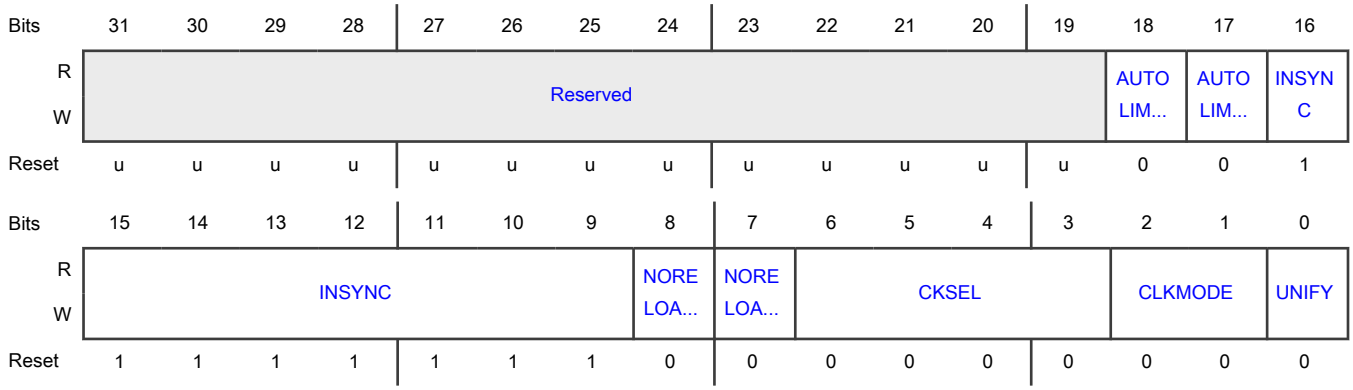
34.7.3.1.1 SCTimer Configuration (CONFIG)

The CONFIG register configures the overall operation of the SCT. Write to the CONFIG register before writing to any other registers. Only word-writes are permitted to this register. Attempting to write a half-word value results in a bus error.

Offset

Register	Offset
CONFIG	0h

Diagram



Fields

Field	Description
31-19 —	Reserved
18 AUTOLIMIT_H	Auto Limit Higher A one in this bit will cause a match on match register 0 to be treated as the LIMIT condition without the need to define an associated event. As with any LIMIT event, this automatic limit causes the counter to be cleared to zero in unidirectional mode or to change the direction of count in bi-directional mode. Software can write to set or clear this bit at any time. This bit is not used when the UNIFY bit is set. 0 - Disable. 1 - Enable. A match on match register 0 is the LIMIT condition. No need to define an associated event.
17 AUTOLIMIT_L	Auto Limit Lower As with any LIMIT event, this automatic limit causes the counter to be cleared to zero in unidirectional mode or to change the direction of count in bi-directional mode. Software can write to set or clear this bit at any time. AUTOLIMIT_L applies to both the higher and lower registers when the UNIFY bit is set. 0 - Disable. 1 - Enable. A match on match register 0 is the LIMIT condition. No need to define an associated event.
16-9 INSYNC	Input Synchronization Synchronization for input N (bit 9 = input 0, bit 10 = input 1, ... bit 16 = input 7). A 1 in one of these bits subjects the corresponding input to synchronization to the SCT clock, before it is used to create an event. If an input is known to already be synchronous with the SCT clock, INSYNC may be set to 0 for faster input response. **NOTE** The SCT clock is the system clock for CKMODEs 0-2. It is the selected, asynchronous SCT input clock for CKMODE3. The INSYNC field only affects inputs used for event generation. INSYNC does not apply to the clock input specified in the CKSEL field.

Table continues on the next page...

Table continued from the previous page...

Field	Description
8 NORELOAD_H	No Reload Higher Match Setting NORELOAD_H eliminates the need to write to the reload registers MATCHREL if the match values are fixed. Software can write to set or clear this bit at any time. This bit is not used when the UNIFY = 1. 0 - Reload. The default setting. 1 - No Reload. Prevents the higher match registers from being reloaded from their respective reload registers.
7 NORELOAD_L	No Reload Lower Match Setting NORELOAD_L eliminates the need to write to the reload registers MATCHREL if the match values are fixed. Software can write to set or clear NORELOAD_L at any time. This bit applies to both the higher and lower registers when the UNIFY = 1. 0 - Reload. The default setting. 1 - No Reload. Prevents the lower match registers from being reloaded from their respective reload registers.
6-3 CKSEL	SCT Clock Select. The specific functionality of the designated input/edge is dependent on the CLKMODE bit selection in this register. 0000 - Rising edges on input 0 0001 - Falling edges on input 0 0010 - Rising edges on input 1 0011 - Falling edges on input 1 0100 - Rising edges on input 2 0101 - Falling edges on input 2 0110 - Rising edges on input 3 0111 - Falling edges on input 3 1000 - Rising edges on input 4 1001 - Falling edges on input 4 1010 - Rising edges on input 5 1011 - Falling edges on input 5 1100 - Rising edges on input 6 1101 - Falling edges on input 6 1110 - Rising edges on input 7 1111 - Falling edges on input 7
2-1 CLKMODE	SCT Clock Mode 00 - System Clock Mode. The system clock clocks the entire SCT module including all counters and counter prescalers.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01 - Sampled System Clock Mode. The system clock clocks the SCT module, but the counter and prescalers are only enabled to count when the designated edge is detected on the input selected by the CKSEL field. The minimum pulse width on the selected clock-gate input is 1 bus clock period. This mode is the high-performance, sampled-clock mode. 10 - SCT Input Clock Mode. The input/edge selected by the CKSEL field clocks the SCT module, including all counters and prescalers, after first being synchronized to the system clock. The minimum pulse width on the clock input is 1 bus clock period. This mode is the low-power, sampled-clock mode. 11 - Asynchronous Mode. The entire SCT module is clocked directly by the input/edge selected by the CKSEL field. In this mode, the SCT outputs are switched synchronously to the SCT input clock - not the system clock. The input clock rate must be at least half the system clock rate and can be the same or faster than the system clock.
0 UNIFY	SCT Operation The UNIFY bit determines whether the SCT is used as a single 32-bit counter/timer or as two independent 16-bit counter/timers named L or H. Choose the setting for the UNIFY bit before accessing other registers. 0 - Dual counter. The SCT operates as two 16-bit counters named COUNTER_L and COUNTER_H. 1 - Unified counter. The SCT operates as a unified 32-bit counter.

34.7.3.1.2 SCT Control (CTRL)

If CONFIG[UNIFY] = 1, only the _L bits are used.

If CONFIG[UNIFY] = 0, the CTRL register can be written to as two registers CTRL_L and CTRL_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

All bits in CTRL can be written to when the counter is stopped or halted. When the counter is running, the only bits that can be written are STOP or HALT. (Other bits can be written in a subsequent write after HALT is set to 1.)

NOTE

If CLKMODE = 0x3 is selected, wait at least 12 system clock cycles between a write access to the H, L or unified version of this register and the next write access. This restriction does not apply when writing to the HALT bit or bits and then writing to the CTRL register again to restart the counters - for example because software must update the MATCH register, which is only allowed when the counters are halted.

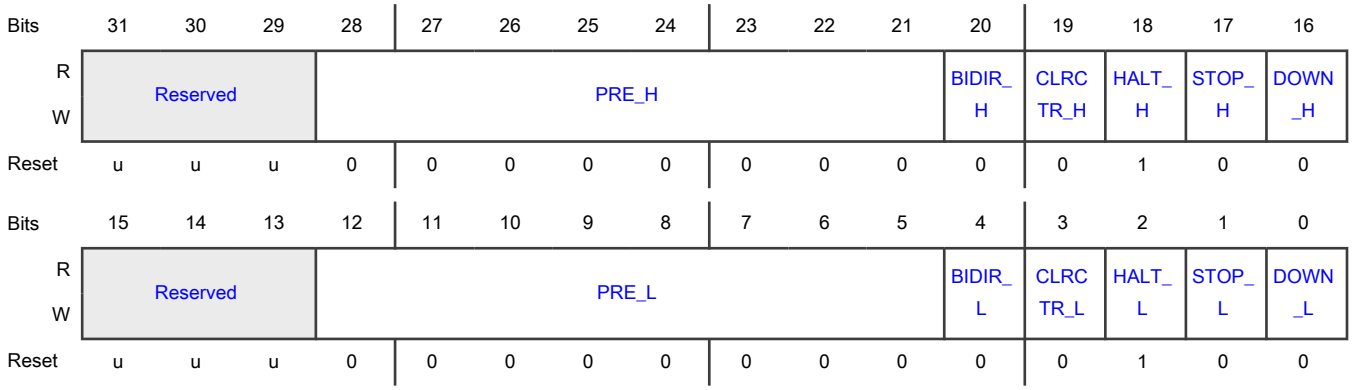
NOTE

If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Offset

Register	Offset
CTRL	4h

Diagram



Fields

Field	Description
31-29 —	Reserved
28-21 PRE_H	Prescaler for High Counter Specifies the factor by which the SCT clock is prescaled to produce the H counter clock. The counter clock is clocked at the rate of the SCT clock divided by PREL_H + 1. **NOTE** Clear the counter (by writing a 1 to the CLRCTR bit) whenever changing the PRE value.
20 BIDIR_H	Bidirectional Select High High direction select 0 - Up. The H counter counts up to its limit condition, then is cleared to zero. 1 - Up-down. The H counter counts up to its limit, then counts down to a limit condition or to 0.
19 CLRCTR_H	Clear Counter High Writing a 1 to CLRCTR_H clears the H counter. CLRCTR_H always reads as 0.
18 HALT_H	Halt Counter High When HALT_H = 1, the H counter does not run and no events can occur. When the HALT_H = 1, the STOP_H bit is cleared. It is possible to remove the halt condition while keeping the SCT in the stop condition (not running) with a single write to this register to simultaneously clear the HALT bit and set the STOP bit. **NOTE** Once set, HALT_H can only be cleared by software to restore counter operation. HALT_H = 1 on reset. 0 - Disable 1 - Enable
17	Stop Counter High

Table continues on the next page...

Table continued from the previous page...

Field	Description
STOP_H	When STOP_H = 1 and HALT = 0, the H counter does not run but I/O events related to the counter can occur. If such an event matches the mask in the Start register, STOP_H is cleared and counting resumes. 0 - Disable 1 - Enable
16 DOWN_H	Down Counter High Hardware sets DOWN_H when the counter is counting up, a counter limit condition occurs, and BIDIR = 1. Hardware clears DOWN_H when the counter is counting down and a limit condition occurs or when the counter reaches 0. 0 - Up. The H counter is counting up. 1 - Down. The H counter is counting down.
15-13 —	Reserved
12-5 PRE_L	Prescaler for Low Counter Specifies the factor by which the SCT clock is prescaled to produce the L or unified counter clock. The counter clock is clocked at the rate of the SCT clock divided by PRE_L + 1. **NOTE** Clear the counter (by writing a 1 to the CLRCTR bit) whenever changing the PRE value.
4 BIDIR_L	Bidirectional Select Low L or unified counter direction select 0 - Up. The counter counts up to a limit condition, then is cleared to zero. 1 - Up-down. The counter counts up to a limit, then counts down to a limit condition or to 0.
3 CLRCTR_L	Clear Counter Low Writing a 1 to this bit clears the L or unified counter. CLRCTR_L always reads as 0.
2 HALT_L	Halt Counter Low When HALT_L = 1, the L or unified counter does not run and no events can occur. When the HALT_L = 1, the STOP_L bit is cleared. It is possible to remove the halt condition while keeping the SCT in the stop condition (not running) with a single write to this register to simultaneously clear the HALT bit and set the STOP bit. Once set, only software can clear HALT_L to restore counter operation. HALT_L = 1 on reset. 0 - Disable 1 - Enable
1 STOP_L	Stop Counter Low When STOP_L = 1 and HALT = 0, the L or unified counter does not run, but I/O events related to the counter can occur. If a designated start event occurs, STOP_L is cleared and counting resumes.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - Disable 1 - Enable
0 DOWN_L	Down Counter Low Hardware sets DOWN_L when the counter is counting up, counter limit occurs, and BIDIR = 1. Hardware clears this bit when the counter is counting down and a limit condition occurs or when the counter reaches 0. 0 - Up. The L or unified counter is counting up. 1 - Down. The L or unified counter is counting down.

34.7.3.1.3 SCT Limit Event Select (LIMIT)

The running counter can be limited by an event. When any of the events selected in this register occur, the counter is cleared to zero from its current value or changes counting direction if in bi-directional mode. Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit causes its associated event to serve as a LIMIT event. When any limit event occurs, the counter is reset to zero in uni-directional mode or changes count direction in bi-directional mode and keeps running. To define the actual limiting event (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

NOTE

Counting up to all ones or counting down to zero is always equivalent to a limit event occurring.

In addition to using LIMIT to specify events that serve as limits, it is also possible to automatically cause a limit condition whenever a match register 0 match occurs. This eliminates the need to define an event for the sole purpose of creating a limit. The AUTOLIMITL and AUTOLIMITH bits in the configuration register enable/disable this feature.

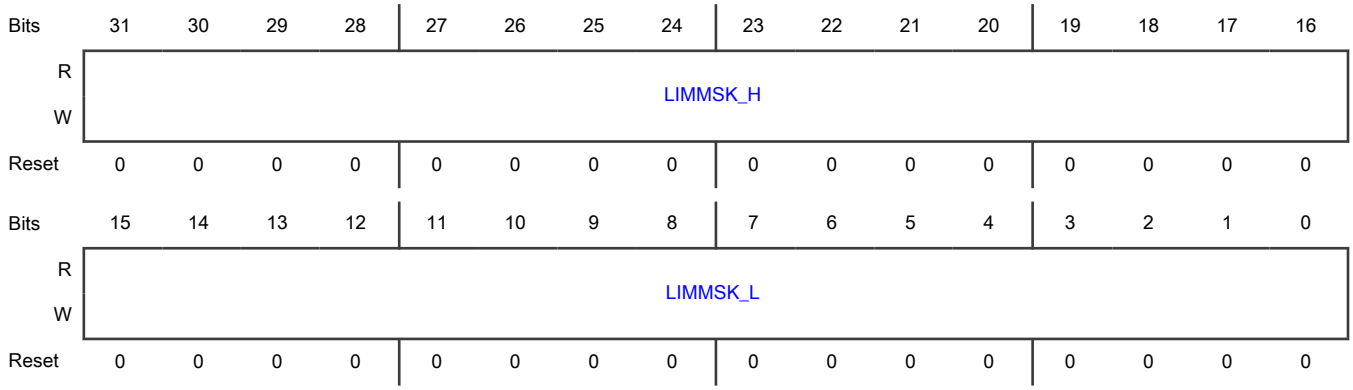
If CONFIG[UNIFY] = 1, only the _L bits are used.

If CONFIG[UNIFY] = 0, the LIMIT register can be written to as two registers LIMIT_L and LIMIT_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

Offset

Register	Offset
LIMIT	8h

Diagram



Fields

Field	Description
31-16 LIMMSK_H	Limit Event Counter High If bit n is one, event n is used as a counter limit for the H counter (event 0 = bit 16, event 1 = bit 17, etc.). The number of bits = number of events in this SCT.
15-0 LIMMSK_L	Limit Event Counter Low If bit n is one, event n is used as a counter limit for the L or unified counter (event 0 = bit 0, event 1 = bit 1, etc.). The number of bits = number of events in this SCT.

34.7.3.1.4 Halt Event Select (HALT)

The running counter can be disabled (halted) by an event. When any of the events selected in the HALT register occur, the counter stops running and all further events are disabled.

Each bit of the HALT register is associated with a different event (bit 0 with event 0, etc.). Setting a bit causes the associated event to serve as a HALT event. To define the actual events that cause the counter to halt (a match, an I/O pin toggle, etc.), see the EVn_CTRL registers.

NOTE

A HALT condition can only be removed when software clears the CTRL[HALT] bit.

If CONFIG[UNIFY] = 1, only the L bits are used.

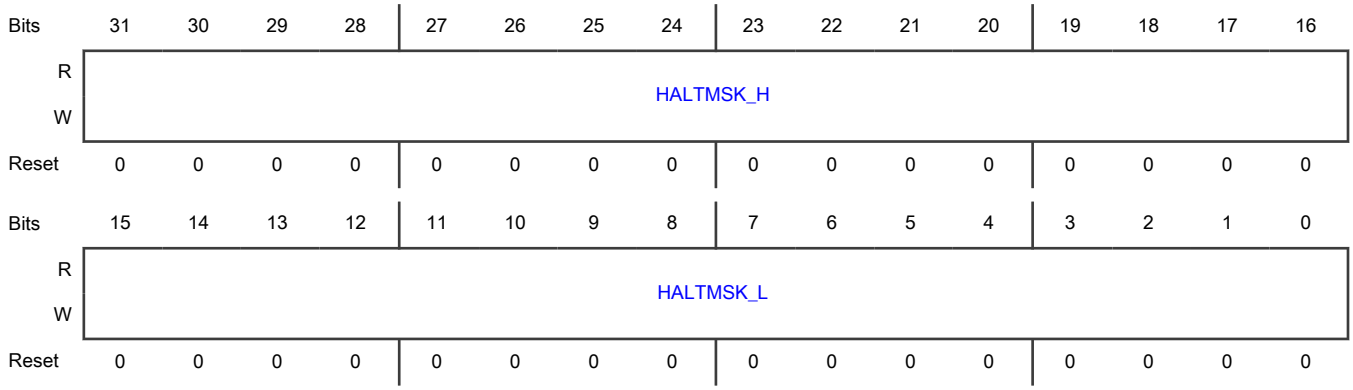
If CONFIG[UNIFY] = 0, the HALT register can be written to as two registers HALT_L and HALT_H.

Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

Offset

Register	Offset
HALT	Ch

Diagram



Fields

Field	Description
31-16 HALTMSK_H	Halt Event High If bit n is one, event n sets CTRL[HALT_H] = 1 (event 0 = bit 16, event 1 = bit 17, etc.). The number of bits = number of events in this SCT.
15-0 HALTMSK_L	Halt Event Low If bit n is one, event n sets CTRL[HALT_L] = 1 (event 0 = bit 0, event 1 = bit 1, etc.). The number of bits = number of events in this SCT.

34.7.3.1.5 Stop Event Select (STOP)

The running counter can be stopped by an event. When any of the events selected in STOP occur, counting is suspended, that is the counter stops running and remains at its current value. Event generation remains enabled, and any event selected in the START register such as an I/O event or an event generated by the other counter can restart the counter.

The STOP register specifies which events stop the counter. Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit causes its associated event to serve as a STOP event. To define the actual event that causes the counter to stop (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

NOTE

Software can stop and restart the counter by writing to the CTRL register.

If CONFIG[UNIFY] = 1, only the _L bits are used.

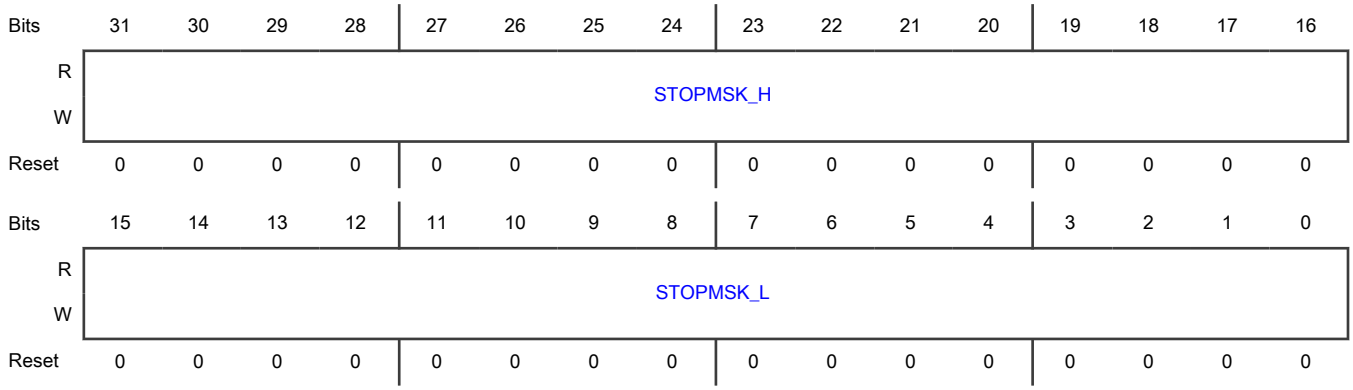
If CONFIG[UNIFY] = 0, the STOP register can be written to as two registers STOPT_L and STOP_H.

Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

Offset

Register	Offset
STOP	10h

Diagram



Fields

Field	Description
31-16 STOPMSK_H	Stop Event High If bit n is one, event n sets the CTRL[STOP_H] = 1 (event 0 = bit 16, event 1 = bit 17, etc.). The number of bits = number of events in this SCT.
15-0 STOPMSK_L	Stop Event Low If bit n is one, event n sets the CTRL[STOP_L] = 1 (event 0 = bit 0, event 1 = bit 1, etc.). The number of bits = number of events in this SCT.

34.7.3.1.6 Start Event Select (START)

The stopped counter can be re-started by an event. When any of the events selected in the START register occur, counting restarts from the current counter value.

Each bit of the START register associates with a different event (bit 0 with event 0, etc.). Setting a bit causes the associated event to serve as a START event. When any START event occurs, hardware clears the STOP bit in the Control Register CTRL. Note that a START event has no effect on the HALT bit. Only software can remove a HALT condition. To define the actual event that starts the counter (an I/O pin toggle or an event generated by the other running counter in dual-counter mode), see the EVn_CTRL register.

If CONFIG[UNIFY] = 1, only the _L bits are used.

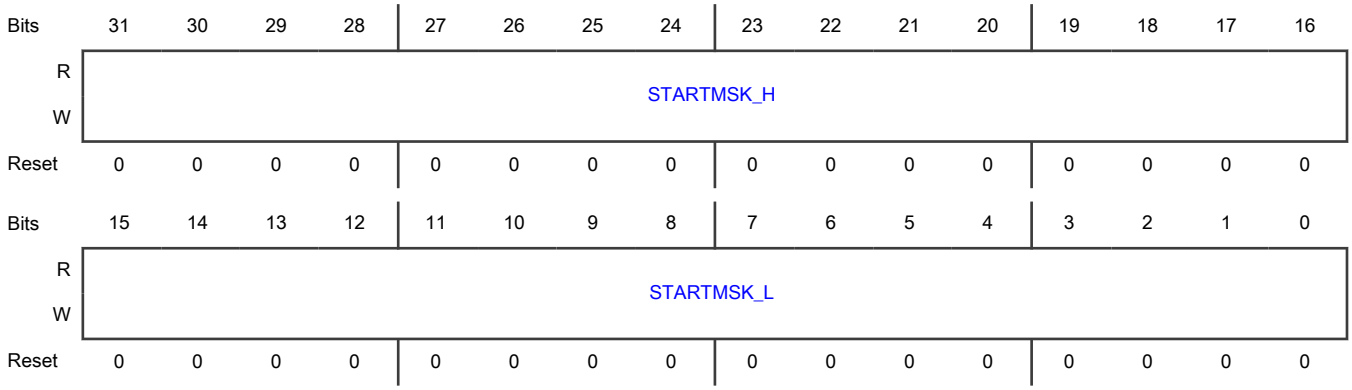
If CONFIG[UNIFY] = 0, the START register can be written to as two registers START_L and START_H.

Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

Offset

Register	Offset
START	14h

Diagram



Fields

Field	Description
31-16 STARTMSK_H	If bit n is one, event n clears the CTRL[STOP_H] = 0 (event 0 = bit 16, event 1 = bit 17, etc.). The number of bits = number of events in this SCT.
15-0 STARTMSK_L	If bit n is one, event n clears the CTRL[STOP_L] = 0 (event 0 = bit 0, event 1 = bit 1, etc.). The number of bits = number of events in this SCT.

34.7.3.1.7 Dither Condition (DITHER)

When the Dither Condition register contains all zeroes (the default value), the dither engine advances to the next count in the dither pattern every time the SCT counter reaches zero (that is, at the start of every new SCT counter cycle).

It is possible, using this register, to alter that behavior by qualifying the advancement through the dither pattern with designated events. As with the other condition/mask registers (HALT, STOP, LIMIT, etc.) each bit in this register corresponds to an event.

Setting one or more of the bits in this register to ones causes the dither engine to advance to the next element in the dither pattern (that is, increment the 16-state cycle counter) only following SCT counter cycles during which one or more of the designated dither events have occurred.

There is one global Dither Condition register per 16-bit SCT. This register controls advancement through the dither patterns for all of the match registers associated with that half of the SCT.

If CONFIG[UNIFY] = 1, only the _L bits are used.

For details on the dither engine and the dither pattern, see Dithering

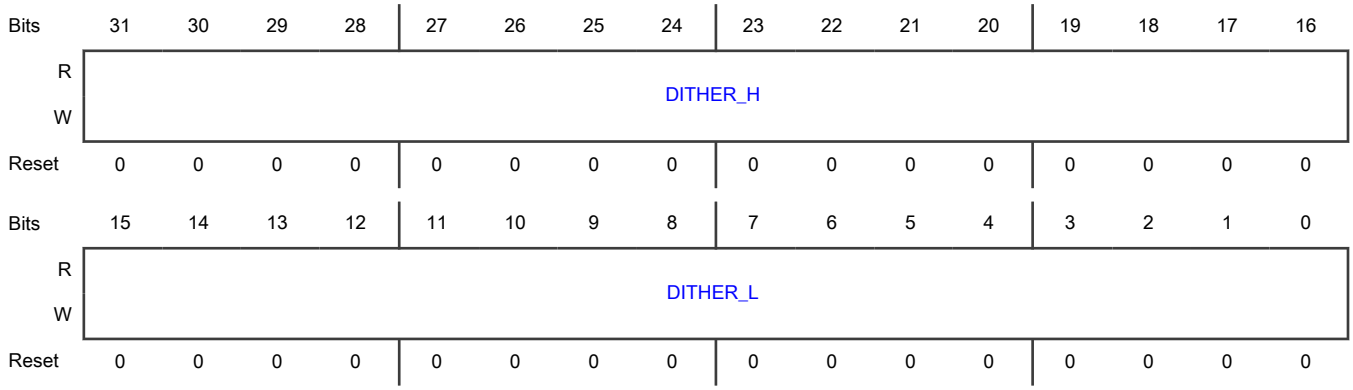
If CONFIG[UNIFY] = 0, the DITHER register can be written to as two registers START_L and START_H.

Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Offset

Register	Offset
DITHER	18h

Diagram



Fields

Field	Description
31-16 DITHER_H	If bit n is one, event n clears the CTRL[STOP_H] = 0 (event 0 = bit 16, event 1 = bit 17, etc.). The number of bits = number of events in this SCT.
15-0 DITHER_L	If bit n is one, event n clears the CTRL[STOP_L] = 0 (event 0 = bit 0, event 1 = bit 1, etc.). The number of bits = number of events in this SCT.

34.7.3.1.8 Counter (COUNT)

If CONFIG[UNIFY] = 1, the counter is a unified 32-bit register and both the _L and _H bits are used.

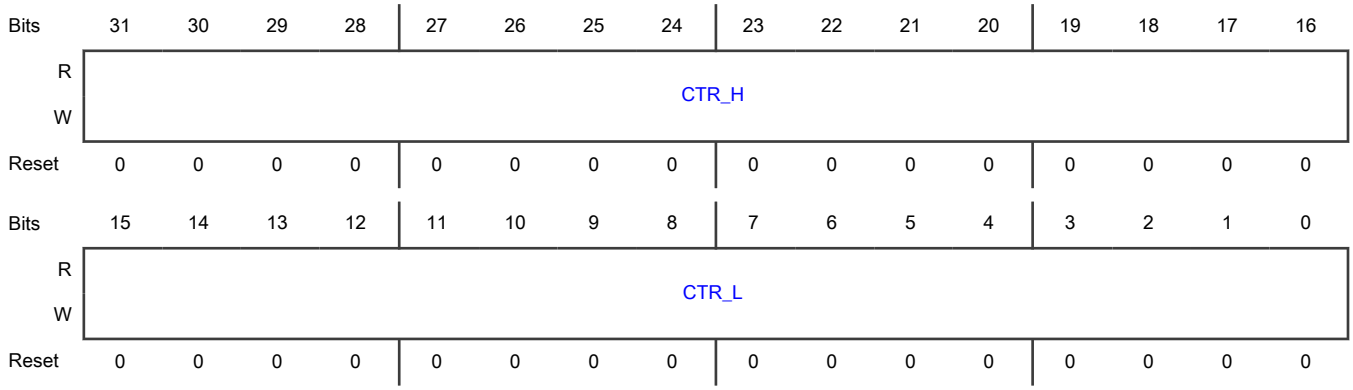
If CONFIG[UNIFY] = 0, the COUNT register can be written to as two registers COUNT_L and COUNT_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register. In this case, the L and H registers count independently under the control of the other registers.

Writing to the COUNT_L, COUNT_H, or unified register is only allowed when the corresponding counter is halted (CTRL[HALT] = 1). Attempting to write to the counter when it is not halted causes a bus error. Software can read the counter registers at any time.

Offset

Register	Offset
COUNT	40h

Diagram



Fields

Field	Description
31-16 CTR_H	Counter High When CONFIG[UNIFY] = 0, read or write the 16-bit H counter value. When CONFIG[UNIFY] = 1, read or write the upper 16 bits of the 32-bit unified counter.
15-0 CTR_L	Counter Low When CONFIG[UNIFY] = 0, read or write the 16-bit L counter value. When CONFIG[UNIFY] = 1, read or write the lower 16 bits of the 32-bit unified counter.

34.7.3.1.9 State (STATE)

Each group of enabled and disabled events is assigned a number called the state variable. For example, a state variable with a value of 0 could have events 0, 2, and 3 enabled and all other events disabled. A state variable with the value of 1 could have events 1, 4, and 5 enabled and all others disabled.

NOTE

The EVm_STATE registers define which event is enabled in each group.

Software can read the state associated with a counter at any time. Writing to the STATE_L or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register). STATE_H can only be written as a word along with STATE_L, and both counters must be halted.

The state variable is the main feature that distinguishes the SCTimer/PWM from other counter/timer/PWM blocks. Events can be made to occur only in certain states. Events, in turn, can perform the following actions:

- Set and clear outputs
- Limit, stop, and start the counter
- Cause interrupts and DMA requests
- Modify the state variable

The value of a state variable is completely under the control of the application. If an application does not use states, the value of the state variable remains zero, which is the default value.

A state variable can be used to track and control multiple cycles of the associated counter in any desired operational sequence.

The STATELD/STADEV fields in the event control registers of all defined events set all possible values for the state variable. The change of the state variable during multiple counter cycles reflects how the associated state machine moves from one state to the next.

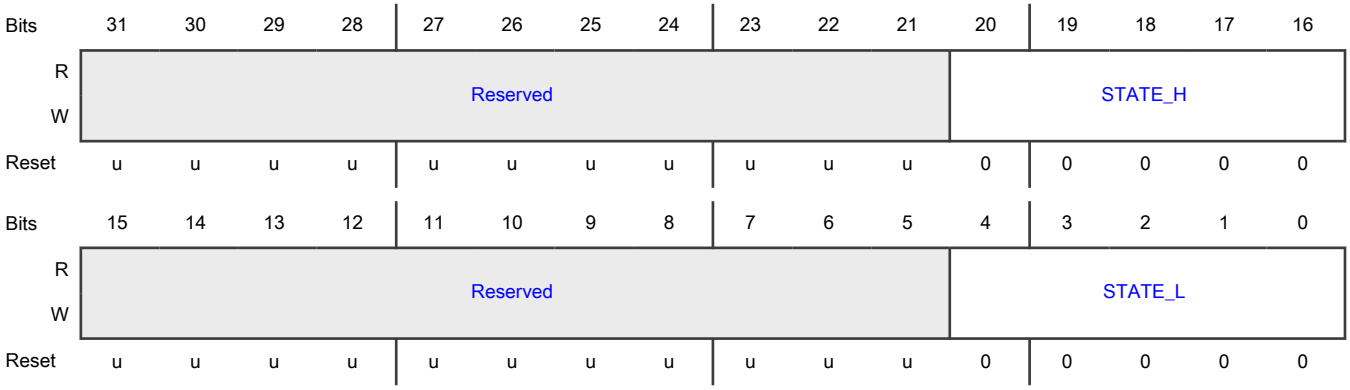
If CONFIG[UNIFY] = 1, only the _L bits are used.

If CONFIG[UNIFY] = 0, the STATE register can be written to as two registers STATE_L and STATE_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

Offset

Register	Offset
STATE	44h

Diagram



Fields

Field	Description
31-21 —	Reserved
20-16 STATE_H	State variable
15-5 —	Reserved
4-0 STATE_L	State variable

34.7.3.1.10 Input (INPUT)

Software can read the state of the SCT inputs in this read-only register in slightly different forms.

1. The AIN bit displays the state of the input captured on each rising edge of the SCT clock. This corresponds to a nearly direct read-out of the input but can cause spurious fluctuations in case of an asynchronous input signal.
2. The SIN bit displays the form of the input as it is used for event detection. This may include additional stages of synchronization, depending on what is specified for that input in the CONFIG[INSYNC] field:

- If CONFIG[INSYNC] = 1, the input is triple-synchronized to the SCT clock resulting in a stable signal that is delayed by three SCT clock cycles.
- If CONFIG[INSYNC] = 0, the SIN bit value is identical to the AIN bit value.

Offset

Register	Offset
INPUT	48h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SIN15	SIN14	SIN13	SIN12	SIN11	SIN10	SIN9	SIN8	SIN7	SIN6	SIN5	SIN4	SIN3	SIN2	SIN1	SIN0
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	AIN15	AIN14	AIN13	AIN12	AIN11	AIN10	AIN9	AIN8	AIN7	AIN6	AIN5	AIN4	AIN3	AIN2	AIN1	AIN0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31 SIN15	Input 15 state. Input 15 state following the synchronization specified by INSYNC.
30 SIN14	Input 14 state. Input 14 state following the synchronization specified by INSYNC.
29 SIN13	Input 13 state. Input 13 state following the synchronization specified by INSYNC.
28 SIN12	Input 12 state. Input 12 state following the synchronization specified by INSYNC.
27 SIN11	Input 11 state. Input 11 state following the synchronization specified by INSYNC.
26 SIN10	Input 10 state. Input 10 state following the synchronization specified by INSYNC.
25	Input 9 state. Input 9 state following the synchronization specified by INSYNC.

Table continues on the next page...

Table continued from the previous page...

Field	Description
SIN9	
24 SIN8	Input 8 state. Input 8 state following the synchronization specified by INSYNC.
23 SIN7	Input 7 state. Input 7 state following the synchronization specified by INSYNC.
22 SIN6	Input 6 state. Input 6 state following the synchronization specified by INSYNC.
21 SIN5	Input 5 state. Input 5 state following the synchronization specified by INSYNC.
20 SIN4	Input 4 state. Input 4 state following the synchronization specified by INSYNC.
19 SIN3	Input 3 state. Input 3 state following the synchronization specified by INSYNC.
18 SIN2	Input 2 state. Input 2 state following the synchronization specified by INSYNC.
17 SIN1	Input 1 state. Input 1 state following the synchronization specified by INSYNC.
16 SIN0	Input 0 state. Input 0 state following the synchronization specified by INSYNC.
15 AIN15	Input 15 state. Input 15 state on the last SCT clock edge.
14 AIN14	Input 14 state. Input 14 state on the last SCT clock edge.
13 AIN13	Input 13 state. Input 13 state on the last SCT clock edge.
12 AIN12	Input 12 state. Input 12 state on the last SCT clock edge.
11 AIN11	Input 11 state. Input 11 state on the last SCT clock edge.

Table continues on the next page...

Table continued from the previous page...

Field	Description
10 AIN10	Input 10 state. Input 10 state on the last SCT clock edge.
9 AIN9	Input 9 state. Input 9 state on the last SCT clock edge.
8 AIN8	Input 8 state. Input 8 state on the last SCT clock edge.
7 AIN7	Input 7 state. Input 7 state on the last SCT clock edge.
6 AIN6	Input 6 state. Input 6 state on the last SCT clock edge.
5 AIN5	Input 5 state. Input 5 state on the last SCT clock edge.
4 AIN4	Input 4 state. Input 4 state on the last SCT clock edge.
3 AIN3	Input 3 state. Input 3 state on the last SCT clock edge.
2 AIN2	Input 2 state. Input 2 state on the last SCT clock edge.
1 AIN1	Input 1 state. Input 1 state on the last SCT clock edge.
0 AIN0	Input 0 state. Input 0 state on the last SCT clock edge.

34.7.3.1.11 Match/Capture Mode (REGMODE)

If CONFIG[UNIFY] = 1, only the _L bits of the REGMODE register are used. In this case, REGMODE_H is not used.

If CONFIG[UNIFY] = 0, the REGMODE register can be written to as two registers REGMODE_L and REGMODE_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register. The _L bits/registers control the L Match/Capture registers, and the _H bits/registers control the H Match/Capture registers.

The SCT contains multiple Match/Capture registers. The Register Mode register selects whether each register acts as a Match register or as a Capture register. Each Match/Capture register has an accompanying register which functions as a Reload register when the primary register is used as a Match register or as a Capture-Control register when the register is used as a Capture register. REGMODE_H is used only when CONFIG[UNIFY] = 0.

Offset

Register	Offset
REGMODE	4Ch

Diagram

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

Fields

Field	Description
31-16 REGMOD_Hn	Register Mode High n REGMOD_H*n* bit controls Match/Capture register n. 0 - Match. Register n operates as a match register 1 - Capture. Register n operates as a capture register
15-0 REGMOD_Ln	Register Mode Low n REGMOD_L*n* bit controls Match/Capture register n. 0 - Match. Register n operates as a match register 1 - Capture. Register n operates as a capture register

34.7.3.1.12 Output (OUTPUT)

Each SCT output has a corresponding bit in the OUTPUT register to allow software to control the output state directly or read its current state.

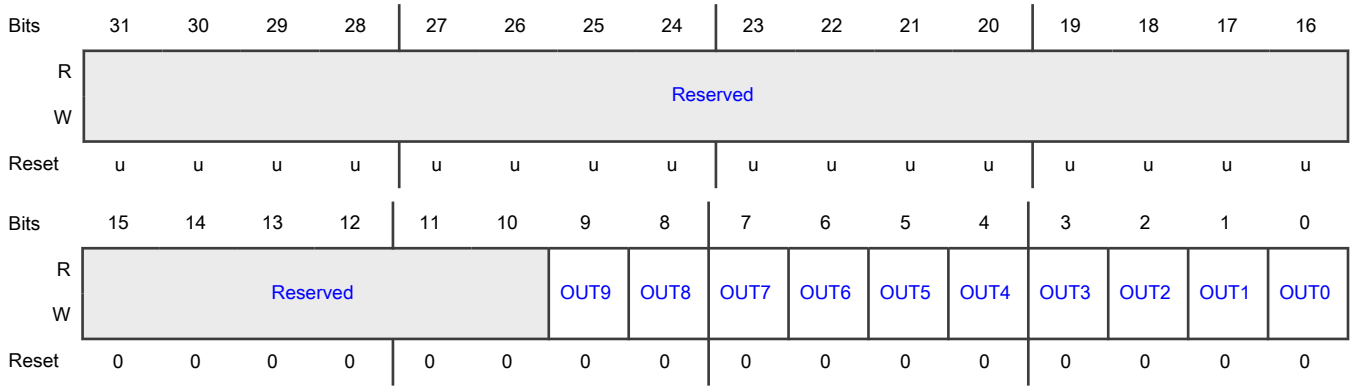
While the counter is running, outputs are set, cleared, or toggled only by events. However, using the OUTPUT register, software can write to any of the output registers when both counters are halted to control the outputs directly. Writing to the OUT register is only allowed when all counters (L-counter, H-counter, or unified counter) are halted (CTRL[HALT] = 1).

Software can read this register at any time to sense the state of the outputs.

Offset

Register	Offset
OUTPUT	50h

Diagram



Fields

Field	Description
31-16 —	Reserved
15-10 —	Reserved
9-0 OUTn	Output n The number of bits equals the number of outputs in this SCT. 0 - Writing a 0 forces the corresponding output low 1 - Writing a 1 forces the corresponding output high

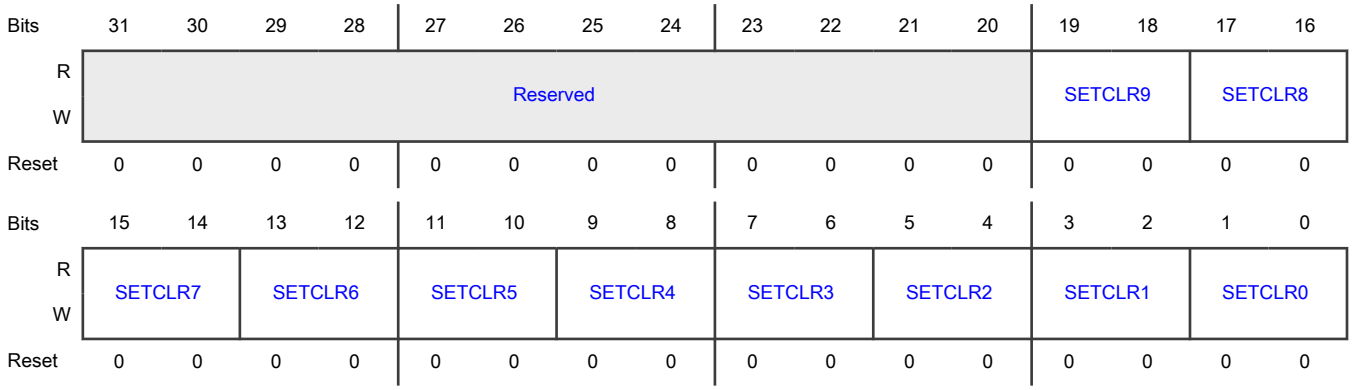
34.7.3.1.13 Output Counter Direction Control (OUTPUTDIRCTRL)

For bi-directional mode, the OUTPUTDIRCTRL register specifies (for each output) the impact of the counting direction on the meaning of set and clear operations on the output. The purpose of the OUTPUTDIRCTRL register is to facilitate the creation of center-aligned output waveforms without the need to define additional events.

Offset

Register	Offset
OUTPUTDIRCTRL	54h

Diagram



Fields

Field	Description
31-20 —	Reserved
19-18: SETCLR9 17-16: SETCLR8 15-14: SETCLR7 13-12: SETCLR6 11-10: SETCLR5 9-8: SETCLR4 7-6: SETCLR3 5-4: SETCLR2 3-2: SETCLR1 1-0: SETCLR0	Set/Clear Operation on Output n 00 - Set and clear do not depend on the direction of any counter. 01 - Set and clear are reversed when counter L or the unified counter is counting down. 10 - Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1. 11 - Reserved. Do not program this value.

34.7.3.1.14 Output Conflict Resolution (RES)

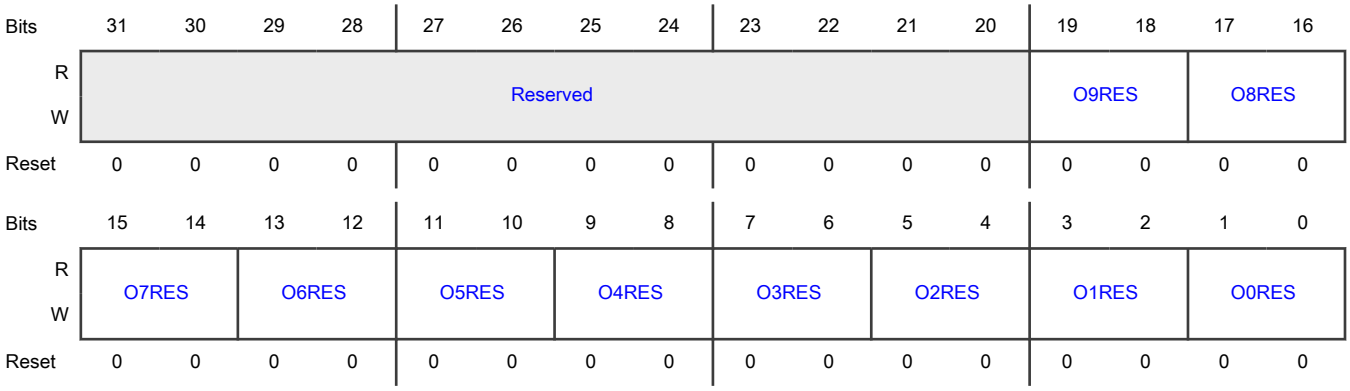
The Output Conflict Resolution register specifies what action should be taken if multiple events (or even the same event) dictate that a given output should be both set and cleared at the same time.

To enable an event to toggle an output each time the event occurs, set the bits for that event in both the OUTn_SET and OUTn_CLR registers and set the On_RES value to 0x3 in this register.

Offset

Register	Offset
RES	58h

Diagram



Fields

Field	Description
31-20 —	Reserved
19-18: O9RES 17-16: O8RES 15-14: O7RES 13-12: O6RES 11-10: O5RES 9-8: O4RES 7-6: O3RES 5-4: O2RES 3-2: O1RES 1-0: O0RES	Effect of simultaneous set and clear on output n 00 - No change 01 - Set output (or clear based on the OUTPUTDIRCTRL[SETCLRn] field) 10 - Clear output (or set based on the OUTPUTDIRCTRL[SETCLRn] field) 11 - Toggle output

34.7.3.1.15 DMA Request 0 (DMAREQ0)

The SCT includes two DMA request outputs. These registers enable the DMA requests to be generated when a particular event occurs or when Counter Match registers are loaded from the Reload registers.

The DMA Request registers are word-write only. Attempting to write a half-word value to these registers results in a bus error.

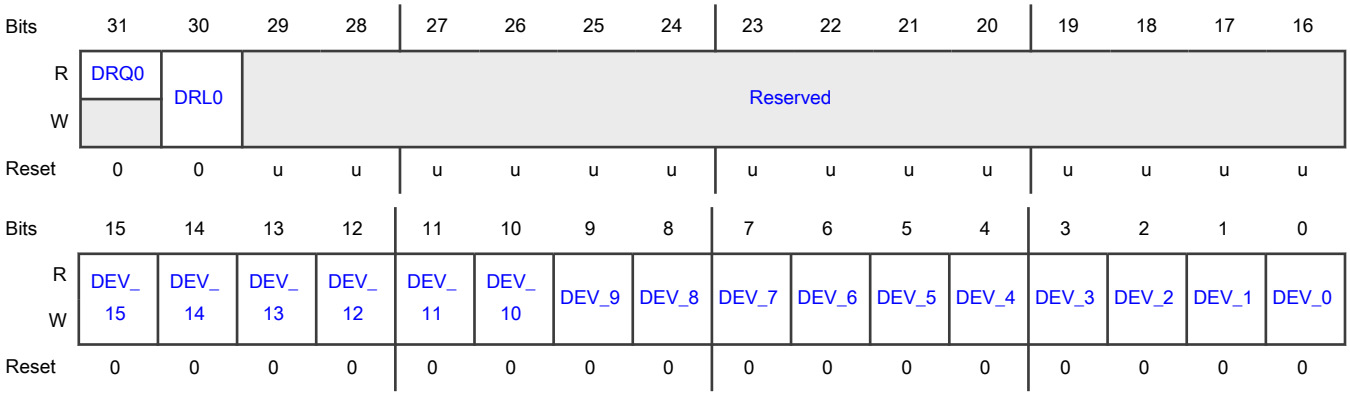
Note that on this device, SCT DMA requests are connected to the DMA trigger inputs, and controlled by the DMA trigger input mux registers.

Event-generated DMA requests are particularly useful for launching DMA activity to or from other peripherals under the control of the SCT.

Offset

Register	Offset
DMAREQ0	5Ch

Diagram



Fields

Field	Description
31 DRQ0	DMA Request 0 State This read-only bit indicates the state of DMA Request 0. Note that if the related DMA channel is enabled and properly set up, it is unlikely that software will see this flag. DRQ0 will be cleared rapidly by the DMA service. If the flag remains set there could be an issue with DMA setup.
30 DRL0	A 1 in this bit triggers DMA request 0 when it loads the MATCH_L/Unified registers from the RELOAD_L/ Unified registers.
29-16 —	Reserved
15-0 DEV_n	DMA Request Event n If bit n is one, event n triggers DMA request 0 (event 0 = bit 0, event 1 = bit 1, etc.). The number of bits = number of events in this SCT.

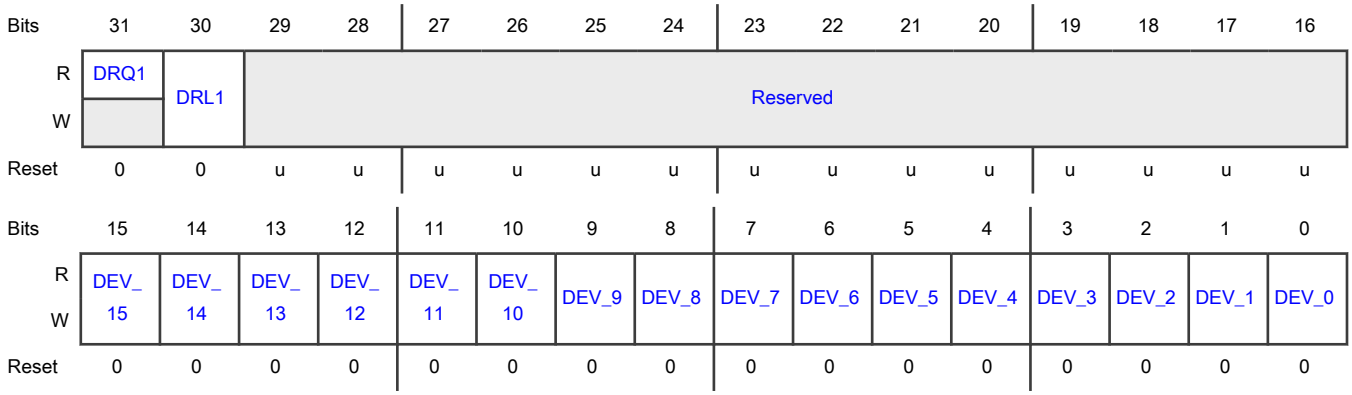
34.7.3.1.16 DMA Request 1 (DMAREQ1)

This is the DMA request register 1.

Offset

Register	Offset
DMAREQ1	60h

Diagram



Fields

Field	Description
31 DRQ1	DMA Request 1 State This read-only bit indicates the state of DMA Request 1. Note that if the related DMA channel is enabled and properly set up, it is unlikely that software will see this flag. DRQ1 will be cleared rapidly by the DMA service. If the flag remains set there could be an issue with DMA setup.
30 DRL1	A 1 in this bit triggers DMA request 1 when it loads the Match L/Unified registers from the Reload L/Unified registers.
29-16 —	Reserved
15-0 DEV_n	DMA Request Event n If bit n is one, event n triggers DMA request 1 (event 0 = bit 0, event 1 = bit 1, etc.). The number of bits = number of events in this SCT.

34.7.3.1.17 Event Interrupt Enable (EVEN)

The EVEN register enables flags to request an interrupt if the FLAGn bit in the SCT Event Flag register is also set.

Offset

Register	Offset
EVEN	F0h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
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 W	IEN15	IEN14	IEN13	IEN12	IEN11	IEN10	IEN9	IEN8	IEN7	IEN6	IEN5	IEN4	IEN3	IEN2	IEN1	IEN0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-16 —	Reserved
15-0 IENn	Event Interrupt Enable n 0 - Disable 1 - Enable

34.7.3.1.18 Event Flag (EVFLAG)

The EVFLAG register records events. Writing ones to this register clears the corresponding flags and negates the SCT interrupt request if all of the enabled flag register bits are zero.

Offset

Register	Offset
EVFLAG	F4h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
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 W	FLAG15	FLAG14	FLAG13	FLAG12	FLAG11	FLAG10	FLAG9	FLAG8	FLAG7	FLAG6	FLAG5	FLAG4	FLAG3	FLAG2	FLAG1	FLAG0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-16 —	Reserved
15-0 FLAGn	Event Flag n Bit n is one if event n has occurred since reset or a 1 was last written to this bit (event 0 = bit 0, event 1 = bit 1, etc.). The number of bits = number of events in this SCT. 0 - No Flag 1 - Event n Flag

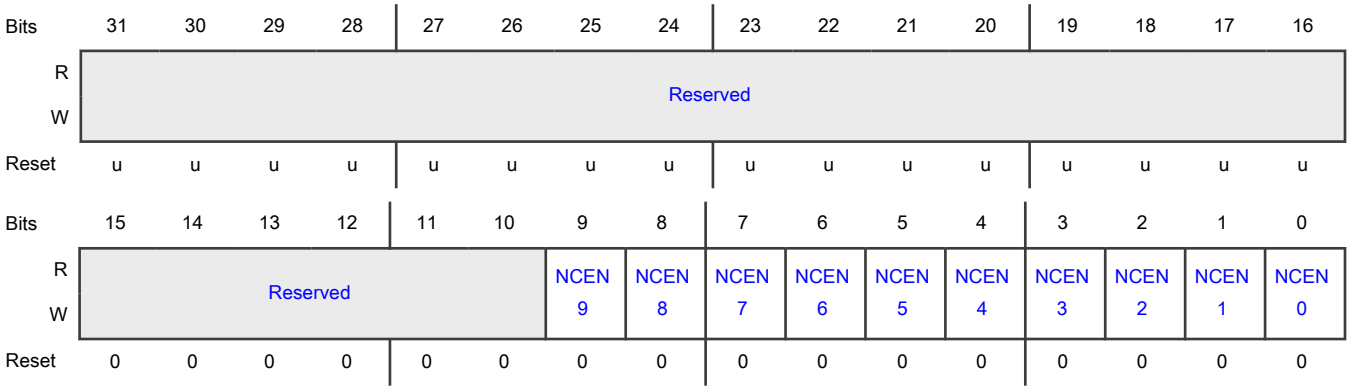
34.7.3.1.19 Conflict Interrupt Enable (CONEN)

The CONEN register enables the no-change conflict events specified in the SCT Conflict Resolution register to generate an interrupt request.

Offset

Register	Offset
CONEN	F8h

Diagram



Fields

Field	Description
31-10 —	Reserved
9-0 NCENn	No Change Conflict Event/Interrupt Enable The SCT requests an interrupt when bit n of this register and the SCT conflict flag register are both one (output 0 = bit 0, output 1 = bit 1, etc.).

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - No interrupt 1 - Interrupt

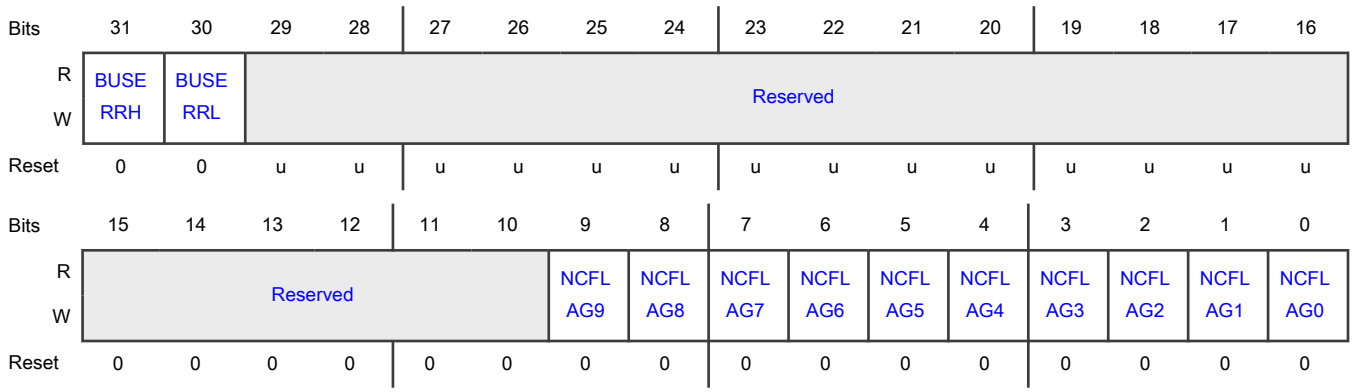
34.7.3.1.20 Conflict Flag (CONFLAG)

The CONFLAG register records a no-change conflict occurrence and also provides details of any bus errors. Writing ones to the NCFLAG bits clears the corresponding read bits and negates the SCT interrupt request if all enabled Flag bits are zero.

Offset

Register	Offset
CONFLAG	FCh

Diagram



Fields

Field	Description
31 BUSERRH	Bus Error High The most recent bus error from this SCT involved writing CTR H, STATE H, MATCH H, or the Output register when the H counter was not halted.
30 BUSERRL	Bus Error Low/Unified The most recent bus error from this SCT involved writing CTR L/Unified, STATE L/Unified, MATCH L/Unified, or the Output register when the L/U counter was not halted. A word write to certain L and H registers can be half successful and half unsuccessful.
29-10 —	Reserved
9-0	No Change Conflict Event Flag

Table continues on the next page...

Table continued from the previous page...

Field	Description
NCFLAGn	0 - No Conflict Event 1 - A No Change Conflict Event occured

34.7.3.1.21 Capture Value (CAP0 - CAP15)

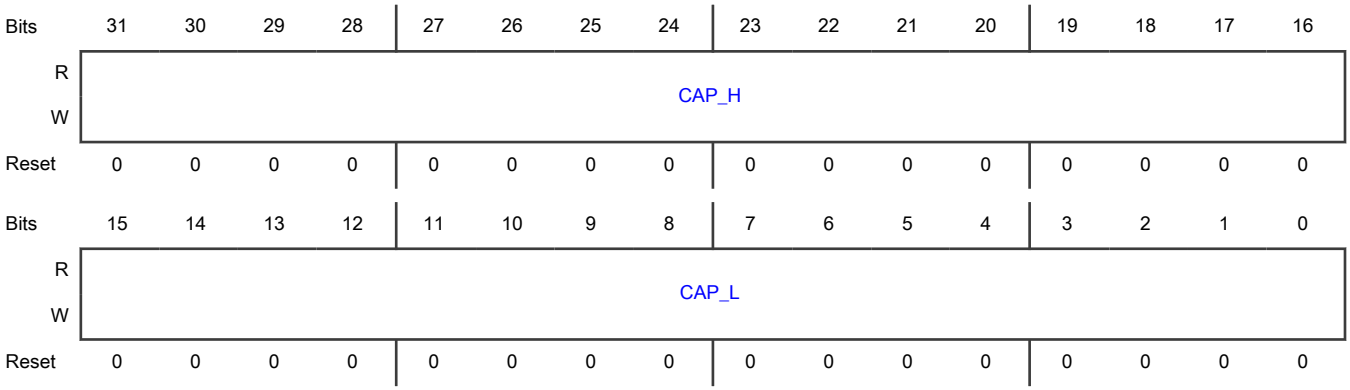
The CAPn registers allow software to record the counter values when the events selected by the corresponding Capture Control registers occur.

Offset

For n = 0 to 15:

Register	Offset
CAPn	100h + (n × 4h)

Diagram



Fields

Field	Description
31-16 CAP_H	Capture High When UNIFY = 0, read the 16-bit counter value at which this register was last captured. When UNIFY = 1, read the upper 16 bits of the 32-bit value at which this register was last captured.
15-0 CAP_L	Capture Low When UNIFY = 0, read the 16-bit counter value at which this register was last captured. When UNIFY = 1, read the lower 16 bits of the 32-bit value where this register was last captured.

34.7.3.1.22 Match Value (MATCH0 - MATCH15)

Match registers are compared to the counters to help create events. When the UNIFY bit is 0, the L and H registers are independently compared to the L and H counters. When UNIFY is 1, the combined L and H registers hold a 32-bit value that is compared to the unified counter. A Match can only occur in a clock in which the counter is running (STOP and HALT are both 0).

Match registers can be read at any time. Writing to the MATCH_L, or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register). MATCH_H can only be written as a word along with MATCH_L, and both counters must be halted. Match events occur in the SCT clock in which the counter is (or would be) incremented to the next value. When a match event limits its counter, the value in the Match register is the last value of the counter before it is cleared to zero (or decremented if BIDIR is 1).

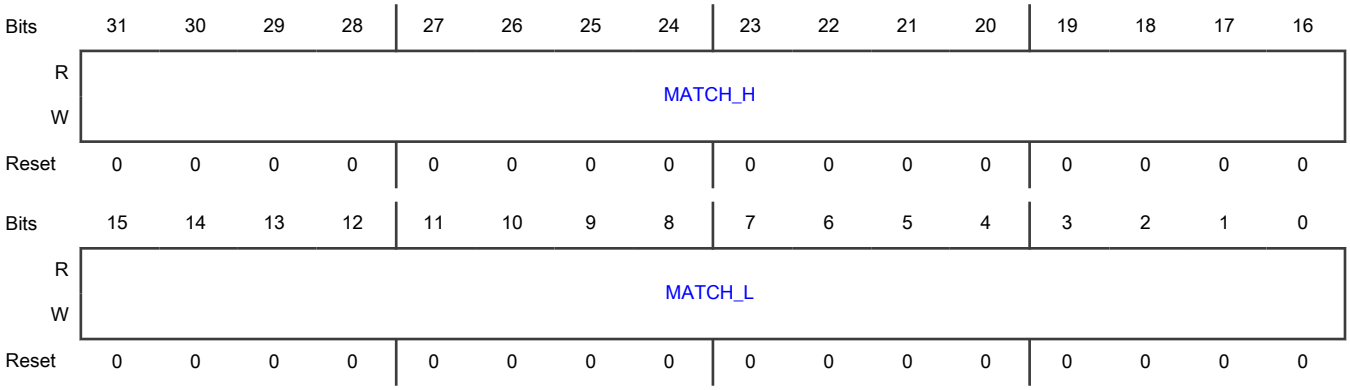
There is no “write-through” from Reload registers to Match registers. Before starting a counter, software can write one value to the Match register used in the first cycle of the counter and a different value to the corresponding Match Reload register used in the second cycle.

Offset

For n = 0 to 15:

Register	Offset
MATCHn	100h + (n × 4h)

Diagram



Fields

Field	Description
31-16 MATCH_H	Match High When UNIFY = 0, read or write the 16-bit value to be compared to the H counter. When UNIFY = 1, read or write the upper 16 bits of the 32-bit value to be compared to the unified counter.
15-0 MATCH_L	Match Low When UNIFY = 0, read or write the 16-bit value to be compared to the L counter. When UNIFY = 1, read or write the lower 16 bits of the 32-bit value to be compared to the unified counter.

34.7.3.1.23 Fractional Match (FRACMAT0 - FRACMAT5)

Fractional Match registers are provided for up to the first six of the match registers. The values programmed in these registers provide higher average resolution over time by applying a dither pattern as described in Dithering. This dither pattern results in delaying recognition of a match for one counter clock for n (0 to 15) out of every 16 counter cycles. The value of n is programmed in these Fractional Match registers.

Fractional Match registers can be read at any time. Writing to the FRACMAT_L, FRACMAT_H, or unified register is only allowed when the corresponding counter is halted (CTRL[HALT] bits are set to 1).

Each Fractional Match register has a Fractional Match Reload register associated with it. The contents of the reload registers are transferred into the Fractional Match registers at the start of every new SCT counter cycle unless the CONFIG[NORELOAD] bit for the appropriate half-counter is set.

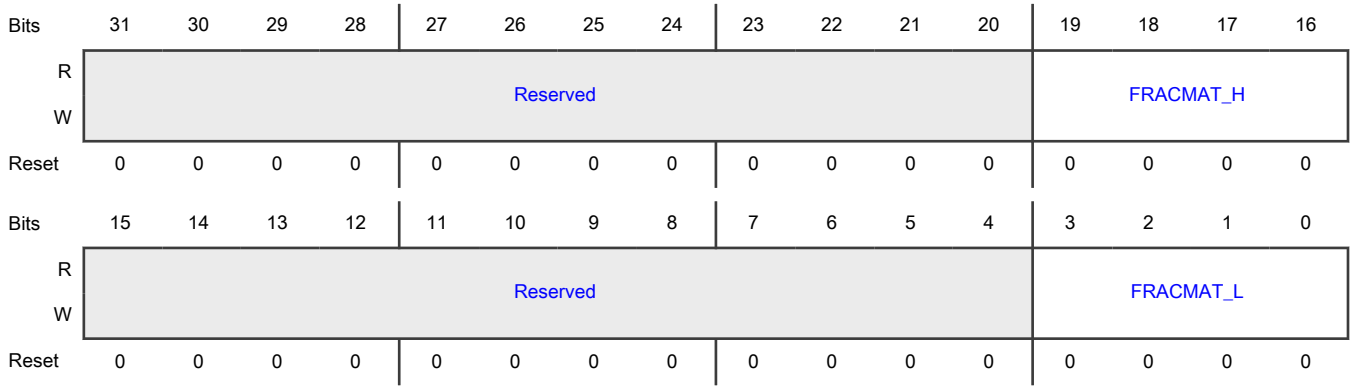
The reload registers may be written to at any time, regardless of whether or not the counter is running.

There is no write-through from the Fractional Match Reload registers to the Fractional Match registers. Before starting a counter, software can write one value to the Fractional Match register that will be used in the first cycle or period of operation, and a different value to the corresponding Fractional Match Reload register that will be used in the second cycle or period.

Offset

Register	Offset
FRACMAT0	140h
FRACMAT1	144h
FRACMAT2	148h
FRACMAT3	14Ch
FRACMAT4	150h
FRACMAT5	154h

Diagram



Fields

Field	Description
31-20 —	Reserved
19-16 FRACMAT_H	Fractional Match High When CONFIG[UNIFY] = 0, read or write 4-bit value specifying the dither pattern to be applied to the corresponding MATCHn_H register (n = 0 to 5).
15-4 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
3-0 FRACMAT_L	Fractional Match Low When CONFIG[UNIFY] = 0, read or write the 4-bit value specifying the dither pattern to be applied to the corresponding MATCHn_L register (n = 0 to 5). When CONFIG[UNIFY] = 1, the value applies to the unified, 32-bit fractional match register.

34.7.3.1.24 Capture Control (CAPCTRL0 - CAPCTRL15)

If CONFIG[UNIFY] = 1, only the _L bits are used.

If CONFIG[UNIFY] = 0, this register can be written to as two registers CAPCTRLn_L and CAPCTRLn_H. Both the L and H registers can be read individually. The L registers can also be written individually, but the H register must be written as a word along with the L register.

Based on a selected event, the Capture registers can be loaded with the current counter value when the event occurs.

Each Capture Control register (L, H, or unified 32-bit) controls which events cause the load of corresponding Capture register from the counter.

Offset

For n = 0 to 15:

Register	Offset
CAPCTRLn	200h + (n × 4h)

Diagram

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

Fields

Field	Description
31-16 CAPCON_H	Capture Control High If bit m is one, event m causes the CAPn_H (UNIFY = 0) register to be loaded (event 0 = bit 16, event 1 = bit 17, etc.). The number of bits = number of match/captures in this SCT.

Table continues on the next page...

Table continued from the previous page...

Field	Description
15-0 CAPCON_L	Capture Control Low If bit m is one, event m causes the CAPn_L (UNIFY = 0) or the CAPn (UNIFY = 1) register to be loaded (event 0 = bit 0, event 1 = bit 1, etc.). The number of bits = number of match/captures in this SCT.

34.7.3.1.25 Match Reload Value (MATCHREL0 - MATCHREL15)

A Match register (L, H, or unified 32-bit) is loaded from its corresponding Reload register at the start of each new counter cycle, that is

- When CTRL[BIDIR] = 0 and the counter is cleared to zero upon reaching it limit condition.

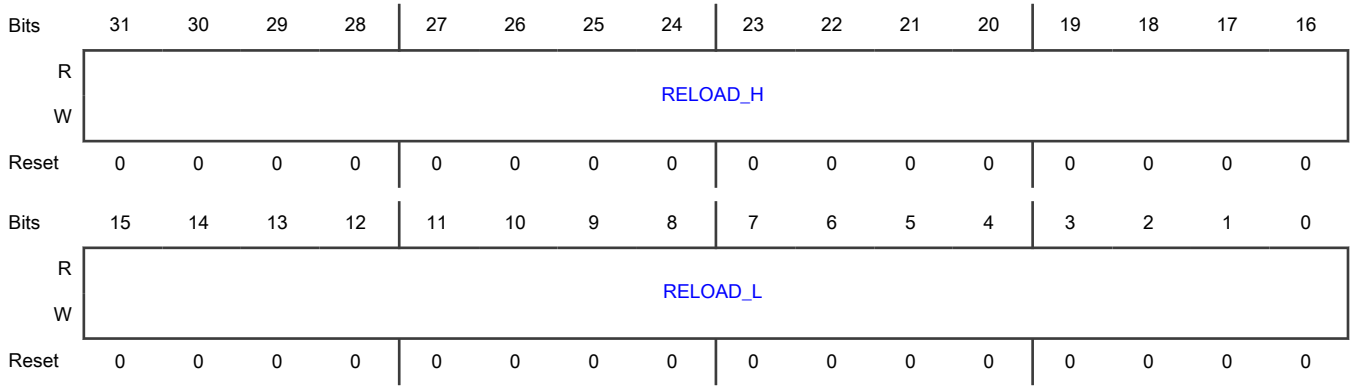
When CTRL[BIDIR] = 1 and the counter counts down to 0, unless CFG[NORELOAD] = 1.

Offset

For n = 0 to 15:

Register	Offset
MATCHRELn	200h + (n × 4h)

Diagram



Fields

Field	Description
31-16 RELOAD_H	Reload High When CONFIG[UNIFY] = 0, specifies the 16-bit to be loaded into the MATCHn_H register. When CONFIG[UNIFY] = 1, specifies the upper 16 bits of the 32-bit value to be loaded into the MATCHn register.
15-0 RELOAD_L	Reload Low When CONFIG[UNIFY] = 0, specifies the 16-bit value to be loaded into the MATCHn_L register. When CONFIG[UNIFY] = 1, specifies the lower 16 bits of the 32-bit value to be loaded into the MATCHn register.

34.7.3.1.26 Fractional Match Reload (FRACMATREL0 - FRACMATREL5)

A Fractional Match register (L, H, or unified 32-bit) is loaded from the corresponding Fractional Match Reload register at the start of each new counter cycle - that is, when CTRL[BIDIR]= 0 and the counter is cleared to zero upon reaching its limit condition, or when CTRL[BIDIR] = 1 and the counter counts down to 0, unless the appropriate CONFIG[NORELOAD] bit is set.

Offset

Register	Offset
FRACMATREL0	240h
FRACMATREL1	244h
FRACMATREL2	248h
FRACMATREL3	24Ch
FRACMATREL4	250h
FRACMATREL5	254h

Diagram

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

Fields

Field	Description
31-20 —	Reserved
19-16 RELFRAC_H	Reload Fractional Match High When CONFIG[UNIFY] = 0, read or write the 4-bit value to be loaded into the FRACMATn_H register. When CONFIG[UNIFY] = 1, read or write the upper 4 bits with the 4-bit value to be loaded into the FRACMATn register.
15-4 —	Reserved
3-0 FRACMAT_L	Reload Fractional Match Low When CONFIG[UNIFY] = 0, read or write the 4-bit value to be loaded into the FRACMATn_L register. When CONFIG[UNIFY] = 1, read or write the lower 4 bits to be loaded into the FRACMATn register.

34.7.3.1.27 Event n State (EV0_STATE - EV15_STATE)

Each event can be enabled in some contexts (or states) and disabled in others. Each event defined in the EV_CTRLn register has one associated Event Enable register that can enable or disable the event for each available state.

Each event has one associated SCT Event State Mask register that allow this event to happen in one or more states of the counter selected by the HEVENT bit in the corresponding EVn_CTRL register.

An event n is disabled when its EVn_STATE register contains all zeros, since it is masked regardless of the current state.

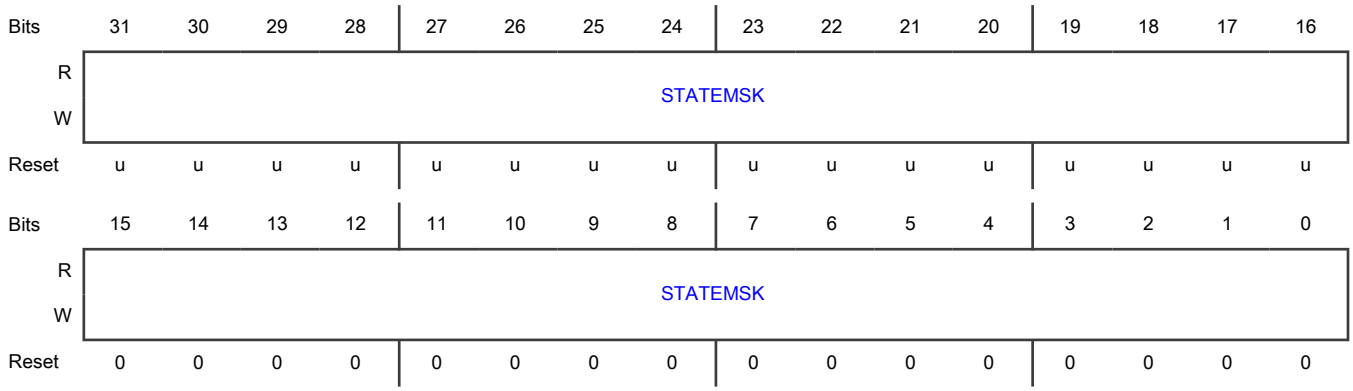
In simple applications that do not use states, write 0x01 to this register to enable each event in exactly one state. Since the state does not change (that is, the state variable always remains at its reset value of 0), writing 0x01 permanently enables this event.

Offset

For n = 0 to 15:

Register	Offset
EVn_STATE	300h + (n × 8h)

Diagram



Fields

Field	Description
31-0 STATEMSK	Event State Mask If bit m is one, event n happens in state m of the counter selected by the HEVENT bit (n = event number, m = state number; state 0 = bit 0, state 1= bit 1, etc.). The number of bits = number of states in this SCT.

34.7.3.1.28 Event n Control (EV0_CTRL - EV15_CTRL)

This register defines the conditions for an event to occur based on the counter values or input and output states. Once the event is configured, it can be selected to trigger multiple actions (for example stop the counter and toggle an output) unless the event is blocked in the current state of the SCT or the counter is halted. To block a particular event from occurring, use the EV_STATEn register. To block all events for a given counter, set the CTRL[HALT] bit equal to one or select an event to halt the counter.

An event can be programmed to occur based on a selected input or output edge or level and/or based on its counter value matching a selected Match register. In bidirectional mode, events can also be enabled based on the direction of count.

When CONFIG[UNIFY] = 0, each event is associated with a particular counter by the HEVENT bit in its Event Control register. An event is permanently disabled when its event state mask register contains all 0s.

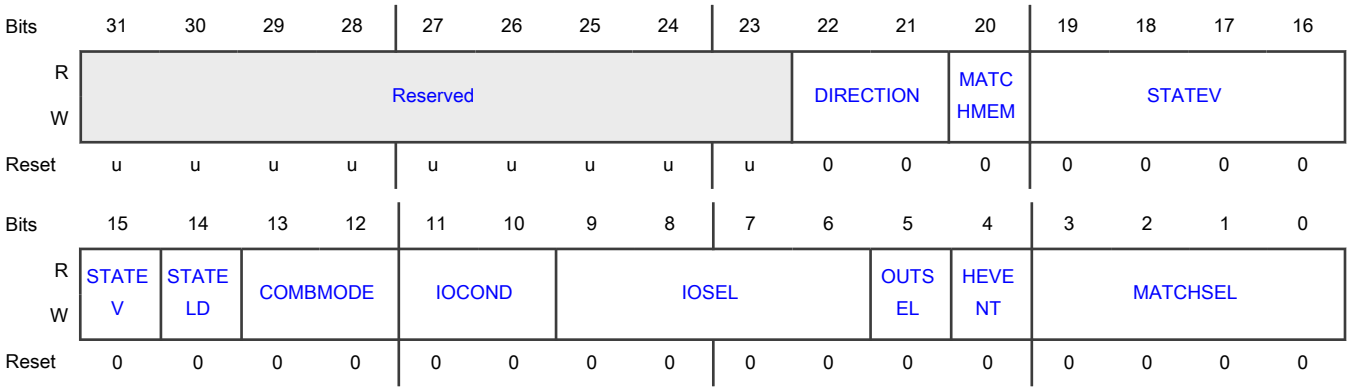
Each event can modify its counter STATE value. If more than one event associated with the same counter occurs in a given clock cycle, only the state change specified for the highest-numbered event among them takes place. Other actions dictated by any simultaneously occurring events all take place.

Offset

For n = 0 to 15:

Register	Offset
EVn_CTRL	304h + (n × 8h)

Diagram



Fields

Field	Description
31-23 —	Reserved
22-21 DIRECTION	Direction Direction qualifier for event generation. This field only applies when the counters are operating in BIDIR mode. If BIDIR = 0, the SCT ignores this field. 00 - Direction independent. This event is triggered regardless of the count direction. 01 - Counting up. This event is triggered only during up-counting when BIDIR = 1. 10 - Counting down. This event is triggered only during down-counting when BIDIR = 1. 11 - Reserved
20 MATCHMEM	Match Mem If this bit is one and the COMBMODE field specifies a match component to the triggering of this event, then a match is considered to be active whenever the counter value is GREATER THAN OR EQUAL TO the value specified in the match register when counting up, LESS THEN OR EQUAL TO the match value when counting down. If this bit is zero, a match is only be active during the cycle when the counter is equal to the match value.
19-15	State Value

Table continues on the next page...

Table continued from the previous page...

Field	Description
STATEV	This value is loaded into or added to the state selected by HEVENT, depending on STATELD, when this event is the highest-numbered event occurring for that state. If STATELD and STATEV are both zero, there is no change to the STATE value.
14 STATELD	State Load This bit controls how the STATEV value modifies the state selected by HEVENT when this event is the highest-numbered event occurring for that state. 0 - Add. STATEV value is added into STATE (the carry-out is ignored). 1 - Load. STATEV value is loaded into STATE.
13-12 COMBMODE	Combination Mode Selects how the specified match and I/O condition are used and combined. 00 - OR. The event occurs when either the specified match or I/O condition occurs. 01 - MATCH. Uses the specified match only. 10 - IO. Uses the specified I/O condition only. 11 - AND. The event occurs when the specified match and I/O condition occur simultaneously.
11-10 IOCOND	Input/Output Condition Selects the I/O condition for event n. (The detection of edges on outputs lag the conditions that switch the outputs by one SCT clock). In order to guarantee proper edge/state detection, an input must have a minimum pulse width of at least one SCT clock period . 00 - Low 01 - Rise 10 - Fall 11 - High
9-6 IOSEL	Input/Output Signal Select Selects the input or output signal number associated with this event (if any). Do not select an input in this register if CKMODE is 1x. In this case the clock input is an implicit ingredient of every event.
5 OUTSEL	Input/Output Select 0 - Selects the inputs selected by IOSEL. 1 - Selects the outputs selected by IOSEL.
4 HEVENT	High Event Select L/H counter. Do not set this bit if UNIFY = 1. 0 - Low Counter. Selects the L state and the L match register selected by MATCHSEL. 1 - High Counter. Selects the H state and the H match register selected by MATCHSEL.
3-0 MATCHSEL	Match Select

Table continues on the next page...

Table continued from the previous page...

Field	Description
	Selects the Match register associated with this event (if any). A match can occur only when the counter selected by the HEVENT bit is running.

34.7.3.1.29 Output n Set (OUT0_SET - OUT9_SET)

Based on a selected event, each SCT output can be set. There is one output set register for each SCT output which selects which events can set that output. Each bit of an output set register is associated with a different event (bit 0 with event 0, etc.). A selected event can set or clear the output depending on the setting of the SETCLRn field in the OUTPUTDIRCTRL register. To define the actual event that sets the output (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

NOTE

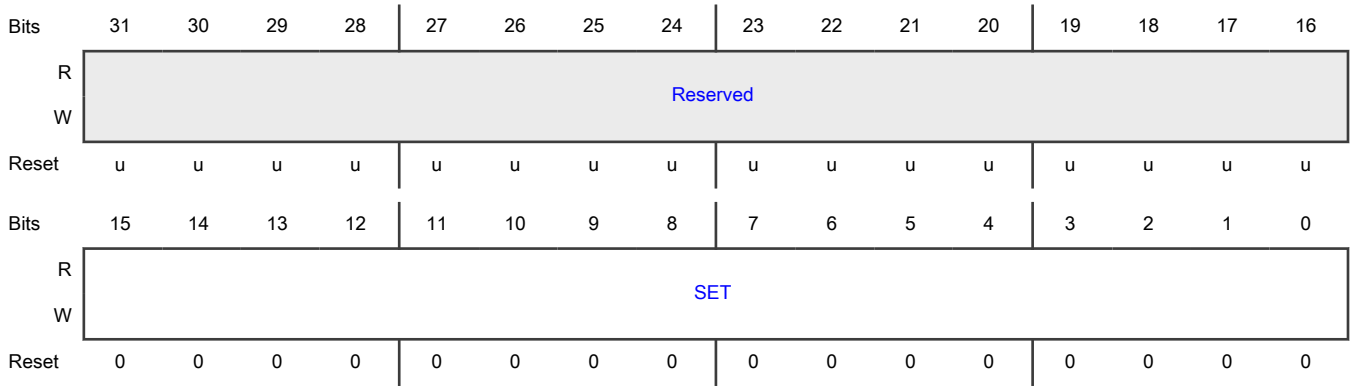
If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Offset

For n = 0 to 9:

Register	Offset
OUTn_SET	500h + (n × 8h)

Diagram



Fields

Field	Description
31-16 —	Reserved
15-0 SET	Set A 1 in bit m selects event m to set output n (or clear it if SETCLRn = 0x1 or 0x2) output 0 = bit 0, output 1 = bit 1, etc. The number of bits = number of events in this SCT. When the counter is used in bidirectional

Table continues on the next page...

Table continued from the previous page...

Field	Description
	mode, it is possible to reverse the action specified by the output set and clear registers when counting down, See the OUTPUTDIRCTRL register.

34.7.3.1.30 Output n Clear (OUT0_CLR - OUT9_CLR)

Based on a selected event, each SCT output can be cleared.

There is one register for each SCT output which selects which events can clear that output. Each bit of an output clear register is associated with a different event (bit 0 with event 0, etc.). A selected event can clear or set the output depending on the setting of the SETCLRn field in the OUTPUTDIRCTRL register. To define the actual event that clears the output (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

NOTE

If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Offset

For n = 0 to 9:

Register	Offset
OUTn_CLR	504h + (n × 8h)

Diagram

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

Fields

Field	Description
31-16 —	Reserved
15-0	Clear

Table continues on the next page...

Table continued from the previous page...

Field	Description
CLR	A 1 in bit m selects event m to clear output n (or set it if SETCLRn = 0x1 or 0x2) event 0 = bit 0, event 1 = bit 1, etc. The number of bits = number of events in this SCT. When the counter is used in bi-directional mode, it is possible to reverse the action specified by the output set and clear registers when counting down, See the OUTPUTCTRL register.

Chapter 35
OS Event Timer (OSTIMER)

35.1 Chip-specific OSTIMER information

Table 321. Reference links to related information

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

35.1.1 Module instances

This device has one instance of the OSTIMER module, OSTIMER_CM33.

35.1.2 OSTIMER configuration

Configure the OS Event timer as follows:

- Use the AHBCLKCTRL1 register in SYSCON, to enable the clock to the OS Event timer register interface and use the OSTIMER register, to enable the clock 32k peripheral clock. To enable FRO/XTAL 32 kHz output clock use RTCOSC32K register in PMC.
- Use the PRESETCTRL1 register in SYSCON, to clear the reset to the OS Event timer. Clear OS Event Timer Interrupt flag. Read the event timer until it increments. Enable Systems Interrupts in the OS Event Timer. Clear the OS Event Timer Interrupt flag. Enable OS Event Timer interrupt in the NVIC.
- For OS Event timer software reset use the OSTIMER register.
- OS Event timer provides an interrupt to the NVIC.
- This module is placed in Always-ON domain, and hence can be running in all low-power modes including deep power-down. It can be DPD wake-up source.
- To enable the OS Event timer interrupt for waking up from deep-sleep and power-down modes, enable the corresponding interrupt in the STARTER1 register in SYSCON and the NVIC.
- To enable the OS Event timer interrupt for waking up from deep power-down, enable the wake-up in the OSTIMER register in PMC.

NOTE

AHB_clk clock source for OSTimer is limited to maximum frequency of 100 MHz.

35.2 Overview

The OS Event Timer (OSTIMER) includes a shared 42-bit timer and separate 42-bit Match and Capture functions for the Cortex-M33.

35.2.1 Block diagram

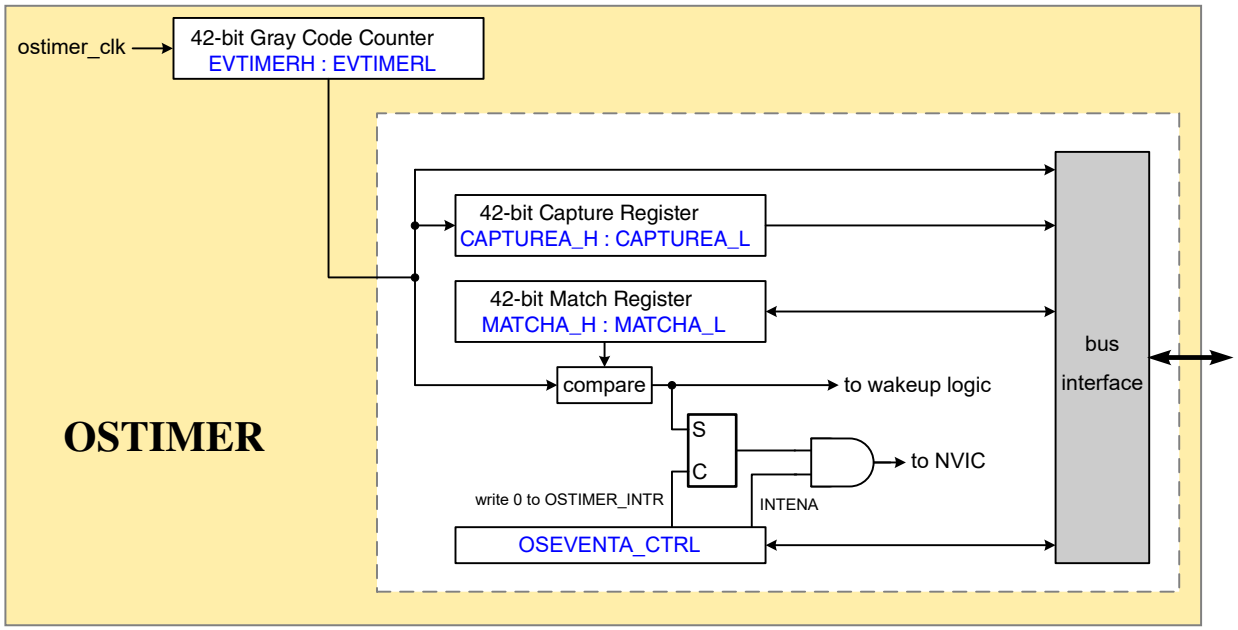


Figure 147. OSTIMER block diagram

35.2.2 Features

- 42-bit Gray code counter. Using Gray code means that the timer can run at a frequency unrelated to the CPU clock and can still be read by the CPU without a synchronization delay. Gray code is a reflected binary code that changes in a single bit position for each increment.
- Separate functions for CPU:
  - A capture register can copy the main counter value when triggered by a CPU request.
  - A match register can be compared to the main counter and can optionally generate an interrupt or wake-up event.

35.3 Functional description

The following sections describe functional details of the OSTIMER module.

35.3.1 Operating modes

Shared event/timestamp timer

The 42-bit shared EVTimer is initialized on device power-up and then counts up continuously. The typical clock for this timer is the 1-MHz low power oscillator.

The shared EVTimer is implemented as a gray-code counter, to enable the counter to be read asynchronously by any of the processing domains or captured by the capture register running on an asynchronous clock. The output of the shared EVTimer is passed to the processor-specific sub-modules.

CPU-specific match, capture and interrupt generation

The submodules associated with CPU include:

- a separate bus interface

- 42-bits of capture values in the Capture_L and Capture_H registers
- 42-bits of match values are available in the Match_H and Match_L registers
- a control register, which includes an interrupt request flag and interrupt enable bit

Capture registers	42-bits of capture values are available in the Capture_L and Capture_H registers for the CPU. A capture command issued by the CPU causes the current value of the main EVTimer to be stored in the Capture registers.
Match registers & Interrupt Request	42-bits of match values are available in the Match_H and Match_L registers for each bus interface. The EVTimer output is compared against this combined value for interrupt/wake-up generation. A match to this register pair will set a flag, which can be enabled to generate the interrupt/wake-up request. The value written to the match register pair must also be specified in gray code. Writes to the match registers should only be done when the status bit in the Control register indicates that it is allowed. When changed, the Match_L register must always be written before the write to the Match_H register.

35.3.2 Reset

For the OS Event Timer, the reset function is more complicated than most functions.

- The shared EVTimer is reset only by power-on reset and by software reset. "Software reset" means a complete reset of the OS Event Timer module.
- The Capture and Match registers are reset by a system reset. "System reset" means a reset from the RESETn pin, or a SYSRESETREQ from the CPU, or a power-on reset, or any other internal low voltage resets.
- The Control register is reset by system reset and by software reset. The effect of resets on OS Event Timer registers is summarized in the next table.

Table 322. Which Resets Cause a Reset of Registers

Registers that are reset by	POR (Power-On Reset)	Software Reset
EVTIMERL and EVTIMERH	Yes	Yes
CAPTUREn_L and CAPTUREn_H	Yes	No
OSEVENT_CTRL	Yes	Yes

35.4 Initialization

To initialize the OSTIMER:

- Enable the clock to the OS Event Timer. This enables the register interface and the peripheral function clock.
- Select a clock source for the OS Event Timer.
- Clear the OS Event Timer peripheral reset.
- The OS Event Timer provides an interrupt to the interrupt controller. To allow interrupts to wake-up the device from deep-sleep mode, enable this interrupt.

35.5 Memory Map and register definition

This section includes the OSTIMER module memory map and detailed descriptions of all registers.

35.5.1 OSTIMER register descriptions

35.5.1.1 OSTIMER memory map

OSTIMER.OSTIMER_CM33 base address: 4002_D000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	EVTIMER Low Register (EVTIMERL)	32	RO	0000_0000
4	EVTIMER High Register (EVTIMERH)	32	RO	0000_0000
8	Local Capture Low Register for CPU (CAPTURE_L)	32	RO	0000_0000
C	Local Capture High Register for CPU (CAPTURE_H)	32	RO	0000_0000
10	Local Match Low Register for CPU (MATCH_L)	32	RW	FFFF_FFFF
14	Local Match High Register for CPU (MATCH_H)	32	RW	FFFF_FFFF
1C	OS Event Timer Control Register for CPU (OSEVENT_CTRL)	32	See section	0000_0000

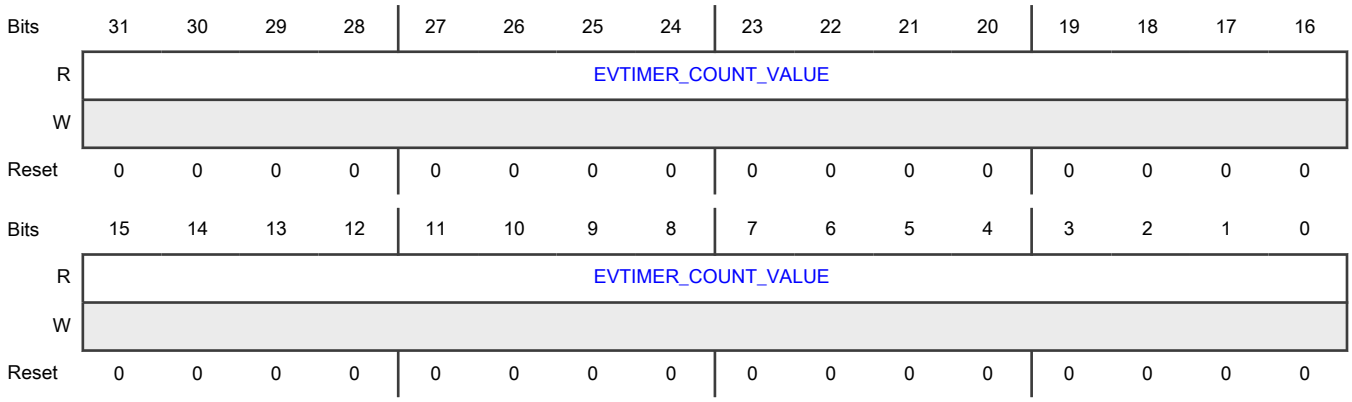
35.5.1.1.1 EVTIMER Low Register (EVTIMERL)

- This counter-register is Gray-encoded. The register is not synchronized to the bus clock.
- Because two 32-bit reads are required to retrieve the entire counter value, the possibility exists that the counter could roll-over between the reads of the low and high EVTIMER registers. Due to the Gray encoding, it can be ensured that even if this roll-over occurs, the value read will represent either the counter value immediately preceding or immediately following the roll-over, if the bus clock rate is at least twice the OS Event Timer clock rate (nominally 1 MHz), or alternatively, if the bus is being clocked by the same 1-MHz LP oscillator that provides the OS Event Timer clock.

Offset

Register	Offset
EVTIMERL	0h

Diagram



Fields

Field	Description
31-0 EVTIMER_COUNT_VALUE	EVTimer Count value A read reflects the current value of the lower 32 bits of the EVTIMER. Note there is physically only one EVTimer, readable from all domains.

35.5.1.1.2 EVTIMER High Register (EVTIMERH)

This register resets only on POR or a software reset.

Offset

Register	Offset
EVTIMERH	4h

Diagram

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

Fields

Field	Description
31-0 EVTIMER_COUNT_VALUE	EVTimer Count value A read reflects the current value of the upper 32 bits of the EVTIMER. Note there is physically only one EVTimer, readable from all domains.

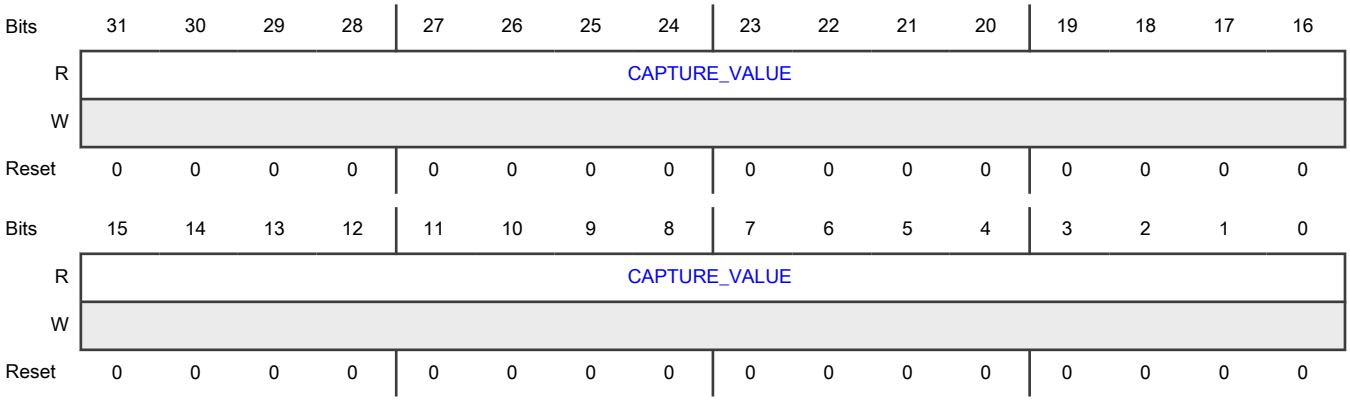
35.5.1.1.3 Local Capture Low Register for CPU (CAPTURE_L)

- This register is gray-encoded to match the EVTIMER. A load is triggered by a load signal from the CPU (for example, an ‘arm_txev’ pulse from the M33 CPU or a ‘tie_expstate’ pulse from the DSP).
- To avoid retrieving incoherent results, a new command to capture the counter should not be executed between reading Capture_L and Capture_H registers.
- For the instance associated with the DSP CPU: If a read of the Capture registers closely follows the command to implement a capture, then it is recommended to insert the `#pragma no_reorder` directive in the DSP code (to direct the compiler to not initiate the read operation until the capture command has executed).

Offset

Register	Offset
CAPTURE_L	8h

Diagram



Fields

Field	Description
31-0 CAPTURE_VALUE	EVTimer Capture value A read reflects the value of the lower 32 bits of the central EVTIMER at the time that the last capture signal was generated by the CPU. A separate pair of CAPTURE registers are implemented for each CPU. Each CPU reads its own capture value at the same pair of addresses.

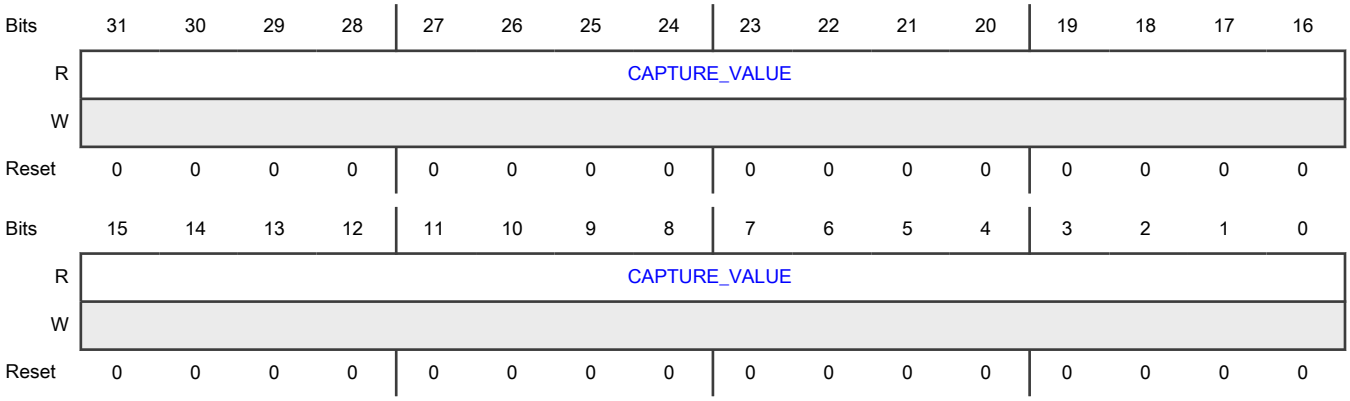
35.5.1.1.4 Local Capture High Register for CPU (CAPTURE_H)

- This register is gray-encoded to match the EVTIMER. A load is triggered by a load signal from the CPU (for example, an ‘arm_txev’ pulse from the M33 CPU or a ‘tie_expstate’ pulse from the DSP).
- To avoid retrieving incoherent results, a new command to capture the counter should not be executed between reading Capture_L and Capture_H registers.
- For the instance associated with the DSP CPU: If a read of the Capture registers closely follows the command to implement a capture, then it is recommended to insert the `#pragma no_reorder` directive in the DSP code (to direct the compiler to not initiate the read operation until the capture command has executed).

Offset

Register	Offset
CAPTURE_H	Ch

Diagram



Fields

Field	Description
31-0 CAPTURE_VALUE	EVTimer Capture value A read reflects the value of the upper 32 bits of the central EVTIMER at the time that the last capture signal was generated by the CPU. A separate pair of CAPTURE registers are implemented for each CPU. Each CPU reads its own capture value at the same pair of addresses.

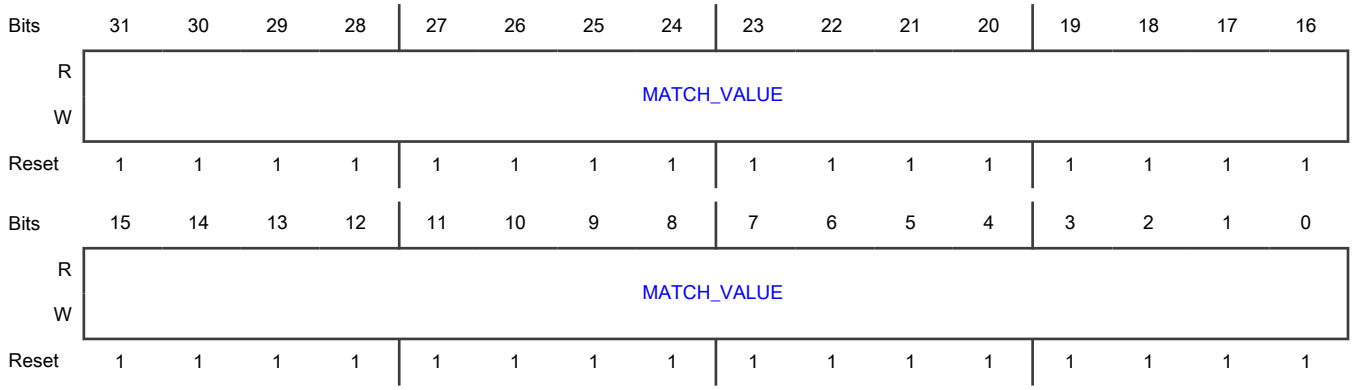
35.5.1.1.5 Local Match Low Register for CPU (MATCH_L)

- This MATCH register is gray-encoded to match the EVTIMER. The value in this pair of registers is stable, so there is no need for multiple reads.
- When writing to the Match Register pair, the Low register must always be written-to first, followed by the write to the High register.
- A second write to the Match Register pair should not be initiated until the first write has taken effect (which is a minimum of 3 bus clocks following by a write to the Match_H register). The EVTimer Match Write Ready status bit (MATCH_WR_RDYn) in the Control register will indicate when it is safe to reload the Match Registers. There is no need to test this status bit if an interrupt due to the previous match value has already occurred, or if it is certain (via some other means) that the required period of time has elapsed.

Offset

Register	Offset
MATCH_L	10h

Diagram



Fields

Field	Description
31-0 MATCH_VALUE	EVTimer Match value The value written to the MATCH (L/H) register pair is compared against the central EVTIMER. When a match occurs, an interrupt request is generated (if the interrupt is enabled). A separate pair of MATCH registers are implemented for each CPU. Each CPU reads its own local value at the same pair of addresses.

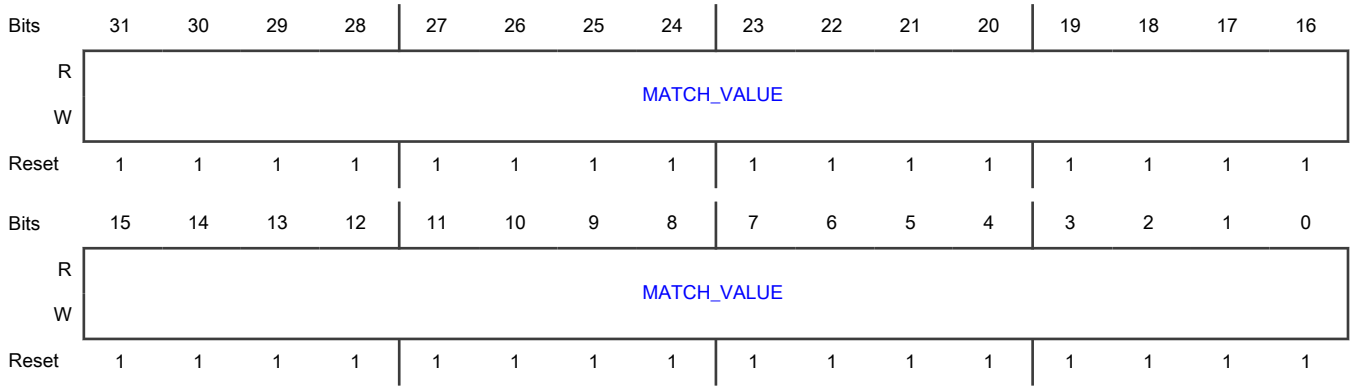
35.5.1.1.6 Local Match High Register for CPU (MATCH_H)

- This MATCH register is gray-encoded to match the EVTIMER. The value in this pair of registers is stable, so there is no need for multiple reads.
- When writing to the Match Register pair, the Low register must always be written-to first, followed by the write to the High register.
- A second write to the Match Register pair should not be initiated until the first write has taken effect (which is a minimum of 3 bus clocks following by a write to the Match_H register). The EVTimer Match Write Ready status bit (MATCH_WR_RDYn) in the Control register will indicate when it is safe to reload the Match Registers. There is no need to test this status bit if an interrupt due to the previous match value has already occurred, or if it is certain (via some other means) that the required period of time has elapsed.

Offset

Register	Offset
MATCH_H	14h

Diagram



Fields

Field	Description
31-0 MATCH_VALUE	EVTimer Match value The value written to the MATCH (L/H) register pair is compared against the central EVTIMER. When a match occurs, an interrupt request is generated (if the interrupt is enabled). A separate pair of MATCH registers are implemented for each CPU. Each CPU reads its own local value at the same pair of addresses.

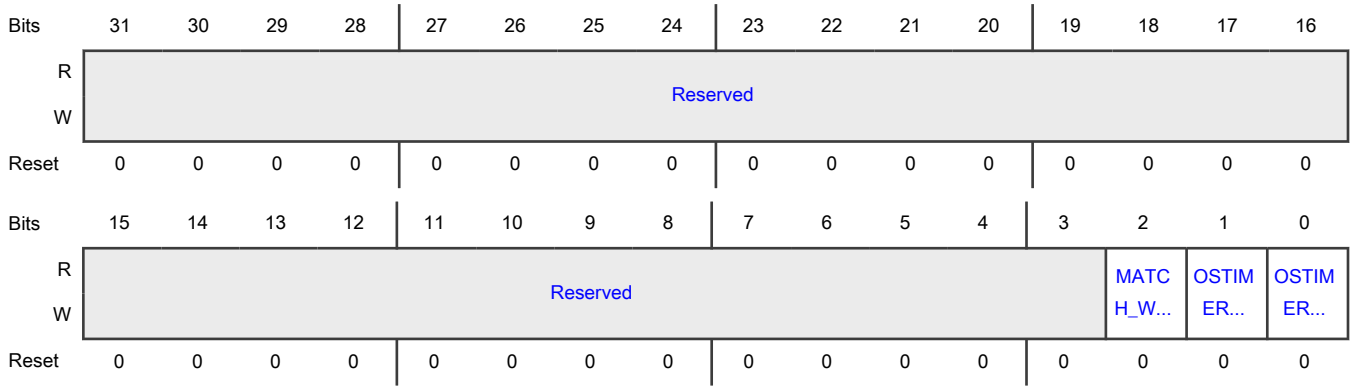
35.5.1.1.7 OS Event Timer Control Register for CPU (OSEVENT_CTRL)

Provides the interrupt flag and interrupt enables for each CPU.

Offset

Register	Offset
OSEVENT_CTRL	1Ch

Diagram



Fields

Field	Description
31-3 —	Reserved
2 MATCH_WR_RDY	EVTimer Match Write Ready When EVTimer Match Write Ready bit is low, it is safe to reload (write) the Match Registers. In typical applications, it should not be necessary to test this bit. The 64-bit MATCH Register value is transferred from a pair of shadow registers to the active Match registers, after the write to the Match_H register. A second write to the Match Registers should not be initiated until after this transfer finishes, as indicated by this status bit returning low. It is not necessary to test this status bit if an interrupt due to the first match value has already occurred, or if it is certain (via some other means) that the required period of time (3 bus clocks) has elapsed.
1 OSTIMER_INTENA	Interrupt/Wake-up Request A separate OSEVENT_CTRL register is implemented for each CPU. Each CPU reads its own local value at the same address. 0 - Interrupt/wake-up requests due to the OSTIMER_INTR flag are blocked. 1 - An interrupt/wake-up request to the domain processor will be asserted when the OSTIMER_INTR flag is set.
0 OSTIMER_INTRFLAG	Interrupt Flag The Interrupt Flag is set when a match occurs between the central 64-bit EVTIMER and the value programmed in the Match-register pair for the associated CPU. • This bit is cleared by writing a 1. • Writes to clear the Interrupt Flag bit are asynchronous. Clearing the Interrupt Flag should be done before a new match value is written into the MATCH_L/H registers.

Chapter 36
Multi-Rate Timer (MRT)

36.1 Chip-specific MRT information

Table 323. Reference links to related information

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

36.1.1 Module instances

This device has one instance of the MRT module, MRT0.

36.2 Overview

The Multi-Rate Timer (MRT) provides a repetitive interrupt timer with 4 channels:

- Each channel can be programmed with an independent time interval.
- Each channel operates independently of the other channels.

36.2.1 Block diagram

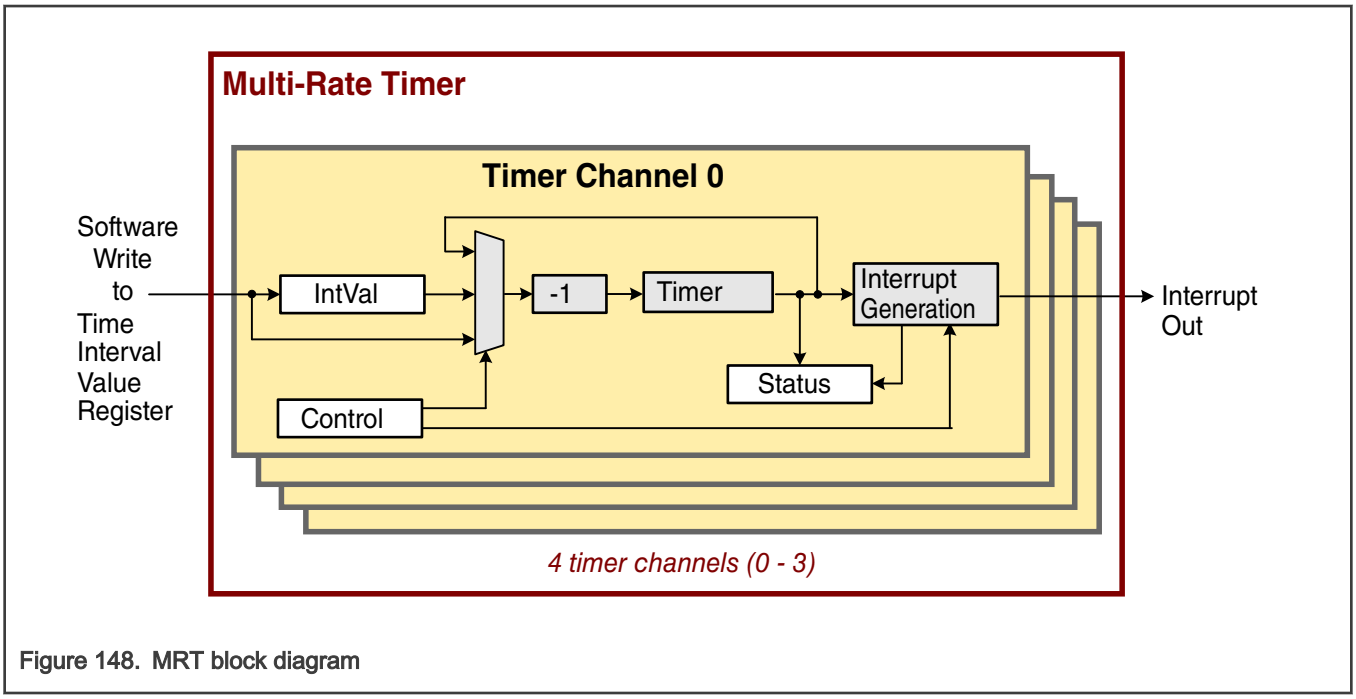


Figure 148. MRT block diagram

36.2.2 Features

- 24-bit interrupt timer
- There are 4 channels independently counting down from individually set values
- Repeat interrupt, one-shot interrupt, and one-shot bus stall modes

36.3 Functional description

The following sections describe functional details of the MRT module.

36.3.1 Operating modes

The Multi-Rate Timer has 3 operating modes:

- Repeat interrupt mode
- One-shot interrupt mode
- One-shot stall mode

36.3.1.1 Repeat interrupt mode

The repeat interrupt mode generates repeated interrupts after a selected time interval. Repeat interrupt mode can be used for software-based Pulse Width Modulation (PWM) or Pulse Position Modulation (PPM) applications.

When the timer *n* is in the idle state, writing a non-zero value (IVALUE) to the INTVAL*n* register immediately loads the time interval value (IVALUE - 1), and the timer begins to count down from this value. After the timer reaches zero:

- an interrupt is generated,
- the value in the INTVAL*n* register (IVALUE - 1) is reloaded automatically,
- and the timer starts to count down again.

While the timer is running in repeat interrupt mode, the following interval timer actions (change or stop) can be performed:

Table 324. Possible actions in repeat interrupt mode

Action	To perform action	Timer does this	Interrupt?
Change the interval value on the next interval timer cycle	• write a new value (>0) to the INTVAL*n* register • and set the LOAD bit to 0	On the next cycle, the timer counts down from the new value	An interrupt is generated when the timer reaches 0
Change the interval value immediately (also called "change on-the-fly") in the interval timer	• write a new value (>0) to the INTVAL*n* register • and set the LOAD bit to 1	The timer immediately starts to count down from the new timer interval value	An interrupt is generated when the timer reaches 0
Stop the interval timer at the end of the time interval	• write a 0 to the INTVAL*n* register • and set the LOAD bit to 0	The timer stops at the end of the time interval	An interrupt is generated when the timer reaches 0
Stop the interval timer immediately	• write a 0 to the INTVAL*n* register • and set the LOAD bit to 1	The timer stops immediately	No interrupt is generated when the INTVAL*n* register is written

36.3.1.2 One-shot interrupt mode

In one-shot interrupt mode, the timer generates one interrupt after a one-time count. Using one-shot interrupt mode, a single interrupt can be generated at any point, and one-shot interrupt mode can be used to introduce a specific delay in a software task.

When the timer is in the idle state, writing a non-zero value (IVALUE) to the INTVAL*n* register immediately loads the time interval value (IVALUE - 1), and the timer starts to count down. After the timer reaches 0, an interrupt is generated and the timer stops (and then enters the idle state).

While the timer is running in one-shot interrupt mode, the following interval timer actions (load or stop) can be performed:

Table 325. Possible actions in one-shot interrupt mode

Action	To perform action	Timer does this	Interrupt
Load a timer interval value	• write a new time interval value (>0) to the INTVAL*n* register • and sett the LOAD bit to 1	The timer immediately reloads the new time interval, and starts counting down from the new value	An interrupt is generated when the timer reaches 0
Stop the interval timer	• write a 0 to the INTVAL*n* register • and set the LOAD bit to 1	The timer immediately stops counting and moves to the idle state	No interrupt is generated when the INTVAL*n* register is updated

36.3.1.3 One-shot stall mode

One-shot stall mode (also called bus stall mode) is similar to one-shot interrupt mode, except that it is intended for very short delays. For example, when the delay needed is less than the time it takes to get to an interrupt service routine. One-shot stall mode is designed for very low software overhead, requiring only a single write to the INTVAL*n* register (if the channel is already configured for one-shot stall mode). The MRT counts the requested delay while stalling the bus write operation, and then concluding the write operation when the delay has finished. No interrupt or status polling is needed.

One-shot stall mode can be used:

- when a short delay is needed between two software-controlled events
- when a delay is expected before the software application can continue

In one-shot stall mode, because there are no bus transactions while the MRT is counting down, the CPU consumes a minimum amount of power during that time.

Note that one-shot stall mode provides a minimum amount of time between the execution of the instruction that performs the write to INTVAL*n* register and the time that software continues. Other system events, such as interrupts or other bus masters accessing the APB bus where the MRT resides, can cause the delay to be longer.

36.4 Signals

There are no external MRT signals.

36.5 Initialization

To initialize the Multi-Rate Timer (MRT):

1. Enable the clock to the MRT.
2. De-assert the MRT reset.
3. Ensure that the MRT can issue an interrupt in the chip.

Please refer Chip-specific section for more details.

36.6 Memory map and register definitions

36.6.1 Multi-Rate Timer (MRT) register descriptions

36.6.1.1 MRT memory map

MRT0 base address: 4000_D000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Time Interval Value (INTVAL0)	32	See section	See section
4	Timer (TIMER0)	32	RO	See section
8	Control (CTRL0)	32	See section	See section
C	Status (STAT0)	32	See section	See section
10	Time Interval Value (INTVAL1)	32	See section	See section
14	Timer (TIMER1)	32	RO	See section
18	Control (CTRL1)	32	See section	See section
1C	Status (STAT1)	32	See section	See section
20	Time Interval Value (INTVAL2)	32	See section	See section
24	Timer (TIMER2)	32	RO	See section
28	Control (CTRL2)	32	See section	See section
2C	Status (STAT2)	32	See section	See section
30	Time Interval Value (INTVAL3)	32	See section	See section
34	Timer (TIMER3)	32	RO	See section
38	Control (CTRL3)	32	See section	See section
3C	Status (STAT3)	32	See section	See section
F0	Module Configuration (MODCFG)	32	See section	See section
F4	Idle Channel (IDLE_CH)	32	RO	See section
F8	Global Interrupt Flag (IRQ_FLAG)	32	See section	See section

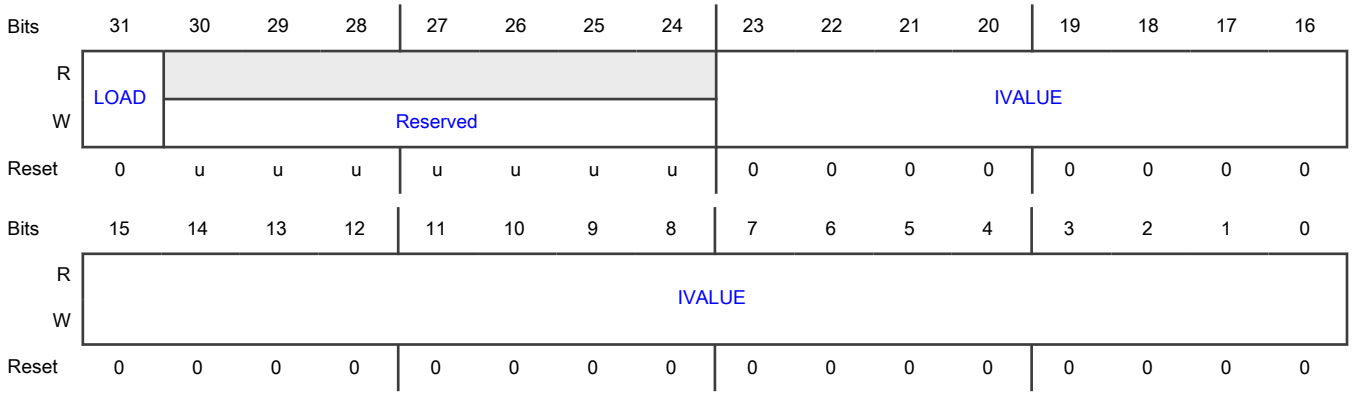
36.6.1.1.1 Time Interval Value (INTVAL0 - INTVAL3)

Contains the MRT load value and controls how the timer is reloaded. The load value is IVALUE -1.

Offset

Register	Offset
INTVAL0	0h
INTVAL1	10h
INTVAL2	20h
INTVAL3	30h

Diagram



Fields

Field	Description
31 LOAD	Determines how the timer interval value (IVALUE -1) is loaded into the TIMER n register. LOAD bit is write-only. Reading LOAD bit always returns 0. 0 - No force load. The load from the INTVAL *n* register to the TIMER *n* register is processed at the end of the time interval, if the repeat mode is selected. 1 - Force load. The INTVAL *n* interval value (IVALUE -1) is immediately loaded into the TIMER *n* register while TIMER *n* is running.
30-24 —	Reserved Read value is undefined; only zero should be written.
23-0 IVALUE	Time interval load value. IVALUE is loaded into the TIMER *n* register, and the MRT channel *n* starts counting down from IVALUE -1. If the timer is idle, then writing a non-zero value to IVALUE starts the timer immediately. If the timer is running, then writing a 0 to IVALUE does the following: if LOAD = 1, then the timer stops immediately; if LOAD = 0, then the timer stops at the end of the time interval.

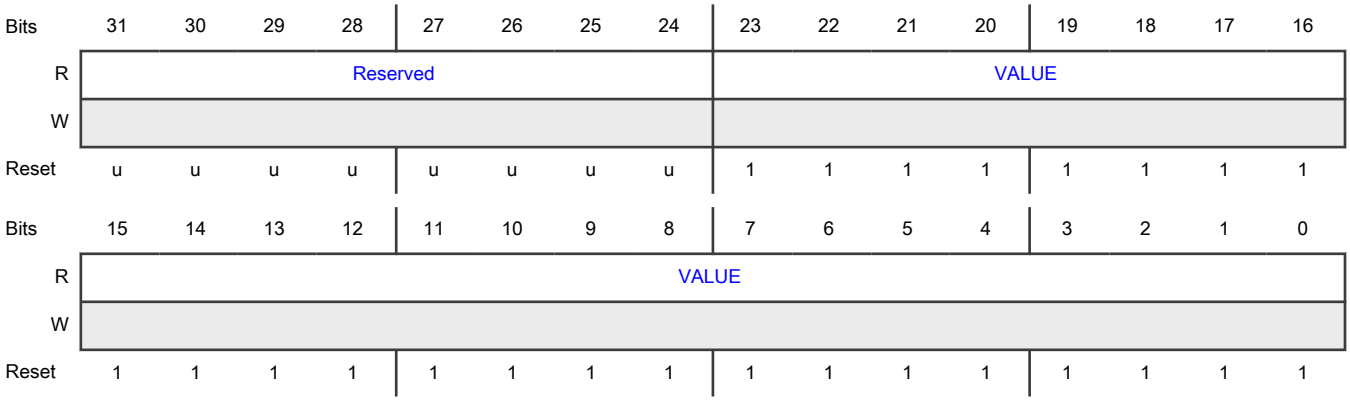
36.6.1.1.2 Timer (TIMER0 - TIMER3)

Holds the current timer value (the TIMER *n* register reads the value of the down-counter). TIMER *n* registers are read-only.

Offset

Register	Offset
TIMER0	4h
TIMER1	14h
TIMER2	24h
TIMER3	34h

Diagram



Fields

Field	Description
31-24 —	Reserved Read value is undefined; only zero should be written.
23-0 VALUE	Holds the current timer value of the down-counter. The initial value of the TIMER *n* register is loaded as (IVALUE - 1) from the INTVAL *n* register, either at the end of the time interval or immediately in the following cases: INTVAL *n* register is updated in the idle state or INTVAL *n* register is updated with LOAD = 1. When the timer is in the idle state, reading VALUE fields returns -1 (0x00FF FFFF).

36.6.1.1.3 Control (CTRL0 - CTRL3)

Controls the modes and enables the interrupt for each MRT.

Offset

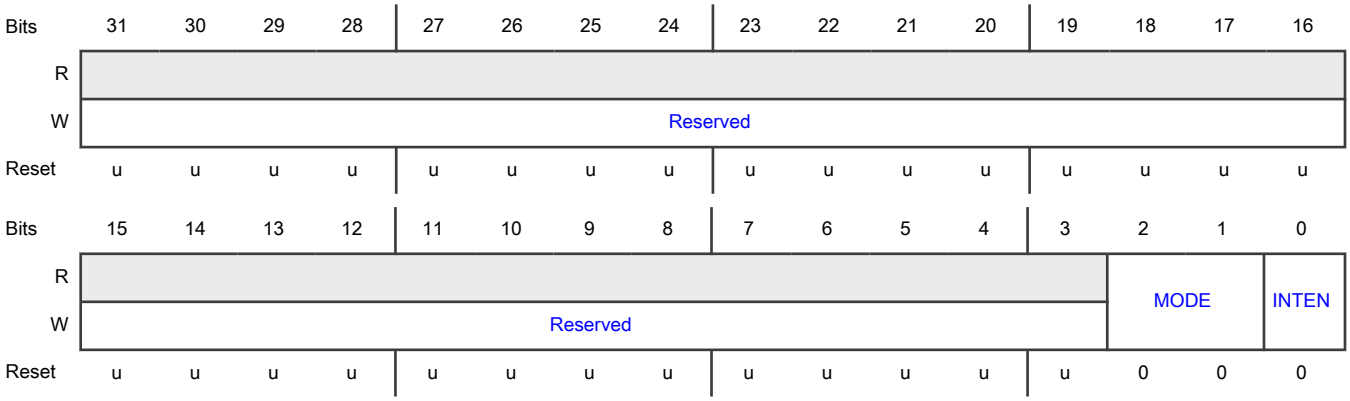
Register	Offset
CTRL0	8h
CTRL1	18h
CTRL2	28h

Table continues on the next page...

Table continued from the previous page...

Register	Offset
CTRL3	38h

Diagram



Fields

Field	Description
31-3 —	Reserved
2-1 MODE	Selects the timer mode 00 - Repeat interrupt mode 01 - One-shot interrupt mode 10 - One-shot stall mode 11 - Reserved
0 INTEN	Enable the TIMER n interrupt. 0 - Disabled. TIMER n interrupt is disabled. 1 - Enabled. TIMER n interrupt is enabled.

36.6.1.1.4 Status (STAT0 - STAT3)

Indicates the status of each MRT timer channel.

Offset

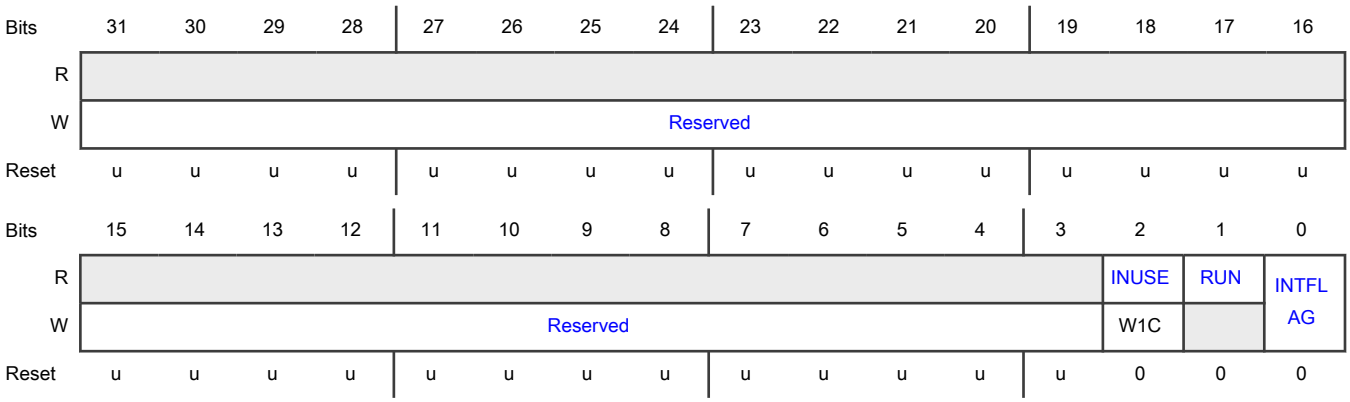
Register	Offset
STAT0	Ch
STAT1	1Ch

Table continues on the next page...

Table continued from the previous page...

Register	Offset
STAT2	2Ch
STAT3	3Ch

Diagram



Fields

Field	Description
31-3 —	Reserved
2 INUSE	Channel-In-Use flag Operating details depend on the operating mode bit in MODCFG[MULTITASK], and affects the use of the Idle Channel register [IDLE_CH]. 0 - This timer channel is not in use. 1 - This timer channel is in use. Writing a 1 to this bit clears the status.
1 RUN	Indicates the state of TIMER n. RUN bit is read-only. 0 - Idle state. TIMER n has stopped. 1 - Running. TIMER n is running.
0 INTFLAG	Monitors the interrupt flag 0 - No pending interrupt. Writing a zero is equivalent to no operation. 1 - Pending interrupt. The interrupt is pending because TIMER *n* has reached the end of the time interval. If the INTEN bit in the CONTROL *n* is also set to 1, then the interrupt for timer channel *n* and the global interrupt are generated. Writing a 1 to INTFLAG bit clears the interrupt request.

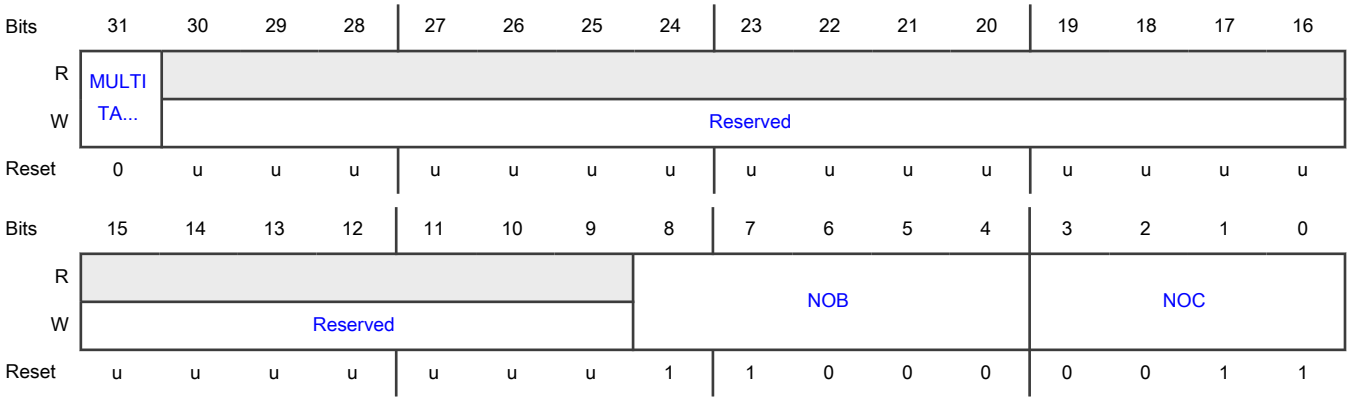
36.6.1.1.5 Module Configuration (MODCFG)

Provides information about this particular MRT instance, and allows selecting an overall mode for the idle channel feature. The MODCFG register provides the configuration (number of channels and timer width) for this MRT.

Offset

Register	Offset
MODCFG	F0h

Diagram



Fields

Field	Description
31 MULTITASK	Selects the operating mode for the INUSE flags and the IDLE_CH register. 0 - Hardware status mode. In this mode, the INUSE(n) flags for all channels are reset. 1 - Multi-task mode
30-9 —	Reserved Read value is undefined; only zero should be written.
8-4 NOB	Number Of Bits: identifies the number of timer bits in this MRT. (24 bits on this device)
3-0 NOC	Number Of Channels: identifies the number of channels in this MRT. (Minus 1 encoded)

36.6.1.1.6 Idle Channel (IDLE_CH)

Returns the number of the first idle channel.

The Idle Channel register can be used to assist software in finding available channels in the MRT. This allows more flexibility by not giving hard assignments to software that makes use of the MRT, without the need to search for an available channel. Generally, IDLE_CH returns the lowest available channel number.

IDLE_CH can be used in two ways, controlled by the value of the MULTITASK bit in the MODCFG register. MULTITASK affects both the function of IDLE_CH, and the function of the INUSE bit for each MRT channel:

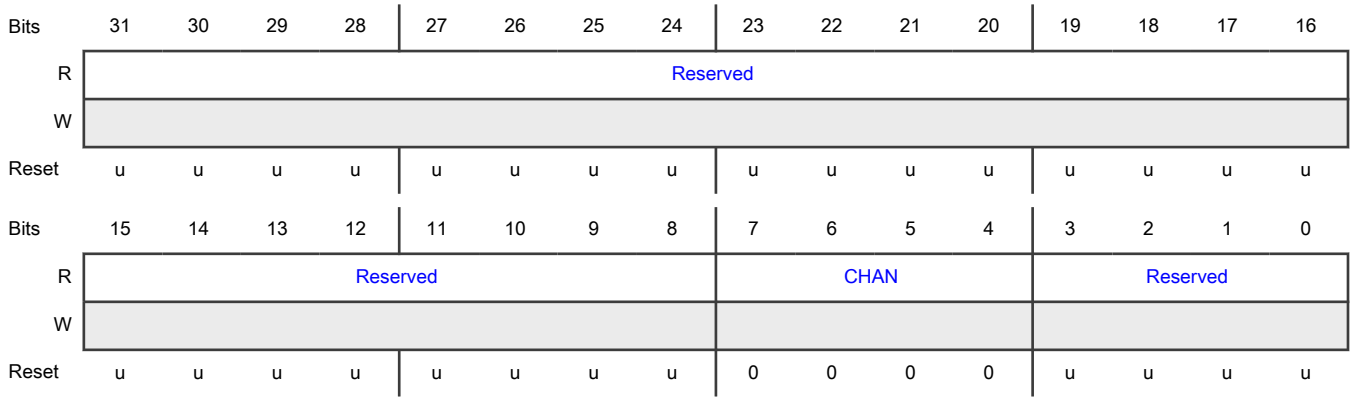
- MULTITASK = 0: hardware status mode. The INUSE flags for all MRT channels are reset. IDLE_CH returns the lowest idle channel number. A channel is considered idle if its RUN flag = 0, and there is no interrupt pending for that channel.
- MULTITASK = 1: multi-task mode. In multi-task mode, the INUSE flags allow more control over when MRT channels are released for further use. When IDLE_CH is read, returning a channel number of an idle channel, the INUSE flag for that

channel is set by hardware. That channel will not be considered idle until its RUN flag = 0, and there is no interrupt pending, and its INUSE flag = 0. This allows reserving an MRT channel with a single register read, and there is no need to start the channel before that channel is no longer considered idle by IDLE_CH. It also allows software to identify a specific MRT channel that software can use, and then software can use that channel more than once without releasing it, removing the need to ask for an available channel for every use (by software).

Offset

Register	Offset
IDLE_CH	F4h

Diagram



Fields

Field	Description
31-8 —	Reserved
7-4 CHAN	Idle channel. Reading the CHAN bits, returns the lowest idle timer channel. The number is positioned so that it can be used as an offset from the MRT base address, to access the registers for the allocated channel. If all timer channels are running, then CHAN = 0xF.
3-0 —	Reserved

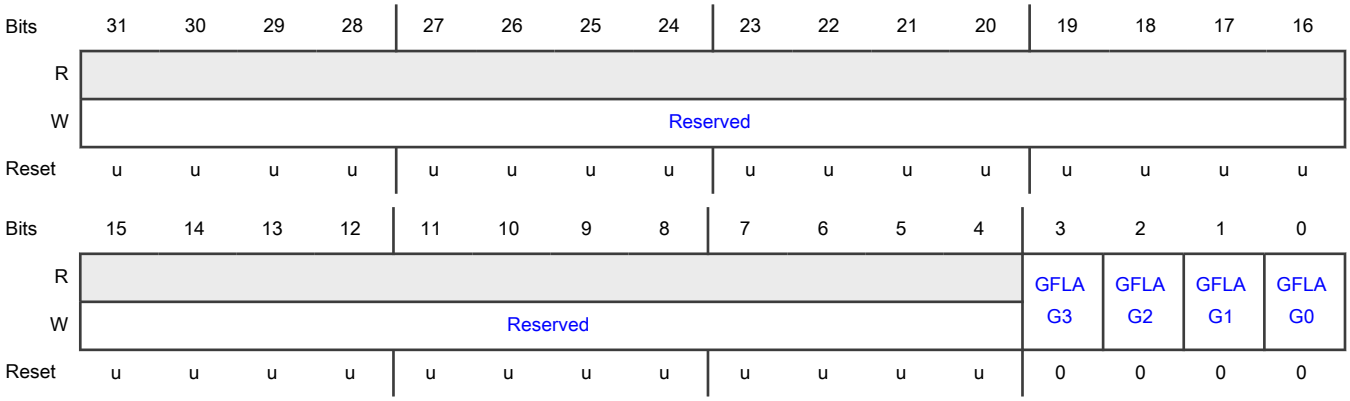
36.6.1.1.7 Global Interrupt Flag (IRQ_FLAG)

The global interrupt register combines the interrupt flags from the individual timer channels into one register. Setting and clearing each flag behaves in the same way as setting and clearing the INTFLAG bit in each of the STATUS *n* registers.

Offset

Register	Offset
IRQ_FLAG	F8h

Diagram



Fields

Field	Description
31-4 —	Reserved Read value is undefined; only zero should be written.
3 GFLAG3	Monitors the interrupt flag of TIMER3, and acts similarly to channel 0.
2 GFLAG2	Monitors the interrupt flag of TIMER2, and acts similarly to channel 0.
1 GFLAG1	Monitors the interrupt flag of TIMER1, and acts similarly to channel 0.
0 GFLAG0	Monitors the interrupt flag of TIMER0. 0 - No pending interrupt. Writing a zero is equivalent to no operation. 1 - Pending interrupt. The interrupt is pending because TIMER0 has reached the end of the time interval. If the INTEN bit in the CONTROL0 register is also set to 1, then the interrupt for timer channel 0 and the global interrupt are generated. Writing a 1 to GFLAG0 bit clears the interrupt request.

Chapter 37
Micro-Tick Timer (UTICK)

37.1 Chip-specific UTICK information

Table 326. Reference links to related information

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

37.1.1 Module instances

This device has one instance of the UTICK module, UTICK0.

37.2 Overview

The Micro-Tick Timer (UTICK) is a 31-bit timer that counts down to 0 and then generates an interrupt, thereby providing a fixed time interval between interrupts. UTICK is a simple, ultra-low power timer that can run and wake up the device in any reduced power mode, including deep power-down modes.

37.2.1 Block diagram

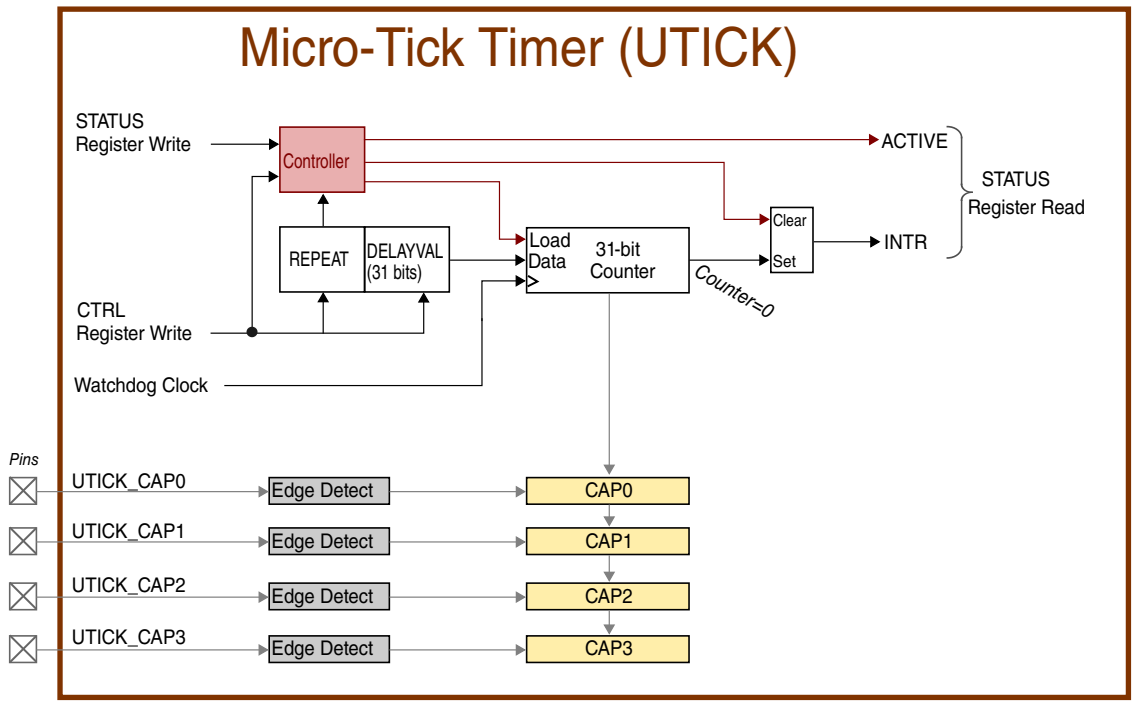


Figure 149. UTICK block diagram

37.2.2 Features

- Can wake up the device in any power mode
- Write once to start
- Interrupt or software polling
- 4 capture registers that can be triggered by external pin transitions

37.3 Functional description

The following sections describe functional details of the UTICK module.

37.3.1 Operations

Before using the Micro-Tick Timer module, decisions must be made:

- Repeat delay: perform a one-time delay or delay continously?
- Delay time: specify a delay value in terms of the timer clock.
- How many events (maximum of 4 events) to capture?
- Capture polarity: capture on a positive or negative edge of the signal?

37.4 Signals

This table describes the UTICK module signals.

Signal	I/O	Description
UTICK_CAP0, UTICK_CAP1, UTICK_CAP2, UTICK_CAP3	I	Capture inputs. The selected transition on a capture pin can be configured to load the related CAP register with the value of counter.

37.5 Initialization

To initialize the Micro-Tick Timer:

- Enable the clock to the Micro-Tick Timer register interface.
- Enable Micro-Tick timer interrupts for waking up from Deep-sleep and Deep power-down modes.
- Configure the pin functions of any Micro-Tick Timer capture pins that will be used.
- Enable the Watchdog oscillator that provides the Micro-Tick Timer clock.

37.6 Memory Map and register definition

This section includes the UTICK module memory map and detailed descriptions of all registers.

37.6.1 UTICK register descriptions

37.6.1.1 UTICK memory map

UTICK0 base address: 4000_E000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Control (CTRL)	32	RW	0000_0000
4	Status (STAT)	32	See section	See section
8	Capture Configuration (CFG)	32	See section	See section
C	Capture Clear (CAPCLR)	32	WO	See section
10 - 1C	Capture (CAP0 - CAP3)	32	RO	0000_0000

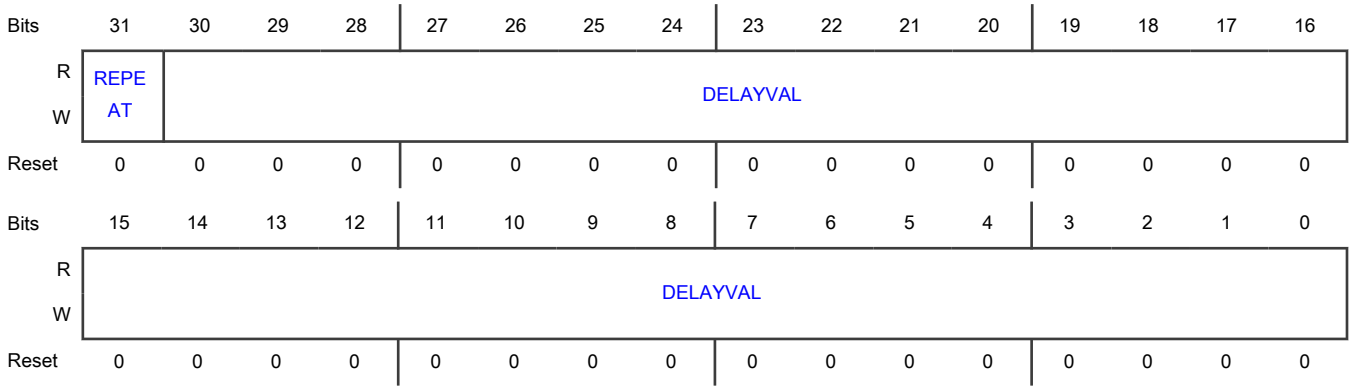
37.6.1.1.1 Control (CTRL)

Any write to the CTRL register resets the counter, meaning a new interval will be measured if one was in progress.

Offset

Register	Offset
CTRL	0h

Diagram



Fields

Field	Description
31 REPEAT	Repeat delay 0 - One-time delay 1 - Delay repeats continuously
30-0 DELAYVAL	Tick interval The delay will be equal to DELAYVAL + 1 periods of the timer clock. The minimum usable tick interval is 1, for a delay of 2 timer clocks. A tick interval = 0 stops the timer.

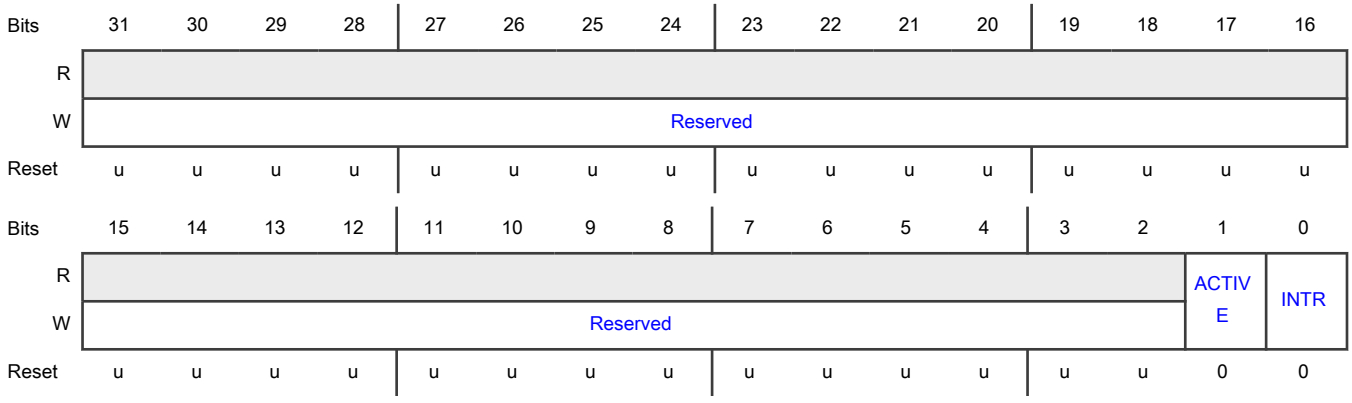
37.6.1.1.2 Status (STAT)

Status

Offset

Register	Offset
STAT	4h

Diagram



Fields

Field	Description
31-2 —	Reserved
1 ACTIVE	Timer active flag 0 - The Micro-Tick Timer is not active (stopped) 1 - The Micro-Tick Timer is currently active
0 INTR	Interrupt flag To clear INTR, write 0 or 1 to INTR. 0 - No interrupt is pending 1 - An interrupt is pending

37.6.1.1.3 Capture Configuration (CFG)

It allows enabling Micro-tick capture functions and selects the polarity of the capture triggers.

Offset

Register	Offset
CFG	8h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	Reserved															
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					CAPPOL3	CAPPOL2	CAPPOL1	CAPPOL0					CAPEN3	CAPEN2	CAPEN1	CAPEN0
W	Reserved								Reserved							
Reset	u	u	u	u	0	0	0	0	u	u	u	u	0	0	0	0

Fields

Field	Description
31-12 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
11 CAPPOL3	Capture Polarity 3 0 - Positive edge capture 1 - Negative edge capture
10 CAPPOL2	Capture Polarity 2 0 - Positive edge capture 1 - Negative edge capture
9 CAPPOL1	Capture Polarity 1 0 - Positive edge capture 1 - Negative edge capture
8 CAPPOL0	Capture Polarity 0 0 - Positive edge capture 1 - Negative edge capture
7-4 —	Reserved
3 CAPEN3	Enable Capture 3 0 - Disabled 1 - Enabled
2 CAPEN2	Enable Capture 2 0 - Disabled 1 - Enabled
1 CAPEN1	Enable Capture 1 0 - Disabled 1 - Enabled
0 CAPEN0	Enable Capture 0 0 - Disabled 1 - Enabled

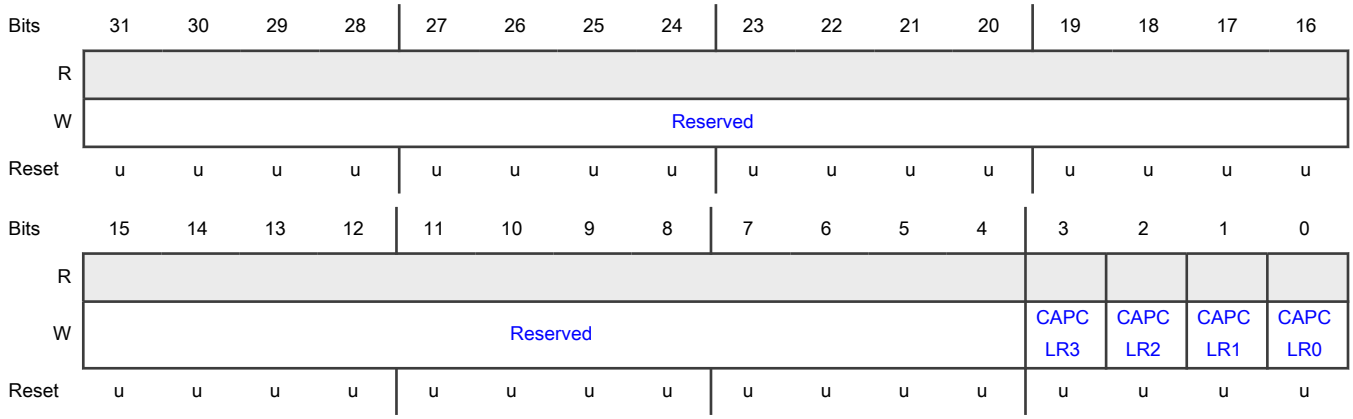
37.6.1.1.4 Capture Clear (CAPCLR)

It allows clearing previous capture values, allowing new captures to take place.

Offset

Register	Offset
CAPCLR	Ch

Diagram



Fields

Field	Description
31-4 —	Reserved
3 CAPCLR3	Clear capture 3 0 - Does nothing 1 - Write 1 to clear the CAP3 register value
2 CAPCLR2	Clear capture 2 0 - Does nothing 1 - Write 1 to clear the CAP2 register value
1 CAPCLR1	Clear capture 1 0 - Does nothing 1 - Write 1 to clear the CAP1 register value
0 CAPCLR0	Clear capture 0 0 - Does nothing 1 - Write 1 to clear the CAP0 register value

37.6.1.1.5 Capture (CAP0 - CAP3)

It contains the Micro-tick time value based on any previously capture events. Each Capture register is associated with one of the capture trigger inputs.

Offset

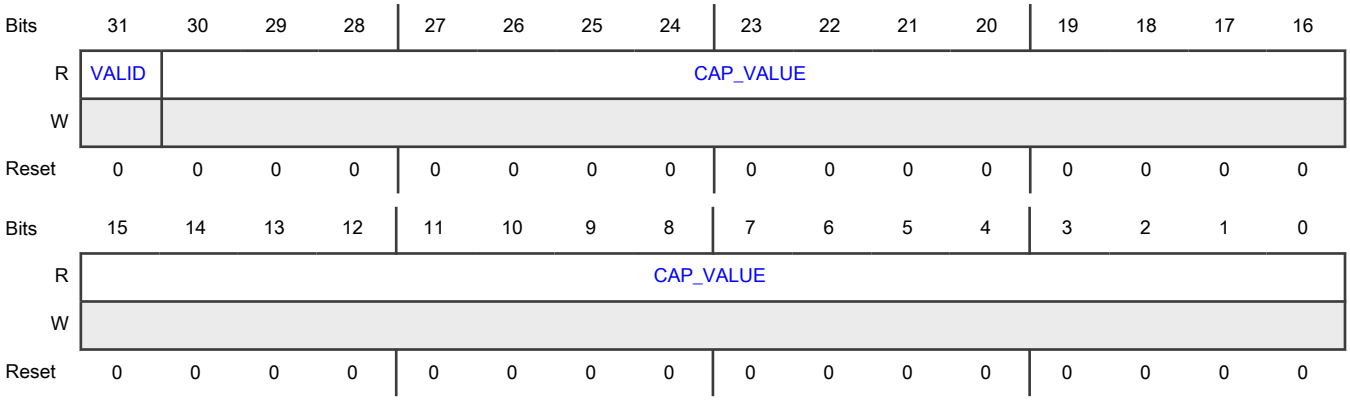
Register	Offset
CAP0	10h

Table continues on the next page...

Table continued from the previous page...

Register	Offset
CAP1	14h
CAP2	18h
CAP3	1Ch

Diagram



Fields

Field	Description
31 VALID	Captured value is valid To clear VALID, write 1 to the related bit in the CAPCLR register. 0 - A valid value has been not been captured 1 - A valid value has been captured, based on a transition of the related UTICK_CAPn pin
30-0 CAP_VALUE	Captured value for the related capture event Note that the captured value is 1 lower than the actual value of the Micro-Tick Timer at the moment of the capture event (actual_captured_value = timer_value - 1).

Chapter 38
Windowed Watchdog Timer (WWDT)

38.1 Chip-specific WWDT information

Table 327. Reference links to related information

Topic	Related module	Reference
Full description	WWDT	WWDT
System memory map		System memory map
Clocking		Clock generation
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

Configuration of the WWDT is accomplished as follows:

- Configure WDTCLKDIV register. Release the reset, disable HALT bit and program DIV[5:0].
- Enable the register interface (WWDT bus clock): set the WWDT bit in the AHBCLKCTRL0 register
- For waking up from a WWDT interrupt, enable the watchdog interrupt for wake-up using low power API.

38.1.1 Module instances

This device has only one instance of the WWDT module, WWDT0.

38.2 Overview

If a microcontroller (or core) enters an erroneous state, WWDT can reset or interrupt the microcontroller within a programmable time. If an application fails to reload (also called feed) a Watchdog timer within a prespecified amount of time, a Watchdog reset (if enabled) is generated.

38.2.1 Block diagram

The Windowed Watchdog Timer block diagram is shown below. The synchronization logic (APB bus clock to WDCLK) is not shown in the block diagram. See the chip-specific WWDT information.

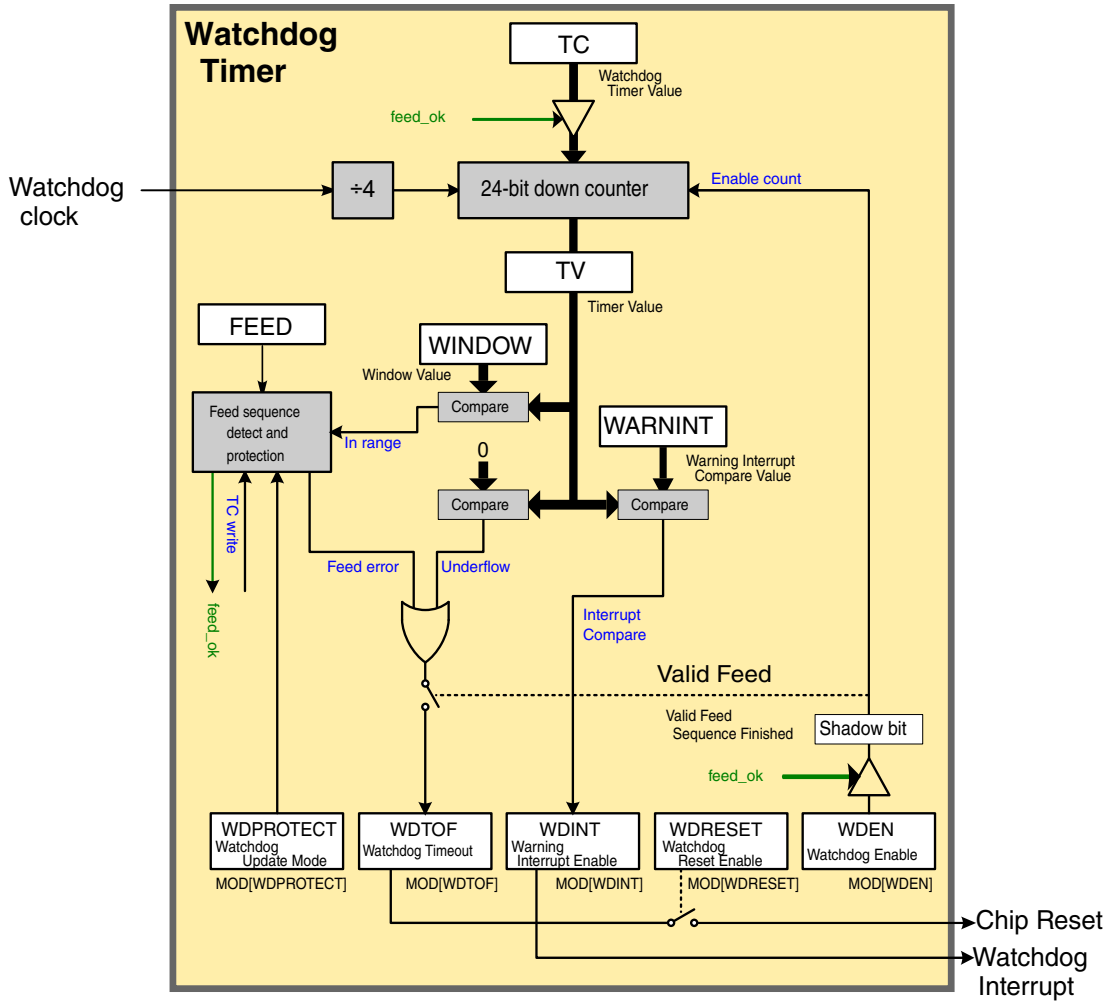


Figure 150. Windowed Watchdog Timer block diagram

38.2.2 Features

- Incorrect feed sequence causes immediate Watchdog event (if enabled).
- Watchdog reset indication flag
- Some WWDT are configurable to reset the chip, if not reloaded during the programmable timeout period. Other WWDT are configurable to reset a core and generate interrupts. See Chip-specific WWDT information.
- Clocking
  - Programmable 24-bit timer with internal fixed pre-scaler
  - Selectable time period in increments of 4 Watchdog clocks:

    from 1,024 Watchdog clocks (T_{WDCLK} x 256 x 4) to over 67 million Watchdog clocks (T_{WDCLK} x 2^{24} x 4)
  - Watchdog clock (WDCLK) source is a selectable frequency in the range of 6 kHz to 1.5 MHz
- WWDT can be configured to run in Deep-sleep mode
- Debug mode
- Optional features

— Optional windowed operation requires reload to occur between programmable minimum and maximum timeout periods.

— Optional warning interrupt can be generated at a programmable time prior to Watchdog timeout.

— Watchdog reload value can optionally be protected so it can only be changed after the warning interrupt time is reached.

— Optional safe operation, which requires a hardware reset or a Watchdog reset to be disabled.

38.3 Functional description

When a Watchdog window is programmed (a timing window), an early Watchdog reload is also treated as a Watchdog event, which prevents situations where a system failure may still reload the Watchdog timer. For example, application code could be stuck in an interrupt service that contains a Watchdog reload (feed). Setting the timing window such that this situation would cause an early Watchdog timer reload, which then would generate a Watchdog event, and thereby enable the system to recover from this situation.

WWDT consists of a fixed (divide by 4) pre-scaler and a 24-bit counter (which decrements when clocked). The minimum value is 0xFF; setting a value lower than 0xFF causes 0xFF to be loaded in the counter.

- Minimum Watchdog interval is (T_{WDCLK} X 256 X 4)
- Maximum Watchdog interval is (T_{WDCLK} X 2^{24} X 4) in multiples of (T_{WDCLK} X 4).

Use WWDT to:

- Enable and configure the Watchdog oscillator. (See Initialization)
- Set the Watchdog timer constant reload value in the Timer Constant (TC) register.
- Set the Watchdog timer Operating mode in the Mode (MOD) register.
- Set a value for the Watchdog window time in the Window Compare Value (WINDOW) register if windowed operation is desired.
- Set a value for the Watchdog warning interrupt in the Warning Interrupt Compare Value (WARNINT) register if a warning interrupt is desired.
- Enable the Watchdog by writing 0xAA followed by 0x55 to the Feed Sequence (FEED) register.

To prevent a Watchdog event, the WWDT must be fed (reloaded) again before the Watchdog counter reaches 0. If a WINDOW value is programmed, then the feed (reload) must also occur after the Watchdog counter passes that programmed WINDOW value.

When the Watchdog timer is configured (MOD[WDRESET]) so that a Watchdog event causes a reset, or when the counter reaches 0 (it causes a reset), then the microcontroller (or core) is also reset (loading the stack pointer and program counter from the vector table, just like what happens for an external reset).

38.3.1 Operating modes

This section describes all functional operation modes of the WWDT module: Disabled, Watchdog Interrupt, Watchdog Reset.

Table 328. Watchdog operating modes selection

MOD[WDEN]	MOD[WDRESET]	Mode	Mode description
0	X (0 or 1)	Disabled	Debug/Operate without the Watchdog running.
1	0	Watchdog Interrupt	The Watchdog warning interrupt generates but the Watchdog reset does not.

Table continues on the next page...

Table 328. Watchdog operating modes selection (continued)

MOD[WDEN]	MOD[WDRESET]	Mode	Mode description
			A Watchdog counter equal to WARNINT sets the WDINT flag and generates a Watchdog interrupt request.
1	1	Watchdog Reset	Both the Watchdog Interrupt and Watchdog Reset are enabled. • A Watchdog counter equal to WARNINT sets the WDINT flag and generates a Watchdog interrupt request. • A Watchdog counter equal to zero resets the microcontroller. • A Watchdog feed prior to reaching the value of WINDOW also causes a Watchdog reset.

38.3.2 Example timing diagrams

The following example timing diagrams show the WWDT in operation.

A feed is correct when both of the following conditions are true:

- A valid feed sequence completes, writing 0xAA followed by 0x55 to the FEED register.
- The TV register value is not greater than the WINDOW register value.

If either of these conditions is not true, a feed error occurs.

In the Correct watchdog feed with windowed mode enabled figure, the sequence of writing 0xAA followed by 0x55 occurs when the counter value (0x11FC) is not greater than WINDOW register value (0x1200). It is a correct feed (reload) event. After the correct feed event occurs, the watchdog counter is reloaded with the TC register value (0x2000). In addition to the above two conditions, when MOD[WDPROTECT] = 1, the watchdog counter must not be greater than the WARNINT register value for a correct feed to occur.

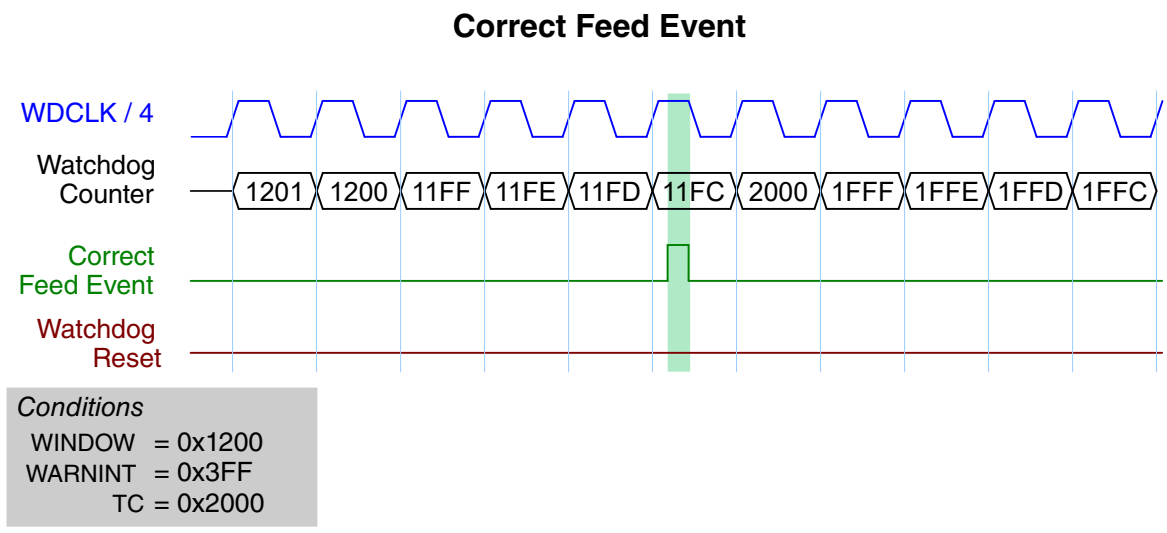


Figure 151. Correct watchdog feed with windowed mode enabled

The Early watchdog feed with windowed mode enabled figure shows a reset being triggered because software fed the watchdog too early.

An early feed event occurs if the feed sequence happens when the watchdog counter value (0x1257) is greater than the WINDOW register value (0x1200). This feed error can generate a reset if MOD[WDRESET] = 1.

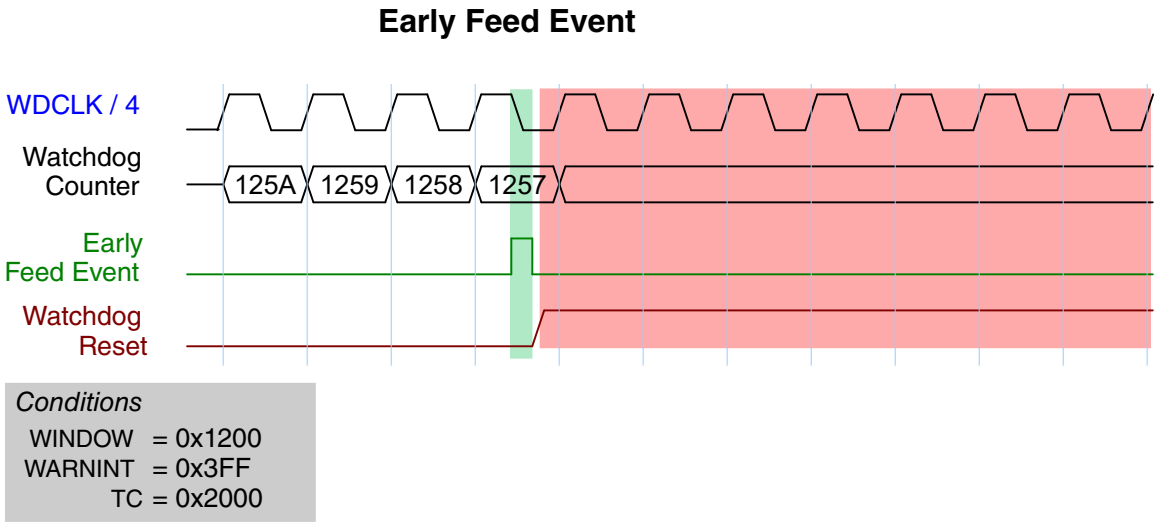


Figure 152. Early watchdog feed with windowed mode enabled

The Watchdog warning interrupt figure below shows the WWDT generating a warning.

A correct feed sequence (done by software) can only happen when the watchdog counter value (TV register) is not greater than the WINDOW value (0x1200). Otherwise, a feed error occurs.

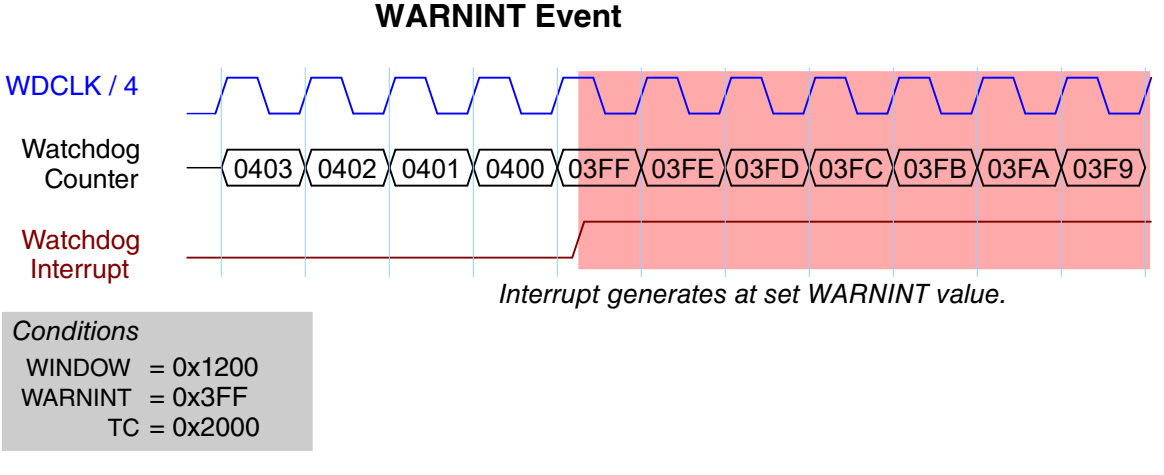


Figure 153. Watchdog warning interrupt

38.3.3 Clocking

WWDT uses two clocks: APB bus clock and WDCLK.

- The APB bus clock is used for the APB accesses to the Watchdog registers and is derived from the system clock.
- The WDCLK is used for the Watchdog timer counting and is derived from the Watchdog oscillator.

The synchronization logic between the two clock domains works as follows:

- When updating the MOD and TC registers, the new value takes effect in 3 WDCLK cycles on the logic in the WDCLK clock domain.
- When the Watchdog timer is counting on WDCLK, the synchronization logic will first lock the value of the counter on WDCLK and then synchronize it with the APB bus clock, so that the CPU can read the TV register.

NOTE

Because of the synchronization step, software must add a delay of three WDCLK clock cycles between the feed sequence and the time software enables MOD[WDPROTECT]. The length of the delay depends on the selected Watchdog clock, WDCLK.

38.3.4 Using WWDT lock

WWDT supports lock features that can ensure that WWDT is always running:

- Preventing the disabling of the WWDT clock source
- Preventing the changing of the WWDT reload value

38.3.4.1 Preventing the disabling of the WWDT clock source

If MOD[LOCK] = 1, the WWDT clock source is locked. Software or hardware cannot disable the clock source when entering Sleep or Deep-sleep modes. Therefore, enable the Watchdog oscillator for each Power mode before setting MOD[LOCK] = 1.

38.3.4.2 Preventing the changing of the WWDT reload value

If the Watchdog Update Mode MOD[WDPROTECT] = 1 (Threshold), then any attempt to change the value of the Timer Constant register (TC) with a feed sequence before the Watchdog counter is less than the values of WARNINT and WINDOW causes a Watchdog reset and sets MOD[WDTOF] = 1.

Any type of reset disables the reload overwrite lock mechanism.

38.4 External signals

This module has no external signals.

38.5 Initialization

Initialize the WWDT as follows:

- Turn on and configure the Watchdog oscillator and the Watchdog oscillator control register.
- Enable the register interface
- Enable the Watchdog interrupt for wake-up.

38.6 Memory map and register definition

This section includes the WWDT module memory map and detailed descriptions of all registers.

38.6.1 WWDT register descriptions

38.6.1.1 WWDT memory map

WWDT0 base address: 4000_C000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Mode (MOD)	32	See section	0000_0000
4	Timer Constant (TC)	32	See section	0000_00FF
8	Feed Sequence (FEED)	32	See section	See section
C	Timer Value (TV)	32	See section	0000_00FF
14	Warning Interrupt Compare Value (WARNINT)	32	See section	0000_0000
18	Window Compare Value (WINDOW)	32	See section	00FF_FFFF

38.6.1.1.1 Mode (MOD)

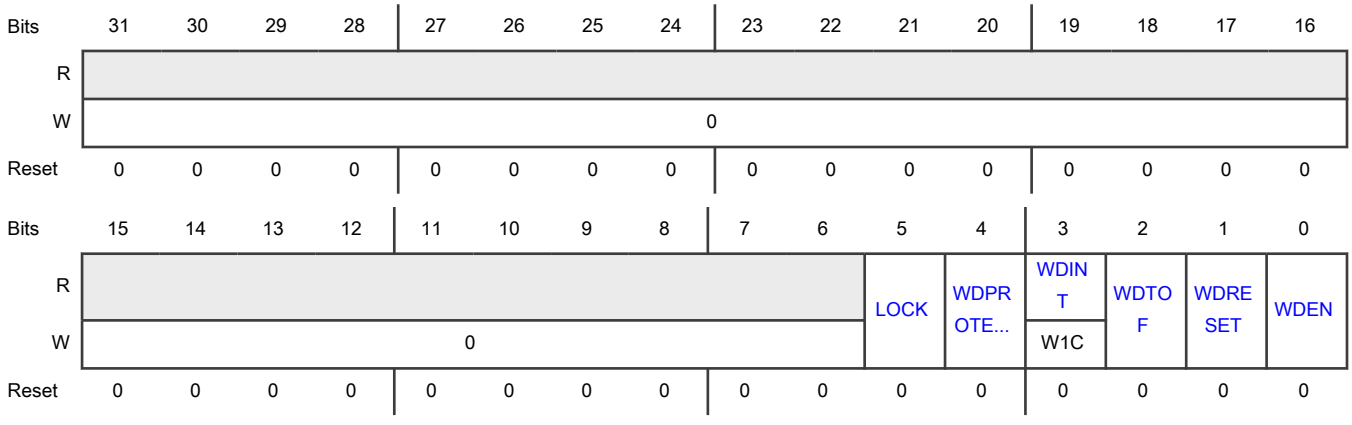
The MOD register controls the operation of WWDT. Note that you must perform a Watchdog feed before any changes to the MOD register takes effect.

After WDEN, WDPROTECT, or WDRESET are set, they cannot be cleared by software, but are cleared by an external reset or a Watchdog timer reset.

Offset

Register	Offset
MOD	0h

Diagram



Fields

Field	Description
31-6 —	Reserved Read value is undefined, only zero should be written.
5 LOCK	Lock After LOCK is set to one and a Watchdog feed is performed, hardware prevents disabling or powering down the Watchdog oscillator. LOCK can be set once by software and is only cleared by any reset. 0 - No Lock 1 - Lock
4 WDPROTECT	Watchdog Update Mode WDPROTECT can be set once by software and is only cleared by a reset. 0 - Flexible. The Timer Constant (TC) can be changed at any time. 1 - Threshold. The Timer Constant (TC) can be changed only after the counter is less than the value of WARNINT and WINDOW. Any attempt to change the value of the Timer Constant register (TC) with a feed sequence before the Watchdog counter is less than the values of WARNINT and WINDOW causes a Watchdog reset and sets MOD[WDTOF] = 1.
3 WDINT	Warning Interrupt Flag Set when the timer is at or below the value in WARNINT. Note that this bit cannot be cleared while the WARNINT value is equal to the value of the TV register. This can occur if the value of WARNINT and WDRESET are both 0 when TV decrements to 0. When the Watchdog timer is configured to generate WDINT, the interrupt occurs when the counter is no longer greater than the value defined by the WARNINT register. Any reset clears this flag. Software clears WDINT by writing a 1. 0 - No flag. 1 - Flag. The Watchdog interrupt flag is set when the Watchdog counter is no longer greater than the value specified by WARNINT.
2 WDTOF	Watchdog Timeout Flag Flag set for these events: • The Watchdog timer times out. • There is a feed error. • When WDPROTECT = 1 and an attempt is made to write to TC. To determine if a Watchdog event has caused the reset condition, examine WDTOF. The flag must be cleared by software by software writing a 0 to WDTOF. **NOTE** WDTOF resets to 0x0 only by an external or a power-on reset. 0 - Clear.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Reset. Causes a chip reset if WDRESET = 1.
1 WDRESET	Watchdog Reset Enable After WDRESET has been written with a 1, WDRESET cannot be re-written with a 0. 0 - Interrupt. A Watchdog timeout will not cause a chip reset. 1 - Reset. A Watchdog timeout will cause a chip reset.
0 WDEN	Watchdog Enable After setting WDEN to one, perform a Watchdog feed (reload) to enable the Watchdog timer. 0 - Stop. The Watchdog timer is stopped. 1 - Run. The Watchdog timer is running.

38.6.1.1.2 Timer Constant (TC)

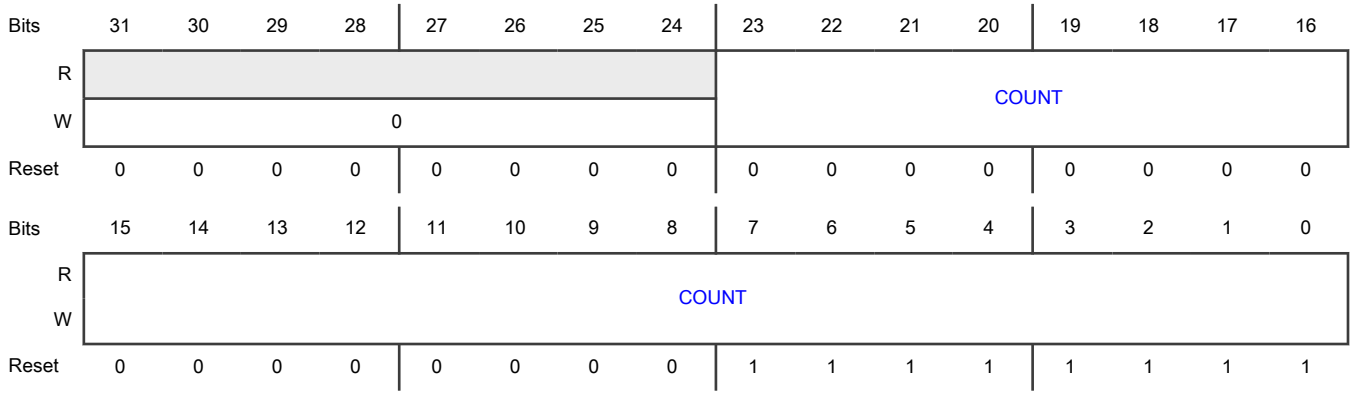
The TC register specifies the desired timeout value. Transferring the TC value into the Watchdog counter requires a feed sequence. The TC resets to 0x00_00FF. Writing a value below 0xFF causes 0x00_00FF to load into the TC. Therefore the minimum time-out interval is T_{WDCLK} X 256 X 4.

If the Watchdog Update Mode (MOD[WDPROTECT]) = 1 (Threshold), then any attempt to change the value of the Timer Constant register (TC) with a feed sequence before the Watchdog counter is less than the values of WARNINT and WINDOW causes a Watchdog reset and sets MOD[WDTOF] = 1.

Offset

Register	Offset
TC	4h

Diagram



Fields

Field	Description
31-24 —	Reserved Read value is undefined, only zero should be written.
23-0 COUNT	Watchdog Timeout Value

38.6.1.1.3 Feed Sequence (FEED)

Writing 0xAA followed by 0x55 to FEED reloads the Watchdog timer with the Timer Constant (TC) register value. This operation also starts WWDT, if the Watchdog timer is enabled using MOD[WDEN] = 1.

However setting MOD[WDEN] is not sufficient to enable WWDT. Before the watchdog can actually generate a reset, MOD[WDEN] must be set, followed by a valid feed sequence that finishes. Until then, the Watchdog ignores feed errors.

After writing 0xAA to FEED, access to any Watchdog register other than writing 0x55 to the FEED register causes an immediate reset/interrupt when WWDT is enabled, and sets the WDTOF flag. The reset is generated during the second APB bus clock after an incorrect access to a Watchdog register during a feed sequence.

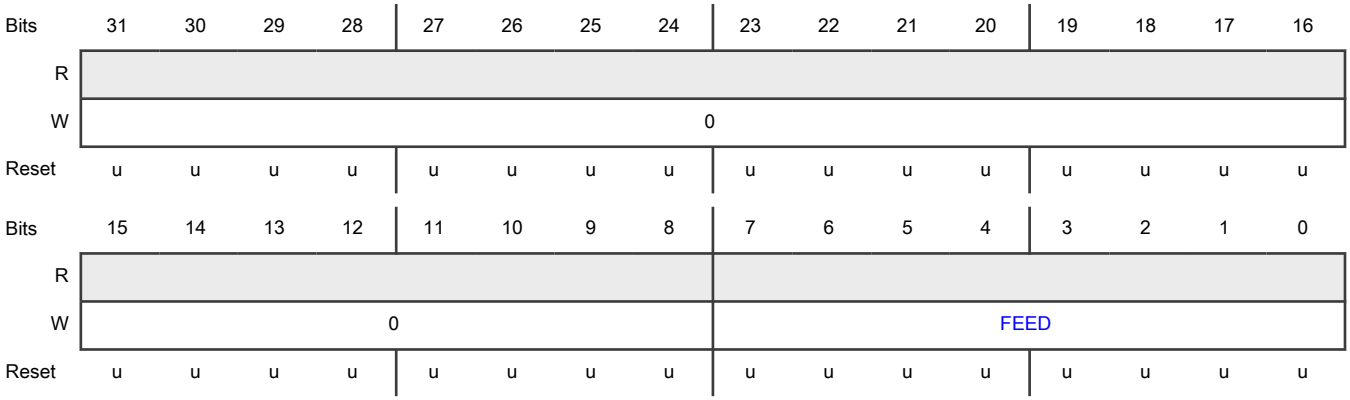
To avoid an unintended interrupt, it is good practice to disable interrupts around a feed sequence. Disable interrupts if they might result in rescheduling processor control away from the current task in the middle of the feed, and then lead to some other access to WWDT before control is returned to the interrupted task.

If a value in the WINDOW register is smaller than the default, it may limit the time when a Watchdog feed is allowed.

Offset

Register	Offset
FEED	8h

Diagram



Fields

Field	Description
31-8 —	Reserved Read value is undefined, only zero should be written.
7-0 FEED	Feed Value Feed value should be 0xAA followed by 0x55.

38.6.1.1.4 Timer Value (TV)

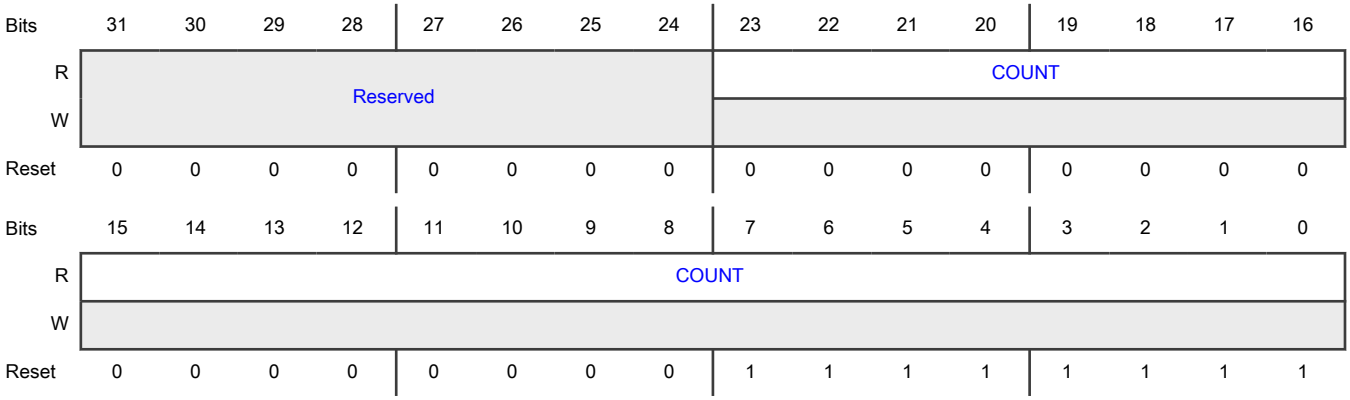
The Timer Value (TV) register reads the current value of the Watchdog timer counter.

When the core (or CPU) reads the TV value, the TV value is older than the actual value of the timer. Reading the timer using the lock and synchronization procedure takes up to 6 WDCLK cycles plus 6 APB bus clock cycles.

Offset

Register	Offset
TV	Ch

Diagram



Fields

Field	Description
31-24 —	Reserved Read value is undefined, only zero should be written.
23-0 COUNT	Counter Timer Value

38.6.1.1.5 Warning Interrupt Compare Value (WARNINT)

The WARNINT register determines the Timer Value (TV) that generates a Watchdog interrupt. When TV[COUNT] is less than or equal to the value defined by WARNINT, an interrupt generates after the subsequent WDCLK.

A match of the Watchdog timer counter to WARNINT occurs when the bottom 10 bits of the timer counter have the same value as the 10 bits of WARNINT, and the remaining upper bits of the counter are all 0. This gives a maximum time of 1,023 Watchdog timer counts (4,096 Watchdog clocks) for the interrupt to occur prior to a Watchdog event. If WARNINT is 0, then the interrupt occurs at the same time as the Watchdog event.

Offset

Register	Offset
WARNINT	14h

Diagram

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

Fields

Field	Description
31-10 —	Reserved Only zero should be written.
9-0 WARNINT	Watchdog Warning Interrupt Compare Value

38.6.1.1.6 Window Compare Value (WINDOW)

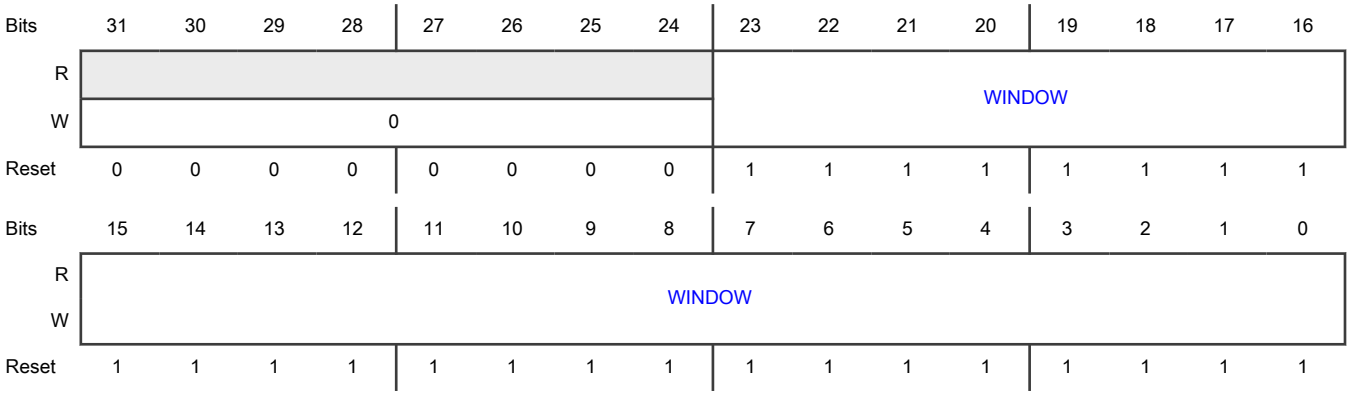
The WINDOW register determines the highest timer value (TV[COUNT]) allowed during a Watchdog feed. If a feed sequence occurs when TV is greater than the value in WINDOW, then a Watchdog event occurs.

WINDOW resets to the maximum TV value, where windowing is not in effect.

Offset

Register	Offset
WINDOW	18h

Diagram



Fields

Field	Description
31-24 —	Reserved Only zero should be written.
23-0 WINDOW	Watchdog Window Value.

Chapter 39
Code Watchdog Timer (CDOG)

39.1 Chip-specific CDOG information

Table 329. Reference links to related information

Topic	Related module	Reference
Full description	CDOG	Code Watchdog Timer
System memory map		System memory map
Clocking		Clock generation
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

39.1.1 Module instances

This device has one instance of the CDOG, CDOG0.

39.1.2 Register reset values after boot

The reset values of the following registers are affected by boot options and are indeterminate:

- INSTRUCTION_TIMER
- CONTROL

39.2 Overview

The Code Watchdog Timer (CDOG) module helps protect the integrity of software by detecting unexpected changes (faults) in code execution flow. The CDOG module can be configured to reset or interrupt the processor core when the module detects a fault.

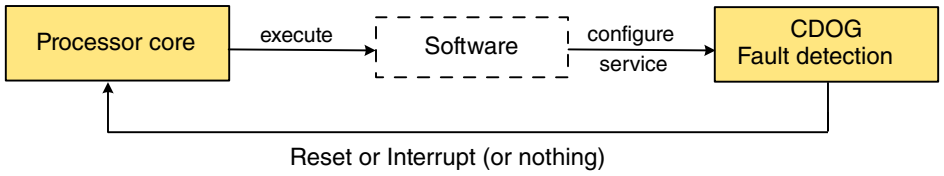


Figure 154. CDOG overview

39.2.1 Block Diagram

The following block diagram shows the components of the CDOG.

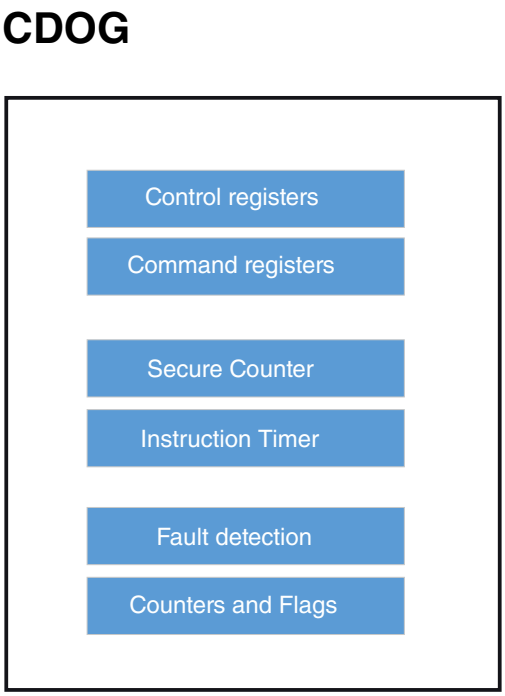


Figure 155. Block diagram of the Code Watchdog Timer (CDOG)

39.2.2 Features

The CDOG includes the following code flow and data integrity checking:

- Faults and flags configurable to generate a system reset, interrupts, or nothing
- Counters for statistics on code behavior patterns for fault types

39.3 Functional Description

The following sections describe functional details of the CDOG module.

39.3.1 Code flow checking

The CDOG provides two primary mechanisms for detecting fault attacks and the execution of unexpected instruction sequences:

- Secure Counter (SECURE_COUNTER)
- Instruction Timer (INSTRUCTION_TIMER)

39.3.1.1 Secure Counter

The Secure Counter (SECURE_COUNTER) is a 32-bit accumulating register that holds a dynamically changing value that can periodically be evaluated to determine if a program is executing as expected. If a mismatch is detected, a fault is generated.

- Secure Counter is an accumulator that when loaded with an initial value lets runtime software issue ADD & SUB commands to increment/decrement the counter.
- Periodically, a Secure Counter value check is initiated by passing the expected value to the CDOG using the STOP and RESTART commands.
- If a mismatch is detected between the Secure Counter and the value passed to it, the execution flow has potentially been altered by a fault attack or some other suspicious activity.

39.3.1.2 Instruction Timer

The Instruction Timer (INSTRUCTION_TIMER) is a 32-bit count-down timer that an application uses to set the number of instructions that software expects to execute. The Instruction Timer counts off the instructions as they are executed (clocks) and if the number is exhausted before the CDOG is serviced, a fault is generated.

The application programmer pre-loads the Instruction Timer with slightly more than the number of instructions in the next execution sequence. The CDOG detects cases where the counter is reset to 0 before all instructions execute, which may indicate that unauthorized instructions are being executed.

- Instruction Timer places a hard upper-limit on the interval between checks of the Secure Counter.
- The Instruction Timer can be programmed to either pause, or keep running while the interrupt service routines are executing.
- The START command loads the internal decremental counter. Before the counter generates an underflow (reaches 0), a STOP or RESTART command must be executed to force a Secure Counter check.

39.3.2 Modes of operation (STATE)

The CDOG has two legal states: IDLE and ACTIVE.

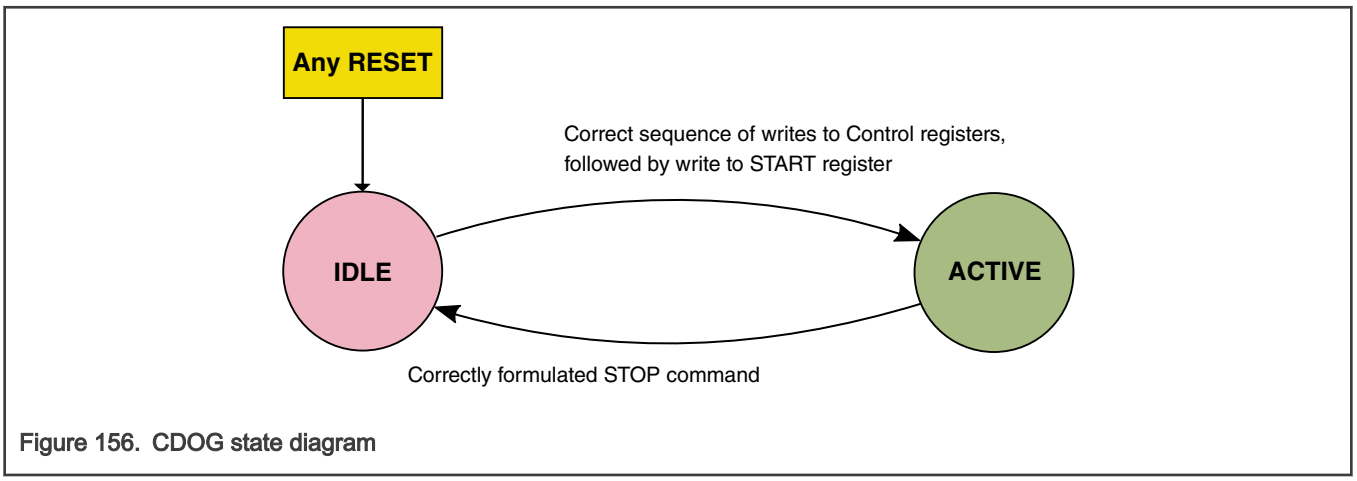


Figure 156. CDOG state diagram

- The two states are encoded in the read-only STATUS[CURST] field.
- After any reset (including a reset generated by the CDOG itself), the module will be in IDLE state.
- To change the module state to ACTIVE, software must execute a correct sequence of writes to the CONTROL group, followed by a START command. See Example use cases.
- Once ACTIVE, a correctly formulated STOP command will change the state back to IDLE.

39.3.3 Faults, flags, and counters

39.3.3.1 Fault types

All fault types (except CONTROL) can be individually controlled to generate either a system reset, an interrupt, or nothing.

Table 330. Fault types

This fault...	Occurs when...
TIMEOUT	The Instruction Timer reaches '0'.

Table continues on the next page...

Table 330. Fault types (continued)

This fault...	Occurs when...
MISCOMPARE	Either a STOP or a RESTART command is issued, and the value passed in the instruction does not match the content of the SECURE_COUNTER register.
SEQUENCE	The prescribed sequence of interactions between SW and CDOG is violated.
CONTROL	Any of the CONTROL register's fields contain an illegal value.
STATE	The internal state machine contains any value other than 5h or Ah.
ADDRESS	Any address in the module's register space, which is not legally defined, is accessed by the CPU.

For CONTROL fault, illegal values in the various bitfields of the CONTROL register itself can generate a system reset if CONTROL[LOCK_CTRL] != 10b. Interrupts are available as an alternative to system reset generation, for all fault types (except CONTROL), primarily to facilitate code development and debug operations. In the final application, interrupts can be enabled for some faults, and reset can be enabled for other faults (or disabled completely), at the risk of reduced security.

39.3.3.2 Fault counters

Each fault type has an associated counter that increments when the fault is detected. The counters are cleared by POR, but not by a system reset. Thus, statistics can be built up over many code watchdog resets to reveal the behavior patterns of a specific type of attack.

39.3.3.3 Flags

Each fault type has an associated flag accessible through the FLAGS register. A flag will be set whenever its associated fault is detected, and can be cleared by SW. The flags themselves (when enabled) generate the module's system reset or interrupt outputs.

The flags retain their value through a system reset (including one generated by the CDOG), but are reset by a POR. Using this mechanism, SW can easily answer the 'How did I get here?' question by reading the FLAGS register after any reset.

To facilitate testing and code development, the flags can be written directly when the module is not locked. When the module is locked, the flags can be cleared by writing '1' to their bit positions.

Table 331. FLAGS summary

Name	POR	ADDR	STATE	CONTROL	SEQUENCE	MISCOMPARE	TIMEOUT
Bit	16	5	4	3	2	1	0
Reset value after POR?	1	0	0	0	0	0	0
Writeable when unlocked?	Y	Y	Y	Y	Y	Y	Y
W-1-C when locked?	Y	Y	Y	Y	Y	Y	Y

39.3.3.4 Fault generation

Some faults are related to when and how registers can be accessed. See Figure 157.

The CDOG supports the following faults:

Table 332. Fault generation

This fault...	Is generated when...
SEQUENCE	• Software does not write to RELOAD before the START command is issued. • Software writes to RELOAD in ACTIVE state. Only write to RELOAD in IDLE state. • Software attempts to write to START in ACTIVE state. Only write to START in IDLE state. • In IDLE state, software attempts to write to a COMMAND group register other than START. • In ACTIVE state, software attempts to write to a CONTROL group register.
ADDRESS	• A read access (within the CDOG address space) to any address outside the CONTROL register group. • A write access (within the CDOG address space) to any address outside both the CONTROL and COMMAND register groups.
TIMEOUT	• The countdown is one clock before the Instruction Timer reaches '0'.
MISCOMPARE	• The STOP register is written and the current Secure Counter value does not match the write data. • The RESTART register is written and the current Secure Counter value does not match the write data.
CONTROL	• The CONTROL[LOCK_CTRL] field is any value other than 01b or 10b. • The CONTROL bit fields: TIMEOUT_CTRL, MISCOMPARE_CTRL, SEQUENCE_CTRL, STATE_CTRL, or ADDRESS_CTRL contain any value other than 001b, 010b, or 100b. • The CONTROL[DEBUG_HALT_CTRL] field contains any value other than 01b or 10b. • The CONTROL[IRQ_PAUSE] field contains any value other than 01b or 10b.
STATE	• The STATE machine contains any value other than 5h or Ah. See STATUS[CURST].

39.4 Application information

This section describes applications supported by the CDOG module.

39.4.1 Example use cases

There is generally a strict sequence with which the CDOG is configured, activated and serviced as follows:

1. Write an Instruction Timer reload value, corresponding to the current code section, to the RELOAD register.
2. Write a control word to the CONTROL register, where each fault type is enabled (or not) to generate a reset, and lock the lock field. The Instruction Timer may be kept running at all times, or only during non-IRQ execution (paused during interrupt processing) based on the value that is written to the CONTROL[IRQ_PAUSE] field.
3. Activate the module by writing to the START command register. The initial value for the Secure Counter register is the value written. The Instruction Timer immediately starts decrementing from the RELOAD value on every clock cycle.

4. At strategically chosen way points in the application's code flow, update the Secure Counter by issuing ADD or SUB commands.
5. When the way point that corresponds to the number of executed instructions represented by the RELOAD value (the end of the current code section) is reached, write the expected value of the Secure Counter register to the STOP command register. Assuming that the written value compares exactly to the contents of the Secure Counter register, the Instruction Timer stops and the process can begin again for the next code section, by repeating steps 1 thru 5.

Another example of a configuration and start procedure would proceed as follows:

1. Write the Instruction Timer reload value to the RELOAD register. It is a good idea to write a very large number to provide a buffer for large values.
2. Write a control word to the CONTROL register, where each fault type is enabled (or not) to generate a reset and then lock the CONTROL[LOCK_CTRL] field.
3. Activate the module by Writing to the START command register. The initial value for the Secure Counter is the value written. The Instruction Timer immediately starts decrementing from the RELOAD value on every clock.
4. At strategically chosen way points in the application's code flow, update the Secure Counter by issuing ADD or SUB commands.
5. At all times within the execution flow, the application must be aware of the instruction counter's approach towards zero. The Instruction Timer may require regular reading to determine the expended time. Software needs to service the CDOG before the TIMER reaches 0, by writing the Secure Counter expected value to the RESTART command register. Assuming the value written compares exactly to the contents of the Secure Counter, the Instruction Timer is reloaded with the value in the RELOAD register as it decrements towards 0.

39.4.2 Using CDOG during code development debug

Following are suggestions for facilitating the code development and debug cycle:

1. Use interrupt-on-fault (instead of reset) to maintain control while adjusting the timing and fine tuning the counting values.
2. Direct-write the bits in the FLAGS register to trigger faults, with the same result as hardware-triggered faults.
3. Use the CONTROL[DEBUG_HALT_CTRL] field to pause the Instruction Timer during a pause of a debug session.

39.5 Memory map and register definition

This section includes the CDOG module memory map and detailed descriptions of all registers.

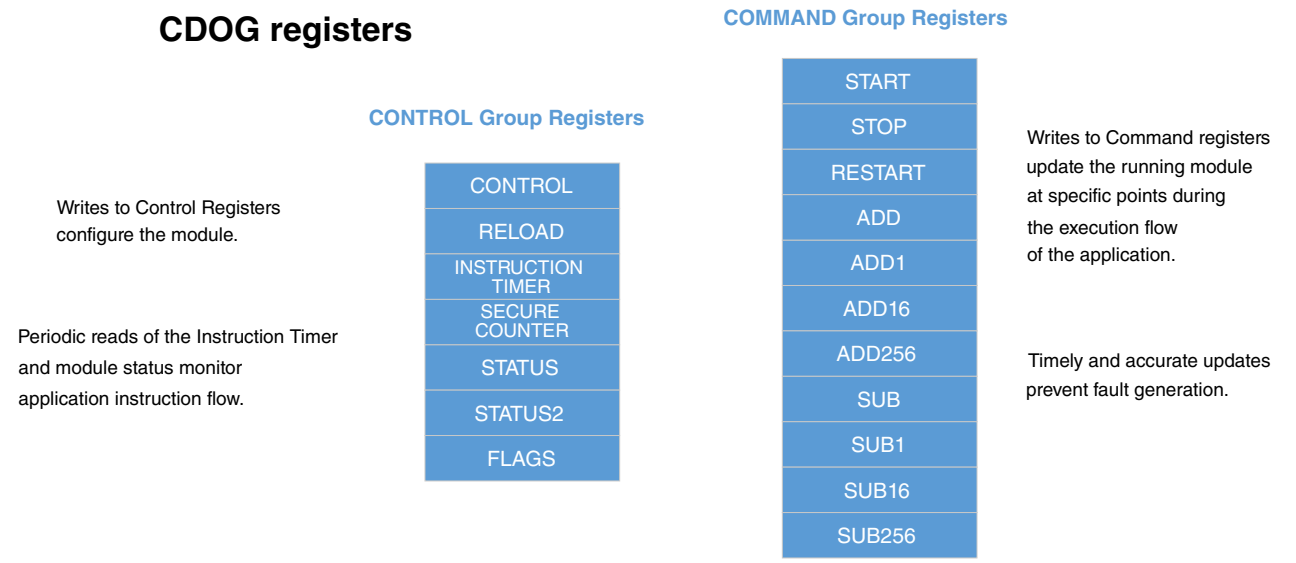


Figure 157. CDOG register groups

39.5.1 CDOG register descriptions

39.5.1.1 CDOG memory map

CDOG0 base address: 400A_1000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Control (CONTROL)	32	See section	5009_2492
4	Instruction Timer reload (RELOAD)	32	RW	FFFF_FFFF
8	Instruction Timer (INSTRUCTION_TIMER)	32	RO	FFFF_FFFF
C	Secure Counter (SECURE_COUNTER)	32	RO	0000_0000
10	Status 1 (STATUS)	32	See section	5000_0000
14	Status 2 (STATUS2)	32	See section	0000_0000
18	Flags (FLAGS)	32	See section	0001_0000
1C	Persistent Data Storage (PERSISTENT)	32	RW	0000_0000
20	START Command (START)	32	WO	0000_0000
24	STOP Command (STOP)	32	WO	0000_0000
28	RESTART Command (RESTART)	32	WO	0000_0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
2C	ADD Command (ADD)	32	WO	0000_0000
30	ADD1 Command (ADD1)	32	WO	0000_0000
34	ADD16 Command (ADD16)	32	WO	0000_0000
38	ADD256 Command (ADD256)	32	WO	0000_0000
3C	SUB Command (SUB)	32	WO	0000_0000
40	SUB1 Command (SUB1)	32	WO	0000_0000
44	SUB16 Command (SUB16)	32	WO	0000_0000
48	SUB256 Command (SUB256)	32	WO	0000_0000

39.5.1.1.1 Control (CONTROL)

The Control register (CONTROL) contains all the controllable attributes of the module, such as how the CDOG module responds when detecting a fault.

Offset

Register	Offset
CONTROL	0h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DEBUG_HALT_CTRL		IRQ_PAUSE		Reserved								ADDRESS_CTRL			STATE_C...
Reset	0	1	0	1	0	0	0	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 W	STATE_CTRL		Reserved		SEQUENCE_CTRL				MISCOMPARE_CTRL				TIMEOUT_CTRL			LOCK_CTRL
Reset	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0

Fields

Field	Description
31-30 DEBUG_HALT_CTRL	DEBUG_HALT control Controls whether the Instruction Timer runs or stops when the CPU asserts DEBUG_HALT.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	Reset value is to keep the timer running. 01 - Keep the timer running 10 - Stop the timer
29-28 IRQ_PAUSE	IRQ pause control Controls whether the Instruction Timer runs or stops when the CPU asserts IRQ_PAUSE. Reset value is to keep the timer running. 01 - Keep the timer running 10 - Stop the timer
27-20 —	Reserved
19-17 ADDRESS_CTRL	ADDRESS fault control Controls how the CDOG module responds when detecting an ADDRESS fault. 001 - Enable reset 010 - Enable interrupt 100 - Disable both reset and interrupt
16-14 STATE_CTRL	STATE fault control Controls how the CDOG module responds when detecting a STATE fault. 001 - Enable reset 010 - Enable interrupt 100 - Disable both reset and interrupt
13-11 —	Reserved Do not change value.
10-8 SEQUENCE_CTRL	SEQUENCE fault control Controls how the CDOG module responds when detecting a SEQUENCE fault. 001 - Enable reset 010 - Enable interrupt 100 - Disable both reset and interrupt
7-5 MISCOMPARE_CTRL	MISCOMPARE fault control Controls how the CDOG module responds when detecting a MISCOMPARE fault. 001 - Enable reset 010 - Enable interrupt

Table continues on the next page...

Table continued from the previous page...

Field	Description
	100 - Disable both reset and interrupt
4-2 TIMEOUT_CTRL	TIMEOUT fault control Controls how the CDOG module responds when detecting a TIMEOUT fault. 001 - Enable reset 010 - Enable interrupt 100 - Disable both reset and interrupt
1-0 LOCK_CTRL	Lock control Resets to unlocked. The CONTROL register is writable and readable only when unlocked, LOCK_CTRL = 10. 01 - Locked 10 - Unlocked

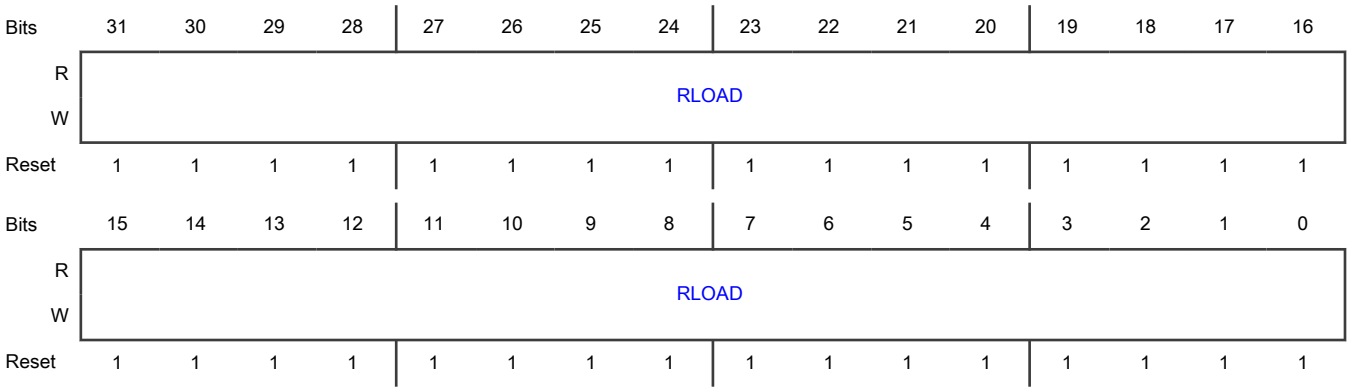
39.5.1.1.2 Instruction Timer reload (RELOAD)

The Instruction Timer RELOAD register contains the value from which the Instruction Timer counts down when a valid START or RESTART command is executed. Write to RELOAD only when the module is in the IDLE state. A write to RELOAD must occur before issuing a START command.

Offset

Register	Offset
RELOAD	4h

Diagram



Fields

Field	Description
31-0 RLOAD	Instruction Timer reload value Contains the starting countdown value used by the Instruction Timer whenever a START or RESTART command is executed.

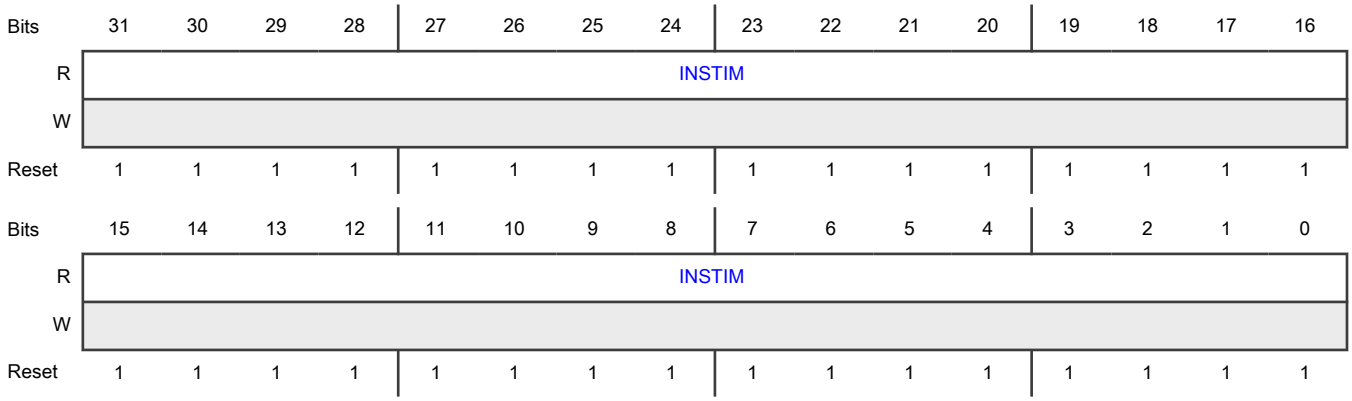
39.5.1.1.3 Instruction Timer (INSTRUCTION_TIMER)

The Instruction Timer register (INSTRUCTION_TIMER) contains the current count value of the Instruction Timer.

Offset

Register	Offset
INSTRUCTION_TIMER	8h

Diagram



Fields

Field	Description
31-0 INSTIM	Current value of the Instruction Timer The current value of the timer can be read from this address. The Software cannot write to INSTRUCTION_TIMER.

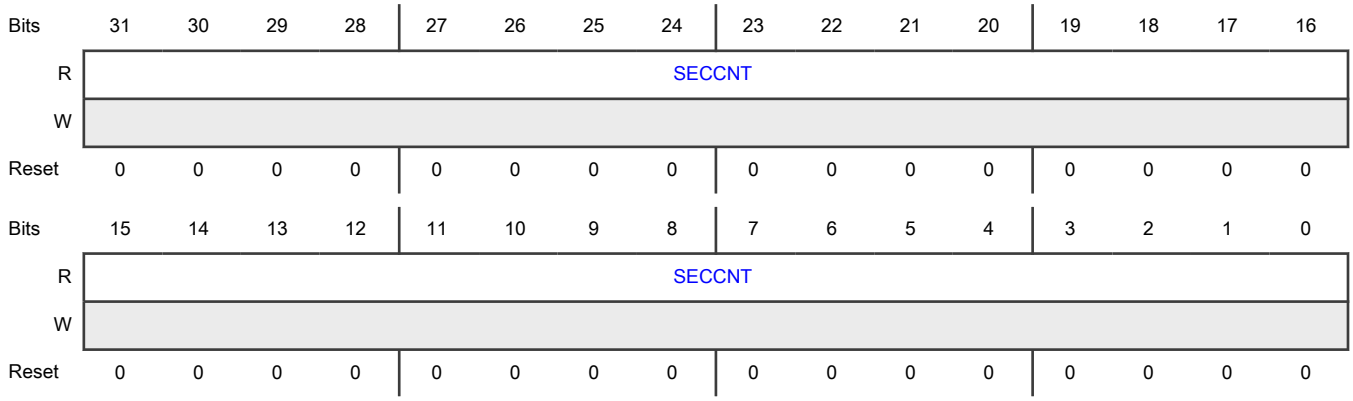
39.5.1.1.4 Secure Counter (SECURE_COUNTER)

The Secure Counter register (SECURE_COUNTER) supplies the start value for the Secure Counter.

Offset

Register	Offset
SECURE_COUNTER	Ch

Diagram



Fields

Field	Description
31-0 SECCNT	Secure Counter Contains the start value for the Secure Counter. Read value is always zero.

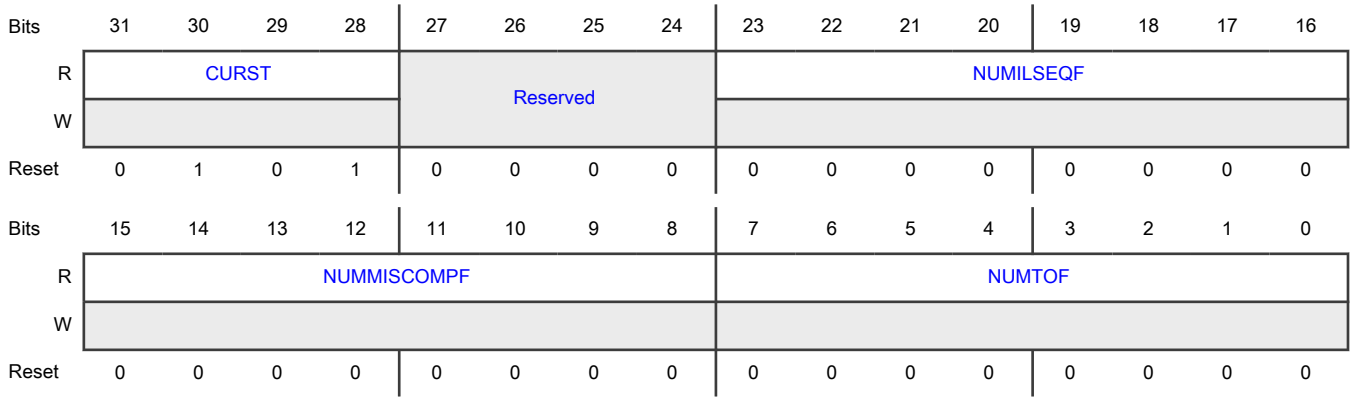
39.5.1.1.5 Status 1 (STATUS)

The Status 1 register (STATUS) holds the current module status and fault counts. The fields in STATUS are reset only by POR.

Offset

Register	Offset
STATUS	10h

Diagram



Fields

Field	Description
31-28 CURST	Current State Indicates the current state of the module. The only legal states are: • 0x5 IDLE • 0xA ACTIVE Any other value generates a STATE fault. Resets to 0x5 = IDLE.
27-24 —	Reserved
23-16 NUMILSEQF	Number of SEQUENCE faults since the last POR Resets to 0. Will not increment past 0xFF.
15-8 NUMMISCOMPF	Number of MISCOMPARE faults since the last POR Resets to 0. Will not increment past 0xFF.
7-0 NUMTOF	Number of TIMEOUT faults since the last POR Resets to 0. Will not increment past 0xFF.

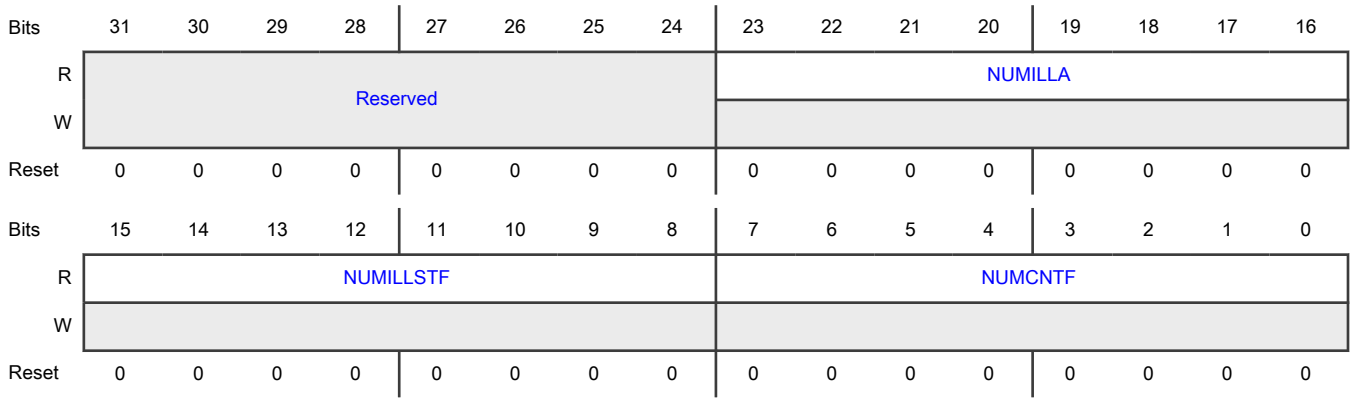
39.5.1.1.6 Status 2 (STATUS2)

Status 2 register (STATUS2) holds additional fault counts. The fields in STATUS2 are reset only by POR.

Offset

Register	Offset
STATUS2	14h

Diagram



Fields

Field	Description
31-24 —	Reserved
23-16 NUMILLA	Number of ADDRESS faults since the last POR Resets to 0. Will not increment past 0xFF.
15-8 NUMILLSTF	Number of STATE faults since the last POR Resets to 0. Will not increment past 0xFF.
7-0 NUMCNTF	Number of CONTROL faults since the last POR Resets to 0. Will not increment past 0xFF.

39.5.1.1.7 Flags (FLAGS)

The Flags register (FLAGS) contains the fault detection flags, as well as a POR event flag. The fault detection flags may generate a reset or an interrupt as configured in the CONTROL register.

The FLAGS fields retain their value through a system reset, including one generated by the CDOG. However, FLAGS is reset by a POR.

Offset

Register	Offset
FLAGS	18h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															POR_ FLAG
Reset	0	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 W	Reserved										ADDR _FL...	STATE _F...	CNT_ FLAG	SEQ_ FLAG	MISC OM_...	TO_ FLAG
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-17	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	Read value is undefined, only zero should be written.
16 POR_FLAG	Power-on reset flag POR_FLAG is set by a Power-on reset event. 0 - A Power-on reset event has not occurred 1 - A Power-on reset event has occurred
15-6 —	Reserved Read value is undefined, only zero should be written.
5 ADDR_FLAG	ADDRESS fault flag Hardware sets ADDR_FLAG when CDOG detects an ADDRESS fault. 0 - An ADDRESS fault has not occurred 1 - An ADDRESS fault has occurred
4 STATE_FLAG	STATE fault flag Hardware sets STATE_FLAG when CDOG detects a STATE fault. 0 - A STATE fault has not occurred 1 - A STATE fault has occurred
3 CNT_FLAG	CONTROL fault flag Hardware sets CNT_FLAG when CDOG detects a CONTROL fault. 0 - A CONTROL fault has not occurred 1 - A CONTROL fault has occurred
2 SEQ_FLAG	SEQUENCE fault flag Hardware sets the SEQ_FLAG when CDOG detects a SEQUENCE fault. 0 - A SEQUENCE fault has not occurred 1 - A SEQUENCE fault has occurred
1 MISCOM_FLAG	MISCOMPARE fault flag Hardware sets MISCOM_FLAG when CDOG detects a MISCOMPARE fault. 0 - A MISCOMPARE fault has not occurred 1 - A MISCOMPARE fault has occurred
0 TO_FLAG	TIMEOUT fault flag Hardware sets TO_FLAG when CDOG detects a TIMEOUT fault. 0 - A TIMEOUT fault has not occurred 1 - A TIMEOUT fault has occurred

39.5.1.1.8 Persistent Data Storage (PERSISTENT)

The Persistent Data Storage register (PERSISTENT) provides an application with 32 bits of scratchpad storage for information that needs to persist through resets other than a Power-On Reset (POR).

Offset

Register	Offset
PERSISTENT	1Ch

Diagram

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

Fields

Field	Description
31-0 PERSIS	Persistent Storage A write value to PERSIS remains unchanged after any reset other than a POR.

39.5.1.1.9 START Command (START)

Writing to the Start Command register (START) issues a Start command:

- The value written to STRT is loaded into the Secure Counter start value (SECURE_COUNTER).
- The RELOAD value is loaded into the Instruction Timer.
- The module enters the ACTIVE state in which the Instruction Timer counts down on every clock.

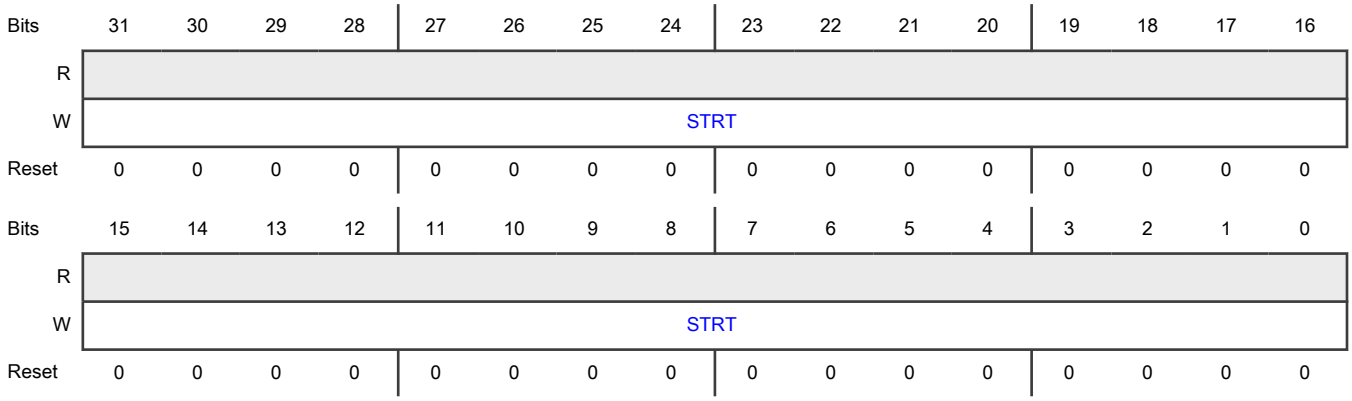
NOTE

Write to START only during the IDLE state, and only after writing to the RELOAD register. A SEQUENCE fault is generated if this specified sequence is not followed.

Offset

Register	Offset
START	20h

Diagram



Fields

Field	Description
31-0 STRT	Start command The value written to STRT is loaded into the Secure Counter start value (SECURE_COUNTER).

39.5.1.1.10 STOP Command (STOP)

Writing to the Stop Command register (STOP) issues a Stop command:

- The Instruction Timer is stopped.
- The value written to STP is compared with the current value of the Secure Counter.

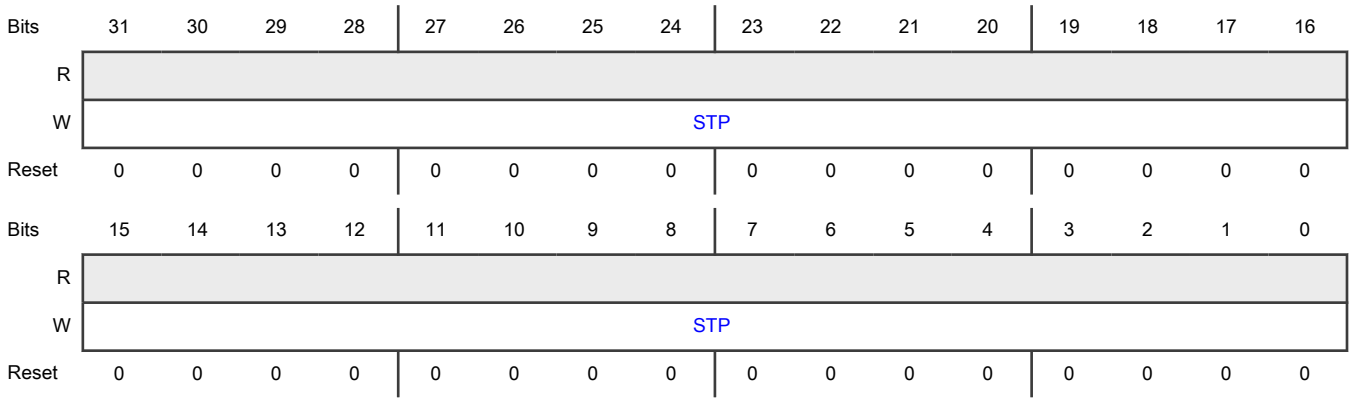
NOTE

Write to STOP only during the ACTIVE state.

Offset

Register	Offset
STOP	24h

Diagram



Fields

Field	Description
31-0 STP	Stop command The value written to STP is compared with the current value of the Secure Counter.

39.5.1.1.11 RESTART Command (RESTART)

Writing to the Restart Command register (RESTART) issues a Restart command:

- The value written to RSTRT is compared with the current value of the Secure Counter.
- If the values match, the Instruction Timer is reloaded (with the RELOAD value) and starts counting down again.

NOTE

Can only be written during ACTIVE state.

Offset

Register	Offset
RESTART	28h

Diagram

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

Fields

Field	Description
31-0 RSTRT	Restart command Write register for issuing the RESTART command

39.5.1.1.12 ADD Command (ADD)

Writing to the Add Command register (ADD) issues an Add command: The value written to AD is added to the current value of the Secure Counter

NOTE

Can only be written during ACTIVE state.

Offset

Register	Offset
ADD	2Ch

Diagram

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

Fields

Field	Description
31-0 AD	ADD Write Value The value written to AD is added to the current value of the Secure Counter.

39.5.1.1.13 ADD1 Command (ADD1)

Writing to the ADD1 Command register (ADD1) issues an ADD1 command: The value 1 is added to the current value of the Secure Counter.

NOTE

Can only be written during ACTIVE state.

Offset

Register	Offset
ADD1	30h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	AD1															
Reset	0	0	0	0	0	0	0	0	0	0	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	AD1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 AD1	ADD 1 The value written to AD1 is ignored.

39.5.1.1.14 ADD16 Command (ADD16)

Writing to the ADD16 Command register (ADD16) issues an ADD16 command: The value 16 is added to the current value of the Secure Counter.

NOTE

Can only be written during ACTIVE state.

Offset

Register	Offset
ADD16	34h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	AD16															
Reset	0	0	0	0	0	0	0	0	0	0	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	AD16															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 AD16	ADD 16 The value written to AD16 is ignored.

39.5.1.1.15 ADD256 Command (ADD256)

Writing to the ADD256 Command register (ADD256) issues an ADD256 command: The value 256 is added to the current value of the Secure Counter.

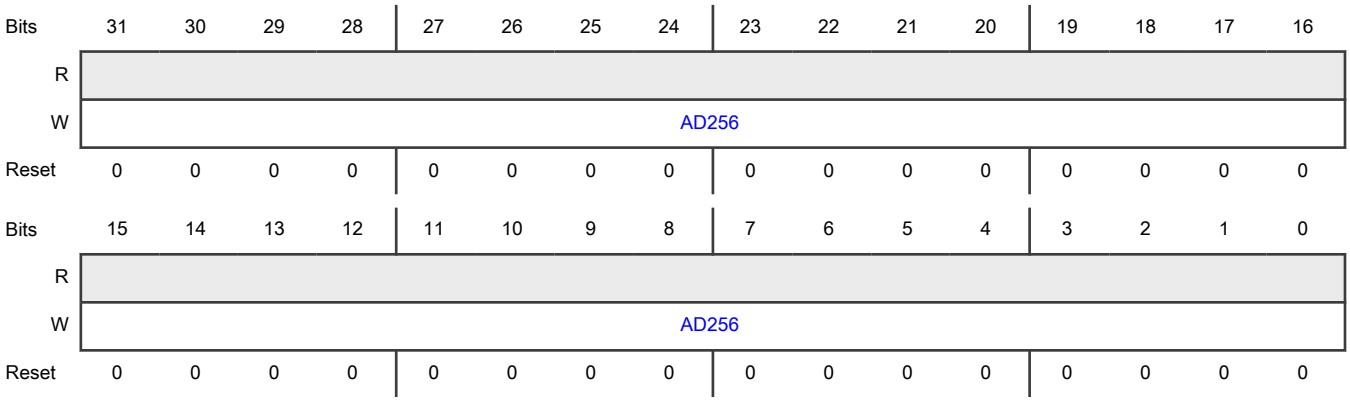
NOTE

Can only be written during ACTIVE state.

Offset

Register	Offset
ADD256	38h

Diagram



Fields

Field	Description
31-0 AD256	ADD 256 The value written to AD256 is ignored.

39.5.1.1.16 SUB Command (SUB)

Writing to the SUB Command register (SUB) issues a SUB command: The value written to S0B is subtracted from the current value of the Secure Counter.

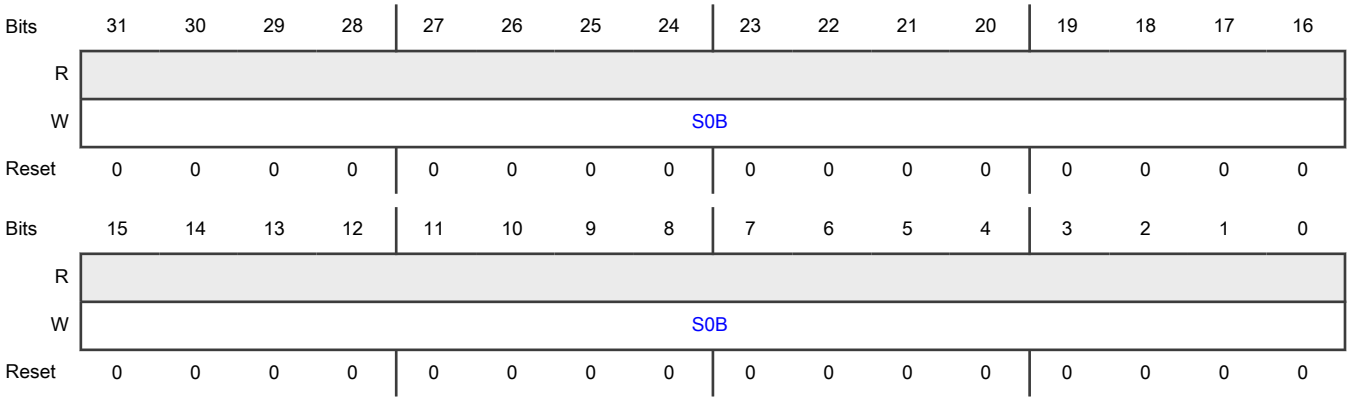
NOTE

Can only be written during ACTIVE state.

Offset

Register	Offset
SUB	3Ch

Diagram



Fields

Field	Description
31-0 S0B	Subtract Write Value The value written to SOB is subtracted from the current value of the Secure Counter.

39.5.1.1.17 SUB1 Command (SUB1)

Writing to the SUB1 Command register (SUB1) issues an SUB1 command: The value 1 is subtracted from the current value of the Secure Counter.

NOTE

Can only be written during ACTIVE state.

Offset

Register	Offset
SUB1	40h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	S1B															
Reset	0	0	0	0	0	0	0	0	0	0	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	S1B															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 S1B	Subtract 1 The value written to S1B is ignored.

39.5.1.1.18 SUB16 Command (SUB16)

Writing to the SUB16 Command register (SUB16) issues an SUB16 command: The value 16 is substracted from the current value of the Secure Counter.

NOTE

Can only be written during ACTIVE state.

Offset

Register	Offset
SUB16	44h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	SB16															
Reset	0	0	0	0	0	0	0	0	0	0	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	SB16															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 SB16	Subtract 16 The value written to SB16 is ignored.

39.5.1.1.19 SUB256 Command (SUB256)

Writing to the SUB256 Command register (SUB256) issues an SUB256 command: The value 256 is subtracted from the current value of the Secure Counter.

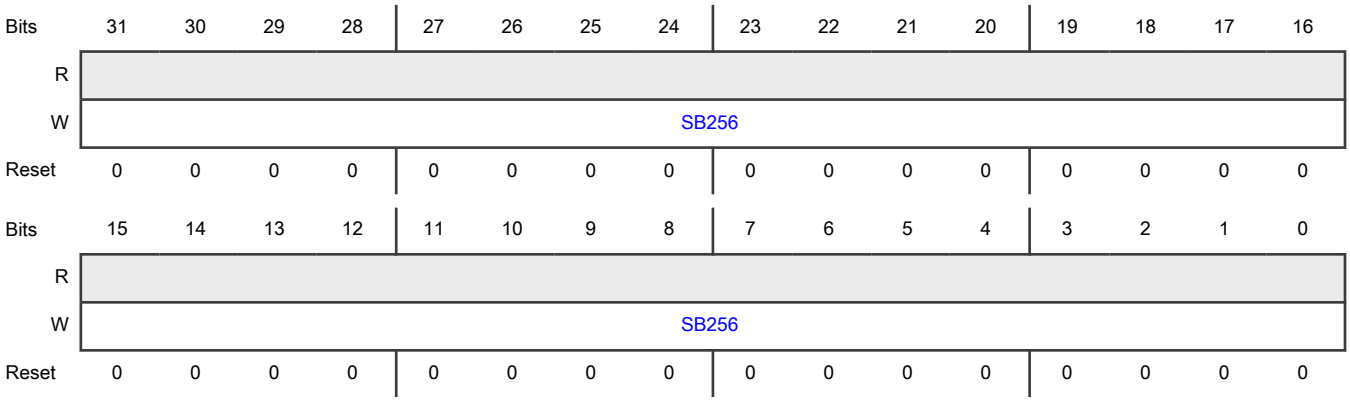
NOTE

Can only be written during ACTIVE state.

Offset

Register	Offset
SUB256	48h

Diagram



Fields

Field	Description
31-0 SB256	Subtract 256 The value written to SB256 is ignored.

Chapter 40
USB Full Speed Host and Device Controller

40.1 Chip-specific USB2.0 FS Host and Device Controller information

Table 333. Reference links to related information

Topic	Related module	Reference
Full description	USBFS	USB2.0 FS Host and Device Controller
System memory map		
Clocking		Clock generation
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

40.1.1 Module instances

This device has two instances of the USBFS module, USBFSD and USBFSH.

40.1.2 Initialization

Initial configuration of the USB0 device controller:

- Configure USB0 signals in the IOCON registers (See IOCON and Pin Multiplexing for more details).
- Enable clock to the USB Device controller register interface through AHBCLKCTRL1[USB0_DIV] bit (See SYSCON).
- Enable port mode configuration by setting AHBCLKCTRL2[USB0_HOSTS] and AHBCLKCTRL2[USB0_HOSTM] bits. Set PORTMODE[DEV_ENABLE] to enable device controller on the USB0 port. Once configured, to save power, clear AHBCLKCTRL2[USB0_HOSTS] and AHBCLKCTRL2[USB0_HOSTM] bits (See SYSCON).
- USB0 Device controller can be reset by toggling PRESETCTRL1[USB0_DEV_RST] (See SYSCON).
- USB0 device controller has two interrupt sources allocated in the NVIC interrupt sources: a general interrupt, USB0, and an activity interrupt, USB0_NEEDCLK. See NVIC[ICPR0]. Clear pending interrupts before enabling them.
- Configure the USB0 main clock.
- Configure the USB0 wake-up signal if needed.

The USB host controller is configured using the following registers:

- Select either PLL1, USB PLL, or FRO clock output as the USB0 clock for full speed operation. The clock must be 48 MHz. Configure the CPU clock to a minimum frequency of 12 MHz. See SYSCON[USB0CLKSEL] and SYSCON[USB0CLKDIV] for more details.

  Set AHBCLKCTRL2[USB0_HOSTS] and AHBCLKCTRL2[USB0_HOSTM] bits.
- Clear PORTMODE[DEV_ENABLE] bit, to ensure that the port is controlled by the USB0 host block. Set PORTMODE[ID_EN] to enable ID pin pull-up.
- Configure USB0 signals in the IOCON registers (See IOCON and Pin Multiplexing for more details).
- Reset USB0 host AHB master and slave by setting PRESETCTRL2[USB0_HOSTM_RST] and PRESETCTRL2[USB0_HOSTS_RST] bits to 1 (See SYSCON).
- Activity on the USB bus port can wake up the chip from deep-sleep mode. .
- USB0 host controller has two interrupt sources allocated in the NVIC interrupt source table: a general interrupt, USB0, and an activity interrupt, USB0_NEEDCLK. See NVIC[ICPR0].

40.1.3 Clocking

The USB device controller has the following clock connections:

- USB main clock: The USB main clock is a 48 MHz clock used for USB functions (see USB clock source select register and USB full-speed clock divider register). If the FRO is used as the USB clock source, it can be configured to adjust automatically to the USB bus rate, see FRO192M control register. The FRO enables crystal-less operation in USB Device mode, see technical note TN00063 for more details.
- CPU clock: The minimum frequency of the CPU clock is 12 MHz when the USB device controller is receiving or transmitting USB packets.
- PLL (PLL1) is added to provide 48 MHz accurate clock-source for USB-FS host.

40.1.4 USB0 Device wake-up

40.1.4.1 Waking up from deep-sleep mode on USB activity

To allow the chip to wake up from deep-sleep mode on USB activity, complete the following steps:

1. Set DEVCMDSTAT[FORCE_NEEDCLK] bit to 0 (default), to enable automatic control of the DEV_NEEDCLK signal.
2. Wait until the USB device is suspended by polling the DEVCMDSTAT[DSUS] bit (DSUS = 1).
3. The DEV_NEEDCLK signal will be de-asserted after another 2 ms. Poll the USB0NEEDCLKSTAT register until the DEV_NEEDCLK status bit is 0.
4. Clear pending USB0_NEEDCLK activity/wake-up interrupt before enabling it. Enable USB0_NEEDCLK in the NVIC.
5. SetUSB0NEEDCLKCTRL[POL_FS_DEV_NEEDCLK] to trigger USB activity wake-up interrupt on the rising edge of the DEV_NEEDCLK signal.
6. Enable the wake-up from deep-sleep mode on the USB activity interrupt via the POWER_EnterDeepSleep low power API.
7. Enter deep-sleep mode via the power API, see Power control API.

The chip will automatically wake up and resume execution on USB activity.

40.1.4.2 Remote wake-up

To issue a remote wake-up when the USB activity is suspended, complete the following steps:

1. Set DEVCMDSTAT[FORCE_NEEDCLK] bit to 0 to enable automatic control of the DEV_NEEDCLK signal.
2. For a remote wake-up, turn on the USB clock and enable the USB clock source.
3. Force the USB clock on by writing a 1 to DEVCMDSTAT[FORCE_NEEDCLK].
4. Write a 0 to the DEVCMDSTAT[DSUS] bit.
5. Wait until the USB leaves the suspend state by polling the DEVCMDSTAT[DSUS] bit.
6. Clear the DEVCMDSTAT[FORCE_NEEDCLK] bit to enable automatic USB clock control.

40.1.5 USB0 host wake-up

To allow the chip to wake up from deep-sleep mode on USB activity, complete the following steps:

1. Send GET_STATUS command to the connected device and check if it is capable of REMOTE_WAKEUP. See "Get Status" command section in "USB Device Framework" in the USB 2.0 Specification.
2. Check HCRHPORTSTATUS register to see if the device is connected. Set HCRHPORTSTATUS[PSS] to 1 to suspend the port. Set HCCONTROL[HCFS] bits to 11b to put the host controller into SUSPEND state and then set 3 ms delay for the device to suspend.

3. Set HCRHSTATUS[DRWE] bit to enable remote wake-up.
4. Poll the USB0NEEDCLKSTAT register until the HOST_NEED_CLK bit is 0 (See SYSCON).
5. Set USB0NEEDCLKCTRL[POL_FS_HOST_NEED_CLK] bit to 1 to trigger the USB activity wake-up interrupt (USB0_NEEDCLK) on the rising edge of the USB0 host NEEDCLK signal.
6. Enable the wake-up from deep-sleep mode on the USB activity interrupt via the POWER_EnterDeepSleep low power API.
7. Clear pending USB0 activity interrupt, USB0_NEEDCLK, before enabling it.
8. Enter deep-sleep mode via power API, see Power control API. The chip will automatically wake-up and resume execution on USB activity.
9. Re-initialize the USB host controller after the USB0_NEEDCLK interrupt is invoked.

40.2 Overview

This section describes the full-speed USB 2.0 controller in host and device mode. The Universal Serial Bus (USB) is a four-wire bus that supports communication between a host and one or more (up to 127) peripherals.

The host controller allocates the USB bandwidth to attached devices through a token based protocol. The bus supports hot plugging, un-plugging, and dynamic configuration of the devices. All transactions are initiated by the host controller. The host controller enables data exchange with various USB devices attached to the bus.

40.2.1 Block diagram

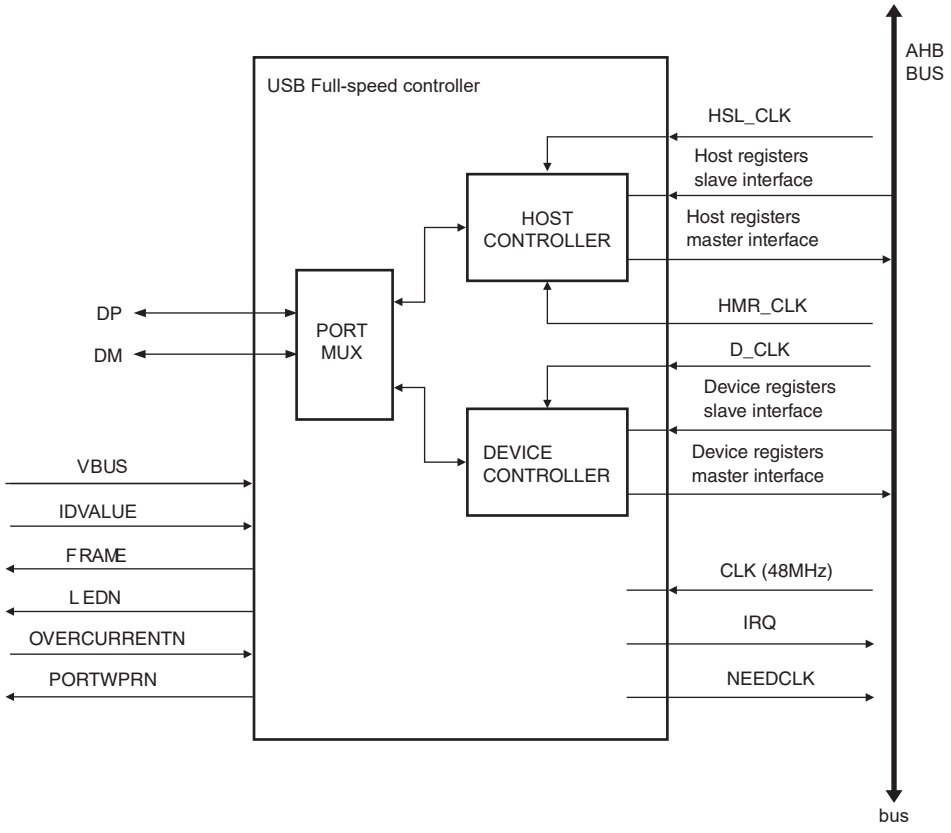


Figure 158. USB2.0 full speed Host/Device Controller block diagram

40.2.2 Features

USB2.0 full-speed Device Controller supports following:

- Crystal-less operation using the software library
- Ten physical (five logical) endpoints including the control endpoints.
  - Control transfers are used to configure the device
  - Interrupt transfers are used for periodic data transfer
  - Bulk transfers are used when the latency of transfer is not critical
  - Isochronous transfers with guaranteed delivery time but no error correction
- Full-speed (12 Mb/s) data exchange with a USB host controller
- Single and double buffering
- Each non-control endpoint supports bulk, interrupt, or isochronous endpoint types
- Wake-up from deep-sleep mode on USB activity and remote wake-up
- SoftConnect internally
- Link Power Management(LPM)

USB2.0 full speed Host Controller supports following:

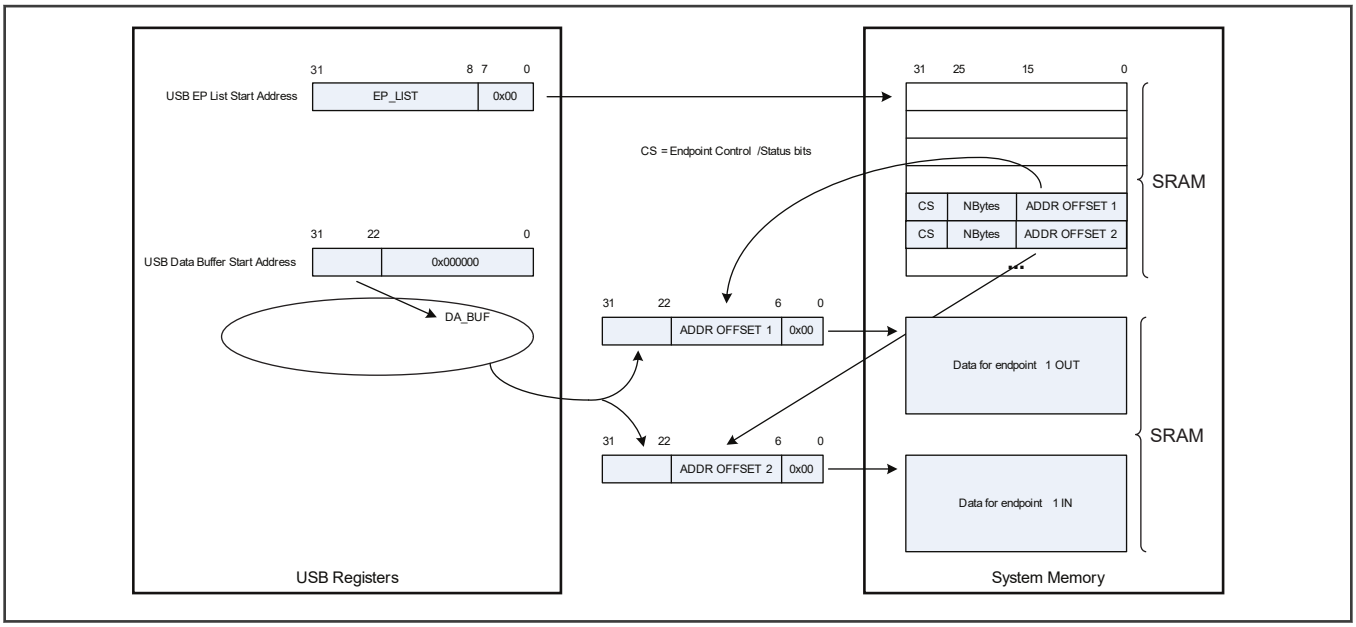
- Serial interface engine and DMA controller
- Schedules transactions in 1 ms frame
- OHCI (Open Host Controller Interface) compliant
- OpenHCI specifies the operation and interface of the USB host controller and SW drive
- The host controller has four USB states visible to the SW driver:
  - USBOperational: Process lists and generate SOF token
  - USBReset: Forces reset signaling on the bus, SOF disable
  - USBSuspend: Monitor USB for wake-up activit
  - USBResume: Forces resume signaling on the bus
- HCCA register points to interrupt and isochronous descriptors list
- ControlHeadED and BulkHeadED registers point to control and bulk descriptors list

40.3 Functional description

The following sections describe functional details of the USB 2.0 Full Speed Controller module.

40.3.1 Software interface

The software interface of the USB consists of a registers and the format definitions for the endpoint descriptors. See the OHCI specification for more details.



40.3.2 Fixed endpoint configuration

Table 334 shows the supported endpoint configurations. The packet size is configurable up to the maximum value for each type of end point.

Table 334. Fixed endpoint configuration

Logical endpoint	Physical endpoint	Endpoint type	Direction	Max packet size (byte)	Double buffer
0	0	Control	Out	64	No
0	1	Control	In	64	No
1	2	Interrupt/Bulk/Isochronous	Out	64/64/1023	Yes
1	3	Interrupt/Bulk/Isochronous	In	64/64/1023	Yes
2	4	Interrupt/Bulk/Isochronous	Out	64/64/1023	Yes
2	5	Interrupt/Bulk/Isochronous	In	64/64/1023	Yes
3	6	Interrupt/Bulk/Isochronous	Out	64/64/1023	Yes
3	7	Interrupt/Bulk/Isochronous	In	64/64/1023	Yes
4	8	Interrupt/Bulk/Isochronous	Out	64/64/1023	Yes
4	9	Interrupt/Bulk/Isochronous	In	64/64/1023	Yes

40.3.3 Soft connect

The softConnect signal is implemented internally. An external pull-up resistor between USB_DP and VDD is not necessary. Software can control the pull-up by setting the DCON bit in the DEVCMDSTAT register. If the DCON bit is set to 1, the USB_DP line is pulled up to VDD through an internal 1.5 KOhm pull-up resistor.

40.3.4 Interrupts

The USB controller has two interrupts, a general USB interrupt (USB0) and a USB activity wake-up interrupt (USB0_NEEDCLK). A general interrupt is generated by the hardware if both the interrupt status bit and the corresponding interrupt enable bit are set. The interrupt status bit is set by hardware if the interrupt condition occurs (irrespective of the interrupt enable bit setting).

40.3.5 Suspend and resume

The USB protocol insists on power management by the USB device. It becomes even more important if the device draws power from the bus (bus-powered device). The following constraints should be met by the bus-powered device:

- A device in the non-configured state should draw a maximum of 100 mA from the USB bus.
- A configured device can draw only up to what is specified in the max power field of the configuration descriptor. The maximum value is 500 mA.
- A suspended device should draw a maximum of 500 µA.

A device will go into the L2 suspend state if there is no activity on the USB bus for more than 3 ms. A suspended device wakes up if there is transmission from the host (host-initiated wake up). The USB controller also supports software initiated remote wake-up (device-initiated wake up). To initiate remote wake-up, the software on the device must enable all clocks and clear the DSUS in DEVCMDSTAT bit. It will cause the hardware to generate a remote wake-up signal upstream.

The USB controller supports Link Power Management (LPM). Link power management defines an additional link power management state, L1 that supplements the existing L2 state by utilizing most of the existing suspend/resume infrastructure but provides much faster transitional latencies between L1 and L0 (ON).

The assertion of the USB suspend signal indicates that there was no activity on the USB bus for the last 3 ms. At this time an interrupt is sent to the processor on which the software can start preparing the device for a suspended state.

If there is no activity for the next 2 ms, the USB DEV_NEEDCLK signal will go LOW. This indicates that the USB main clock can be switched off.

When any activity is detected on the USB bus, the DEV_NEEDCLK signal is activated. This process is fully combinatorial and therefore, USB main clock is not required to activate the DEV_NEEDCLK signal. See Chip-specific USB information for more details on waking up from suspend mode.

40.3.6 Frame toggle output

The USB0_FRAME output pin reflects the 1 kHz clock derived from the incoming start of frame tokens sent by the USB host.

40.3.7 Endpoint command/status list

The figure below gives an overview on how the list of fixed endpoints is organized in memory. The USB EP command/status list start register points to the start of the list that contains all the endpoint information in memory.

USB EP Command/Status FIFO start

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Offset
A	R	S	TR	TV	R	EP0 OUT Buffer NBytes															EP0 OUT Buffer Address Offset												0x00
R	R	R	R	R	R	Reserved															SETUP Bytes Buffer Address Offset												0x04
A	R	S	TR	TV	R	EP0 IN Buffer NBytes															EP0 IN Buffer Address Offset												0x08
R	R	R	R	R	R	Reserved															Reserved												0x0C
A	D	S	TR	RF TV	T	EP1 OUT Buffer 0 NBytes															EP1 OUT Buffer 0 Address Offset												0x10
A	D	S	TR	RF TV	T	EP1 OUT Buffer 1 NBytes															EP1 OUT Buffer 1 Address Offset												0x14
A	D	S	TR	RF TV	T	EP1 IN Buffer 0 NBytes															EP1 IN Buffer 0 Address Offset												0x18
A	D	S	TR	RF TV	T	EP1 IN Buffer 1 NBytes															EP1 IN Buffer Address Offset												0x1C
A	D	S	TR	RF TV	T	EP2 OUT Buffer 0 NBytes															EP2 OUT Buffer 0 Address Offset												0x20
A	D	S	TR	RF TV	T	EP2 OUT Buffer 1 NBytes															EP2 OUT Buffer 1 Address Offset												0x24
A	D	S	TR	RF TV	T	EP2 IN Buffer 0 NBytes															EP2 IN Buffer Address Offset												0x28
A	D	S	TR	RF TV	T	EP2 IN Buffer 1 NBytes															EP2 IN Buffer Address Offset												0x2C
...																																
A	D	S	TR	RF TV	T	EP5 OUT Buffer 0 NBytes															EP5 OUT Buffer 0 Address Offset												0x50
A	D	S	TR	RF TV	T	EP5 OUT Buffer 1 NBytes															EP5 OUT Buffer 1 Address Offset												0x54
A	D	S	TR	RF TV	T	EP5 IN Buffer 0 NBytes															EP5 IN Buffer 0 Address Offset												0x58
A	D	S	TR	RF TV	T	EP5 IN Buffer 1 NBytes															EP5 IN Buffer 1 Address Offset												0x5C

aaa-021724

Figure 159. Endpoint command/status list

Table 335. Endpoint command/status bit definitions

Symbol	Access	Description
A	R/W	Active The buffer is enabled. Hardware can use the buffer to store received OUT data or to transmit data on the IN endpoint. Software can only set this bit to 1. As long as this bit is set to one, software is not allowed to update any of the values in this 32-bit word. In case software wants to deactivate the buffer, it must write a 1 to the corresponding "skip" bit in the USB endpoint skip register. Hardware can only write this bit to 0. It will do this when it receives a short packet or when the NBytes field transitions to 0 or when software has written a 1 to the "skip" bit. When data is to be received for the OUT endpoint, the software sets the active bit to 1 and programs the NBytes field to the maximum number of bytes it can receive. When data must be transmitted for an IN endpoint, the software sets the active bit to 1 and programs the NBytes field to the number of bytes that must be transmitted.
D	R/W	Disabled 0: The selected endpoint is enabled. 1: The selected endpoint is disabled. If a USB token is received for an endpoint that has the disabled bit set, hardware will ignore the token and not return any data or handshake. When a bus reset is received, software must set the disable bit of all endpoints to 1.

Table continues on the next page...

Table 335. Endpoint command/status bit definitions (continued)

Symbol	Access	Description
		Software can only modify this bit when the active bit is 0.
S	R/W	Stall 0: The selected endpoint is not stalled. 1: The selected endpoint is stalled. The active bit has always a higher priority than the stall bit. This means that a Stall handshake is only sent when the active bit is 0 and the stall bit is 1. Software can only modify this bit when the active bit is 0.
TR	R/W	Toggle reset When software sets this bit to 1, the hardware will set the toggle value equal to the value indicated in the "toggle value" (TV) bit. For the control endpoint 0, this is not needed to be used because the hardware resets the endpoint toggle to one for both directions when a setup token is received. For the other endpoints, the toggle can only be reset to 0 when the endpoint is reset.
RF / TV	R/W	Rate feedback mode / Toggle value For bulk endpoints and isochronous endpoints this bit is reserved and must be set to 0. For the control endpoint 0 this bit is used as the toggle value. When the toggle reset bit is set, the data toggle is updated with the value programmed in this bit. When the endpoint is used as an interrupt endpoint, it can be set to the following values. 0: Interrupt endpoint in ‘toggle mode’. 1: Interrupt endpoint in ‘rate feedback mode’. This means that the data toggle is fixed to 0 for all data packets. When the interrupt endpoint is in ‘rate feedback mode’, the TR bit must always be set to 0.
T	R/W	Endpoint type 0: Generic endpoint. The endpoint is configured as a bulk or interrupt endpoint. 1: Isochronous endpoint.
NBytes	R/W	For OUT endpoints this is the number of bytes that can be received in this buffer. For IN endpoints this is the number of bytes that must be transmitted. HW decrements this value with the packet size every time when a packet is successfully transferred. Note: If a short packet is received on an OUT endpoint, the active bit will be cleared and the NBytes value indicates the remaining buffer space that is not used. Software calculates the received number of bytes by subtracting the remaining NBytes from the programmed value.
Address Offset	R/W	Bits 21 to 6 of the buffer start address. The address offset is updated by hardware after each successful reception/transmission of a packet. Hardware increments the original value with the integer value when the packet size is divided by 64. Examples:

Table continues on the next page...

Table 335. Endpoint command/status bit definitions (continued)

Symbol	Access	Description
		• If an isochronous packet of 200 bytes is successfully received, the address offset is incremented by 3. • If a packet of 64 bytes is successfully received, the address offset is incremented by 1. • If a packet of less than 64 bytes is received, the address offset is not incremented.

When receiving a SETUP token for endpoint 0, the hardware only reads the SETUP bytes buffer address offset to determine where to store the received SETUP bytes. The hardware ignores all other fields. In case the SETUP stage contains more than eight bytes, only the first eight bytes are written to memory. No more than eight bytes should be sent by the USB compliant host during the SETUP stage.

For EP0 transfers, the hardware performs auto handshaking as long as the ACTIVE bit is set in EP0_IN/OUT command list. Unlike other endpoints, the hardware will not clear the ACTIVE bit after the transfer is completed. Thus, the software should manually clear the bit whenever it receives a new setup packet and set it only after it has queued the data for control transfer.

40.3.8 Control endpoint 0

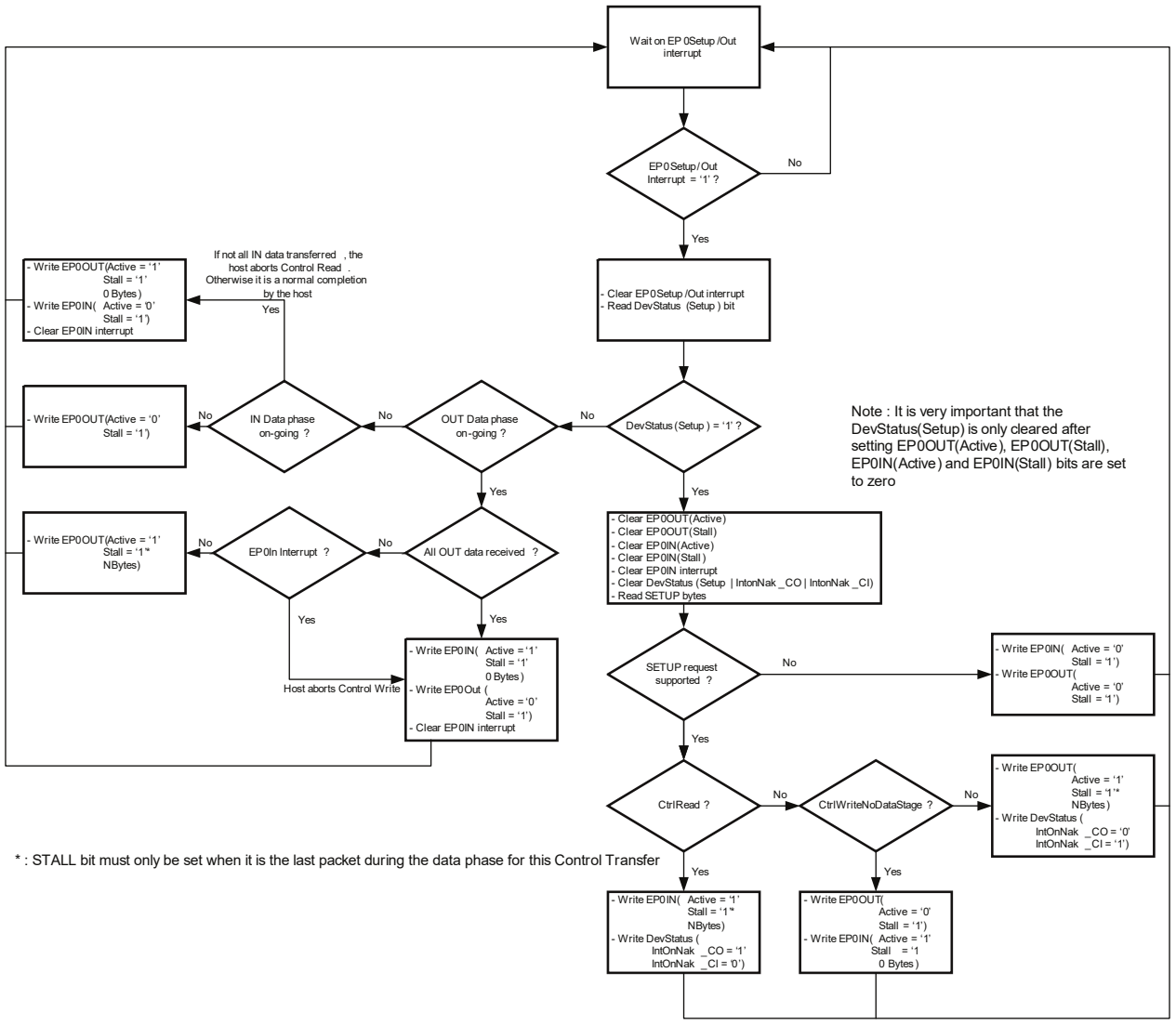
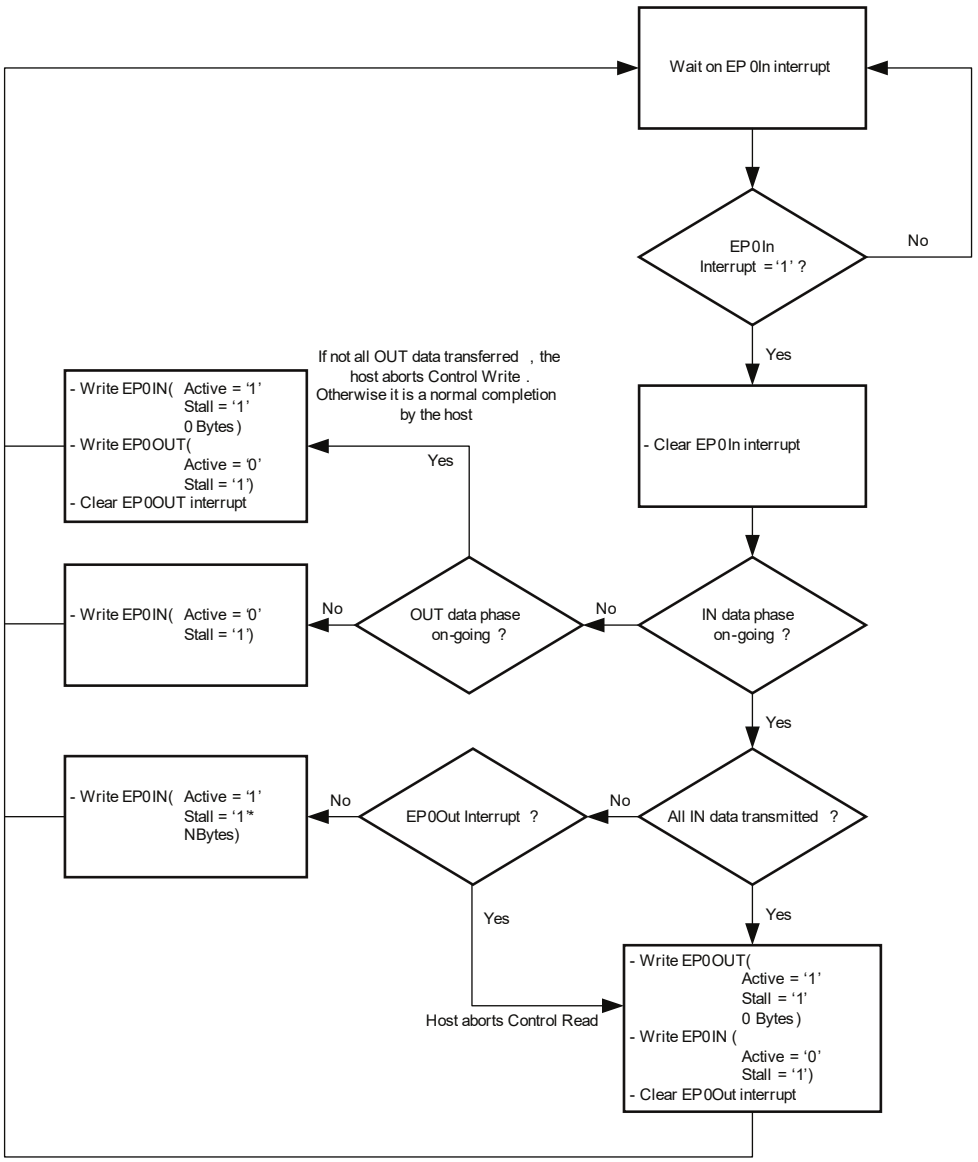


Figure 160. Flowchart of control endpoint 0 - OUT



* : STALL bit must only be set when it is the last packet during the data phase for this Control Transfer

Figure 161. Flowchart of control endpoint 0 - IN

40.3.9 Generic endpoint: single buffering

To enable single buffering, the software must set the corresponding "BUF_SB bit in the "USB EP Buffer Configuration *register* to 0. In the *USB EP Buffer in use* register, the software can indicate which buffer is used in this case.

When software wants to transfer data, it programs the different bits in the endpoint command/status list entry for the desired endpoint and sets the active bit. The hardware will transmit/receive multiple packets for this endpoint until the NBytes value is equal to 0. When NBytes 0, hardware clears the active bit and sets the corresponding endpoint interrupt status bit in INTSTAT.

Software must wait until hardware has cleared the active bit to change the command/status bits in the endpoint command/status list entry. It prevents hardware from overwriting a new value programmed by software with old values that were still cached.

If software wants to disable the active bit before the hardware has finished handling the complete buffer, it can do this by setting the corresponding endpoint SKIP bit in USB endpoint skip register (EPSKIP).

40.3.10 Generic endpoint: double buffering

To enable double buffering, the software must set the corresponding *USB EP Buffer Config* bit to 1. The *USB EP Buffer in use* register indicates which buffer will be used by hardware when the next token is received.

When hardware clears the active bit of the current buffer in use, it will switch the buffer in use. Software can also force hardware to use a certain buffer by writing to the corresponding *USB EP Buffer in use* bit.

40.3.11 Special cases

40.3.11.1 Generation of a STALL handshake

Special care must be taken when programming the endpoint to send a STALL handshake. A STALL handshake is only sent in the following situations:

- The endpoint is enabled (Disabled bit = 0).
- The active bit of the endpoint is set to 0. (No packet needs to be received/transmitted for that endpoint).
- The stall bit of the endpoint is set to 1.

40.3.11.2 Clear feature (endpoint halt)

When a non-control endpoint has returned a STALL handshake, the host will send a clear feature (Endpoint Halt) for that endpoint. When the device receives this request, the endpoint must be un-stalled and the toggle bit for that endpoint must be reset to 0. To accomplish this, the software must program the following items for the endpoint that is indicated.

If the endpoint is used in single buffer mode, program the following:

- Set STALL bit (S) to 0.
- Set toggle reset bit (TR) to 1 and set toggle value bit (TV) to 0.

If the endpoint is used in double buffer mode, program the following:

- Set the STALL bit of both buffer 0 and buffer 1 to 0.
- Read the buffer in use bit for this endpoint.
- Set the toggle reset bit (TR) to 1 and set the toggle value bit (TV) to 0 for the buffer indicated by the buffer in use bit.

40.3.11.3 Set configuration

When a SetConfiguration request is received with a configuration value different from 0, the device software must enable all endpoints that will be used in this configuration and reset all the toggle values. To do so, it must generate the procedure explained in Clear feature (endpoint halt) for every endpoint that will be used in this configuration.

For all endpoints that are not used in this configuration, it must set the Disabled bit (D) to 1.

40.4 Signals

Table 336. USB Device Signals

Name	Direction	Description
VBUS	I	USB VBUS status input. When this function is not enabled via its corresponding IOCON register, it is driven LOW internally.
DP	I/O	Positive differential data.
DM	I/O	Negative differential data.

Table continues on the next page...

Table 336. USB Device Signals (continued)

Name	Direction	Description
IDVALUE	I	A-device (host role) or B-device (peripheral role) indication. Enable this function when using a micro USB receptacle to identify whether a micro-A or micro-B plug is inserted. When enabled, the pull-up on the corresponding port pin should be enabled.
FRAME	O	1 kHz clock derived from the incoming Start of Frame tokens sent by the USB host. It is an optional function.
LEDN	O	USB connection indication. It is an optional function.
PORTPWRN	-	Host only function.
OVERCURRENTN	-	Host only function.

Table 337. USB Host Signals

Pin name	Direction	Description
DP	I/O	Positive differential data.
DM	I/O	Negative differential data.
IDVALUE	I	A-device (host role) or B-device (peripheral role) indication.
PORTPWRN	O	VBUS drive signal (towards external charge pump or power management unit).
OVERCURRENTN	I	Port power fault signal indicating over-current condition; this signal monitors over-current on the USB bus (external circuitry is required to detect over-current condition).
VBUS	I	USB VBUS status input. When this function is not enabled via its corresponding IOCON register, it is driven LOW internally. Enable this pin for correct host operation. For example, it is required detect a change in the connection status.
FRAME	-	Device only function.
LEDN	-	Device only function.

40.5 Memory Map and register definition

This section includes the USB2.0 Host and Device controller module memory map and detailed descriptions of all registers.

40.5.1 USB 2.0 Full Speed Device Controller register descriptions

40.5.1.1 USBFSD memory map

USBFSD base address: 4008_4000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	USB Device Command/Status (DEVCMDSTAT)	32	See section	See section
4	USB Info (INFO)	32	See section	See section
8	USB EP Command/Status List start address (EPLISTSTART)	32	See section	See section
C	USB Data buffer start address (DATABUFSTART)	32	See section	See section
10	USB Link Power Management (LPM)	32	See section	See section
14	USB Endpoint skip (EPSKIP)	32	See section	See section
18	USB Endpoint Buffer in use (EPINUSE)	32	See section	See section
1C	USB Endpoint Buffer Configuration (EPBUFCFG)	32	See section	See section
20	USB interrupt status (INTSTAT)	32	See section	See section
24	USB interrupt enable (INTEN)	32	See section	See section
28	USB set interrupt status (INTSETSTAT)	32	See section	See section
34	USB Endpoint toggle (EPTOGGLE)	32	See section	See section

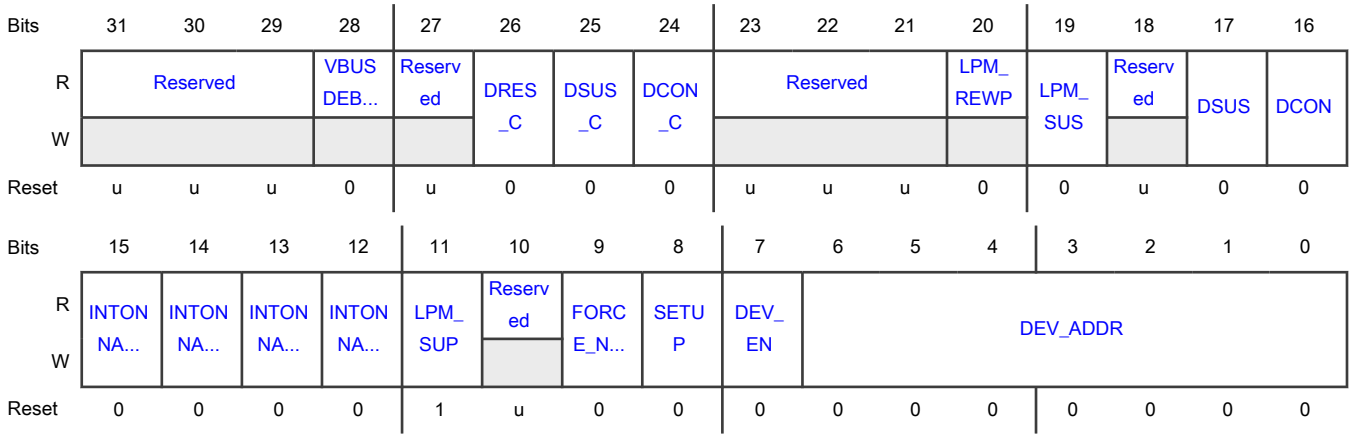
40.5.1.1.1 USB Device Command/Status (DEVCMDSTAT)

This register controls the behavior of the full-speed USB device.

Offset

Register	Offset
DEVCMDSTAT	0h

Diagram



Fields

Field	Description
31-29 —	Reserved
28 VBUSDEBOUNCED	This bit indicates if Vbus is detected or not. The bit raises immediately when Vbus becomes high. It drops to zero if Vbus is low for at least 3 ms. If this bit is high and the DCon bit is set, the Hardware will enable the pull-up resistor to signal a connect.
27 —	Reserved
26 DRES_C	Device status - reset change This bit is set when the device received a bus reset. On a bus reset the device will automatically go to the default state (unconfigured and responding to address 0). The bit is reset by writing a one to it.
25 DSUS_C	Device status - suspend change This bit is set to 1 when the suspend bit toggles. The suspend bit can toggle because: - The device goes in the suspended state - The device is disconnected - The device receives resume signaling on its upstream port. The bit is reset by writing a one to it.
24 DCON_C	Device status - connect change This bit is set when the device's pull-up resistor is disconnected because VBus disappeared. The bit is reset by writing a one to it.
23-21 —	Reserved
20 LPM_REWP	LPM Remote Wake-up Enabled by USB host Hardware sets this bit to one when the bRemoteWake bit in the LPM extended token is set to 1. Hardware will reset this bit to 0 when it receives the host initiated LPM resume, when a remote wake-up

Table continues on the next page...

Table continued from the previous page...

Field	Description
	is sent by the device or when a USB bus reset is received. Software can use this bit to check if the remote wake-up feature is enabled by the host for the LPM transaction.
19 LPM_SUS	Device status - LPM Suspend This bit represents the current LPM suspend state. It is set to 1 by Hardware when the device has acknowledged the LPM request from the USB host and the Token Retry Time of 10 ms has elapsed. When the device is in the LPM suspended state (LPM suspend bit = 1) and the software writes a zero to this bit, the device will generate a remote walk-up. Software can only write a zero to this bit when the LPM_REWP bit is set to 1. Hardware resets this bit when it receives a host initiated resume. Hardware only updates the LPM_SUS bit when the LPM_SUPP bit is equal to one.
18 —	Reserved
17 DSUS	Device status - suspend This bit indicates the current suspend state. It is set to 1 when the device hasn't seen any activity on its upstream port for more than 3 milliseconds. It is reset to 0 on any activity. When the device is suspended (Suspend bit DSUS = 1) and the software writes a 0 to it, the device will generate a remote wake-up. This will only happen when the device is connected (Connect bit = 1). When the device is not connected or not suspended, a writing a 0 has no effect. Writing a 1 never has an effect.
16 DCON	Device status - connect The connect bit must be set by SW to indicate that the device must signal a connect. The pull-up resistor on USB_DP will be enabled when this bit is set and the VBUSDEBOUNCED bit is one.
15 INTONNAK_CI	Interrupt on NAK for control IN EP 0 - Only acknowledged packets generate an interrupt 1 - Both acknowledged and NAKed packets generate interrupts.
14 INTONNAK_CO	Interrupt on NAK for control OUT EP 0 - Only acknowledged packets generate an interrupt 1 - Both acknowledged and NAKed packets generate interrupts.
13 INTONNAK_AI	Interrupt on NAK for interrupt and bulk IN EP 0 - Only acknowledged packets generate an interrupt 1 - Both acknowledged and NAKed packets generate interrupts.
12 INTONNAK_AO	Interrupt on NAK for interrupt and bulk OUT EP 0 - Only acknowledged packets generate an interrupt 1 - Both acknowledged and NAKed packets generate interrupts.
11 LPM_SUP	LPM Support 0 - LPM not supported. 1 - LPM supported.

Table continues on the next page...

Table continued from the previous page...

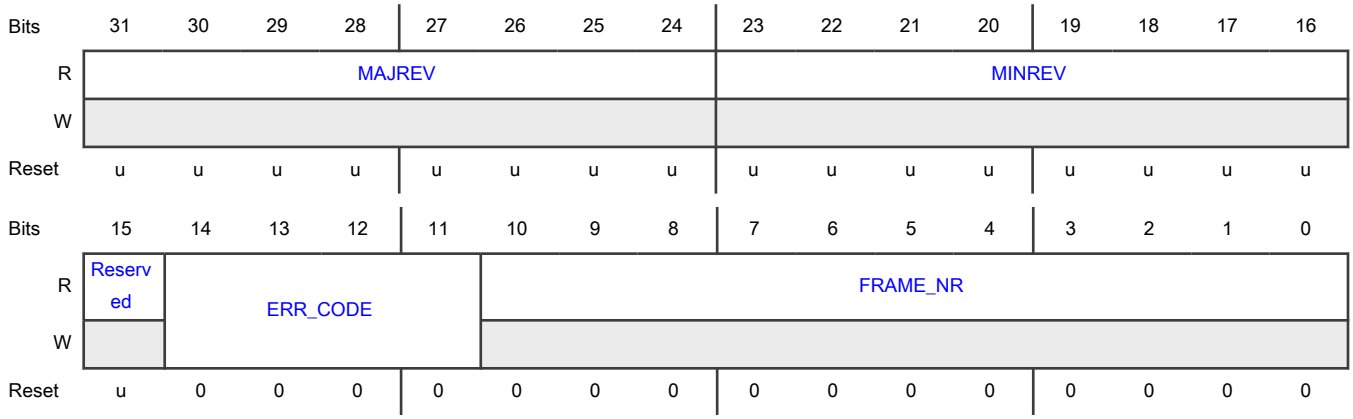
Field	Description
10 —	Reserved
9 FORCE_NEEDCLK	Forces the NEEDCLK output to always be on: 0 - USB_NEEDCLK has normal function. 1 - USB_NEEDCLK always 1. Clock will not be stopped in case of suspend.
8 SETUP	SETUP token received If a SETUP token is received and acknowledged by the device, this bit is set. As long as this bit is set all received IN and OUT tokens will be NAKed by Hardware. SW must clear this bit by writing a one. If this bit is zero, Hardware will handle the tokens to the CTRL EP0 as indicated by the CTRL EP0 IN and OUT data information programmed by software.
7 DEV_EN	USB device enable If this bit is set, the Hardware will start responding on packets for function address DEV_ADDR.
6-0 DEV_ADDR	USB device address After bus reset, the address is reset to 0x00. If the enable bit is set, the device will respond on packets for function address DEV_ADDR. When receiving a SetAddress Control Request from the USB host, software must program the new address before completing the status phase of the SetAddress Control Request.

40.5.1.1.2 USB Info (INFO)

Offset

Register	Offset
INFO	4h

Diagram



Fields

Field	Description
31-24 MAJREV	Major Revision.
23-16 MINREV	Minor Revision.
15 —	Reserved
14-11 ERR_CODE	The error code which last occurred: 0000 - No error 0001 - PID encoding error 0010 - PID unknown 0011 - Packet unexpected 0100 - Token CRC error 0101 - Data CRC error 0110 - Time out 0111 - Babble 1000 - Truncated EOP 1001 - Sent/Received NAK 1010 - Sent Stall 1011 - Overrun 1100 - Sent empty packet 1101 - Bitstuff error 1110 - Sync error 1111 - Wrong data toggle
10-0 FRAME_NR	Frame number This contains the frame number of the last successfully received SOF. In case no SOF was received by the device at the beginning of a frame, the frame number returned is that of the last successfully received SOF. In case the SOF frame number contained a CRC error, the frame number returned will be the corrupted frame number as received by the device.

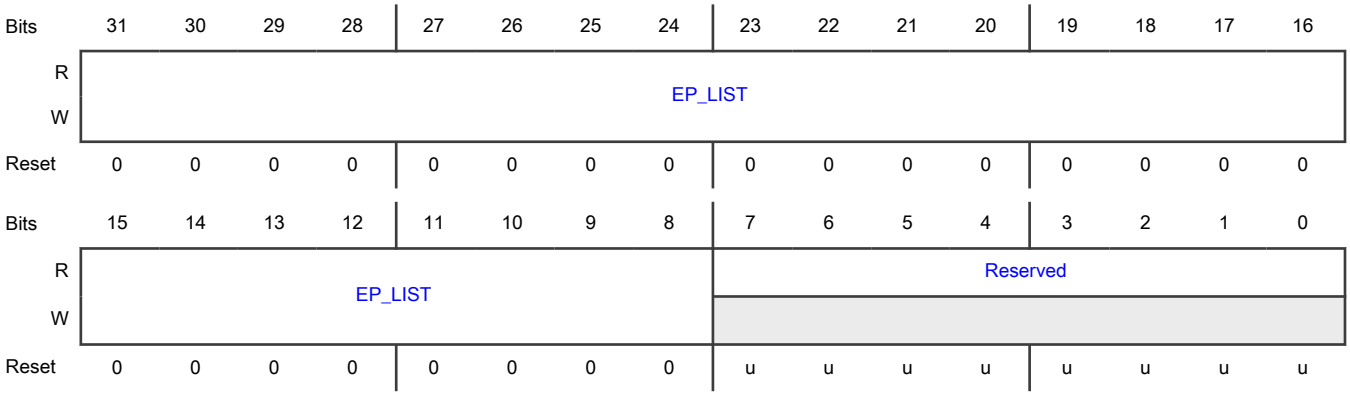
40.5.1.1.3 USB EP Command/Status List start address (EPLISTSTART)

This 32-bit register indicates the start address of the USB EP command/status list. Only a subset of these bits is programmable by software. The 8 least-significant bits are hard coded to 0 because the list must start on a 256 byte boundary. Bits 31 to 8 can be programmed by software.

Offset

Register	Offset
EPLISTSTART	8h

Diagram



Fields

Field	Description
31-8 EP_LIST	Start address of the USB EP Command/Status List.
7-0 —	Reserved

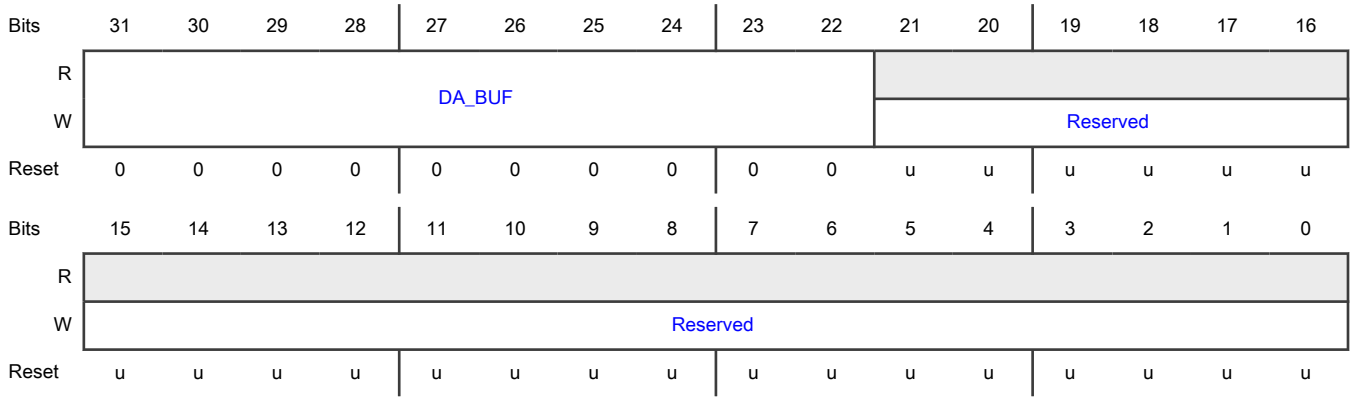
40.5.1.1.4 USB Data buffer start address (DATABUFSTART)

This register indicates the page of the AHB address where the endpoint data can be located. The 22 LSBs are fixed to 0 so that the location resides on a 4 MB boundary. The start address of each individual endpoint's buffer is an offset to the data buffer start address. The buffer address of the endpoint is set using the address offset field of the endpoint's corresponding entry in the endpoint command/status list.

Offset

Register	Offset
DATABUFSTART	Ch

Diagram



Fields

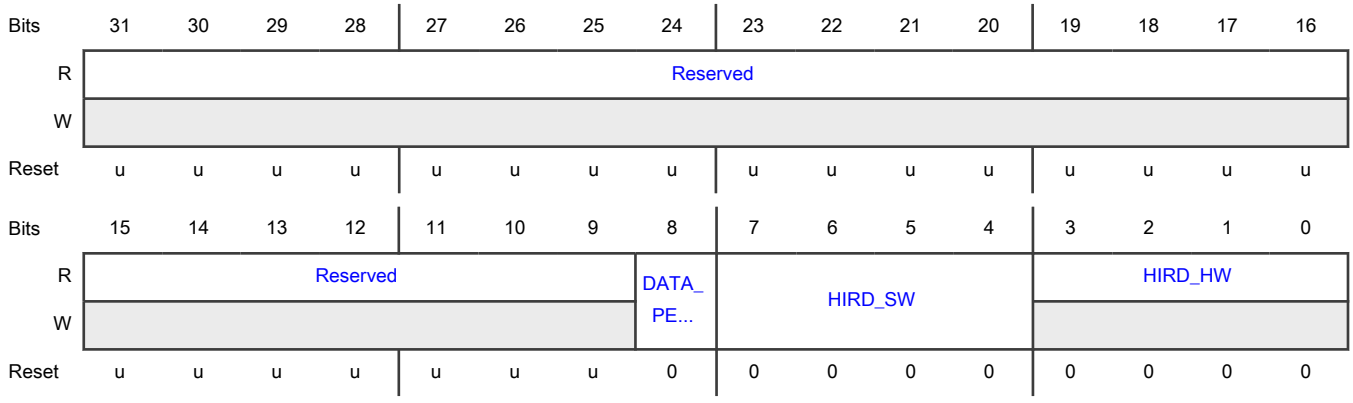
Field	Description
31-22 DA_BUF	Start address of the buffer pointer page where all endpoint data buffers are located.
21-0 —	Reserved

40.5.1.1.5 USB Link Power Management (LPM)

Offset

Register	Offset
LPM	10h

Diagram



Fields

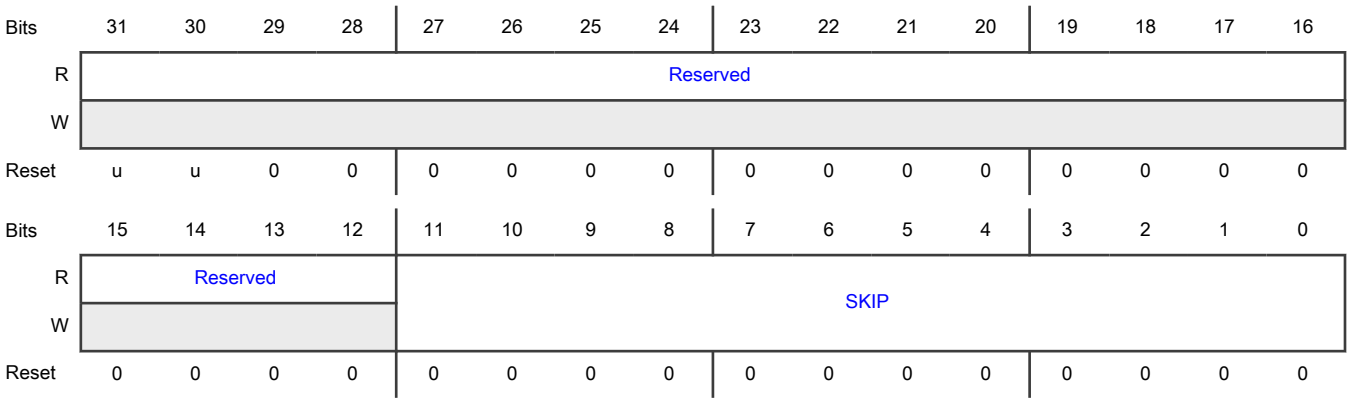
Field	Description
31-9 —	Reserved
8 DATA_PENDING	Data pending As long as this bit is set to one and LPM supported bit is set to one, Hardware will return a NYET handshake on every LPM token it receives. If LPM supported bit is set to one and this bit is zero, Hardware will return an ACK handshake on every LPM token it receives. If software has still data pending and LPM is supported, it must set this bit to 1.
7-4 HIRD_SW	Host Initiated Resume Duration - software This is the time duration required by the USB device system to come out of LPM initiated suspend after receiving the host initiated LPM resume.
3-0 HIRD_HW	Host Initiated Resume Duration - hardware This is the HIRD value from the last received LPM token.

40.5.1.1.6 USB Endpoint skip (EPSKIP)

Offset

Register	Offset
EPSKIP	14h

Diagram



Fields

Field	Description
31-12	Reserved

Table continues on the next page...

Table continued from the previous page...

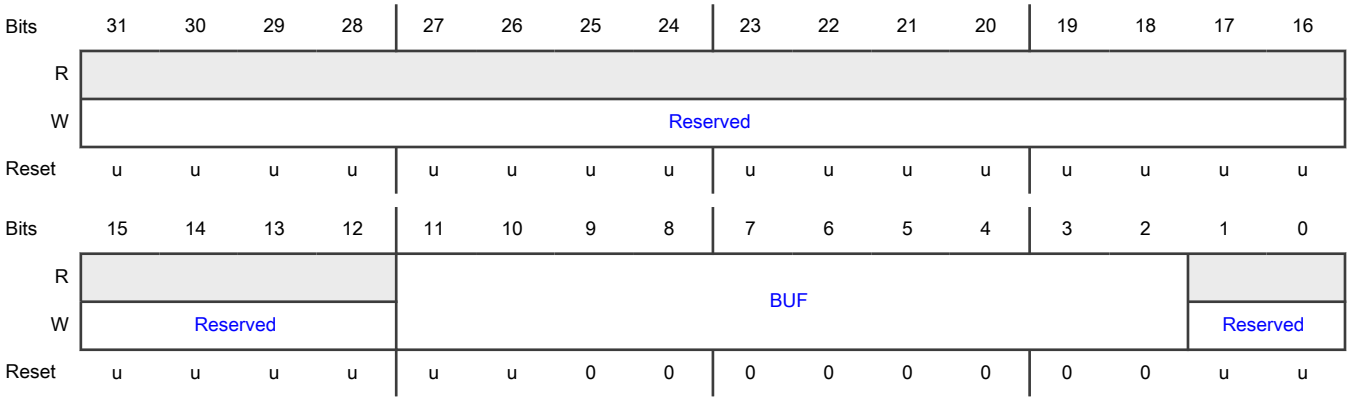
Field	Description
—	
11-0 SKIP	Endpoint skip Writing 1 to one of these bits, will indicate to Hardware that it must deactivate the buffer assigned to this endpoint and return control back to software. When Hardware has deactivated the endpoint, it will clear this bit, but it will not modify the EPINUSE bit. An interrupt will be generated when the Active bit goes from 1 to 0. Note: In case of double-buffering, Hardware will only clear the Active bit of the buffer indicated by the EPINUSE bit.

40.5.1.1.7 USB Endpoint Buffer in use (EPINUSE)

Offset

Register	Offset
EPINUSE	18h

Diagram



Fields

Field	Description
31-12 —	Reserved
11-2 BUF	Buffer in use One bit per physical endpoint. 00_0000_0000 - Hardware is accessing buffer 0 00_0000_0001 - Hardware is accessing buffer 1

Table continues on the next page...

Table continued from the previous page...

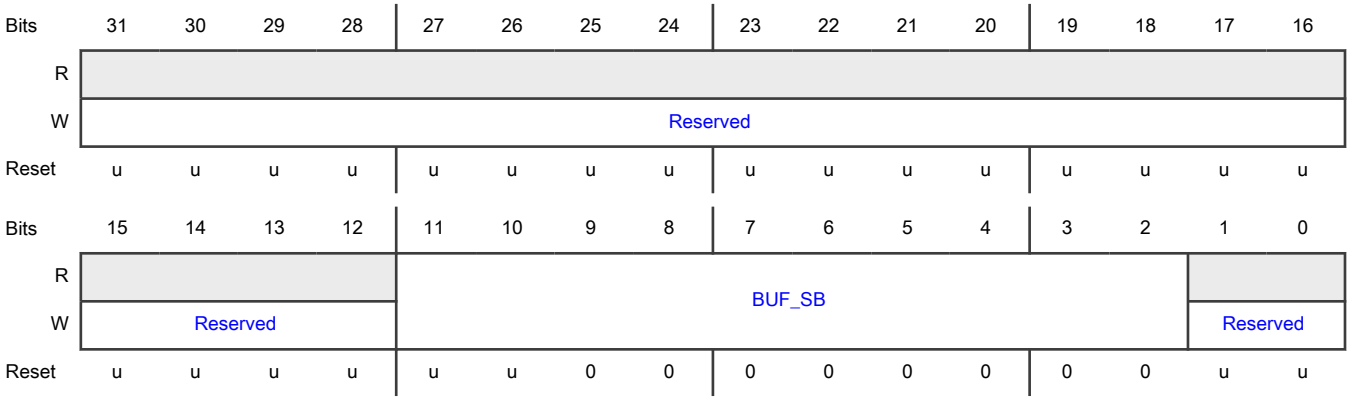
Field	Description
1-0 —	Reserved Fixed to zero because the control endpoint zero is fixed to single-buffering for each physical endpoint.

40.5.1.1.8 USB Endpoint Buffer Configuration (EPBUFCFG)

Offset

Register	Offset
EPBUFCFG	1Ch

Diagram



Fields

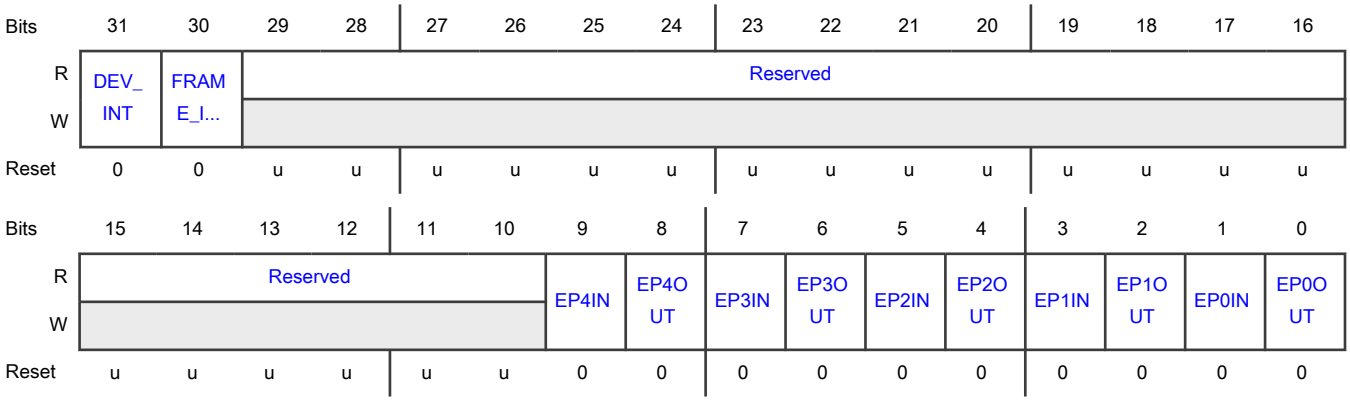
Field	Description
31-12 —	Reserved
11-2 BUF_SB	Buffer usage There is one bit per physical endpoint. If the bit is set to single-buffer (0), it will not toggle the corresponding EPINUSE bit when it clears the active bit. If the bit is set to double-buffer (1), Hardware will toggle the EPINUSE bit when it clears the Active bit for the buffer. 00_0000_0000 - Single-buffer 00_0000_0001 - Double-buffer
1-0 —	Reserved Fixed to zero because the control endpoint zero is fixed to single-buffering for each physical endpoint.

40.5.1.1.9 USB interrupt status (INTSTAT)

Offset

Register	Offset
INTSTAT	20h

Diagram



Fields

Field	Description
31 DEV_INT	Device status interrupt This bit is set by Hardware when one of the bits in the Device Status Change register are set. Software can clear this bit by writing a one to it.
30 FRAME_INT	Frame interrupt This bit is set to one every millisecond when the VbusDebounced bit and the DCON bit are set. This bit can be used by software when handling isochronous endpoints. Software can clear this bit by writing a one to it.
29-10 —	Reserved
9 EP4IN	EP4 IN direction This bit will be set if the corresponding Active bit is cleared by Hardware. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP4 IN direction. Software can clear this bit by writing a one to it.
8 EP4OUT	EP4 OUT direction This bit will be set if the corresponding Active bit is cleared by Hardware. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP4 OUT direction. Software can clear this bit by writing a one to it.
7	EP3 IN direction

Table continues on the next page...

Table continued from the previous page...

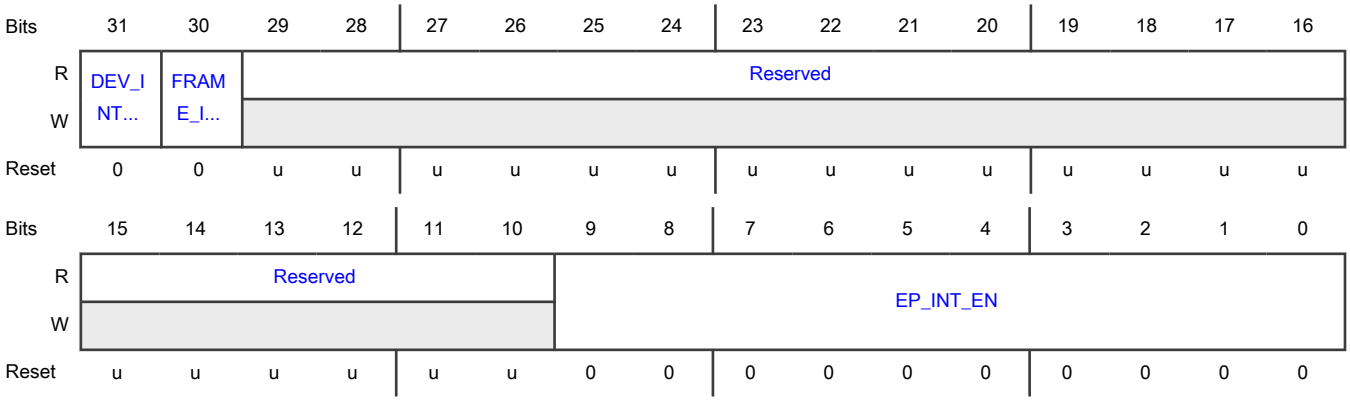
Field	Description
EP3IN	This bit will be set if the corresponding Active bit is cleared by Hardware. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP3 IN direction. Software can clear this bit by writing a one to it.
6 EP3OUT	EP3 OUT direction This bit will be set if the corresponding Active bit is cleared by Hardware. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP3 OUT direction. Software can clear this bit by writing a one to it.
5 EP2IN	EP2 IN direction This bit will be set if the corresponding Active bit is cleared by Hardware. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP2 IN direction. Software can clear this bit by writing a one to it.
4 EP2OUT	EP2 OUT direction This bit will be set if the corresponding Active bit is cleared by Hardware. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP2 OUT direction. Software can clear this bit by writing a one to it.
3 EP1IN	EP1 IN direction This bit will be set if the corresponding Active bit is cleared by Hardware. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the IntOnNAK_AI is set, this bit will also be set when a NAK is transmitted for the EP1 IN direction. Software can clear this bit by writing a one to it.
2 EP1OUT	EP1 OUT direction This bit will be set if the corresponding Active bit is cleared by Hardware. This is done in case the programmed NBytes transitions to zero or the skip bit is set by software. If the IntOnNAK_AO is set, this bit will also be set when a NAK is transmitted for the EP1 OUT direction. Software can clear this bit by writing a one to it.
1 EP0IN	Control EP0 IN direction This bit will be set if NBytes transitions to zero or the skip bit is set by software. If the IntOnNAK_CI is set, this bit will also be set when a NAK is transmitted for the Control EP0 IN direction. Software can clear this bit by writing a one to it.
0 EP0OUT	Control EP0 OUT direction This bit will be set if NBytes transitions to zero or the skip bit is set by software or a SETUP packet is successfully received for the control EP0. If the IntOnNAK_CO is set, this bit will also be set when a NAK is transmitted for the Control EP0 OUT direction. Software can clear this bit by writing a one to it.

40.5.1.1.10 USB interrupt enable (INTEN)

Offset

Register	Offset
INTEN	24h

Diagram



Fields

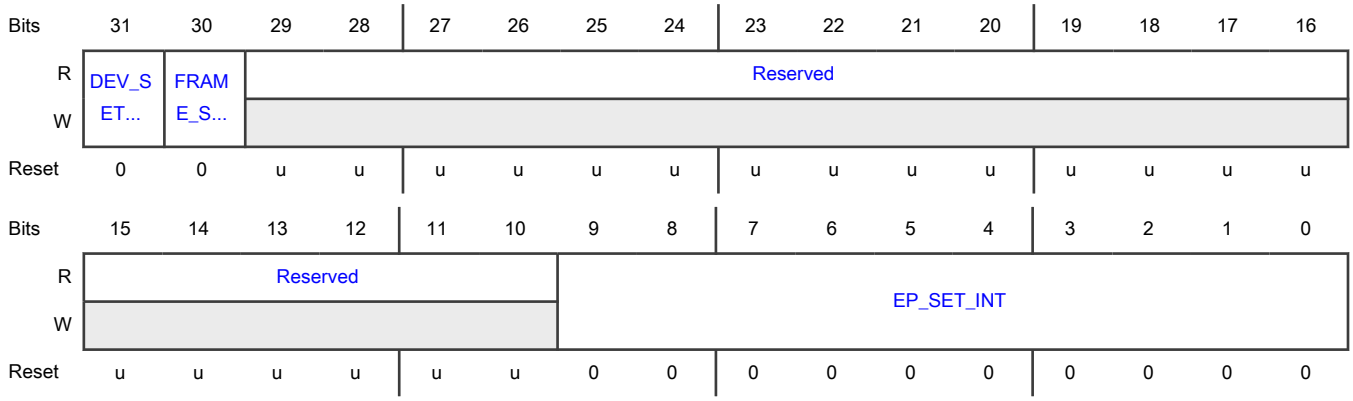
Field	Description
31 DEV_INT_EN	Device interrupt enable If this bit is set and the corresponding USB interrupt status bit is set, a Hardware interrupt is generated on the interrupt line indicated by the corresponding USB interrupt routing bit.
30 FRAME_INT_EN	Frame interrupt enable If this bit is set and the corresponding USB interrupt status bit is set, a Hardware interrupt is generated on the interrupt line indicated by the corresponding USB interrupt routing bit.
29-10 —	Reserved
9-0 EP_INT_EN	End point interrupt enable If this bit is set and the corresponding USB interrupt status bit is set, a Hardware interrupt is generated on the interrupt line indicated by the corresponding USB interrupt routing bit.

40.5.1.1.11 USB set interrupt status (INTSETSTAT)

Offset

Register	Offset
INTSETSTAT	28h

Diagram



Fields

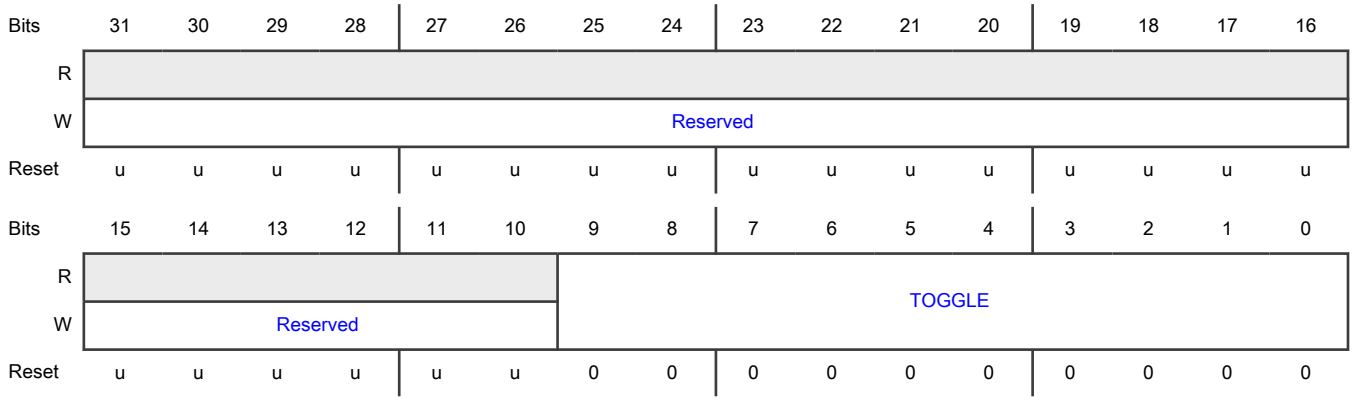
Field	Description
31 DEV_SET_INT	Device set interrupt If software writes a one to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.
30 FRAME_SET_INT	Frame set interrupt If software writes a one to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.
29-10 —	Reserved
9-0 EP_SET_INT	End point set interrupt If software writes a one to one of these bits, the corresponding USB interrupt status bit is set. When this register is read, the same value as the USB interrupt status register is returned.

40.5.1.1.12 USB Endpoint toggle (EPTOGGLE)

Offset

Register	Offset
EPTOGGLE	34h

Diagram



Fields

Field	Description
31-10 —	Reserved
9-0 TOGGLE	Endpoint data toggle This field indicates the current value of the data toggle for the corresponding endpoint.

40.5.2 USB 2.0 Full-speed Host controller register descriptions

The following registers are located in the AHB clock domain. They can be accessed directly by the processor. All registers are 32 bits wide and aligned with word address boundaries. See the OHCI specification for further details on the OHCI registers.

40.5.2.1 USBFSH memory map

USBFSH base address: 400A_2000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Host controller revision (HCREVISION)	32	RO	See section
4	Host controller control (HCCONTROL)	32	See section	See section
8	Host controller command status (HCCOMMANDSTATUS)	32	See section	See section
C	Host controller interrupt status (HCINTERRUPTSTATUS)	32	See section	See section
10	Host Controller interrupt enable (HCINTERRUPTENABLE)	32	See section	See section

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
14	The bits in this register are used to disable corresponding bits in the HCInterruptStatus register and in turn disable that event leading to hardware interrupt. (HCINTERRUPTDISABLE)	32	See section	See section
18	Host controller communication area (HCHCCA)	32	See section	See section
1C	Host controller period current endpoint descriptor (HCPERIODCURRENTED)	32	See section	See section
20	Host controller control head endpoint descriptor (HCCONTROLHEADED)	32	See section	See section
24	Host controller control current endpoint descriptor (HCCONTROLCURRENTED)	32	See section	See section
28	Host controller bulk head endpoint descriptor (HCBULKHEADED)	32	See section	See section
2C	Host controller bulk current endpoint descriptor (HCBULKCURRENTED)	32	See section	See section
30	Host controller done head (HCDONEHEAD)	32	See section	See section
34	Host controller frame interval (HCFMINTERVAL)	32	See section	See section
38	Host controller frame remaining (HCFMREMAINING)	32	See section	See section
3C	Host controller frame number (HCFMNUMBER)	32	See section	See section
40	Host controller periodic start (HCPERIODICSTART)	32	See section	See section
44	Host controller low speed threshold (HCLSTHRESHOLD)	32	See section	See section
48	Host controller root hub descriptor A (HCRHDESCRIPTORA)	32	See section	See section
4C	Host controller root hub descriptor B (HCRHDESCRIPTORB)	32	RW	See section
50	(HCRHSTATUS)	32	See section	See section
54	Host controller root hub port status (HCRHPORTSTATUS)	32	See section	See section
5C	Port Mode (PORTMODE)	32	See section	See section

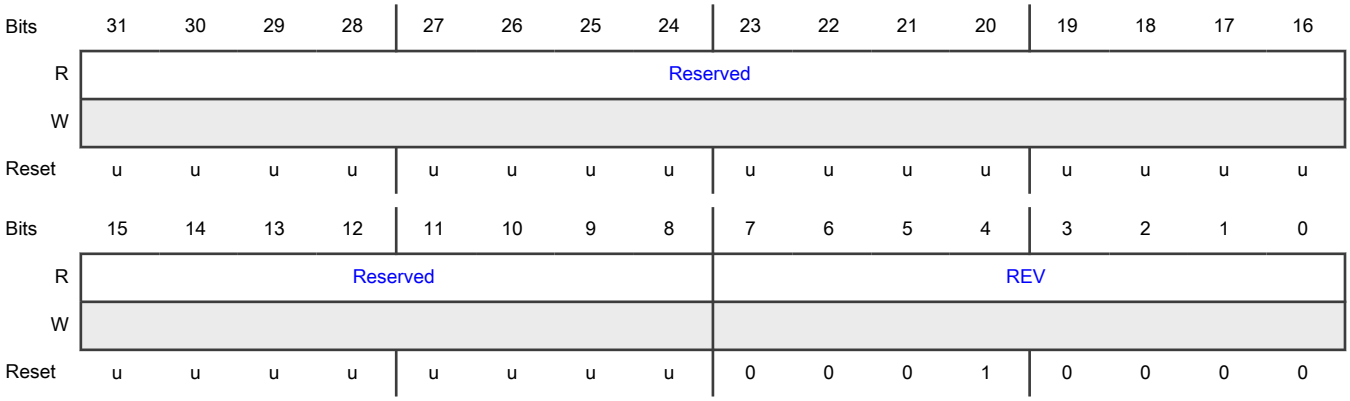
40.5.2.1.1 Host controller revision (HCREVISION)

BCD representation of the version of the HCI specification that is implemented by the Host Controller (HC).

Offset

Register	Offset
HCREVISION	0h

Diagram



Fields

Field	Description
31-8 —	Reserved Read value is undefined, only zero should be written.
7-0 REV	Revision This read-only field contains the BCD representation of the version of the HCI specification that is implemented by this HC.

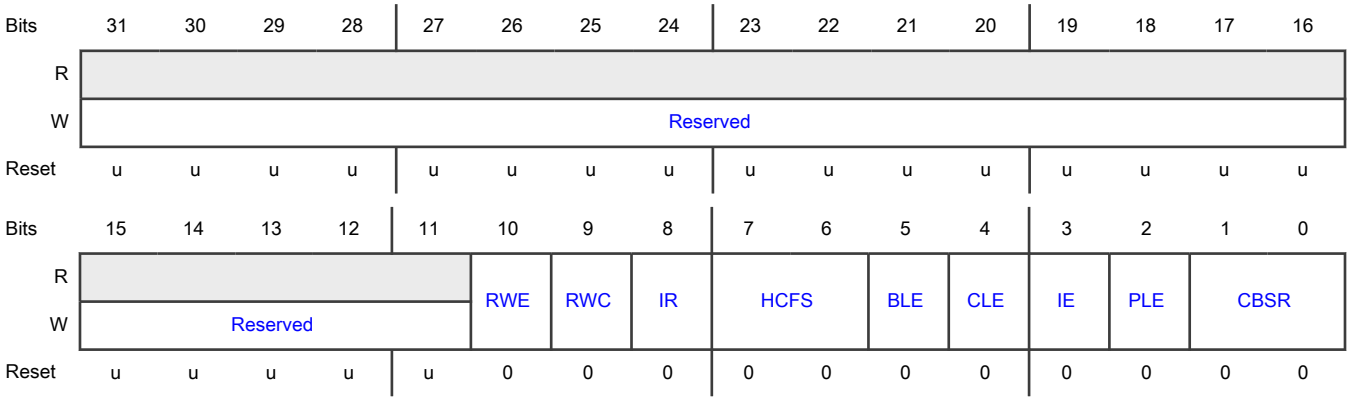
40.5.2.1.2 Host controller control (HCCONTROL)

This register defines the operating modes of the HC.

Offset

Register	Offset
HCCONTROL	4h

Diagram



Fields

Field	Description
31-11 —	Reserved Read value is undefined, only zero should be written.
10 RWE	RemoteWakeupEnable This bit is used by HCD to enable or disable the remote wake-up feature upon the detection of upstream resume signaling.
9 RWC	RemoteWakeupConnected This bit indicates whether HC supports remote wake-up signaling.
8 IR	InterruptRouting This bit determines the routing of interrupts generated by events registered in HcInterruptStatus.
7-6 HCFS	HostControllerFunctionalState A transition to USBOPERATIONAL from another state causes SOFgeneration to begin 1 ms later. HCD may determine whether HC has begun sending SOFs by reading the StartofFrame field of HcInterruptStatus. This field may be changed by HC only when in the USBSUSPEND state. HC may move from the USBSUSPEND state to the USBRESUME state after detecting the resume signaling from a downstream port. HC enters USBSUSPEND after a software reset, whereas it enters USBRESET after a hardware reset. The latter also resets the root hub and asserts subsequent reset signaling to downstream ports. 00 - USBRESET 01 - USBRESUME 10 - USBOPERATIONAL 11 - USBSUSPEND
5 BLE	BulkListEnable This bit is set to enable the processing of the Bulk list in the next Frame. This bit is set to enable the processing of the bulk list in the next frame. If cleared by HCD, processing of the bulk list does not occur after the next SOF. HC checks this bit whenever it determines to process the

Table continues on the next page...

Table continued from the previous page...

Field	Description
	list. When disabled, HCD may modify the list. If HcBulkCurrentED is pointing to an ED to be removed, HCD must advance the pointer by updating HcBulkCurrentED before re-enabling processing of the list.
4 CLE	ControlListEnable. This bit is set to enable the processing of the control list in the next frame. If cleared by HCD, processing of the control list does not occur after the next SOF. HC must check this bit whenever it determines to process the list. When disabled, HCD may modify the list. If HcControlCurrentED is pointing to an ED to be removed, HCD must advance the pointer by updating HcControlCurrentED before re-enabling processing of the list.
3 IE	IsochronousEnable. This bit is used by HCD to enable/disable processing of isochronous EDs. While processing the periodic list in a frame, HC checks the status of this bit when it finds an Isochronous ED (F=1). If set (enabled), HC continues processing the EDs. If cleared (disabled), HC halts processing of the periodic list (that contains only isochronous EDs) and begins processing the bulk/control lists. Setting this bit is guaranteed to take effect in the next frame (not the current frame).
2 PLE	PeriodicListEnable. This bit is set to enable the processing of the periodic list in the next frame. If cleared by HCD, processing of the periodic list does not occur after the next SOF. HC must check this bit before it starts processing the list.
1-0 CBSR	ControlBulkServiceRatio. This specifies the service ratio between control and bulk EDs. Before processing any of the nonperiodic lists, HC must compare the ratio specified with its internal count on how many nonempty control EDs have been processed, in determining whether to continue serving another control ED or switching to bulk EDs. The internal count will be retained when crossing the frame boundary. In case of reset, HCD is responsible for restoring this value.

CBSR Value	Number of control EDs Over bulk EDs Served
0	1:1
1	2:1
2	3:1
3	4:1

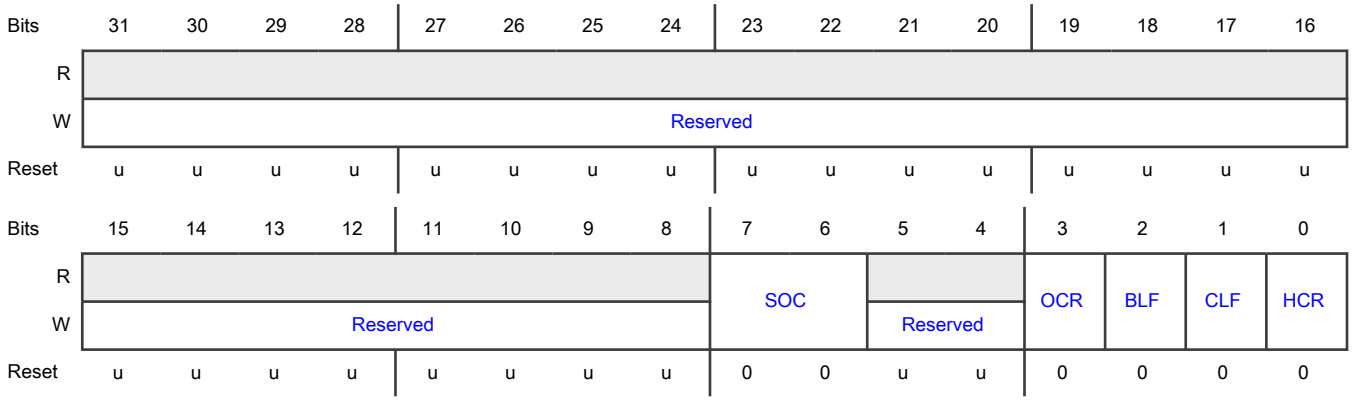
40.5.2.1.3 Host controller command status (HCCOMMANDSTATUS)

This register is used to receive the commands from the Host Controller Driver (HCD) and also indicates status of the host controller.

Offset

Register	Offset
HCCOMMANDSTATUS	8h

Diagram



Fields

Field	Description
31-8 —	Reserved Read value is undefined, only zero should be written.
7-6 SOC	SchedulingOverrunCount These bits are incremented on each scheduling overrun error. These bits are incremented on each scheduling overrun error. It is initialized to 00b and wraps around at 11b. This will be incremented when a scheduling overrun is detected even if SchedulingOverrun in HcInterruptStatus has already been set. It is used by HCD to monitor any persistent scheduling problems.
5-4 —	Reserved Read value is undefined, only zero should be written.
3 OCR	OwnershipChangeRequest This bit is set by an OS HCD to request a change of control of the HC. This bit is set by an OS HCD to request a change of control of the HC. When set, HC will set the OwnershipChange field in HcInterruptStatus. After the change over, this bit is cleared and remains so until the next request from OS HCD.
2 BLF	BulkListFilled This bit is used to indicate whether there are any TDs on the Bulk list. This bit is used to indicate whether there are any TDs on the bulk list. It is set by HCD whenever it adds a TD to an ED in the bulk list. When HC begins to process the head of the bulk list, it checks BF. As long as BulkListFilled is 0, HC will not start processing the bulk list. If BulkListFilled is 1, HC will start processing the bulk list and will set BF to 0. If HC finds a TD on the list, then HC will set BulkListFilled to 1 causing the bulk list processing to continue. If no TD is found on the bulk list, and if HCD does not set BulkListFilled, then BulkListFilled will still be 0 when HC completes processing the bulk list and bulk list processing will stop.
1 CLF	ControlListFilled This bit is used to indicate whether there are any TDs on the Control list. This bit is used to indicate whether there are any TDs on the control list. It is set by HCD whenever it adds a TD to an ED in the control list. When HC begins to process the head of the control list, it checks CLF. As long as ControlListFilled is 0, HC will not start processing the control list. If CF is 1, HC will start processing the control list and will set ControlListFilled to 0. If HC finds a TD on the list, then HC will set ControlListFilled to 1 causing the control list processing to continue. If no TD is found on the control list,

Table continues on the next page...

Table continued from the previous page...

Field	Description
	and if the HCD does not set ControlListFilled, then ControlListFilled will still be 0 when HC completes processing the control list and control list processing will stop.
0 HCR	HostControllerReset This bit is set by HCD to initiate a software reset of HC. Regardless of the functional state of HC, it moves to the USBSUSPEND state in which most of the operational registers are reset except those stated otherwise; For example, the InterruptRouting field of HcControl and no host bus accesses are allowed. This bit is cleared by HC when the reset operation is completed. The reset operation must be completed within 10 ☐s. This bit, when set, should not cause a reset to the Root Hub and no subsequent reset signaling should be asserted to its downstream ports.

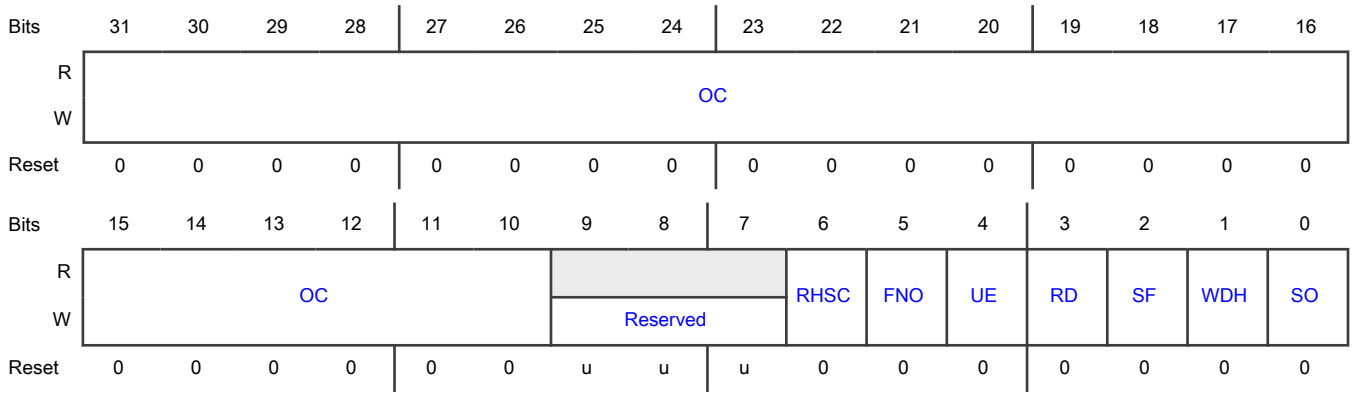
40.5.2.1.4 Host controller interrupt status (HCINTERRUPTSTATUS)

The HC interrupt status register provides status on various events that cause hardware interrupts. When an event occurs, host controller sets the corresponding bit in this register. When a bit is set, a hardware interrupt is generated if the interrupt is enabled in the HcInterruptEnable register and the MasterInterruptEnable bit is set. The host controller driver may clear specific bits in this register by writing 1 to bit positions to be cleared. The host controller driver may not set any of these bits. The host controller will never clear the bit.

Offset

Register	Offset
HCINTERRUPTSTATUS	Ch

Diagram



Fields

Field	Description
31-10 OC	OwnershipChange

Table continues on the next page...

Table continued from the previous page...

Field	Description
	This bit is set by HC when HCD sets the OwnershipChangeRequest field in HcCommandStatus. This event, when unmasked, will always generate an System Management Interrupt (SMI) immediately. This bit is tied to 0b when the SMI pin is not implemented.
9-7 —	Reserved Read value is undefined, only zero should be written.
6 RHSC	RootHubStatusChange This bit is set when the content of HcRhStatus or the content of any of HcRhPortStatus[NumberofDownstreamPort] has changed. This bit is set when the content of HcRhStatus or the content of any of HcRhPortStatus[NumberofDownstreamPort] has changed.
5 FNO	FrameNumberOverflow This bit is set when the MSb of HcFmNumber (bit 15) changes value, from 0 to 1 or from 1 to 0, and after HccaFrameNumber has been updated.
4 UE	UnrecoverableError This bit is set when HC detects a system error not related to USB. HC should not proceed with any processing nor signaling before the system error has been corrected. HCD clears this bit after HC has been reset.
3 RD	ResumeDetected This bit is set when HC detects that a device on the USB is asserting resume signaling. It is the transition from no resume signaling to resume signaling causing this bit to be set. This bit is not set when HCD sets the USBRESUME state.
2 SF	StartofFrame This bit is set by HC at each start of a frame and after the update of HccaFrameNumber. HC also generates a SOF token at the same time.
1 WDH	WritebackDoneHead This bit is set immediately after HC has written HcDoneHead to HccaDoneHead. Further updates of the HccaDoneHead will not occur until this bit has been cleared. HCD should only clear this bit after it has saved the content of HccaDoneHead.
0 SO	SchedulingOverrun This bit is set when the USB schedule for the current frame overruns and after the update of HccaFrameNumber. A scheduling overrun will also cause the SchedulingOverrunCount of HcCommandStatus to be incremented.

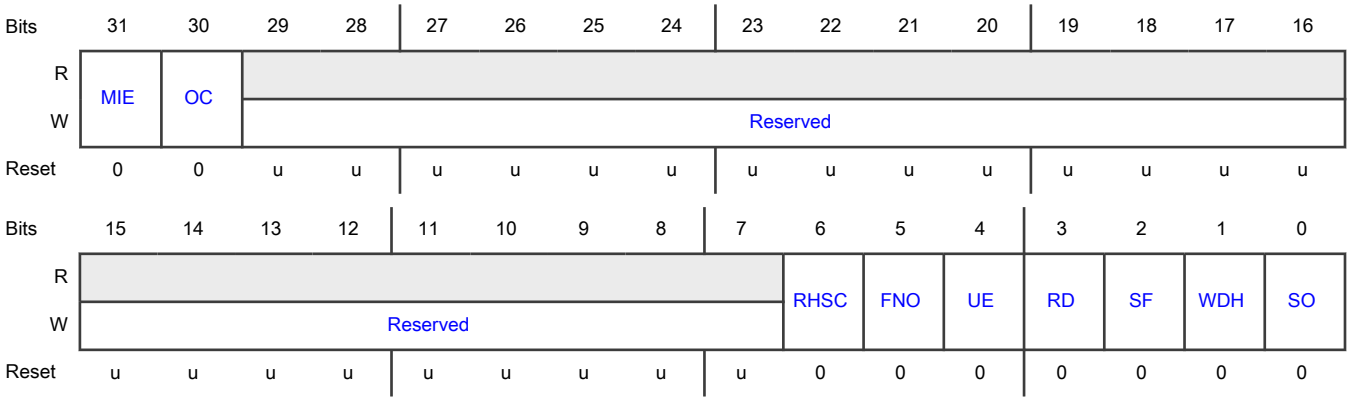
40.5.2.1.5 Host Controller interrupt enable (HCINTERRUPTENABLE)

In the HcInterruptEnable register, each enable bit corresponds to an associated interrupt bit in the HcInterruptStatus register. The HcInterruptEnable register is used to control which events generate a hardware interrupt. When a bit is set in the HcInterruptStatus register AND the corresponding bit in the HcInterruptEnable register is set AND the MasterInterruptEnable bit is set, then a hardware interrupt is requested on the host bus. Writing a '1' to a bit in this register sets the corresponding bit, whereas writing a '0' to a bit in this register leaves the corresponding bit unchanged. On read, the current value of this register is returned.

Offset

Register	Offset
HCINTERRUPTENABLE	10h

Diagram



Fields

Field	Description
31 MIE	Master Interrupt Enable. It is used by HCD as a master interrupt enable. A 0 written to this field is ignored by HC. A 1 written to this field enables interrupt generation because of events specified in the other bits of this register.
30 OC	Ownership Change interrupt. 0 - No effect 1 - Enables interrupt
29-7 —	Reserved Read value is undefined, only zero should be written.
6 RHSC	Root Hub Status Change interrupt. 0 - No effect 1 - Enables interrupt
5 FNO	Frame Number Overflow interrupt. 0 - No effect 1 - Enables interrupt
4 UE	Unrecoverable Error interrupt. 0 - No effect 1 - Enables interrupt
3	Resume Detect interrupt.

Table continues on the next page...

Table continued from the previous page...

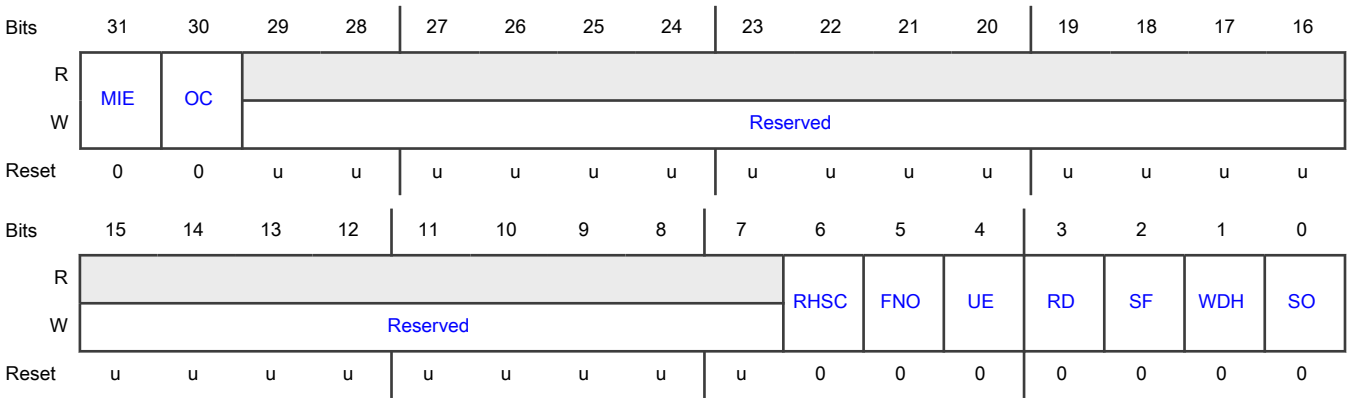
Field	Description
RD	0 - No effect 1 - Enables interrupt
2 SF	Start of Frame interrupt. 0 - No effect 1 - Enables interrupt
1 WDH	HcDoneHead Writeback interrupt. 0 - No effect 1 - Enables interrupt
0 SO	Scheduling Overrun interrupt. 0 - No effect 1 - Enables interrupt

40.5.2.1.6 The bits in this register are used to disable corresponding bits in the HCInterruptStatus register and in turn disable that event leading to hardware interrupt. (HCINTERRUPTDISABLE)

Offset

Register	Offset
HCINTERRUPTDISABLE	14h

Diagram



Fields

Field	Description
31 MIE	A 0 written to this field is ignored by HC. 0 - No effect 1 - Disables interrupt
30 OC	Ownership Change interrupt. 0 - No effect 1 - Disables interrupt
29-7 —	Reserved Read value is undefined, only zero should be written.
6 RHSC	Root Hub Status Change interrupt. 0 - No effect 1 - Disables interrupt
5 FNO	Frame Number Overflow interrupt. 0 - No effect 1 - Disables interrupt
4 UE	Unrecoverable Error interrupt. 0 - No effect 1 - Disables interrupt
3 RD	Resume Detect interrupt. 0 - No effect 1 - Disables interrupt
2 SF	Start of Frame interrupt. 0 - No effect 1 - Disables interrupt
1 WDH	HcDoneHead Writeback interrupt. 0 - No effect 1 - Disables interrupt
0 SO	Scheduling Overrun interrupt. 0 - No effect 1 - Disables interrupt

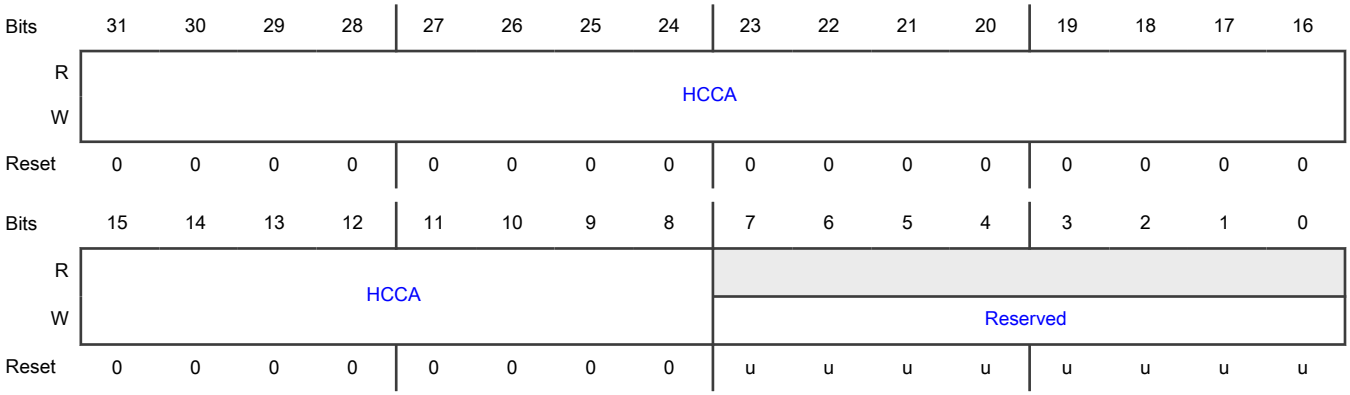
40.5.2.1.7 Host controller communication area (HCHCCA)

This register contains the physical address of the host controller communication area.

Offset

Register	Offset
HCHCCA	18h

Diagram



Fields

Field	Description
31-8 HCCA	Host Controller Communication Area Base address of Host controller communication area
7-0 —	Reserved Read value is undefined, only zero should be written.

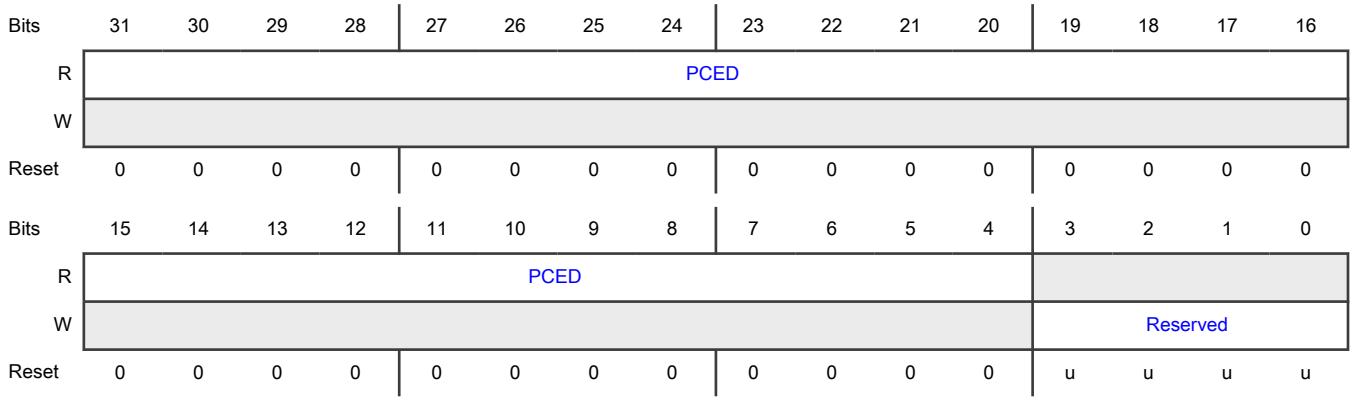
40.5.2.1.8 Host controller period current endpoint descriptor (HCPERIODCURRENTED)

The host controller period current ED register is used by the host controller to point to the head of one of the Periodic lists, which will be processed in the current frame. It contains the physical address of the current isochronous or interrupt endpoint descriptor.

Offset

Register	Offset
HCPERIODCURRENTED	1Ch

Diagram



Fields

Field	Description
31-4 PCED	The content of this register is updated by HC after a periodic ED is processed. HCD may read the content in determining which ED is currently being processed at the time of reading.
3-0 —	Reserved Read value is undefined, only zero should be written.

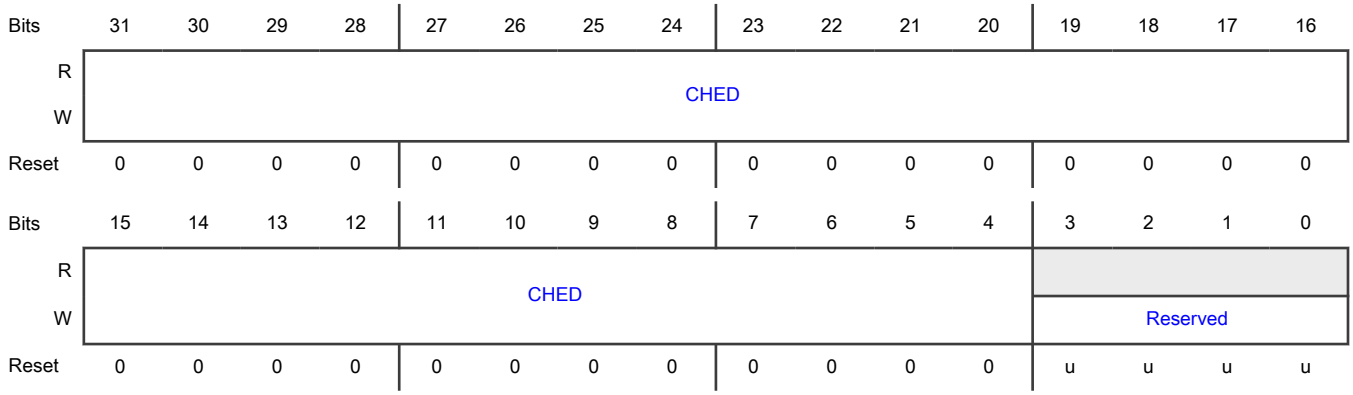
40.5.2.1.9 Host controller control head endpoint descriptor (HCCONTROLHEADED)

This register contains the physical address of the first endpoint descriptor of the control list.

Offset

Register	Offset
HCCONTROLHEADED	20h

Diagram



Fields

Field	Description
31-4 CHED	HC traverses the Control list starting with the HcControlHeadED pointer. The content is loaded from HCCA during the initialization of HC.
3-0 —	Reserved Read value is undefined, only zero should be written.

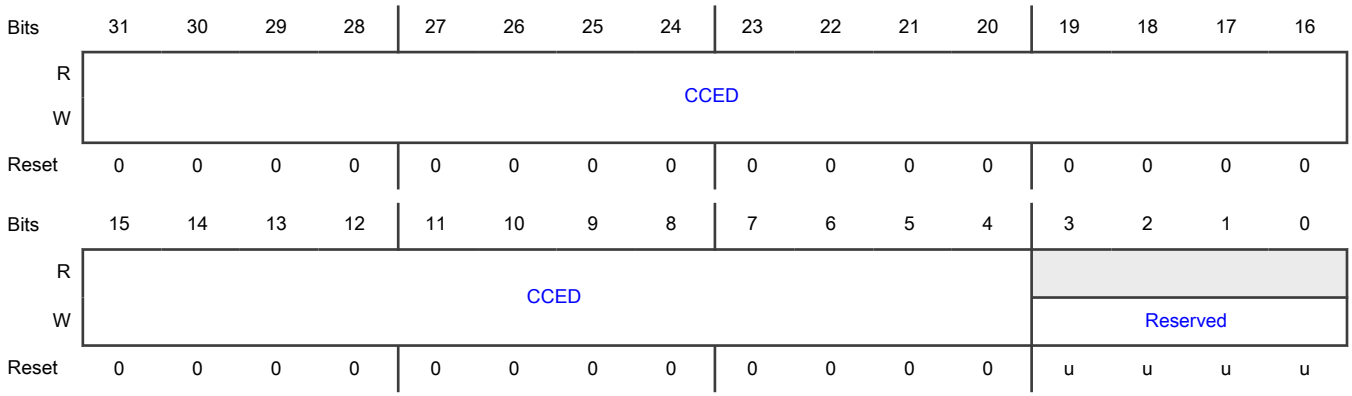
40.5.2.1.10 Host controller control current endpoint descriptor (HCCONTROLCURRENTED)

This register contains the physical address of the current endpoint descriptor of the control list.

Offset

Register	Offset
HCCONTROLCURRENTED	24h

Diagram



Fields

Field	Description
31-4 CCED	ControlCurrentED This pointer is advanced to the next ED after serving the current one. HC continues to process the list from where it left off in the last frame. When it reaches the end of the control list, HC checks the ControlListFilled (CLF) bit in the HcCommandStatus. If set, it copies the contents of HcControlHeadED to HcControlCurrentED and clears the bit. If not set, it does nothing. HCD is allowed to modify this register only when the ControlListEnable of HcControl is cleared. When set, HCD only reads the instantaneous value of this register. Initially, it is set to 0 to indicate the end of the control list.
3-0 —	Reserved Read value is undefined, only zero should be written.

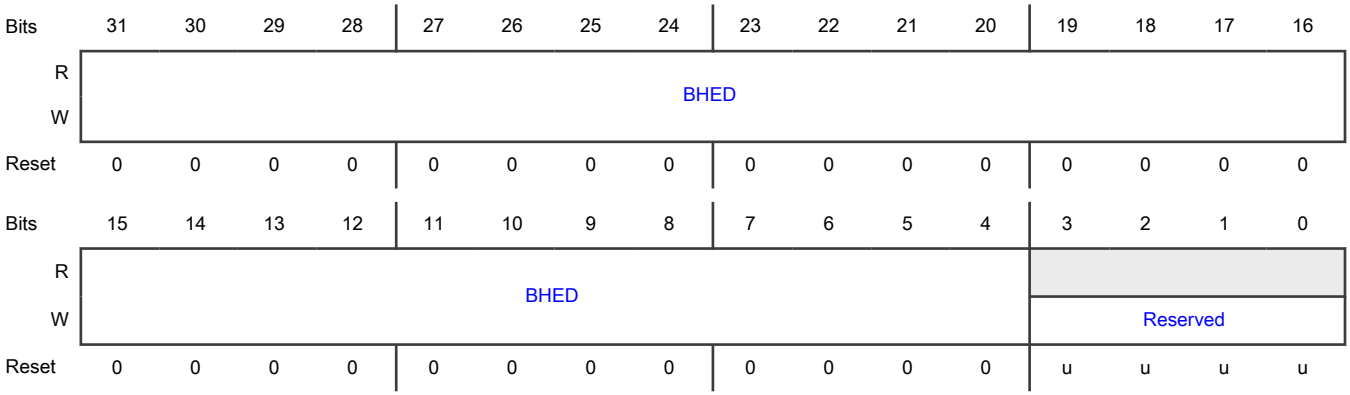
40.5.2.1.11 Host controller bulk head endpoint descriptor (HCBULKHEADED)

This register contains the physical address of the first endpoint descriptor of the bulk list.

Offset

Register	Offset
HCBULKHEADED	28h

Diagram



Fields

Field	Description
31-4 BHED	BulkHeadED HC traverses the bulk list starting with the HcBulkHeadED pointer. The content is loaded from HCCA during the initialization of HC.
3-0 —	Reserved Read value is undefined, only zero should be written.

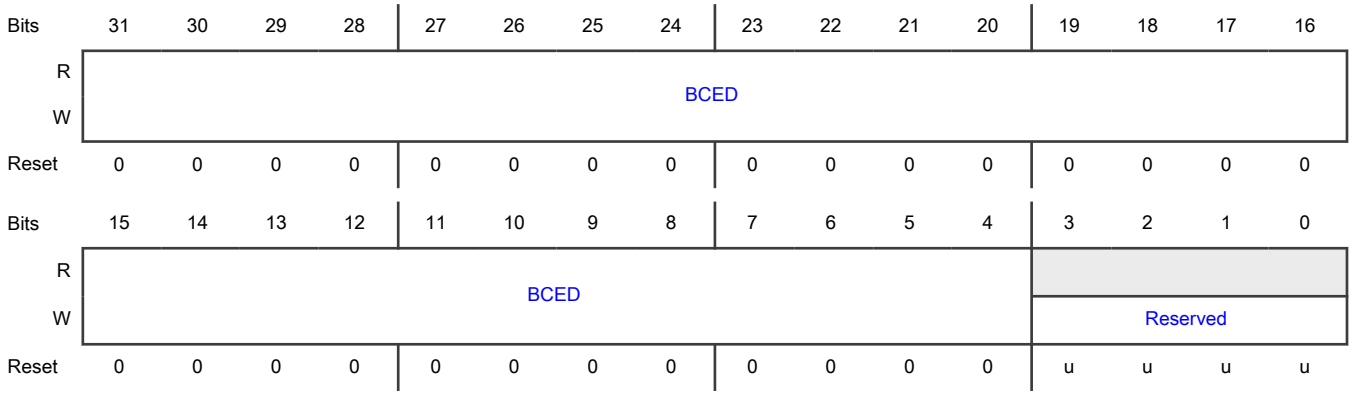
40.5.2.1.12 Host controller bulk current endpoint descriptor (HCBULKCURRENTED)

This register contains the physical address of the current endpoint descriptor of the bulk list.

Offset

Register	Offset
HCBULKCURRENTED	2Ch

Diagram



Fields

Field	Description
31-4 BCED	BulkCurrentED It is advanced to the next ED after the HC has served the current one. HC continues to process the list from where it left off in the last frame. When it reaches the end of the bulk list, HC checks the ControlListFilled of HcControl. If set, it copies the content of HcBulkHeadED to HcBulkCurrentED and clears the bit. If it is not set, it does nothing. HCD is only allowed to modify this register when the BulkListEnable of HcControl is cleared. When set, the HCD only reads the instantaneous value of this register. It is initially set to 0 to indicate the end of the bulk list.
3-0 —	Reserved Read value is undefined, only zero should be written.

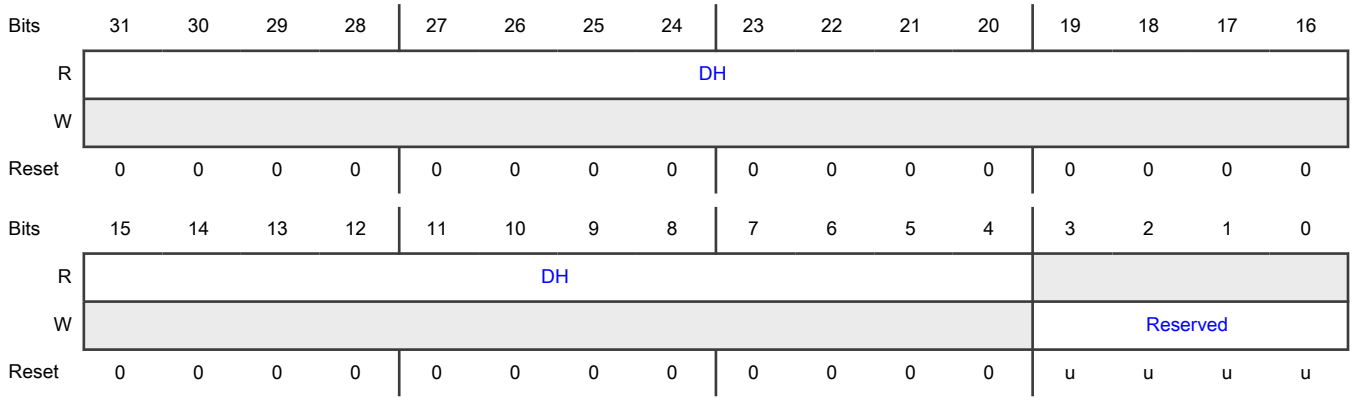
40.5.2.1.13 Host controller done head (HCDONEHEAD)

This register contains the physical address of the last transfer descriptor added to the 'Done' queue.

Offset

Register	Offset
HCDONEHEAD	30h

Diagram



Fields

Field	Description
31-4 DH	DoneHead When a TD is completed, HC writes the content of HcDoneHead to the NextTD field of the TD. HC then overwrites the content of HcDoneHead with the address of this TD. It is set to 0 whenever HC writes the content of this register to HCCA. It also sets the WritebackDoneHead of HcInterruptStatus.
3-0 —	Reserved Read value is undefined, only zero should be written.

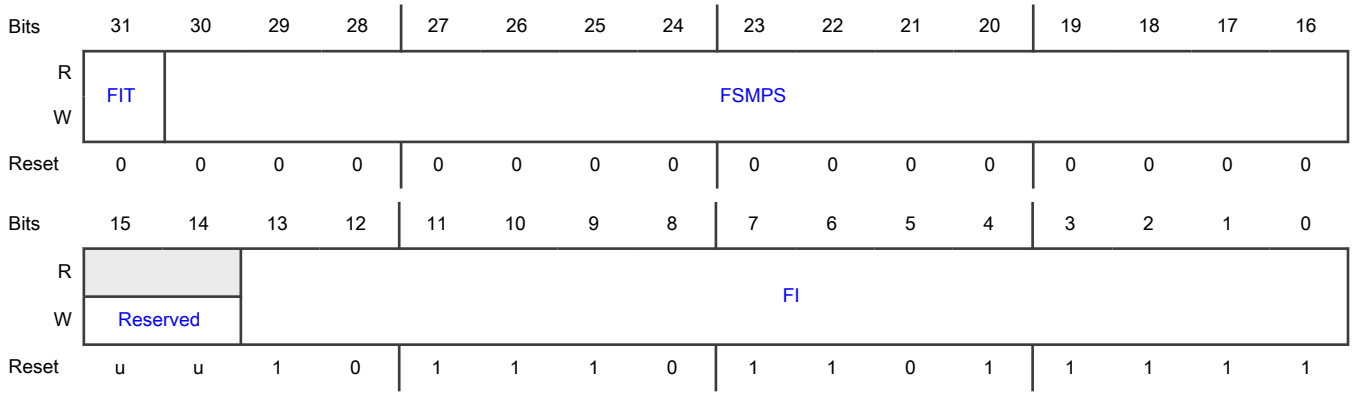
40.5.2.1.14 Host controller frame interval (HCFMINTERVAL)

This register defines the bit time interval in a frame and the full speed maximum packet size which would not cause an overrun.

Offset

Register	Offset
HCFMINTERVAL	34h

Diagram



Fields

Field	Description
31 FIT	FrameIntervalToggle HCD toggles this bit whenever it loads a new value to FrameInterval.
30-16 FSMPS	FSLargestDataPacket This field specifies a value which is loaded into the Largest Data Packet Counter at the beginning of each frame. The counter value represents the largest amount of data in bits which can be sent or received by the HC in a single transaction at any given time without causing scheduling overrun. The field value is calculated by the HCD.
15-14 —	Reserved Read value is undefined, only zero should be written.
13-0 FI	FrameInterval This specifies the interval between two consecutive SOFs in bit times. The nominal value is set to be 11,999. HCD should store the current value of this field before resetting HC. By setting the HostControllerReset field of HcCommandStatus, the HC resets this field to its nominal value. HCD may choose to restore the stored value on completion of the Reset sequence.

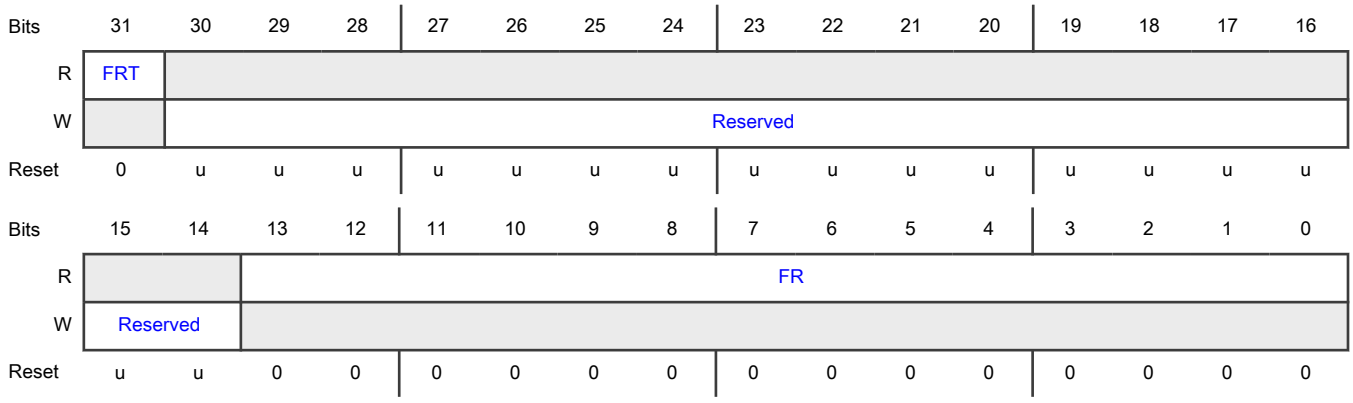
40.5.2.1.15 Host controller frame remaining (HCFMREMAINING)

The host controller frame remaining register is a 14-bit down counter showing the bit time remaining in the current frame.

Offset

Register	Offset
HCFMREMAINING	38h

Diagram



Fields

Field	Description
31 FRT	FrameRemainingToggle This bit is loaded from the FrameIntervalToggle field of HcFmInterval whenever FrameRemaining reaches 0. This bit is used by HCD for the synchronization between FrameInterval and FrameRemaining.
30-14 —	Reserved Read value is undefined, only zero should be written.
13-0 FR	FrameRemaining This counter is decremented at each bit time. When it reaches 0, it is reset by loading the FrameInterval value specified in HcFmInterval at the next bit time boundary. When entering the USBOPERATIONAL state, HC re-loads the content with the FrameInterval of HcFmInterval and uses the updated value from the next SOF.

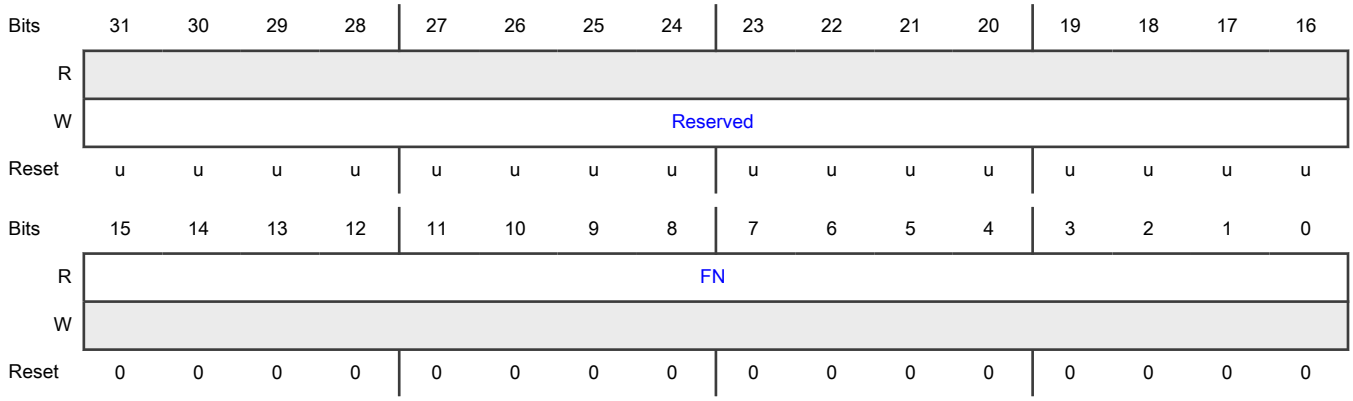
40.5.2.1.16 Host controller frame number (HCFMNUMBER)

The host controller frame number register is a 16-bit counter. It provides a timing reference among events happening in the host controller and the host controller driver. The host controller driver may use the 16-bit value specified in this register and generate a 32-bit frame number without requiring frequent access to the register.

Offset

Register	Offset
HCFMNUMBER	3Ch

Diagram



Fields

Field	Description
31-16	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
15-0 FN	FrameNumber It is incremented when HcFmRemaining is re-loaded. It will be rolled over to 0h after FFFH. When entering the USBOPERATIONAL state, it is incremented automatically. The content is written to HCCA after HC has incremented the FrameNumber at each frame boundary and sent a SOF but before HC reads the first ED in that frame. After writing to HCCA, HC sets the StartofFrame in HcInterruptStatus.

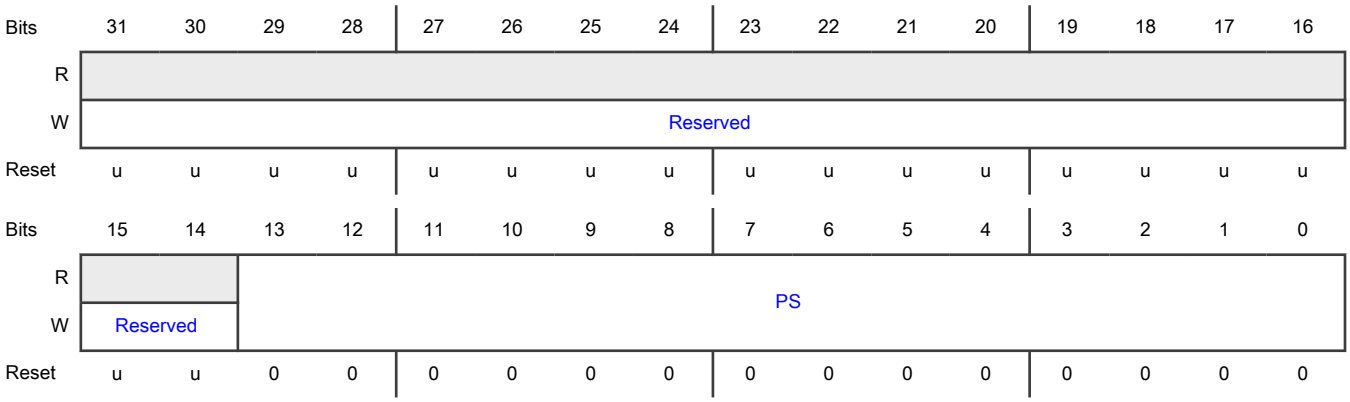
40.5.2.1.17 Host controller periodic start (HCPERIODICSTART)

The host controller periodic start register has a 14-bit programmable value that determines the earliest time when HC should start processing the periodic list.

Offset

Register	Offset
HCPERIODICSTART	40h

Diagram



Fields

Field	Description
31-14 —	Reserved Read value is undefined, only zero should be written.
13-0 PS	PeriodicStart After a hardware reset, this field is cleared and then set by HCD during the HC initialization. The value is calculated approximately as 10% off from HcFmInterval.. A typical value will be 3E67h. When HcFmRemaining reaches the value specified, processing of the periodic lists will have priority over control/bulk processing. HC will therefore start processing the interrupt list after completing the current control or bulk transaction that is in progress.

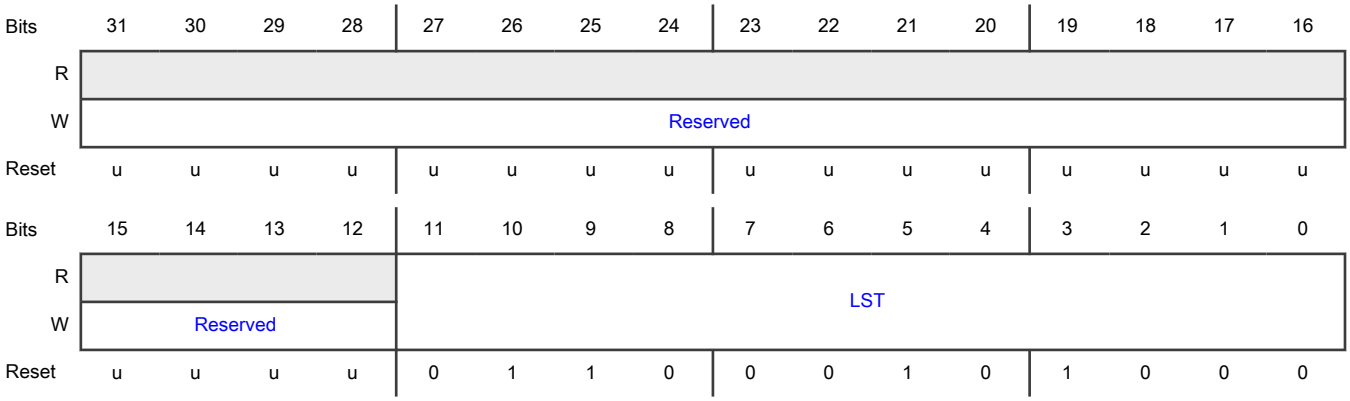
40.5.2.1.18 Host controller low speed threshold (HCLSTHRESHOLD)

This register contains an 11-bit value used by the host controller to determine whether to commit to the transfer of a maximum of 8-byte LS packet before EOF. The host controller and the host controller driver are not allowed to change this value.

Offset

Register	Offset
HCLSTHRESHOLD	44h

Diagram



Fields

Field	Description
31-12 —	Reserved Read value is undefined, only zero should be written.
11-0 LST	LSThreshold This field contains a value which is compared to the FrameRemaining field prior to initiating a Low Speed transaction. The transaction is started only if FrameRemaining ☐ this field. The value is calculated by HCD with the consideration of transmission and setup overhead.

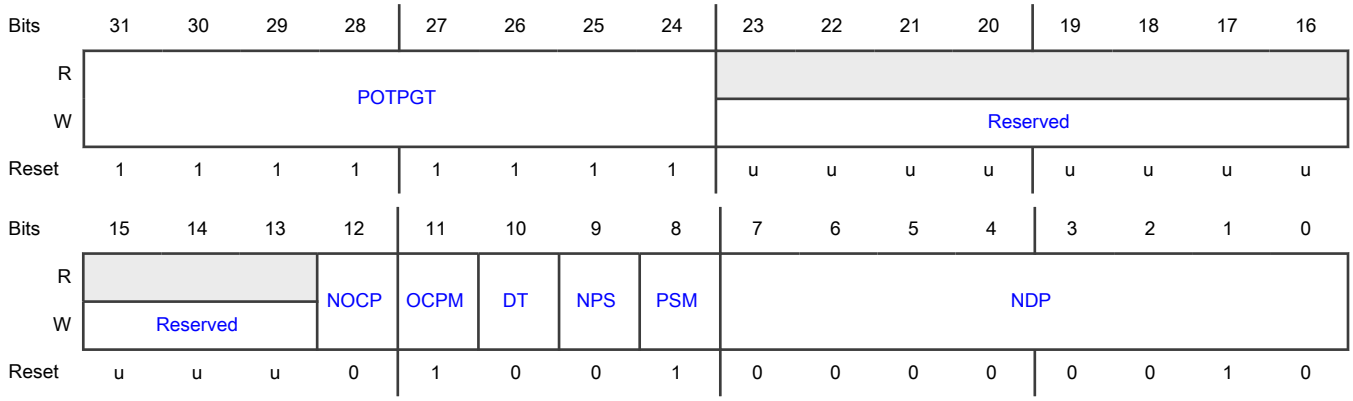
40.5.2.1.19 Host controller root hub descriptor A (HCRHDESCRIPTORA)

The host controller root hub descriptor A register is the first register describing the characteristics of the root hub. Reset values are implementation specific. The descriptor length (11), descriptor type, and hub controller current (0) fields of the hub Class Descriptor are emulated by the HCD. All other fields are located in the HcRhDescriptorA and HcRhDescriptorB registers.

Offset

Register	Offset
HCRHDESCRIPTORA	48h

Diagram



Fields

Field	Description
31-24 POTPGT	PowerOnToPowerGoodTime This byte specifies the duration the HCD has to wait before accessing a powered-on port of the root hub. The unit of time is 2 ms. The duration is calculated as POTPGT * 2 ms.
23-13 —	Reserved Read value is undefined, only zero should be written.
12 NOCP	NoOverCurrentProtection This bit describes how the overcurrent status for the root hub ports are reported. 0 - Over-current status is reported collectively for all downstream ports 1 - No overcurrent protection supported
11 OCPM	OverCurrentProtectionMode This bit describes how the overcurrent status for the root hub ports are reported. 0 - Over-current status is reported collectively for all downstream ports 1 - Over-current status is reported on a per-port basis
10 DT	DeviceType This bit specifies that the root hub is not a compound device. The root hub is not permitted to be a compound device. This field should always read/write 0.
9 NPS	NoPowerSwitching These bits are used to specify whether power switching is supported or port are always powered. When this bit is cleared, the PowerSwitchingMode specifies global or per-port switching. 0 - Ports are power switched 1 - Ports are always powered on when the host controller is powered on
8 PSM	PowerSwitchingMode This bit is used to specify how the power switching of the root hub ports is controlled. This field is only valid if the NoPowerSwitching field is cleared. This mode allows portpower to be controlled by either the

Table continues on the next page...

Table continued from the previous page...

Field	Description
	global switch or per-port switching. If the PortPowerControlMask bit is set, the port responds only to port power commands (Set/ClearPortPower). If the port mask is cleared, the port is controlled only by the global power switch (Set/ClearGlobalPower). 0 - All ports are powered at the same time 1 - Each port is powered individually.
7-0 NDP	NumberDownstreamPorts These bits specify the number of downstream ports supported by the root hub.

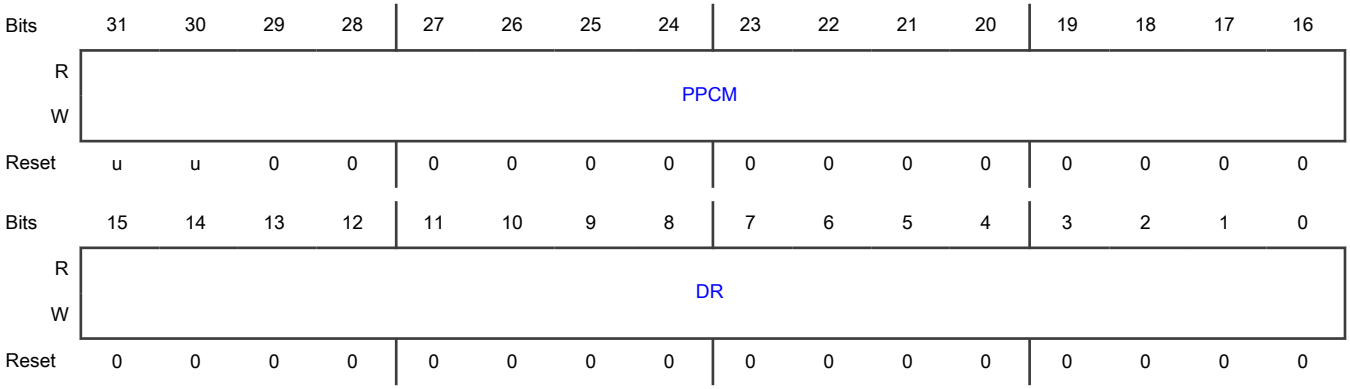
40.5.2.1.20 Host controller root hub descriptor B (HCRHDESCRIPTORB)

The host controller root hub descriptor B register is the second register describing the characteristics of the root hub. These fields are written during initialization to correspond with the system implementation. Reset values are implementation-specific.

Offset

Register	Offset
HCRHDESCRIPTORB	4Ch

Diagram



Fields

Field	Description
31-16 PPCM	PortPowerControlMask Each bit indicates if a port is affected by a global power control command when PowerSwitchingMode is set. When set, the port's power state is only affected by per-port power control (Set/ClearPortPower). When cleared, the port is controlled by the global power switch (Set/ClearGlobalPower). If the device is configured to global switching mode (PowerSwitchingMode = 0), this field is not valid. Bit 0: Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
	Bit 1: Ganged-power mask on port #1 Bit 2: Ganged-power mask on port #2 ..... Bit 15: Ganged-power mask on port #15
15-0 DR	DeviceRemovable Each bit is dedicated to a port of the Root Hub. When cleared, the attached device is removable. When set, the attached device is not removable. Bit 0: Reserved Bit 1: Device attached to port #1 Bit 2: Device attached to port #2 ..... Bit 15: Device attached to port #15

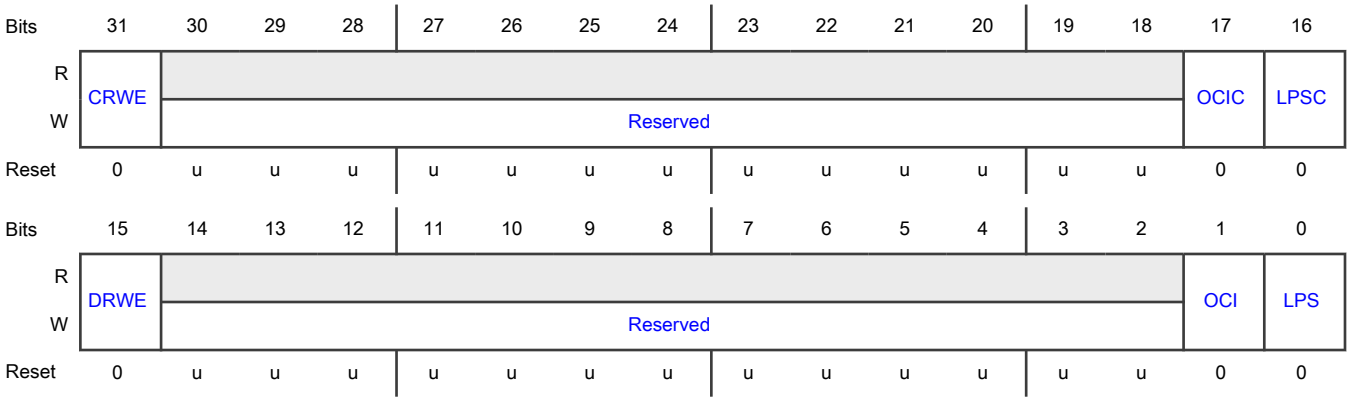
40.5.2.1.21 (HCRHSTATUS)

This register is divided into two parts. The lower word of a Dword represents the hub status field and the upper word represents the hub status change field. Reserved bits should always be written 0.

Offset

Register	Offset
HCRHSTATUS	50h

Diagram



Fields

Field	Description
31 CRWE	(write) ClearRemoteWakeupEnable . Writing a 1 clears DeviceRemoveWakeupEnable
30-18 —	Reserved Read value is undefined, only zero should be written.
17 OCIC	OverCurrentIndicatorChange This bit is set by hardware when a change has occurred to the OCI field of this register. The HCD clears this bit by writing a 1. Writing a 0 has no effect.
16 LPSC	(read) LocalPowerStatusChange(Write) SetGlobalPower . The root hub does not support the local power status feature. In global power mode (PowerSwitchingMode=0), this bit is written to 1 to turn on power to all ports (clear PortPowerStatus). In per-port power mode, it sets PortPowerStatus only on ports whose PortPowerControlMask bit is not set. Writing a 0 has no effect.
15 DRWE	(read) DeviceRemoteWakeupEnable (Write) SetRemoteWakeupEnable This bit enables a ConnectStatusChange bit as a resume event, causing a USBSUSPEND to USBRESUME state transition and setting the ResumeDetected interrupt. 0 - ConnectStatusChange is not a remote wakeup event 1 - ConnectStatusChange is a remote wakeup event
14-2 —	Reserved Read value is undefined, only zero should be written.
1 OCI	OverCurrentIndicator This bit reports overcurrent conditions when the global reporting is implemented. When set, an overcurrent condition exists. When cleared, all power operations are normal. If per-port overcurrent protection is implementedthis bit is always 0.
0 LPS	(read) LocalPowerStatus (write) ClearGlobalPower The Root Hub does not support the local power status feature; thus, this bit is always read as 0. In global power mode (PowerSwitchingMode=0), this bit is written to 1 to turn off power to all ports (clear PortPowerStatus). In per-port power mode, it clears PortPowerStatus only on ports whose PortPowerControlMask bit is not set. Writing a 0 has no effect.

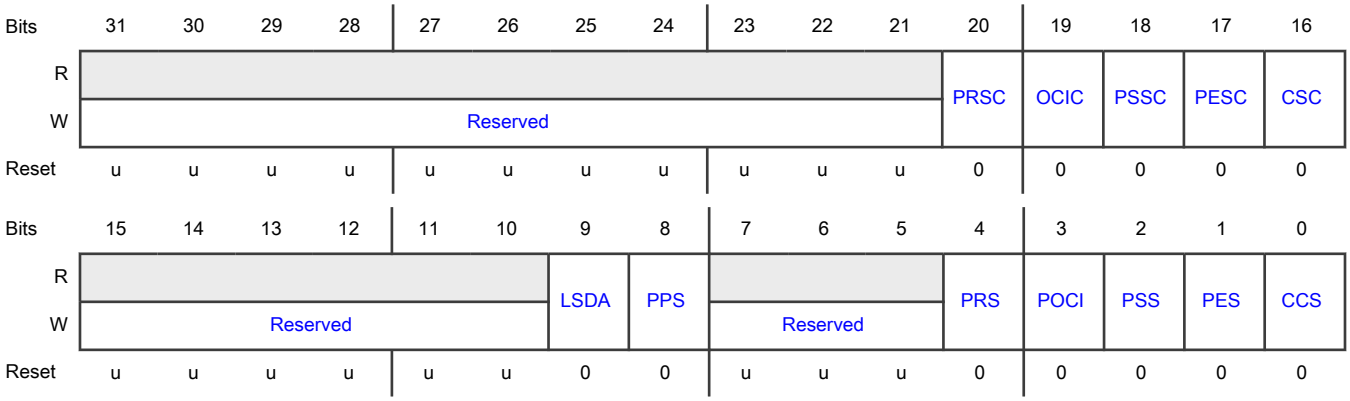
40.5.2.1.22 Host controller root hub port status (HCRHPORTSTATUS)

This register controls and reports the port events on a per-port basis. NumberDownstreamPorts represents the number of HcRhPortStatus registers that are implemented in the hardware. The lower word is used to reflect the port status and the upper word reflects the status change bits. Some status bits are implemented with special write behavior. If a transaction (token through handshake) is in progress when a write to change port status occurs, the resulting port status change must be postponed until the transaction completes. Reserved bits should always be written 0.

Offset

Register	Offset
HCRHPORTSTATUS	54h

Diagram



Fields

Field	Description
31-21 —	Reserved Read value is undefined, only zero should be written.
20 PRSC	PortResetStatusChange This bit is set at the end of the 10 ms port reset signal. The HCD writes a 1 to clear this bit. Writing a 0 has no effect. 0 - Port reset is not complete 1 - Port reset is complete
19 OCIC	PortOverCurrentIndicatorChange This bit is valid only if overcurrent conditions are reported on a per-port basis. This bit is set when root hub changes the PortOverCurrentIndicator bit. The HCD writes a 1 to clear this bit. Writing a 0 has no effect. 0 - PortOverCurrentIndicator has not changed 1 - PortOverCurrentIndicator has changed
18 PSSC	PortSuspendStatusChange This bit is set when the full resume sequence is completed. This sequence includes the 20 ms K-state resume pulse. LS EOP, and 3 ms resychronization delay. The HCD writes a 1 to clear this bit. Writing a 0 has no effect. This bit is also cleared when ResetStatusChange is set. 0 - Resume sequence is not complete 1 - Resume sequence is complete

Table continues on the next page...

Table continued from the previous page...

Field	Description
17 PESC	PortEnableStatusChange This bit is set when hardware events cause the PortEnableStatus bit to be cleared. Changes from HCD writes do not set this bit. The HCD writes a 1 to clear this bit. Writing a 0 has no effect. 0 - PortEnableStatus has not changed 1 - PortEnableStatus has changed
16 CSC	ConnectStatusChange This bit is set whenever a connect or disconnect event occurs. The HCD writes a 1 to clear this bit. Writing a 0 has no effect. If CurrentConnectStatus is cleared when a SetPortReset, SetPortEnable, or SetPortSuspend write occurs, this bit is set to force the driver to re-evaluate the connection status because these writes should not occur if the port is disconnected. 0 - CurrentConnectStatus has not changed 1 - CurrentConnectStatus has changed
15-10 —	Reserved Read value is undefined, only zero should be written.
9 LSDA	(read) LowSpeedDeviceAttached (write) ClearPortPower This bit indicates the speed of the device attached to this port. When set, a low-speed device is attached to this port. When clear, a full-speed device is attached to this port. This field is valid only when the CurrentConnectStatus is set. The HCD clears the PortPowerStatus bit by writing a 1 to this bit. Writing a 0 has no effect. 0 - Full speed device is attached 1 - Low speed device is attached
8 PPS	(read) PortPowerStatus This bit reflects the port's power status, regardless of the type of power switching implemented. This bit is cleared if an overcurrent condition is detected. HCD sets this bit by writing SetPortPower or SetGlobalPower. HCD clears this bit by writing ClearPortPower or ClearGlobalPower. The PowerSwitchingMode and PortPortControlMask[NDP] determine which power control switches are enabled. In global switching mode (PowerSwitchingMode=0), only Set/ClearGlobalPower controls this bit. In per-port power switching (PowerSwitchingMode=1), if the PortPowerControlMask[NDP] bit for the port is set, only Set/ClearPortPower commands are enabled. If the mask is not set, only Set/ClearGlobalPower commands are enabled. When port power is disabled, CurrentConnectStatus, PortEnableStatus, PortSuspendStatus, and PortResetStatus should be reset. The HCD writes a 1 to set the PortPowerStatus bit. Writing a 0 has no effect. 0 - Port power is off 1 - Port power is on
7-5	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
4 PRS	(read) PortResetStatus When this bit is set by a write to SetPortReset, port reset signaling is asserted. (Write) SetPortReset When this bit is set by a write to SetPortReset, port reset signaling is asserted. When reset is completed, this bit is cleared when PortResetStatusChange is set. This bit cannot be set if CurrentConnectStatus is cleared. The HCD sets the port reset signaling by writing a 1 to this bit. Writing a 0 has no effect. If CurrentConnectStatus is cleared, this write does not set PortResetStatus, instead, sets ConnectStatusChange. This informs the driver that it attempted to reset a disconnected port. 0 - Port reset signal is not active 1 - Port reset signal is active
3 POCI	(read) PortOverCurrentIndicator (write) ClearSuspendStatus This bit is only valid when the Root Hub is configured in such a way that overcurrent conditions are reported on a per-port basis. If per-port overcurrent reporting is not supported, this bit is set to 0. If cleared, all power operations are normal for this port. If set, an overcurrent condition exists on this port. This bit always reflects the overcurrent input signal. The HCD writes a 1 to initiate a resume. Writing a 0 has no effect. A resume is initiated only if PortSuspendStatus is set. 0 - Overcurrent condition is not detected 1 - Overcurrent condition is detected
2 PSS	(Read) PortSuspendStatus (Write) SetPortSuspend This bit indicates the port is suspended or in the resume sequence. It is set by a SetSuspendState write and cleared when PortSuspendStatusChange is set at the end of the resume interval. This bit cannot be set if CurrentConnectStatus is cleared. This bit is also cleared when PortResetStatusChange is set at the end of the port reset or when the HC is placed in the USBRESUME state. If an upstream resume is in progress, it should propagate to the HC. The HCD sets the PortSuspendStatus bit by writing a 1 to this bit. Writing a 0 has no effect. If CurrentConnectStatus is cleared, this write does not set PortSuspendStatus, instead, it sets ConnectStatusChange. This informs the driver that it attempted to suspend a disconnected port. 0 - Port is not suspended 1 - Port is suspended
1 PES	(Read)PortEnableStatus (Write) ClearPortEnable This bit indicates whether the port is enabled or disabled. The root hub may clear this bit when an overcurrent condition, disconnect event, switched-off power, or operational bus error, such as, babble is detected. The change also causes PortEnabledStatusChange to be set. HCD sets this bit by writing

Table continues on the next page...

Table continued from the previous page...

Field	Description
	SetPortEnable and clears it by writing ClearPortEnable.This bit cannot be set when CurrentConnectStatus is cleared. This bit is also set, if not already, at the completion of a port reset when ResetStatusChange is set or port suspend when SuspendStatusChange is set. The HCD sets PortEnableStatus by writing a 1. Writing a 0 has no effect. If CurrentConnectStatus is cleared, this write does not set PortEnableStatus, but instead sets ConnectStatusChange. This informs the driver that it attempted to enable a disconnected port. 0 - Port is disabled 1 - Port is enabled
0 CCS	CurrentConnectStatus (write) ClearPortEnable This bit reflects the current state of the downstream port. The HCD writes a 1 to this bit to clear the PortEnableStatus bit. Writing a 0 has no effect. The CurrentConnectStatus is not affected by any write. **NOTE** This bit is always read 1b when the attached device is nonremovable. 0 - Device is not connected 1 - Device is connected

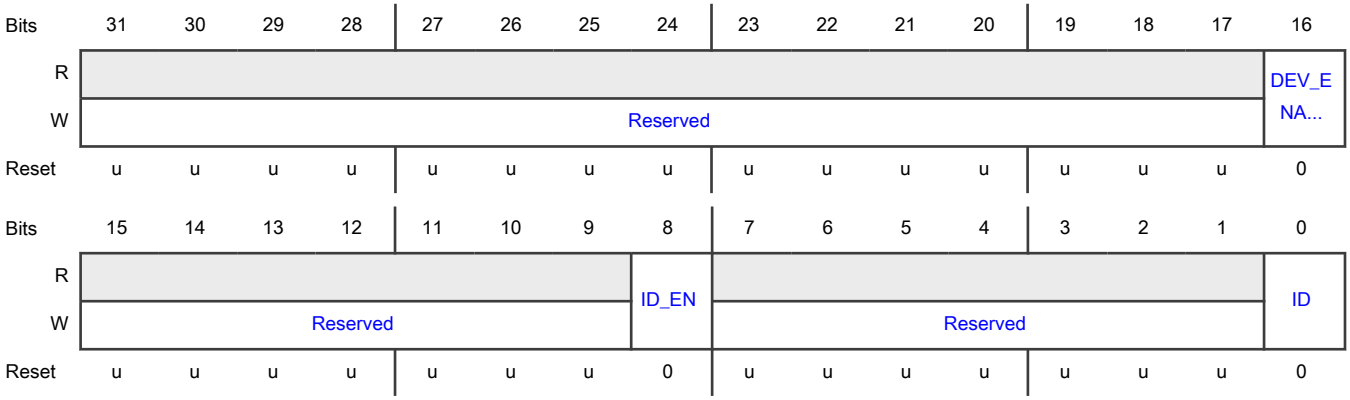
40.5.2.1.23 Port Mode (PORTMODE)

The port mode register controls the host or device role in addition to setting the polarity of the ID pin.

Offset

Register	Offset
PORTMODE	5Ch

Diagram



Fields

Field	Description
31-17 —	Reserved Read value is undefined, only zero should be written.
16 DEV_ENABLE	Device Enable 0 - Device 1 - Host
15-9 —	Reserved Read value is undefined, only zero should be written.
8 ID_EN	Port ID pin pull-up enable. This bit indicates the current value of USB0_IDVALUE.
7-1 —	Reserved Read value is undefined, only zero should be written.
0 ID	Port ID pin value. This bit indicates the current value of USB0_IDVALUE.

Chapter 41
Flexcomm Serial Communication

41.1 Chip-specific Flexcomm information

Table 338. Reference links to related information

Topic	Related module	Reference
Full description	Flexcomm	Flexcomm
System memory map		System memory map
Clocking		Clock generation
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

41.1.1 Module instances

This device has 9 Flexcomm modules. Each Flexcomm module has a separate base address.

- Flexcomm module 0 - 7 can be configured by the user for one of these protocols: USART, SPI, I^2C, or I^2S.
- Flexcomm module 7, configured as I2S, can stream DMIC channel pair 0 and 1. See I2S configured for DMIC output.
- Flexcomm module 8 can be configured for HS SPI.

Specific part numbers and package variations may include a subset of this list.

Table 339. Flexcomm module base addresses and functions

Flexcomm number	Base address	USART	SPI	I^2C	I^2S
0	0x4008_6000	Yes	Yes	Yes	Yes
1	0x4008_7000	Yes	Yes	Yes	Yes
2	0x4008_8000	Yes	Yes	Yes	Yes
3	0x4008_9000	Yes	Yes	Yes	Yes
4	0x4008_A000	Yes	Yes	Yes	Yes
5	0x4009_6000	Yes	Yes	Yes	Yes
6	0x4009_7000	Yes	Yes	Yes	Yes (with 4 channel pairs)
7	0x4009_8000	Yes	Yes	Yes	Yes (with 4 channel pairs)
8 (HS SPI)	0x4009_F000	No	Yes	No	No

41.1.2 I2S configured for DMIC output

The DMIC channel pair 0 and 1 can be streamed to the Flexcomm 7 module configured as I2S, accessible by the CPU and both DMA controllers.

Use I2S register CFG1[PDMDATA] and the DMIC registers FIFO_CTRLn[ENABLE].

41.2 Flexcomm Overview

41.2.1 Introduction

This chapter describes the overall Flexcomm module and how to choose and configure each function. Multiple Flexcomm modules are available. Each Flexcomm module provides one peripheral function from a choice of several. Details of the different peripheral functions are found in the separate subsections: USART, SPI, I2C, and I2S.

41.2.1.1 Block diagram

The Flexcomm module block diagram shows the overall structure of one Flexcomm module.

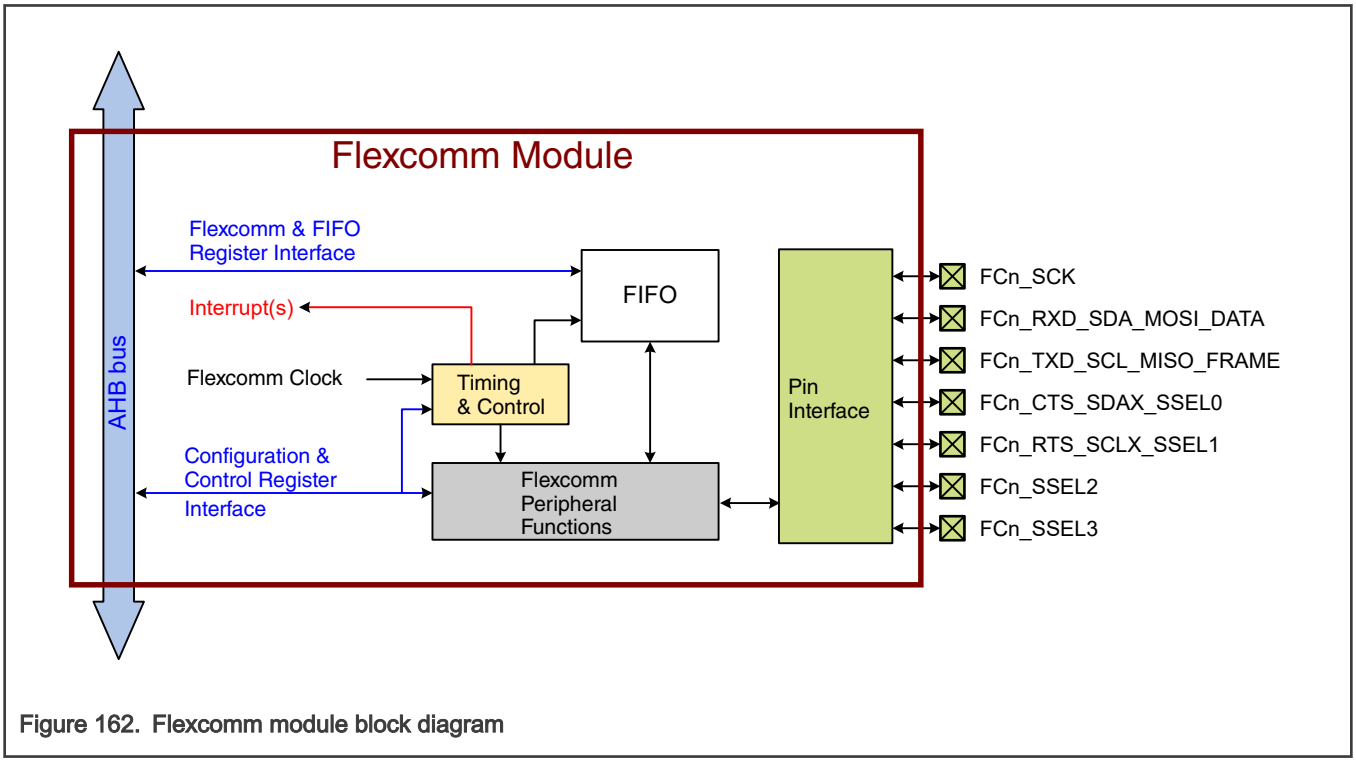


Figure 162. Flexcomm module block diagram

41.2.1.2 Features

The Flexcomm modules provide a choice of peripheral functions:

- USART with asynchronous operation or synchronous master or slave operation.
- SPI master or slave, with up to 4 slave selects.
- I2C, including separate master, slave, and monitor functions.
- Some Flexcomm modules may also provide an I2S function. If so, there may be from one to four I2S channel pairs, one of which may optionally be a master and the rest slaves, configured together for either transmit or receive.
- Data for USART, SPI, and I2S traffic uses the Flexcomm FIFO. The I2C function does not use the FIFO.

41.2.2 Functional description

The following sections describe functional details of the Flexcomm module.

41.2.2.1 Choosing a peripheral function

Use software to select a peripheral function, from among those supported by a particular Flexcomm module.

- Choose the peripheral function by writing to the PSELID register.
- Read the PSELID register to see which peripheral functions are available on that Flexcomm module.

After software selects a specific peripheral function, the PID register supplies an identifier for the selected peripheral. Software may use this information to confirm the selection before proceeding.

41.2.2.1.1 FIFO usage

See the specific peripheral function for information on how the FIFO is used.

41.2.2.1.2 DMA

The Flexcomm module generates DMA requests if wanted, based on a selectable FIFO level. To enable the DMA requests, set the bit for the related Flexcomm module instance in the INPUTMUX DMA controller request enable register. For some Flexcomm module instances, a DMA device selection must also be made in the DMA controller channel mux select register.

41.2.3 Signals

Each Flexcomm module allows up to 7 pin connections. Specific uses of a Flexcomm Interface typically do not use all 7 pin connections, and some Flexcomm instances may not provide a means to connect all functions to device pins. Pin usage for a specific peripheral function is described in the subsection for that peripheral.

Table 340. Flexcomm pins

Pin	Type	Description
SCK	I/O	Clock input or output for the USART function in synchronous modes.
	I/O	Clock input or output for the SPI function.
	I/O	Clock input or output for the I^2S function (if present).
RXD_SDA_MOSI or RXD_SDA_MOSI_DATA	Input	Receive data input for the USART function.
	I/O	SDA (data) input/output for the I^2C function.
	I/O	Master data output/slave data input for the SPI function.
	I/O	Data input or output for the I^2S function (if present).
TXD_SCL_MISO or TXD_SCL_MISO_WS	Output	Transmit data output for the USART function.
	I/O	SCL input/output for the I^2C function.
	I/O	Master data input/slave data output for the SPI function.
	I/O	WS (also known as LRCLK) input or output for the I^2S function (if present).
CTS_SDA_SSEL0	Input	Clear To Send input for the USART function.

Table continues on the next page...

Table 340. Flexcomm pins (continued)

Pin	Type	Description
	I/O	SDA (data) input/output for the I^2C function.
	I/O	Slave Select 0 input or output for the SPI function.
RTS_SCL_SSEL1	Output	Request To Send output for the USART function.
	I/O	SCL (clock) input/output for the I^2C function.
	I/O	Slave Select 1 input or output for the SPI function.
SSEL2	I/O	Slave Select 2 input or output for the SPI function.
SSEL3	I/O	Slave Select 3 input or output for the SPI function.

41.2.4 Initialization

To Initialize the Flexcomm module:

1. Enable the peripheral clock for the Flexcomm module.
2. Select a clock source for the related Flexcomm module.
3. Select the desired Flexcomm function by writing to the PSELID register of the chosen Flexcomm module.
4. See specific peripheral subsections for information on configuration: USART, SPI, I^2C, I^2S.

41.2.5 Memory map and register definition

This section includes the Flexcomm module memory map and detailed descriptions of Flexcomm ID registers.

Each Flexcomm module contains registers that are related to the specific Flexcomm peripheral function, including registers related to peripheral FIFOs and data access. These registers depend on the chosen peripheral functions (USART, SPI, I^2C, and I^2S , if present in a specific Flexcomm module). They are described in the subsections for each Flexcomm peripheral. Each Flexcomm module has a separate base address.

NOTE

Generally, the bus interface to the Flexcomm registers (including the serial peripheral functions) support only word writes. Do not use byte and halfword writes. An exception is the FIFOWR register when used with SPI. When the Flexcomm module is configured for SPI, Flexcomm allows byte and halfword writes. This allows support for control information embedded in the DMA buffers. See the SPI subsections for more information.

41.2.5.1 Flexcomm register descriptions

41.2.5.1.1 Flexcomm memory map

FLEXCOMM0.SELECT base address: 4008_6000h

FLEXCOMM1.SELECT base address: 4008_7000h

FLEXCOMM2.SELECT base address: 4008_8000h

FLEXCOMM3.SELECT base address: 4008_9000h

FLEXCOMM4.SELECT base address: 4008_A000h

FLEXCOMM5.SELECT base address: 4009_6000h

FLEXCOMM6.SELECT base address: 4009_7000h

FLEXCOMM7.SELECT base address: 4009_8000h

FLEXCOMM8.SELECT base address: 4009_F000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
FF8	Peripheral Select and Flexcomm module ID (PSELID)	32	See section	See section
FFC	Peripheral Identification (PID)	32	See section	0000_0000

41.2.5.1.1.1 Peripheral Select and Flexcomm module ID (PSELID)

The PSELID register identifies the Flexcomm module and the peripheral function supported by each Flexcomm module. The PSELID register provides the means to select one peripheral function for each Flexcomm module.

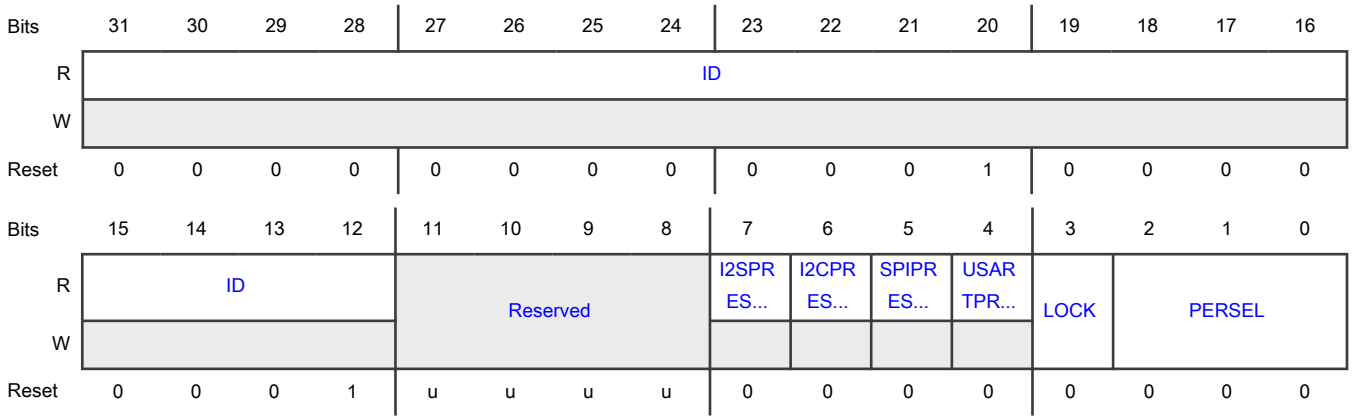
NOTE

Reset value reflects the data stored in used fields only. The reset value does not include reserved field content.

Offset

Register	Offset
PSELID	FF8h

Diagram



Fields

Field	Description
31-12 ID	Flexcomm ID ID is the IP identifier. Bits 31 - 20 indicate the IP version.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	Bits 19 - 12 indicate the IP revision. Flexcomm ID = 0x00103
11-8 —	Reserved Read value is undefined, only zero should be written.
7 I2SPRESENT	I2S Present I^2S present indicator. This field is read-only. 0 - I2S Not Present. This Flexcomm module does not include the I^2S function. 1 - I2S Present. This Flexcomm module includes the I^2S function.
6 I2CPRESENT	I2C present indicator This field is read-only. 0 - I2C Not Present. This Flexcomm module does not include the I^2C function. 1 - I2C Present. This Flexcomm module includes the I^2C function.
5 SPIPRESENT	SPI present indicator This field is read-only. 0 - This Flexcomm module does not include the SPI function. 1 - This Flexcomm module includes the SPI function.
4 USARTPRESENT	USART present indicator This field is read-only. 0 - This Flexcomm module does not include the USART function. 1 - This Flexcomm module includes the USART function.
3 LOCK	Lock the peripheral select This field is writable by software. 0 - Peripheral select can be changed by software. 1 - Peripheral select is locked and cannot be changed until this Flexcomm module or the entire device is reset.
2-0 PERSEL	Peripheral Select This field is writable by software. 000 - No peripheral selected. 001 - USART function selected 010 - SPI function selected 011 - I2C. I^2C function selected

Table continues on the next page...

Table continued from the previous page...

Field	Description
	100 - I2S Transmit. I^2S transmit function selected 101 - I2S Receive. I^2S receive function selected 110 - Reserved 111 - Reserved

41.2.5.1.1.2 Peripheral Identification (PID)

After the Flexcomm module is configured for a function, this register confirms the selection by returning the module ID for that function, and identifies the revision of that function. A software driver could make use of this information register to implement module type or revision-specific behavior.

NOTE

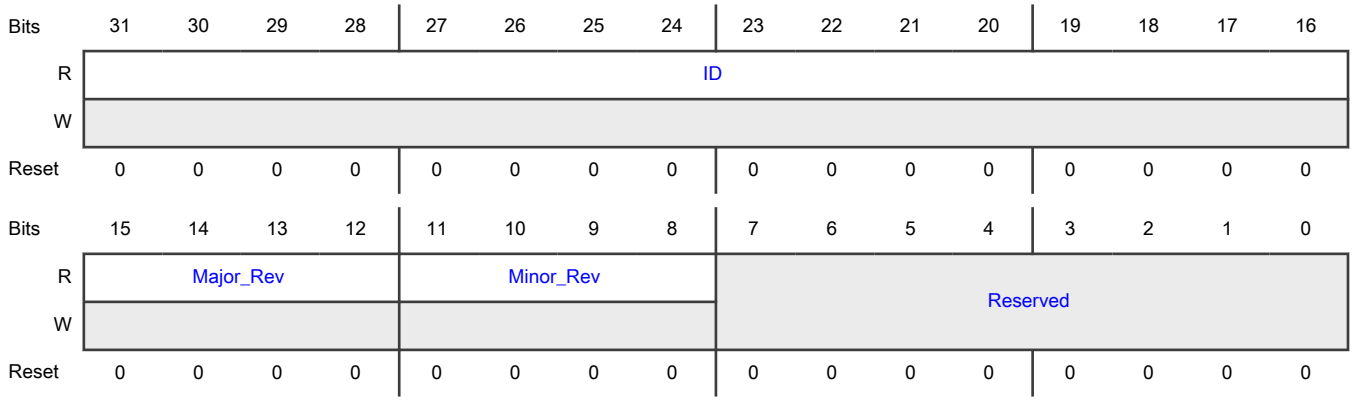
Reset value reflects the data stored in used fields only. The reset value does not include reserved field content.

This register is read-only and will read as 0 until a specific Flexcomm module function is selected through the PID register.

Offset

Register	Offset
PID	FFCh

Diagram



Fields

Field	Description
31-16 ID	Module identifier for the selected function
15-12	Major revision of module implementation

Table continues on the next page...

Table continued from the previous page...

Field	Description
Major_Rev	
11-8 Minor_Rev	Minor revision of module implementation
7-0 —	Reserved Read value is undefined, only zero should be written.

41.3 USART

41.3.1 Overview

This section provides an introduction to the USART function.

- The USART receiver monitors the serial input, RXD, for valid input. The receiver shift register assembles characters as they are received, after which they are passed to the receiver FIFO to await access by the CPU or DMA controller.
- The USART transmitter accepts data written by the CPU or DMA controller to the transmit FIFO. When the transmitter is available, the transmit shift register takes that data, formats it, and serializes it to the serial output, TXD.
- The Baud Rate Generator divides the incoming clock to create an oversample clock (typically 16x) in the standard asynchronous operating mode. The BRG clock input source is the shared Fractional Rate Generator that runs from the USART function clock. The 32 kHz operating mode generates a specially timed internal clock based on the RTC oscillator frequency.
- In Synchronous Slave mode, data is transmitted and received using the serial clock directly. In Synchronous Master mode, data is transmitted and received using the baud rate clock without division.
- Status information from the transmitter and receiver is provided via the STAT register. Many of the status flags are able to generate interrupts, as selected by software. The INTSTAT register provides a view of all interrupts that are both enabled and pending.

41.3.1.1 Block diagram

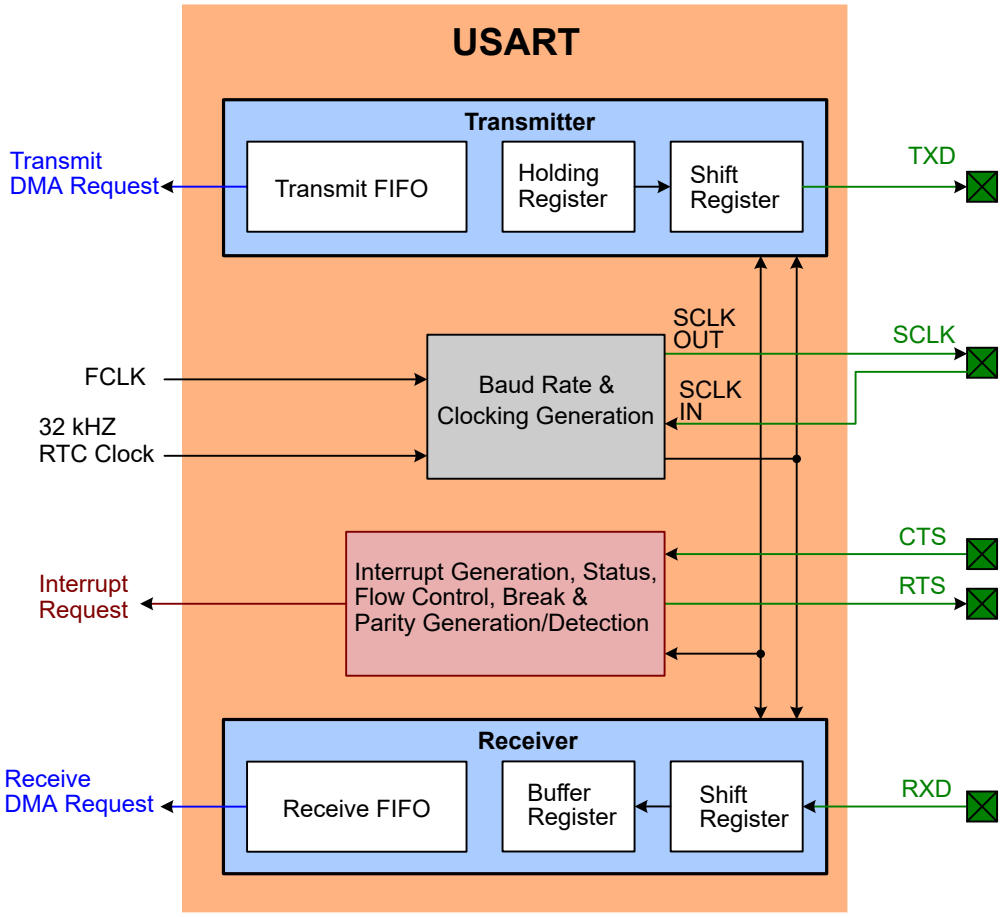


Figure 163. USART block diagram

41.3.1.2 Features

- Synchronous mode with master or slave operation. Includes data phase selection and continuous clock option.
- One transmit and one receive data buffer
- USART transmit and receive functions can operated with the system DMA controller.
- Multiprocessor/multidrop (9-bit) mode with software address compare
- 7, 8, or 9 data bits and 1 or 2 stop bits
- RS-485 transceiver output enable

Transmit and receive FIFOs

- The USART function supports separate transmit and receive FIFOs with 16 entries each.
- Available Interrupts include: FIFO receive level reached, FIFO transmit level reached, FIFO overflow or underflow, Transmitter Idle, change in receiver break detect, Framing error, Parity error, Delta CTS detect, and receiver sample noise detected.

Data checking

- Parity generation and checking: odd, even, or none

- Software selectable oversampling from 5 to 16 clocks in asynchronous mode
- Receive data is 2 out of 3 sample "voting". Status flag set when one sample differs.

Flow control

- RTS/CTS for hardware signaling for automatic flow control. Software flow control can be performed using Delta CTS detect, Transmit Disable control, and any GPIO as an RTS output.
- Loopback mode for testing of data and flow control
- Break generation and detection

Clocking

- Built-in Baud Rate Generator
- Auto-baud mode for automatic baud rate detection
- Special operating mode allows operation at up to 9600 baud using the 32 kHz RTC oscillator as the USART clock. This mode can be used while the device is in Deep-sleep mode and can wake-up the device when a character is received.
- A fractional rate divider is shared among all USARTs

41.3.2 Functional description

41.3.2.1 Operating modes

This section describes all functional operation modes of the USART module.

41.3.2.1.1 Synchronous mode

In synchronous mode, a master generates a clock as defined by the clock selection and the Baud Rate Generator (BRG), which is used to transmit and receive data. As a slave, the external clock is used to transmit and receive data. There is no overclocking or oversampling in synchronous mode.

41.3.2.1.2 Asynchronous mode

This section describes the oversampling and auto-baud function available in asynchronous mode.

41.3.2.1.2.1 Oversampling

Typical industry standard USARTs use a 16x oversample clock to transmit and receive asynchronous data. This is the number of BRG clocks used for one data bit. The Oversample Select Register (OSR) allows this USART to use a 16x down to a 5x oversample clock.

Reducing the oversampling can sometimes help in getting better baud rate matching when the baud rate is very high, or the function clock is very low. For example, the closest actual rate near 115,200 baud with a 12 MHz function clock and 16x oversampling is 107,143 baud, giving a rate error of 7%. Changing the oversampling to 15x gets the actual rate to 114,286 baud, a rate error of 0.8%. Reducing the oversampling to 13x gets the actual rate to 115,385 baud, a rate error of only 0.16%.

There is a cost for altering the oversampling. The USART takes three samples of incoming data on consecutive oversample clocks, as close to the center of a bit time as can be done. When the oversample rate is reduced, the three samples spread out and occupy a larger proportion of a bit time. For example, with 5x oversampling, there is one oversample clock, then three data samples taken, then one more oversample clock before the end of the bit time. Since the oversample clock is running asynchronously from the input data, skew of the input data relative to the expected timing has little room for error. At 16x oversampling, there are several oversample clocks before actual data sampling is done, making the sampling more robust. Generally speaking, it is recommended to use the highest oversampling where the rate error is acceptable in the system.

41.3.2.1.2.2 Auto-baud function (asynchronous mode only)

The auto-baud function attempts to measure the start bit time of the next received character. For this to work, the measured character must have a 1 in the least significant bit position, so that the start bit is bounded by a falling and rising edge. Before an auto-baud operation is requested, the BRG value must be set to 0. The measurement is made using the current clocking settings, including the oversampling configuration. The result is that a value is stored in the BRG register that is as close as possible to the correct setting for the sampled character and the current clocking settings. The sampled character is provided in the RXDAT and RXDATSTAT registers, allowing software to double check for the expected character.

Auto-baud includes a time-out that is flagged by ABERR if no character is received at the expected time. It is recommended that auto-baud only be enabled when the USART receiver is idle. Once enabled, either data will become available in the FIFO or ABERR will be asserted at some point, at which time software should turn off auto-baud.

When the USART is in synchronous mode, auto-baud has no meaning and should not be enabled.

41.3.2.1.3 DMA

A DMA request is provided for each USART direction, and can be used in lieu of interrupts for transferring data by configuring the DMA controller and FIFO level triggering appropriately. The DMA controller provides an acknowledgement signal that clears the related request when it completes handling that request. The transmitter DMA request is asserted when the transmitter can accept more data. The receiver DMA request is asserted when received data is available to be read.

When DMA is used to perform USART data transfers, other mechanisms can be used to generate interrupts when needed. For instance, completion of the configured DMA transfer can generate an interrupt from the DMA controller. Also, interrupts for special conditions, such as a received break, can still generate useful interrupts.

41.3.2.2 Flow control

The USART supports both hardware and software flow control.

41.3.2.2.1 Hardware flow control

The USART supports hardware flow control using RTS and/or CTS signaling. If RTS is configured to appear on a device pin so that it can be sent to an external device, it indicates to an external device the ability of the receiver to receive more data. It can also be used internally to throttle the transmitter from the receiver, which can be especially useful if loopback mode is enabled. RTS is asserted (active low) when the RX FIFO is not full AND the receiver is not receiving new data (i.e. RX is idle). RTS is de-asserted (high) when a new data transmission is in progress. When the new data is received and the RX FIFO is not full, RTS will be asserted indicating it is ready to receive more data. As soon as the new data transmission is started, RTS is de-asserted again and so on.

If connected to a pin, and if enabled to do so, the CTS input can allow an external device to throttle the USART transmitter. Both internal and external CTS can be used separately or together. CTS is sampled continuously with the bit clock by the transmitter. The CTS from external device is synchronized with the bit clock. This CTS_sync is the CTS delayed by 1 to 2 bit clock cycles due to synchronization. When the current data transmission is completed and CTS_sync is asserted, the transmitter will continue to send new data if the TX FIFO is not empty. If CTS_sync is not asserted, the transmitter keeps sampling CTS, and will start sending new data as soon as CTS_sync is asserted.

This figure shows an overview of RTS and CTS within the USART.

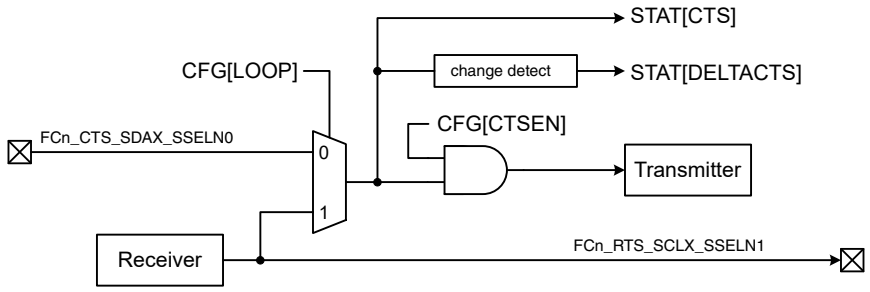


Figure 164. Hardware flow control using RTS and CTS

41.3.2.2.2 Software flow control

Software flow control could include XON/XOFF flow control, or other mechanisms. These are supported by the ability to check the current state of the CTS signal input (STAT[CTS]), and/or have an interrupt when the CTS signal changes state (STAT[DELTACTS]), and by the ability of software to gracefully turn off the transmitter (using the CTL[TXDIS] bit).

41.3.2.3 RS-485 support

RS-485 support requires some form of address recognition and data direction control.

This USART has provisions for hardware address recognition (see CFG[AUTOADDR] and the ADDR register), as well as software address recognition (see CTL[ADDRDET]).

To set up automatic data direction control with the RTS pin, use CFG[OESEL], CFG[OEPOL], and CFG[OETA]. Data direction control can also be implemented in software using a GPIO pin.

41.3.2.4 Break generation and detection

A line break may be sent at any time, regardless of other USART activity. Received break is also detected at any time, including during reception of a character. Received break is signaled when the receive input remains low for 16 bit times. Both the beginning and end of a received break are noted by the STAT[DELTARXBRK] status flag, which can be used as an interrupt. See LIN bus for details of LIN mode break.

To avoid corrupting any character currently being transmitted:

1. Disable the transmitter: CTL[TXDIS] = 1
2. Wait until the transmitter is idle: STAT[TXDISSTAT] = 1
3. Start sending line break: CTL[TXBRKEN] = 1
4. Wait for the desired length
5. Stop sending line break: CTL[TXBRKEN] = 0

41.3.2.5 LIN bus

The only difference between standard mode and LIN (Local Interconnect Network) mode is that LIN mode alters the way that line or received break generation/detection is performed. See Break generation and detection for more information about the standard break process.

When a break is requested (by setting CTL[TXBRKEN] = 1 and then sending a dummy character) a 13-bit time break is sent.

- A received break is flagged when the RX input remains low for 11 bit times.
- For non-LIN mode (standard mode), a received character is also flagged, and accompanied by a framing error status.

As a LIN slave, the auto-baud feature can be used to synchronize to a LIN sync byte, and will return the value of the sync byte as a confirmation of synchronization success.

Wake-up for LIN can be handled in a number of ways, depending on the system, and which clocks may be running in a slave device. For example, as long as the USART is receiving internal clocks (allowing the USART to function), USART can be set to wake up the CPU for any interrupt, including a received start bit. If there are no clocks running, then the GPIO function of the USART RX pin can be programmed to wake up the device.

41.3.2.6 Clocking and baud rates

For the USART, clocking details must be defined such as setting up the clock source selection, the BRG, and setting up the FRG if it is the selected clock source.

Also see Configure the Flexcomm clock and USART baud rate

41.3.2.6.1 Fractional Rate Generator (FRG)

Use the FRG to obtain more precise baud rates when the function clock is not a good multiple of standard (or otherwise desirable) baud rates.

A typical set up has the FRG produce an integer multiple of the highest required baud rate, or a very close approximation. Then use the BRG to obtain the actual baud rate needed.

The FRG register controls the Fractional Rate Generator, which provides the base clock that may be used by any Flexcomm module. The FRG creates a lower rate output clock by suppressing selected input clocks. When FRG is not needed, set FRG = 0, which will then not divide the input clock.

The FRG output clock is defined as the input clock divided by 1 + (MULT / 256), where MULT is in the range of 1 to 255. This produces an output clock that ranges from the input clock divided by 1+1/256 to 1+255/256 (just more than 1 to just less than 2). Make further clock division to each USART block with the integer BRG divider contained in each USART.

The base clock produced by the FRG cannot be perfectly symmetrical, so the FRG distributes the output clocks as evenly as is practical. Since USARTs normally uses 16x overclocking, the jitter in the fractional rate clock in these cases tends to disappear in the ultimate USART output.

41.3.2.6.2 Baud Rate Generator (BRG)

The Baud Rate Generator (see BRG register) is used to divide the base clock to produce a rate 16 times the desired baud rate. Typically, standard baud rates can be generated by integer divides of higher baud rates.

The Baud Rate Generator is a simple 16-bit integer divider controlled by the BRG register. The BRG register contains the value used to divide the Flexcomm clock (FCLK) to produce the clock used for USART internal operations. A 16-bit value allows producing standard baud rates from 300 baud and lower at the highest frequency of the device, up to 921,600 baud from a base clock as low as 14.7456 MHz. Typically, the baud rate clock is 16 times the actual baud rate. This overclocking allows for centering the data sampling time within a bit cell, and for noise reduction and detection by taking three samples of incoming data. Note that in 32 kHz mode, the baud rate generator is still used and must be set to 0 if 9600 baud is required.

NOTE

To change a baud rate after a USART is running, use the following sequence:

1. Make sure the USART is not currently sending or receiving data.
2. Disable the USART by writing CFG[ENABLE] = 0 (0 may be written to the entire register).
3. Write the new BRG[BRGVAL].
4. Set CFG[ENABLE] = 1.

41.3.2.6.3 32 KHz mode

For the 32 KHz clock to operate a USART at a reasonable speed, make these changes:

- Do not use 16x overclocking, because then the maximum data rate would be very low.

- Multiple samples of each data bit must be reduced to one, also to maximize the data rate.
- Use special clocking for individual bit times because 32 KHz is not close to an integer multiple of any standard baud rate.

When the 32 KHz mode is enabled, clocking comes from the RTC oscillator. The FRG is bypassed, and the BRG can be used to divide down the default 9600 baud to lower rates. Other adaptations required to make the USART work for rates up to 9600 baud are done internally. Rate error are less than one half percent in this mode, when the RTC oscillator is operating at the intended frequency of 32.768 kHz.

41.3.3 Signals

The USART receive, transmit, and control signals are movable Flexcomm functions and are assigned to external pins through I/O pin configuration.

Table 341. USART signals

Signal	I/O	Description
TXD	O	Transmitter output for USART on Flexcomm module. Serial transmit data.
RXD	I	Receiver input for USART on Flexcomm module. Serial receive data.
RTS_B	O	Request To Send output for USART on Flexcomm module. This signal supports inter-processor communication through the use of hardware flow control. This signal can also be configured to act as an output enable for an external RS-485 transceiver. RTS is active when the USART RTS signal is configured to appear on a device pin.
CTS_B	I	Clear To Send input for USART on Flexcomm module. Active low signal indicates that the external device that is in communication with the USART is ready to accept data. This feature is active when enabled by the CTSEn bit in CFG register and when configured to appear on a device pin. When deasserted (high) by the external device, the USART will complete transmitting any character already in progress, then stop until CTS is again asserted (low).
SCLK	I/O	Serial clock input/output for USART on Flexcomm module in synchronous mode. Clock input or output in synchronous mode. **NOTE** When the USART is configured as a master, such that SCK is an output, SCK must be connected to a pin in order for the USART to work properly.

41.3.4 Initialization

To initialize the USART module:

- Reset the Flexcomm module that is about to have a specific peripheral function selected.
- Select the USART module function.
- Configure the FIFOs for operation.
- Configure USART for receiving and transmitting data:
  - Enable the clock to the USART register interface.
  - Enable or disable the related Flexcomm module interrupts in the SoC.
  - Configure the related Flexcomm module pin functions in the SoC.
  - Configure the Flexcomm module clock and USART baud rate.

NOTE

The Flexcomm clock frequency should not be above 48 MHz.

- Configure the USART to wake up the chip from any low power modes.
- Configure the USART to receive and transmit data in synchronous slave mode.

See the chip-specific USART information for configuration details.

41.3.4.1 Configure the Flexcomm clock and USART baud rate

Each Flexcomm module has a separate clock selection, which can include a shared fractional divider.

- If a fractional value is needed to obtain a particular baud rate, program the chip.
- In Asynchronous mode: configure the baud rate divider BRG[BRGVAL]. The baud rate divider divides the Flexcomm Clock (FCLK) to create the clock needed to produce the desired baud rate.
- In Synchronous Master mode: The serial clock is SCLK = FCLK / (BRGVAL+1).

The USART can also be clocked by the 32 KHz RTC oscillator. Set the MODE32K bit to enable this 32 KHz mode. See 32 KHz mode

41.3.4.2 Configure the USART for wake-up

A USART can wake up the system from Sleep mode in asynchronous or synchronous mode on any enabled USART interrupt.

In Deep-sleep mode, there are two options for configuring USART for wake-up:

- If the USART is configured for synchronous slave mode, the USART block can create an interrupt on a received signal even when the USART block receives no on-chip clocks - that is in Deep-sleep mode.

  As long as the USART receives a clock signal from the master, it can receive up to one byte in the RXDAT register while in Deep-sleep mode. Any interrupt raised as part of the receive data process can then wake up the part.
- If the 32 KHz mode is enabled, the USART can run in asynchronous mode using the 32 KHz RTC oscillator and create interrupts.

41.3.4.2.1 Wake-up from Sleep mode

- Configure the USART in either asynchronous mode or synchronous mode.
- Enable the USART interrupts in the SoC.
- Any enabled USART interrupt wakes up the part from Sleep mode.

41.3.4.2.2 Wake-up from Deep-sleep mode

- Configure the USART in synchronous slave mode.
- Enable the USART interrupt on the chip.
- The USART wakes up the chip from Deep-sleep mode on all events that are enabled and cause an interrupt. Typical wake-up events are:
  - A start bit has been received.
  - Received data becomes available.
  - In Synchronous mode, data is available in the FIFO to be transmitted, and a serial clock from the master has been received.
  - A change in the state of the CTS pin if the CTS function is connected.

NOTE

By enabling or disabling specific USART interrupts, you can customize when the wake-up occurs.

Wake-up for DMA only

The device can optionally be woken up only far enough to perform a needed DMA before returning to Deep-sleep mode, without waking the CPU. To accomplish this:

- Set up the appropriate USART function to use DMA, and set the related WAKE bit (WAKETX for the transmit function, and WAKERX for the receive function) in the FIFOCFG register.
- Configure the DMA controller, including a transfer complete interrupt.
- Disable the related USART interrupt on the chip.
- Enable the DMA interrupt on the chip.
- Enable the wake-up controls on the chip.

See Chip-specific USART information.

41.3.5 Memory map and register definition

This section includes the USART module memory map and detailed descriptions of all registers.

NOTE

When writing to USART registers, only word accesses are supported. Byte and halfword writes are not supported.

The reset value reflects the data stored in used bits only. It does not include the content of reserved bits. Address offsets are within the related Flexcomm module address space after the USART function has been selected for that Flexcomm module.

41.3.5.1 Flexcomm USART register descriptions

41.3.5.1.1 USART memory map

FLEXCOMM0.USART base address: 4008_6000h

FLEXCOMM1.USART base address: 4008_7000h

FLEXCOMM2.USART base address: 4008_8000h

FLEXCOMM3.USART base address: 4008_9000h

FLEXCOMM4.USART base address: 4008_A000h

FLEXCOMM5.USART base address: 4009_6000h

FLEXCOMM6.USART base address: 4009_7000h

FLEXCOMM7.USART base address: 4009_8000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	USART Configuration (CFG)	32	See section	See section
4	USART Control (CTL)	32	See section	See section
8	USART Status (STAT)	32	See section	See section
C	Interrupt Enable Read and Set for USART (not FIFO) Status (INTENSET)	32	See section	See section

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
10	Interrupt Enable Clear (INTENCLR)	32	See section	See section
20	Baud Rate Generator (BRG)	32	See section	See section
24	Interrupt Status (INTSTAT)	32	See section	See section
28	Oversample Selection Register for Asynchronous Communication (OSR)	32	See section	See section
2C	Address Register for Automatic Address Matching (ADDR)	32	See section	See section
E00	FIFO Configuration (FIFOCFG)	32	See section	See section
E04	FIFO Status (FIFOSTAT)	32	See section	See section
E08	FIFO Trigger Settings for Interrupt and DMA Request (FIFOTRIG)	32	See section	See section
E10	FIFO Interrupt Enable (FIFOINTENSET)	32	See section	See section
E14	FIFO Interrupt Enable Clear (FIFOINTENCLR)	32	See section	See section
E18	FIFO Interrupt Status (FIFOINTSTAT)	32	See section	See section
E20	FIFO Write Data (FIFOWR)	32	See section	See section
E30	FIFO Read Data (FIFORD)	32	See section	See section
E40	FIFO Data Read with No FIFO Pop (FIFORDNOPOP)	32	See section	See section
E48	FIFO Size (FIFOSIZE)	32	RO	0000_0010
E4C	FIFO Receive Timeout Configuration (FIFORXTIMEOUTCFG)	32	See section	0000_0000
E50	FIFO Receive Timeout Counter (FIFORXTIMEOUTCNT)	32	See section	0000_0000
FFC	Peripheral Identification (ID)	32	RO	E010_2100

41.3.5.1.1.1 USART Configuration (CFG)

The CFG register contains communication and mode settings for aspects of the USART that would normally be configured only once in an application.

NOTE

Only the CFG register can be written when the ENABLE = 0. The CFG register can be set up by software with ENABLE = 1, then the rest of the USART can be configured.

NOTE

If software needs to change configuration values, use this sequence:

1. Make sure the USART is not currently sending or receiving data.
2. Disable the USART by writing ENABLE = 0 (0 may be written to the entire register).
3. Write the new configuration value, with ENABLE = 1.

Offset

Register	Offset
CFG	0h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXPOL	RXPOL	OEPOL	OESEL	AUTOADDR	OETA	Reserved	
Reset	u	u	u	u	u	u	u	u	0	0	0	0	0	0	u	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LOOP	SYNCMST	Reserved	CLKPOL	SYNCEN	Reserved	CTSEN	LINMODE	MODE32K	STOPLEN	PARITYSEL		DATALEN		Reserved	ENABLE
Reset	0	0	u	0	0	u	0	0	0	0	0	0	0	0	u	0

Fields

Field	Description
31-24 —	Reserved Read value is undefined; only zero should be written.
23 TXPOL	Transmit data polarity 0 - Standard. Standard. The TX signal is sent out without change. This means that the TX rest value is 1, the start bit is 0, data is not inverted, and the stop bit is 1. 1 - Inverted. Inverted. The TX signal is inverted by the USART before being sent out. This means that the TX rest value is 0, start bit is 1, data is inverted, and the stop bit is 0.
22 RXPOL	Receive Data Polarity 0 - Standard. Standard. The RX signal is used as it arrives from the pin. This means that the RX rest value is 1, start bit is 0, data is not inverted, and the stop bit is 1.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Inverted. Inverted. The RX signal is inverted before being used by the USART. This means that the RX rest value is 0, start bit is 1, data is inverted, and the stop bit is 0.
21 OEPOL	Output Enable Polarity 0 - Low. If selected by OESEL, the output enable is active low. 1 - High. If selected by OESEL, the output enable is active high.
20 OESEL	Output Enable Select 0 - Standard. The RTS signal is used as the standard flow control function. 1 - RS-485. The RTS signal is configured to provide an output enable signal to control an RS-485 transceiver.
19 AUTOADDR	Automatic Address Matching Enable 0 - Disabled. Disabled. When addressing is enabled by CTL[ADDRDET], address matching is done by software. This provides the possibility of versatile addressing (e.g. respond to more than one address). 1 - Enabled. Enabled. When addressing is enabled by CTL[ADDRDET], address matching is done by hardware, using the value in the ADDR register as the address to match.
18 OETA	Output Enable Turnaround Time Enable for RS-485 Operation. 0 - Disabled. Disabled. If selected by OESEL, the Output Enable signal is deasserted at the end of the last stop bit of a transmission. 1 - Enabled. Enabled. If selected by OESEL, the Output Enable signal remains asserted for one character time after the end of the last stop bit of a transmission. The Output Enable signal also remains asserted if another transmit begins before Output Enable is deasserted.
17-16 —	Reserved Read value is undefined; only zero should be written.
15 LOOP	Loopback Mode Selects data Loopback mode. 0 - Normal operation 1 - Loopback mode. Loopback mode. This provides a mechanism to perform diagnostic loopback testing for USART data. Serial data from the transmitter (TXD) is connected internally to serial input of the receiver (RXD). TXD and RTS activity also appear on external pins if these functions are configured to appear on device pins. The receiver RTS signal is also looped back to CTS and performs flow control if enabled by CTSEN.
14 SYNCMST	Synchronous mode Master Select 0 - Slave. When synchronous mode is enabled, the USART is a slave. 1 - Master. When synchronous mode is enabled, the USART is a master.
13 —	Reserved Read value is undefined; only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
12 CLKPOL	Clock Polarity Selects the clock polarity and sampling edge of received data in synchronous mode. 0 - Falling edge. RXD is sampled on the falling edge of SCLK. 1 - Rising edge. RXD is sampled on the rising edge of SCLK.
11 SYNCEN	Synchronous Enable. Selects synchronous or asynchronous operation. 0 - Asynchronous mode 1 - Synchronous mode
10 —	Reserved Read value is undefined; only zero should be written.
9 CTSEN	CTS Enable Determines whether CTS is used for flow control. CTS can be from the input pin, or from the USART's own RTS if Loopback mode is enabled. 0 - No flow control. The transmitter does not receive any automatic flow control signal. 1 - Flow control enabled. The transmitter uses the CTS input (or RTS output in Loopback mode) for flow control purposes.
8 LINMODE	LIN Break Mode Enable 0 - Disabled. Break detect and generate is configured for normal operation. 1 - Enabled. Break detect and generate is configured for LIN bus operation.
7 MODE32K	Mode 32 kHz Selects standard or 32 kHz clocking mode. 0 - Disabled. USART uses standard clocking. 1 - Enabled. Enabled. USART uses the 32 kHz clock from the RTC oscillator as the clock source to the BRG, and uses a special bit clocking scheme.
6 STOPLEN	Stop Length Number of stop bits appended to transmitted data. Only a single stop bit is required for received data. 0 - 1 stop bit 1 - 2 stop bits. This setting should be used only for asynchronous communication.
5-4 PARITYSEL	Parity Select. Selects what type of parity is used by the USART. 00 - No parity 01 - Reserved 10 - Even parity. Even parity. Adds a bit to each character such that the number of 1s in a transmitted character is even, and the number of 1s in a received character is expected to be even.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	11 - Odd parity. Odd parity. Adds a bit to each character such that the number of 1s in a transmitted character is odd, and the number of 1s in a received character is expected to be odd.
3-2 DATALEN	Data Length. Selects the data size for the USART. 00 - 7 bit data length 01 - 8 bit data length 10 - 9 bit data length. The 9th bit is commonly used for addressing in multidrop mode. See the ADDRDET[CTL]. 11 - Reserved
1 —	Reserved Read value is undefined; only zero should be written.
0 ENABLE	USART Enable 0 - Disabled. Disabled. The USART is disabled and the internal state machine and counters are reset. While ENABLE = 0, all USART interrupts and DMA transfers are disabled. When ENABLE is set again, CFG and most other control bits remain unchanged. When re-enabled, the USART is immediately ready to transmit because the transmitter has been reset and is therefore available. 1 - Enabled. The USART is enabled for operation.

41.3.5.1.1.2 USART Control (CTL)

The CTL register controls USART operations that are more likely to change during operation.

Offset

Register	Offset
CTL	4h

Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved						CLRC CON...	CC	Reserved	TXDIS	Reserved			ADDR DET	TXBR KEN	Reserved
Reset	u	u	u	u	u	u	0	0	u	0	u	u	u	0	0	u

Fields

Field	Description
31-17 —	Reserved Read value is undefined; only zero should be written.
16 AUTOBAUD	Autobaud Enable 0 - Disabled. Disabled. USART is in normal operating mode. 1 - Enabled. Enabled. USART is in Autobaud mode. AUTOBAUD should only be set when the USART receiver is idle. The first start bit of RX is measured and used to update the BRG register to match the received data rate. AUTOBAUD is cleared once this process is complete, or if there is an AERR.
15-10 —	Reserved Read value is undefined; only zero should be written.
9 CLRCCONRX	Clear Continuous Clock 0 - No effect. No effect on the CC bit. 1 - Auto-clear. Auto-clear. The CC bit is automatically cleared when a complete character has been received. This bit is cleared at the same time.
8 CC	Continuous Clock Generation By default, SCLK is only output while data is being transmitted in synchronous mode. 0 - Clock on character. Clock on character. In synchronous mode, SCLK cycles only when characters are being sent on TXD or to complete a character that is being received. 1 - Continuous clock. Continuous clock. SCLK runs continuously in synchronous mode, allowing characters to be received on RxD independently from transmission on TXD).
7 —	Reserved Read value is undefined; only zero should be written.
6 TXDIS	Transmit Disable 0 - Not disabled. USART transmitter is not disabled. 1 - Disabled. USART transmitter is disabled after any character currently being transmitted is complete. This feature can be used to facilitate software flow control.
5-3 —	Reserved Read value is undefined; only zero should be written.
2 ADDRDET	Enable Address Detect Mode 0 - Disabled. The USART presents all incoming data. 1 - Enabled. Enabled. The USART receiver ignores incoming data that does not have the most significant bit (msb) of the data (typically the 9th bit) = 1. When the data msb = 1, the receiver treats the incoming data normally, generating a received data interrupt. Software can then check the data to see if this is an address that should be handled. If it is, the ADDRDET bit is cleared by software and further incoming data is handled normally.

Table continues on the next page...

Table continued from the previous page...

Field	Description
1 TXBRKEN	Break Enable A break may be sent without danger of corrupting any currently transmitting character if the transmitter is first disabled (CTL[TXDIS] = 1) and then waiting for the transmitter to be disabled (STAT[TXDISINT] = 1) before writing 1 to TXBRKEN. 0 - Normal operation 1 - Continuous break. Continuous break. Continuous break is sent immediately when this bit is set, and remains until this bit is cleared.
0 —	Reserved Read value is undefined; only zero should be written.

41.3.5.1.1.3 USART Status (STAT)

The STAT register primarily provides a set of USART status flags (not including FIFO status) for software to read. Flags other than read-only flags may be cleared by writing ones to corresponding bits in STAT. Interrupt status flags that are read-only and cannot be cleared by software, can be masked using the INTENCLR register.

The error flags for received noise, parity error, and framing error are set immediately upon detection and remain set until cleared by software action in STAT.

Offset

Register	Offset
STAT	8h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															ABERR
W																W1C
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	RXNOISE...	PARITYE...	FRAMERR...	START	DELTARX...	RXBRK	Reserved	Reserved	Reserved	TXDISST...	DELTACTS	CTS	TXIDLE	Reserved	RXIDLE	Reserved
W	W1C	W1C	W1C	W1C	W1C						W1C					
Reset	u	u	u	u	u	0	u	u	u	0	u	0	1	u	1	u

Fields

Field	Description
31-17 —	Reserved Read value is undefined; only zero should be written.
16 ABERR	Auto Baud Error An error can occur if the BRG counts to its limit before the end of the start bit that is being measured, essentially an auto baud time-out.
15 RXNOISEINT	Received Noise Interrupt Flag Three samples of received data are taken in order to determine the value of each received data bit, except in synchronous mode. This sampling acts as a noise filter if one sample does not match. RXNOISEINT is set when a received data bit contains one nonmatching sample. This could indicate line noise, a baud rate or character format mismatch, or loss of synchronization during data reception.
14 PARITYERRINT	Parity Error Interrupt Flag PARITYERRINT is set when a parity error is detected in a received character.
13 FRAMERRINT	Framing Error Interrupt Flag FRAMERRINT is set when a character is received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source.
12 START	Start START is set when a start is detected on the receiver input. START's purpose is primarily to allow wake-up from Deep-sleep mode immediately when a start is detected. Cleared by software.
11 DELTARXBRK	Delta Received Break DELTARXBRK is set when a change in the state of receiver break detection occurs. Cleared by software.
10 RXBRK	Received Break RXBRK reflects the current state of the receiver break detection logic. RXBRK is set when the RXD pin remains low for 16 bit times. Note that FRAMERRINT is also set when this condition occurs because the stop bit(s) for the character would be missing. RXBRK is cleared when the RXD pin goes high.
9 —	Reserved Read value is undefined; only zero should be written.
8 —	Reserved Read value is undefined; only zero should be written.
7 —	Reserved Read value is undefined; only zero should be written.
6 TXDISSTAT	Transmitter Disabled Status Flag 0 - Not Idle. Indicates that the USART transmitter is NOT fully idle after being disabled.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Idle. Indicates that the USART transmitter is fully idle after being disabled (CTL[TXDIS] = 1).
5 DELTACTS	Delta CTS DELTACTS is set when the state of CTS changes. DELTACTS is cleared by software.
4 CTS	CTS value CTS reflects the current state of the CTS signal, regardless of the setting of the CFG[CTSEN] bit. The CTS bit has the same value as the value of the CTS input pin unless Loopback mode is enabled.
3 TXIDLE	Transmitter Idle TXIDLE is read only. 0 - The transmitter is currently sending data. 1 - The transmitter is not currently sending data.
2 —	Reserved Read value is undefined; only zero should be written.
1 RXIDLE	Receiver Idle RXIDLE is read only. 0 - The receiver is currently receiving data. 1 - The receiver is not currently receiving data.
0 —	Reserved Read value is undefined; only zero should be written.

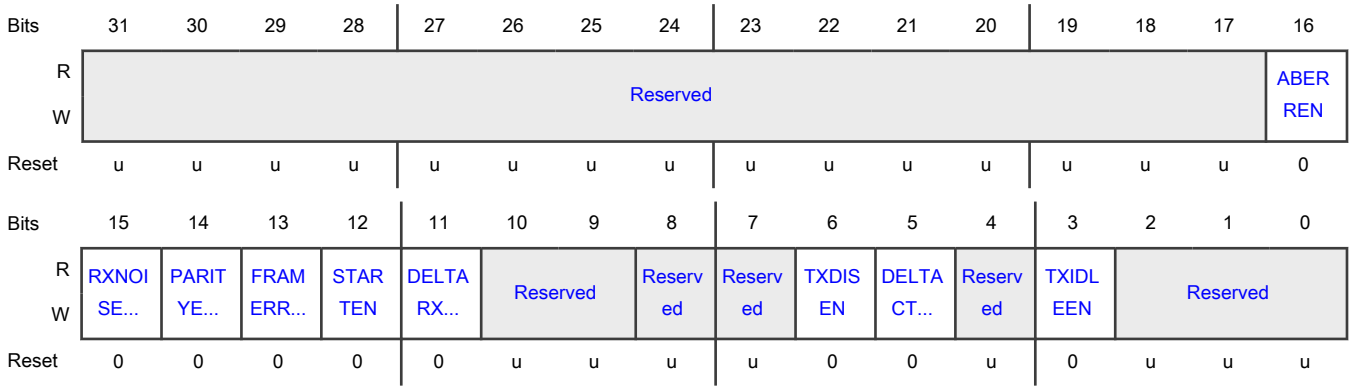
41.3.5.1.1.4 Interrupt Enable Read and Set for USART (not FIFO) Status (INTENSET)

The INTENSET register is used to enable various USART interrupt sources (not including FIFO interrupts). Enable bits in the INTENSET register are mapped in locations that correspond to the flags in the STAT register. Interrupt enables may also be read back from this register. Writing a 1 to any implemented bits in the INTENSET register causes those bits to be set. The INTENCLR register is used to clear bits in the INTENSET register.

Offset

Register	Offset
INTENSET	Ch

Diagram



Fields

Field	Description
31-17 —	Reserved Read value is undefined; only zero should be written.
16 ABERREN	Auto Baud Error Enable 1 - Enables an interrupt when an auto baud error occurs.
15 RXNOISEEN	Receive Noise Enable 1 - Enables an interrupt when noise is detected. See the description of the CTL[RXNOISEINT] bit.
14 PARITYERREN	Parity Error Enable 1 - Enables an interrupt when a parity error has been detected.
13 FRAMERREN	Frame Error Enable 1 - Enables an interrupt when a framing error has been detected.
12 STARTEN	Start Enable 1 - Enables an interrupt when a received start bit has been detected.
11 DELTARXBRKEN	Delta Receive Break Enable 1 - Enable. Enables an interrupt when a change of state has occurred in the detection of a received break condition (break condition asserted or deasserted).
10-9 —	Reserved Read value is undefined; only zero should be written.
8 —	Reserved Read value is undefined; only zero should be written.
7 —	Reserved Read value is undefined; only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
6 TXDISEN	Transmit Disabled Flag 1 - Enables an interrupt when the transmitter is fully disabled as indicated by the STAT[TXDISINT] flag. See the description of the STAT[TXDISINT] flag.
5 DELTACTSEN	Delta CTS Input Flag 1 - Enables an interrupt when there is a change in the state of the CTS input.
4 —	Reserved Read value is undefined; only zero should be written.
3 TXIDLEEN	Transmit Idle Flag 1 - Enables an interrupt when the transmitter becomes idle (STAT[TXIDLE] = 1).
2-0 —	Reserved Read value is undefined; only zero should be written.

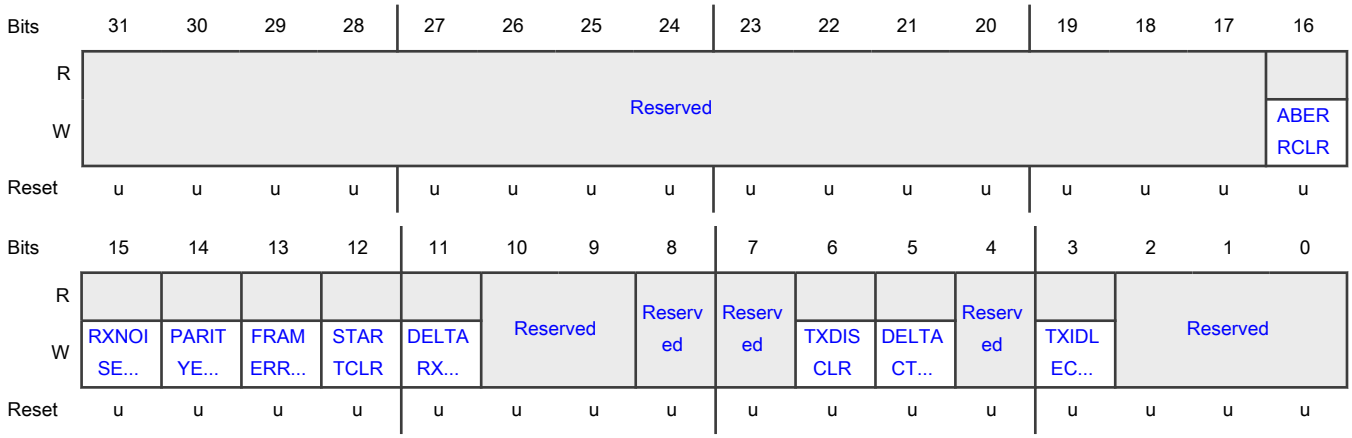
41.3.5.1.1.5 Interrupt Enable Clear (INTENCLR)

Allows clearing any combination of bits in the INTENSET register. Writing a 1 to any implemented bit position causes the corresponding bit to be cleared.

Offset

Register	Offset
INTENCLR	10h

Diagram



Fields

Field	Description
31-17 —	Reserved Read value is undefined; only zero should be written.
16 ABERRCLR	Auto Baud Error Clear Writing 1 clears INTENSET[ABERREN].
15 RXNOISECLR	Receive Noise Clear Writing 1 clears INTENSET[RXNOISEEN].
14 PARITYERRCLR	Parity Error Clear Writing 1 clears INTENSET[PARITYERREN].
13 FRAMERRCLR	Frame Error Clear Writing 1 clears INTENSET[FRAMERREN].
12 STARTCLR	Start Clear Writing 1 clears INTENSET[STARTEN].
11 DELTARXBRKCLR	Delta Receive Break Clear Writing 1 clears INTENSET[DELTARXBRKEN].
10-9 —	Reserved Read value is undefined; only zero should be written.
8 —	Reserved Read value is undefined; only zero should be written.
7 —	Reserved Read value is undefined; only zero should be written.
6 TXDISCLR	Transmit Disable Clear Writing 1 clears INTENSET[TXDISEN].
5 DELTACTSCLR	Delta CTS Clear Writing 1 clears INTENSET[DELTACTSEN].
4 —	Reserved Read value is undefined; only zero should be written.
3 TXIDLECLR	Transmit Idle Clear Writing 1 clears INTENSET[TXIDLEEN].

Table continues on the next page...

Table continued from the previous page...

Field	Description
2-0 —	Reserved Read value is undefined; only zero should be written.

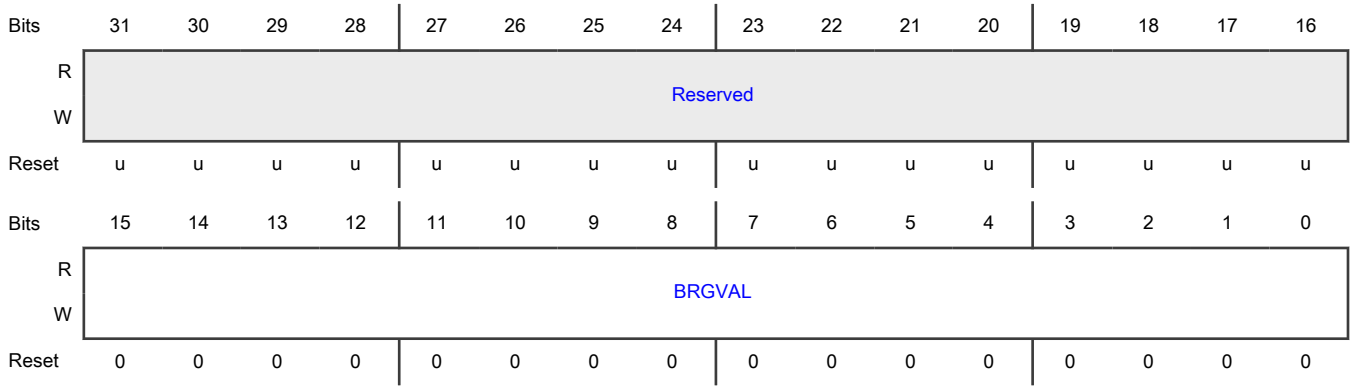
41.3.5.1.1.6 Baud Rate Generator (BRG)

The Baud Rate Generator is a simple 16-bit integer divider controlled by the BRG register. The BRG register contains the value used to divide the Flexcomm Clock (FCLK) to produce the clock used for USART internal operations.

Offset

Register	Offset
BRG	20h

Diagram



Fields

Field	Description
31-16 —	Reserved Read value is undefined; only zero should be written.
15-0 BRGVAL	Baud Rate Generator Value This value is used to divide the USART input clock to determine the baud rate, based on the input clock from the FRG. 0000_0000_0000_0000 - FCLK is used directly by the USART function. 0000_0000_0000_0001 - FCLK is divided by 2 before use by the USART function. 0000_0000_0000_0010 - FCLK is divided by 3 before use by the USART function. 1111_1111_1111_1111 - FCLK is divided by 65,536 before use by the USART function.

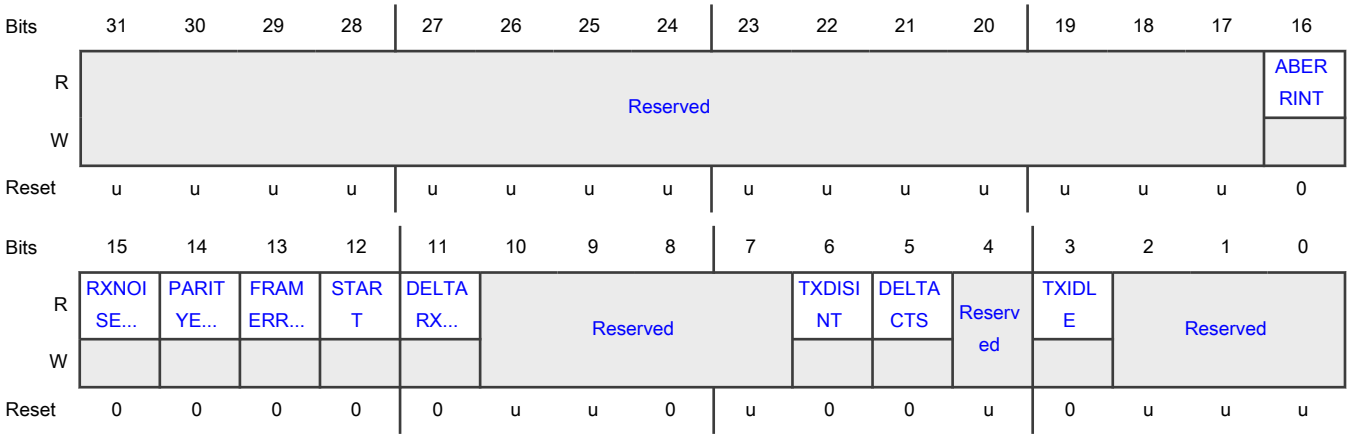
41.3.5.1.1.7 Interrupt Status (INTSTAT)

The read-only INTSTAT register provides a view of the status of the interrupt flags. This can simplify software handling of interrupts. See the STAT register for detailed descriptions of the interrupt flags. The interrupt status bit is set when it is enabled in INTENSET register and is set in the STAT register.

Offset

Register	Offset
INTSTAT	24h

Diagram



Fields

Field	Description
31-17 —	Reserved Read value is undefined; only zero should be written.
16 ABERRINT	Auto Baud Error Interrupt Flag
15 RXNOISEINT	Received Noise Interrupt Flag
14 PARITYERRINT	Parity Error Interrupt Flag
13 FRAMERRINT	Framing Error Interrupt Flag
12	Start Detected on Receiver Flag

Table continues on the next page...

Table continued from the previous page...

Field	Description
START	This bit is set when a start is detected on the receiver input.
11 DELTARXBRK	Delta Receiver Break Change Flag This bit is set when a change in the state of receiver break detection occurs.
10-7 —	Reserved Read value is undefined; only zero should be written.
6 TXDISINT	Transmitter Disabled Interrupt Flag
5 DELTACTS	Delta CTS Change Flag This bit is set when a change in the state of the CTS input is detected.
4 —	Reserved Read value is undefined; only zero should be written.
3 TXIDLE	Transmitter Idle Flag
2-0 —	Reserved Read value is undefined; only zero should be written.

41.3.5.1.1.8 Oversample Selection Register for Asynchronous Communication (OSR)

The OSR register allows selection of oversampling in asynchronous modes. The oversample value is the number of BRG clocks used to receive one data bit. The default is industry standard 16x oversampling.

Changing the oversampling can sometimes allow better matching of baud rates in cases where the function clock rate is not a multiple of 16 times the expected maximum baud rate. For all modes where the OSR setting is used, the USART receiver takes three consecutive samples of input data in the approximate middle of the bit time. Smaller values of OSR can make the sampling position within a data bit less accurate and may potentially cause more noise errors or incorrect data.

Offset

Register	Offset
OSR	28h

Diagram

Fields

Field	Description
31-4 —	Reserved The value read from a reserved bit is not defined.
3-0 OSRVAL	Oversample Selection Value 0000 - Not supported 0001 - Not supported 0010 - Not supported 0011 - Not supported 0100 - 5 function clocks are used to transmit and receive each data bit. 0101 - 6 function clocks are used to transmit and receive each data bit. 1111 - 16 function clocks are used to transmit and receive each data bit.

41.3.5.1.1.9 Address Register for Automatic Address Matching (ADDR)

The ADDR register holds the value for hardware address matching.

Offset

Register	Offset
ADDR	2Ch

Diagram

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

Fields

Field	Description
31-8 —	Reserved The value read from a reserved bit is not defined.
7-0 ADDRESS	Address The 8-bit value is used with automatic address matching. Used when address detection is enabled (CTL[ADDRDET] = 1) and automatic address matching is enabled (CFG[AUTOADDR] = 1).

41.3.5.1.1.10 FIFO Configuration (FIFOCFG)

This register configures how the module uses the FIFO. Before configuring the FIFO, a peripheral function within the Flexcomm module must be selected.

Offset

Register	Offset
FIFOCFG	E00h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved													POPDBG	EMPTYRX	EMPTYTX
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WAKERX	WAKETX	DMARX	DMATX	Reserved						SIZE		Reserved		ENABLERX	ENABLETX
Reset	0	0	0	0	u	u	u	u	u	u	0	0	u	u	0	0

Fields

Field	Description
31-19 —	Reserved Read value is undefined; only zero should be written.
18 POPDBG	Pop FIFO for Debug Reads 0 - Debug reads of the FIFO do not pop the FIFO. 1 - A debug read will cause the FIFO to pop.
17 EMPTYRX	Empty Command for the Receive FIFO When a 1 is written to EMPTYRX, the RX FIFO is emptied. 0 - No effect 1 - The RX FIFO is emptied.
16 EMPTYTX	Empty Command for the Transmit FIFO When a 1 is written to EMPTYTX, the TX FIFO is emptied. 0 - No effect 1 - The TX FIFO is emptied.
15 WAKERX	Wake-up for Receive FIFO Level WAKERX allows the device to be woken from low power modes (as long as the peripheral function works in that power mode) without enabling the RXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion. The feature is enabled by the Hardware Wake Up Control register in the chip if the feature is supported. 0 - Only enabled interrupts will wake up the device from low power modes. 1 - A device wake-up for DMA will occur if the receive FIFO level reaches the value specified by FIFOTRIG[RXLVL], even when the RXLVL interrupt is not enabled.
14 WAKETX	Wake-up for Transmit FIFO Level WAKETX allows the device to be woken from low power modes (as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion. The feature is enabled by the Hardware Wake Up Control register in the chip if the feature is supported. 0 - Only enabled interrupts will wake up the device from low power modes. 1 - A device wake-up for DMA will occur if the transmit FIFO level reaches the value specified by FIFOTRIG[TXLVL], even when the TXLVL interrupt is not enabled.
13 DMARX	DMA Configuration for Receive 0 - DMA is not used for the receive function. 1 - Triggers DMA for the receive function if the FIFO is not empty. Generally, data interrupts would be disabled if DMA is enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Description
12 DMATX	DMA Configuration for Transmit 0 - DMA is not used for the transmit function. 1 - Triggers DMA for the transmit function if the FIFO is not full. Generally, data interrupts would be disabled if DMA is enabled.
11-6 —	Reserved Read value is undefined; only zero should be written.
5-4 SIZE	FIFO Size Configuration This is a read-only field. The settings 0x1, 0x2, 0x3 are not applicable to USART. 00 - FIFO is configured as 16 entries of 8 bits. 01 - Not used 10 - Not used 11 - Not used
3-2 —	Reserved Read value is undefined; only zero should be written.
1 ENABLERX	Enable the Receive FIFO 0 - The receive FIFO is not enabled. 1 - The receive FIFO is enabled.
0 ENABLETX	Enable the Transmit FIFO. 0 - The transmit FIFO is not enabled. 1 - The transmit FIFO is enabled.

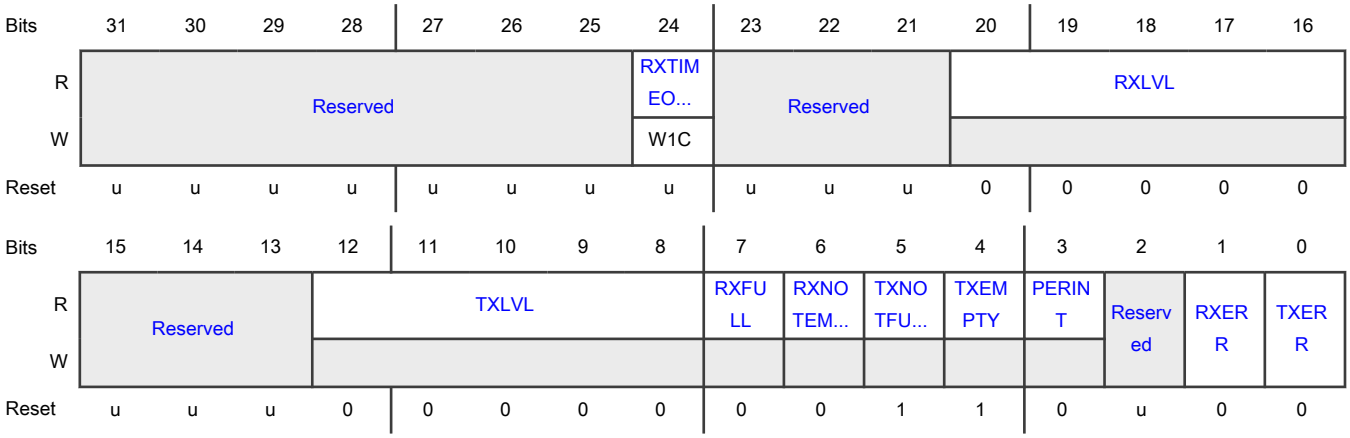
41.3.5.1.1.11 FIFO Status (FIFOSTAT)

Provides status information for the FIFO, and also indicates if a peripheral has issued an interrupt.

Offset

Register	Offset
FIFOSTAT	E04h

Diagram



Fields

Field	Description
31-25 —	Reserved Read value is undefined; only zero should be written.
24 RXTIMEOUT	Receive FIFO Timeout Writing 1 clears this bit and lowers the corresponding interrupt. 0 - RX FIFO on 1 - RX FIFO has timed out, based on the timeout configuration in the FIFORXTIMEOUTCFG register.
23-21 —	Reserved Read value is undefined; only zero should be written.
20-16 RXLVL	Receive FIFO Current Level RXLVL tells how much data is in the RX FIFO when the read occurs. RXLVL = 0 means the RX FIFO is currently empty, and the RXFULL and RXNOTEMPTY flags will be 0. If the RX FIFO is full, then the RXFULL and RXNOTEMPTY flags will be 1.
15-13 —	Reserved Read value is undefined; only zero should be written.
12-8 TXLVL	Transmit FIFO Current Level TXLVL tells how much data is in the TX FIFO when the read occurs. TXLVL = 0 means the TX FIFO is currently empty, and the TXEMPTY and TXNOTFULL flags will be 1. If the TX FIFO is full, then the TXEMPTY and TXNOTFULL flags will be 0.
7 RXFULL	Receive FIFO is Full 0 - The receive FIFO is not full. The receive FIFO is not full. 1 - The receive FIFO is full. The receive FIFO is full. To prevent the peripheral from causing an overflow, data should be read out.

Table continues on the next page...

Table continued from the previous page...

Field	Description
6 RXNOTEMPTY	Receive FIFO is Not Empty 0 - The receive FIFO is empty. 1 - The receive FIFO is not empty, so data can be read.
5 TXNOTFULL	Transmit FIFO is Not Full 0 - The transmit FIFO is full and another write would cause it to overflow. 1 - The transmit FIFO is not full, so more data can be written.
4 TXEMPTY	Transmit FIFO Empty 0 - The transmit FIFO is not empty. 1 - The transmit FIFO is empty, although the peripheral may still be processing the last piece of data.
3 PERINT	Peripheral Interrupt 0 - No Peripheral Interrupt. The peripheral function has not asserted an interrupt. 1 - Peripheral Interrupt. Indicates that the peripheral function has asserted an interrupt. More information can be found by reading the peripheral's status register (STAT).
2 —	Reserved Read value is undefined; only zero should be written.
1 RXERR	RX FIFO Error Cleared by writing a 1 to itself. 0 - A receive FIFO overflow has not occurred 1 - A receive FIFO overflow has occurred, caused by software or DMA not emptying the FIFO fast enough
0 TXERR	TX FIFO Error Cleared by writing a 1 to itself. 0 - A transmit FIFO error has not occurred. 1 - A transmit FIFO error has occurred. This error could be an overflow caused by pushing data into a full FIFO, or by an underflow if the FIFO is empty when data is needed.

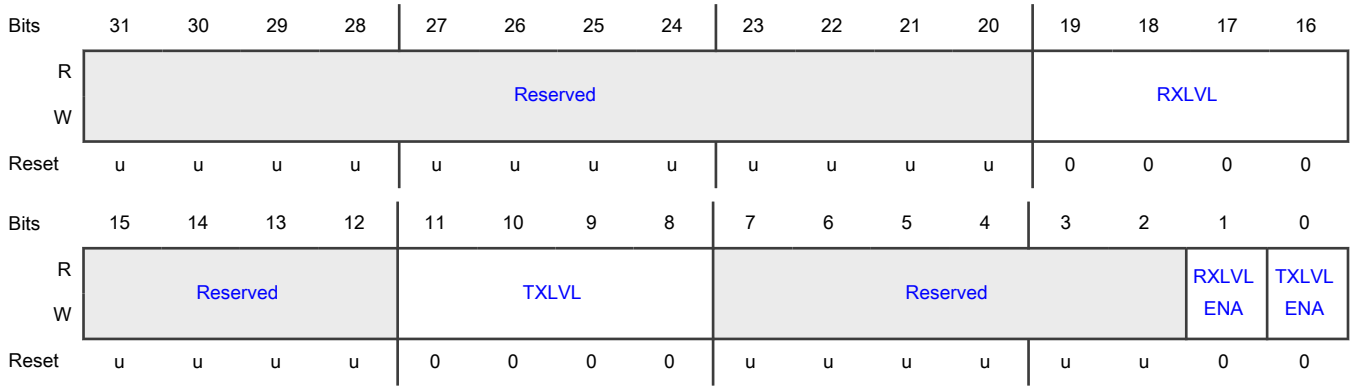
41.3.5.1.1.12 FIFO Trigger Settings for Interrupt and DMA Request (FIFOTRIG)

Allows selecting when FIFO-level related interrupts occur, and includes FIFO trigger settings for interrupt and DMA requests.

Offset

Register	Offset
FIFOTRIG	E08h

Diagram



Fields

Field	Description
31-20 —	Reserved Read value is undefined; only zero should be written.
19-16 RXLVL	Receive FIFO Level Trigger Point The RX FIFO level is checked when a new piece of data is received. RXLVL field is used only when RXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the low power mode. The feature is enabled by the Hardware Wake Up Control register in the chip if the feature is supported. **NOTE** FIFOCFG[WAKERX] allows the device to be woken from reduced power modes. 0000 - Trigger when the RX FIFO has received 1 entry (is no longer empty) 0001 - Trigger when the RX FIFO has received 2 entries 1111 - Trigger when the RX FIFO has received 16 entries (has become full)
15-12 —	Reserved Read value is undefined; only zero should be written.
11-8 TXLVL	Transmit FIFO Level Trigger Point TXLVL field is used only when TXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the low power mode. The feature is enabled by the Hardware Wake Up Control register in the chip if the feature is supported. **NOTE** FIFOCFG[WAKETX] allows the device to be woken from reduced power modes. 0000 - Trigger when the TX FIFO becomes empty 0001 - Trigger when the TX FIFO level decreases to 1 entry 1111 - Trigger when the TX FIFO level decreases to 15 entries (is no longer full)

Table continues on the next page...

Table continued from the previous page...

Field	Description
7-2 —	Reserved Read value is undefined; only zero should be written.
1 RXLVLENA	Receive FIFO Level Trigger Enable RXLVLENA trigger will become an interrupt if enabled in FIFOINTENSET, or will become a DMA trigger if FIFOCFG[DMARX] is set. 0 - Receive FIFO level does not generate a FIFO level trigger. 1 - An trigger will be generated if the receive FIFO level reaches the value specified by the RXLVL field.
0 TXLVLENA	Transmit FIFO Level Trigger Enable. TXLVLENA trigger will become an interrupt if enabled in FIFOINTENSET, or a DMA trigger if DMATX in FIFOCFG is set. 0 - Transmit FIFO level does not generate a FIFO level trigger. 1 - A trigger will be generated if the transmit FIFO level reaches the value specified by the TXLVL field.

41.3.5.1.1.13 FIFO Interrupt Enable (FIFOINTENSET)

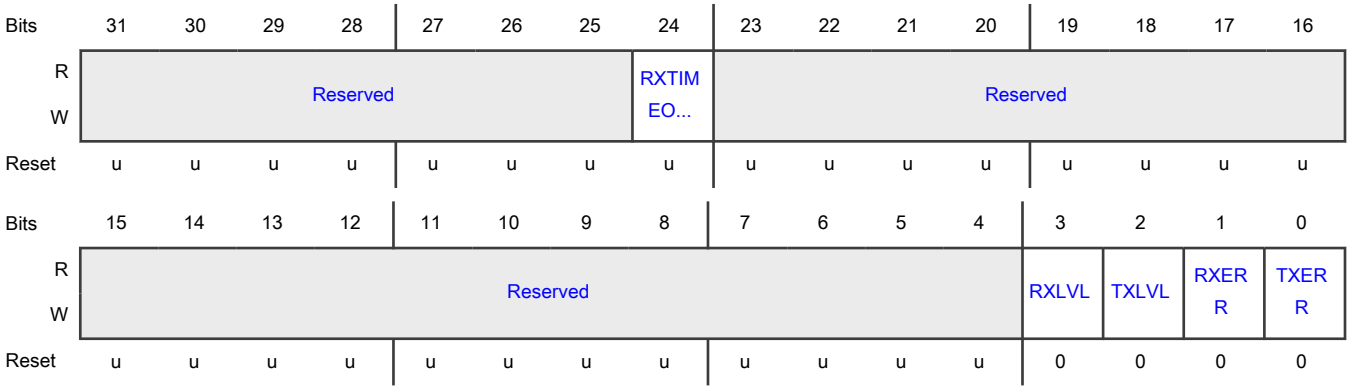
FIFOINTENSET includes the complete set of FIFO interrupt enables. Use the FIFO Interrupt Enable Register to enable various interrupt sources. Writing ones to implemented bits in FIFOINTENSET register forces those bits to be set.

Use the FIFOINTENCLR register to clear the interrupt enable bits in this register.

Offset

Register	Offset
FIFOINTENSET	E10h

Diagram



Fields

Field	Description
31-25 —	Reserved Read value is undefined; only zero should be written.
24 RXTIMEOUT	Receive Timeout The interrupt cause can be cleared by writing 1 to the FIFOSTAT[RXTIMEOUT] bit. 0 - No RX interrupt will be generated. 1 - Asserts RX interrupt if RX FIFO Timeout event occurs.
23-4 —	Reserved Read value is undefined; only zero should be written.
3 RXLVL	Receive FIFO Level Interrupt Enable Determines whether an interrupt occurs when a the receive FIFO reaches the level specified by the FIFOTRIG[RXLVL] field. 0 - No interrupt will be generated based on the RX FIFO level. 1 - If FIFOTRIG[RXLVLENA] = 1, an interrupt will be generated when the when the RX FIFO level increases to the level specified by FIFOTRIG[RXLVL].
2 TXLVL	Transmit FIFO Level Interrupt Enable Determines whether an interrupt occurs when a the transmit FIFO reaches the level specified by the FIFOTRIG[TXLVL] field 0 - No interrupt will be generated based on the TX FIFO level. 1 - If FIFOTRIG[TXLVLENA] = 1, then an interrupt will be generated when the TX FIFO level decreases to the level specified by FIFOTRIG[TXLVL]
1 RXERR	Receive Error Interrupt Enable Determines whether an interrupt occurs when a receive error occurs, based on the FIFOSTAT[RXERR] flag 0 - No interrupt will be generated for a receive error. 1 - An interrupt will be generated when a receive error occurs.
0 TXERR	Transmit Error Interrupt Enable Determines whether an interrupt occurs after a transmit error occurs, based on the FIFOSTAT[TXERR] flag 0 - No interrupt will be generated for a transmit error. 1 - An interrupt will be generated when a transmit error occurs.

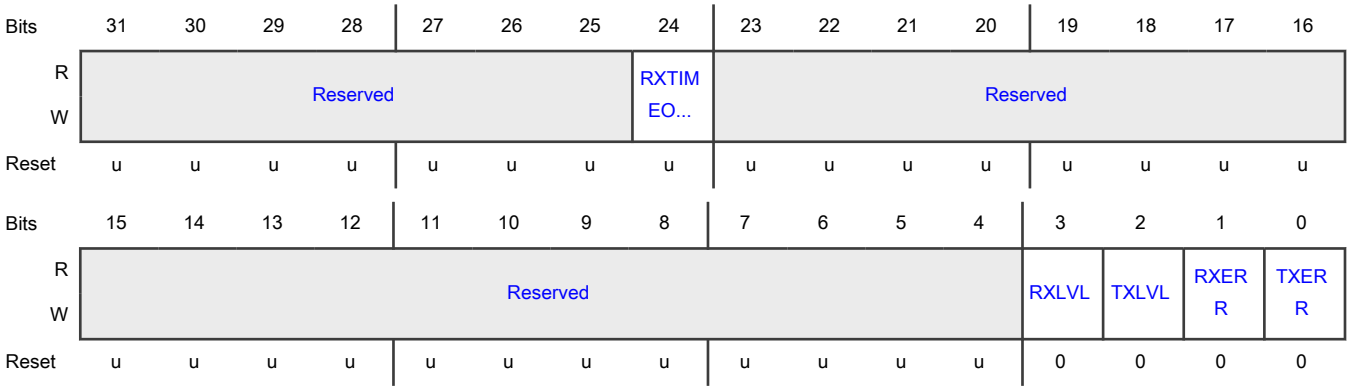
41.3.5.1.1.14 FIFO Interrupt Enable Clear (FIFOINTENCLR)

Use the FIFOINTENCLR register to clear interrupt enable bits in FIFOINTENSET. Writing 1 clears the corresponding bits in the FIFOINTENSET register.

Offset

Register	Offset
FIFOINTENCLR	E14h

Diagram



Fields

Field	Description
31-25 —	Reserved Read value is undefined; only zero should be written.
24 RXTIMEOUT	Receive Timeout Writing 1 will clear this bit; it is described in FIFOINTSET. The cause can be cleared by writing 1 to the FIFOSTAT[RXTIMEOUT]. 0 - No effect 1 - Clear the interrupt
23-4 —	Reserved Read value is undefined; only zero should be written.
3 RXLVL	Receive FIFO Level Interrupt Enable Writing 1 clears FIFOINTENSET[RXLVL] 0 - No effect 1 - Clear the interrupt
2 TXLVL	Transmit FIFO Level Interrupt Enable Writing 1 clears FIFOINTENSET[TXLVL] 0 - No effect 1 - Clear the interrupt
1	Receive Error Interrupt Enable

Table continues on the next page...

Table continued from the previous page...

Field	Description
RXERR	Clear the interrupt 0 - No effect 1 - Clear the interrupt
0 TXERR	Transmit Error Interrupt Enable Writing 1 clears FIFOINTENSET[TXERR] 0 - No effect 1 - Clear the interrupt

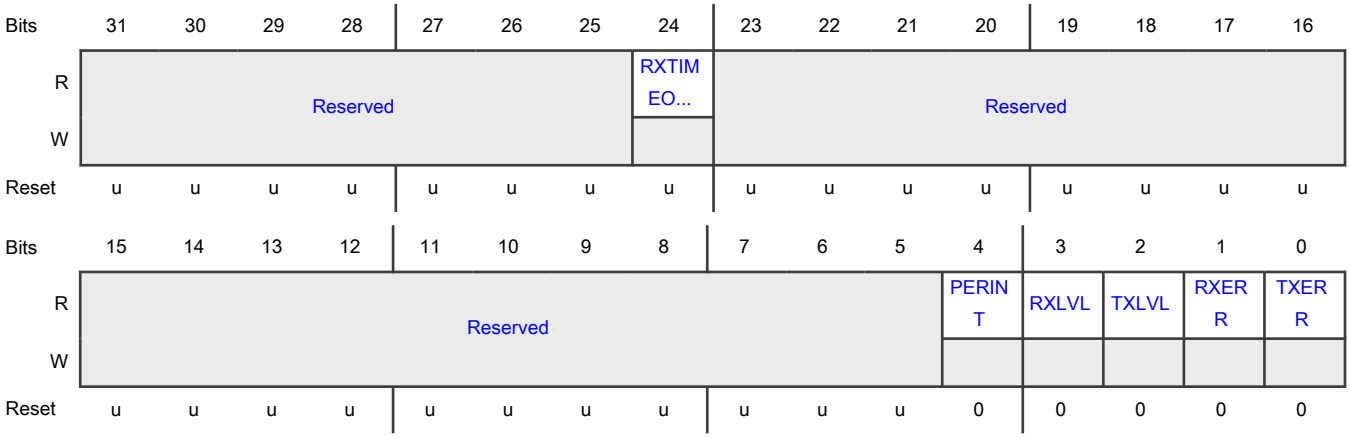
41.3.5.1.1.15 FIFO Interrupt Status (FIFOINTSTAT)

The read-only FIFOINTSTAT register provides a view of the interrupt flags that are pending. This can simplify the software handling of interrupts.

Offset

Register	Offset
FIFOINTSTAT	E18h

Diagram



Fields

Field	Description
31-25 —	Reserved Read value is undefined; only zero should be written.
24	Receive Timeout Status

Table continues on the next page...

Table continued from the previous page...

Field	Description
RXTIMEOUT	1 if both FIFOSTAT[RXTIMEOUT] and FIFOINTSET[RXTIMEOUT] = 1. Note that FIFOSTAT[RXTIMEOUT] may be cleared by writing 1 to it. 0 - Not pending 1 - Pending
23-5 —	Reserved Read value is undefined; only zero should be written.
4 PERINT	Peripheral Interrupt Status 0 - Not pending 1 - Pending
3 RXLVL	Receive FIFO Level Interrupt Status 0 - Not pending 1 - Pending
2 TXLVL	Transmit FIFO Level Interrupt Status 0 - Not pending 1 - Pending
1 RXERR	RX FIFO Error Interrupt Status 0 - Not pending 1 - Pending
0 TXERR	TX FIFO Error Interrupt Status 0 - Not pending 1 - Pending

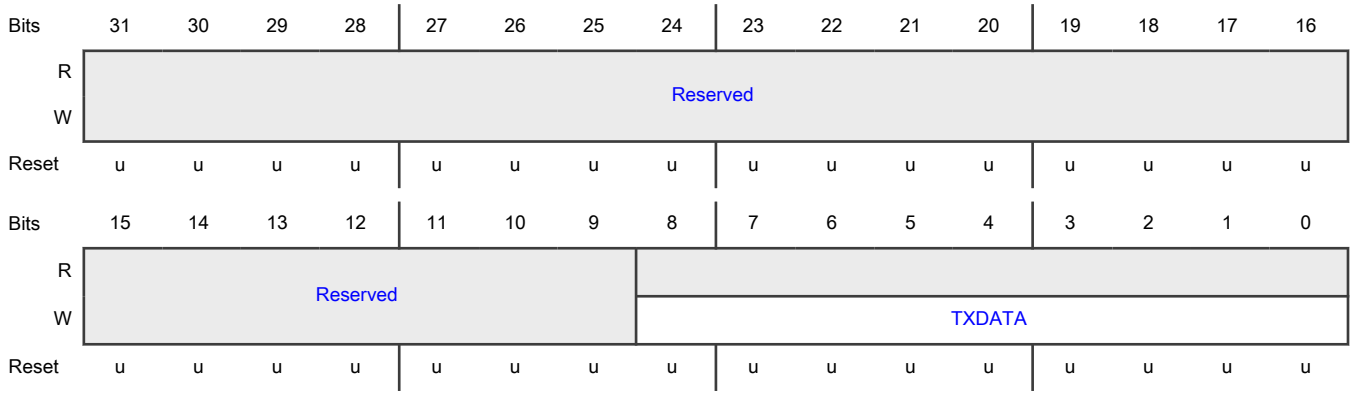
41.3.5.1.1.16 FIFO Write Data (FIFOWR)

The FIFOWR register is used to write values to be transmitted to the FIFO.

Offset

Register	Offset
FIFOWR	E20h

Diagram



Fields

Field	Description
31-9 —	Reserved Read value is undefined; only zero should be written.
8-0 TXDATA	Transmit data to the FIFO

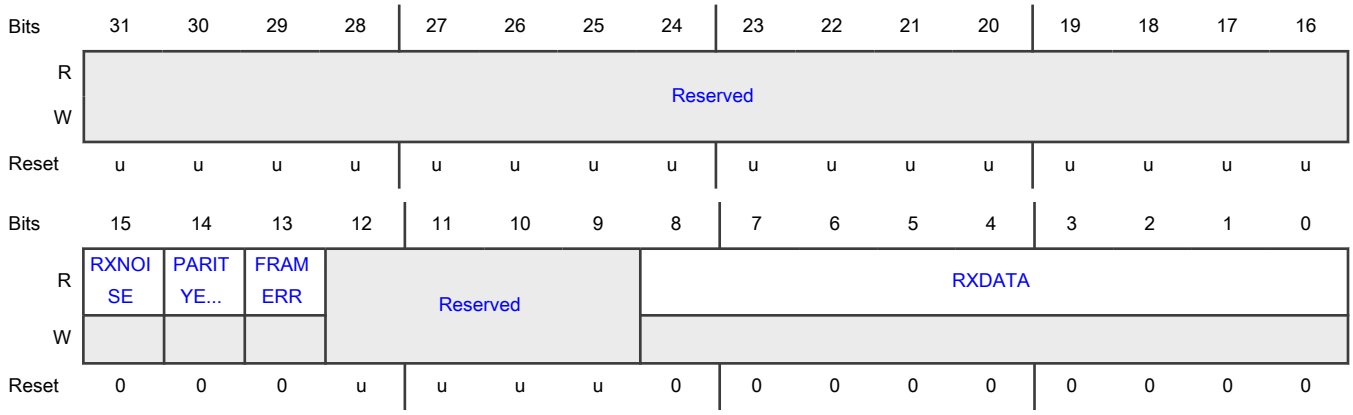
41.3.5.1.1.17 FIFO Read Data (FIFORD)

Use the FIFORD register to read values that have been received by the FIFO.

Offset

Register	Offset
FIFORD	E30h

Diagram



Fields

Field	Description
31-16 —	Reserved The value read from a reserved bit is not defined.
15 RXNOISE	Received Noise Flag See description of STAT[RXNOISEINT].
14 PARITYERR	Parity Error Status Flag PARITYERR reflects the status for the data that is read from the FIFO. PARITYERR will be set when a parity error is detected in a received character.
13 FRAMERR	Framing Error Status Flag FRAMERR reflects the status for the data that is read from the FIFO, and indicates that the character was received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source.
12-9 —	Reserved The value read from a reserved bit is not defined.
8-0 RXDATA	Received Data from the FIFO The number of bits used depends on the CFG[DATALEN] and CFG[PARITYSEL] settings.

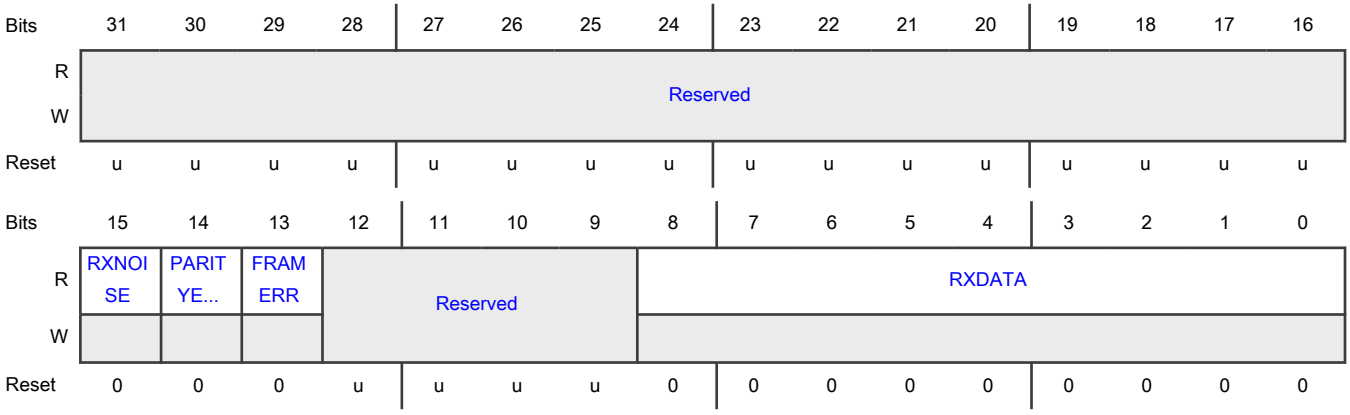
41.3.5.1.1.18 FIFO Data Read with No FIFO Pop (FIFORDNOPOP)

The FIFORDNOPOP register acts in exactly the same way as the FIFORD register, except that FIFORDNOPOP supplies data from the top of the FIFO without popping the FIFO (thereby leaving the FIFO state unchanged). This can be used to allow the system software to observe incoming data without interfering with the peripheral driver.

Offset

Register	Offset
FIFORDNOPOP	E40h

Diagram



Fields

Field	Description
31-16 —	Reserved The value read from a reserved bit is not defined.
15 RXNOISE	Received Noise Flag See description of STAT[RXNOISEINT].
14 PARITYERR	Parity Error Status Flag PARITYERR reflects the status for the data that is read from the FIFO. PARITYERR will be set when a parity error is detected in a received character.
13 FRAMERR	Framing Error Status Flag FRAMERR reflects the status for the data that is read from the FIFO, and indicates that the character was received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source.
12-9 —	Reserved The value read from a reserved bit is not defined.
8-0 RXDATA	Received Data from the FIFO The number of bits used depends on the CFG[DATALEN] and CFG[PARITYSEL] settings.

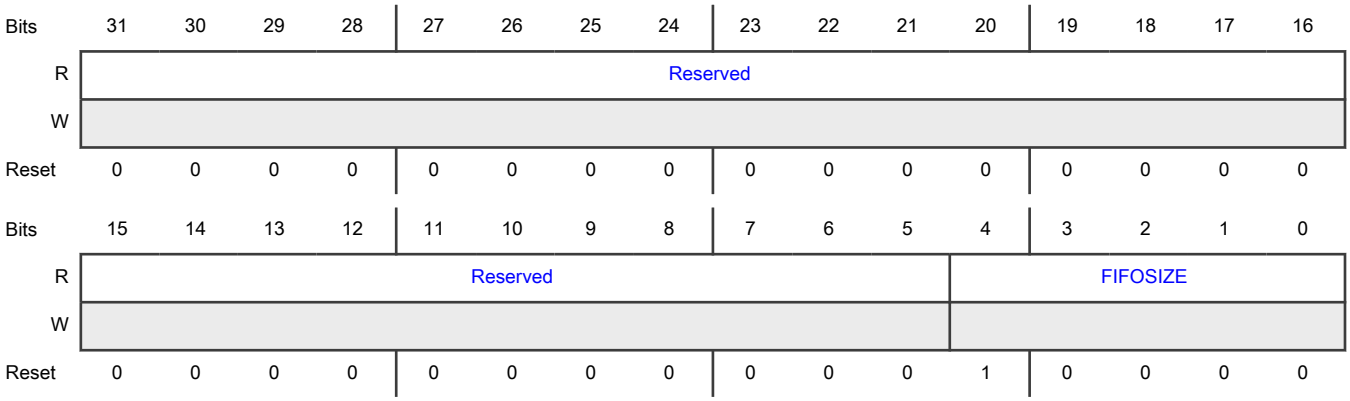
41.3.5.1.1.19 FIFO Size (FIFOSIZE)

Provides the FIFO size.

Offset

Register	Offset
FIFOSIZE	E48h

Diagram



Fields

Field	Description
31-5 —	Reserved
4-0 FIFOSIZE	FIFO Size Provides the size of the FIFO for software. FIFOSIZE is 16 entries for this chip. This field is read only.

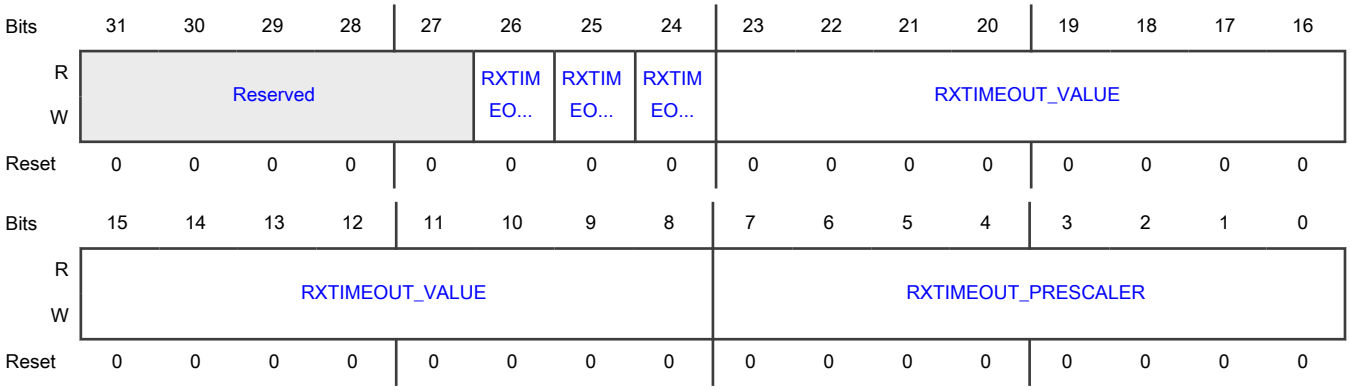
41.3.5.1.1.20 FIFO Receive Timeout Configuration (FIFORXTIMEOUTCFG)

When RXTIMEOUT_EN is enabled, the timeout counter is counting based on the configuration in this FIFORXTIMEOUTCFG register. When the counter reaches FIFORXTIMEOUTCFG[RXTIMEOUT_VALUE] + 1, FIFOSTAT[RXTIMEOUT] is set. The RX interrupt will be asserted if FIFOINTSET[RXTIMEOUT] bit is also set. FIFOSTAT[RXTIMEOUT] can be cleared by writing 1 to it.

Offset

Register	Offset
FIFORXTIMEOUTCFG	E4Ch

Diagram



Fields

Field	Description
31-27 —	Reserved
26 RXTIMEOUT_COE	Receive Timeout Continue On Empty RXTIMEOUT_COE allows the timeout to be used to flag idle peripherals, and could potentially be used to indicate the end of a transmission of indeterminate length. 0 - RX FIFO timeout counter is reset when the RX FIFO becomes empty.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - RX FIFO timeout counter is not reset when the RX FIFO becomes empty.
25 RXTIMEOUT_COW	Receive Timeout Continue On Write RXTIMEOUT_COW allows the timeout to be applied to accumulated data, perhaps related to the FIFO threshold. 0 - RX FIFO timeout counter is reset every time data is transferred from the peripheral into the RX FIFO. 1 - RX FIFO timeout counter is not reset every time data is transferred from the peripheral into the RX FIFO.
24 RXTIMEOUT_EN	Receive Timeout Enable 0 - Disable RX FIFO timeout 1 - Enable RX FIFO timeout
23-8 RXTIMEOUT_VALUE	Receive Timeout Value When RXTIMEOUT_VALUE is enabled and timeout counter reaches RXTIMEOUT_VALUE + 1, FIFOSTAT[RXTIMEOUT] is set.
7-0 RXTIMEOUT_PRESCALER	Receive Timeout Counter Clock Prescaler The clock frequency for FIFORXTIMEOUTCNT[RXTIMEOUT_CNT] is the AHB bus clock frequency divided by 16 * (RXTIMEOUT_PRESCALER + 1).

41.3.5.1.1.21 FIFO Receive Timeout Counter (FIFORXTIMEOUTCNT)

Timeout counter gets reset on one of the following conditions:

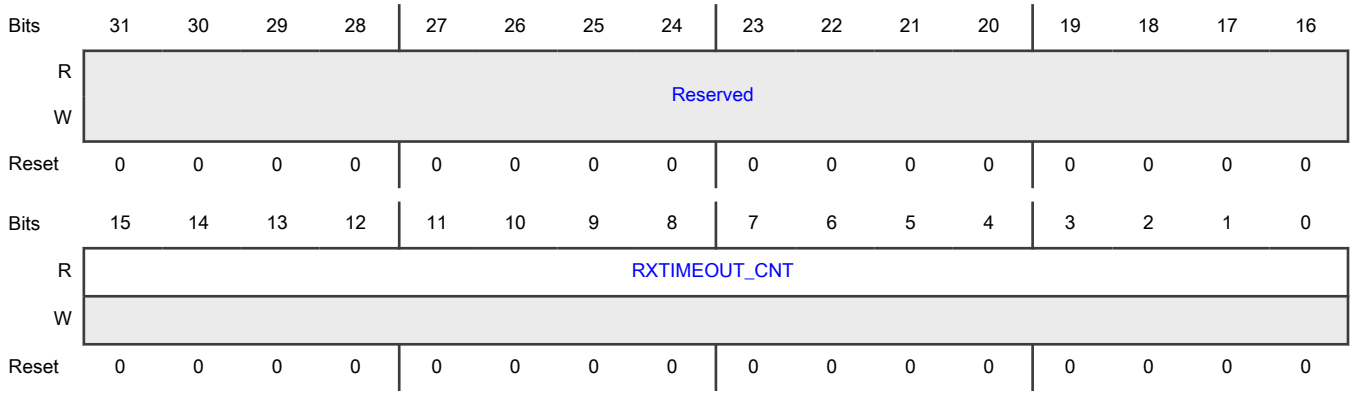
- FIFORXTIMEOUTCFG[RXTIMEOUT_EN] is 0
- FIFOSTAT[RXTIMEOUT] is written with 1
- Rx FIFO is empty and FIFORXTIMEOUTCFG[RXTIMEOUT_COE] is 0
- Rx FIFO is written and FIFORXTIMEOUTCFG[RXTIMEOUT_COW] is 0

Timeout counter is paused when FIFOSTAT[RXTIMEOUT] is set by hardware and is cleared when FIFOSTAT[RXTIMEOUT] is written with 1 by software.

Offset

Register	Offset
FIFORXTIMEOUTCNT	E50h

Diagram



Fields

Field	Description
31-16 —	Reserved
15-0 RXTIMEOUT_CNT	Current RX FIFO timeout counter value

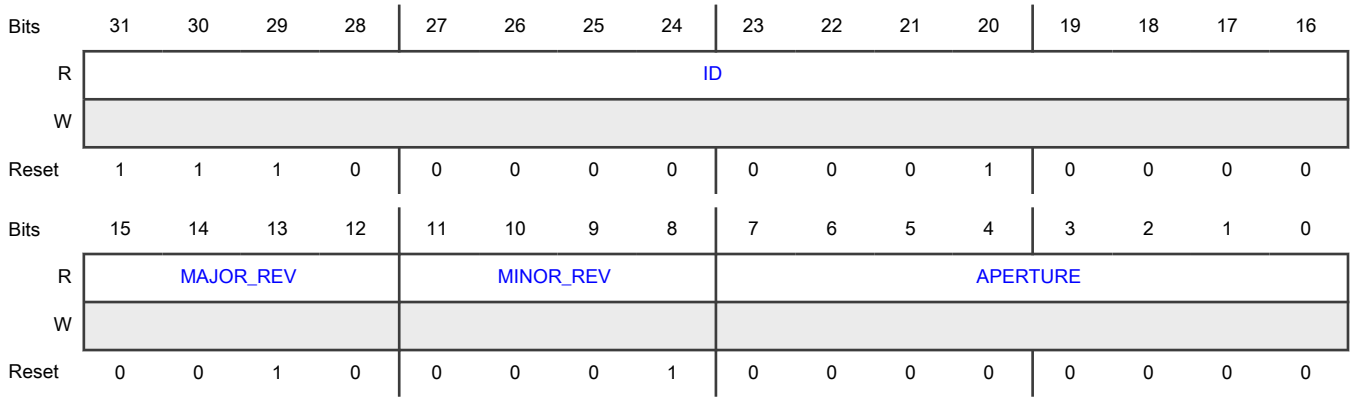
41.3.5.1.1.22 Peripheral Identification (ID)

Identifies the type and revision of the module. A generic software driver can make use of this information to implement specific behavior, based on a module type or a specific revision of that module.

Offset

Register	Offset
ID	FFCh

Diagram



Fields

Field	Description
31-16 ID	Module identifier for the selected function
15-12 MAJOR_REV	Major revision of module implementation
11-8 MINOR_REV	Minor revision of module implementation
7-0 APERTURE	Aperture aperture size = (APERTURE + 1) x 4K APERTURE encoded as 0x00 means 4K aperture: aperture size = (0x00 + 1) x 4K

41.4 Serial Peripheral Interfaces (SPI)

41.4.1 Overview

The SPI is a Serial Peripheral Interface (SPI) module that supports an efficient interface to an SPI bus, either as a master and/or as a slave.

- The SPI is designed to use little CPU overhead, with DMA offloading of FIFO register accesses.
- The SPI can continue operating in stop modes, if an appropriate clock is available.
- The SPI supports DMA accesses and generates a DMA request.

The Serial Peripheral Interface bus (SPI) is a synchronous serial communication interface used in embedded systems, typically to perform short distance communications between microcontrollers and peripheral devices, on printed circuit boards. Typical applications include interfacing to Secure Digital cards and LCD displays.

- SPI devices communicate in full duplex mode using a master-slave scheme with a single master. This enables communication in both directions to happen simultaneously.
- Multiple slave devices are selected using individual slave select (SS) lines.
- The master device originates the frame for reading and writing.
- SPI accesses are always synchronous to the external clock.
- Each SPI can only assert one chip select at a time, but technically there can be multiple SPI slaves sharing the same chip select (in the form of a larger shift register). This requires:
  - the SDO of the master to connect to the SDI of the first slave,
  - the SDO of first slave to connect to the SDI of the next slave,
  - and the SDO of last slave to connect to the SDI of the SPI master.

  Note that some SPI modules use MISO/MOSI for the input/output data.

41.4.1.1 Block diagram

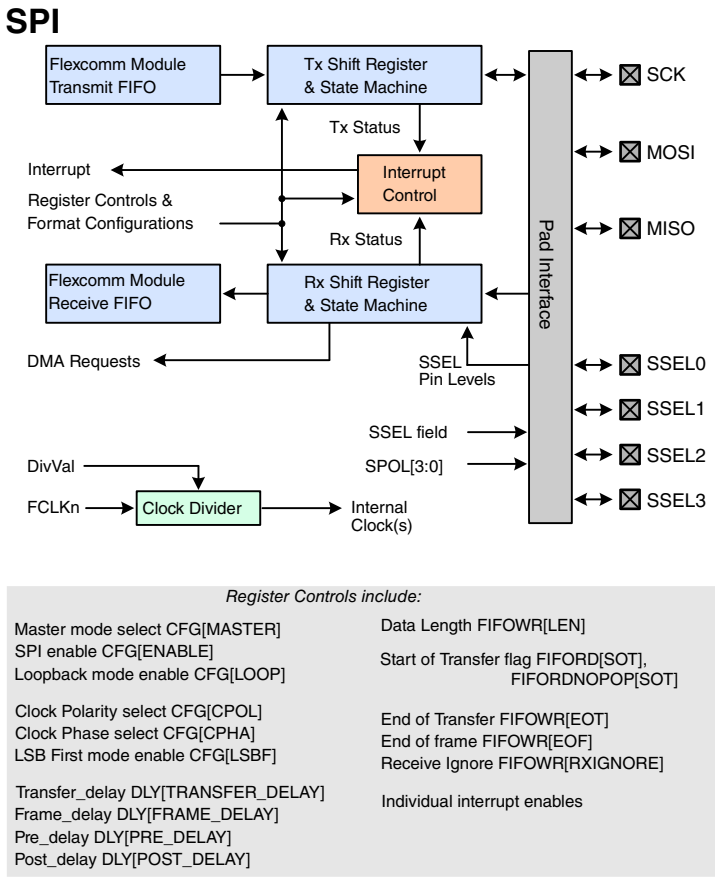


Figure 165. SPI block diagram

41.4.1.2 Features

- Master and slave operations
- Data transmits of 4 to 16 bits supported directly; larger frames are supported by software
- The SPI function supports separate transmit and receive FIFOs with 8 entries each
- Supports DMA transfers: SPIn transmit and receive functions can be operated with the system DMA controller
- Data can be transmitted to a slave without the need to read incoming data,which can be useful while setting up an SPI memory
- Up to 4 Slave Select input/outputs with selectable polarity and flexible usage

NOTE

Texas Instruments SSI and National Microwire modes are not supported.

41.4.2 Functional description

41.4.2.1 Operating modes

This section describes the operating modes of the SPI module.

41.4.2.1.1 Slave select

The SPI block provides for 4 Slave Select (SSEL*n*)inputs in slave mode or 4 Slave Select outputs in master mode. Each SSEL can be set for normal polarity (active low), or can be inverted (active high). Representation of the 4 SSELs in a register is always active low. If an SSEL is inverted, then this signal inversion is performed as the signal leaves or enters the SPI block.

In Slave mode:

- Any asserted SSEL that is connected to a pin will activate the SPI.
- All 4 SSEL states and the received data are saved in the FIFO Read Data Register (FIFORD).

In Master mode:

- All SSELs that are connected to a pin will be output as defined in the SPI registers. The SSELs could potentially be decoded externally, to address more than 4 slave devices. Note that at least 1 SSEL is asserted when data is transferred in master mode.
- Slave Selects come from the TXSSEL*n*_N bits in the FIFO Write Data Register (FIFOWR).

41.4.2.1.2 DMA operation

A DMA request is provided for each SPI direction, and can be used instead of interrupts for transferring data (by configuring the DMA controller, FIFO registers, and INPUTMUX registers appropriately). The DMA controller provides an acknowledgement signal that clears the related request when it (the DMA controller) completes handling that request.

- The transmitter DMA request is asserted when Tx DMA is enabled and the transmitter can accept more data.
- The receiver DMA request is asserted when Rx DMA is enabled and received data is available to be read.

41.4.2.1.2.1 DMA Master Mode End-Of-Transfer

When using polled or interrupt methods to transfer data in master mode, the transition to end-of-transfer status (drive SSEL inactive) is straightforward. The End of Transfer bit (FIFOWR[EOT]) bit would be set just before the last data to be sent or along with the writing of the last data to be sent.

When using the DMA in master mode, the end-of-transfer status (the drive SSEL is inactive) can be generated in 4 ways:

- **Using DMA interrupt and a 2nd DMA transfer:**

  To use only 8 or 16 bit wide DMA transfers for all the data, a 2nd DMA transfer can be used to terminate the transfer (drive SSEL inactive). The transfer would be started by configuring the control bits and then initiating the DMA transfer of all but the last byte/halfword of data. The DMA completion interrupt function must modify the control bits to set the End of Transfer bit (FIFOWR[EOT]) and then configure DMA to send the last data.

- **Using DMA and SPI interrupts (or background SPI status polling):**

  To use only one 8-bit or 16-bit wide DMA transfer for all the data, 2 interrupts will be required to properly terminate the transfer (drive SSEL inactive). The SPI Tx DMA completion interrupt function sets the Transmit FIFO Level Trigger Point (FIFOTRIG[TXLVL]) to 0, and also sets the Transmit FIFO Level Interrupt Enable interrupt enable (FIFOINTENSET[TXLVL]). To force termination after all data output has finished, the interrupt function handling the SPI TXLVL would set the the End Transfer bit (STAT[ENDTRANSFER]).

- **Using DMA linked descriptor:**

  The DMA controller provides for a linked list of DMA transfer control descriptors. The initial descriptor(s) can be used to transfer all but the last data byte/halfword. These data transfers can be done as 8-bit or 16-bit wide DMA operations. A final DMA descriptor, linked to the first DMA descriptor, can be used to send the last data along with control bits to the FIFOWR register. The control bits would include the setting of the End of Transfer bit (FIFOWR[EOT]).

  NOTE

  The DMA interrupt function cannot set the control end transfer bit (STAT[ENDTRANSFER]), because this may terminate the transfer while the FIFO still has data to send.

- **Using 32-bit wide DMA:**

Write both data and control bits to the FIFO Write Data Register (FIFOWR) for all data. The control bits for the last entry would include the setting of the End of Transfer bit (FIFOWR[EOT]). This also allows a series of SPI transactions involving multiple slaves with one DMA operation, by changing the TXSSELn_N bits.

41.4.2.1.3 Data lengths greater than 16 bits

The SPI interface handles data frame sizes from 4 to 16 bits directly. Larger sizes can be handled by splitting data up into groups of 16 bits or less. For example, 24 bits can be supported as 2 groups of 16 bits and 8 bits, or 2 groups of 12 bits, among others. Frames of any size, including frames greater than 32 bits, can supported in the same way.

How to handle larger data widths depends partially on other SPI configuration options. For example, if it is intended for Slave Selects to be deasserted between frames, then the deassertion of Slave Selects must be suppressed when a larger frame is split into more than one part. Sending 2 groups of 12 bits with SSEL deasserted between 24-bit increments, for example, would require changing the value of the End of Frame bit (FIFOWR[EOF]) on alternate 12-bit frames.

41.4.2.1.4 Data stalls

Types of data stalls include:

- **Transmitter stall:**

  A stall for Master transmit data can happen in modes 0 and 2 when SCK cannot be returned to the rest state until the MSB of the next data frame can be driven on MOSI. In this case, if the next piece of data is not yet available, then the stall happens just before the final clock edge of data.

- **Receiver stall:**

  A stall for Master receive can happen when a FIFO overflow would otherwise occur (see RX FIFO Error Interrupt Status bit FIFOSTAT[RXERR]), if the transmitter was not stalled. In modes 0 and 2, this occurs if the FIFO is full when the next piece of data is received. This receiver stall happens 1 clock edge earlier than the transmitter stall.

- **Receiver stall:**

  In modes 1 and 3, the same kind of receiver stall can occur, but just before the final clock edge of the received data. Also, a transmitter stall will not happen in modes 1 and 3, because the transmitted data is complete at the point where a stall would otherwise occur, so it is not needed.

Stalls are reflected by the Stalled status flag (STAT[STALLED]), which indicates the current SPI status. The transmitter will be stalled until the data is read from the receive FIFO. To avoid having to read the received data, use the Receive Ignore control bit (FIFOWR[RXIGNORE]=1).

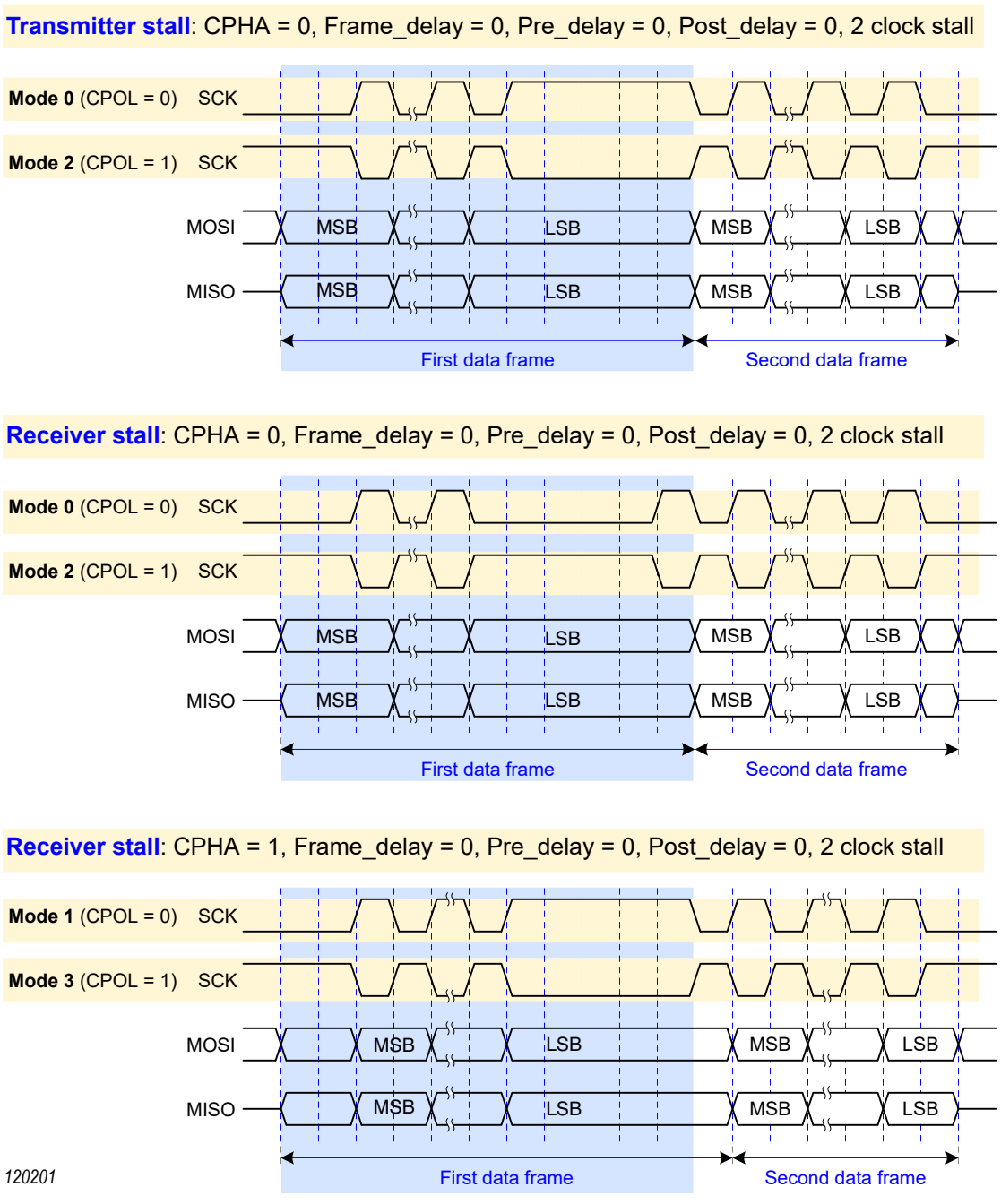


Figure 166. Data stall examples

41.4.2.2 Operations

This section describes the SPI module's operations.

41.4.2.2.1 Frame delays

Several types of delays can be specified for SPI frames:

- Pre-delay: a delay after SSEL is asserted before data clocking begins (DLY[PRE_DELAY])

- Post-delay: a delay at the end of a data frame before SSEL is deasserted (DLY[POST_DELAY])
- Frame delay: a delay between data frames when SSEL is not deasserted (DLY[FRAME_DELAY])
- Transfer delay: a minimum duration of SSEL in the deasserted state between transfers (DLY[TRANSFER_DELAY])

41.4.2.2.1.1 Pre-delay and Post-delay

The pre-delay value (DLY[PRE_DELAY]) controls the amount of time between SSEL being asserted and the beginning of the subsequent data frame. The post-delay value (DLY[POST_DELAY]) controls the amount of time between the end of a data frame and the deassertion of the slave select line (SSEL).

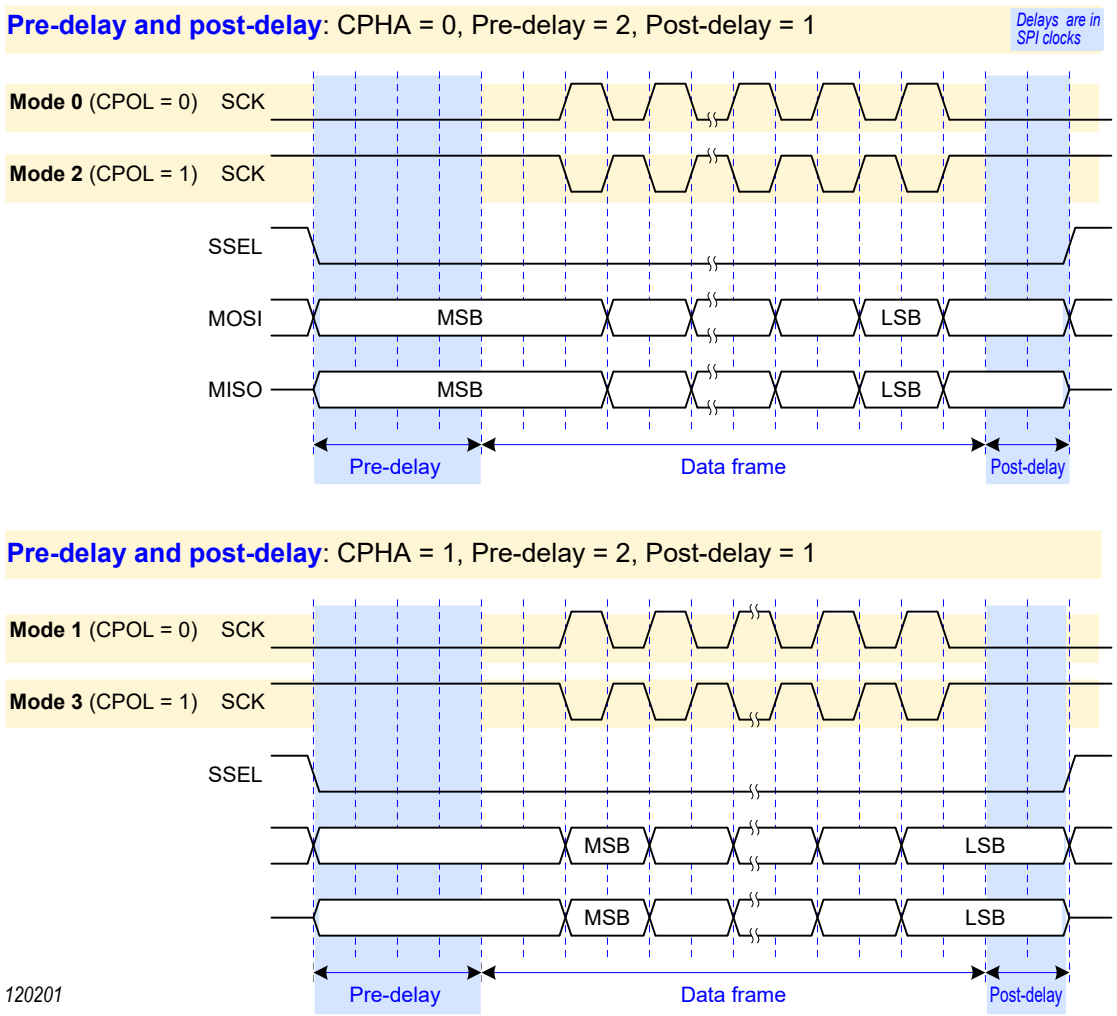


Figure 167. Pre-delay and post-delay

41.4.2.2.1.2 Frame delay

The frame delay value controls the amount of time at the end of each frame. The frame delay value is inserted when the End of frame bit (FIFOWR[EOF]) = 1. Note that frame boundaries occur only where specified, because frame lengths can be any size, involving multiple data writes.

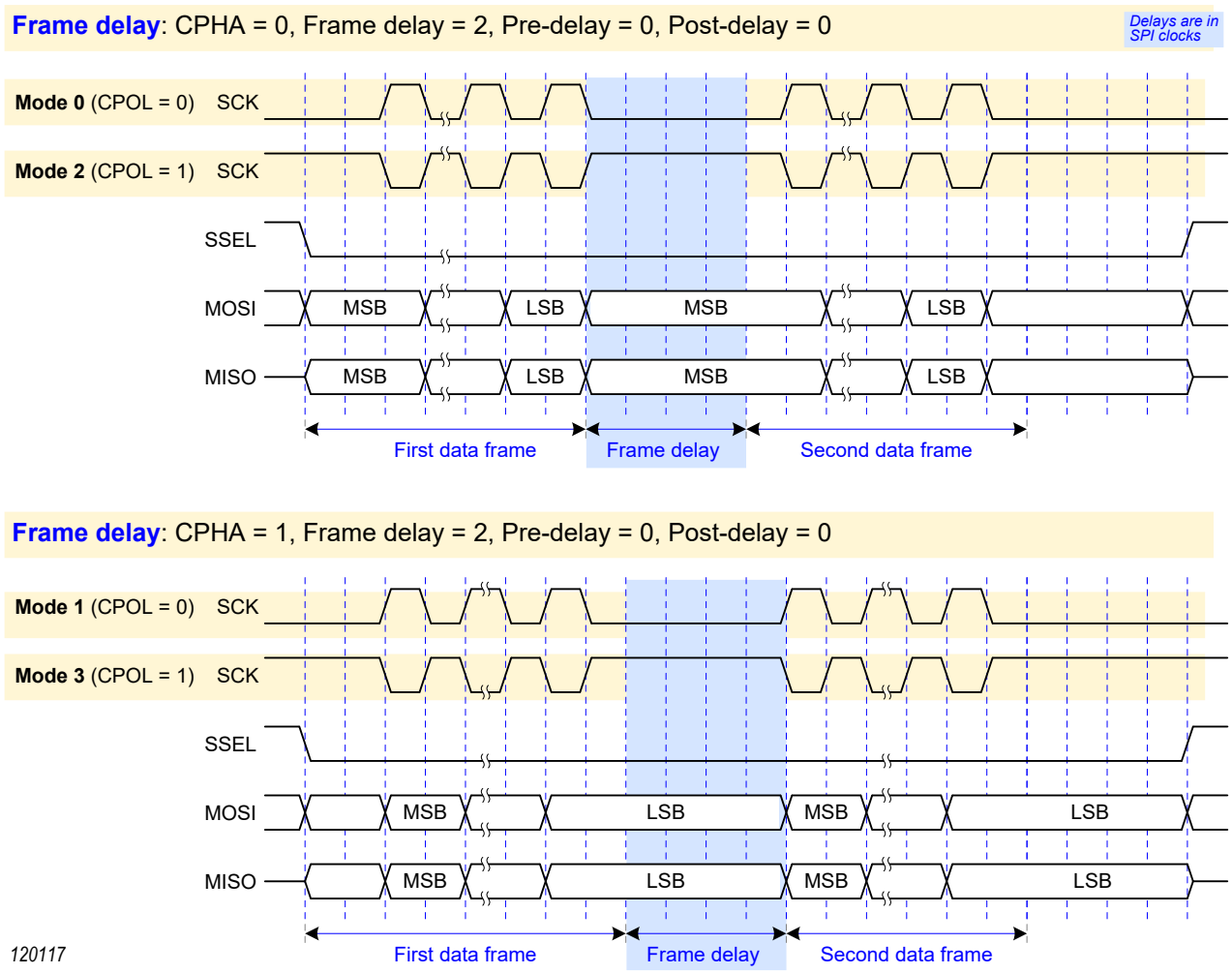


Figure 168. Frame delay

41.4.2.2.1.3 Transfer delay

The transfer delay (DLY[TRANSFER_DELAY]) controls the minimum amount of time that a slave select line (SSEL0 is deasserted between transfers, because the End of Transfer bit = 1 (FIFOWR[EOT]). When the transfer delay = 0 (DLY[TRANSFER_DELAY]), the slave select line (SSEL) may be deasserted for a minimum of 1 SPI clock time.

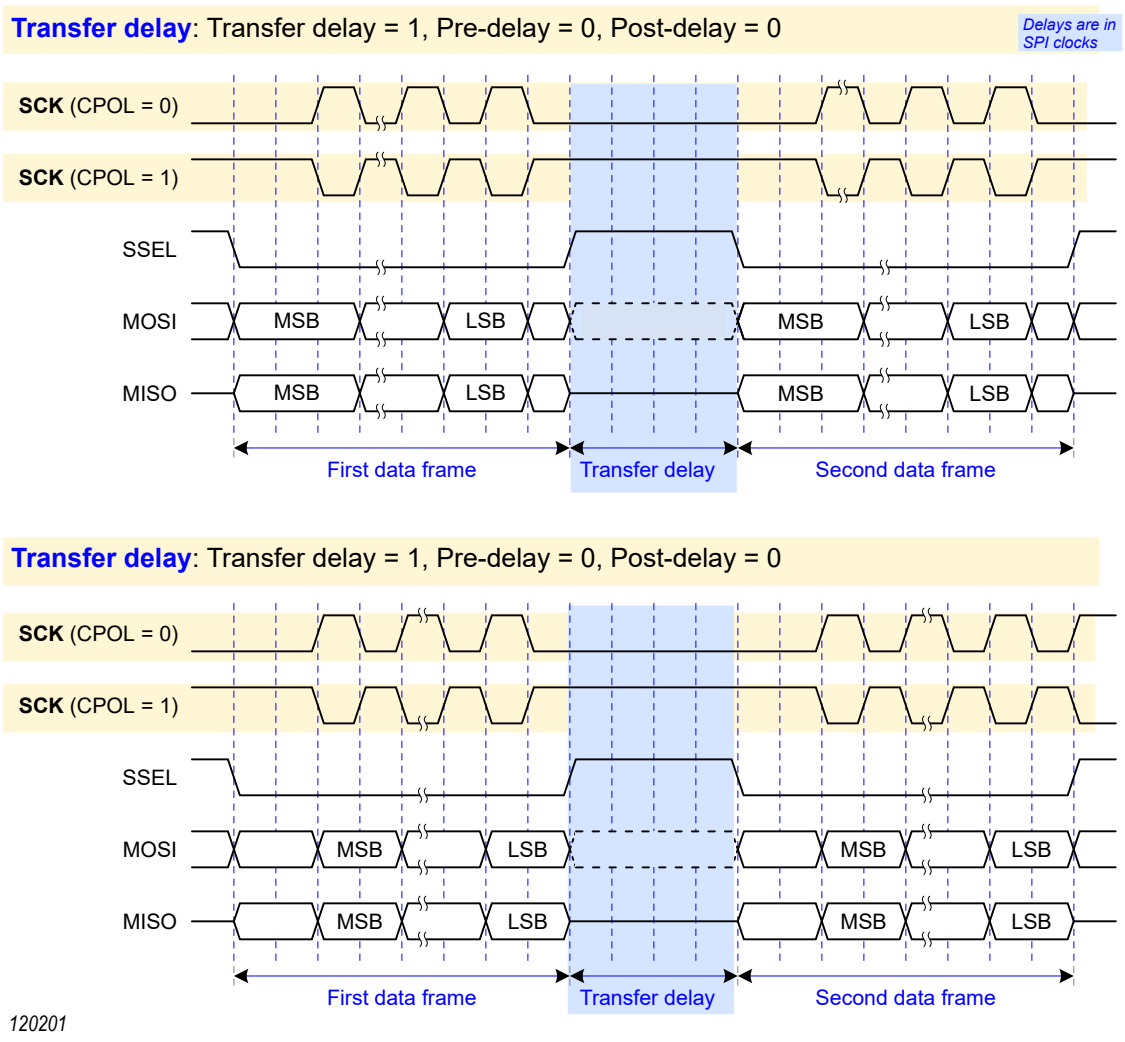


Figure 169. Transfer delay

41.4.2.3 Clocks

This section describes clocks and special clocking requirements of the SPI module.

41.4.2.3.1 Operating modes: clock and phase selection

SPI interfaces typically allow the clock phase and polarity to be configured. These clock configurations are sometimes referred to as numbered SPI modes.

Table 342. SPI mode summary

Clock Polarity Select CPOL	Clock Phase Select CPHA	SPI Mode	Description	SCK rest state	SCK data change edge	SCK data sample edge
0	0	0	The SPI captures serial data on the 1st clock transition of the transfer (when the clock changes away from the rest state). Data is changed on the following edge.	low	falling	rising
0	1	1	The SPI changes serial data on the 1st clock transition of the transfer (when the clock changes away from the rest state). Data is captured on the following edge.	low	rising	falling
1	0	2	Same as mode 0, but with SCK inverted.	high	rising	falling
1	1	3	Same as mode 1, but with SCK inverted.	high	falling	rising

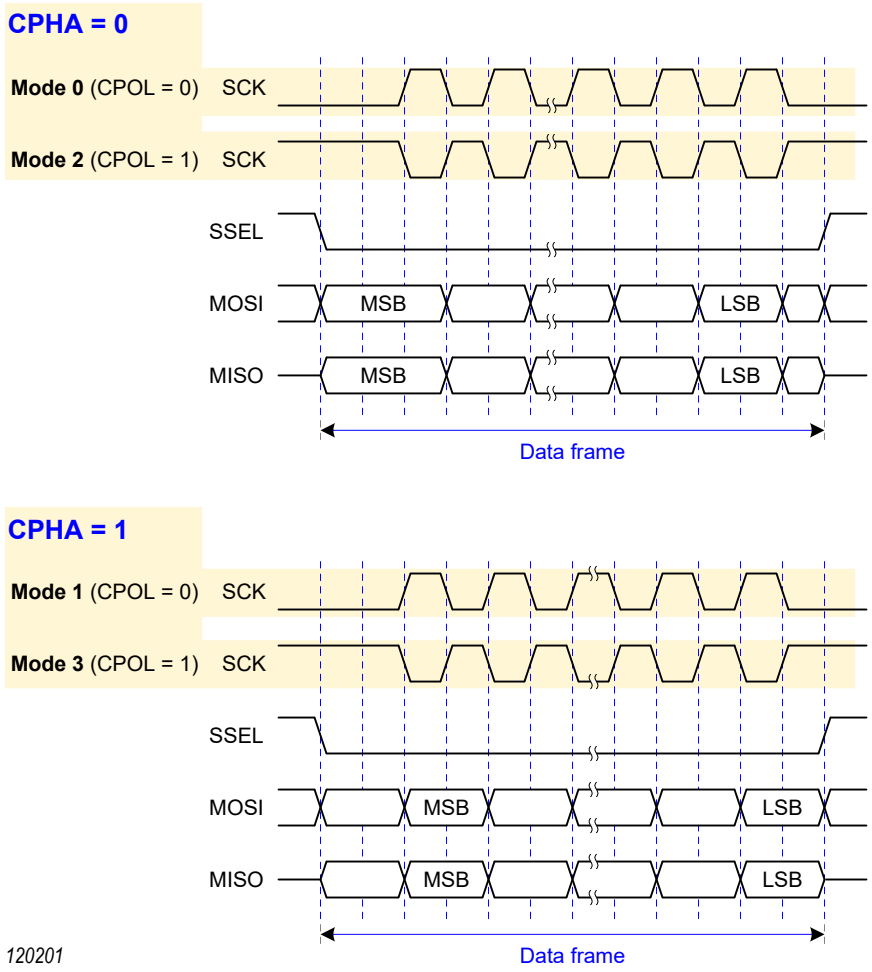


Figure 170. Basic SPI operating modes

41.4.2.3.2 Clocking and data rates

Before using the SPI, the clocking must be defined, which includes configuring the system clock and selecting the clock divider value (DIV[DIVVAL]).

41.4.2.3.2.1 Data rate calculations

The SPI interface is designed to operate asynchronously from any on-chip clocks, and without the need for overclocking.

- In master mode: the SPI rate clock produced by the SPI clock divider is used directly as the outgoing SCK.
- In slave mode: the SCK from the external master is used directly to run the transmit and receive shift registers and other logic.

The SPI clock divider is an integer divider.

- In master mode: the SPI can be set to run at the same speed as the selected FCLK, or at lower integer divide rates. The SPI rate will be = (FCLK of Flexcomm module) / DIVVAL.
- In slave mode: the clock is taken from the SCK input and the SPI clock divider is not used.

41.4.3 Signals

The SPI signals are movable Flexcomm module functions and are assigned to external pins via the SoC's IO controller.

Table 343. SPI signals

Signal	I/O	Description
SCK	I/O	Serial Clock for SPI on Flexcomm module • SCK is a clock signal used to synchronize the transfer of data. • SCK is driven by the master and received by the slave. SCK is driven whenever the Master bit in CFG equals 1, regardless of the state of the Enable bit. • When the SPI interface is used, the clock is programmable to be active-high or active-low. • SCK only switches during a data transfer.
MOSI	I/O	Master-Out Slave-In for SPI on Flexcomm module • The MOSI signal transfers serial data from the master to the slave. • When the SPI is a master, the SPI outputs serial data on this signal. • When the SPI is a slave, the SPI clocks in serial data from this signal. • MOSI is driven whenever the Master bit in CFG equals 1, regardless of the state of the Enable bit.
MISO	I/O	Master In Slave Out for SPI on Flexcomm module • The MISO signal transfers serial data from the slave to the master. • When the SPI is a master, serial data is input from this signal. • When the SPI is a slave, serial data is output to this signal. • MISO is driven when the SPI block is enabled, the Master bit in CFG equals 0, and when the slave is selected by one or more SSEL signals.
SSEL0	I/O	Slave Select 0 for SPI on Flexcomm module

Table continues on the next page...

Table 343. SPI signals (continued)

Signal	I/O	Description
		• When the SPI interface is a master, the SPI will drive the SSEL signals to an active state before the start of serial data, and then release them (the SSEL signals) to an inactive state after the serial data has been sent. By default, this slave select signal is active low, but can be selected to operate as active high. • When the SPI is a slave, any SSEL in an active state indicates that this slave is being addressed. The SSEL pin is driven whenever the Master bit in the CFG register equals 1, regardless of the state of the Enable bit.
SSEL1	I/O	Slave Select 1 for SPI on Flexcomm module
SSEL2	I/O	Slave Select 2 for SPI on Flexcomm module
SSEL3	I/O	Slave Select 3 for SPI on Flexcomm module

41.4.4 Initialization

To initially configure an SPI module:

- Reset the Flexcomm module that is about to have a specific peripheral function selected.
- Select the desired Flexcomm module function.
- Configure the FIFOs for operation.
- Configure the SPI for receiving and transmitting data:
  - Enable the clock to the SPI register interface.
  - Enable or disable the related Flexcomm interrupts in the SoC.
  - Configure the required Flexcomm pin functions in the SoC.
  - Configure the Flexcomm clock (FCLK) and SPI data rate.

NOTE

The Flexcomm clock frequency should not be greater than 48 MHz.

  - Set the Receive Ignore control bit (FIFOWR[RXIGNORE] bit to only transmit data and not read the incoming data. Otherwise, the transmit will halt when the FIFORD buffer is full.
  - Configure the SPI function to wake up the part from low power modes.

41.4.4.1 Configure the SPI for wake-up

- **In sleep mode:**, any signal that triggers an SPI interrupt can wake up the device, if that the interrupt is enabled in the INTENSET register and in the SoC interrupt controller. As long as the SPI clock is configured to be active in sleep mode, the SPI can wake up the part independently of whether the SPI block is configured in master mode or slave mode.
- **In deep-sleep mode:** the SPI clock is turned off. However, if the SPI is configured in slave mode and an external master on the provides the clock signal, then the SPI can create an interrupt asynchronously and wake up the device. The appropriate interrupt(s) must be enabled in the SPI and the SoC interrupt controller.

41.4.4.1.1 Wake-up from sleep mode

- Configure the SPI in either master or slave mode.
- Enable the SPI interrupt in the SoC interrupt controller.

- Any enabled SPI interrupt will now wake up the device from sleep mode.

41.4.4.1.2 Wake-up from deep-sleep mode

- Configure the SPI in slave mode. The SCK function must be connected to a pin and the pin must be connected to the SPI master.
- Enable the SPI interrupt.
- Enable the SPI interrupt in the SoC interrupt controller.
- Enable desired SPI interrupts. For examples, wake-up events can be:
  - A change in the state of the SSEL pins
  - Data available to be received
  - Receive FIFO overflow

Wake-up for DMA only

The device can optionally be woken up only enough to perform the needed DMA operations before returning to deep-sleep mode, without the CPU waking up at all. To do this:

- Set up the appropriate SPI function to use DMA, and set the related WAKE bit (FIFOCFG[WAKETX] for the transmit function; FIFOCFG[WAKERX] for the receive function).
- Configure the DMA controller appropriately, including a transfer complete interrupt.
- Disable the related SPI interrupt in the SoC interrupt controller.
- Enable the DMA interrupt in the SoC interrupt controller.
- Enable HWWAKE[FCWAKE] and HWWAKE[WAKEDMA].

41.4.5 Memory map and register definition

This section includes the SPI module memory map and detailed descriptions of all registers.

NOTE

Except for the FIFOWR register, the bus interface to the SPI registers contained in the Flexcomm module support only word writes. Byte and halfword writes are not supported in the SPI function for those registers. The FIFOWR register also supports byte and halfword (data only) writes, to allow writing FIFO data without affecting the SPI control fields above bit 15.

41.4.5.1 Serial Peripheral Interfaces (SPI) register descriptions

41.4.5.1.1 SPI memory map

FLEXCOMM0.SPI base address: 4008_6000h

FLEXCOMM1.SPI base address: 4008_7000h

FLEXCOMM2.SPI base address: 4008_8000h

FLEXCOMM3.SPI base address: 4008_9000h

FLEXCOMM4.SPI base address: 4008_A000h

FLEXCOMM5.SPI base address: 4009_6000h

FLEXCOMM6.SPI base address: 4009_7000h

FLEXCOMM7.SPI base address: 4009_8000h

FLEXCOMM8.SPI base address: 4009_F000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
400	Configuration Register (CFG)	32	See section	0000_0000
404	Delay Register (DLY)	32	See section	0000_0000
408	Status Register (STAT)	32	See section	0000_0100
40C	Interrupt Enable Register (INTENSET)	32	See section	0000_0000
410	Interrupt Enable Clear Register (INTENCLR)	32	See section	See section
424	Clock Divider Register (DIV)	32	See section	0000_0000
428	Interrupt Status Register (INTSTAT)	32	See section	0000_0000
E00	FIFO Configuration Register (FIFOCFG)	32	See section	See section
E04	FIFO Status Register (FIFOSTAT)	32	See section	0000_0030
E08	FIFO Trigger Register (FIFOTRIG)	32	See section	0000_0000
E10	FIFO Interrupt Enable Register (FIFOINTENSET)	32	See section	0000_0000
E14	FIFO Interrupt Enable Clear Register (FIFOINTENCLR)	32	See section	0000_0000
E18	FIFO Interrupt Status Register (FIFOINTSTAT)	32	See section	0000_0000
E20	FIFO Write Data Register (FIFOWR)	32	See section	See section
E30	FIFO Read Data Register (FIFORD)	32	See section	See section
E40	FIFO Data Read with no FIFO Pop Register (FIFORDNOPOP)	32	See section	See section
E48	FIFO Size Register (FIFOSIZE)	32	RO	See section
E4C	FIFO Receive Timeout Configuration (FIFORXTIMEOUTCFG)	32	See section	0000_0000
E50	FIFO Receive Timeout Counter (FIFORXTIMEOUTCNT)	32	See section	0000_0000
FFC	Peripheral Identification Register (ID)	32	RO	E020_1200

41.4.5.1.1.1 Configuration Register (CFG)

The CFG register is used to initially configure the SPI, and typically the SPI configuration is not changed during normal operations. A setup sequence is recommended for initial SPI setup (after the SPI function has been selected), and for when changes must be made to settings in the CFG register after the SPI interface has been in use.

SPI Setup Sequence: To initialize the SPI interface or to change the existing SPI configuration when the SPI interface is operating:

- Disable the FIFO, by clearing the FIFOCFG[ENABLETX] and FIFOCFG[ENABLERX] bits
- Set up the SPI interface in the CFG register, leaving CFG[ENABLE] = 0
- Enable the FIFO, by setting the FIFOCFG[ENABLETX] and/or FIFOCFG[ENABLERX] bits
- Enable the SPI, by setting the CFG[ENABLE] bit

Also see the Master Idle Status flag (STAT[MSTIDLE]).

Offset

Register	Offset
CFG	400h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	Reserved				SPOL3	SPOL2	SPOL1	SPOL0	LOOP	Reserved	CPOL	CPHA	LSBF	MASTER	Reserved	ENABLE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-12 —	Reserved Read value is undefined; only zero should be written.
11 SPOL3	SSEL3 Polarity Select 0 - Low. The SSEL3 pin is active low. 1 - High. The SSEL3 pin is active high.
10 SPOL2	SSEL2 Polarity Select 0 - Low. The SSEL2 pin is active low. 1 - High. The SSEL2 pin is active high.

Table continues on the next page...

Table continued from the previous page...

Field	Description
9 SPOL1	SSEL1 Polarity Select 0 - Low. The SSEL1 pin is active low. 1 - High. The SSEL1 pin is active high.
8 SPOL0	SSEL0 Polarity Select 0 - Low. The SSEL0 pin is active low. 1 - High. The SSEL0 pin is active high.
7 LOOP	Loopback Mode Enable Loopback mode applies only to Master mode, and has transmit and receive data connected together, to enable simple software testing. 0 - Disabled 1 - Enabled
6 —	Reserved Read value is undefined; only zero should be written.
5 CPOL	Clock Polarity Select 0 - Low. The rest state of the clock (between transfers) is low. 1 - High. The rest state of the clock (between transfers) is high.
4 CPHA	Clock Phase Select 0 - Change. Change. The SPI captures serial data on the 1st clock transition of the transfer (when the clock changes away from the rest state); data is **changed** on the following edge. 1 - Capture. Capture. The SPI changes serial data on the 1st clock data is **captured** on the following edge.
3 LSBF	LSB First Mode Enable 0 - Standard. Data is transmitted and received in standard MSB-first order. 1 - Reverse. Data is transmitted and received in reverse order (LSB first).
2 MASTER	Master Mode Select 0 - Slave mode. The SPI will operate in slave mode. SCK, MOSI, and the SSEL signals are inputs; MISO is an output. 1 - Master mode. The SPI will operate in master mode. SCK, MOSI, and the SSEL signals are outputs; MISO is an input.
1 —	Reserved Read value is undefined; only zero should be written.
0 ENABLE	SPI Enable 0 - Disabled. The SPI is disabled and the internal state machine and counters are reset. 1 - Enabled. The SPI is enabled for operation.

41.4.5.1.1.2 Delay Register (DLY)

The DLY register controls several programmable delays related to SPI signalling. These delays apply only to master mode, and are all stated in the number of SPI clocks. Timing details are shown in the PRE_DELAY, POST_DELAY, FRAME_DELAY, and TRANSFER_DELAY fields.

Offset

Register	Offset
DLY	404h

Diagram

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

Fields

Field	Description
31-16 —	Reserved Read value is undefined; only zero should be written.
15-12 TRANSFER_DELAY	Transfer Delay Controls the minimum amount of time that the SSEL is deasserted between transfers. 0000 - The minimum time that SSEL is deasserted is 1 SPI clock time (zero-added time) 0001 - The minimum time that SSEL is deasserted is 2 SPI clock times 0010 - The minimum time that SSEL is deasserted is 3 SPI clock times 1111 - The minimum time that SSEL is deasserted is 16 SPI clock times
11-8 FRAME_DELAY	Frame Delay • If the End of Frame bit (FIFOWR[EOF]) flag is set, then FRAME_DELAY controls the minimum amount of time between the current frame and the next frame. • If the End of Transfer bit (FIFOWR[EOT]) is set, then FRAME_DELAY controls the minimum amount of time between the current frame and SSEL deassertion. 0000 - No additional time is inserted 0001 - 1 SPI clock time is inserted

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0010 - 2 SPI clock times are inserted 1111 - 15 SPI clock times are inserted
7-4 POST_DELAY	Post-Delay Controls the amount of time between the end of a data transfer and SSEL deassertion. 0000 - No additional time is inserted 0001 - 1 SPI clock time is inserted 0010 - 2 SPI clock times are inserted 1111 - 15 SPI clock times are inserted
3-0 PRE_DELAY	Pre-Delay Controls the amount of time between SSEL assertion and the beginning of a data transfer. There is always 1 SPI clock time between SSEL assertion and the 1st clock edge, and this is not considered part of the pre-delay time. 0000 - No additional time is inserted 0001 - 1 SPI clock time is inserted 0010 - 2 SPI clock times are inserted 1111 - 15 SPI clock times are inserted

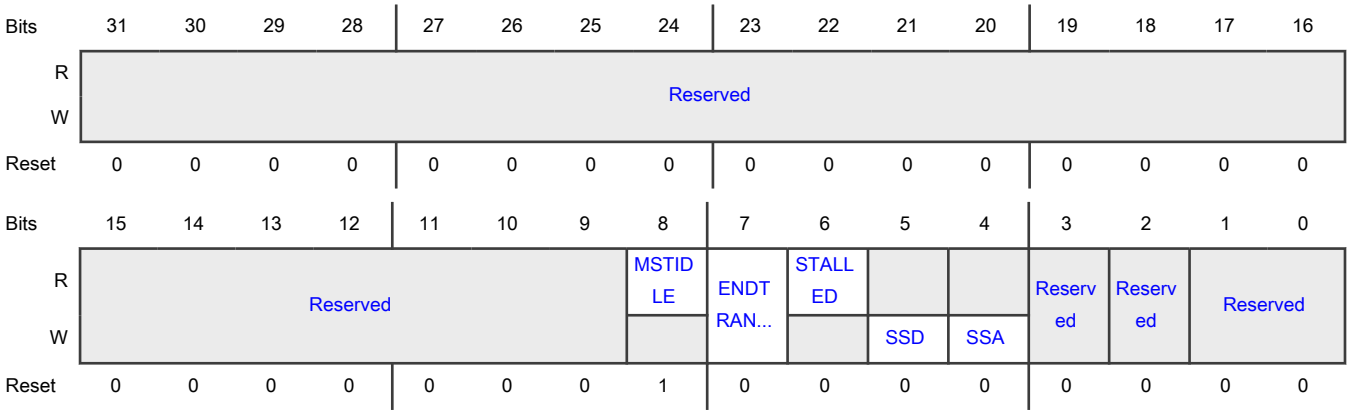
41.4.5.1.1.3 Status Register (STAT)

The STAT register provides SPI status flags for software to read, and a control bit for forcing an end of transfer. Flags other than read-only flags may be cleared by writing ones to corresponding bits in the STAT register.

Offset

Register	Offset
STAT	408h

Diagram



Fields

Field	Description
31-9 —	Reserved Read value is undefined; only zero should be written.
8 MSTIDLE	Master Idle Status Flag MSTIDLE bit is 1 whenever the SPI master function is fully idle. This means that the transmit holding register is empty and the transmitter is not in the process of sending data.
7 ENDTRANSFER	End Transfer Control Software can set ENDTRANSFER bit to force an end to the current transfer when the transmitter finishes any activity already in progress, as if the End of Transfer bit (FIFOWR[EOT]) had been set before to the last transmission. This capability is included to support cases where it is not known when transmit data is written and if it will be the end of a transfer. • The ENDTRANSFER bit is cleared when the transmitter becomes idle as the transfer ends. • Forcing an end of transfer using the ENDTRANSFER bit causes any specified FRAME_DELAY and TRANSFER_DELAY to be inserted.
6 STALLED	Stalled Status Flag Indicates whether the SPI is currently in a stall condition.
5 SSD	Slave Select Deassert The Slave Select Deassert flag is set whenever any asserted slave selects transition to deasserted, in both master and slave modes. This allows determining when the SPI transmit/receive functions become idle. SSD flag is cleared by software.
4 SSA	Slave Select Assert The Slave Select Assert flag is set whenever any slave select transitions from deasserted to asserted, in both master and slave modes. This allows determining when the SPI transmit/receive functions become busy, and allows waking up the device from reduced power modes when a slave mode access begins. SSA flag is cleared by software.
3 —	Reserved Read value is undefined; only zero should be written.
2 —	Reserved Read value is undefined; only zero should be written.
1-0 —	Reserved Read value is undefined; only zero should be written.

41.4.5.1.1.4 Interrupt Enable Register (INTENSET)

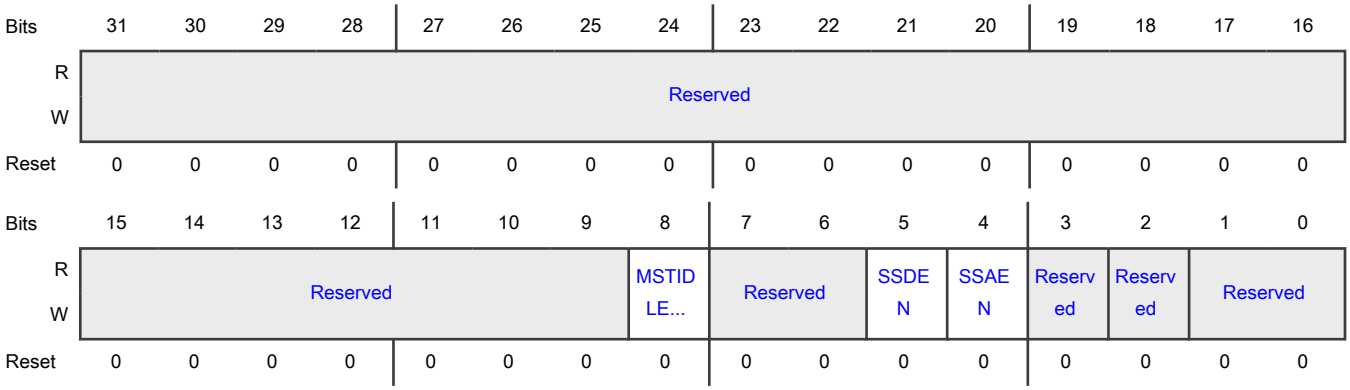
Contains the complete set of interrupt enables for the SPI. Use the INTENSET register to enable various SPI interrupt sources. Enable bits in INTENSET are mapped to locations that correspond to the flags in the STAT register. Writing a 1 to any implemented bit position causes that interrupt to be set.

Use the INTENCLR register to clear the interrupt enable bits in the INTENSET register.

Offset

Register	Offset
INTENSET	40Ch

Diagram



Fields

Field	Description
31-9 —	Reserved Read value is undefined; only zero should be written.
8 MSTIDLEEN	Master Idle Interrupt Enable 0 - No interrupt will be generated when the SPI master function is idle. 1 - An interrupt will be generated when the SPI master function is fully idle.
7-6 —	Reserved Read value is undefined; only zero should be written.
5 SSDEN	Slave Select Deassert Interrupt Enable Determines whether an interrupt occurs when the Slave Select is deasserted. 0 - Disabled. No interrupt will be generated when all asserted Slave Selects transition to deasserted. 1 - Enabled. An interrupt will be generated when all asserted Slave Selects transition to deasserted.
4 SSAEN	Slave Select Assert Interrupt Enable Determines whether an interrupt occurs when the Slave Select is asserted. 0 - Disabled. No interrupt will be generated when any Slave Select transitions from deasserted to asserted. 1 - Enabled. An interrupt will be generated when any Slave Select transitions from deasserted to asserted.
3 —	Reserved Read value is undefined; only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
2 —	Reserved Read value is undefined; only zero should be written.
1-0 —	Reserved Read value is undefined; only zero should be written.

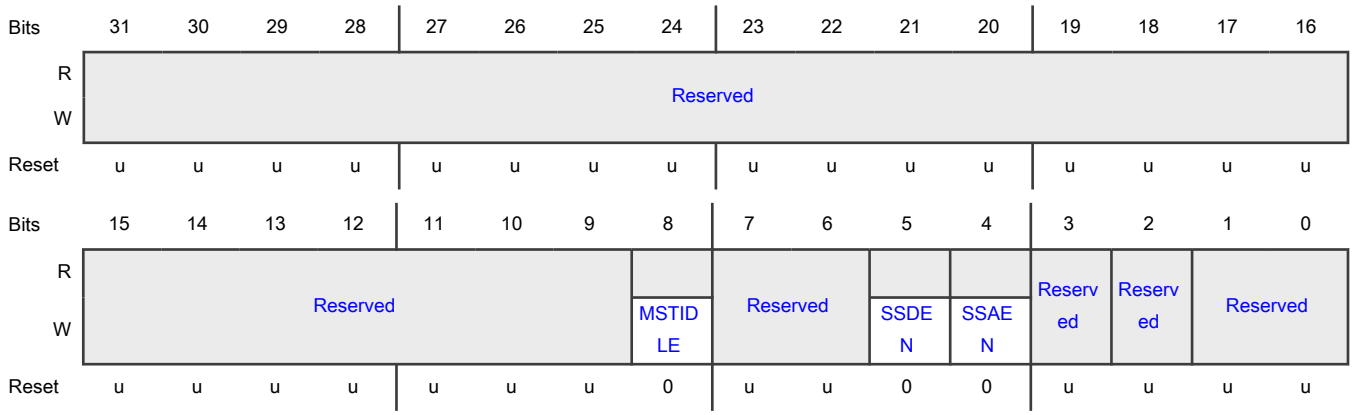
41.4.5.1.1.5 Interrupt Enable Clear Register (INTENCLR)

Use the INTENCLR register to clear interrupt enable bits in the INTENSET register. Writing a 1 to any implemented bit position causes the corresponding bit in the INTENSET register to be cleared.

Offset

Register	Offset
INTENCLR	410h

Diagram



Fields

Field	Description
31-9 —	Reserved Read value is undefined; only zero should be written.
8 MSTIDLE	Master Idle Interrupt Enable Writing 1 clears the Master Idle Interrupt Enable bit (INTENSET[MSTIDLEEN]). 0 - No effect 1 - Clear the Master Idle Interrupt Enable bit (INTENSET[MSTIDLE])

Table continues on the next page...

Table continued from the previous page...

Field	Description
7-6 —	Reserved Read value is undefined; only zero should be written.
5 SSDEN	Slave Select Deassert Interrupt Enable Writing 1 clears the Slave Select Deassert Interrupt Enable bit (INTENSET[SSDEN]). 0 - No effect 1 - Clear the Slave Select Deassert Interrupt Enable bit (INTENSET[SSDEN])
4 SSAEN	Slave Select Assert Interrupt Enable Writing 1 clears the Slave Select Assert Interrupt Enable bit (INTENSET[SSAEN]). 0 - No effect 1 - Clear the Slave Select Assert Interrupt Enable bit (INTENSET[SSAEN])
3 —	Reserved Read value is undefined; only zero should be written.
2 —	Reserved Read value is undefined; only zero should be written.
1-0 —	Reserved Read value is undefined; only zero should be written.

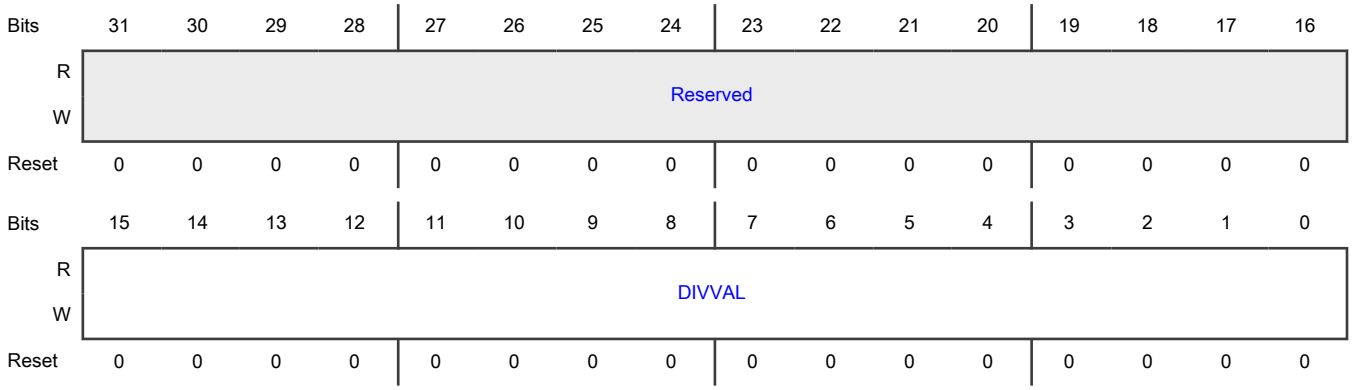
41.4.5.1.1.6 Clock Divider Register (DIV)

The DIV register specifies the clock divider value used to generate the SPI clock in SPI master mode.

Offset

Register	Offset
DIV	424h

Diagram



Fields

Field	Description
31-16 —	Reserved Read value is undefined; only zero should be written.
15-0 DIVVAL	Rate Divider Value Specifies how the Flexcomm clock (FCLK) is divided, to produce the SPI clock rate in master mode. DIVVAL is -1 encoded such that the value 0 results in FCLK/1, the value 1 results in FCLK/2, up to the maximum possible divide value of 0xFFFF, which results in FCLK/65536.

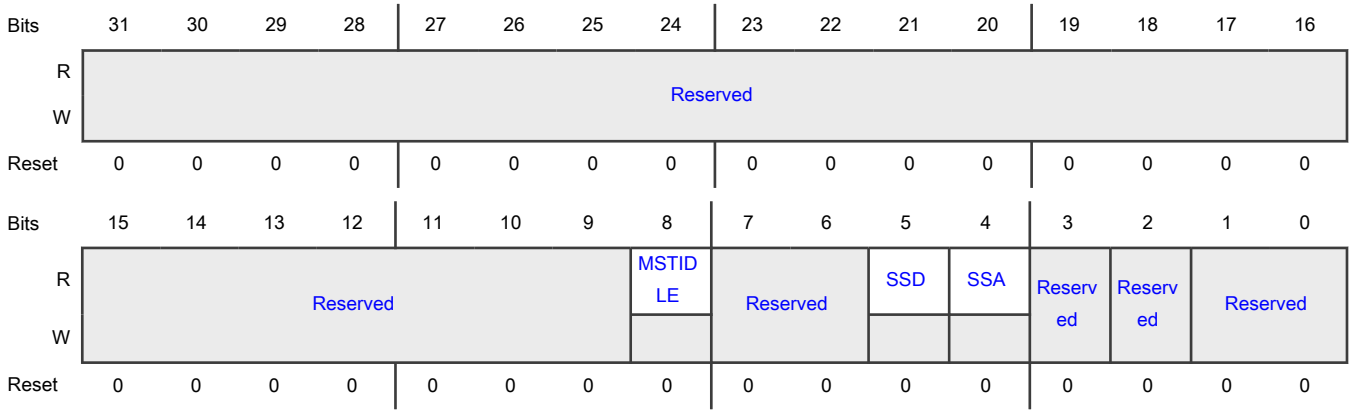
41.4.5.1.1.7 Interrupt Status Register (INTSTAT)

The read-only INTSTAT register provides a view of the interrupt flags that are currently enabled. This can simplify the software handling of interrupts.

Offset

Register	Offset
INTSTAT	428h

Diagram



Fields

Field	Description
31-9 —	Reserved Read value is undefined; only zero should be written.
8 MSTIDLE	Master Idle Status Flag Interrupt 0 - Disabled 1 - Enabled
7-6 —	Reserved Read value is undefined; only zero should be written.
5 SSD	Slave Select Deassert Interrupt 0 - Disabled 1 - Enabled
4 SSA	Slave Select Assert Interrupt 0 - Disabled 1 - Enabled
3 —	Reserved Read value is undefined; only zero should be written.
2 —	Reserved Read value is undefined; only zero should be written.
1-0 —	Reserved Read value is undefined; only zero should be written.

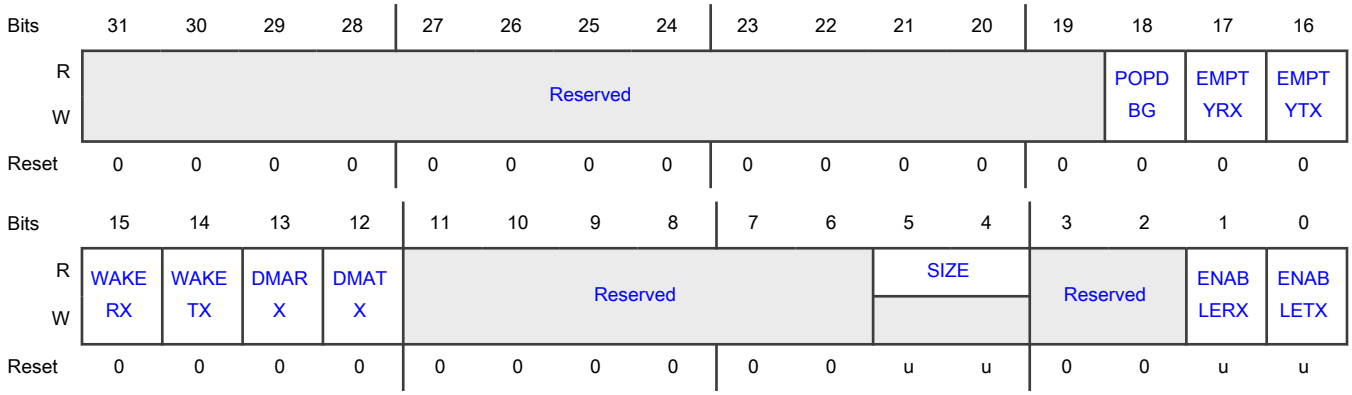
41.4.5.1.1.8 FIFO Configuration Register (FIFOCFG)

This register configures how the SPI uses the FIFO. Before configuring the FIFO, a peripheral function within the Flexcomm module must be selected.

Offset

Register	Offset
FIFOCFG	E00h

Diagram



Fields

Field	Description
31-19 —	Reserved Read value is undefined; only zero should be written.
18 POPDBG	Pop FIFO for Debug Reads 0 - Debug reads of the FIFO do not pop the FIFO 1 - A debug read will cause the FIFO to pop
17 EMPTYRX	Empty Command for the Receive FIFO When a 1 is written to EMPTYRX bit, the RX FIFO is emptied. 0 - No effect 1 - The RX FIFO is emptied
16 EMPTYTX	Empty Command for the Transmit FIFO When a 1 is written to EMPTYTX bit, the TX FIFO is emptied. 0 - No effect 1 - The TX FIFO is emptied
15 WAKERX	Wake-up for Receive FIFO Level WAKERX allows the device to be woken from reduced power modes (as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and then goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion. See the Hardware Wake-up control register. 0 - Only enabled interrupts will wake up the device form reduced power modes. 1 - A device wake-up for DMA will occur if the receive FIFO level reaches the value specified by FIFOTRIG[RXLVL], even when the RXLVL interrupt is not enabled.
14 WAKETX	Wake-up for Transmit FIFO Level

Table continues on the next page...

Table continued from the previous page...

Field	Description
	WAKETX allows the device to be woken from reduced power modes (as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and then goes back to sleep. The CPU will remain stopped until woken by another cause, such as DMA completion. See the Hardware Wake-up control register. 0 - Only enabled interrupts will wake up the device form reduced power modes 1 - A device wake-up for DMA will occur if the transmit FIFO level reaches the value specified by TXLVL in FIFOTRIG, even when the TXLVL interrupt is not enabled.
13 DMARX	DMA Configuration for Receive 0 - DMA is not used for the receive function. 1 - Issues a DMA request for the receive function if the FIFO is not empty. Generally, data interrupts would be disabled if DMA is enabled.
12 DMATX	DMA Configuration for Transmit 0 - DMA is not used for the transmit function 1 - Issues DMA request for the transmit function if the FIFO is not full. Generally, data interrupts would be disabled if DMA is enabled.
11-6 —	Reserved Read value is undefined; only zero should be written.
5-4 SIZE	FIFO Size Configuration This is a read-only field. 00 - FIFO is configured as 16 entries of 8 bits. 01 - FIFO is configured as 8 entries of 16 bits. 10 - Not used 11 - Not used
3-2 —	Reserved Read value is undefined; only zero should be written.
1 ENABLERX	Enable the Receive FIFO 0 - The receive FIFO is not enabled 1 - The receive FIFO is enabled
0 ENABLETX	Enable the Transmit FIFO 0 - The transmit FIFO is not enabled 1 - The transmit FIFO is enabled

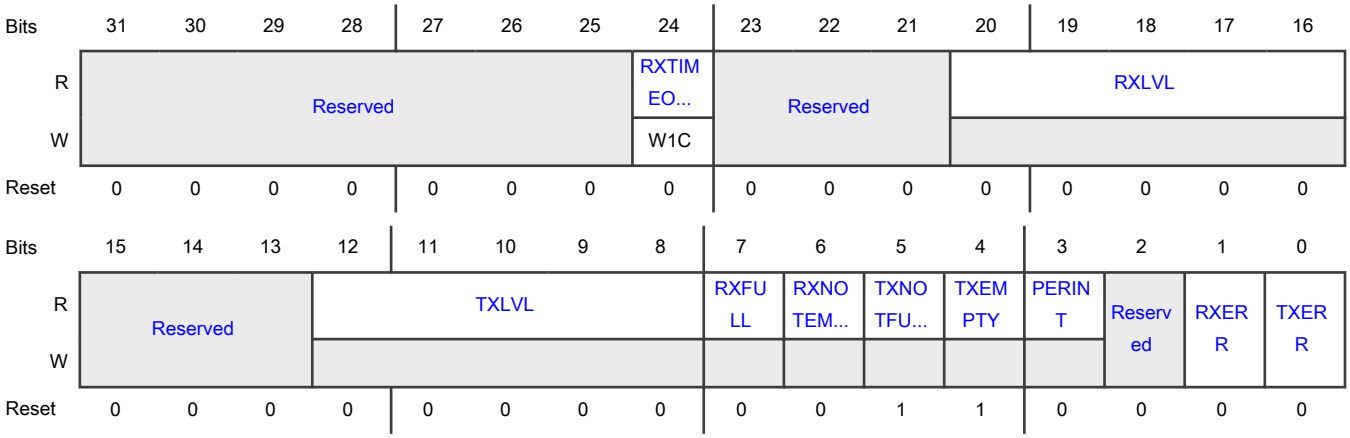
41.4.5.1.1.9 FIFO Status Register (FIFOSTAT)

Provides status information for the FIFO, and also indicates if a peripheral has issued an interrupt.

Offset

Register	Offset
FIFOSTAT	E04h

Diagram



Fields

Field	Description
31-25 —	Reserved Read value is undefined; only zero should be written.
24 RXTIMEOUT	Receive FIFO Timeout Writing 1 clears this bit and lowers the corresponding interrupt. 0 - RX FIFO on 1 - RX FIFO has timed out, based on the timeout configuration in the FIFORXTIMEOUTCFG register.
23-21 —	Reserved Read value is undefined; only zero should be written.
20-16 RXLVL	Receive FIFO Current Level RXLVL=0 means the RX FIFO is currently empty, and the RXFULL and RXNOTEMPTY flags will be 0. RXLVL="other values" tell how much data is actually in the RX FIFO at the point where the read occurs. If the RX FIFO is full, then the RXFULL and RXNOTEMPTY flags will be 1.
15-13 —	Reserved Read value is undefined; only zero should be written.
12-8 TXLVL	Transmit FIFO Current Level TXLVL=0 means the TX FIFO is currently empty, and the TXEMPTY and TXNOTFULL flags will be 1. TXLVL="other values" tell how much data is actually in the TX FIFO at the point where the read occurs. If the TX FIFO is full, then the TXEMPTY and TXNOTFULL flags will be 0.

Table continues on the next page...

Table continued from the previous page...

Field	Description
7 RXFULL	Receive FIFO is Full 0 - The receive FIFO is not full 1 - The receive FIFO is full. To prevent the peripheral from causing an overflow, data should be read out.
6 RXNOTEMPTY	Receive FIFO is Not Empty 0 - When 0, the receive FIFO is empty 1 - When 1, the receive FIFO is not empty, so data can be read
5 TXNOTFULL	Transmit FIFO is Not Full 0 - The transmit FIFO is full and another write would cause it to overflow 1 - The transmit FIFO is not full, so more data can be written
4 TXEMPTY	Transmit FIFO Empty 0 - The transmit FIFO is not empty 1 - The transmit FIFO is empty, although the peripheral may still be processing the last piece of data.
3 PERINT	Peripheral Interrupt 0 - The peripheral function has not asserted an interrupt 1 - Indicates that the peripheral function has asserted an interrupt. More information can be found by reading the peripheral's status register (STAT).
2 —	Reserved Read value is undefined; only zero should be written.
1 RXERR	RX FIFO Error Cleared by writing a 1 to itself. 0 - A receive FIFO overflow has not occurred 1 - A receive FIFO overflow has occurred, caused by software or DMA not emptying the FIFO fast enough
0 TXERR	TX FIFO Error Cleared by writing a 1 to itself. 0 - A transmit FIFO error has not occurred. 1 - A transmit FIFO error has occurred. This error could be an overflow caused by pushing data into a full FIFO, or by an underflow if the FIFO is empty when data is needed.

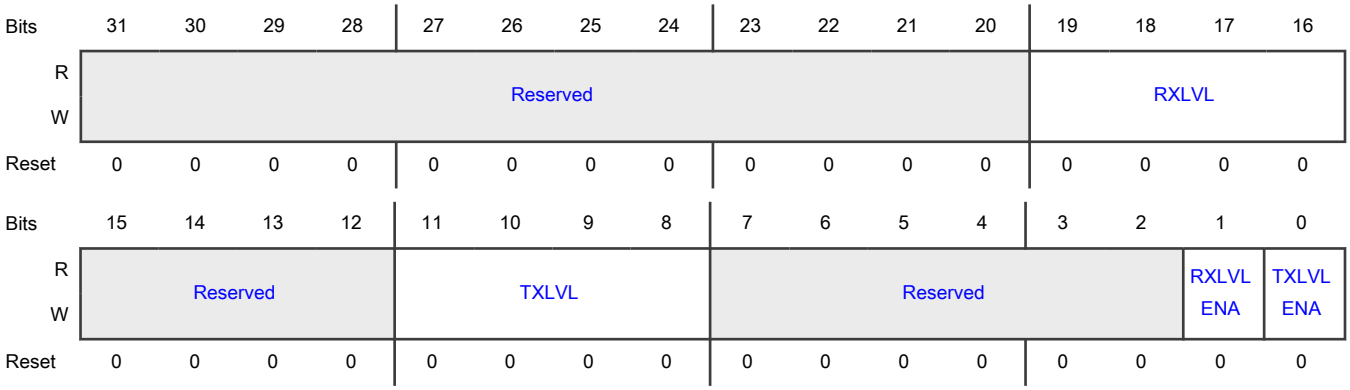
41.4.5.1.1.10 FIFO Trigger Register (FIFOTRIG)

Allows selecting when FIFO-level related interrupts occur, and includes FIFO trigger settings for interrupt and DMA requests.

Offset

Register	Offset
FIFOTRIG	E08h

Diagram



Fields

Field	Description
31-20 —	Reserved Read value is undefined; only zero should be written.
19-16 RXLVL	Receive FIFO Level Trigger Point The RX FIFO level is checked when a new piece of data is received. RXLVL field is used only when RXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode. See the Hardware Wake-up control register. **NOTE** FIFOCFG[WAKERX] allows the device to be woken from reduced power modes. 0000 - Trigger when the RX FIFO has received 1 entry (is no longer empty) 0001 - Trigger when the RX FIFO has received 2 entries 1111 - Trigger when the RX FIFO has received 16 entries (has become full)
15-12 —	Reserved Read value is undefined; only zero should be written.
11-8 TXLVL	Transmit FIFO Level Trigger Point TXLVL field is used only when TXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode. See the Hardware Wake-up control register. **NOTE** FIFOCFG[WAKETX] allows the device to be woken from reduced power modes.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0000 - Trigger when the TX FIFO becomes empty 0001 - Trigger when the TX FIFO level decreases to 1 entry 1111 - Trigger when the TX FIFO level decreases to 15 entries (is no longer full)
7-2 —	Reserved Read value is undefined; only zero should be written.
1 RXLVLENA	Receive FIFO Level Trigger Enable RXLVLENA trigger will become an interrupt if enabled in FIFOINTENSET, or will become a DMA trigger if DMARX in FIFOCFG is set. 0 - Receive FIFO level does not generate a FIFO level trigger 1 - An trigger will be generated if the receive FIFO level reaches the value specified by the FIFOTRIG[RXLVL] field.
0 TXLVLENA	Transmit FIFO Level Trigger Enable TXLVLENA trigger will become an interrupt if enabled in FIFOINTENSET, or will become a DMA trigger if DMATX in FIFOCFG is set. 0 - Transmit FIFO level does not generate a FIFO level trigger 1 - An trigger will be generated if the transmit FIFO level reaches the value specified by the FIFOTRIG[TXLVL] field.

41.4.5.1.1.11 FIFO Interrupt Enable Register (FIFOINTENSET)

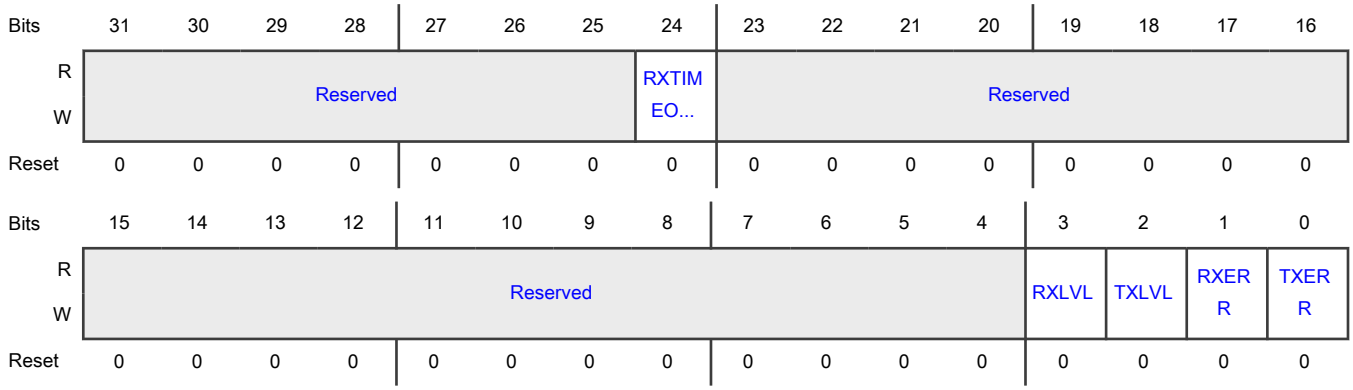
FIFOINTENSET includes the complete set of FIFO interrupt enables. Use the FIFO Interrupt Enable Register to enable various interrupt sources. Writing ones to implemented bits in FIFOINTENSET register forces those bits to be set.

Use the FIFOINTENCLR register to clear the interrupt enable bits in this register.

Offset

Register	Offset
FIFOINTENSET	E10h

Diagram



Fields

Field	Description
31-25 —	Reserved Read value is undefined; only zero should be written.
24 RXTIMEOUT	Receive Timeout The interrupt cause can be cleared by writing 1 to the FIFOSTAT[RXTIMEOUT] bit. 0 - No RX interrupt will be generated. 1 - Asserts RX interrupt if RX FIFO Timeout event occurs.
23-4 —	Reserved Read value is undefined; only zero should be written.
3 RXLVL	Receive FIFO Level Interrupt Enable Determines whether an interrupt occurs when a the receive FIFO reaches the level specified by the FIFOTRIG[TXLVL] field 0 - No interrupt will be generated based on the RX FIFO level 1 - If FIFOTRIG[RXLVLENA]=1, then an interrupt will be generated when the RX FIFO level increases to the level specified by FIFOTRIG[RXLVL]
2 TXLVL	Transmit FIFO Level Interrupt Enable Determines whether an interrupt occurs when a the transmit FIFO reaches the level specified by the FIFOTRIG[TXLVL] field 0 - No interrupt will be generated based on the TX FIFO level 1 - If FIFOTRIG[TXLVLENA]=1, then an interrupt will be generated when the TX FIFO level decreases to the level specified by FIFOTRIG[TXLVL]
1 RXERR	Receive Error Interrupt Enable Determines whether an interrupt occurs when a receive error occurs, based on the FIFOSTAT[RXERR] flag

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - No interrupt will be generated for a receive error 1 - An interrupt will be generated when a receive error occurs
0 TXERR	TX Error Interrupt Enable Determines whether an interrupt occurs after a transmit error occurs, based on the FIFOSTAT[TXERR flag] 0 - No interrupt will be generated for a transmit error 1 - An interrupt will be generated when a transmit error occurs

41.4.5.1.1.12 FIFO Interrupt Enable Clear Register (FIFOINTENCLR)

Use the FIFOINTENCLR register to clear interrupt enable bits in FIFOINTENSET. Writing 1 clears the corresponding bits in the FIFOINTENSET register.

Offset

Register	Offset
FIFOINTENCLR	E14h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved							RXTIMEO...	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 W	Reserved												RXLVL	TXLVL	RXERR	TXERR
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-25 —	Reserved Read value is undefined; only zero should be written.
24 RXTIMEOUT	Receive Timeout Writing 1 will clear this bit; it is described in FIFOINTSET. The cause can be cleared by writing 1 to the FIFOSTAT[RXTIMEOUT].

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - No effect 1 - Clear the interrupt
23-4 —	Reserved Read value is undefined; only zero should be written.
3 RXLVL	Receive FIFO Level Interrupt Enable Writing 1 clears FIFOINTENSET[RXLVL] 0 - No effect 1 - Clear the Receive FIFO Level Interrupt Enable bit FIFOINTENSET[RXLVL]
2 TXLVL	Transmit FIFO Level Interrupt Enable Writing 1 clears FIFOINTENSET[TXLVL] 0 - No effect 1 - Clear the Transmit FIFO Level Interrupt Enable bit FIFOINTENSET[TXLVL]
1 RXERR	Receive Error Interrupt Enable Writing 1 clears FIFOINTENSET[RXERR] 0 - No effect 1 - Clear the Receive Error Interrupt Enable bit FIFOINTENSET[RXERR]
0 TXERR	TX Error Interrupt Enable Writing 1 clears FIFOINTENSET[TXERR] 0 - No effect 1 - Clear the TX Error Interrupt Enable bit FIFOINTENSET[TXERR]

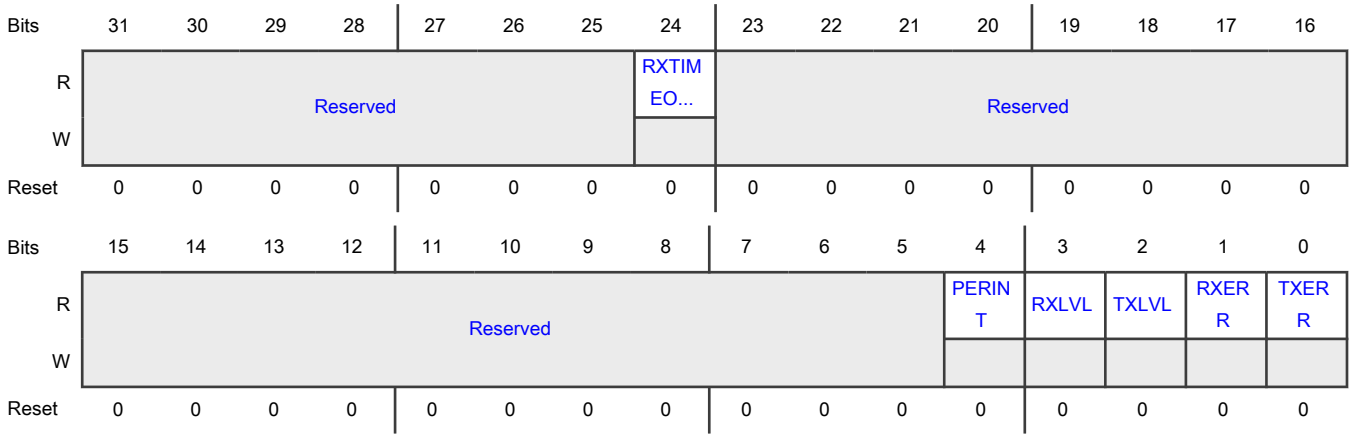
41.4.5.1.1.13 FIFO Interrupt Status Register (FIFOINTSTAT)

The read-only FIFOINTSTAT register provides a view of the interrupt flags that are both pending and currently enabled. This can simplify the software handling of interrupts.

Offset

Register	Offset
FIFOINTSTAT	E18h

Diagram



Fields

Field	Description
31-25 —	Reserved Read value is undefined; only zero should be written.
24 RXTIMEOUT	Receive Timeout Status 1 if both FIFOSTAT[RXTIMEOUT] and FIFOINTSET[RXTIMEOUT] = 1. Note that FIFOSTAT[RXTIMEOUT] may be cleared by writing 1 to it. 0 - Not pending 1 - Pending
23-5 —	Reserved Read value is undefined; only zero should be written.
4 PERINT	Peripheral Interrupt Status 0 - Not pending 1 - Pending
3 RXLVL	Receive FIFO Level Interrupt Status 0 - Not pending 1 - Pending
2 TXLVL	Transmit FIFO Level Interrupt Status 0 - Not pending 1 - Pending
1 RXERR	RX FIFO Error Interrupt Status 0 - Not pending 1 - Pending

Table continues on the next page...

Table continued from the previous page...

Field	Description
0 TXERR	TX FIFO Error Interrupt Status 0 - Not pending 1 - Pending

41.4.5.1.1.14 FIFO Write Data Register (FIFOWR)

The FIFOWR register is used to write values to be transmitted to the FIFO. FIFOWR is a write only register; do not read-modify-write the FIFOWR register.

FIFOWR allows you to alter some SPI controls while sending new data. For example, this can allow a series of SPI transactions involving multiple slaves to be stored in a DMA buffer and sent automatically. These added fields are described in bits 16 - 27.

Each FIFO entry holds data and associated control bits. Before data and control bits are pushed into the FIFO, the control bit settings can be modified.

- Halfword writes to just the control bits (offset 0xE22) doesn't push anything into the FIFO.
- A zero written to the upper halfword will not modify the control settings.
- Non-zero writes to it will modify all the control bits.

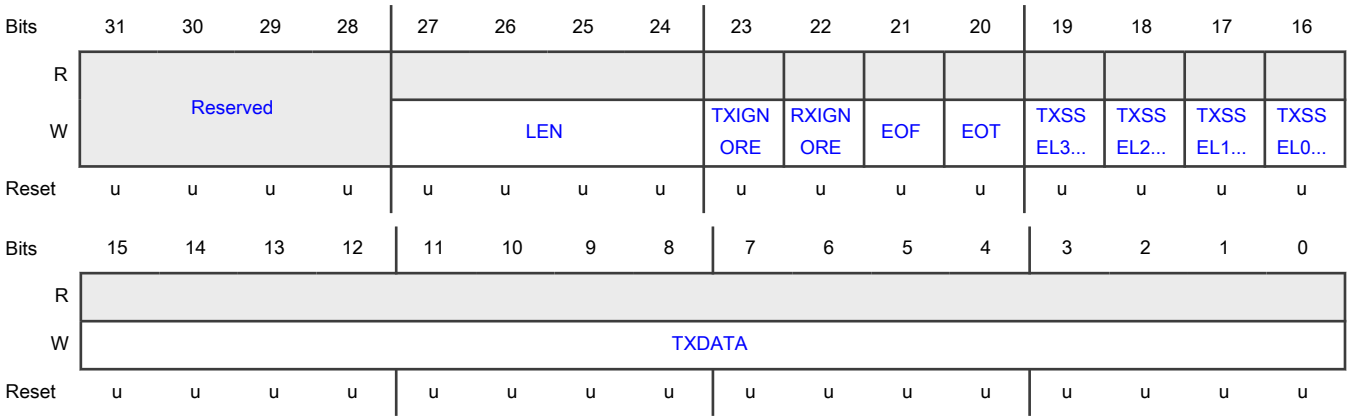
Byte, halfword or word writes to FIFOWR will push the data and control bits into the FIFO.

- Word writes with the upper halfword of zero, byte writes or halfword writes to FIFOWR will push the data and the current control bits, into the FIFO.
- Word writes with a non-zero upper halfword will modify the control bits before pushing them (the control bits) onto the stack.

Offset

Register	Offset
FIFOWR	E20h

Diagram



Fields

Field	Description
31-28 —	Reserved Read value is undefined; only zero should be written.
27-24 LEN	Data Length Specifies the data length in bits, from 4 to 16 bits. Note that transfer lengths greater than 16 bits are supported by implementing multiple sequential transmits. Except for the reserved values (0x0 - 0x2), for LEN=n, the data length is n+1 bits. 0000 - Reserved 0001 - Reserved 0010 - Reserved 0011 - Data transfer is 4 bits in length 0100 - Data transfer is 5 bits in length 1111 - Data transfer is 16 bits in length
23 TXIGNORE	Transmit Ignore TXIGNORE allows data to be received using the SPI in slave mode without the need to transmit data that is not needed. **NOTE** TXIGNORE can only be set by writing to the upper 16 bits of FIFOWR, that is, a halfword write to offset 0xE22. 0 - Write transmit data. Write transmit data. Transmit data must be written for each data exchange between master and slave. In slave mode, an underrun error occurs if transmit data is not provided before needed in a data frame. 1 - Ignore transmit data. Ignore transmit data. Data can be received without transmitting data (after FIFOWR has been initialized to set TXIGNORE). No transmitter flags are generated. When configured with TXIGNORE = 1, the slave sets the data to always 0.
22 RXIGNORE	Receive Ignore RXIGNORE allows data to be transmitted using the SPI, without the need to read unneeded data from the receiver. Setting RXIGNORE bit simplifies the transmit process and can be used with DMA. 0 - Read received data. Received data must be read, to allow transmission to proceed. SPI transmit will halt when the receive data FIFO is full. In slave mode, an overrun error will occur if received data is not read before new data is received. 1 - Ignore received data. Received data is ignored, allowing transmission without reading unneeded received data. No receiver flags are generated.
21 EOF	End of Frame Between frames, a delay may be inserted, as defined by the Frame_delay value in the DLY register. The end of a frame may not be particularly meaningful if the Frame_delay value = 0. This EOF control can be used as part of the support for frame lengths greater than 16 bits.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - Data not EOF. This piece of data transmitted is not treated as the end of a frame. 1 - Data EOF. This piece of data is treated as the end of a frame, causing the Frame_delay time to be inserted before subsequent data is transmitted.
20 EOT	End of Transfer The asserted SSEL will be deasserted at the end of a transfer, and remain to at least the time specified by the Transfer_delay value in the DLY register. 0 - SSEL is not deasserted. This piece of data is not treated as the end of a transfer. SSEL will not be deasserted at the end of this data. 1 - SSEL is deasserted. This piece of data is treated as the end of a transfer. SSEL will be deasserted at the end of this piece of data.
19 TXSSEL3_N	Transmit Slave Select 3 TXSSEL3_N field asserts SSEL3 in master mode. The output on the pin is active LOW by default. 0 - SSEL3 is asserted 1 - SSEL3 is not asserted
18 TXSSEL2_N	Transmit Slave Select 2 TXSSEL2_N field asserts SSEL2 in master mode. The output on the pin is active LOW by default. 0 - SSEL2 is asserted 1 - SSEL2 is not asserted
17 TXSSEL1_N	Transmit Slave Select 1 TXSSEL1_N field asserts SSEL1 in master mode. The output on the pin is active LOW by default. 0 - SSEL1 is asserted 1 - SSEL1 is not asserted
16 TXSSEL0_N	Transmit Slave Select 0 TXSSEL0_N field asserts SSEL0 in master mode. The output on the pin is active LOW by default. 0 - SSEL0 is asserted 1 - SSEL0 is not asserted
15-0 TXDATA	Transmit Data to the FIFO

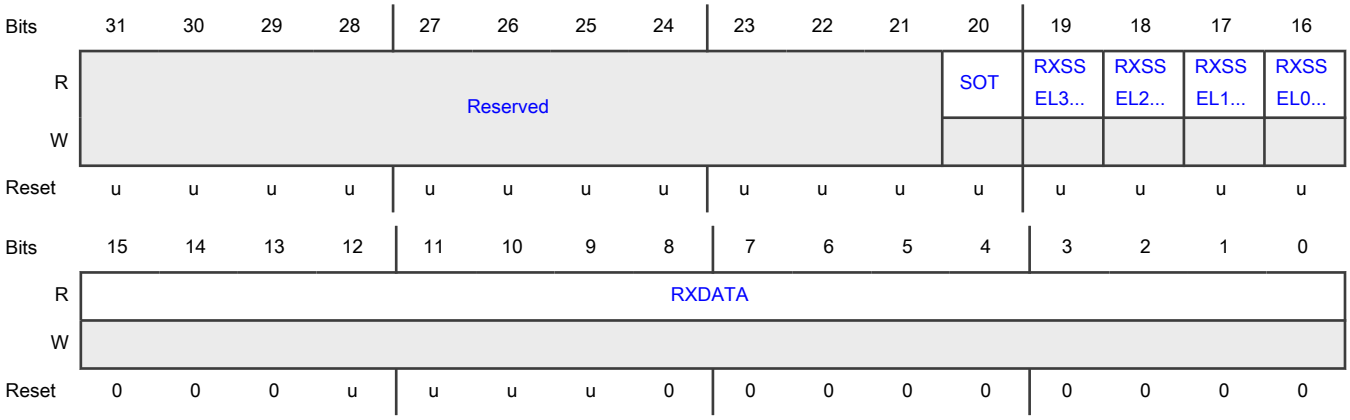
41.4.5.1.1.15 FIFO Read Data Register (FIFORD)

Use the FIFORD register to read values that have been received by the FIFO.

Offset

Register	Offset
FIFORD	E30h

Diagram



Fields

Field	Description
31-21 —	Reserved The value read from a reserved bit is not defined.
20 SOT	Start of Transfer Flag 0 - This is not the 1st data after the SSELs went from deasserted to asserted 1 - This is the 1st data after the SSELs went from deasserted to asserted (i.e., any previous transfer has ended). This information can be used to identify the 1st piece of data in cases where the transfer length is greater than 16 bits.
19 RXSSEL3_N	Slave Select 3 for Receive RXSSEL3_N field allows the state of the SSEL3 pin to be saved, along with the data received at that pin. The value indicates the SSEL3 pin for both master and slave operations. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG. 0 - Slave Select 3 is active 1 - Slave Select 3 is not active
18 RXSSEL2_N	Slave Select 2 for Receive RXSSEL2_N field allows the state of the SSEL2 pin to be saved, along with the data received at that pin. The value indicates the SSEL2 pin for both master and slave operations. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG. 0 - Slave Select 2 is active 1 - Slave Select 2 is not active

Table continues on the next page...

Table continued from the previous page...

Field	Description
17 RXSSEL1_N	Slave Select 1 for Receive RXSSEL1_N field allows the state of the SSEL1 pin to be saved, along with the data received at that pin. The value indicates the SSEL1 pin for both master and slave operations. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG. 0 - Slave Select 1 is active 1 - Slave Select 1 is not active
16 RXSSEL0_N	Slave Select 0 for Receive RXSSEL0_N field allows the state of the SSEL0 pin to be saved, along with the data received at that pin. The value indicates the SSEL0 pin for both master and slave operations. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG. 0 - Slave Select 0 is active 1 - Slave Select 0 is not active
15-0 RXDATA	Received Data from the FIFO

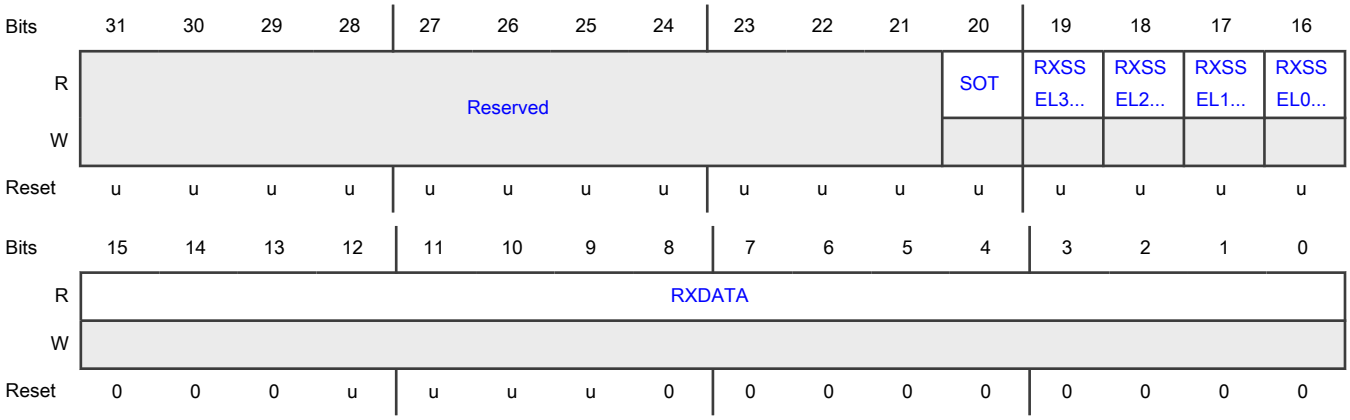
41.4.5.1.1.16 FIFO Data Read with no FIFO Pop Register (FIFORDNOPOP)

The FIFORDNOPOP register acts in exactly the same way as the FIFORD register, except that FIFORDNOPOP supplies data from the top of the FIFO without popping the FIFO (i.e., thereby leaving the FIFO state unchanged). This can be used to allow the system software to observe incoming data without interfering with the peripheral driver.

Offset

Register	Offset
FIFORDNOPOP	E40h

Diagram



Fields

Field	Description
31-21 —	Reserved The value read from a reserved bit is not defined.
20 SOT	Start of Transfer Flag 0 - Not active 1 - Active
19 RXSSEL3_N	Slave Select 3 for Receive 0 - Not selected 1 - Selected
18 RXSSEL2_N	Slave Select 2 for Receive 0 - Not selected 1 - Selected
17 RXSSEL1_N	Slave Select 1 for Receive 0 - Not selected 1 - Selected
16 RXSSEL0_N	Slave Select 0 for Receive 0 - Not selected 1 - Selected
15-0 RXDATA	Received Data from the FIFO

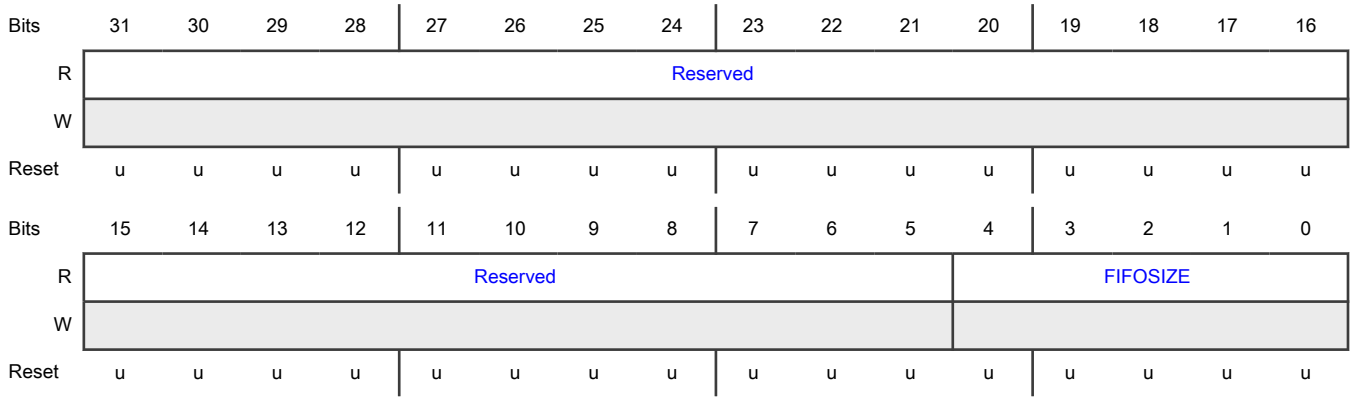
41.4.5.1.1.17 FIFO Size Register (FIFOSIZE)

Provides the FIFO size.

Offset

Register	Offset
FIFOSIZE	E48h

Diagram



Fields

Field	Description
31-5 —	Reserved
4-0 FIFOSIZE	FIFO Size Provides the size of the FIFO for software. FIFOSIZE is 8 entries for this chip. This field is read only.

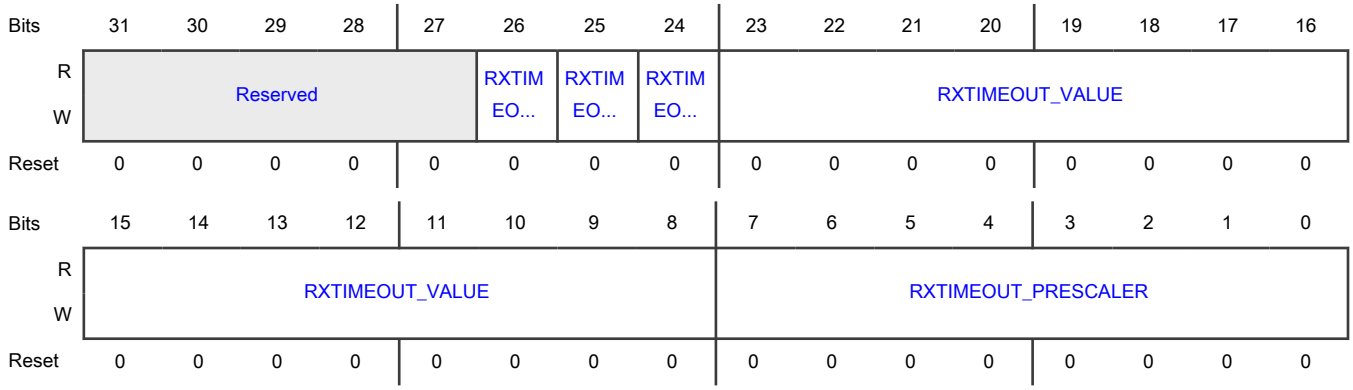
41.4.5.1.1.18 FIFO Receive Timeout Configuration (FIFORXTIMEOUTCFG)

When RXTIMEOUT_EN is enabled, the timeout counter is counting based on the configuration in this FIFORXTIMEOUTCFG register. When the counter reaches FIFORXTIMEOUTCFG[RXTIMEOUT_VALUE] + 1, FIFOSTAT[RXTIMEOUT] is set. The RX interrupt will be asserted if FIFOINTSET[RXTIMEOUT] bit is also set. FIFOSTAT[RXTIMEOUT] can be cleared by writing 1 to it.

Offset

Register	Offset
FIFORXTIMEOUTCFG	E4Ch

Diagram



Fields

Field	Description
31-27 —	Reserved
26 RXTIMEOUT_COE	Receive Timeout Continue On Empty RXTIMEOUT_COE allows the timeout to be used to flag idle peripherals, and could potentially be used to indicate the end of a transmission of indeterminate length. 0 - RX FIFO timeout counter is reset when the RX FIFO becomes empty. 1 - RX FIFO timeout counter is not reset when the RX FIFO becomes empty.
25 RXTIMEOUT_COW	Receive Timeout Continue On Write RXTIMEOUT_COW allows the timeout to be applied to accumulated data, perhaps related to the FIFO threshold. 0 - RX FIFO timeout counter is reset every time data is transferred from the peripheral into the RX FIFO. 1 - RX FIFO timeout counter is not reset every time data is transferred from the peripheral into the RX FIFO.
24 RXTIMEOUT_EN	Receive Timeout Enable 0 - Disable RX FIFO timeout 1 - Enable RX FIFO timeout
23-8 RXTIMEOUT_VALUE	Receive Timeout Value When RXTIMEOUT_VALUE is enabled and timeout counter reaches RXTIMEOUT_VALUE + 1, FIFOSTAT[RXTIMEOUT] is set.
7-0 RXTIMEOUT_PRESCALER	Receive Timeout Counter Clock Prescaler The clock frequency for FIFORXTIMEOUTCNT[RXTIMEOUT_CNT] is the AHB bus clock frequency divided by 16 * (RXTIMEOUT_PRESCALER + 1).

41.4.5.1.1.19 FIFO Receive Timeout Counter (FIFORXTIMEOUTCNT)

Timeout counter gets reset on one of the following conditions:

- FIFORXTIMEOUTCFG[RXTIMEOUT_EN] is 0
- FIFOSTAT[RXTIMEOUT] is written with 1
- Rx FIFO is empty and FIFORXTIMEOUTCFG[RXTIMEOUT_COE] is 0
- Rx FIFO is written and FIFORXTIMEOUTCFG[RXTIMEOUT_COW] is 0

Timeout counter is paused when FIFOSTAT[RXTIMEOUT] is set by hardware and is cleared when FIFOSTAT[RXTIMEOUT] is written with 1 by software.

Offset

Register	Offset
FIFORXTIMEOUTCNT	E50h

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	RXTIMEOUT_CNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-16 —	Reserved
15-0 RXTIMEOUT_CNT	Current RX FIFO timeout counter value

41.4.5.1.1.20 Peripheral Identification Register (ID)

Identifies the type and revision of the SPI module. A generic software driver can make use of this information to implement specific behavior, based on a module type or a specific revision of that module. This value appears in the shared Flexcomm peripheral ID register when the SPI is selected.

Offset

Register	Offset
ID	FFCh

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ID															
W																
Reset	1	1	1	0	0	0	0	0	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	MAJOR_REV				MINOR_REV				APERTURE							
W																
Reset	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-16 ID	Module identifier for the selected function
15-12 MAJOR_REV	Major revision of module implementation
11-8 MINOR_REV	Minor revision of module implementation
7-0 APERTURE	Aperture aperture size = (APERTURE + 1) x 4K APERTURE encoded as 0x00 means 4K aperture: aperture size = (0x00 + 1) x 4K

41.5 Inter-integrated Circuit (I2C)

41.5.1 Overview

The Inter-integrated Circuit (I2C) module enables multiple “slave” digital modules to communicate with one or more “master” modules, for short distance communications within a single device.

41.5.1.1 Block diagram

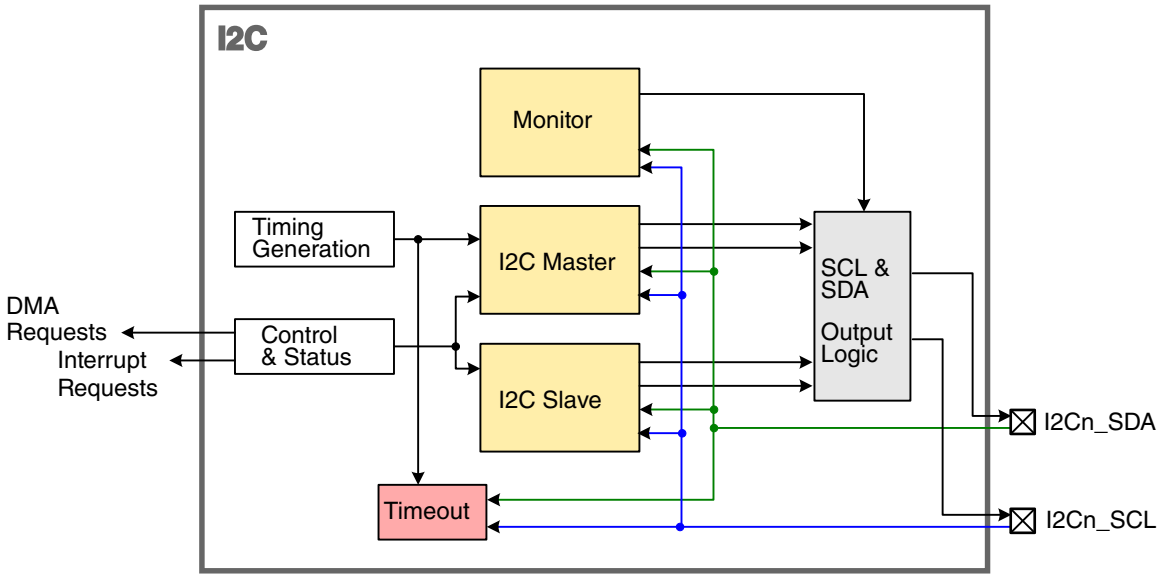


Figure 171. I2C block diagram

41.5.1.2 Features

I2C bus functions are available on the device as a selectable function in each Flexcomm Interface peripheral. The I2C module includes:

- Independent Master, Slave, and Monitor functions
- Bus speeds supported:
  - Standard mode, up to 100 kbits/s
  - Fast-mode, up to 400 kbits/s
  - Fast-mode Plus, up to 1 Mbits/s (on pins that include specific I2C support)
  - High speed mode, 3.4 Mbits/s as a Slave only (on pins that include specific I2C support)
- Supports both Multi-master and Multi-master with Slave functions
- Multiple I2C slave addresses supported in hardware
- To respond to multiple I2C-bus addresses, one slave address can be selectively qualified with a bit mask or an address range
- 10-bit addressing supported with software assist
- Supports System Management Bus (SMBus)
- Separate DMA requests for Master, Slave, and Monitor functions
- No chip clocks are required to receive and compare an address as a Slave, so this event (a Slave address match) can wake up the device from deep sleep mode
- Automatic modes optionally allow less software overhead for some use cases

41.5.2 Functional description

The following sections describe functional details of the I2C module.

41.5.2.1 Operating modes

This section describes all functional operation modes of the I2C module.

41.5.2.1.1 I2C transmit/receive in master mode

The Flexcomm interface is configured to be an I2C master. The master sends 8 bits to the slave and then receives 8 bits from the slave. The transmission of the address and data bits is controlled by the state of the MSTPENDING status bit. Whenever the status is Master pending, the master can read or write to the MSTDAT register and go to the next step of the transmission protocol (by writing to the MSTCTL register).

If specialized I2C pins are used, then the pins should be configured as required for the I2C bus mode that will be used (SM, FM, FM+, HS). If specialized I2C or standard pins are used, then the pins should be configured correctly.

To configure the I2C bit rate:

- Select a clock source for the Flexcomm interface that will provide the desired I2C bus rate, and divide the clock as needed.
- To further divide the source clock (if needed) using the CLKDIV register.
- Set the SCL high and low times to complete the bus rate setup.

Master write to slave:

- Configure the Flexcomm interface to be an I2C interface.
- Configure the I2C as a master: set the CFG[MSTEN] bit to 1.
- Write data to the slave:

1. Write the slave address with the RW bit set to 0 to the Master data register (MSTDAT).
2. Start the transmission by setting the MSTCTL[MSTSTART] bit to 1. The following happens:
   - The pending status is cleared and the I2C bus is busy.
   - The I2C master sends the start bit and address with the RW bit to the slave.
3. Wait for the pending status to be set (MSTPENDING = 1), by polling the STAT register.
4. Write 8 bits of data to the MSTDAT register.
5. Continue with the transmission of data by setting the MSTCTL[MSTCONT] bit to 1. The following happens:
   - The pending status is cleared and the I2C bus is busy.
   - The I2C master sends the data bits to the slave address.

- Wait for the pending status to be set (MSTPENDING = 1), by polling the STAT register.
- Stop the transmission, by setting the MSTCTL[MSTSTOP] bit to 1.

Master write to slave: code example

```
//Master write 1 byte to slave. Address 0x23, Data 0xdd. Polling mode.
I2C->CFG = I2C_CFG_MSTEN;
while(!(I2C->STAT  & I2C_STAT_MSTPENDING));
if((I2C->STAT  & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
I2C->MSTDAT = (0x23 << 1) | 0;      // address and 0 for RWn bit
I2C->MSTCTL = I2C_MSTCTL_MSTSTART;         // send start
  while(!(I2C->STAT  & I2C_STAT_MSTPENDING));
  if((I2C->STAT  & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_TX) abort();
  I2C->MSTDAT = 0xdd;                     // send data
  I2C->MSTCTL = I2C_MSTCTL_MSTCONTINUE;   // continue transaction
  while(!(I2C->STAT  & I2C_STAT_MSTPENDING));
  if((I2C->STAT  & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_TX) abort();
  I2C->MSTCTL = I2C_MSTCTL_MSTSTOP;      // send stop
  while(!(I2C->STAT  & I2C_STAT_MSTPENDING));
  if((I2C->STAT  & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
```

Master read from slave:

- Configure the Flexcomm interface to be an I2C interface.
- Configure the I2C as a master: set the CFG[MSTEN] bit to 1.
- Read data from the slave:
  1. Write the slave address (with the RW bit set to 1) to the Master data register (MSTDAT).
  2. Start the transmission, by setting the MSTCTL[MSTSTART] bit to 1. The following events happen next:
     - The pending status is cleared and the I2C bus is busy.
     - The I2C master sends the start bit and address with the RW bit to the slave.
     - The slave sends 8 bit of data.
  3. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
  4. Read 8 bits of data from the MSTDAT register.
  5. Stop the transmission by setting the MSTCTL[MSTSTOP] bit to 1.

Master read from slave: code example

```
    // Master read 1 byte from slave. Address 0x23. Polling mode. No error checking.
uint8_t data;
I2C->CFG = I2C_CFG_MSTEN;

while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
I2C->MSTDAT = (0x23 << 1) | 1;      // address and 1 for RWn bit
I2C->MSTCTL = I2C_MSTCTL_MSTSTART;         // send start
  while(!(I2C->STAT & I2C_STAT_MSTPENDING));
  if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_RX) abort();
  data = I2C->MSTDAT;                      // read data
  if(data != 0xdd) abort();
  I2C->MSTCTL = I2C_MSTCTL_MSTSTOP;      // send stop
  while(!(I2C->STAT & I2C_STAT_MSTPENDING));
  if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
```

41.5.2.1.2 I_2C receive/transmit in slave mode

The Flexcomm interface is configured to be an I2C slave. The slave receives 8 bits from the master and then sends 8 bits to the master. The transmission of the address and data bits is controlled by the state of the SLVPENDING status bit. Whenever the status is Slave pending, the slave can acknowledge (“ack”) or send or receive an address and data. The received data or the data to be sent to the master are available in the SLVDAT register. After sending and receiving data, the application should continue to the next step of the transmission protocol by writing to the SLVCTL register.

The SCL and SDA functions must be enabled on its pins. The pins should be configured as required for the I2C bus mode that will be used (SM, FM, FM+, HS).

Slave read from master:

- Configure the Flexcomm interface to be an I2C interface.
- Configure the I2C as a slave with address x:
- — Set the CFG[SLVEN] bit to 1.
  — Write the slave address x to the address 0 match register.
- Read data from the master:
  1. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
  2. Acknowledge (“ack”) the address by setting SLVCTL[SLVCONTINUE] = 1.
  3. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
  4. Read 8 bits of data from the SLVDAT register.
  5. Acknowledge (“ack”) the data by setting SLVCTL[SLVCONTINUE] = 1.

Slave read from master: code example

```
//Slave read 1 byte from master. Address 0x23. Polling mode.
uint8_t data;
I2C->SLVADR0 = 0x23 << 1;        // put address in address 0 register
I2C->CFG = I2C_CFG_SLVEN;
  while(!(I2C->STAT & I2C_STAT_SLVPENDING));
  if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();
  I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE;        // ack address
```

```
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_RX) abort();
data = I2C->SLVDAT;                          // read data
if(data != 0xdd) abort();
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE;        // ack data
```

Slave write to master:

- Configure the Flexcomm interface to be an I2C interface.
- Configure the I2C as a slave with address x:
  - Set the CFG[SLVEN] bit to 1.
  - Write the slave address x to the address 0 match register.
- Write data to the master:
  1. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
  2. ACK the address by setting SLVCTL[SLVCONTINUE] = 1.
  3. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
  4. Write 8 bits of data to SLVDAT register.
  5. Continue the transaction by setting SLVCTL[SLVCONTINUE] = 1.

Slave write to master: code example

```
//Slave write 1 byte to master. Address 0x23, Data 0xdd. Polling mode.
I2C->SLVADR0 = 0x23 << 1;           // put address in address 0 register
I2C->CFG = I2C_CFG_SLVEN;
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE;          // ack address
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_TX) abort();
I2C->SLVDAT = 0xdd;                           // write data
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE;       // continue transaction
```

41.5.2.1.3 Configure the I2C for wake-up

In sleep mode, any activity on the I2C bus that triggers an I2C interrupt can wake up the device, if the interrupt is enabled in the INTENSET register and in the interrupt controller. As long as the Flexcomm interface clock remains active in sleep mode, the I2C can wake up the part independently of whether the I2C interface is configured in master or slave mode.

In deep-sleep mode, the I2C clock and all peripheral clocks are turned off. However, if the I2C is configured in slave mode and an external master on the I2C bus provides the clock signal, then the I2C interface can create an interrupt asynchronously. This asynchronous interrupt can then wake up the core, if that interrupt is enabled in the interrupt controller and in the I2C interface INTENCLR register.

Wake-up from sleep mode:

- Enable the I2C interrupt in the interrupt controller.
- Enable the I2C wake-up event in the INTENSET register. Wake-up on any enabled interrupts is supported. Examples of wake-up events:
  - Master pending
  - Change to idle state

- Start/stop error
- Slave pending
- Address match (in slave mode)
- Data available/ready

Wake-up from deep-sleep mode:

- Enable the I2C interrupt .
- Enable the I2C interrupt to create the interrupt signal asynchronously while the core and the peripheral are not clocked.
- Configure the I2C in slave mode.
- Enable the I2C interrupt in the INTENCLR register, which configures the interrupt as a wake-up event. Examples of events are:
  - Slave de-select
  - Slave pending (wait for read, write, or ACK)
  - Address match
  - Data available/ready for the Monitor function

41.5.2.2 Operations

This section describes the I2C module's operations.

41.5.2.2.1 Bus timing

AHB bus access limitations

- The internal bus interface to the I2C registers contained in the Flexcomm interface supports only word writes.
- Byte and halfword writes are not supported in the I2C function.

Bus rates and timing considerations

Due to the nature of the I2C bus, it is generally not possible to guarantee a specific clock rate on the SCL pin. On the I2C bus, the clock can be stretched by any slave device, extended by software overhead time, and using other methods. In a multi-master system, the master that provides the shortest SCL high time will cause that time to appear on SCL, as long as that master is participating in I2C traffic (i.e., when it is the only master on the bus, or during arbitration between masters).

In addition, I2C implementations generally base subsequent actions on what actually happens on the bus lines. For example, a bus master allows SCL to go high. It (the bus master) then monitors the line to make sure that the line actually did go high (this would be required in a multi-master system). This results in a small delay before the next action on the bus, caused by the rise time of the open drain bus line. Rate calculations give a base frequency that represents the fastest that the I2C bus can operate if nothing slows it (the I2C bus) down.

Rate calculations: Master timing

SCL high time (in Flexcomm interface function clocks) **=** I2C clock divider x SCL high multiplier

SCL low time (in Flexcomm interface function clocks) **=** I2C clock divider x SCL low multiplier

Nominal SCL rate = Flexcomm Interface function clock rate / (SCL high time + SCL low time)

NOTE

- DIVVAL must be ≥ 1.
- For 400 kHz clock rate, the clock frequency after the I2C divider (divval) must be ≤ 2 MHz. The next table shows the recommended settings for 400 kHz clock rate.

Table 344. Settings for 400 kHz clock rate

Input clock to I2C (MHz)	DIVVAL for CLKDIV register	MSTSCLHIGH for MSTTIME register	MSTSCLLOW for MSTTIME register
96	59	0	0
48	29	0	0
48	23	0	1
30	14	0	1
24	14	0	0
24	11	0	1
12	5	0	1

Rate calculations: Slave timing

Most aspects of slave operation are controlled by the SCL signal received from the I2C bus master. However, if the slave stretches SCL to allow more time for a software response, it (the slave) must provide sufficient data setup time to the master before releasing the stretched clock. This is accomplished by inserting 1 clock time of CLKDIV at that point. If CLKDIV is already configured for master operation, then that is sufficient. If only the slave function is used, then CLKDIV should be configured such that 1 clock time is greater than the $t_{SU;DAT}$ value noted in the I2C bus specification for the I2C mode that is being used.

41.5.2.2.2 DMA

DMA with the I2C is performed only for data transfers; DMA cannot handle control of the I2C bus. After DMA is transferring data, I2C acknowledges are handled implicitly. While DMA is transferring data, no CPU intervention is required.

Data transfers can be handled by DMA:

- for Master mode: after an address is sent and acknowledged by a slave.
- for Slave mode: after software has acknowledged an address.

In either mode (Master or Slave), software is always involved in the address part of a message. In master and slave modes, data received and transmit data can be transferred by the DMA. The DMA supports 3 types of DMA requests:

- Data transfer in master mode
- Data transfer in slave mode
- Monitor mode

NOTE

DMA may be used in connection with Automatic Operation, to minimize software overhead time for I2C handling.

A received NACK (from a slave in Master mode, or from a master in Slave mode) will cause DMA to stop, and then an interrupt will be generated. A Repeated Start sensed on the I2C bus will similarly cause DMA to stop and then an interrupt to be generated. The Monitor function may be used with DMA if a channel is available. See “DMA with I2C monitor mode” for how DMA channels are used with the Monitor function.

DMA as a Master transmitter: basic sequence

- Software sets up DMA to transmit a message.
- Software causes a slave address with a write command to be sent, and then checks to see if the address was acknowledged.
- Software turns on DMA mode in the I2C.
- DMA transfers data and eventually completes the transfer.
- Software causes a stop (or repeated start) to be sent.

Software will be invoked to handle any exceptions to the standard transfer, like when a slave sends a NACK before the end of the transfer.

DMA as a Master receiver: basic sequence

- Software sets up DMA to receive a message.
- Software causes a slave address with read command to be sent and then checks that the address was acknowledged.
- Software starts DMA.
- DMA finishes.
- Software causes a stop or repeated start to be sent.

Software will be invoked to handle any exceptions to the standard transfer.

DMA as a Slave transmitter: basic sequence

- Software acknowledges an I2C address.
- Software sets up DMA to transmit a message.
- Software starts DMA.
- DMA finishes.

DMA as a Slave receiver: basic sequence

- Software receives an interrupt for a slave address received, and acknowledges the address.
- Software sets up DMA to receive a message, less the final data byte.
- Software starts DMA.
- DMA finishes.
- Software sets SLVNACK before receiving the final data byte.
- Software receives the final data byte.

41.5.2.2.3 Automatic operation

Automatic operation modes provide a way to reduce software overhead for I2C slave functions with some limitations, and these automatic operation modes are intended to be used primarily with slave DMA. Related control bits are SLVDMA, AUTOACK, and AUTOMATCHREAD in the SLCCTL register, and the AUTONACK bit in the SLVADR0 register. The next table shows how these control bits may be used. These cases apply when an address that matches SLVADR0 (and is qualified by SLVQUAL0) is received.

Table 345. Automatic operation cases

AUTONACK bit	AUTOACK bit	Received R/W bit matches AUTOMATCHREAD	SLVPENDING interrupt generated?	ACK/NACK on I2C-bus	Description
0	0	X	Yes	None	Normal, non-automatic operation
0	1	No	Yes	None	Automatic slave DMA: unexpected read/write case. Same as normal non-automatic operation.
X	1	Yes	No	ACK	Automatic slave DMA: expected read/write case. When the automatic Ack is sent, the SLVDMA bit is set and the AUTOACK bit is cleared.
1	0	X	No	NACK	Bus is ignored until software changes the setup.
1	1	No	No	NACK	Bus is ignored until software changes the setup.

41.5.2.2.4 Time-outs on I2C bus

A time-out feature on an I2C interface can be used to detect a “stuck” I2C bus and potentially do something to fix the condition. Two different types of time-out are supported: event timeout (STAT[EVENTTIMEOUT}) and SCL timeout (STAT[SCLTIMEOUT]). Both time-out types apply whenever the I2C interface and the time-out function are both enabled. Master, Slave, or Monitor functions do not need to be enabled.

- **EVENTTIMEOUT flag:** In the STAT register, the EVENTTIMEOUT flag is asserted when the time between bus events is longer than the time configured in the TIMEOUT register. These bus events include Start, Stop, and all changes on the I2C clock (SCL). The EVENTTIMEOUT time-out can be used to monitor an I2C bus within a system, as part of a scheme to keep the bus running after problems occur.
- **SCLTIMEOUT flag:** In the STAT register, the SCLTIMEOUT flag is asserted when the SCL signal remains low longer than the time configured in the TIMEOUT register. This corresponds to the SMBus time-out parameter TTIMEOUT. In this situation, a slave could reset its own I2C interface, in case it (the slave) is the offending device. If all listening slaves (including masters that can be addressed as slaves) do this, then the I2C bus will be released, unless a current master is causing the problem. Refer to the SMBus specification for more details.

Both types of I2C bus time-outs are generated only when the I2C bus is considered busy, for example, when there has been a Start condition more recently than a Stop condition.

41.5.2.2.5 10-bit addressing

Ten-bit addressing is accomplished by the I2C master sending a second address byte to extend a specific range of standard 7-bit addresses.

- For a master writing to a slave, the I2C frame simply continues with data after the 2 address bytes.
- For a master reading from a slave, the master needs to reverse the data direction after the second address byte. This is done by sending a Repeated Start, followed by a repeat of the same standard 7-bit address, with a Read bit. The slave must remember that it (the slave) has been addressed by the previous write operation and the slave must stay selected for the subsequent read with the correct partial I2C address.

For the Master function: the I2C is simply instructed to perform the 2-byte addressing as a normal write operation, followed either by more write data, or by a Repeated Start with a repeat of the first part of the 10-bit slave address and then reading in the normal fashion.

For the Slave function: the first part of the address is automatically matched in the same way as 7-bit addressing. The slave address qualifier feature can be used to intercept all potential 10-bit addresses (first address byte values F0 through F6), or just one 10-bit address. In the case of Slave Receiver mode, data is received in the normal way after software matches the first data byte to the remaining portion of the 10-bit address. The Slave should record the fact that it (the slave) has been addressed, in case there is a follow-up read operation.

For Slave Transmitter mode: the slave responds to the initial address in the same way as for Slave Receiver mode, and checks that it (the slave) has previously been addressed with a full 10-bit address. If the address matched is address 0, and address qualification is enabled, then software must verify that the first part of the 10-bit address is a complete match to the previous address, before the software acknowledges the address.

41.5.2.3 Clocks

This section describes clocks and special clocking requirements of the I2C module.

- The Master function of the I2C always requires a peripheral clock to be running.
- The Slave function actually can operate without any internal clocking when the slave is not currently addressed.

This means that reduced power modes up to deep-sleep mode can be entered, and the device will wake up when the I2C Slave function recognizes an address. Monitor mode can similarly wake up the device from a reduced power mode when information becomes available.

41.5.3 Initialization

I2C initialization

To initialize the I2C and related clocks:

- Reset the Flexcomm interface that is about to use the I2C function.
- Select the desired Flexcomm interface function by writing to the PSELID register of the related Flexcomm Interface. Note that any previous selection that has been made will be cleared if the Flexcomm interface itself is reset via the PRESETCTRL1 register.
- Configure the I2C for the desired functions:
  - Enable the clock to the Flexcomm register interface.
  - Enable or disable the related Flexcomm interface interrupts.
  - Configure the related Flexcomm Interface pin functions.
  - Configure the I2C clock and data rate, for both master and slave modes. The Flexcomm interface clock frequency should not be above 48 MHz.

NOTE

Although the I2C function is incorporated into the Flexcomm interface, the I2C function does not use the Flexcomm interface FIFO.

41.5.4 Memory Map and register definition

This section includes the I2C module memory map and detailed descriptions of all registers.

41.5.4.1 I2C Bus Interface register descriptions

41.5.4.1.1 I2C memory map

FLEXCOMM0.I2C base address: 4008_6000h

FLEXCOMM1.I2C base address: 4008_7000h

FLEXCOMM2.I2C base address: 4008_8000h

FLEXCOMM3.I2C base address: 4008_9000h

FLEXCOMM4.I2C base address: 4008_A000h

FLEXCOMM5.I2C base address: 4009_6000h

FLEXCOMM6.I2C base address: 4009_7000h

FLEXCOMM7.I2C base address: 4009_8000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
800	Configuration Register (CFG)	32	See section	See section
804	Status Register (STAT)	32	See section	See section
808	Interrupt Enable Set Register (INTENSET)	32	See section	See section
80C	Interrupt Enable Clear Register (INTENCLR)	32	See section	See section
810	Time-out Register (TIMEOUT)	32	See section	See section
814	Clock Divider Register (CLKDIV)	32	See section	See section
818	Interrupt Status Register (INTSTAT)	32	See section	See section
820	Master Control Register (MSTCTL)	32	See section	See section
824	Master Timing Register (MSTTIME)	32	See section	See section
828	Master Data Register (MSTDAT)	32	See section	See section
840	Slave Control Register (SLVCTL)	32	See section	See section
844	Slave Data Register (SLVDAT)	32	See section	See section
848 - 854	Slave Address Register (SLVADR0 - SLVADR3)	32	See section	See section
858	Slave Qualification for Address 0 Register (SLVQUAL0)	32	See section	See section
880	Monitor Receiver Data Register (MONRXDAT)	32	See section	See section
FFC	Peripheral Identification Register (ID)	32	RO	E030_1300

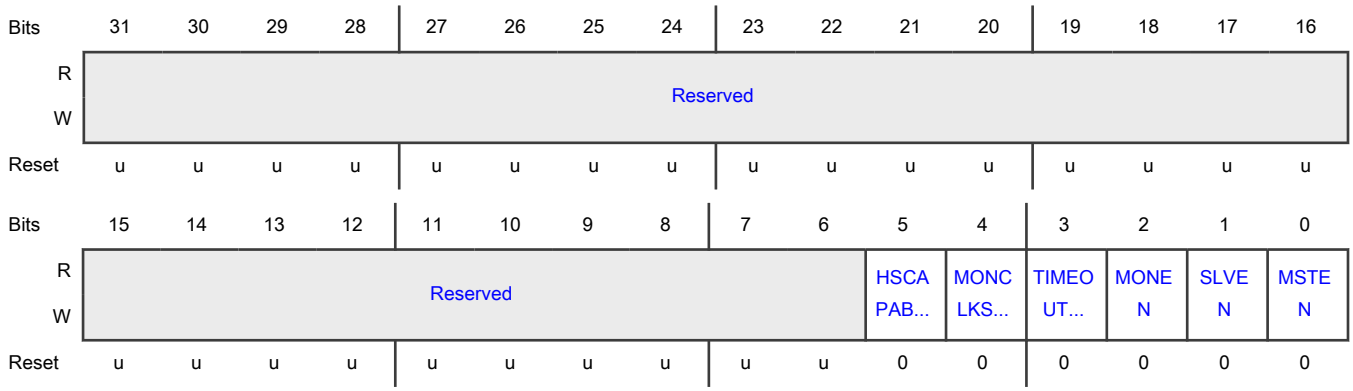
41.5.4.1.1.1 Configuration Register (CFG)

The CFG register contains mode settings that apply to Master, Slave, and Monitor functions.

Offset

Register	Offset
CFG	800h

Diagram



Fields

Field	Description
31-6 —	Reserved Read value is undefined; only zero should be written.
5 HSCAPABLE	High Speed mode Capable enable Enabling High Speed mode applies to all functions (Master, Slave, Monitor), because High Speed mode alters the way that the I2C pins drive and filter, as well as the timing for certain I2C signalling. The I2C interface will support Standard mode, Fast mode, and Fast mode Plus, if and only if those pins 'hardware support these modes. Any changes to be made to the pin controls, like changing the drive strength or filtering, must be made by software using the IOCON register associated with each I2C pin. 0 - Fast mode Plus enable 1 - High Speed mode enable
4 MONCLKSTR	Monitor function Clock Stretching 0 - Disabled. The Monitor function will not perform clock stretching. Software or DMA may not always be able to read data provided by the Monitor function before it (the data) is overwritten. This mode can be used when non-invasive monitoring is critical. 1 - Enabled. The Monitor function will perform clock stretching, to ensure that the software or DMA can read all incoming data supplied by the Monitor function.
3	I2C bus Time-out Enable 0 - Disabled. The time-out function is disabled. When disabled, the time-out function is internally reset.

Table continues on the next page...

Table continued from the previous page...

Field	Description
TIMEOUTEN	1 - Enabled. The time-out function is enabled. Both types of time-out flags will be generated and will cause interrupts if those flags are enabled. Typically, only one time-out flag will be used in a system.
2 MONEN	Monitor Enable 0 - Disabled. The I2C Monitor function is disabled. When disabled, the Monitor function configuration settings are not changed, but the Monitor function is internally reset. 1 - Enabled. The I2C Monitor function is enabled.
1 SLVEN	Slave Enable 0 - Disabled. The I2C slave function is disabled. When disabled, the Slave configuration settings are not changed, but the Slave function is internally reset. 1 - Enabled. The I2C slave function is enabled.
0 MSTEN	Master Enable 0 - Disabled. The I2C Master function is disabled. When disabled, the Master configuration settings are not changed, but the Master function is internally reset. 1 - Enabled. The I2C Master function is enabled.

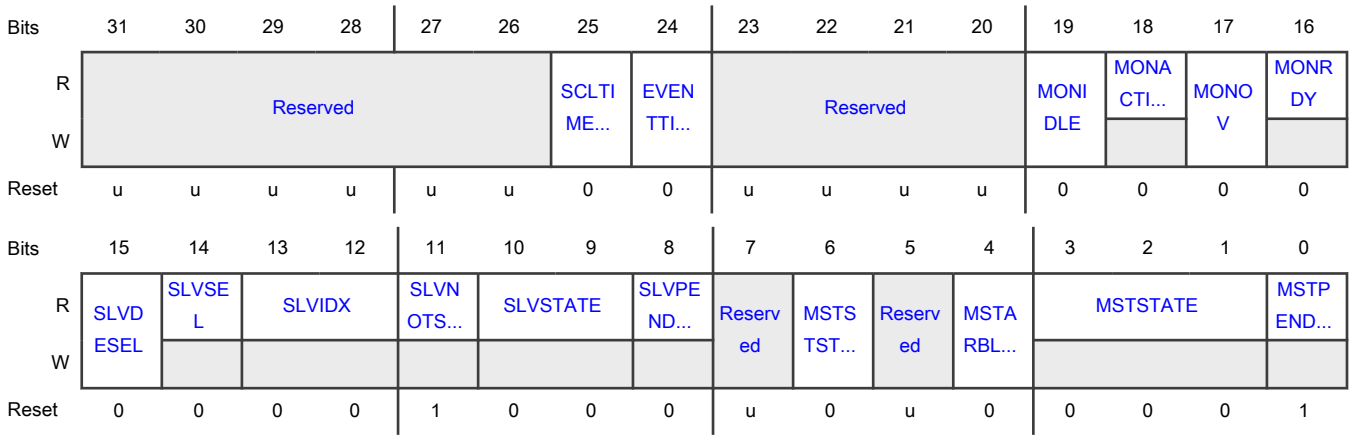
41.5.4.1.1.2 Status Register (STAT)

Status register for Master, Slave, and Monitor functions. The STAT register provides status flags and state information about all of the functions of the I2C interface. Access to bits in this register varies. RO = Read-only, W1C = write 1 to clear.

Offset

Register	Offset
STAT	804h

Diagram



Fields

Field	Description
31-26 —	Reserved Read value is undefined; only zero should be written.
25 SCLTIMEOUT	SCL Time-out Interrupt flag Indicates when SCL has remained low longer than the time specific by the TIMEOUT register. Can be cleared by writing a 1 to itself (SCLTIMEOUT). 0 - No time-out. SCL low time has not caused a time-out. 1 - Time-out. SCL low time has caused a time-out.
24 EVENTTIMEOUT	Event Time-out Interrupt flag Indicates when the time between events has been longer than the time specified by the TIMEOUT register. Events include Start, Stop, and clock edges. • No time-out is created when the I2C bus is idle. • Can be cleared by writing a 1 to itself (EVENTTIMEOUT). 0 - No time-out. I2C bus events have not caused a time-out. 1 - Event time-out. The time between I2C bus events has been longer than the time specified by the TIMEOUT register.
23-20 —	Reserved Read value is undefined; only zero should be written.
19 MONIDLE	Monitor Idle flag MONIDLE flag is set when the Monitor function sees the I2C bus change from active to inactive. This can be used by software to decide when to process data accumulated by the Monitor function. • MONIDLE flag will cause an interrupt when set (if enabled via the INTENSET register). • Can be cleared by writing a 1 to itself (MONIDLE). 0 - Not idle. The I2C bus is not idle, or MONIDLE flag has been cleared by software. 1 - Idle. The I2C bus has gone idle at least once, since the last time MONIDLE flag was cleared by software.
18 MONACTIVE	Monitor Active flag Indicates when the Monitor function considers the I2C bus to be active. Active is defined here as when a Master is on the bus, meaning that a bus Start has occurred more recently than a bus Stop. 0 - Inactive. The Monitor function considers the I2C bus to be inactive. 1 - Active. The Monitor function considers the I2C bus to be active.
17 MONOV	Monitor Overflow flag 0 - No overrun. Monitor data has not overrun. 1 - Overrun. A Monitor data overrun has occurred. An overrun can only happen when Monitor clock stretching not enabled via the CFG[MONCLKSTR] bit. Writing 1 to MONOV bit clears the MONOV flag.

Table continues on the next page...

Table continued from the previous page...

Field	Description
16 MONRDY	Monitor Ready MONRDY flag is cleared when the MONRXDAT register is read. 0 - No data. The Monitor function does not currently have data available. 1 - Data waiting. The Monitor function has data waiting to be read.
15 SLVDESEL	Slave Deselected flag • SLVDESEL flag will cause an interrupt when set (if enabled via INTENSET). • SLVDESEL flag can be cleared by writing a 1 to itself (SLVDESEL). 0 - Not deselected. The Slave function has not become deselected. This does not mean that the Slave is currently selected. That information is in the SLVSEL flag. 1 - Deselected. The Slave function has become deselected. This is specifically caused by the SLVSEL flag changing from 1 to 0. See SLVSEL for details about when that event occurs.
14 SLVSEL	Slave selected flag SLVSEL is set after an address match • when software tells the Slave function to acknowledge the address, • or when the address has been automatically acknowledged. SLVSEL is cleared • when another address cycle presents an address that does not match an enabled address on the Slave function, • or when slave software decides to NACK a matched address, • or when there is a Stop detected on the bus, • or when the master NACKs slave data, and in some combinations of Automatic Operation. SLVSEL is not cleared if software NACKs data. 0 - Not selected. The Slave function is not currently selected. 1 - Selected. The Slave function is currently selected.
13-12 SLVIDX	Slave address match Index T SLVIDX field is valid when the I2C slave function has been selected, by receiving an address that matches one of the slave addresses defined by any enabled slave address registers, and also provides an identification of the address that was matched. It is possible that more than one address could have been matched, but only one match can be reported here. 00 - Address 0. Slave address 0 was matched. 01 - Address 1. Slave address 1 was matched. 10 - Address 2. Slave address 2 was matched. 11 - Address 3. Slave address 3 was matched.
11	Slave Not Stretching

Table continues on the next page...

Table continued from the previous page...

Field	Description
SLVNOTSTR	Indicates when the slave function is stretching the I2C clock. Clock stretching is needed to gracefully invoke Deep Sleep modes during slave operations. This read-only flag reflects the slave function status in real time. 0 - Stretching. The slave function is currently stretching the I2C bus clock. Deep-Sleepmode cannot be entered at this time. 1 - Not stretching. The slave function is not currently stretching the I2C bus clock. Deep-sleep mode can be entered at this time.
10-9 SLVSTATE	Slave State Each value of the SLVSTATE field indicates a specific required service for the Slave function. Note that the occurrence of some states and how they are handled are affected by DMA mode and Automatic Operation modes. 00 - Slave address. Address plus R/W received. At least one of the 4 slave addresses has been matched by hardware. 01 - Slave receive. Received data is available (in Slave Receiver mode). 10 - Slave transmit. Data can be transmitted (in Slave Transmitter mode). 11 - Reserved
8 SLVPENDING	Slave Pending Indicates that the Slave function is waiting to continue communication on the I2C bus and needs software service. • The SLVPENDING flag will cause an interrupt when set (and if the interrupt is enabled via INTENSET register). • The SLVPENDING flag is not set when the DMA is handling an event (if SLVCTL[SLVDMA] bit is set). • The SLVPENDING flag is read-only and is automatically cleared when a 1 is written to SLVCTL[SLVCONTINUE]. The point in time when SlvPending is set depends on whether the I2C interface is in HSCAPABLE mode. When the I2C interface is configured to be HSCAPABLE, HS master codes are detected automatically. Due to HS I2C specification requirements, slave addresses must also be detected automatically, because the address must be acknowledged before the clock can be stretched. 0 - In progress. The Slave function does not currently need software service. 1 - Pending. The Slave function needs software service. Information about what is needed is in the Slave state field (SLVSTATE).
7 —	Reserved Read value is undefined; only zero should be written.
6 MSTSTSTPERR	Master Start/Stop Error flag • Cleared by software writing a 1 to this bit (MSTSTSTPERR) • Cleared automatically when a 1 is written to MSTCONTINUE

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - No Start/Stop Error has occurred. 1 - The Master function has experienced a Start/Stop Error. A Start or Stop was detected at a time when Start or Stop is not allowed by the I2C specification. The Master interface has stopped driving the bus and gone into an idle state; no action is required. A request for a Start could be made, or software could attempt to make sure that the bus has not stalled.
5 —	Reserved Read value is undefined; only zero should be written.
4 MSTARBLOSS	Master Arbitration Loss flag • Cleared by software writing a 1 to this bit (MSTARBLOSS) • Cleared automatically when a 1 is written to MSTCONTINUE 0 - No Arbitration Loss has occurred 1 - Arbitration loss. The Master function has experienced an Arbitration Loss. At this point, the Master function has already stopped driving the bus and has gone into an idle state. Software can respond by doing nothing, or by sending a Start (to attempt to gain control of the bus when the bus next becomes idle).
3-1 MSTSTATE	Master State code The master state code indicates the master state when the MSTPENDING bit is set, which means that the master is pending or is in the idle state. Each value of the MSTSTATE field indicates a specific required service for the Master function. All other MSTSTATE values are reserved. 000 - Idle. The Master function is available to be used for a new transaction. 001 - Receive ready. Received data is available (in Master Receiver mode). Address plus Read was previously sent and Acknowledged by a slave. 010 - Transmit ready. Data can be transmitted (in Master Transmitter mode). Address plus Write was previously sent and Acknowledged by a slave. 011 - NACK Address. Slave NACKed address. 100 - NACK Data. Slave NACKed transmitted data.
0 MSTPENDING	Master Pending Indicates that the Master is either waiting to continue communications on the I2C bus (pending) or is idle. When the master is pending, the MSTSTATE bits indicate what type of software service (if any) that the master expects. • The MSTPENDING flag will cause an interrupt when set (if enabled via the INTENSET register). • The MSTPENDING flag is not set when the DMA is handling an event (if the MSTDMA bit in the MSTCTL register is set). • If the master is in the idle state, and if no communication is needed, then mask this interrupt (MSTPENDING). 0 - In progress. Communication is in progress and the Master function is busy and cannot currently accept a command.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Pending. The Master function needs software service or is in the idle state. If the master is not in the idle state, then the master is waiting to receive or transmit data, or is waiting for the NACK bit.

41.5.4.1.1.3 Interrupt Enable Set Register (INTENSET)

The INTENSET register controls which I2C status flags generate interrupts. Writing a 1 to a bit position in this register enables an interrupt in the corresponding position in the STAT register, if an interrupt is supported there. Reading INTENSET indicates which interrupts are currently enabled.

Offset

Register	Offset
INTENSET	808h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved						SCLTIME...	EVENTTI...	Reserved				MONIDLE...	Reserved	MONOVEN	MONRDYEN
Reset	u	u	u	u	u	u	0	0	u	u	u	u	0	u	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SLVDESE...	Reserved			SLVNOTS...	Reserved		SLVPEND...	Reserved	MSTSTST...	Reserved	MSTARBL...	Reserved			MSTPEND...
Reset	0	u	u	u	0	u	u	0	u	0	u	0	u	u	u	0

Fields

Field	Description
31-26 —	Reserved Read value is undefined; only zero should be written.
25 SCLTIMEOUTEN	SCL Time-out interrupt Enable 0 - Disabled. The SCL time-out interrupt is disabled. 1 - Enabled. The SCL time-out interrupt is enabled.
24 EVENTTIMEOUTEN	Event Time-out interrupt Enable 0 - Disabled. The Event time-out interrupt is disabled. 1 - Enabled. The Event time-out interrupt is enabled.
23-20	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	Read value is undefined; only zero should be written.
19 MONIDLEEN	Monitor Idle interrupt Enable 0 - Disabled. The MonIdle interrupt is disabled. 1 - Enabled. The MonIdle interrupt is enabled.
18 —	Reserved Read value is undefined; only zero should be written.
17 MONOVEN	Monitor Overrun interrupt Enable 0 - Disabled. The MonOv interrupt is disabled. 1 - Enabled. The MonOv interrupt is enabled.
16 MONRDYEN	Monitor data Ready interrupt Enable 0 - Disabled. The MonRdy interrupt is disabled. 1 - Enabled. The MonRdy interrupt is enabled.
15 SLVDESELEN	Slave Deselect interrupt Enable 0 - Disabled. The SlvDeSel interrupt is disabled. 1 - Enabled. The SlvDeSel interrupt is enabled.
14-12 —	Reserved Read value is undefined; only zero should be written.
11 SLVNOTSTREN	Slave Not Stretching interrupt Enable 0 - Disabled. The SlvNotStr interrupt is disabled. 1 - Enabled. The SlvNotStr interrupt is enabled.
10-9 —	Reserved Read value is undefined; only zero should be written.
8 SLVPENDINGEN	Slave Pending interrupt Enable 0 - Disabled. The SlvPending interrupt is disabled. 1 - Enabled. The SlvPending interrupt is enabled.
7 —	Reserved Read value is undefined; only zero should be written.
6 MSTSTSTPEREN	Master Start/Stop Error interrupt Enable 0 - Disabled. The MstStStpErr interrupt is disabled. 1 - Enabled. The MstStStpErr interrupt is enabled.
5	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	Read value is undefined; only zero should be written.
4 MSTARBLOSSEN	Master Arbitration Loss interrupt Enable 0 - Disabled. The MstArbLoss interrupt is disabled. 1 - Enabled. The MstArbLoss interrupt is enabled.
3-1 —	Reserved Read value is undefined; only zero should be written.
0 MSTPENDINGEN	Master Pending interrupt Enable 0 - Disabled. The MstPending interrupt is disabled. 1 - Enabled. The MstPending interrupt is enabled.

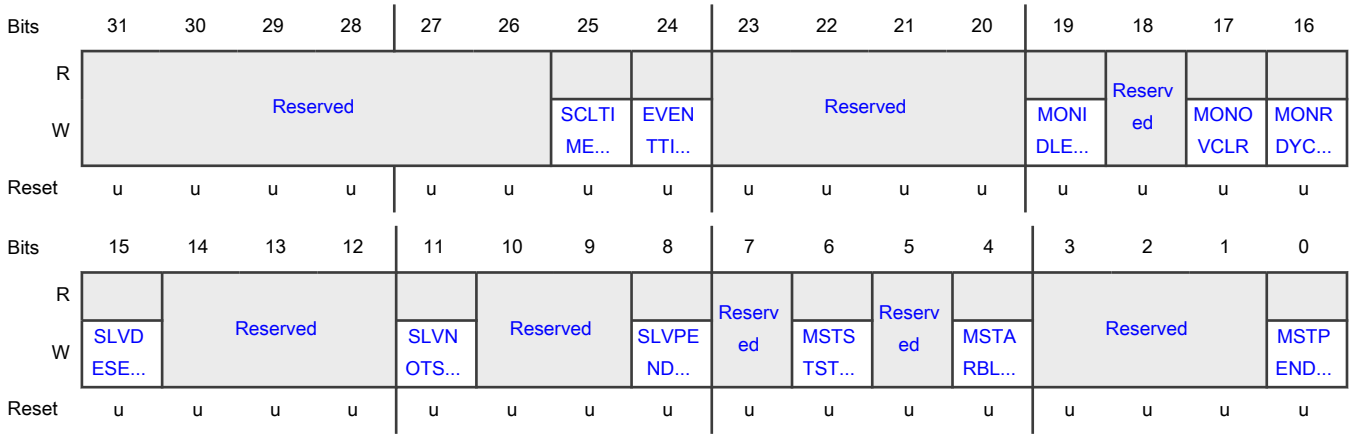
41.5.4.1.1.4 Interrupt Enable Clear Register (INTENCLR)

Writing a 1 to a bit position in INTENCLR clears the corresponding position in the INTENSET register, disabling that interrupt. INTENCLR is a write-only register. Bits that do not correspond to defined bits in INTENSET are reserved and only zeros should be written to them.

Offset

Register	Offset
INTENCLR	80Ch

Diagram



Fields

Field	Description
31-26 —	Reserved Read value is undefined; only zero should be written.
25 SCLTIMEOUTCLR	SCL time-out interrupt clear 0 - No effect on interrupt 1 - Clears the interrupt bit in INTENSET register
24 EVENTTIMEOUTCLR	Event time-out interrupt clear 0 - No effect on interrupt 1 - Clears the interrupt bit in INTENSET register
23-20 —	Reserved Read value is undefined; only zero should be written.
19 MONIDLECLR	Monitor Idle interrupt clear 0 - No effect on interrupt 1 - Clears the interrupt bit in INTENSET register
18 —	Reserved Read value is undefined; only zero should be written.
17 MONOVCLR	Monitor Overrun interrupt clear 0 - No effect on interrupt 1 - Clears the interrupt bit in INTENSET register
16 MONRDYCLR	Monitor data Ready interrupt clear 0 - No effect on interrupt 1 - Clears the interrupt bit in INTENSET register
15 SLVDESELCLR	Slave Deselect interrupt clear 0 - No effect on interrupt 1 - Clears the interrupt bit in INTENSET register
14-12 —	Reserved Read value is undefined; only zero should be written.
11 SLVNOTSTRCLR	Slave Not Stretching interrupt clear 0 - No effect on interrupt 1 - Clears the interrupt bit in INTENSET register
10-9 —	Reserved Read value is undefined; only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
8 SLVPENDINGCLR	Slave Pending interrupt clear 0 - No effect on interrupt 1 - Clears the interrupt bit in INTENSET register
7 —	Reserved Read value is undefined; only zero should be written.
6 MSTSTSTPERRCLR	Master Start/Stop Error interrupt clear 0 - No effect on interrupt 1 - Clears the interrupt bit in INTENSET register
5 —	Reserved Read value is undefined; only zero should be written.
4 MSTARBLOSSCLR	Master Arbitration Loss interrupt clear 0 - No effect on interrupt 1 - Clears the interrupt bit in INTENSET register
3-1 —	Reserved Read value is undefined; only zero should be written.
0 MSTPENDINGCLR	Master Pending interrupt clear 0 - No effect on interrupt 1 - Clears the interrupt bit in INTENSET register

41.5.4.1.1.5 Time-out Register (TIMEOUT)

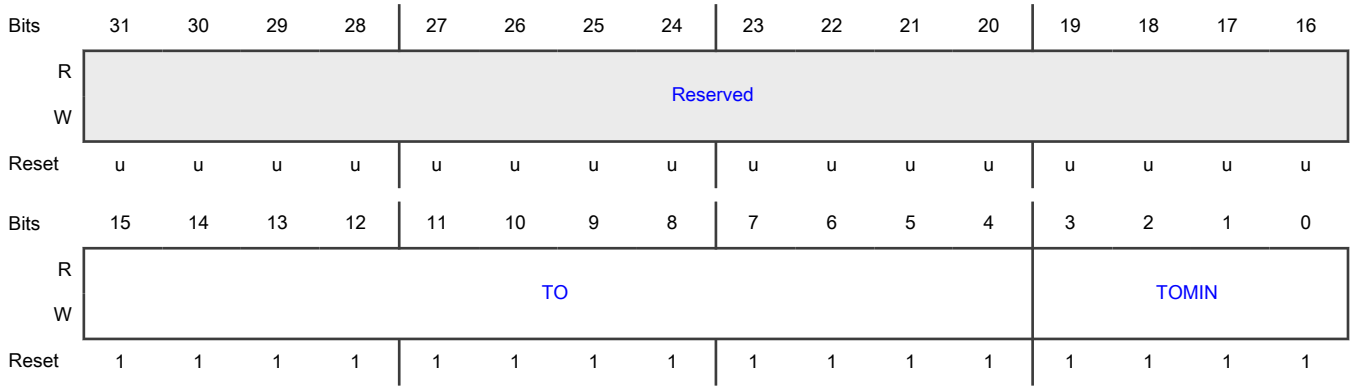
The TIMEOUT register allows setting an upper limit to certain I2C-bus times, informing by status flag and/or interrupt when those times are exceeded. Two time-outs are generated, and software can elect to use either of the timeouts.

- EVENTTIMEOUT checks the time between bus events while the bus is not idle: Start, SCL rising, SCL falling, and Stop. The STAT[EVENTTIMEOUT] status flag is set if the time between any two events becomes longer than the time configured in the TIMEOUT register. The EVENTTIMEOUT status flag can cause an interrupt (if enabled to do so by the INTENSET[EVENTTIMEOUTEN] bit).
- SCLTIMEOUT checks only the time that the SCL signal remains low while the bus is not idle. The STAT[SCLTIMEOUT] status flag is set if SCL remains low longer than the time configured in the TIMEOUT register. The SCLTIMEOUT status flag can cause an interrupt (if enabled to do so by the INTENSET[SCLTIMEOUTEN] bit). The SCLTIMEOUT can be used with the SMBus.

Offset

Register	Offset
TIMEOUT	810h

Diagram



Fields

Field	Description
31-16 —	Reserved Read value is undefined; only zero should be written.
15-4 TO	Time-out time value Specifies the time-out interval value in increments of 16 I2C function clocks, as defined by the CLKDIV register. To change this Time-out value while I2C is in operation, disable all time-outs, write a new value to TIMEOUT, then re-enable time-outs. 0000_0000_0000 - A time-out will occur after 16 counts of the I2C function clock. 0000_0000_0001 - A time-out will occur after 32 counts of the I2C function clock. 1111_1111_1111 - A time-out will occur after 65,536 counts of the I2C function clock.
3-0 TOMIN	Time-out time value, the bottom 4 bits The bottom 4 bits are hard-wired to 0xF, which gives a minimum time-out of 16 I2C function clocks, and also a time-out resolution of 16 I2C function clocks.

41.5.4.1.1.6 Clock Divider Register (CLKDIV)

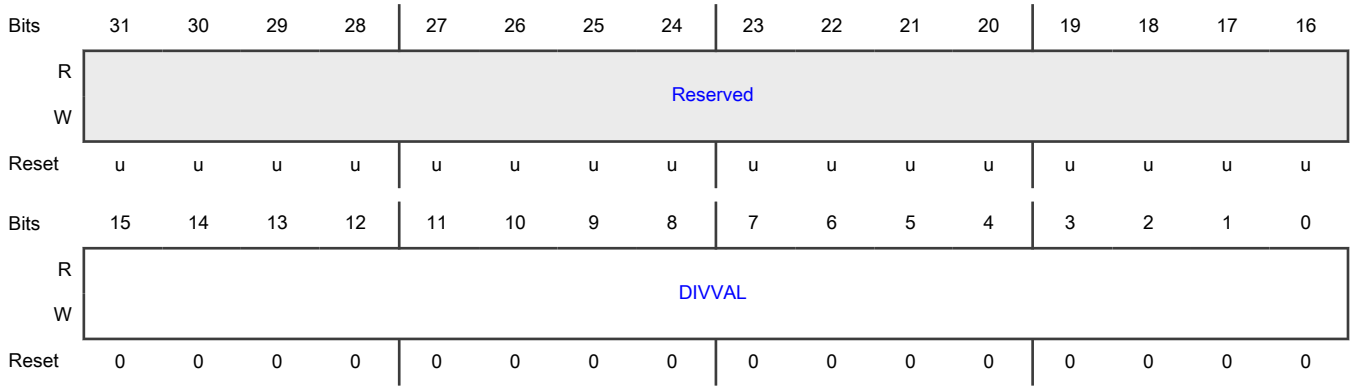
Clock pre-divider for the entire I2C interface. This determines what time increments are used for the MSTTIME register, and controls some timing of the Slave function.

The CLKDIV register divides down the Flexcomm clock (FCLK) to produce the I2C function clock that is used to time various aspects of the I2C interface. The I2C function clock is used for some internal operations in the I2C interface and to generate the timing required by the I2C-bus specification, some of which are user-configured in the MSTTIME register for Master operation. Slave operation uses CLKDIV for some timing functions.

Offset

Register	Offset
CLKDIV	814h

Diagram



Fields

Field	Description
31-16 —	Reserved Read value is undefined; only zero should be written.
15-0 DIVVAL	Divider Value Controls how the Flexcomm clock (FCLK) is used by the I2C functions that need an internal clock (to operate). 0000_0000_0000_0000 - FCLK is used directly by the I2C. 0000_0000_0000_0001 - FCLK is divided by 2 before being used by the I2C. 0000_0000_0000_0010 - FCLK is divided by 3 before being used by the I2C. 1111_1111_1111_1111 - FCLK is divided by 65,536 before being used by the I2C.

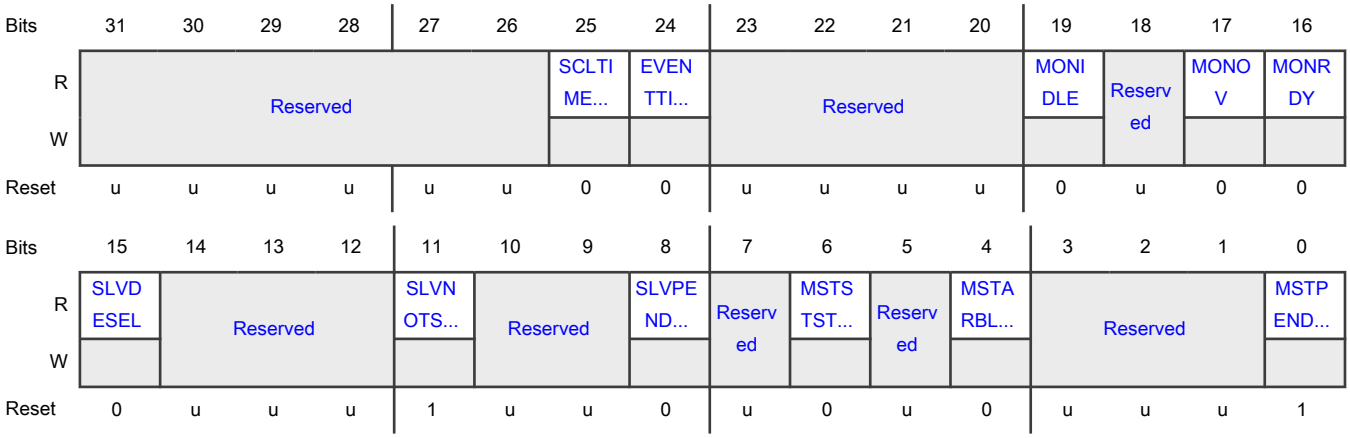
41.5.4.1.1.7 Interrupt Status Register (INTSTAT)

The INTSTAT register provides register provides a view of the Master, Slave, and Monitor interrupt flags that are currently enabled. This register can simplify the software handling of interrupts.

Offset

Register	Offset
INTSTAT	818h

Diagram



Fields

Field	Description
31-26 —	Reserved Read value is undefined; only zero should be written.
25 SCLTIMEOUT	SCL Time-out Interrupt flag 0 - Not active 1 - Active
24 EVENTTIMEOUT	Event Time-out Interrupt flag 0 - Not active 1 - Active
23-20 —	Reserved Read value is undefined; only zero should be written.
19 MONIDLE	Monitor Idle flag 0 - Not active 1 - Active
18 —	Reserved Read value is undefined; only zero should be written.
17 MONOV	Monitor Overflow flag 0 - Not active 1 - Active
16 MONRDY	Monitor Ready 0 - Not active 1 - Active

Table continues on the next page...

Table continued from the previous page...

Field	Description
15 SLVDESEL	Slave Deselected flag 0 - Not active 1 - Active
14-12 —	Reserved Read value is undefined; only zero should be written.
11 SLVNOTSTR	Slave Not Stretching status 0 - Not active 1 - Active
10-9 —	Reserved Read value is undefined; only zero should be written.
8 SLVPENDING	Slave Pending 0 - Not active 1 - Active
7 —	Reserved Read value is undefined; only zero should be written.
6 MSTSTSTPERR	Master Start/Stop Error flag 0 - Not active 1 - Active
5 —	Reserved Read value is undefined; only zero should be written.
4 MSTARBLOSS	Master Arbitration Loss flag 0 - Not active 1 - Active
3-1 —	Reserved
0 MSTPENDING	Master Pending 0 - Not active 1 - Active

41.5.4.1.1.8 Master Control Register (MSTCTL)

The MSTCTL register contains bits that control various functions of the I2C Master interface. Only write to the MSTCTL register when the master is pending (STAT[MSTPENDING] = 1). Software should always write a complete value to MSTCTL,

and not OR new control bits into the register, as is possible in other registers such as CFG. This is because MSTSTART and MSTSTOP are not self-clearing flags. ORing in new data following a Start or Stop may cause undesirable side effects.

After an initial I2C Start, MSTCTL should generally only be written when the MSTPENDING flag in the STAT register is set, after the last bus operation has completed. An exception is when DMA is being used and a transfer completes; in this case there is no MSTPENDING flag, and the MSTDMA control bit would be cleared by software potentially at the same time as when setting either the MSTSTOP or MSTSTART control bit.

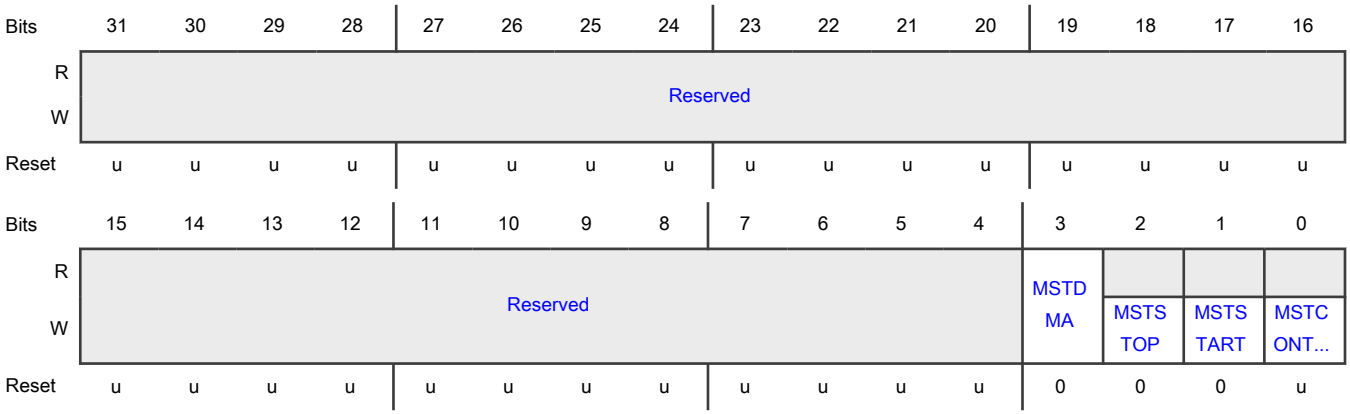
NOTE

When in the idle or slave NACKed states, set the MSTDMA bit either with or after the MSTCONTINUE bit. MSTDMA can be cleared at any time.

Offset

Register	Offset
MSTCTL	820h

Diagram



Fields

Field	Description
31-4 —	Reserved Read value is undefined; only zero should be written.
3 MSTDMA	Master DMA enable Data operations of the I2C can be performed with DMA. • Protocol type operations such as Start, address, Stop, and address match must always be done with software, typically via an interrupt. • Address acknowledgement must also be done by software, except when the I2C is configured to be High Speed mode capable (CFG[HSCAPABLE]) and address acknowledgement is handled entirely by hardware, or when Automatic Operation is enabled. • When a DMA data transfer is complete, MSTDMA must be cleared before beginning the next operation (typically a Start or Stop).

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - Disable. No DMA requests are generated for master operation. 1 - Enable. A DMA request is generated for I2C master data operations. When this I2C master is generating Acknowledge bits in Master Receiver mode, the acknowledge is generated automatically.
2 MSTSTOP	Master Stop control(write-only) 0 - No effect 1 - Stop. A Stop will be generated on the I2C bus at the next allowed time, preceded by a NACK to the slave if the master is receiving data from the slave (in Master Receiver mode).
1 MSTSTART	Master Start control(write-only) 0 - No effect 1 - Start. A Start will be generated on the I2C bus at the next allowed time.
0 MSTCONTINUE	Master Continue(write-only) 0 - No effect 1 - Continue. Informs the Master function to continue to the next operation. This action must done after writing transmit data, reading received data, or any other housekeeping related to the next bus operation.

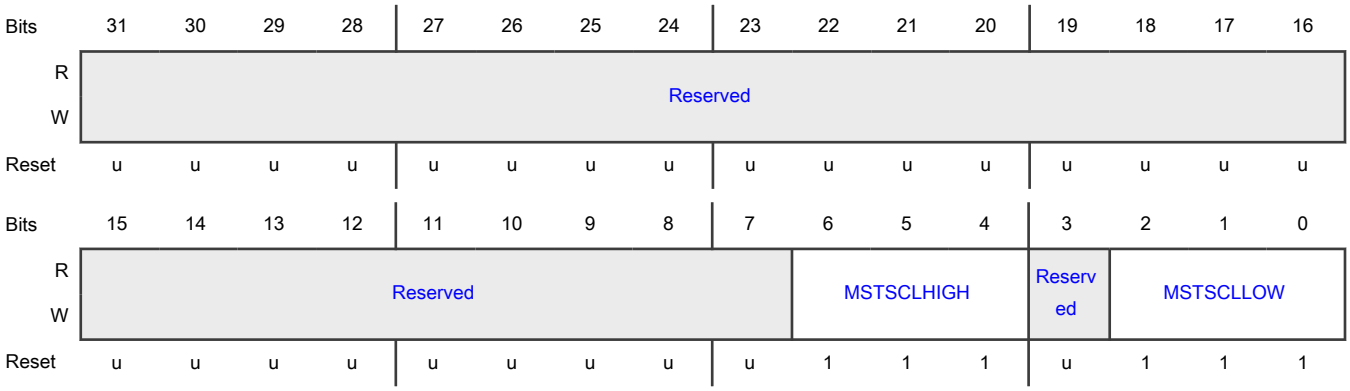
41.5.4.1.1.9 Master Timing Register (MSTTIME)

The MSTTIME register allows programming of certain times that may be controlled by the Master function. These include the clock (SCL) high and low times, repeated Start setup time, and transmitted data setup time.

Offset

Register	Offset
MSTTIME	824h

Diagram



Fields

Field	Description
31-7 —	Reserved Read value is undefined; only zero should be written.
6-4 MSTSCLHIGH	Master SCL High time Specifies the minimum high time that will be asserted by this master on SCL. Other masters in a multi-master system could shorten this time. This corresponds to the parameter t_{HIGH} in the I2C bus specification. I2C bus specification parameters $t_{SU;STO}$ and $t_{HD;STA}$ have the same values and are also controlled by MSTSCLHIGH. 000 - 2 clocks. Minimum SCL high time is 2 clocks of the I2C clock pre-divider. 001 - 3 clocks. Minimum SCL high time is 3 clocks of the I2C clock pre-divider . 010 - 4 clocks. Minimum SCL high time is 4 clocks of the I2C clock pre-divider. 011 - 5 clocks. Minimum SCL high time is 5 clocks of the I2C clock pre-divider. 100 - 6 clocks. Minimum SCL high time is 6 clocks of the I2C clock pre-divider. 101 - 7 clocks. Minimum SCL high time is 7 clocks of the I2C clock pre-divider. 110 - 8 clocks. Minimum SCL high time is 8 clocks of the I2C clock pre-divider. 111 - 9 clocks. Minimum SCL high time is 9 clocks of the I2C clock pre-divider.
3 —	Reserved
2-0 MSTSCLLOW	Master SCL Low time Specifies the minimum low time that will be asserted by this master on SCL. Other devices on the bus (masters or slaves) could lengthen this time. This corresponds to the parameter t_{LOW} in the I2C bus specification. I2C bus specification parameters t_{BUF} and $t_{SU;STA}$ have the same values and are also controlled by MSTSCLLOW. 000 - 2 clocks. Minimum SCL low time is 2 clocks of the I2C clock pre-divider. 001 - 3 clocks. Minimum SCL low time is 3 clocks of the I2C clock pre-divider. 010 - 4 clocks. Minimum SCL low time is 4 clocks of the I2C clock pre-divider. 011 - 5 clocks. Minimum SCL low time is 5 clocks of the I2C clock pre-divider. 100 - 6 clocks. Minimum SCL low time is 6 clocks of the I2C clock pre-divider. 101 - 7 clocks. Minimum SCL low time is 7 clocks of the I2C clock pre-divider. 110 - 8 clocks. Minimum SCL low time is 8 clocks of the I2C clock pre-divider. 111 - 9 clocks. Minimum SCL low time is 9 clocks of the I2C clock pre-divider.

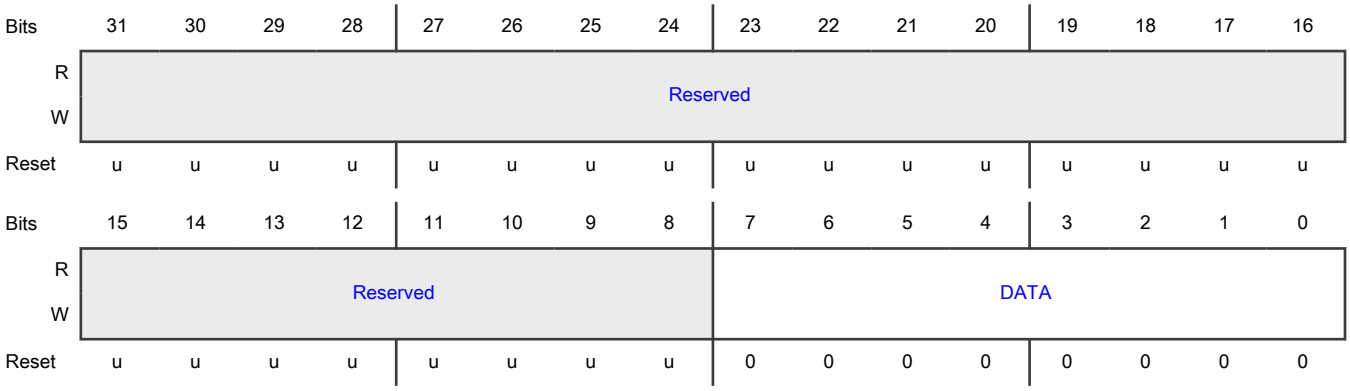
41.5.4.1.1.10 Master Data Register (MSTDAT)

The MSTDAT register provides the means to read the most recently received data for the Master function, and to transmit data using the Master function.

Offset

Register	Offset
MSTDAT	828h

Diagram



Fields

Field	Description
31-8 —	Reserved Read value is undefined; only zero should be written.
7-0 DATA	Master function data register • Read: read the most recently received data for the Master function. • Write: transmit data using the Master function.

41.5.4.1.1.11 Slave Control Register (SLVCTL)

The SLVCTL register contains bits that control various functions of the I2C Slave interface. Only write to the SLVCTL register when the slave is pending (STAT[SLVPENDING] = 1).

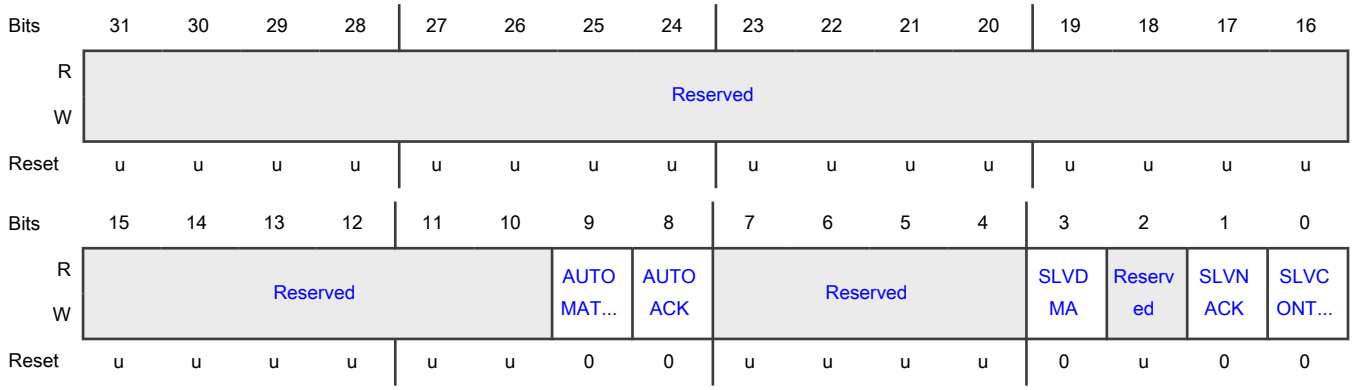
NOTE

When in the slave address state (slave state 0), set the SLVDMA bit either with or after the SLVCONTINUE bit. SLVDMA can be cleared at any time.

Offset

Register	Offset
SLVCTL	840h

Diagram



Fields

Field	Description
31-10 —	Reserved Read value is undefined; only zero should be written.
9 AUTOMATCHREAD	Automatic Match Read When AUTOACK is set, AUTOMATCHREAD bit controls whether it matches a read or write request on the next header with an address matching SLVADR0. Because DMA needs to be configured to match the transfer direction, the direction needs to be specified. AUTOMATCHREAD bit allows a direction to be chosen for the next operation. 0 - In Automatic Mode, the expected next operation is an I2C write. 1 - In Automatic Mode, the expected next operation is an I2C read.
8 AUTOACK	Automatic Acknowledge • When AUTOACK bit is set, it will cause an I2C header that matches SLVADR0 and the direction set by AUTOMATCHREAD to be ACKed immediately; this is used with DMA to allow processing of the data without intervention. • If AUTOACK bit is clear and a header matches SLVADR0, then the behavior is controlled by SLVADR0[AUTONACK], allowing NACK or interrupt. 0 - Normal, non-automatic operation. If AUTONACK = 0, then a SlvPending interrupt is generated when a matching address is received. If AUTONACK = 1, then received addresses are NACKed (ignored). 1 - A header with matching SLVADR0 and matching direction as set by AUTOMATCHREAD will be ACKed immediately, allowing the master to move on to the data bytes. If the address matches SLVADR0, but the direction does not match AUTOMATCHREAD, then the behavior will depend on the SLVADR0[AUTONACK] bit: if AUTONACK is set, then it will be Nacked; if AUTONACK is clear, then a SlvPending interrupt is generated.
7-4 —	Reserved Read value is undefined; only zero should be written.
3 SLVDMA	Slave DMA enable 0 - Disabled. No DMA requests are issued for Slave mode operation.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Enabled. DMA requests are issued for I2C slave data transmission and reception.
2 —	Reserved Read value is undefined; only zero should be written.
1 SLVNACK	Slave NACK 0 - No effect 1 - NACK. Causes the Slave function to NACK the master when the slave is receiving data from the master (in Slave Receiver mode).
0 SLVCONTINUE	Slave Continue 0 - No effect 1 - Continue. Informs the Slave function to continue to the next operation, by clearing the STAT[SLVPENDING] flag. This must be done after writing transmit data, reading received data, or any other housekeeping related to the next bus operation. Automatic Operation has different requirements. SLVCONTINUE should not be set unless SLVPENDING = 1.

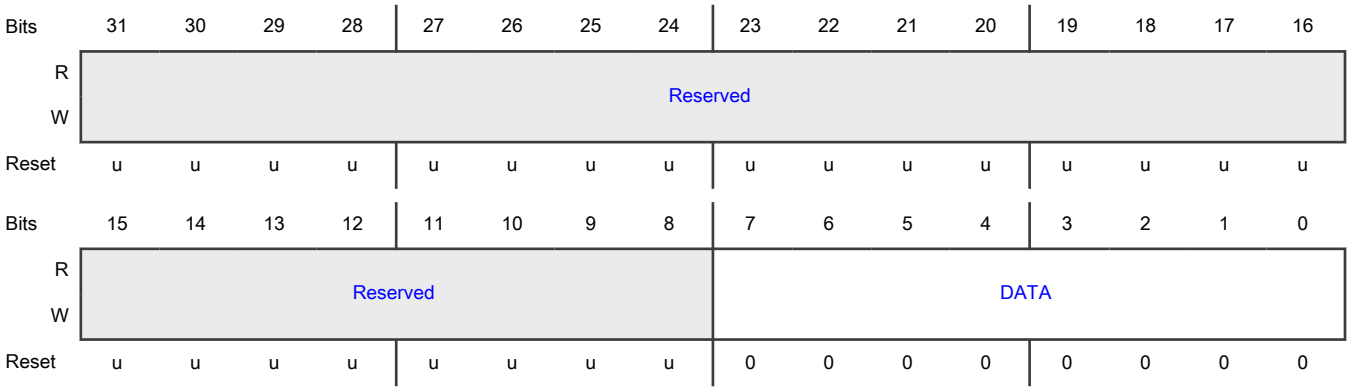
41.5.4.1.1.12 Slave Data Register (SLVDAT)

The SLVDAT register provides the means to read the most recently received data for the Slave function and to transmit data using the Slave function.

Offset

Register	Offset
SLVDAT	844h

Diagram



Fields

Field	Description
31-8 —	Reserved Read value is undefined; only zero should be written.
7-0 DATA	Slave function data register • Read: read the most recently received data for the Slave function. • Write: transmit data using the Slave function.

41.5.4.1.1.13 Slave Address Register (SLVADR0 - SLVADR3)

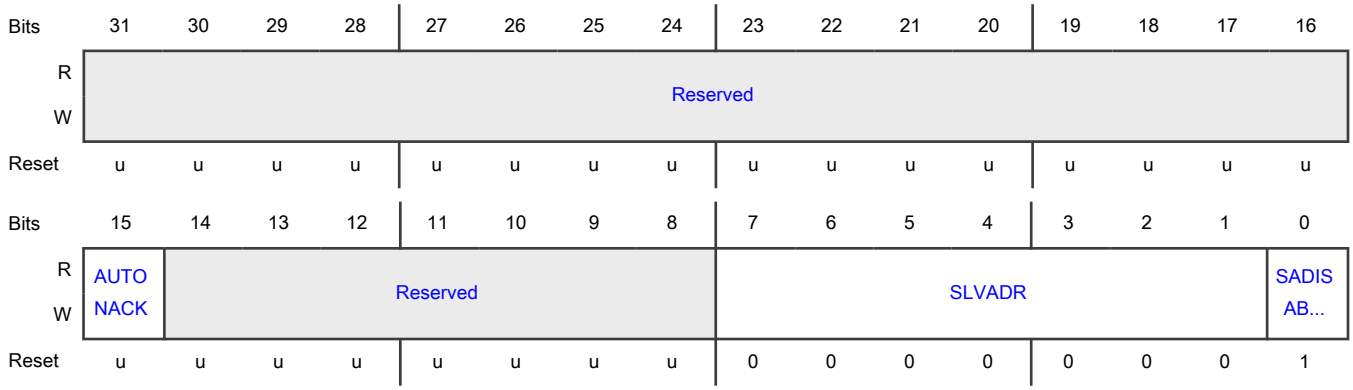
The SLVADR0 register allows enabling and defining one of the addresses that can be automatically recognized by the I2C slave hardware. The I2C slave function has a total of 4 address comparators. The value in SLVADR0 can be qualified by the setting of the SLVQUAL0 register.

The slave address registers (SLVADR1 - SLVADR3) provide 3 additional addresses that can be automatically recognized by the I2C slave hardware. The additional 3 address comparators do not include the address qualifier feature. For handling of the general call address, one of the 4 address registers can be programmed to respond to address 0.

Offset

Register	Offset
SLVADR0	848h
SLVADR1	84Ch
SLVADR2	850h
SLVADR3	854h

Diagram



Fields

Field	Description
31-16 —	Reserved Read value is undefined; only zero should be written.
15 AUTONACK	Automatic NACK operation Used in conjunction with AUTOACK and AUTOMATCHREAD, AUTONACK allows software to ignore I2C traffic while handling previous I2C data or other operations. 0 - Normal operation, matching I2C addresses are not ignored. 1 - Automatic-only mode. All incoming addresses are ignored (NACKed), unless AUTOACK is set, and the address matches SLVADRn, and AUTOMATCHREAD matches the direction.
14-8 —	Reserved Read value is undefined; only zero should be written.
7-1 SLVADR	Slave Address. The 7-bit slave address that is compared to received addresses, if enabled.
0 SADISABLE	Slave Address n Disable 0 - Enabled. Slave Address n is enabled. 1 - Ignored. Slave Address n is ignored.

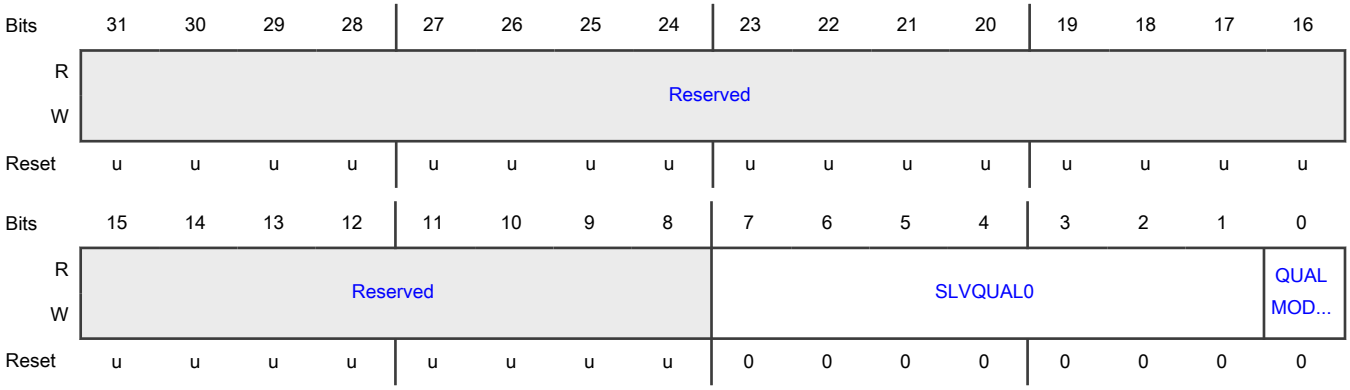
41.5.4.1.1.14 Slave Qualification for Address 0 Register (SLVQUAL0)

The SLVQUAL0 register can alter how Slave Address 0 (specified by the SLVADR0 register) is interpreted.

Offset

Register	Offset
SLVQUAL0	858h

Diagram



Fields

Field	Description
31-8 —	Reserved Read value is undefined; only zero should be written.
7-1 SLVQUAL0	Slave address Qualifier for address 0 SLVQUAL0=0 causes the address in SLVADR0 to be used as-is, assuming that it is enabled. • If QUALMODE0 = 0, then any bit in SLVQUAL0 field that is set to 1 will cause an automatic match of the corresponding bit of the received address when it (the address) is compared to the SLVADR0 register. • If QUALMODE0 = 1, then an address range is matched for address 0. This range extends from the value defined by SLVADR0 to the address defined by SLVQUAL0 (address matches when SLVADR0[7:1] <= received address <= SLVQUAL0[7:1]).
0 QUALMODE0	Qualify mode for slave address 0 0 - Mask. The SLVQUAL0 field is used as a logical mask for matching address 0. 1 - Extend. The SLVQUAL0 field is used to extend address 0 matching in a range of addresses.

41.5.4.1.1.15 Monitor Receiver Data Register (MONRXDAT)

The read-only MONRXDAT register provides information about events on the I2C-bus, primarily to facilitate debugging of the I2C during application development. All data addresses and data passing on the bus and whether these were acknowledged, as well as Start and Stop events, are reported.

The Monitor function must be enabled by the CFG[MONEN] bit. Monitor mode can be configured to stretch the I2C clock if data is not read from the MONRXDAT register in time to prevent it, via the CFG[MONCLKSTR] bit. This can help ensure that nothing is missed, but can cause the Monitor function to be somewhat intrusive (by potentially adding clock delays, depending on software or DMA response time). To improve the chance of collecting all Monitor information if clock stretching is not enabled, Monitor data is buffered such that it (the monitor data) is available until the end of the next piece of information from the I2C-bus.

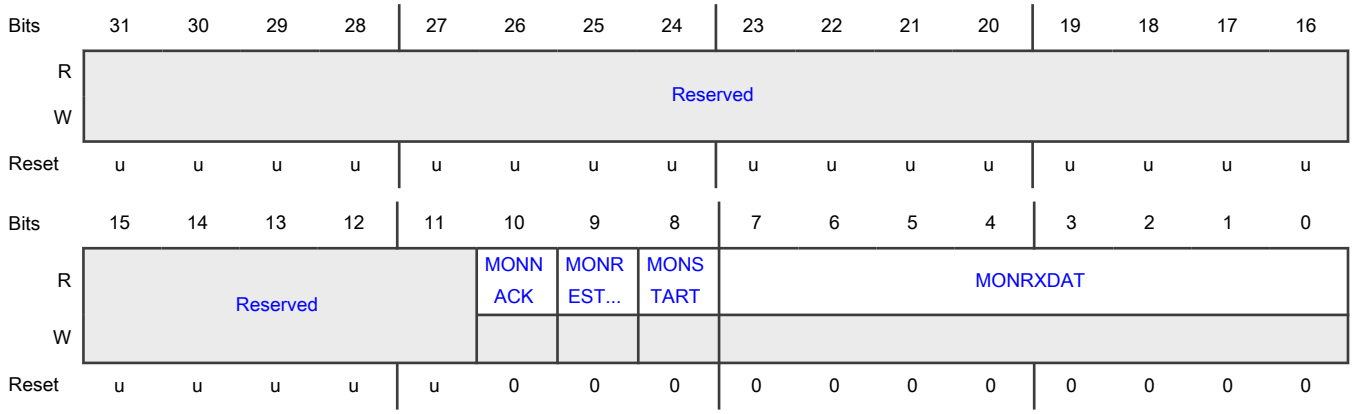
NOTE

Details of clock stretching are different in HS mode.

Offset

Register	Offset
MONRXDAT	880h

Diagram



Fields

Field	Description
31-11 —	Reserved Read value is undefined; only zero should be written.
10 MONNACK	Monitor Received NACK 0 - Acknowledged. The data currently being provided by the Monitor function was acknowledged by at least one master or slave receiver. 1 - Not acknowledged. The data currently being provided by the Monitor function was not acknowledged by any receiver.
9 MONRESTART	Monitor Received Repeated Start 0 - No repeated start detected. The Monitor function has not detected a Repeated Start event on the I2C bus. 1 - Repeated start detected. The Monitor function has detected a Repeated Start event on the I2C bus.
8 MONSTART	Monitor Received Start 0 - No start detected. The Monitor function has not detected a Start event on the I2C bus. 1 - Start detected. The Monitor function has detected a Start event on the I2C bus.
7-0 MONRXDAT	Monitor function Receiver Data This reflects every data byte that passes on the I2C pins.

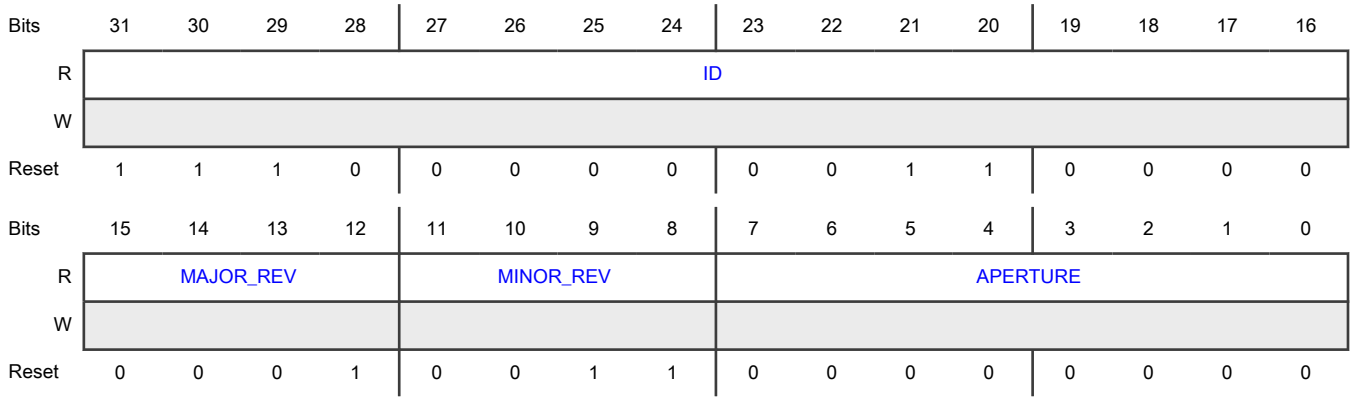
41.5.4.1.1.16 Peripheral Identification Register (ID)

The ID register identifies the type and revision of the I2C module. A generic software driver can make use of this information register, to implement module type or revision-specific behavior.

Offset

Register	Offset
ID	FFCh

Diagram



Fields

Field	Description
31-16 ID	Module identifier for the selected function
15-12 MAJOR_REV	Major revision of module implementation
11-8 MINOR_REV	Minor revision of module implementation
7-0 APERTURE	Aperture aperture size = (APERTURE + 1) x 4K APERTURE encoded as 0x00 means 4K aperture: aperture size = (0x00 + 1) x 4K

41.6 Inter-IC Sound (I2S)

41.6.1 Overview

The I^2S bus provides a standard communication interface for streaming data transfer applications such as digital audio or data collection. The I^2S bus specification defines a 3-wire serial bus, having one data, one clock, and one word select/frame trigger signal, providing single or dual (mono or stereo) audio data transfer.

The I^2S function within one Flexcomm module provides at least one channel pair that can be configured as a master or a slave. Other channel pairs, if present, always operate as slaves. All of the channel pairs within one Flexcomm module share one set of I^2S signals, and are configured together for either transmit or receive operation, using the same mode, same data configuration, and frame configuration. All such channel pairs can participate in a time division multiplexing (TDM) arrangement. For cases requiring an MCLK input and/or output, this is handled outside of the I^2S block in the system level clocking scheme.

41.6.1.1 Block diagram

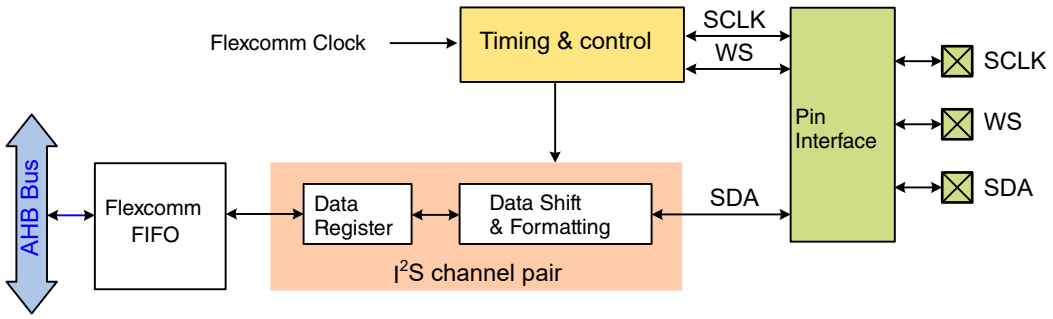


Figure 172. I²S block diagram

41.6.1.2 Features

- A Flexcomm module may implement one or more I²S channel pairs, the first of which could be a master or a slave, and the rest of which would be slaves.
  - All channel pairs are configured together for either transmit or receive and other shared attributes.
  - The number of channel pairs is defined for each Flexcomm module, and may be from 0 to 4.
- Configurable data size for all channels within one Flexcomm module, from 4 bits to 32 bits. Each channel pair can also be configured independently to act as a single channel (mono as opposed to stereo operation).
- Signals configuration:
  - All channel pairs within one Flexcomm module share a single bit clock (SCK) and word select/frame trigger (WS), and data line (SDA).
  - SLK and WS can be selectively inverted.
- Data for all I²S traffic within one Flexcomm module uses the Flexcomm module FIFO. The FIFO depth is 8 entries.
- Left justified and right justified data modes.
- DMA support using FIFO level triggering.
- TDM (Time Division Multiplexing) with a several stereo slots and/or mono slots is supported. Each channel pair can act as any data slot. Multiple channel pairs can participate as different slots on one TDM data line.
- Sampling frequencies depend on the specific device configuration and applications constraints (e.g. system clock frequency, PLL availability, etc.) but generally supports standard audio data rates.

41.6.1.3 Terminology

Table 346. List of some terminology used in I²S

Term	Description
Channel	Essentially, one piece of information on a single SDA line. In classic I²S, there is a single set of stereo data, which is 2 channels (left and right). In TDM modes, there may be many channels on a single SDA line.
Channel Pair	Two channels of data can be carried on one wire in classic I²S: left and right. On a microcontroller, this is typically what is implemented in a single instance of an I²S module.

Table continues on the next page...

Table 346. List of some terminology used in I²S (continued)

Term	Description
Classic I²S	This term refers to the original I²S bus specification from Philips Semiconductors. That specification defines 2 channel stereo data on SDA, where the WS state identifies the left (low) and right (high) channel, and data is delayed by 1 clock after WS transitions. The many variations of I²S that may be found have descended from this original specification.
DSP mode	DSP mode packs channel data together in the bit stream (left data followed by right data for each slot) and does not use WS to identify left and right data. WS may be a single SCK pulse, or a single data slot long pulse, in addition to a 50% duty cycle pulse. DSP mode may be used in conjunction with TDM mode.
MCLK	Master Clock. In some I²S systems, this is provided as a multiple of the sample rate (fs), higher than the bit rate, such as 256 fs. Devices could potentially use this clock to construct a bit clock, or for internal operations such as data filtering.
SCK	Serial Clock. Sometimes referred to as BCK. This is a bit clock for data on the SDA line.
SDA	Serial Data. A single SDA provides one data stream, which may have many formats.
Slot	One data position in an I²S stream, typically each with the same slot length. For classic I²S, there is only one slot for stereo data. In a TDM mode, there can be several slots. In MONO mode, each slot is defined as one piece of data, rather than both left and right data.
TDM mode	TDM mode uses multiple data slots to put more channels of data into a single stream. TDM mode may be used in conjunction with DSP mode or I²S mode.
WS	Word Select. Sometimes called LRCLK. Distinguishes left versus right data in most single stereo formats. Used as a frame delimiter in DSP and TDM modes.

41.6.2 Functional description

41.6.2.1 Formats and modes

The format of data frames and WS is determined by several fields in the CFG1 register and the CFG2 register. CFG1 and CFG2 together control the formatting of the data and the format of the frame which contains the data.

41.6.2.1.1 Frame format

The overall frame format is defined by fields in the CFG1 and CFG2 registers. The frame includes data related to the primary channel pair and any other channel pairs implemented by this I²S module. These fields plus the position of data for each channel pair, as determined by the CFG2[POSITION] field, define the main features of the frame.

- The CFG1[MODE] field defines the overall character of the frame.
- The CFG2[FRAMELEN] field defines the length of the data frame this I²S participates in. This field is minus 1 encoded: the value 63 means 64 clocks and bit positions in each frame.
- The CFG1[DATALEN] field defines the number of data bits that are used by the transmitter or receiver. This field is minus 1 encoded: the value 15 means 16 data bits. For each channel pair, data is only driven to or received from SDA for the number of bits defined by the CFG1[DATALEN] field.

The CFG1[DATALEN] field is also used in these ways:

- Determines the size of data transfers between the FIFO and the I²S serializer/deserializer

- Determines the position of Right data following Left data within the frame, when the CFG1[MODE] field = 0x1, 0x2, or 0x3 (i.e. not 0x0).
- Determines the duration of the WS pulse, when the CFG1[MODE] field = 0x3.

41.6.2.1.2 Example frame configurations

The following figures show a sampling of frame slot formats. These are not all of the possible formats, but shows the various frame formatting concepts. Note that slot identifications are examples only. Data positions are flexible and there are no predefined slots for the hardware.

The first example shows the frame format for the Classic I²S mode. The data region (SDA signal) begins one clock after the leading WS edge for the frame.

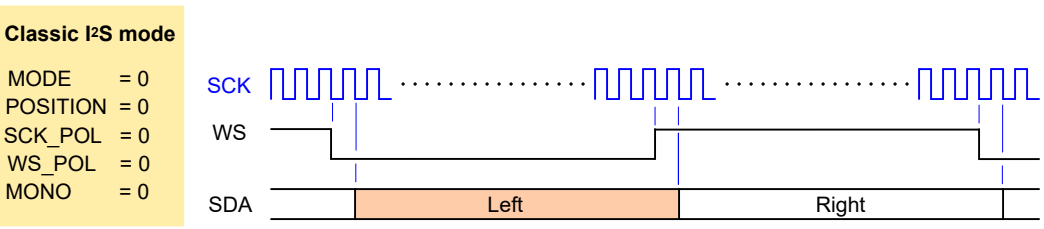


Figure 173. Classic I²S mode

This frame format shows the DSP mode with 50% WS duty cycle and inverted WS (Word Select) polarity.

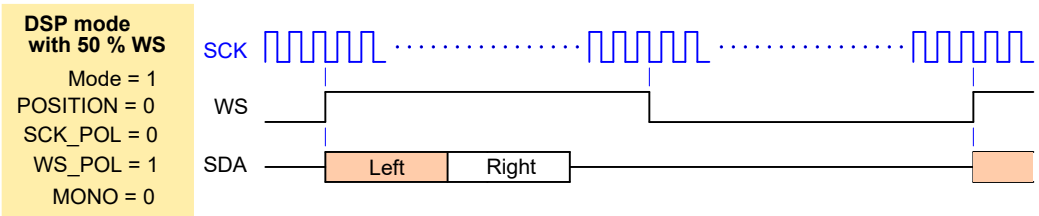


Figure 174. DSP mode with 50% WS

This frame format shows a DSP mode with 1 SCK pulsed WS. This is a DSP mode where WS (Word Select) polarity is inverted and the frame format has a one clock long pulse at the beginning of each data frame.

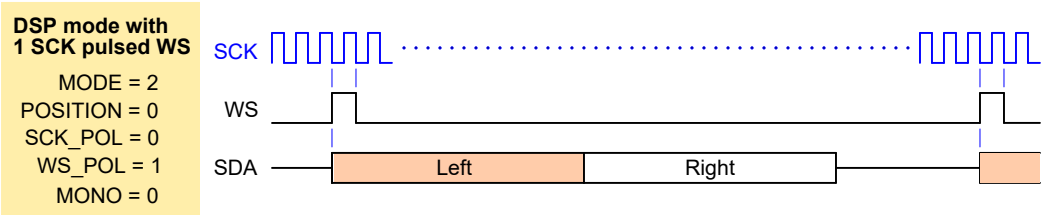


Figure 175. DSP mode with 1 SCK pulsed WS

This frame format shows a DSP mode with 1 slot pulsed WS. This is a DSP mode where WS has a one data slot long pulse at the beginning of each data frame.

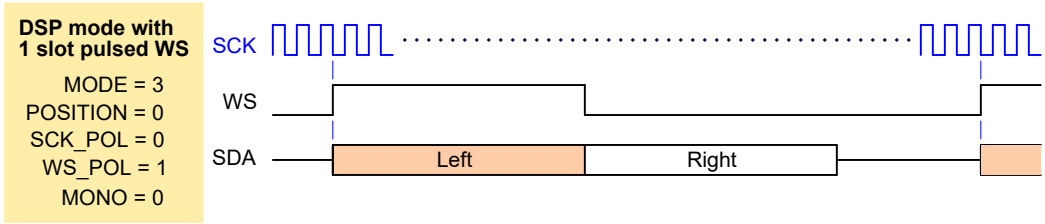


Figure 176. DSP mode with 1 slot pulsed WS

This frame format shows TDM in classic I²S mode.

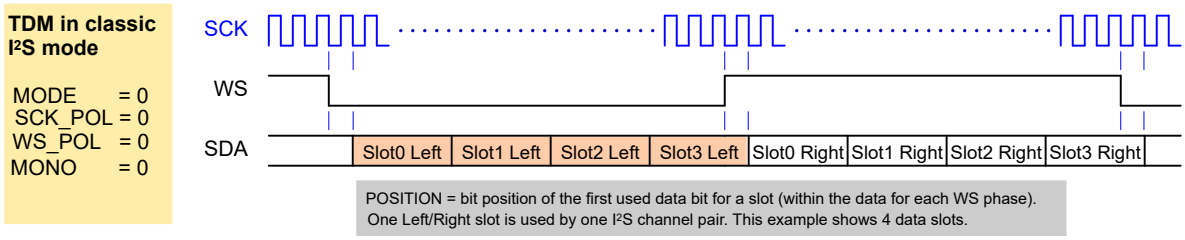


Figure 177. TDM in classic I²S mode

This example shows the frame format for TDM and DSP modes with 50% WS. This is a TDM and DSP mode where WS has a 50% duty cycle and WS (Word Select) polarity is inverted.

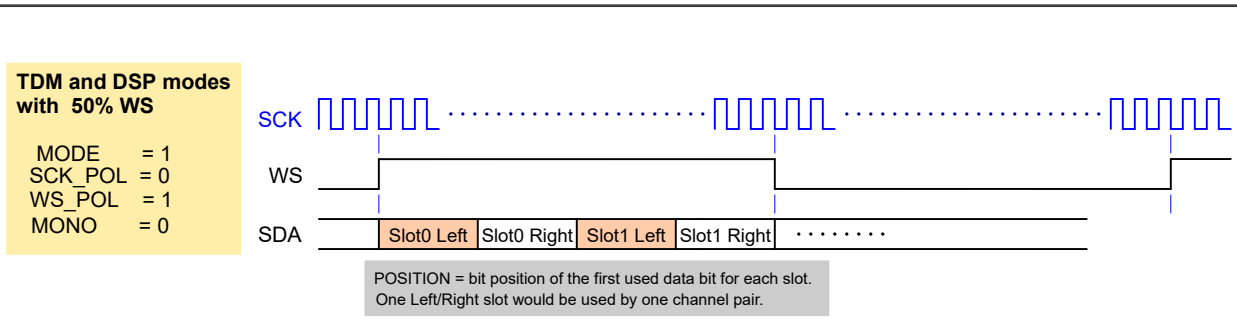


Figure 178. TDM and DSP modes with 50% WS

This frame format shows TDM and DSP modes with 1 SCK pulsed WS. This is a TDM and DSP mode where WS (Word Select) polarity is inverted and the format has a one clock long pulse at the beginning of each data frame.

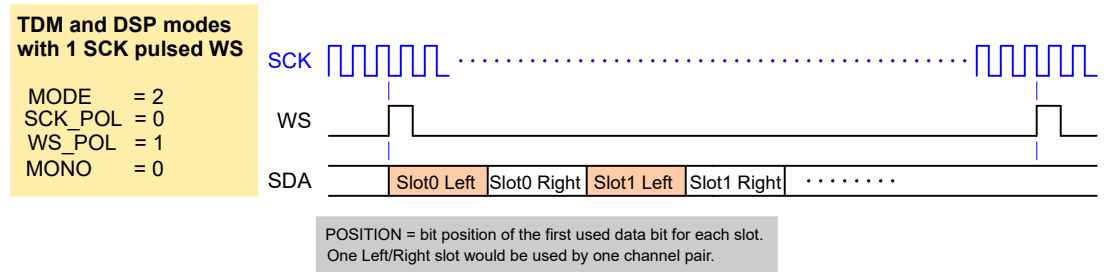


Figure 179. TDM and DSP modes with 1 SCK pulsed WS

This example shows the frame format for TDM and DSP modes with 1 slot pulsed WS. This is a TDM and DSP mode where WS has a one data slot long pulse at the beginning of each data frame. The frame format shows WS (Word Select) polarity is inverted.

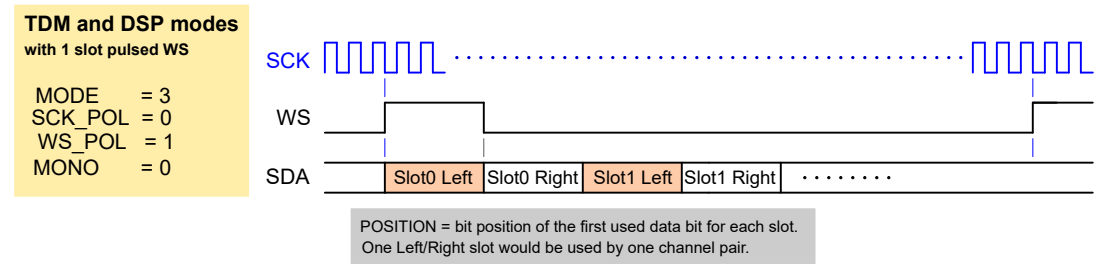


Figure 180. TDM and DSP modes with 1 slot pulsed WS

This example shows a I²S mode with mono or single channel selected.

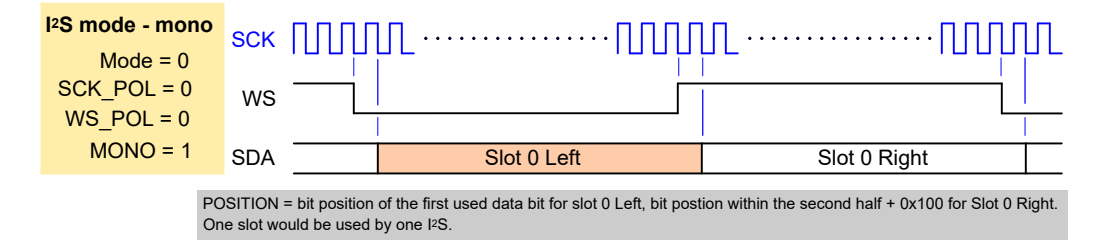


Figure 181. I²S mode, mono

This frame format shows DSP mode with mono or single channel selected. The frame format shows WS (Word Select) polarity is inverted.

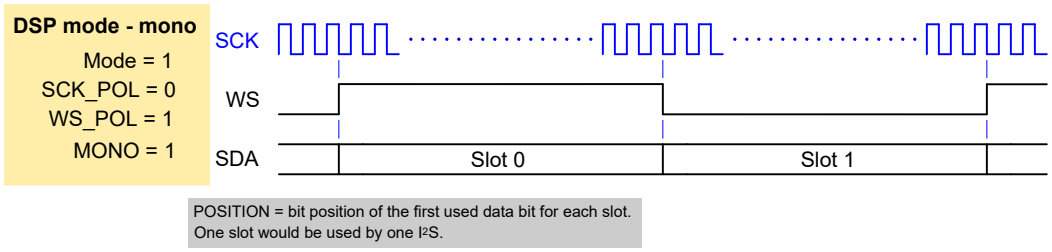


Figure 182. DSP mode, mono

This example shows TDM and DSP modes with mono or single channel and WS pulsed for one SCK time. This is a TDM and DSP mode where WS (Word Select) polarity is inverted and the format has a one clock long pulse at the beginning of each data frame.

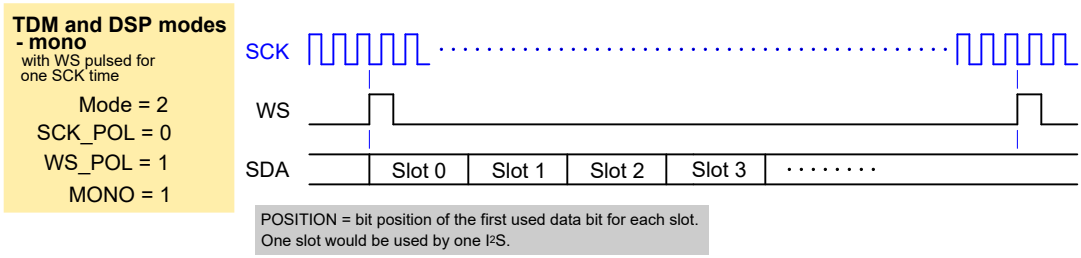


Figure 183. TDM and DSP modes, mono, with WS pulsed for one SCK time

41.6.2.1.3 I²S signal polarities

This diagram shows SCK (Serial Clock) and WS (Word Select) polarities and how they relate to data positions at the start of a frame.

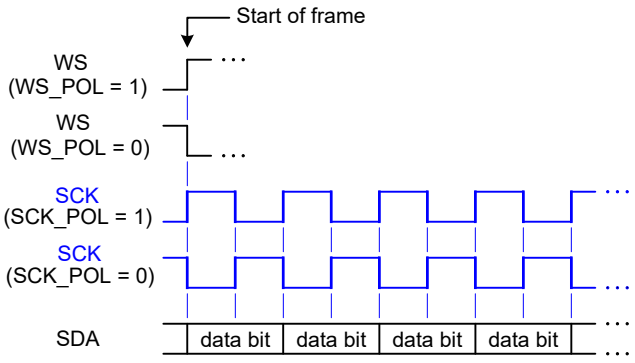


Figure 184. SCK and WS polarities

41.6.2.2 Data rates

41.6.2.2.1 Rate support

The actual I^{2}S clock rates and sample rates that can be supported depend on the clocking. As a slave, the module receives SCK from a master. In that case, there is an upper limit to how fast the module can operate and a limit to how much data can be transferred across clock domains and handled by the CPU.

In general, the I^{2}S can support:

- Standard sample rates such as 16, 22.05, 32, 44.1, 48, and 96 kHz
- External MCLK inputs up to approximately 25 MHz (256 fs of a 96 kHz sample rate) and more.

41.6.2.2.2 Rate calculations

For operation as a master, the frequency need as the clock input of the I^{2}S is generally an integer multiple of:

- Frame/sample rate x number of bits/clocks in a data frame

If this is a multiple of the desired frequency, the I^{2}S function divider can be used to produce the desired frequency.

Example 1

This I^{2}S channel pair transfers stereo audio data with 32 bit data slots and a 96 kHz sample rate.

Setup:

- Sample rate is 96 kHz.
- The frame is configured for two 32-bit data slots (32-bit stereo).
- The function clock divider output rate would be 96,000 x (2 x 32) = 6.144 MHz.

The value in the DIV register would be (function clock divider input frequency / the required divider output frequency) - 1. If the divider input is 24.576 MHz (256 fs of the 96 kHz sample rate), the divider needs to divide by 4 (DIV = 3) to obtain the target divider output rate of 6.144 MHz.

Example 2

This I^{2}S channel pair supplies one 16-bit data slot in a 4 slot frame with a frame rate of 50 kHz.

Setup:

- Sample rate is 50 kHz.
- The frame is configured for four 16-bit data slots.
- The function clock divider output rate would be 50,000 x (4 x 16) = 3.2 MHz.

The value in the DIV register would be (function clock divider input frequency / the required divider output frequency) - 1. If the divider input is 16 MHz, the divider needs to divide by 5 (DIV = 4) to obtain the target divider output rate of 3.2 MHz.

41.6.2.3 FIFO buffer configurations and usage

The Flexcomm module supports several possibilities of data packing/unpacking depending on the size of data being handled.

Some details of FIFO usage are determined by the value of the I^{2}S CFG1[DATALEN] field, and other configuration bits as follows:

- If CFG1[DATALEN] specifies a number of data bits from 4 to 16:
  - The FIFO will be configured as 32 bits wide and 8 entries deep.
  - Each data transfer between the bus and the FIFO will be a pair of left and right values, which fit into a 32-bit word. The order of left and right data is selectable via the CFG1[RIGHTLOW] configuration bit.
  - If a channel pair is configured with CFG1[ONECHANNEL] = 1, then only one value is transferred, nominally the left.
- If CFG1[DATALEN] specifies a number of data bits from 17 to 24:

- The FIFO will be configured as 48 bits wide and 8 entries deep.
- Data transfer between the bus and the FIFO depends the FIFOCFG[PACK48] configuration bit and whether or not DMA is enabled. When DMA is enabled, all transfers are done with the FIFOWR or FIFORD register. When DMA is not enabled, transfers will alternate between the FIFOWR or FIFORD and FIFOWR48H or FIFORD48H registers, depending on the data direction selected for the I^2S function. In all cases, the 2 transfers constitute a pair of left and right values. The order of left and right data is selectable via the CFG1[RIGHTLOW] configuration bit.
- If FIFOCFG[PACK48] = 0, each of the two transfers both define 17 to 24 bits of data. If PACK48 = 1, the first transfer provides 32 bits of data, the second provides the remainder needed to complete the paired data as defined.
- If a channel pair is configured with CFG1[ONECHANNEL] = 1, then only the left value is transferred using the FIFOWR or FIFORD register.

- If CFG1[DATALEN] specifies a number of data bits from 25 to 32:
  - The FIFO will be configured as 32 bits wide and 8 entries deep.
  - Each data transfer between the bus and the FIFO is a single value, starting with left, then right.
  - If a channel pair is configured with CFG1[ONECHANNEL]= 1, then only one value is transferred.

41.6.2.4 DMA

The Flexcomm module can generate DMA requests based on FIFO levels. Data transfers for any channel can be handled by DMA.

To enable DMA data transfers: .

- Configure I^2S clocking
- Configure the channel
- Configure the DMA. To enable the DMA requests, set the bit for the related Flexcomm module instance in the INPUTMUX DMA controller request enable register. For some Flexcomm module instances, a DMA device selection must also be made in the DMA controller channel mux select register.

The I^2S bus must be running to generate DMA requests.

DMA operation is similar to any other serial peripheral.

DMA related configurations in the Flexcomm module I^2S may be found in the FIFO Configuration and Enable (FIFOCFG) register [DMATX, DMARX, WAKETX, WAKERX, and PACK48 bits] and in the FIFO Trigger Settings (FIFOTRIG) register [TXLVLENA, RXLVLENA bits and fields TXLVL and RXLVL].

41.6.2.5 Clocking and power considerations

The master function of the I^2S requires the Flexcomm clock (FCLK) to be running in order to operate. The slave function can operate using external clocks, and can wake up the CPU when data is needed or available.

41.6.3 Signals

This table describes the module signals.

NOTE

When the I^2S function is outputting SCK and/or WS, it uses a return signal from the related pin to adjust internal timing. For that reason, those signals must in fact be connected to a device pin, via IOCON selection, in order for the I^2S to operate.

Table 347. I²S Signals

Signal	I/O	Description
SCK	I/O	Serial Clock for I²S. Clock signal used to synchronize the transfer of data on the SDA signal. It is normally driven by the master and received by one or more slaves. **NOTE** When the primary I²S channel pair of a Flexcomm module is configured as a master, such that SCK is an output, it must actually be connected to a pin for the I²S to work properly.
WS	I/O	Word Select for I²S. Synchronizing signal for the beginning of each data frame and, in some modes, left vs. right channel data. It is normally driven by the master and received by one or more slaves. **NOTE** When the primary I²S channel pair of a Flexcomm module is configured as a master, such that WS is an output, it must actually be connected to a pin for the I²S to work properly.
SDA	I/O	Serial Data for a single data stream used by one or more I²S channel pairs of I²S. The format of data is configurable. It is driven by one or more transmitters and read by one or more receivers.
MCLK	I/O	Master Clock. A multiple of the sample clock can optionally be provided by a master to other devices in the system, or can be received and divided down within a Flexcomm module to locally generate SCK and/or WS. This clock is not created inside the I²S block. If MCLK is supported as an input to the device, it can be routed to the I²S block and used to operate its functions. If MCLK is an output from the device, the clock that is used to create that MCLK can also be routed to the I²S block and used to operate its functions.

41.6.4 Initialization

To initialize the I²S module:

1. Enable the clocks to the I²S registers depending on whether they function as master or slave.
2. Reset the Flexcomm module for the I²S module function.
3. Enable the I²S signals.
4. Enable I²S channel pair interrupts
5. If using DMA, enable the I²S channel pair master and slave functions to work with the system DMA controller.

See the chip-specific I²S information for configuration details.

41.6.4.1 Configure the I²S for wake-up

In Sleep mode, any I²S interrupt can wake up the part, provided that the interrupt is enabled. As long as the I²S clock is configured to be active in Sleep mode, the I²S can wake up the part independently of whether the I²S block is configured in master or slave mode.

In Deep-sleep mode, I²S clocks are normally turned off. However, if the I²S module is configured to use the MCLK input or as a slave receiving an external SCK, the I²S can generate an interrupt and wake up the device.

41.6.4.1.1 Wake-up from Sleep mode

- Configure the I²S for the desired mode.
- Enable the I²S interrupts on the chip.
- Any enabled I²S interrupt wakes up the part from Sleep mode.

41.6.4.1.2 Wake-up from Deep-sleep mode

- Configure the I²S for the desired mode. The selected function clock of the I²S (whether internal or externally sourced) must be running.
- Enable the I²S interrupts on the chip.
- Any enabled I²S interrupt wakes up the part from Deep-sleep mode.

Wake-up for DMA only

The device can optionally be woken up only far enough to perform needed DMA before returning to Deep-sleep mode, without the CPU waking up at all. To accomplish this:

- Configure the I²S for the desired mode. The selected function clock of the I²S (whether internal or externally sourced) must be running.
- Configure the DMA controller appropriately, including a transfer complete interrupt.
- Disable the related I²S interrupt on the chip.
- Enable the DMA interrupt in the chip.
- Enable the wake-up controls on the chip.

41.6.5 Memory map and register definition

This section includes the I²S module memory map and detailed descriptions of all registers.

NOTE

The bus interface to the I²S registers contained in the Flexcomm module support only word writes. Byte and halfword writes are not supported in conjunction with the I²S function.

41.6.5.1 I2S Interface register descriptions

41.6.5.1.1 I2S memory map

FLEXCOMM0.I2S base address: 4008_6000h

FLEXCOMM1.I2S base address: 4008_7000h

FLEXCOMM2.I2S base address: 4008_8000h

FLEXCOMM3.I2S base address: 4008_9000h

FLEXCOMM4.I2S base address: 4008_A000h

FLEXCOMM5.I2S base address: 4009_6000h

FLEXCOMM6.I2S base address: 4009_7000h

FLEXCOMM7.I2S base address: 4009_8000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
C00	Configuration Register 1 for the Primary Channel Pair (CFG1)	32	See section	See section
C04	Configuration Register 2 for the Primary Channel Pair (CFG2)	32	See section	See section

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
C08	Status Register for the Primary Channel Pair (STAT)	32	See section	See section
C1C	Clock Divider (DIV)	32	See section	See section
C20	Configuration Register 1 for Channel Pair 1 (P1CFG1)	32	See section	See section
C24	Configuration Register 2 for Channel Pair 1 (P2CFG1)	32	See section	See section
C28	Status Register for Channel Pair 1 (PSTAT1)	32	See section	See section
C40	Configuration Register 1 for Channel Pair 2 (P1CFG2)	32	See section	See section
C44	Configuration Register 2 for Channel Pair 2 (P2CFG2)	32	See section	See section
C48	Status Register for Channel Pair 2 (PSTAT2)	32	See section	See section
C60	Configuration Register 1 for Channel Pair 3 (P1CFG3)	32	See section	See section
C64	Configuration Register 2 for Channel Pair 3 (P2CFG3)	32	See section	See section
C68	Status Register for Channel Pair 3 (PSTAT3)	32	See section	See section
E00	FIFO Configuration and Enable (FIFOCFG)	32	See section	See section
E04	FIFO Status (FIFOSTAT)	32	See section	See section
E08	FIFO Trigger Settings (FIFOTRIG)	32	See section	See section
E10	FIFO Interrupt Enable Set and Read (FIFOINTENSET)	32	See section	See section
E14	FIFO Interrupt Enable Clear and Read (FIFOINTENCLR)	32	See section	See section
E18	FIFO Interrupt Status (FIFOINTSTAT)	32	See section	See section
E20	FIFO Write Data (FIFOWR)	32	WO	See section
E24	FIFO Write Data for Upper Data Bits (FIFOWR48H)	32	See section	See section

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
E30	FIFO Read Data (FIFORD)	32	RO	See section
E34	FIFO Read Data for Upper Data Bits (FIFORD48H)	32	See section	See section
E40	FIFO Data Read with No FIFO Pop (FIFORDNOPOP)	32	RO	See section
E44	FIFO Data Read for Upper Data Bits with No FIFO Pop (FIFORD48HNOPOP)	32	See section	See section
E48	FIFO Size Register (FIFOSIZE)	32	RO	See section
E4C	FIFO Receive Timeout Configuration (FIFORXTIMEOUTCFG)	32	See section	0000_0000
E50	FIFO Receive Timeout Counter (FIFORXTIMEOUTCNT)	32	See section	0000_0000
FFC	I2S Module Identification (ID)	32	RO	E090_1000

41.6.5.1.1.1 Configuration Register 1 for the Primary Channel Pair (CFG1)

The CFG1 register contains mode settings, most of which apply to all I^2S channel pairs within one Flexcomm module. A few settings apply only to the primary channel pair, as noted.

Offset

Register	Offset
CFG1	C00h

Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved		WS_POL	SCK_POL	PDMDATA	ONECHAN...	LEFTJUST	RIGHTLOW	MODE		MSTSLVCFG		PAIRCOUNT		DATAPAU...	MAINENA...
Reset	u	u	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-21 —	Reserved Read value is undefined, only zero should be written.
20-16 DATALEN	Data Length Data length minus 1 when encoded defines the number of data bits to be transmitted or received for all I2S channel pairs in this Flexcomm module. Note that data is only driven to or received from the SDA signal for the number of bits defined by DATALEN. DATALEN is also used in these ways by the I2S: • Determines the size of data transfers between the FIFO and the I2S serializer/deserializer. See FIFO buffer configurations and usage • In CFG1[MODE] for settings 0x1, 0x2, and 0x3, DATALEN determines the location of right data following left data in the frame. • In CFG1[MODE] = 0x3 (where the WS signal has a one data slot long pulse at the beginning of each data frame), DATALEN determines the duration of the WS pulse. Values: 0x00 to 0x02 are not supported Settings 0x03 - 0x1F represent 4 bits to 32 bits length. 0_0011 - Data is 4 bits in length. 0_0100 - Data is 5 bits in length. 0_0111 - Data is 8 bits in length. 1_1110 - Data is 31 bits in length. 1_1111 - Data is 32 bits in length.
15-14 —	Reserved Read value is undefined, only zero should be written.
13 WS_POL	WS Polarity 0 - Not inverted. Data frames begin at a falling edge of the WS signal (standard for classic I2S). 1 - Inverted. The WS signal is inverted, resulting in a data frame beginning at a rising edge of the WS signal (standard for most 'non-classic' variations of I2S).
12 SCK_POL	SCK Polarity 0 - Falling edge. Data is launched on the SCK signal falling edges and sampled on the SCK signal rising edges (standard for I2S). 1 - Rising edge. Data is launched on the SCK signal rising edges and sampled on the SCK signal falling edges.
11 PDMDATA	PDM Data Selection PDMDATA controls the data source for I2S transmit, and cannot be set in Rx mode. PDMDATA only has an effect if the device includes a DMIC subsystem. See chip-specific information. 0 - Normal Operation. Data is transferred to or from the Flexcomm FIFO.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - DMIC subsystem. The data source is the DMIC subsystem. When PDMDATA = 1, only the primary channel pair can be used in this Flexcomm module. If ONECHANNEL = 1, only the PDM left data is used. the WS signal rate must match the Fs (sample rate) of the DMIC decimator. A rate mismatch will at some point cause the I^2S to overrun or underrun.
10 ONECHANNEL	Single Channel Mode Applies to both transmit and receive. ONECHANNEL applies only to the first I^2S channel pair. Other channel pairs may select this mode independently in their separate CFG1 registers. **NOTE** In mode 0 only, the right side of the frame begins at POSITION = 0x400 (the number is calculated from Max Frame length (2048/2). This is because mode 0 makes a clear distinction between the left and right sides of the frame. When ONECHANNEL = 1, the single channel of data may be placed on the right by setting POSITION to 0x100 + the data position within the right side (e.g. 0x108 would place data starting at the 8th clock after the middle of the frame). In other modes, data for the single channel of data is placed at the clock defined by POSITION. 0 - Dual channel. I^2S data for this channel pair is treated as left and right channels. 1 - Single channel. I^2S data for this channel pair is treated as a single channel, functionally the left channel for this pair.
9 LEFTJUST	Left-Justify Data 0 - Right-justified. Data is transferred between the FIFO and the I^2S serializer/deserializer right justified, i.e. starting from bit 0 and continuing to the position defined by DATALEN. This corresponds to right-justified data in the stream on the data bus. 1 - Left-justified. Data is transferred between the FIFO and the I^2S serializer/deserializer left justified, i.e., starting from the MSB of the FIFO entry and continuing for the number of bits defined by DATALEN. This corresponds to left-justified data in the stream on the data bus.
8 RIGHTLOW	Right Channel Low Right channel data is in the low portion of FIFO data. Essentially, this swaps left and right channel data as it is transferred to or from the FIFO. This bit is not used if the data width is greater than 24 bits or if PDMDATA = 1. Note that if ONECHANNEL = 1, then the one channel to be used is the left channel. CFG2[POSITION] can still place that data in the frame where right channel data is normally located. **NOTE** If all enabled channel pairs have ONECHANNEL = 1, then RIGHTLOW = 1 is not allowed. 0 - Right high. The right channel is taken from the high part of the FIFO data. For example, when data is 16 bits, FIFO bits 31:16 are used for the right channel. 1 - Right low. The right channel is taken from the low part of the FIFO data. For example, when data is 16 bits, FIFO bits 15:0 are used for the right channel.
7-6 MODE	Mode Selects the basic I^2S operating mode. Other configurations modify this setting to obtain all supported cases. See Formats and modes for examples.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	NOTE For a 50% WS duty cycle, FRAMELEN must define an even number of I^2S clocks for the frame. If FRAMELEN defines an odd number of clocks per frame, the extra clock occurs on the right. 00 - Classic Mode. The I^2S mode where in each enabled channel pair the WS signal has a 50% duty cycle, with one piece of left channel data occurring during the first phase, and one piece of right channel data occurring during the second phase. In this mode, the data region begins one clock after the leading WS signal edge for the frame. See note for Mode with 50% duty cycle. 0. 01 - DSP mode WS 50% duty cycle. DSP mode where the WS signal has a 50% duty cycle. See note for Mode with 50% duty cycle. 0. 10 - DSP mode WS 1 clock. DSP mode where the WS signal has a one clock long pulse at the beginning of each data frame. 11 - DSP mode WS 1 data. DSP mode where the WS signal has a one data slot long pulse at the beginning of each data frame.
5-4 MSTSLVCFG	Master/Slave Configuration Selection MSTSLVCFG determines how the SCK and WS signals are used by all channel pairs in this Flexcomm module. 00 - Normal Slave Mode. Default mode. The SCK and WS signals are received from a master and used to transmit or receive data. 01 - WS Synchronized Master Mode. WS is received from another master and used to synchronize the generation of the SCK signal, when divided from the Flexcomm clock (FCLK). 10 - Master Using an Existing SCK Mode. The SCK signal is received and used directly to generate the WS signal, as well as transmitting or receiving data. 11 - Normal Master Mode. The SCK and WS signals are generated so they can be sent to one or more slave devices.
3-2 PAIRCOUNT	Pair Count Provides the number of I^2S channel pairs in this Flexcomm module. This is a read-only field whose value may be different in other Flexcomm modules. 00 - One Pair. 1 I^2S channel pairs in this Flexcomm module 01 - Two Pairs. 2 I^2S channel pairs in this Flexcomm module 10 - Three Pairs. 3 I^2S channel pairs in this Flexcomm module 11 - Four Pairs. 4 I^2S channel pairs in this Flexcomm module
1 DATAPAUSE	Data Flow Pause Allows pausing data flow between the I^2S serializer/deserializer and the FIFO. This could be done in order to change streams, or while restarting after a data underflow or overflow. When paused, FIFO operations can be done without corrupting data that is in the process of being sent or received.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	Once a data pause has been requested, the module may need to complete sending data that was in progress before interrupting the flow of data. Software must check that the pause is actually in effect before taking action. The check is done by monitoring the STAT[DATAPAUSED] flag. When DATAPAUSE is cleared, data transfer resumes at the beginning of the next frame. 0 - Normal operation. Normal operation, or resuming normal operation at the next frame if the I^2S has already been paused. 1 - Pause. A pause in the data flow is being requested. The data pause is in effect when STAT[DATAPAUSED] = 1.
0 MAINENABLE	Main Enable Main enable for I^2S function in this Flexcomm 0 - Disabled. All I^2S channel pairs in this Flexcomm are disabled and the internal state machines, counters, and flags are reset. No other channel pairs can be enabled. 1 - Enabled. This I^2S channel pair is enabled. Other channel pairs in this Flexcomm may be enabled in their individual PAIRENABLE bits.

41.6.5.1.1.2 Configuration Register 2 for the Primary Channel Pair (CFG2)

The CFG2 register controls data configuration.

Offset

Register	Offset
CFG2	C04h

Diagram

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

Fields

Field	Description
31-27 —	Reserved Read value is undefined, only zero should be written.
26-16 POSITION	Data Position Defines the location within the frame of the data for this channel pair. POSITION + CFG1[DATALEN] must be less than FRAMELEN. See Frame format. When CFG1[MODE] = 0x0, POSITION defines the location of data in both the left phase and right phase, starting one clock after the WS edge. In other modes, POSITION defines the location of data within the entire frame. CFG1[ONECHANNEL] = 1 while CFG1[MODE] = 0x0 is a special case, see the description of ONECHANNEL. The combination of CFG1[DATALEN] and the POSITION fields of all channel pairs must be made such that the channels do not overlap within the frame. 000_0000_0000 - Data begins at bit position 0 (the first bit position) within the frame or WS phase 000_0000_0001 - Data begins at bit position 1 within the frame or WS phase 000_0000_0010 - Data begins at bit position 2 within the frame or WS phase
15-11 —	Reserved Read value is undefined, only zero should be written.
10-0 FRAMELEN	Frame Length Frame length minus 1 when encoded defines the number of clocks and data bits in the frames that this channel pair participates in. See Frame format 0x000 to 0x002 are not supported **NOTE** If FRAMELEN defines an odd length frame (for example 33 clocks) in CFG1[MODE] = 0x0 or 0x1, the extra clock appears in the right half. **NOTE** When CFG1[MODE] = 0x3, FRAMELEN must be larger than DATALEN for the WS pulse to generate correctly. 000_0000_0011 - Frame is 4 bits in total length 000_0000_0100 - Frame is 5 bits in total length 001_1111_1111 - Frame is 512 bits in total length 111_1111_1111 - Frame is 2048 bits in total length

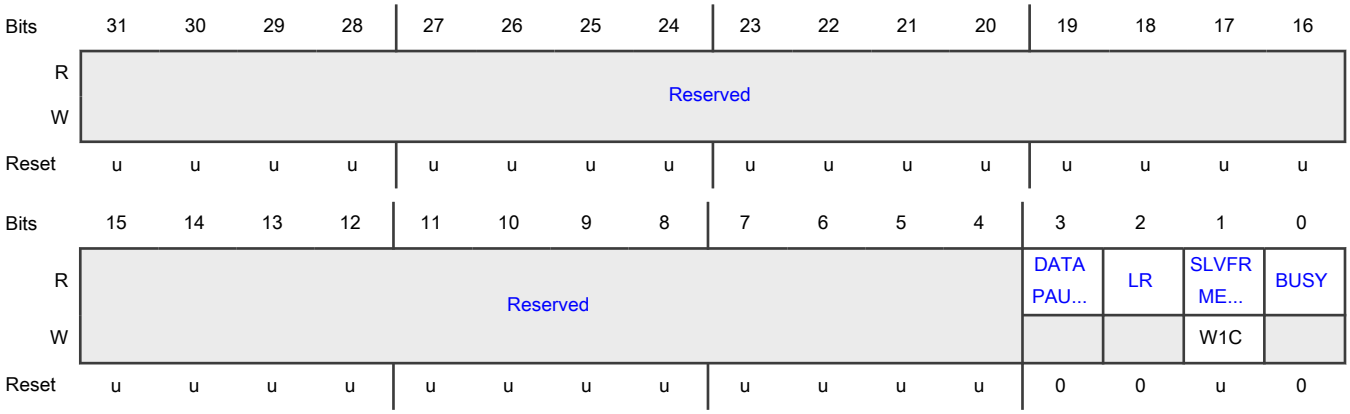
41.6.5.1.1.3 Status Register for the Primary Channel Pair (STAT)

The STAT register provides status flags for the I^2S function, and does not include FIFO status. Note that the FIFO status register supplies peripheral interrupt notification and would be the status register normally observed first for an interrupt service. Some information in this register is read-only; some flags can be cleared by writing a 1 to them.

Offset

Register	Offset
STAT	C08h

Diagram



Fields

Field	Description
31-4 —	Reserved Read value is undefined, only zero should be written.
3 DATAPAUSED	Data Paused The DATAPAUSED status flag applies to all I^2S channels. 0 - Not Paused. Data is not currently paused. A data pause may have been requested but is not yet in force, waiting for an allowed pause point. Refer to the description of the CFG1[DATAPAUSE] control bit. 1 - Paused. A data pause has been requested and is now in force.
2 LR	Left/Right Indication This LR flag is considered to be a debugging aid and is not expected to be used by an I^2S driver. Valid when one channel pair is busy. Indicates left or right data being processed for the currently busy channel pair. 0 - Left channel 1 - Right channel
1 SLVFRMERR	Slave Frame Error The SLVFRMERR flag applies when at least one channel pair is operating as a slave. An error indicates that the incoming WS signal did not transition as expected due to a mismatch between CFG2[FRAMELEN] and the actual incoming I^2S stream. 0 - No error. No error has been recorded.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Error. An error has been recorded for some channel pair that is operating in slave mode. ERROR is cleared by writing a 1 to this bit position.
0 BUSY	Busy Status Busy status for the primary channel pair Other BUSY flags may be found in the STAT register for each channel pair. 0 - Idle. The transmitter/receiver for channel pair is currently idle. 1 - Busy. The transmitter/receiver for channel pair is currently processing data.

41.6.5.1.1.4 Clock Divider (DIV)

The DIV register, used by all channel pairs, controls how the Flexcomm clock (FLCK) is used. See Data rates for more details.

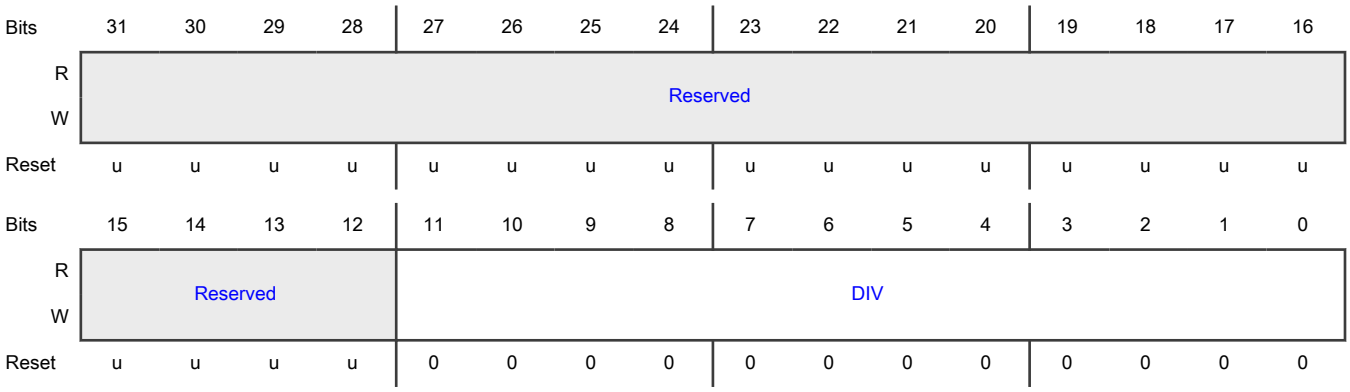
NOTE

DIV must be set to 0 if the SCK signal is used as an input clock for the I^2S function, which is the case when CFG1[MSTSLVCFG] is 0x1 or 0x2.

Offset

Register	Offset
DIV	C1Ch

Diagram



Fields

Field	Description
31-12 —	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
11-0 DIV	Divider DIV controls how this I2S block uses the Flexcomm clock (FLCK). The I2S uses FCLK/(DIV + 1). 0000_0000_0000 - FCLK is used directly. 0000_0000_0001 - FCLK is divided by 2. 0000_0000_0010 - FCLK is divided by 3. 1111_1111_1111 - FCLK is divided by 4,096.

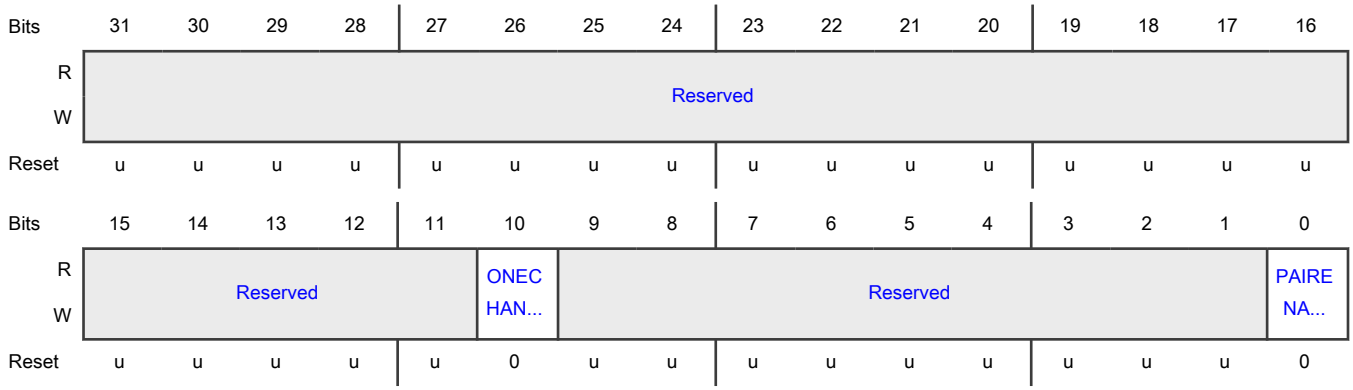
41.6.5.1.1.5 Configuration Register 1 for Channel Pair a (P1CFG1 - P1CFG3)

The P1CFG1, P2CFG1, and P3CFG1 registers contain mode settings that apply to channel pairs other than the first pair, within the same Flexcomm module.

Offset

Register	Offset
P1CFG1	C20h
P1CFG2	C40h
P1CFG3	C60h

Diagram



Fields

Field	Description
31-11 —	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
10 ONECHANNEL	Single Channel Mode 0 - Dual Channel. I2S data for this channel pair is treated as left and right channels. 1 - Single Channel. I2S data for this channel pair is treated as a single channel, functionally the left channel for this pair.
9-1 —	Reserved Read value is undefined, only zero should be written.
0 PAIRENABLE	Pair Enable Enable for this channel pair 0 - Disabled. This I2S channel pair is disabled. 1 - Enabled. This I2S channel pair is enabled. Other channel pairs in this Flexcomm module may be enabled in their individual PAIRENABLE bits.

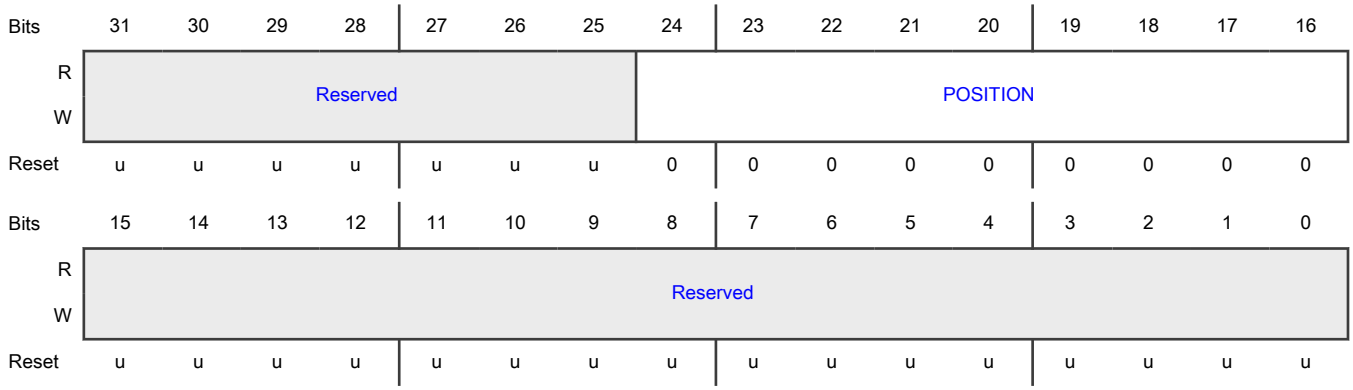
41.6.5.1.1.6 Configuration Register 2 for Channel Pair a (P2CFG1 - P2CFG3)

These registers contain the frame position for channel pairs beyond the main pair.

Offset

Register	Offset
P2CFG1	C24h
P2CFG2	C44h
P2CFG3	C64h

Diagram



Fields

Field	Description
31-25 —	Reserved Read value is undefined, only zero should be written.
24-16 POSITION	Data Position Defines the location within the frame of the data for this channel pair. See details in the description of POSITION for the primary channel pair.
15-0 —	Reserved Read value is undefined, only zero should be written.

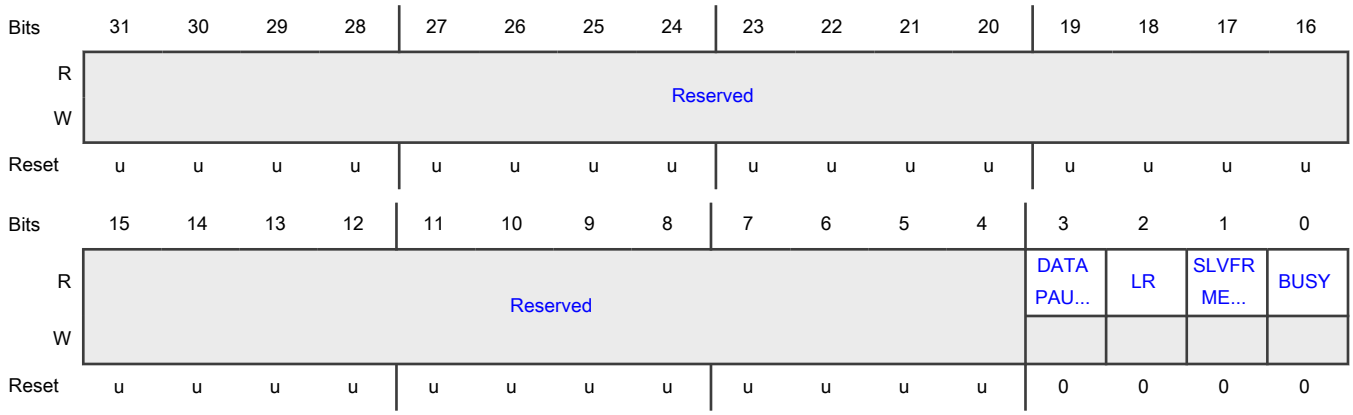
41.6.5.1.1.7 Status Register for Channel Pair a (PSTAT1 - PSTAT3)

These read-only registers provide status flags for additional channel pairs beyond the primary channel pair.

Offset

Register	Offset
PSTAT1	C28h
PSTAT2	C48h
PSTAT3	C68h

Diagram



Fields

Field	Description
31-4 —	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
3 DATAPAUSED	Data Paused Status Flag This flag is shared by the STAT and PnSTAT registers. 0 - Data Not Paused. Data is not currently paused. A data pause may have been requested but is not yet in force, waiting for an allowed pause point. Refer to the description in CFG1[DATAPAUSE]. 1 - Data Paused. A data pause has been requested and is now in force.
2 LR	Left/Right Indication This flag is shared by the STAT and PnSTAT registers. Refer to the STAT register description for details. 0 - Left channel 1 - Right channel
1 SLVFRMERR	Save Frame Error Flag This flag is shared by the STAT and PnSTAT registers. Refer to the STAT register description for details. 0 - No Error. No error has been recorded. 1 - Error. An error has been recorded for some channel pair that is operating in slave mode. ERROR is cleared by writing a 1 to this bit position.
0 BUSY	Busy Status for Channel Pair 0 - Idle. The transmitter/receiver for this channel pair is currently idle. 1 - Busy. The transmitter/receiver for this channel pair is currently processing data.

41.6.5.1.1.8 FIFO Configuration and Enable (FIFOCFG)

FIFOCFG configures FIFO use. A peripheral must be selected within the Flexcomm module prior to configuring the FIFO.

NOTE

Because all I^2S channels in a single Flexcomm module move data in the same direction, only TX-related or RX-related flags and controls are meaningful at any particular time.

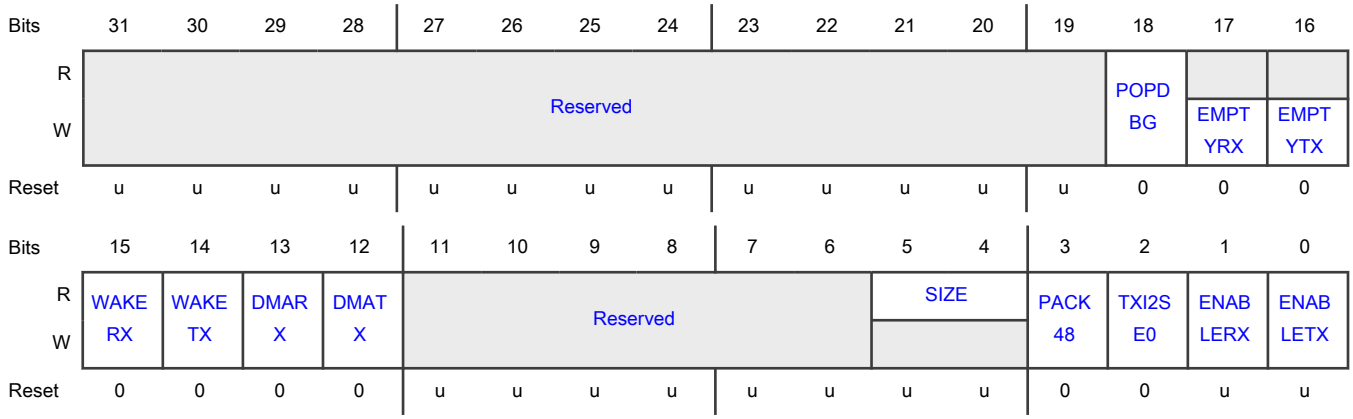
NOTE

The FIFO for the selected I^2S data direction must be enabled because the FIFO is the only means for accessing I^2S data.

Offset

Register	Offset
FIFOCFG	E00h

Diagram



Fields

Field	Description
31-19 —	Reserved Read value is undefined, only zero should be written.
18 POPDBG	Pop FIFO for Debug Reads 0 - Debug reads of the FIFO do not pop the FIFO. 1 - A debug read causes the FIFO to pop.
17 EMPTYRX	Empty command for the receive FIFO. When a 1 is written to this bit, the RX FIFO is emptied.
16 EMPTYTX	Empty command for the transmit FIFO. When a 1 is written to this bit, the TX FIFO is emptied.
15 WAKERX	Wake-up for Receive FIFO Level WAKERX allows the device to be woken from reduced power modes (as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU remains stopped until woken by another cause, such as DMA completion. The feature is enabled by the Hardware Wake-Up Control register in the chip if the feature is supported. 0 - Only enabled interrupts wake up the device from reduced power modes. 1 - A device wake-up for DMA occurs if the receive FIFO level reaches the value specified by FIFOTRIG[RXLVL], even when the RXLVL interrupt is not enabled.
14 WAKETX	Wake-up for Transmit FIFO Level WAKETX allows the device to be woken from reduced power modes (as long as the peripheral function works in that power mode) without enabling the TXLVL interrupt. Only DMA wakes up, processes data, and goes back to sleep. The CPU remains stopped until woken by another cause, such as DMA completion. The feature is enabled by the Hardware Wake-Up Control register in the chip if the feature is supported.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - Disabled. Only enabled interrupts wake up the device from reduced power modes. 1 - Enabled. A device wake up for DMA occurs if the transmit FIFO level reaches the value specified by FIFOTRIG[TXLVL], even when the TXLVL interrupt is not enabled.
13 DMARX	DMA Receive DMA configuration for receive 0 - Disabled. DMA is not used for the receive function. 1 - Enabled. Trigger DMA for the receive function if the FIFO is not empty. Generally, data interrupts would be disabled if DMA is enabled.
12 DMATX	DMA Transmit DMA configuration for transmit 0 - Disabled. DMA is not used for the transmit function. 1 - Enabled. Trigger DMA for the transmit function if the FIFO is not full. Generally, data interrupts would be disabled if DMA is enabled.
11-6 —	Reserved Read value is undefined, only zero should be written.
5-4 SIZE	FIFO Size Configuration This is a read-only field. 0x0, 0x1 are not applicable to I^2S. 10 - Size 32 Bits. FIFO is configured as 8 entries of 32 bits, each corresponding to 2 16-bit data values for left and right channels. This setting occurs when the I^2S CFG1[DATALEN] value is less than 16 bits, or from 25 to 32 bits. 11 - Size 48 Bits. FIFO is configured as 8 entries of 48 bits, each corresponding to 2 24-bit (?) data values for left and right channels. This setting occurs when the I^2S CFG1[DATALEN] value is from 17 to 24 bits.
3 PACK48	Packing Format 48-bit data PACK48 relates to how data is entered into or taken from the FIFO by software or DMA. 0 - Bits_24. 48-bit I^2S FIFO entries are handled as all 24-bit values. 1 - Bits_32_16. 48-bit I^2S FIFO entries are handled as alternating 32-bit and 16-bit values.
2 TXI2SE0	Transmit I2S Empty 0 Transmit I^2S Empty 0 Determines the value sent by the I^2S in transmit mode if the transmit FIFO becomes empty. This value is sent repeatedly until the I^2S is paused, the error is cleared, new data is provided, and the I^2S is un-paused. 0 - Last value. If the TX FIFO becomes empty, the last value is sent. This setting may be used when the data length is 24 bits or less, or when MONO = 1 for this channel pair.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Zero. If the TX FIFO becomes empty, 0 is sent. Use if the data length is greater than 24 bits or if zero fill is preferred.
1 ENABLERX	Enable Receive FIFO 0 - Disabled. The receive FIFO is not enabled. 1 - Enabled. The receive FIFO is enabled.
0 ENABLETX	Enable Transmit FIFO 0 - Disabled Transmit. The transmit FIFO is not enabled. 1 - Enabled transmit. The transmit FIFO is enabled.

41.6.5.1.1.9 FIFO Status (FIFOSTAT)

FIFOSTAT provides status information for the FIFO and also indicates an interrupt from the peripheral function.

NOTE

Because all I^2S channels in a single Flexcomm module move data in the same direction, only TX-related or RX-related flags and controls are meaningful at any particular time.

Offset

Register	Offset
FIFOSTAT	E04h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved							RXTIMEO...	Reserved			RXLVL				
W								W1C								
Reset	u	u	u	u	u	u	u	u	u	u	u	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved			TXLVL					RXFULL	RXNOTEM...	TXNOTFU...	TXEMPTY	PERINT	Reserved	RXERR	TXERR
W															W1C	W1C
Reset	u	u	u	0	0	0	0	0	0	0	1	1	0	u	0	0

Fields

Field	Description
31-25 —	Reserved Read value is undefined; only zero should be written.
24 RXTIMEOUT	Receive FIFO Timeout Writing 1 clears this bit and lowers the corresponding interrupt. 0 - RX FIFO on 1 - RX FIFO has timed out, based on the timeout configuration in the FIFORXTIMEOUTCFG register.
23-21 —	Reserved Read value is undefined; only zero should be written.
20-16 RXLVL	Receive FIFO Current Level A 0 means the RX FIFO is currently empty, and the RXFULL and RXNOTEMPTY flags are 0. Other values tell how much data is in the RX FIFO at the point where the read occurs. If the RX FIFO is full, then the RXFULL and RXNOTEMPTY flags are 1. 0_0000 - RX FIFO is empty
15-13 —	Reserved Read value is undefined, only zero should be written.
12-8 TXLVL	Transmit FIFO Current Level A 0 means the transmit FIFO is currently empty, and the TXEMPTY and TXNOTFULL flags are 1. Other values tell how much data is in the TX FIFO at the point where the read occurs. If the TX FIFO is full, then the TXEMPTY and TXNOTFULL flags are 0. 0_0000 - TX FIFO is empty
7 RXFULL	Receive FIFO Full When 1, the receive FIFO is full. To prevent the peripheral from causing an overflow in the FIFO, data must be read out of the FIFO. 0 - Receive FIFO is not full 1 - Receive FIFO is full
6 RXNOTEMPTY	Receive FIFO Not Empty 0 - Receive FIFO is empty 1 - Receive FIFO is not empty, so data can be read.
5 TXNOTFULL	Transmit FIFO Not Full 0 - Transmit FIFO is full, and another write would cause an overflow 1 - Transmit FIFO is not full, so more data can be written
4	Transmit FIFO Empty 0 - Transmit FIFO is not empty

Table continues on the next page...

Table continued from the previous page...

Field	Description
TXEMPTY	1 - Transmit FIFO is empty; however, the peripheral may still be processing the last piece of data.
3 PERINT	Peripheral Interrupt When 1, this indicates that the peripheral function has asserted an interrupt. The details can be found by reading the peripheral's STAT register. 0 - No interrupt 1 - Interrupt
2 —	Reserved Read value is undefined, only zero should be written.
1 RXERR	RX FIFO Error Set if a receive FIFO overflow occurs, caused by software or DMA not emptying the FIFO fast enough. Cleared by writing a 1 to this bit. 0 - No receive FIFO error occurred 1 - Receive FIFO error occurred
0 TXERR	TX FIFO Error Set if a transmit FIFO error occurs. This could be an overflow caused by pushing data into a full FIFO, or by an underflow if the FIFO is empty when data is needed. Cleared by writing a 1 to this bit. 0 - No transmit FIFO error occurred 1 - Transmit FIFO error occurred

41.6.5.1.1.10 FIFO Trigger Settings (FIFOTRIG)

The FIFOTRIG register allows selecting the trigger for FIFO-level related interrupts and DMA requests.

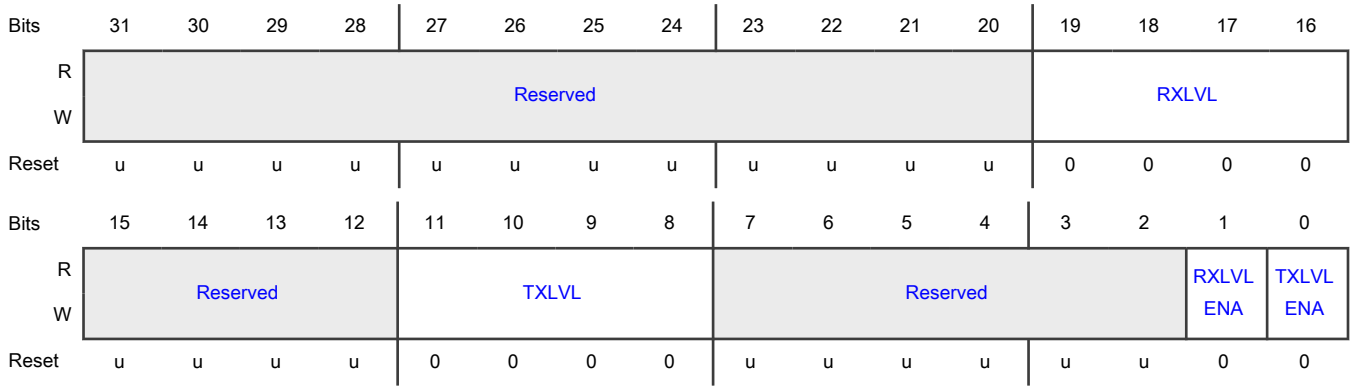
NOTE

Since all I^2S channels in a single Flexcomm module move data in the same direction, only TX-related or RX-related flags and controls are meaningful at any particular time.

Offset

Register	Offset
FIFOTRIG	E08h

Diagram



Fields

Field	Description
31-20 —	Reserved Read value is undefined, only zero should be written.
19-16 RXLVL	Receive FIFO Level Trigger Point The RX FIFO level is checked when a new piece of data is received. The RXLVL field is used only when RXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode. The RXLVL feature is enabled by the Hardware Wake Up Control register (if the chip supports it). **NOTE** FIFOCFG[WAKERX] allows the device to be woken from reduced power modes. 0000 - Trigger when the RX FIFO has received 1 entry (the FIFO is no longer empty). 0001 - Trigger when the RX FIFO has received 2 entries. 1111 - Trigger when the RX FIFO has received 16 entries (the FIFO has become full).
15-12 —	Reserved Read value is undefined, only zero should be written.
11-8 TXLVL	Transmit FIFO Level Trigger Point TXLVL is used only when TXLVLENA = 1. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode. The feature is enabled by the Hardware Wake Up Control register in the chip if the feature is supported **NOTE** FIFOCFG[WAKETX] allows the device to be woken from reduced power modes. 0000 - Trigger when the TX FIFO becomes empty. 0001 - Trigger when the TX FIFO level decreases to one entry. 1111 - Trigger when the TX FIFO level decreases to 15 entries (is no longer full).

Table continues on the next page...

Table continued from the previous page...

Field	Description
7-2 —	Reserved Read value is undefined, only zero should be written.
1 RXLVLENA	Receive FIFO Level Trigger Enable This trigger becomes an interrupt if enabled in FIFOINTENSET[RXLVL], or a DMA trigger if FIFOCFG[DMARX] = 1. 0 - Receive FIFO level does not generate a FIFO level trigger. 1 - An trigger generates if the receive FIFO level reaches the value specified by the RXLVL.
0 TXLVLENA	Transmit FIFO Level Trigger Enable This trigger becomes an interrupt if enabled in FIFOINTENSET[TXLVL], or a DMA trigger if FIFOCFG[DMATX] = 1. 0 - Transmit FIFO level does not generate a FIFO level trigger. 1 - An trigger generates if the transmit FIFO level reaches the value specified by the TXLVL field in this register.

41.6.5.1.1.11 FIFO Interrupt Enable Set and Read (FIFOINTENSET)

The FIFOINTENSET register enables various interrupt sources. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in the FIFOINTENSET register causes those bits to be set. The FIFOINTENCLR register can clear bits in this FIFOINTENSET register.

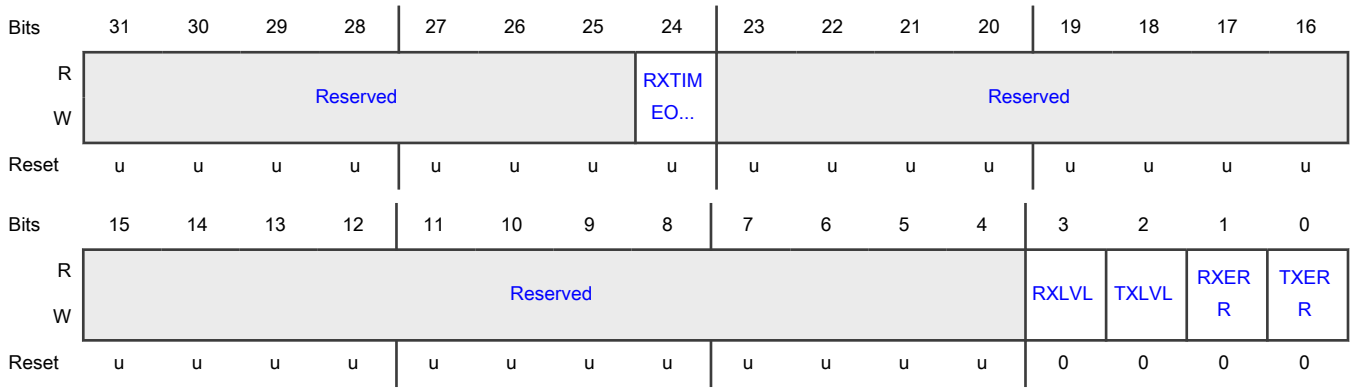
NOTE

Since all I^2S channels in a single Flexcomm module move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time.

Offset

Register	Offset
FIFOINTENSET	E10h

Diagram



Fields

Field	Description
31-25 —	Reserved Read value is undefined; only zero should be written.
24 RXTIMEOUT	Receive Timeout The interrupt cause can be cleared by writing 1 to the FIFOSTAT[RXTIMEOUT] bit. 0 - No RX interrupt will be generated. 1 - Asserts RX interrupt if RX FIFO Timeout event occurs.
23-4 —	Reserved Read value is undefined; only zero should be written.
3 RXLVL	Receive Level Interrupt Determines whether an interrupt occurs when a the receive FIFO reaches the level specified by the FIFOTRIG[RXLVL] flag. 0 - Disabled. No interrupt generates based on the RX FIFO level. 1 - Enabled. If FIFOTRIG[RXLVLENA] = 1, an interrupt generates when the RX FIFO level increases to the level specified by FIFOTRIG[RXLVL].
2 TXLVL	Transmit Level Interrupt Determines whether an interrupt occurs when a the transmit FIFO reaches the level specified by the FIFOTRIG[TXLVL] flag. 0 - Disabled. No interrupt generates based on the TX FIFO level. 1 - Enabled. If FIFOTRIG[TXLVLENA] = 1, an interrupt generates when the TX FIFO level decreases to the level specified by the FIFOTRIG[TXLVL] flag.
1 RXERR	Receive Error Interrupt Determines whether an interrupt occurs when a receive error occurs, based on the FIFOSTAT[RXERR] flag. 0 - Disabled. No interrupt generates for a receive error. 1 - Enabled. An interrupt generates when a receive error occurs.
0 TXERR	Transmit Error Interrupt Determines whether an interrupt occurs when a transmit error occurs, based on the FIFOSTAT[TXERR] flag. 0 - Disabled. No interrupt generates for a transmit error. 1 - Enabled. An interrupt generates when a transmit error occurs.

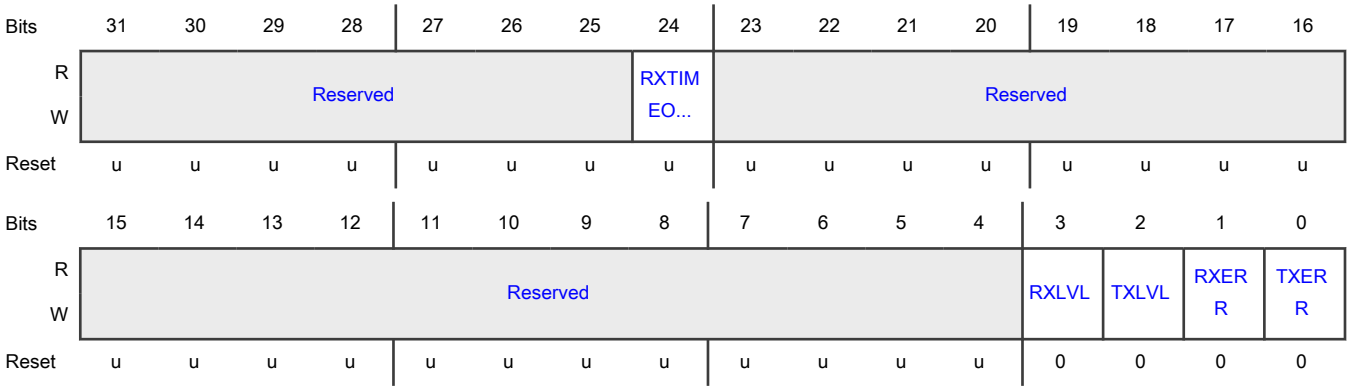
41.6.5.1.1.12 FIFO Interrupt Enable Clear and Read (FIFOINTENCLR)

The FIFOINTENCLR register clears interrupt enable bits in FIFOINTENSET. The complete set of interrupt enables may also be read from this register as well as FIFOINTENSET.

Offset

Register	Offset
FIFOINTENCLR	E14h

Diagram



Fields

Field	Description
31-25 —	Reserved Read value is undefined; only zero should be written.
24 RXTIMEOUT	Receive Timeout Writing 1 will clear this bit; it is described in FIFOINTSET. The cause can be cleared by writing 1 to the FIFOSTAT[RXTIMEOUT]. 0 - No effect 1 - Clear the interrupt
23-4 —	Reserved Read value is undefined; only zero should be written.
3 RXLVL	Receive Level Interrupt Clear Writing one clears the corresponding bits in the FIFOINTENSET register. 0 - Interrupt is not cleared. 1 - Interrupt is cleared.
2 TXLVL	Transmit Level Interrupt Clear Writing one clears the corresponding bits in the FIFOINTENSET register. 0 - Interrupt is not cleared. 1 - Interrupt is cleared.
1	Receive Error Interrupt Clear

Table continues on the next page...

Table continued from the previous page...

Field	Description
RXERR	Writing one clears the corresponding bits in the FIFOINTENSET register. 0 - Interrupt is not cleared. 1 - Interrupt is cleared.
0 TXERR	Transmit Error Interrupt Clear Writing one clears the corresponding bits in the FIFOINTENSET register. 0 - Interrupt is not cleared. 1 - Interrupt is cleared.

41.6.5.1.1.13 FIFO Interrupt Status (FIFOINTSTAT)

The read-only FIFOINTSTAT register provides a view of those interrupt flags that are pending. This can simplify software handling of interrupts. Refer to the descriptions of interrupts in FIFOSTAT and FIFOTRIG for details.

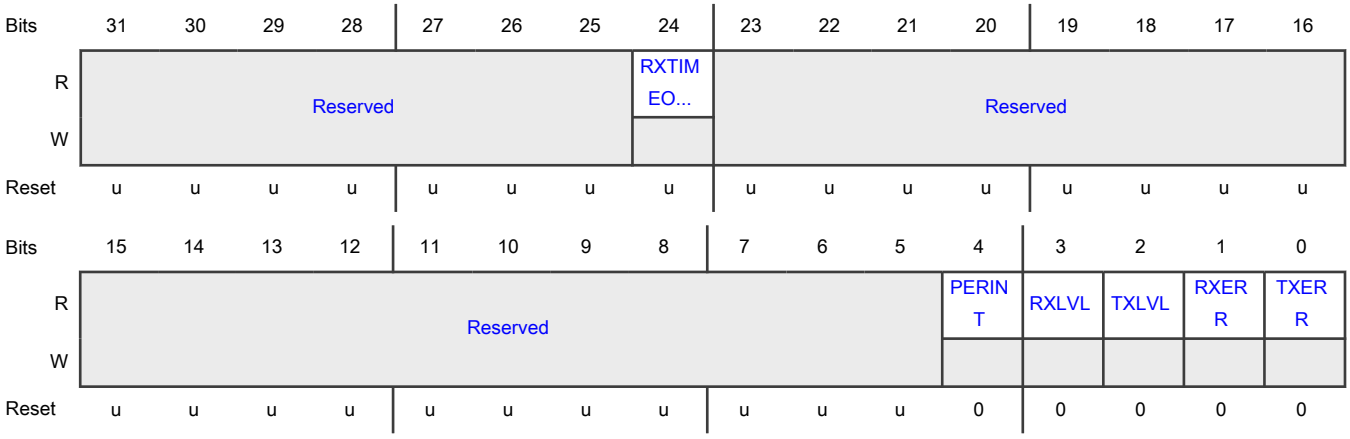
NOTE

Since all I^2S channels in a single Flexcomm module move data in the same direction, only TX related or RX related flags and controls are meaningful at any particular time.

Offset

Register	Offset
FIFOINTSTAT	E18h

Diagram



Fields

Field	Description
31-25	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	Read value is undefined; only zero should be written.
24 RXTIMEOUT	Receive Timeout Status 1 if both FIFOSTAT[RXTIMEOUT] and FIFOINTSET[RXTIMEOUT] = 1. Note that FIFOSTAT[RXTIMEOUT] may be cleared by writing 1 to it. 0 - Not pending 1 - Pending
23-5 —	Reserved Read value is undefined; only zero should be written.
4 PERINT	Peripheral Interrupt Status 0 - Not pending 1 - Pending
3 RXLVL	Receive FIFO Level Interrupt Status 0 - Not pending 1 - Pending
2 TXLVL	Transmit FIFO Level Interrupt Status 0 - Not pending 1 - Pending
1 RXERR	RX FIFO Error Interrupt Status 0 - Not pending 1 - Pending
0 TXERR	TX FIFO Error Interrupt Status 0 - Not pending 1 - Pending

41.6.5.1.1.14 FIFO Write Data (FIFOWR)

The FIFOWR register is used to write values to be transmitted to the FIFO.

Details of how FIFOWR and FIFOWR48H are used can be found in FIFO buffer configurations and usage.

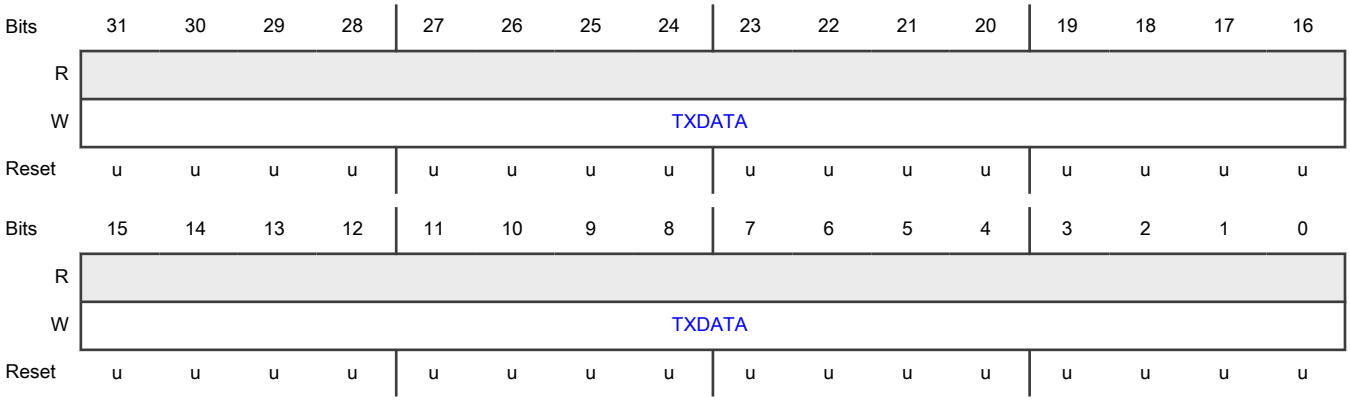
NOTE

Since all I^2S channels in a single Flexcomm module move data in the same direction, only FIFO read or write is meaningful at any particular time.

Offset

Register	Offset
FIFOWR	E20h

Diagram



Fields

Field	Description
31-0 TXDATA	Transmit Data to the FIFO The number of bits used depends on configuration details.

41.6.5.1.1.15 FIFO Write Data for Upper Data Bits (FIFOWR48H)

The FIFOWR48H register may only be used if the I^2S is configured for 2x 24-bit data and not using DMA.

>

Details of how FIFOWR and FIFOWR48H are used can be found in FIFO buffer configurations and usage.

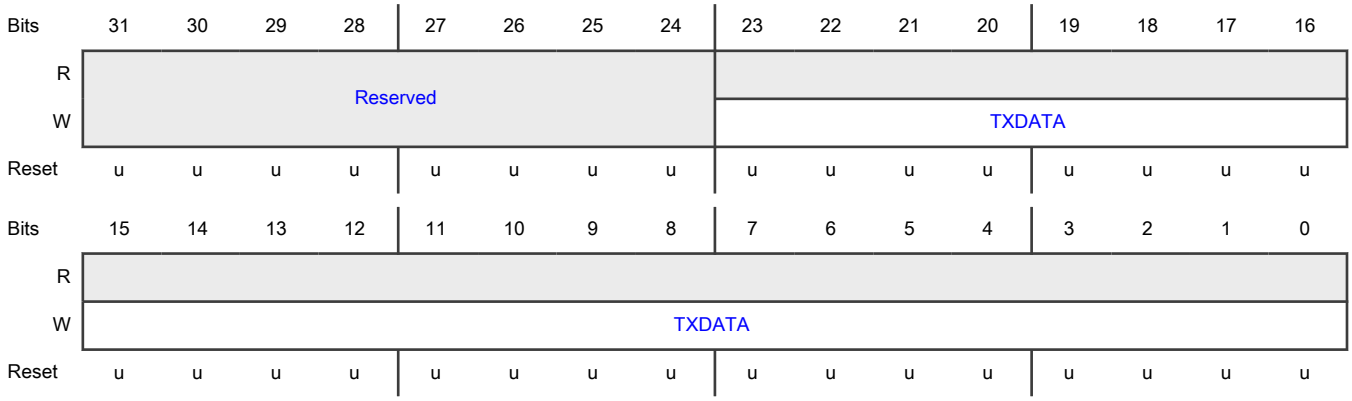
NOTE

Since all I^2S channels in a single Flexcomm module move data in the same direction, only FIFO read or write is meaningful at any particular time.

Offset

Register	Offset
FIFOWR48H	E24h

Diagram



Fields

Field	Description
31-24 —	Reserved Read value is undefined, only zero should be written.
23-0 TXDATA	Transmit Data to the FIFO Whether the FIFOWR48H register is used and the number of bits used depends on configuration details.

41.6.5.1.1.16 FIFO Read Data (FIFORD)

The FIFORD register is used to read values that have been received by the FIFO. Details of how FIFORD and FIFORD48H are used can be found in FIFO buffer configurations and usage

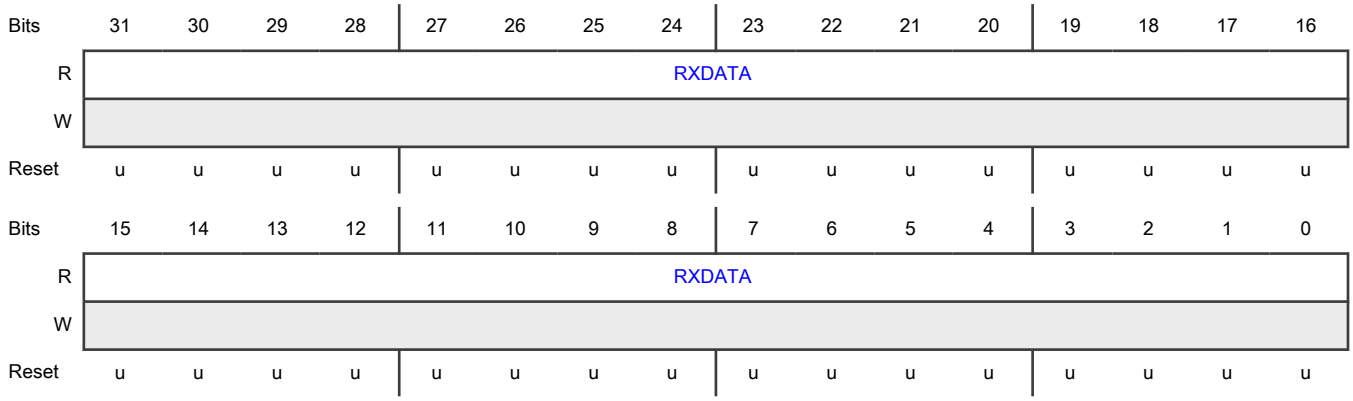
NOTE

Since all I^2S channels in a single Flexcomm module move data in the same direction, only FIFO read or write is meaningful at any particular time.

Offset

Register	Offset
FIFORD	E30h

Diagram



Fields

Field	Description
31-0 RXDATA	Received Data from the FIFO The number of bits used depends on configuration details.

41.6.5.1.1.17 FIFO Read Data for Upper Data Bits (FIFORD48H)

The FIFORD48H register may only be used if the I^2S is configured for 2x 24-bit data and is not using DMA.

The FIFORD48H register is used under certain conditions to read values from the FIFO. Details of how FIFORD and FIFORD48H are used can be found in FIFO buffer configurations and usage.

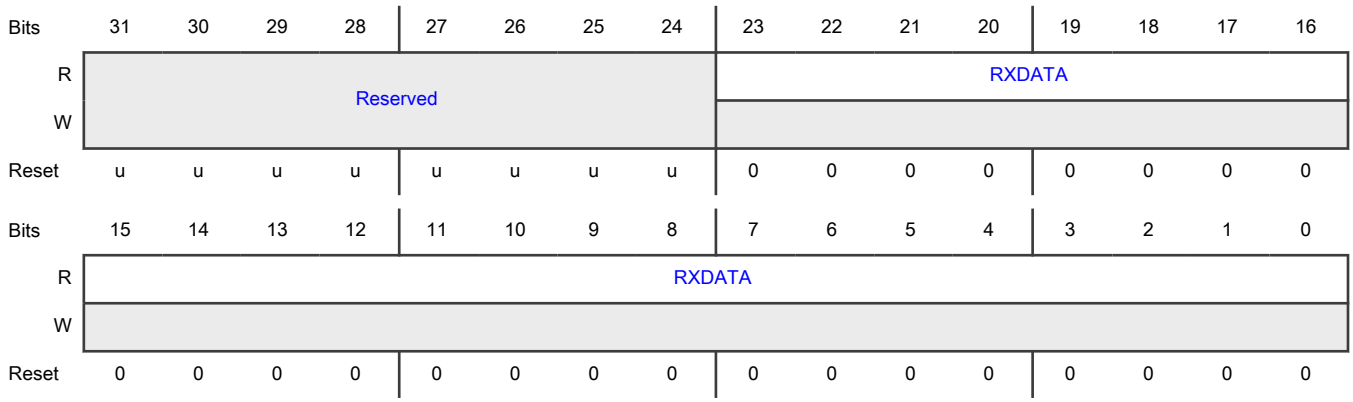
NOTE

Since all I^2S channels in a single Flexcomm module move data in the same direction, only FIFO read or write is meaningful at any particular time.

Offset

Register	Offset
FIFORD48H	E34h

Diagram



Fields

Field	Description
31-24 —	Reserved Read value is undefined, only zero should be written.
23-0 RXDATA	Received Data from the FIFO Whether the FIFORD48H register is used and the number of bits used depends on configuration details.

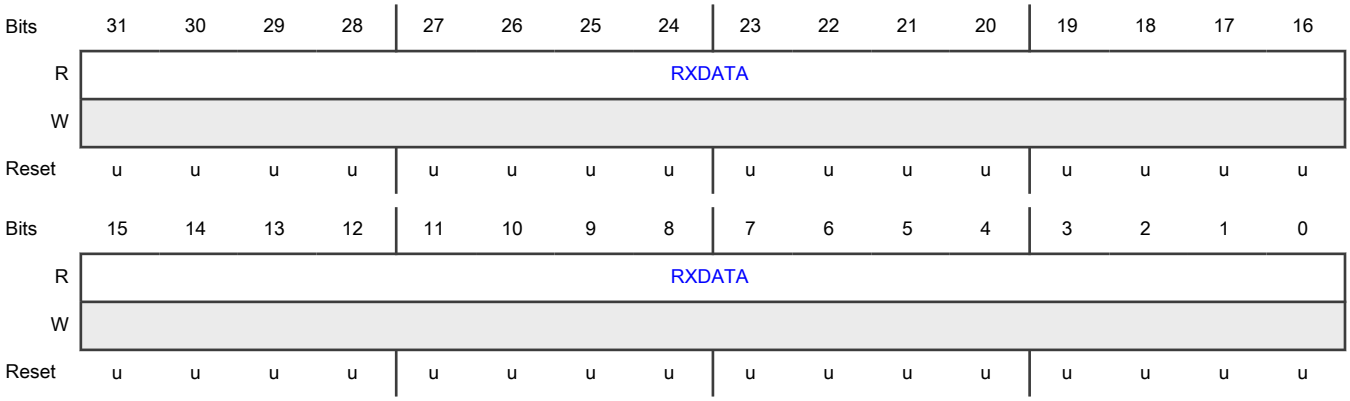
41.6.5.1.1.18 FIFO Data Read with No FIFO Pop (FIFORDNOPOP)

The FIFORDNOPOP register acts in exactly the same way as FIFORD, except that it supplies data from the top of the FIFO without popping the FIFO (i.e., leaving the FIFO state unchanged). This could be used to allow system software to observe incoming data without interfering with the peripheral driver.

Offset

Register	Offset
FIFORDNOPOP	E40h

Diagram



Fields

Field	Description
31-0 RXDATA	Received Data from the FIFO The number of bits used depends on configuration details.

41.6.5.1.1.19 FIFO Data Read for Upper Data Bits with No FIFO Pop (FIFORD48HNOPOP)

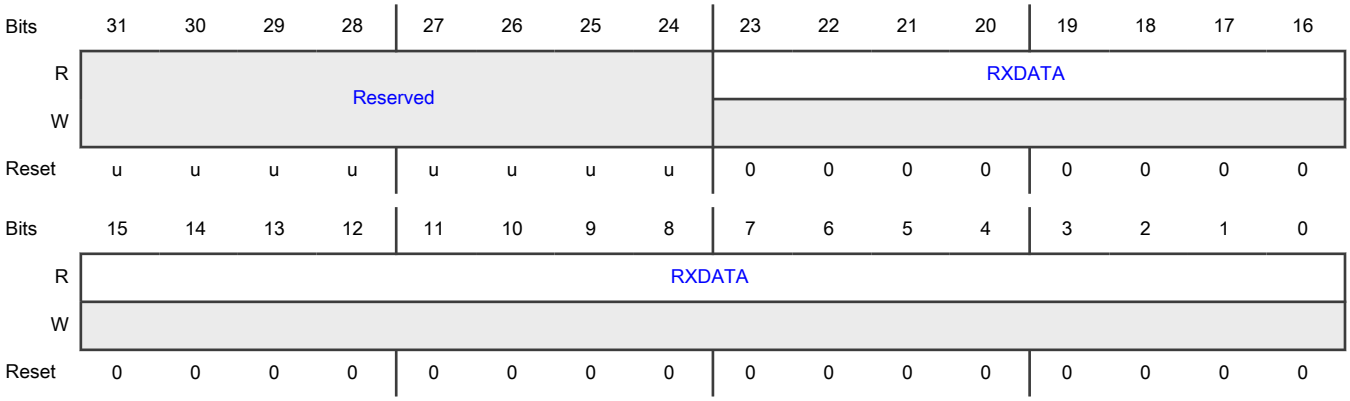
May only be used if the I^2S is configured for 2x 24-bit data and is not using DMA.

The FIFORD48HNOPOP register acts in exactly the same way as FIFORD48H, except that it supplies data from the top of the FIFO without popping the FIFO (i.e., leaving the FIFO state unchanged). This could be used to allow system software to observe incoming data without interfering with the peripheral driver.

Offset

Register	Offset
FIFORD48HNOPOP	E44h

Diagram



Fields

Field	Description
31-24 —	Reserved Read value is undefined, only zero should be written.
23-0 RXDATA	Received Data from the FIFO Whether the FIFORD48HNOPOP register is used and the number of bits used depends on configuration details.

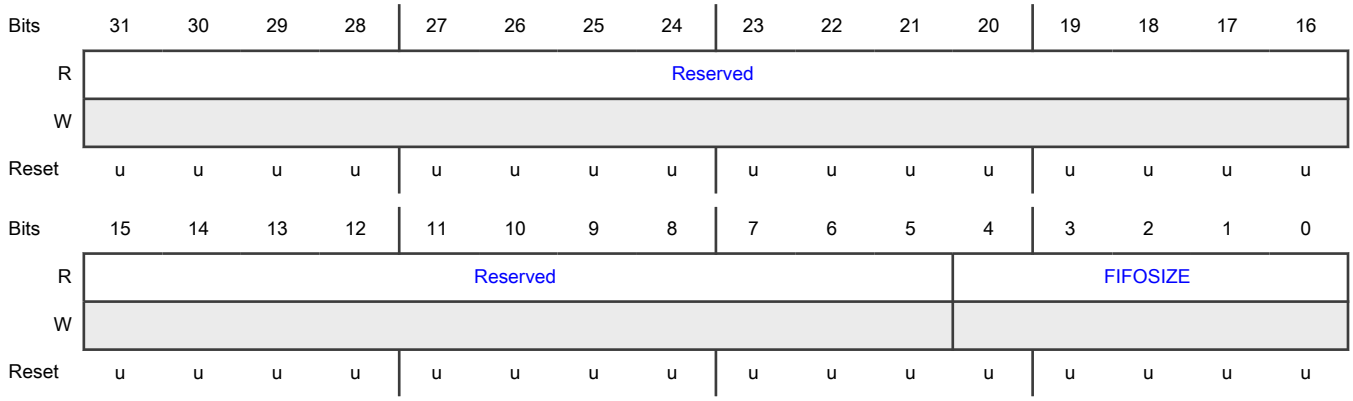
41.6.5.1.1.20 FIFO Size Register (FIFOSIZE)

Provides the FIFO size.

Offset

Register	Offset
FIFOSIZE	E48h

Diagram



Fields

Field	Description
31-5 —	Reserved
4-0 FIFOSIZE	FIFO Size Provides the size of the FIFO for software. FIFOSIZE is 8 entries for this chip. This field is read only.

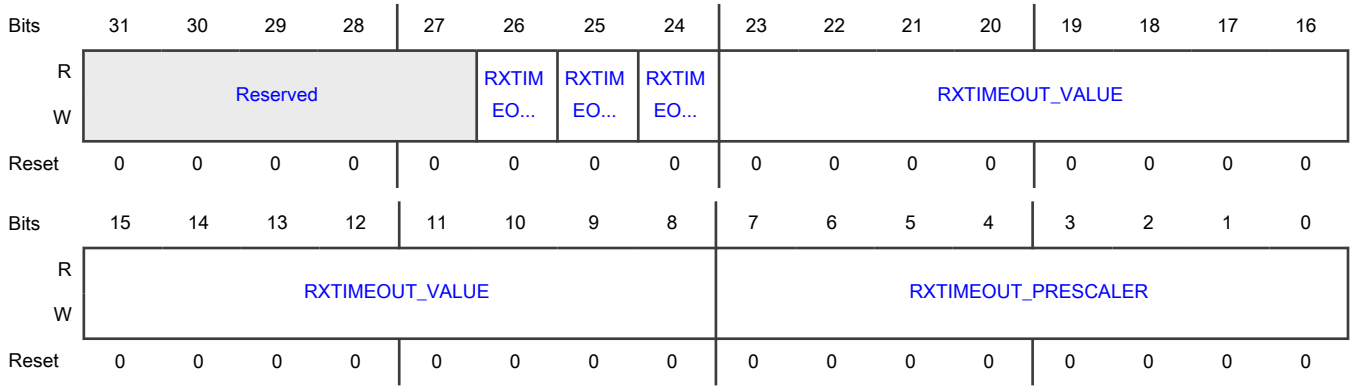
41.6.5.1.1.21 FIFO Receive Timeout Configuration (FIFORXTIMEOUTCFG)

When RXTIMEOUT_EN is enabled, the timeout counter is counting based on the configuration in this FIFORXTIMEOUTCFG register. When the counter reaches FIFORXTIMEOUTCFG[RXTIMEOUT_VALUE] + 1, FIFOSTAT[RXTIMEOUT] is set. The RX interrupt will be asserted if FIFOINTSET[RXTIMEOUT] bit is also set. FIFOSTAT[RXTIMEOUT] can be cleared by writing 1 to it.

Offset

Register	Offset
FIFORXTIMEOUTCFG	E4Ch

Diagram



Fields

Field	Description
31-27 —	Reserved
26 RXTIMEOUT_COE	Receive Timeout Continue On Empty RXTIMEOUT_COE allows the timeout to be used to flag idle peripherals, and could potentially be used to indicate the end of a transmission of indeterminate length. 0 - RX FIFO timeout counter is reset when the RX FIFO becomes empty. 1 - RX FIFO timeout counter is not reset when the RX FIFO becomes empty.
25 RXTIMEOUT_COW	Receive Timeout Continue On Write RXTIMEOUT_COW allows the timeout to be applied to accumulated data, perhaps related to the FIFO threshold. 0 - RX FIFO timeout counter is reset every time data is transferred from the peripheral into the RX FIFO. 1 - RX FIFO timeout counter is not reset every time data is transferred from the peripheral into the RX FIFO.
24 RXTIMEOUT_EN	Receive Timeout Enable 0 - Disable RX FIFO timeout 1 - Enable RX FIFO timeout
23-8 RXTIMEOUT_VALUE	Receive Timeout Value When RXTIMEOUT_VALUE is enabled and timeout counter reaches RXTIMEOUT_VALUE + 1, FIFOSTAT[RXTIMEOUT] is set.
7-0 RXTIMEOUT_PRESCALER	Receive Timeout Counter Clock Prescaler The clock frequency for FIFORXTIMEOUTCNT[RXTIMEOUT_CNT] is the AHB bus clock frequency divided by 16 * (RXTIMEOUT_PRESCALER + 1).

41.6.5.1.1.22 FIFO Receive Timeout Counter (FIFORXTIMEOUTCNT)

Timeout counter gets reset on one of the following conditions:

- FIFORXTIMEOUTCFG[RXTIMEOUT_EN] is 0
- FIFOSTAT[RXTIMEOUT] is written with 1
- Rx FIFO is empty and FIFORXTIMEOUTCFG[RXTIMEOUT_COE] is 0
- Rx FIFO is written and FIFORXTIMEOUTCFG[RXTIMEOUT_COW] is 0

Timeout counter is paused when FIFOSTAT[RXTIMEOUT] is set by hardware and is cleared when FIFOSTAT[RXTIMEOUT] is written with 1 by software.

Offset

Register	Offset
FIFORXTIMEOUTCNT	E50h

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	RXTIMEOUT_CNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-16 —	Reserved
15-0 RXTIMEOUT_CNT	Current RX FIFO timeout counter value

41.6.5.1.1.23 I2S Module Identification (ID)

The ID register identifies the type and revision of the module. A generic software driver can use this information to implement specific behavior, based on a module type or a specific revision of that module.

Offset

Register	Offset
ID	FFCh

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ID															
W																
Reset	1	1	1	0	0	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	MAJOR_REV				MINOR_REV				APERTURE							
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-16 ID	Module Identifier Unique module identifier for this IP block
15-12 MAJOR_REV	Major Revision Major revision of module implementation, starting at 0
11-8 MINOR_REV	Minor Revision Minor revision of module implementation, starting at 0
7-0 APERTURE	Aperture aperture size = (APERTURE + 1) x 4K APERTURE encoded as 0x00 means 4K aperture: aperture size = (0x00 + 1) x 4K

Chapter 42
Improved Inter-Integrated Circuit (I3C)

42.1 Chip-specific I3C information

Table 348. Reference links to related information

Topic	Related module	Reference
Full description	I3C	I3C
System memory map		System memory map
Clocking		Clock distribution
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

42.1.1 Module instances

This device has only one instance of the I3C module, I3C0. This chip supports an I3C master/slave interface that is accessible by both CPUs and both DMA controllers.

42.2 Overview

The MIPI Alliance Improved Inter-Integrated Circuit (MIPI I3C) improves upon the use and power of I2C, and provides an alternative to SPI for mid-speed applications.

The I3C bus protocol supports:

- In-band interrupts (IBI). These interrupts go from target to controller without extra wires, and the controller knows which target sent the interrupt.
- Common Command Codes (CCC)
- Dynamic addressing
- Multi-controller/multi-drop
- Hot-Join (HJ)
- I2C compatibility

The I3C peripheral supports all required and most optional features of the MIPI Alliance Specification for I3C, v1.0, except for ternary data rates (HDR-TSP and HDR-TSL).

NOTE

Terminology in this chapter has been updated to align with MIPI I3C Specification v1.1.1.

Table 349. Updated terms

Updated term	Deprecated term
Controller	Master
Target	Slave
Controller Request (CR)	Master Request (MR)

42.2.1 Block diagram

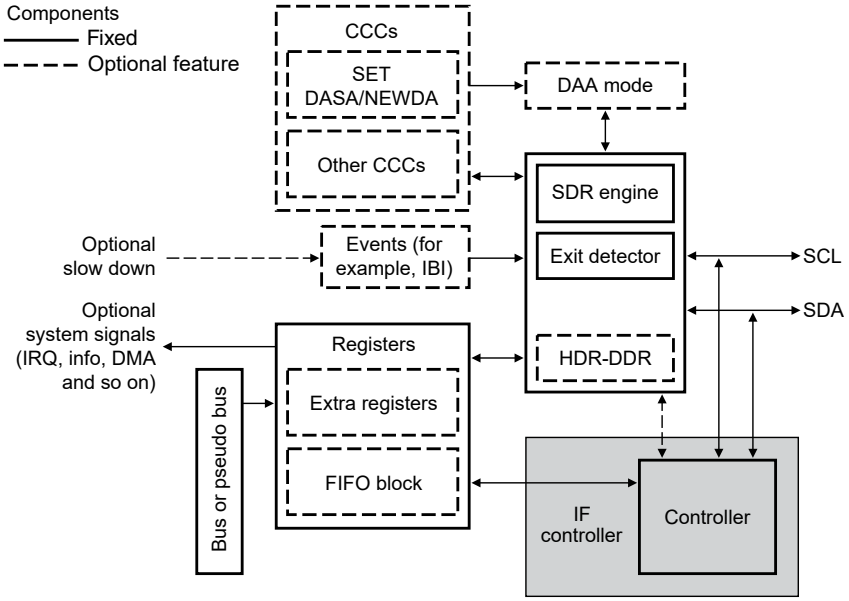


Figure 185. I3C block diagram

42.2.2 Features

I3C module:

- Two-wire multi-drop bus capable of 12.5 MHz clock speeds, with up to 11 devices.
  - Uses standard pads with 4 mA drive.
  - Dynamically assigns target addresses, and targets do not require static addresses. However, targets may have an I2C static address assigned at start-up, so the target can operate on an I2C bus. By default, I3C supports seven-bit I2C-style addresses.
  - Allows targets to use the inbound SCL clock as the peripheral clock (instead of the clock from the controller) so devices can have slow or inaccurate clocks internally.
  - Allows simple targets, such as temperature sensors, to have no internal clock.
  - I3C controller supports handoff from Open Drain to Push-Pull mode for ACK to data transfer.
  - Normally the controller terminates the read, but for I3C, the target can also end the read.
- In-Band Interrupts (IBI) allow targets to send notifications to a controller.
  - Can be equivalent to a separate GPIO, but can also be directly data-bearing.
  - Can be prioritized. When multiple targets send interrupts to a controller at the same time, the order is resolved. Dynamic addresses establish the priority of the targets, so the controller controls the priority of the targets. Targets with lower-value dynamic addresses are higher priority level IBIs.
  - Can start interrupts even when the controller is not active on the bus. No free-running clock is needed, but starting an interrupt requires a Bus Available condition.
  - Can resolve an initial event via a time-stamping option, not requiring an interrupt.
- Built-in commands are in a separate space. These commands do not collide with normal controller-to-target messages.
  - Controls bus behavior, modes and states, low-power state, inquiries, and more.

  - Has additional room for new built-in commands to be used by other groups.
- Organized forms of multi-controller modes:
  - Secondary controllers, which use clean handoffs between different controllers.
- Hot-join onto I3C bus allows devices to connect to the bus later than when the bus starts.
  - Enables a device or module to get onto the I3C bus when it woke up after power-up or was physically inserted onto the I3C bus.
  - Provides a clean method for notification when new devices or modules get onto the I3C bus.
- Can use both I2C and I3C buses.
  - I3C supports specific legacy I2C devices on the bus.
  - I3C target devices can operate on I2C buses.
  - Supports bridging to I2C, SPI, UART, and other buses.
  - Special mode for old-style I2C buses (no targets) with clock stretching
- Higher data rate modes are available.
  - Has a High Data Rate - Double Data Rate (HDR-DDR) mode, which is double the data rate of SDR
  - Only the controller and the specific target must support the higher data rate. The other targets can ignore it.

The I3C peripheral supports the full I3C feature set, except for the ternary data rates (HDR-TSP and HDR-TSL).

42.3 Functional description

The following sections describe functional details of the I3C module.

42.3.1 Operating modes

This section describes all functional operation modes of the I3C module.

42.3.1.1 Controller and target roles for I3C

The I3C protocol defines these roles for devices on the I3C bus:

- Main Controller. Controls the I3C bus.
- Secondary Controller. Controls I3C with permission from the Main Controller, and returns control of the bus to the Main Controller when the Secondary Controller has finished its tasks.
- Target. Responds to commands from any I3C controller.
- I2C target. Responds to commands from any I2C target.

The I3C peripheral contains both controller and target components and can be parameterized to be either controller or target, or both controller and target. However, if the I3C is chosen to be a controller, then the I3C peripheral supports Target mode to facilitate handoffs to multiple controllers.

In general, any I3C controller can be a controller or a target, because a controller becomes a target when giving over control to another controller. The only exceptions to this rule occur when using a point-to-point controller or when using a controller that never shares controllership.

42.3.1.1.1 Controller requirements

Controller mode uses software that supports the requirements of the controller (including as a Secondary Controller):

- Managing the Enter Dynamic Address Assignments (ENTDAA) assigning dynamic addresses to each target. This process is supported by the I3C peripheral but requires the software to make choices.

- Driving the serial data (SDA) signal in both Open Drain and Push-Pull mode. After Start, SDA is in Open Drain mode and after Repeated Start it is in Push-Pull mode.
  - SDA is subject to arbitration, because both controller and target can drive the SDA line. Arbitration is also useful for handoff from controller to target.
  - The ninth bit of SDA controller write data is an odd parity bit.
- Building a table of targets and their capabilities to control which actions and commands may be sent to the targets.
- Managing requests such as IBI and controller request (handoff).
- Adjusting clock speed or write-to-read timing to match the limitations of the target. This adjustment can be done in hardware using dividers and uneven duty cycles, but the software must decide.
- Adjusting the maximum data length.
- Being a target after a controller handoff to another controller.

Additionally, a controller needs three pins unless the SDA pad includes a precise, high-strength pullup. The three-pin model allows one pad/pin to enable a system-provided pullup that is sized to system requirements.

The controller needs an accurate clock, capable of running at a frequency that is a multiple of a frequency between 11 MHz and 12.5 MHz. For example:

- 24 MHz (multiple of 12 MHz)
- 48 MHz (multiple of 12 MHz)
- 25 MHz (multiple of 12.5 MHz)
- 50 MHz (multiple of 12.5 MHz)
- 33 MHz (multiple of 11 MHz)
- 66 MHz (multiple of 11 MHz)

Greater multiples can also be used.

42.3.1.1.2 I3C target acts like I2C target on I3C buses

If it is assigned an I2C static address, the I3C target acts like an I2C device when it first turns on. If the I3C target is placed on an I2C bus with an I2C controller, then the I3C target stays in I2C mode and operates normally. The software is aware that the I3C target is in I2C mode, because:

- There has not been a SSTATUS[DACHG] interrupt indicating that a dynamic address was assigned.

For full I2C support of Fast Mode (FM) and Fast-mode Plus (Fm+), the pads must support a 50-ns spike filter. This filter must be turned off when the I3C 7Eh broadcast address is received (indicating an I3C controller). You can turn off the spike filters via hardware using the raw net indicating that the address was received or via software. A 50-ns spike filter is not needed for the I3C to operate on an I2C bus. Therefore, the spike filter is not a requirement for the I3C peripheral.

Depending on the configuration, the block supports most I2C features, except for clock stretching. Supported features include extended 10-bit address, DeviceID, software reset, and high-speed mode (affects pads).

42.3.1.1.3 How a target rejoins the I3C bus

When a target tries to rejoin the I3C bus following a power-up or hard reset, the target needs a new Dynamic Address (DA). The target can rejoin the I3C bus in these ways:

- If the dynamic address is lost, Hot-Join is used.
- If the dynamic address is retained in the peripheral (for example, with state retention flip-flops), SCONFIG[OFFLINE] is used (see below).

When SCONFIG[SLVENA] is 1, SCONFIG[OFFLINE] may become 1. This setting causes the peripheral to rejoin the bus safely. It does so by ensuring that the I3C bus is not in HDR mode, using the same approach as TE0 or TE1 exit. The peripheral waits for the HDR Exit Pattern, or for 60 µs of SCL and SDA lines not changing, whichever occurs first.

After using SCONFIG[OFFLINE], the I3C peripheral cannot safely use IBI until SSTATUS[STOP] is 1. This status ensures that the next START is safe to use for IBI.

If the application must perform an IBI, it should wait for SSTATUS[STOP] to become 1, or for SCL and SDA lines to remain high for 200 µs. This process can be done using the SSTATUS and SINTSET controls.

- If SSTATUS indicates that the bus is not busy and the peripheral interrupts on START or STOP, use a timer to measure 200 µs. If the timer finishes with no START or STOP, it is safe to use IBI.
- If a START causes an interrupt, then the timer should be turned off and the application should wait for a STOP.
- If a STOP causes an interrupt, it is safe to use IBI.

42.3.1.2 Using the I3C controller for I2C and I3C

The I3C controller operates with this set of built-in capabilities with the application flow:

- Built-in Enter Dynamic Address Assignment (ENTDAA) mechanism to simplify the assignment of dynamic addresses to targets. This feature is used when Set Dynamic Address from Static Address (SETDASA) is not available.
- Request for START + Address with IBI support, including both I2C and I3C modes.
- SDR Write flow via FIFO, with automatic parity in I3C and ACK or NACK detection in I2C.
- SDR Read flow via the FIFO, with an automatic NACK generator for I2C, and optional use of a terminate generator for I3C.
- Request for repeated START + Address or STOP when finished with previous, including both I2C and I3C.
- Auto-IBI mode, which responds immediately to a target-initiated IBI and can be used when in sleep or deep sleep.
- Special message mode for SDR and DDR to simplify for DMA use.
- Automated SCL, SDA, pullup, and High-Keeper controls.
- I2C and I3C frequency and duty cycle configurations from functional clock.

NOTE

The Controller Configuration (MCONFIG) and Controller In-band Interrupt Registry and Rules (MIBIRULES) registers must be configured before using the controller.

42.3.1.3 Protocol modes and states

The following modes are activated in I3C:

- I2C mode is the default mode on startup.
  - If no I2C Static Address is used, this default has no effect other than to track frames looking for I3C transitional frames.
  - If an I2C Static Address is used, the device can interact with the I2C bus controller using the address.
- While in I2C mode, I3C Transitory Frame mode occurs whenever the I3C broadcast address is received (7Eh). In particular:
  - This mode occurs when 7Eh is broadcast followed by a SETDASA from the controller. If there is a Static Address match or 01h, the target is assigned a Dynamic Address and it enters I3C SDR mode.
  - This mode occurs when 7Eh is broadcast followed by an ENTDAA from the controller. If the target can send a 48-bit provisioned ID (48b ID), a Dynamic Address is assigned, and the target enters I3C SDR mode.

— If a Hot-Join arbitrated event occurs (sending 02h when the controller generates another address such as 7Eh), there is a conceptual Hot-Join mode. ENTDAA or SETDASA sets a Dynamic Address to resolve the event, and the target enters I3C SDR mode.

NOTE

An I3C target can operate as a normal I2C target only when it has a Static Address. Otherwise, it matches nothing in I2C and waits for the above events.

- I3C SDR mode is the standard mode once I3C activates, which is defined by a Dynamic Address being assigned. This mode is the resting mode of I3C, and all devices are normally in SDR mode.
  - RSTDAA CCC causes an exit from SDR mode and a return to I2C mode. This transition is not normally used.
  - SETNEWDA does not affect the SDR mode, it only changes the Dynamic Address of an I3C device that already had a Dynamic Address assigned.
  - When in SDR mode, the device ignores messages to or from its original I2C Static Address.
- I3C CCC Command submode of type Direct or Broadcast.
  - Exit the Broadcast CCC submode by a repeated START or by a STOP.
  - Exit the Direct CCC submode by a 7Eh broadcast address after a repeated START or by a STOP.
  - I3C Dynamic Address Assignment (DAA) CCC command enters DAA mode. This mode is a special mode for dynamic addressing. Use STOP to exit. If a target has a Dynamic Address, the command is ignored.
- I3C SETDASA CCC command enters the Static Address Match submode. This mode allows matching the Static Address of the device (if any).
  - The command is ignored when a target has a Dynamic Address, or when the target in I2C mode has no Static Address.

NOTE

The special point-to-point address is also matched.

- I3C HDR modes are activated by ENTHDRn CCC commands, where n represents the type of HDR (0 to 7). This mode is valid until an HDR Exit Pattern, whether HDR mode for a slave is supported or not.
  - DDR included
  - Exit-pattern detection must always be on to prevent errors. Alternatively, it could be set to only activate on ENTHDR.
- Internal states such as IBI/no-IBI and low-power mode, are flagged internally. See SSTATUS[EVDET]. These states are exported to the application and affect the engine to prohibit it from performing a prohibited operation.

42.3.1.4 Address match

For an address match to occur, the target matches the I3C broadcast address and either the I2C-style Static Address or the I3C Dynamic Address.

- I3C broadcast address, 111 1110b (written as 7Eh in spec)
- I2C-style Static Address, but only if:
  - The device has a Static Address.
  - Not in I3C mode. Only matches until the ENTDAA or the SETDASA command has assigned a Dynamic Address.

NOTE

SETDASA matches the Static Address or the special one-controller-to-one-target point-to-point address.

- I3C Dynamic Address once assigned by ENTDAA or SETDASA or modified by SETNEWDA.

The address match is inactive (just listening) for all repeated START commands and also for START, unless an IBI, CR, or Hot-Join has been activated. If inactive, it simply tries to match. If not matched, it waits for a repeated START or a STOP. If an IBI, CR, or Hot-Join has been activated, the arbitration mechanism is used for START (but never for a repeated START).

If matching the 7Eh broadcast address, it listens for a CCC until the next repeated START or a STOP.

42.3.2 Operations

This section describes the operations of the I3C module.

42.3.2.1 Reading and writing I2C messages using the normal method

NOTE

I2C FM and I2C FM+ modes are supported for legacy I2C devices as per the I3C standard specifications. See I2C configuration for meeting timing requirement for FM and FM+ modes.

The normal method is as follows.

1. Set up interrupts.
   - MCTRLDONE: Indicates when the I3C module completes an MCTRL request.
   - COMPLETE: Indicates when data has finished sending or being received.
   - RXPEND: For read operations. Used with the Controller Data Control (MDATACTRL) register to set the FIFO trigger. Can also be used to allow DMA to read out data.
   - TXNOTFULL: For write operations. Used with the MDATACTRL register to set the FIFO trigger. Can also be used to allow DMA to supply data.
   - IBIWON: Indicates that an IBI, CR, or HJ has won the arbitration on a header address.
   - MERRWARN: Indicates causes of errors and warnings.
2. Configure these fields in the Controller Control (MCTRL) register simultaneously.
   a. Write 1 to MCTRL[REQUEST] (EmitStartAddr).
   b. Write 1 to MCTRL[TYPE] (I2C).
   c. Set MCTRL[IBIRESP] to respond to IBIs in your chosen manner.
   d. Write 1 to MCTRL[DIR] for read, or write 0 for write.
   e. Write the static address of the I2C target to MCTRL[ADDR].
   f. For read operations, you can set MCTRL[RDTERM] to the maximum length to auto-terminate, or you can set it to stop as the data is read out. For example, set it to 1 (with REQUEST = 0) to stop after the next character.
3. Write or read the data.
   - For write operations, write the Controller Write Data Byte (MWDATAB) register for each byte before the last byte, then write the Controller Write Data Byte End (MWDATABE) register for the last byte.
     — This operation can be done or started before setting up interrupts.
     — If there is more data than the FIFO can hold, use the TXNOTFULL interrupt based on the trigger level. This interrupt allows the application to provide more data or to use DMA.
   - For read operations, wait for RXPEND, and then read out data via the Controller Read Data Byte (MRDATAB) register. DMA may also be used.
4. On COMPLETE, the message may be ended with a STOP, or a new message started with a repeated START.
   a. Write 2 to MCTRL[REQUEST] (EmitStop to STOP). Then wait for MCTRLDONE status to be asserted for its completion. (While sending STOP in I2C mode MCONFIG[ODSTOP] should be 1 and MCTRL[TYPE] should be 1.)

b. Write 1 to MCTRL[REQUEST] (EmitStartAddr to restart). IBI is not possible in this case.

42.3.2.2 Reading and writing I3C messages using the normal methods (SDR and HDR-DDR)

The normal method for I3C is the same as the method for I2C with a few differences.

1. Set up interrupts:
   - MCTRLDONE: Indicates when the I3C module completes an MCTRL request.
   - COMPLETE: Indicates when data has finished sending or being received.
   - RXPEND: For read operations. Used with the Controller Data Control (MDATACTRL) register to set the FIFO trigger. Can also be used to allow DMA to read out bytes.
   - TXNOTFULL: For write operations. Used with the Controller Data Control (MDATACTRL) register to set the FIFO trigger. Can also be used to allow DMA to supply bytes.
   - IBIWON: Indicates that an IBI, CR, or HJ has won the arbitration on a header address.
   - MERRWARN: Indicates causes of errors and warnings for software to check.
2. Set up the Controller Control (MCTRL) register.
   - **Option 1:** Configure the MCTRL register fields simultaneously in this way:
     a. Write 1 to MCTRL[REQUEST] (EmitStartAddr).
     b. Write 0 to MCTRL[TYPE] (for I3C SDR mode) or write 2 (for DDR mode)
     c. Set MCTRL[IBIRESP] to respond to IBIs in your chosen manner.
     d. Write 1 to MCTRL[DIR] for read, or write 0 for write.
     e. Write the dynamic address of the I3C target to MCTRL[ADDR].
     f. For read operations, you can set MCTRL[RDTERM] to the maximum length to auto-terminate.
     g. For write operations, pre-writing the data (MWDATAB or MWDATAH) is preferred to ensure that there are no time delays waiting on the data.

        For DMA with MCTRL, use Controller Write Byte Data 1(to bus) (MWDATAB1) register.

     NOTE

     HDR-DDR mode requires writing an 8-bit command value for read or write. This value must be written into the TX FIFO via Controller Write Data Byte (MWDATAB). The END bit is not used for this byte.

   - **Option 2:** This option is preferred when stopped (bus free condition) in SDR mode, and not in HDR-DDR mode. It allows any target to issue an IBI, and it avoids collisions with an IBI address. Also, it is faster (when the MSB of the Dynamic Address is always 0).

     Configure the MCTRL register in this way:
     a. Write 1 to MCTRL[REQUEST] (EmitStartAddr). No pre-written transmit data can be in the FIFO.
     b. Write 0 to MCTRL[TYPE] (I3C).
     c. Set MCTRL[IBIRESP] to respond to IBIs in your chosen manner.
     d. Write 0 to MCTRL[DIR].
     e. Write 7Eh to MCTRL[ADDR].
     f. Wait for MCTRLDONE (via interrupt, for example), then proceed as in Option 1. The Option 2 method has advantages for IBIs when 7Eh is sent on START (but not on repeated STARTs).
3. Write or read the data.

- For write operations, write the MWDATAB register for each byte before the last byte, then write the Controller Write Data Byte End (MWDATABE) register for the last byte. For HDR-DDR, the byte with END must be even (second, fourth, sixth, and so on) because DDR uses byte pairs.
  - This operation can be done or started before step 1 (REQUEST = 1) of the Option 1 method, but not before the Option 2 method.
  - If there is more data than the FIFO can hold, use the TXNOTFULL interrupt based on the trigger level. This interrupt allows the application to provide more data or to use DMA.
- For read operations, wait for RXPEND, and then read out data via the Controller Read Data Byte (MRDATAB) or Controller Read Data Halfword (MRDATAH) register. DMA may also be used. If using DMA, read using same registers.

4. On COMPLETE, the message may be ended with STOP (or EXIT in HDR mode). Alternatively, a new message may be started with a repeated START (or HDR-Restart in HDR mode).
   - In SDR mode, write 2 to MCTRL[REQUEST] (EmitStop to STOP). Then wait for MCTRLDONE status to be asserted for its completion.
   - For HDR mode, write 6 to MCTRL[REQUEST] (ForceExit) to end HDR mode. Then wait for MCTRLDONE status to be asserted for its completion. When sending the HDR exit pattern, MCONFIG[ODSTOP] must be 0.
   - Write 1 to MCTRL[REQUEST] (EmitStartAddr to start another message). IBI is not possible in this case.

42.3.2.3 Sending a CCC to I3C targets

The normal Common Command Code (CCC) method is to use an I3C write operation with an address of 7Eh. The first byte is the CCC.

- For Broadcast type, any remaining bytes are sent with the CCC. This operation ends with a STOP or a repeated START and 7Eh.
- For Direct type, this write is followed only by a CCC (or by a CCC and a defining byte, if required by the CCC).

  These bytes are followed by a repeated START and the address of the I3C target (for SETDASA, this address is its I2C Static address). This sequence may be repeated with more repeated STARTs and addresses until done. It may end in STOP or a repeated START and 7Eh. After the repeated start and target address, the values are read or written depending on the CCC.
- I3C provides interrupts (by hardware) for Unhandled and Handled CCC when received by the target. Its state is reflected in the Target Status (SSTATUS) register.

The first broadcast 7Eh must meet an open-drain timing check:

- Use the configurations MCONFIG[ODHPP] = 0 and the normal MCONFIG[ODBAUD], ensuring at least 200 ns for the open-drain SCL half-clock period.
- Follow with a repeated START or a STOP.

All subsequent messages must use MCONFIG[ODHPP] = 1.

The I3C spec requires I3C controllers to emit a single START, 7E/W sequence with both SCL High and SCL Low half periods at full open-drain timing (for example, 200 ns). This sequence allows I3C targets acting as I2C legacy devices to turn off their I2C 50-ns spike filters, if they have them.

NOTE

START, 7E/W is notation indicating a START command, followed by the 7Eh broadcast address, followed by a write command.

After that sequence, addresses following START may be sent with Open-Drain Low (for example, 200 ns) but High of Push-Pull timing (for example, 40 ns). The I3C controller should emit this START, 7E/W sequence with MCONFIG[ODHPP] = 0. ODHPP is Open-Drain High period at Push-Pull speeds, and MCONFIG[ODHPP] is usually 1.

NOTE

MCONFIG[ODHPP] is ignored when sending a message to an I2C legacy device on an I3C bus.

NOTE

Each repeated START is a new EmitStart request. This request can be chained by interrupt or pushed by message model using DMA.

42.3.2.4 In-Band Interrupt (IBI) handling

An IBI occurs when a target sends its address after a START, and that address is numerically the lowest. That is, it is lower than the address sent by the controller and addresses sent by any other targets. When the controller sends 7Eh, the controller always loses the arbitration, by design.

The IBI can occur unexpectedly when any new START (not a repeated START) is sent. The IBI can also occur in response to a target pulling SDA low. This condition can occur in one of two ways:

- The target has set the request to AutoIBI mode, so the IBI occurs automatically. The engine sends 7Eh to allow the target to win the arbitration.
- The application receives the SLVSTART (target START request) interrupt, so it sends 7Eh.

The IBI response is configured by MCTRL[IBIRESP], which can be set to:

- NACK. Always reject the IBI.
- ACK. Always accept the IBI.
  - The Controller In-band Interrupt Registry and Rules (MIBIRULES) register must be configured so that the engine knows whether bytes follow.
  - If IBI bytes follow (also known as IBI Mandatory byte), then the COMPLETE field does not become 1 when IBIWON becomes 1. RXPEND becomes 1 for one or more bytes in the Receive FIFO. When the last byte is received, then COMPLETE becomes 1. The controller automatically stops IBI data after nine total bytes (including Mandatory Data byte). MCTRL[RDTERM] can be used with MCTRL[REQUEST] = 0 to end sooner. I3C supports address ACK to Mandatory Byte transition during IBI from Open_Drain (controller ACKs the target address) to Push Pull (target sends SDR data).
  - I3C target supports up to seven bytes of Extended IBI Data following a Mandatory Data byte. Extended Data to send, if any, is present in the Extended IBI Data 1 (IBIEXT1) and Extended IBI Data 2 (IBIEXT2) registers.
- Manual. Allow the decision to be made by the application on a case-by-case basis.
  - The application rewrites it when stopped, pending an IBI.
  - The application can ACK or NACK the IBI based on the IBI address in the MSTATUS register.
  - This mode chooses whether there is an IBI byte or not when accepting.

Accurate timestamping of I3C target data is supported in I3C though Async Mode 0. In I3C Asynchronous Timing Control mode, a target device timestamp event occurs within that target. The target notifies the controller about the event by generating an IBI.

42.3.2.5 Assigning dynamic addresses to I3C devices

If Dynamic Addresses (DAs) are all assigned below 40h (seven-bit values from 3Fh down to 03h, except where not allowed), then MIBIRULES[MSB0] can be 1. This setting optimizes the START timing. When dynamic addresses are assigned, the Controller In-band Interrupt Registry and Rules (MIBIRULES) register must be programmed based on which I3C target uses IBI bytes (known from Bus Characteristics Register (BCR)).

Any Set Dynamic Address from Static Address (SETDASA) assignments can be done via the normal Common Command Code (CCC) model. These assignments cannot be done when using the static address for the directed part.

There is a built-in mechanism to process the Dynamic Address Assignment (DAA) mode.

1. Set up interrupts (see Controller Interrupt Set (MINTSET)):

- MCTRLDONE. Indicates when a target has sent its ID and BCR or DCR, and a new DA is needed.
- COMPLETE. Indicates when DAA is done (NACKed by all targets).
- RXPEND. Indicates the reading of the IDs of the targets. Can also be used to allow DMA to read out bytes.
- IBIWON. Indicates when an IBI has occurred, which should not be possible normally.
- MERRWARN. Indicates an error.

2. Configure the Controller Control (MCTRL) register.
   - Write 4 to MCTRL[REQUEST] (ProcessDAA).
   - Set MCTRL[IBIRESP] to IBI response, if possible. If it is not possible, set to the first target assignment.
3. Wait for the MCTRLDONE interrupt, while reading ID using the RXPEND interrupt. If MSTATUS[STATE] = 5 (DAA mode) and MSTATUS[BETWEEN] = 1, it is waiting for a DA for the target whose ID was read.
   - Write the dynamic address into the Controller Write Data Byte (MWDATAB) register using bits 6:0, such as 14h for DA = 7'h14.
   - Write 4 to MSTATUS[STATE] (ProcessDAA again). This option writes the DA, then moves to the next DA.
   - If MSTATUS[COMPLETE] = 1 and MSTATUS[STATE] = 0, then all targets have been assigned.
   - If MSTATUS indicates NACK after writing a new DA, the DA was not accepted by the target. The next step is to write 2 to MCTRL[REQUEST] (EmitStop) and start over. It is also acceptable to write 4 to MCTRL[REQUEST] (ProcessDAA again).

42.3.2.6 Using Controller Message mode

Controller Message mode is intended for use with DMA, though Message mode can also be used by the processor. In Message mode, all writes are to the same location, including the control and the data. Reads occur from an associated location.

NOTE

The data writes for message mode work in the same way as the halfword access registers Controller Write Data Halfword (MWDATAH) and Controller Read Data Halfword (MRDATAH).

To send a message via Single Data Rate (SDR), follow these steps (from DMA or processor):

1. Write the control request to the Controller Write Message Control in SDR mode (MWMSG_SDR_CONTROL) register. This request includes:
   - The address.
   - I2C or I3C.
   - Read or write.
   - The count of bytes to process.
   - How to end (on STOP or ready for repeated START).
2. Process the Data.
   - For write operations, write the rest of the data to Controller Write Message Data in SDR mode (MWMSG_SDR_DATA). Use the DMA trigger (or interrupt) to keep the Transmit FIFO full, and the controller stops using the data when the program count MWMSG_SDR_CONTROL register reaches 0.
   - For read operations, a DMA trigger (or interrupt) indicates when the RX FIFO is ready to be read out via the Controller Read Message in SDR mode (MRMSG_SDR) register.
3. In-band Interrupt (IBI) behavior is selected in MCTRL[IBIRESP].
4. To exit Message mode with the original message, use END type STOP (MCTRL[REQUEST] = 2) with a zero-length message or by using the MCTRL register.

To send a message via Double Data Rate (DDR), follow these steps (from DMA or processor):

1. Write the control request to the Controller Write Message in DDR mode: First Control Word (MWMSG_DDR_CONTROL) register. This request includes:
   - The count of byte-pairs.
   - How to end (on HDR Exit or ready for HDR Restart).
2. Write the second control data to MWMSG_DDR_CONTROL. This second write includes:
   - The address.
   - Read or write.
   - The seven-bit command value.
3. Process the Data:
   - For write operations, write the rest of the data to MWMSG_DDR_DATA. Use the DMA trigger (or interrupt) to keep the Transmit FIFO full, and the controller stops using the data when the program count MWMSG_DDR_CONTROL register reaches 0.
   - For read operations, a DMA trigger (or interrupt) indicates when the RX FIFO is ready to be read out.
4. In-band Interrupt (IBI) behavior is selected in MCTRL[IBIRESP].
5. To exit DDR Message mode with the original message, use END type Exit (MCTRL[REQUEST] = 6) with a zero-length message or by using the MCTRL register.

42.3.2.7 Handing off controllership to another target and getting it back

To hand off controllership, the controller can do one of two things:

- Wait for a Controller Request (CR) (that is, an MSTATUS[IBIWON] = 1 interrupt), indicating that the CR is using MSTATUS[IBITYPE].
- Push the request manually.

In either case, the controller sends a GETACCMST request, which is a directed GET informing the target that the target is being assigned controllership. If the target accepts this request, it returns its Dynamic Address in bits 7:1 and the negative parity of its dynamic address in bit 0. A STOP must be issued, and MCONFIG[MSTENA] must be set to 2 (switching to the Target mode).

To gain controllership, the target sends an CR using the SCTRL register.

- If MCONFIG[MSTENA] is 2, the GETMSTACC CCC accepts.
- If MCONFIG[MSTENA] is not 2, the GETMSTACC CCC refuses.

After controllership has been granted, MSTATUS[NOWMASTER] is 1. The application must enable the MINTSET[NOWMASTER] bit, so the application is interrupted when a controllership transfer occurs.

42.3.2.8 Controller engine flow diagram

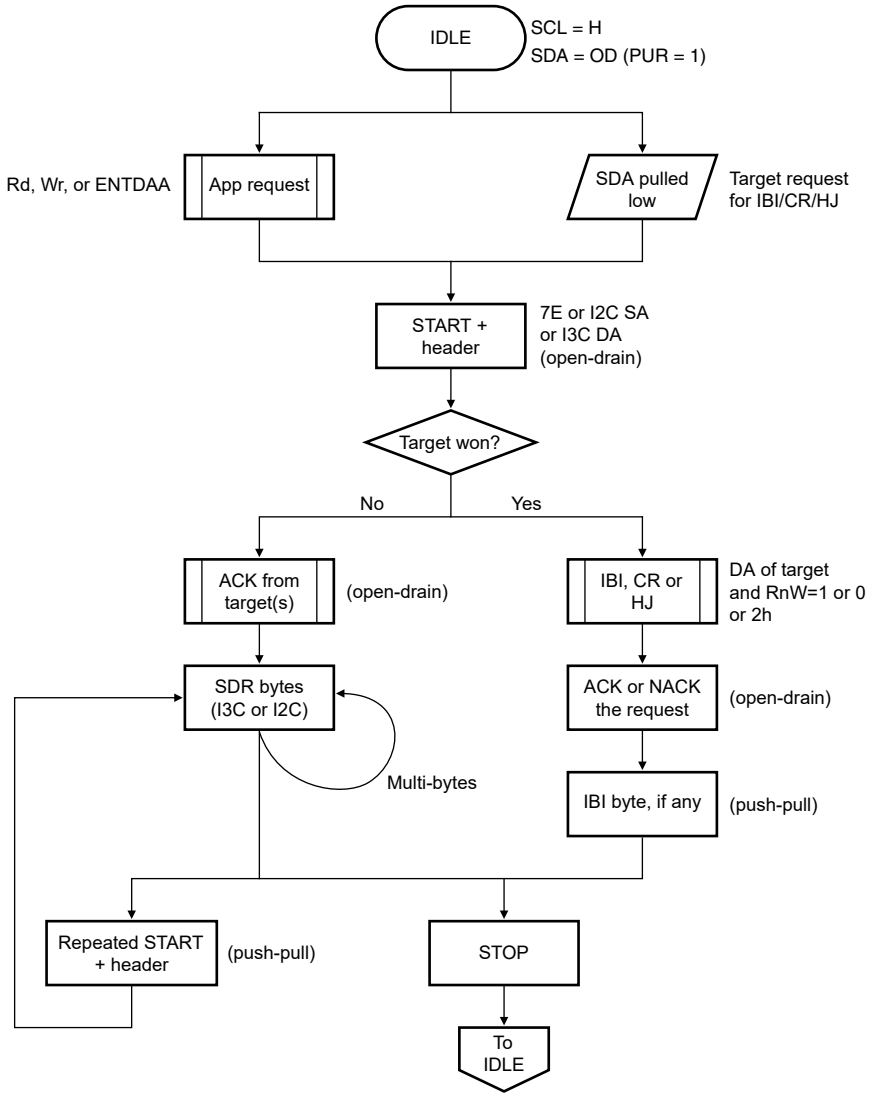


Figure 186. Controller engine flow diagram for SDR (HDR flow progresses from SDR bytes)

42.3.2.9 Target data write from controller

When the address match is valid for the target Dynamic Address and the type is W (write from controller), the target ACKs the address. It does so unless some condition prevents it, such as a full buffer. On ACK, the WRITE state is entered. Once ACKed, it waits for each complete data byte. This process occurs in two steps:

1. Eight bits are clocked in and stored in the next buffer location while in the WRITE state.
2. On the ninth bit:
   - In I3C mode, the W9TH state (In I3C the ninth data bit written by the Controller is the Parity of the preceding eight Data bits.) is used and the parity is checked. The buffer is marked as complete with or without a parity error.
   - In I2C mode, the W9TH state (In I2C, the ninth Data bit written by the Controller is an ACK by the Target.) is used and the data is ACKed and stored.

When the address match is valid for the 7Eh broadcast address, the target acknowledges it. On the first data complete, it sets the in_ccc Broadcast or Direct flag (based on bit 7 of the command). If recognized, it parses the command, setting that bit as well. The

recognized commands are for Dynamic Address work and modes. The CCC and DAA blocks handle the workload afterward. If not supported, then the commands are passed up to the system.

42.3.2.10 Target data write to controller

When the address match is valid for the target Dynamic Address, and the type is R (read by controller), the target ACKs the address. The target does so unless there is no data waiting for the read. On ACK, the READ state is entered.

Once ACKed, it emits the data byte at a time, with the ninth bit using R9TH state. (In I2C, the ninth Data bit from Target to Controller is an ACK by the Controller. In I3C this bit allows the Target to end a Read, and allows the Controller to Abort a Read.)

- In I3C mode, the T bit is emitted to allow the device to indicate whether this byte is the last (from input signal from upper layers). If it is not last byte, the controller may terminate the read via the T bit.
- In I2C mode, the ninth bit allows the controller to terminate via NACK, or else it allows the read to continue via ACK. On completion of each byte read, the done signal is pulsed to get the next byte.

If the read is terminated by the controller unexpectedly (abort in I3C, NACK before END marked), the application is notified.

42.3.3 Clocks

The controller block has two basic clocks feeding:

- The system clock, which controls the access to the memory mapped registers.
- A functional clock (FCLK), which is used to generate the SCL clock rate on the I2C or I3C bus.

I3C supports adjusting clock frequency and accuracy via the Target Time Control Clock (STCCLOCK) register.

CLK_SLOW is used to check conditions like Bus Availability, Bus Free for IBI, or Hot-Join. The slow clock helps to save power.

- To determine the Bus Available condition for IBI, the target needs a clock to generate the ~1 µs timing. To support this requirement, a slow clock (CLK_SLOW) can be provided to save on power.
- Additionally, the block provides a slow clock gate (slow_gate). If CLK_SLOW can be turned off, slow_gate is 1.
- I3C supports a counter field, SCONFIG[BAMATCH], to calculate the Bus Available condition. BAMATCH provides the count of the slow clock. This field counts 1 µs or more to allow an IBI to drive SDA low when the controller is free. The maximum width and maximum values are controlled by the I3C module.

To provide the slow clock, the system has three basic choices:

- Use the same clock as CLK, which may be divided. If it is variant, the slow match register can be used to change the match rule.
- Provide a truly slow clock, which makes it inexpensive in power to count 1 µs or more.
- Provide some other clock (other than CLK or a truly slow clock), but use the slow_gate signal to gate the clock near its source.

I3C supports a timeout when stalled too long in a frame. This timeout occurs when:

- The Transmit FIFO or Receive FIFO is not handled and the bus is stuck in the middle of a message.
- No STOP was issued and the bus is between messages.
- IBI manual is used and no decision was made.

This model is used with the parameters for the width of the counter and the match rule.

42.3.4 Resets

The I3C resets are:

- Global/system reset fed into block. Everything is reset by this global/system reset. Release is assumed to be synchronized to the system clock (if provided). It is not synchronized to SCL or SDA. Those signals should not be active when the block is released, otherwise I3C would be in Hot-Join mode and therefore not active.
- A software reset is provided by the I3C block.
- STOP state is reset by a START as well.

42.3.5 Interrupts

This section describes all the interrupts (IRQs) that the I3C module generates. All status interrupts are updated in the Controller Status (MSTATUS) and Target Status (SSTATUS) registers.

Supported interrupts occur:

- For pending IBI, CR, or Hot-Join that have been sent by a target.
- When a CCC is received and is handled by the block (SSTATUS[CHANDLED]).
- When an error or warning has occurred, such as data underrun, data overrun, parity error, HDR-DDR, or CRC error.

42.4 External signals

This table describes the I3C module signals.

Signal	I/O	Description
SCL	I/O	Serial clock
SDA	I/O	Serial data
PUR	O	Pull up resistance. There is internal pull-up resistance on SDA, which is controlled by the I3C controller. If the internal pullup is not enough, PUR can be used to control an external pull-up resistance on SDA actively.

42.5 Initialization

42.5.1 Configuration initialization

The configuration is handled when initializing the I3C module. Configuration is done using the Controller Configuration (MCONFIG) register, which controls the frequencies, duty cycle, optimizations for performance, and other parameters.

Parameters govern several features that are supported in the design.

High-keeper controls are also included in the below configuration list.

Table 350. Configuration parameters used to initialize the I3C module

Configuration parameter		Description
MSTENA	Controller Enable	Determines whether the I3C peripheral starts in Controller mode (Main Controller in I3C terms) or starts in Target mode (and switches to Controller mode later).
HKEEP	High Keeper	Determines how the high-keeper (weak pullup) is implemented, depending on the device capabilities.

Table continues on the next page...

Table 350. Configuration parameters used to initialize the I3C module (continued)

Configuration parameter		Description
PPBAUD	Push-Pull Baud	Sets the push-pull frequency as a divider from the FCLK (alternative clock fed to the peripheral). This frequency sets the SCL half-clock period baseline (used at least for the high time of SCL). The formula for SCL in Push-Pull mode using PPBAUD and PPLOW is: SCL freq (in MHz) = FCLK / ((2 + 2 * PPBAUD) + PPLOW) If PPLOW is 0, then SCL high = SCL low (in ns) = (1 + PPBAUD) * 1000 / FCLK (in MHz) Examples: • If FCLK = 24 MHz, then PPBAUD = 0 yields 12 MHz (42.67 ns per half period) • If FCLK = 50 MHz, then PPBAUD = 1 yields 12.5 MHz (20 + 20 = 40 ns per half period)
PPLOW	Push-Pull Low	Changes the duty cycle for Push-Pull. It indicates how many more FCLK cycles to use for low. The formula for SCL in Push-Pull mode using PPBAUD and PPLOW is: SCL freq (in MHz) = FCLK / ((2 + 2 * PPBAUD) + PPLOW) If PPLOW is non-zero, then • SCL High (in ns) = (1 + PPBAUD) * 1000 / FCLK (in MHz) • SCL Low (in ns) = (1 + PPBAUD + PPLOW) * 1000 / FCLK (in MHz) For example, when FCLK is 50 MHz, PPBAUD is 1, and PPLOW is 1, then the periods are: • (1 + 1) * 1000 / 50 = 20 + 20 = 40 ns high • (1 + 1 + 1) * 1000 / 50 = 20 + 20 + 20 = 60 ns low This timing is equivalent to 10 MHz SCL, but this timing maintains the 40 ns high needed so I2C devices do not see the high periods. **NOTE** The PPLOW value does not have any impact on Open-Drain mode and I2C mode SCL rate calculations.
ODBAUD	Open Drain Baud	The number of PPBAUD periods to make up one I3C Open-Drain half-clock baseline. If ODHPP is 0, the ODBAUD half-clock time period = (ODBAUD + 1) * PPBAUD high period If ODHPP is 0, the formula for SCL in Open-Drain mode is: SCL (in MHz) = FCLK / (2 + 2 * PPBAUD) * (ODBAUD + 1) Target to get open-drain half-clock period is 200 ns. For example: • If PPBAUD yields 12.5 MHz (40 ns per PPBAUD period), then ODBAUD = 4 can be used to get 200 ns. See also ODHPP for details on short High and long Low. • If PPBAUD = 1, FCLK = 50 MHz, and ODBAUD = 4, then open-drain SCL = 50 / (2 + 2 * 1) * 5 = 2.5 MHz.
ODHPP	Open-Drain High Push-Pull	Optional field that allows the I3C open-drain to be long low and short high. The High period of SCL is the PPBAUD period. This period leaves enough time for the pull-up resistor to

Table continues on the next page...

Table 350. Configuration parameters used to initialize the I3C module (continued)

Configuration parameter		Description
		pull the SDA high when SCL is low. It is also quick when SCL is high and there are no changes happening. ODBAUD low half-clock time period = (ODBAUD + 1) * PPBAUD high period. ODBAUD high half-clock time period = PPBAUD high period If ODHPP is 1, the formula for SCL in Open-Drain mode is: SCL (in MHz) = FCLK / (1 + PPBAUD) * (ODBAUD + 1) + (1 + PBAUD) For example: • If PPBAUD produces 12.5 MHz (40 ns per PPBAUD period), then ODBAUD = 4 and ODHPP = 1. These settings provide a high period of 40 ns and a low period of 200 ns. • If PPBAUD = 1, FCLK = 50 MHz, and ODBAUD = 4 then open-drain SCL = 50 / (1 + 1) * 5 + (1 + 1) = 4.16 MHz.
I2CBAUD	I2C Baud	Indicates how many ODBAUD periods are required to communicate with I2C devices (MCTRL[TYPE] = 2 or MWMSG_SDR_CONTROL[I2C] = 1). For example, if ODBAUD gives 200 ns, and the goal is Fm+ (Fast Mode, 1 MHz), then the sum must be 1 µs. I2CBAUD acts differently for odd and even values. For example, if I2CBAUD = 3, it gives 3 ODBAUD periods low and 2 ODBAUD periods high. • If I2CBAUD = 4, then it gives 3 ODBAUD periods for low and 3 ODBAUD periods for high. • Also, if I2CBAUD = 3, this yields 200 * 3 = 600 ns and 200 * 2 = 400 ns, with the sum 600 + 400 = 1000 ns = 1 µs.
SKEW	Skew	The normal SDA skew from SCL is handled by using the time for the SCL to reach its pad and come back to the design. This time is normally 2 ns to 5 ns (or sometimes more). If the SKEW is too fast, then add more delay, the SKEW allows specifying the number of FCLKs to insert.

Additional optimizations:

- Use MIBIRULES[MSB0] to obtain faster START header times. If the controller application assigns all I3C dynamic addresses to be less than 40h (it does not have MSB set), MIBIRULES[MSB0] can be set. When the controller emits 7Eh (broadcast) and a target does not drive the first bit low, the rest of the header can be at push-pull speeds. This speed is two times faster or more, depending on optimizations.
- Auto-emit 7Eh speeds up the frame when used with MIBIRULES[MSB0]. It allows the processor to sleep when the frame starts automatically (in response to a target).

42.5.2 Interrupt service flow

Set up interrupts from Controller Status (MSTATUS):

Table 351. MSTATUS fields

Field	Description
STATE	State Of The Controller. Indicates the current controller state.
BETWEEN	Between. Between messages or Dynamic Address Assignments (DAA).

Table continues on the next page...

Table 351. MSTATUS fields (continued)

Field	Description
NACKED	Not Acknowledged. Indicates whether the last Start and Address sequence was NACKed (was not ACKed by the addressed target).
IBITYPE	In-Band Interrupt (IBI) Type. Indicates the type of IBI of the last event that won the arbitration, whether the interrupt is ACKED or NACKED or pending.
SLVSTART	Target Start. Indicates whether a target is/was requesting a START by holding SDA low.
MCTRLDONE	Controller Control Done. Indicates whether the module has completed an MCTRL request.
COMPLETE	Complete. Indicates whether a message has completed.
RXPEND	Receive Pending. Indicates whether a message is being received from a target and bytes are in the input buffer or FIFO.
TXNOTFULL	TX Buffer or FIFO Not Full. Indicates whether the buffer, FIFOm or message register can accept another byte or halfword.
IBIWON	In-Band Interrupt (IBI) Won. Indicates whether an IBI, CR, or HJ has won the arbitration on a header address, regardless of whether it was NACKed or ACKed.
ERRWARN	Error Or Warning. Indicates whether an error occurred, such as improper register use, overrun or underrun of FIFO or buffer, or invalid parity or CRC in a DDR read.
NOWMASTER	Module Is Now Controller. Indicates when the module is now a controller. That is, it was previously a target, controllership acceptance was requested from the previous controller, and controllership was accepted.
IBIADDR	IBI Address. Indicates the address of the IBI, CR, or 7'h2 when HJ.

Set up interrupts from Target Status (SSTATUS):

Table 352. SSTATUS fields

Field	Description
STNOTSTOP	Status Not Stop. After a TE0 or TE1 error, when the I3C module is waiting for an Exit Pattern.
STMSG	Status Message. Bus target is listening for the next repeated START or STOP on the bus traffic.
STCCCH	Status Common Command Code Handler. CCC message is being handled automatically.
STREQRD	Status Request Read. REQ in process is an SDR read from this target, or an In-Band Interrupt (IBI) is being pushed out.
STREQWR	Status Request Write. REQ in process is SDR write data from the controller to this bus target (or all I3C targets), but not in ENTDAA mode.
STDAA	Status Dynamic Address Assignment. I3C bus is in Enter Dynamic Address Assignment (ENTDAA) mode, regardless of whether this bus target has a Dynamic Address or not.
STHDR	Status High Data Rate. The I3C bus is in HDR-DDR mode.
START	Start. A START or repeated START was seen after the START bit was last cleared.
MATCHED	Matched. Incoming header matched the I3C Dynamic or I2C Static address of this device (if any) since the bus was last cleared.
STOP	Stop. Stopped state detected. A STOP state was present on the bus since the bus was last cleared.

Table continues on the next page...

Table 352. SSTATUS fields (continued)

Field	Description
RX_PEND	Received Message Pending. Indicates when receiving a message from the controller that is not being handled by the I3C module (not a CCC message).
TXNOTFULL	Transmit Buffer Is Not Full. To-bus buffer or FIFO can accept more data to be transmitted.
DACHG	Dynamic Address Change. Indicates occurrence of dynamic address (DA) change. Actual DA can be seen in the DYNADDR register. Also used when the MAP Auto feature is configured.
CCC	Common Command Code. Indicates whether a CCC has been received, and is not handled by the I3C module.
ERRWARN	Error Warning. An error or warning has occurred, such as data underrun, data overrun, parity error, HDR-DDR CRC error, or other error or warning condition.
HDRMATCH	High Data Rate Command Match. Indicates whether HDR command matched the I3C Dynamic Address of this device.
CHANDLED	Common Command Code Handled. Indicates whether a CCC is being handled by the module.
EVENT	Event. For a target, indicates that a pending In-Band Interrupt (IBI), Controller Request (CR), or Hot-Join (HJ) has been sent as requested.
EVDET	Event Details. Current details of the last (SSTATUS[EVENT] = 1) or pending event.
IBIDIS	In-Band Interrupts Are Disabled. No response to CTRL requests.
MRDIS	Controller Requests Are Disabled. No response to CTRL requests.
HJDIS	Hot-Join Disabled. No response to CTRL requests.
ACTSTATE	Activity State from Common Command Codes (CCC). Latency of normal to 10 seconds.
TIMECTRL	Time Control. Indicates whether time control is enabled, and in which mode.

42.6 Application information

This section describes applications supported by the I3C module.

42.6.1 I2C configuration for meeting timing requirement for FM and FM+ modes

I2C FM and I2C FM+ mode is supported as per the I3C standard specifications.

Configuration for FM mode (Below configurations are to meet close to 400 kMhz) But In this case T_{SU_STA} is not met:

Table 353. Configuration for FM mode, where frequency is close to 400 kHz but timing requirement is not met

FCLK	PPBAUD	ODBAUD	I2CBAUD	SCL FREQ	Timing Requirement* violations
24 MHz	0	4	11	370 kHz	T_{SU_STA} - 353 ns
48 MHz	1	4	11	370 kHz	T_{SU_STA} - 353 ns
134 MHz	5	4	11/12	412 kHz / 382 kHz	T_{SU_STA} - 267 ns and T_{SU_STO} - 529 ns
160 MHz	6	4	10	380 kHz	T_{SU_STA} - 312 ns

Table 354. Configuration for FM mode, where all timing requirements are met

FCLK	PPBAUD	ODBAUD	I2CBAUD	SCL FREQ	Timing Requirement* met?
24 MHz	1	2	9	363 kHz	Yes
48 MHz	3	2	9	363 kHz	Yes
134 MHz	11	2	8	372 kHz	Yes
160 MHz	13	2	9/8	345 kHz / 422 kHz	Yes

Configuration for FM+ mode, where the frequency is close to 1 MHz. Although, in this case the T_{SU_STA} timing requirement is not met.

Table 355. Configuration for FM+ mode, where frequency is close to 1 MHz but timing requirement is not met

FCLK	PPBAUD	ODBAUD	I2CBAUD	SCL FREQ	Timing Requirement* violations
24 MHz	0	4	3	960 kHz	T_{SU_STA} - 228 ns
48 MHz	1	4	3	960 kHz	T_{SU_STA} - 228 ns
134 MHz	4	4	3	1.087 MHz	T_{SU_STA} - 156 ns
160 MHz	6	4	3	912 kHz	T_{SU_STA} - 181 ns

To meet all the timing specifications, frequency may differ from 1 MHz. Use the following configurations.

Table 356. Configuration for FM+ mode, where all timing requirements are met

FCLK	PPBAUD	ODBAUD	I2CBAUD	SCL FREQ	Timing Requirement* met?
24 MHz	0	3	6	749 kHz	Yes
48 MHz	1	3	6	749 kHz	Yes
134 MHz	4	3	6	839 kHz	Yes
160 MHz	6	3	6	713 kHz	Yes

NOTE

The I3C controller module provides an option to slow down the clock for standard speed mode to support legacy I2C standard targets as well. Although, this option may not necessarily meet the exact timing requirement of the standard mode per the standard I2C timing specification.

For the timing requirements under consideration, see the "I3C Timing Requirements When Communicating With I2C Legacy Devices" table in the MIPI I3C v1.1.1 specification available on www.mipi.org.

42.7 I3C register descriptions

This section describes the registers in the I3C module.

Table 357. Register Types

Type	Description	Notes
RO	Read-Only	
RW	Read/Write	
W1C	Write 1 to Clear	W1S and W1C registers are readable.
W1S	Write 1 to Set	
WO	Write-Only	

NOTE

Writing to a Read-Only (RO) register, except the Controller Interrupt Mask (MINTMASKED) register, causes bus errors. This module does not verify that programmed values in the registers are correct. The application software must ensure that valid programmed values are being written.

However, the I3C peripheral ignores writes to RO registers, and the I3C peripheral returns 0 for reads from WO or nonexistent registers. It requires all 32-bit registers to be read and written as 32-bit and aligned to 32 bits. The peripheral uses the Advanced Peripheral Bus (APB), so the peripheral does not know whether partial read or write operations (less than 32 bits) are used.

42.7.1 I3C memory map

I3C0 base address: 4001_6000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Controller Configuration (MCONFIG)	32	See section	0000_0000
4	Target Configuration (SCONFIG)	32	See section	0001_0000
8	Target Status (SSTATUS)	32	See section	0000_1400
C	Target Control (SCTRL)	32	See section	0000_0000
10	Target Interrupt Set (SINTSET)	32	See section	0000_0000
14	Target Interrupt Clear (SINTCLR)	32	See section	See section
18	Target Interrupt Mask (SINTMASKED)	32	See section	0000_0000
1C	Target Errors and Warnings (SERRWARN)	32	See section	0000_0000
20	Target DMA Control (SDMACTRL)	32	See section	0000_0010

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
2C	Target Data Control (SDATACTRL)	32	See section	8000_0030
30	Target Write Data Byte (SWDATAB)	32	See section	See section
34	Target Write Data Byte End (SWDATABE)	32	See section	See section
38	Target Write Data Half-word (SWDATAH)	32	See section	See section
3C	Target Write Data Half-word End (SWDATAHE)	32	See section	See section
40	Target Read Data Byte (SRDATAB)	32	See section	0000_0000
48	Target Read Data Halfword (SRDATAH)	32	See section	See section
54	Target Write Data Byte (SWDATAB1)	32	See section	0000_0000
5C	Target Capabilities 2 (SCAPABILITIES2)	32	See section	0000_0300
60	Target Capabilities (SCAPABILITIES)	32	See section	E83F_FE78
64	Target Dynamic Address (SDYNADDR)	32	See section	0000_0000
68	Target Maximum Limits (SMAXLIMITS)	32	See section	0000_0000
6C	Target ID Part Number (SIDPARTNO)	32	RW	0000_0000
70	Target ID Extension (SIDEXT)	32	See section	0000_EF00
74	Target Vendor ID (SVENDORID)	32	See section	0000_011B
78	Target Time Control Clock (STCCLOCK)	32	See section	0000_0214
7C	Target Message Map Address (SMSGMAPADDR)	32	See section	0000_0000
84	Controller Control (MCTRL)	32	See section	0000_0000
88	Controller Status (MSTATUS)	32	See section	0000_1000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
8C	Controller In-band Interrupt Registry and Rules (MIBIRULES)	32	RW	0000_0000
90	Controller Interrupt Set (MINTSET)	32	See section	0000_0000
94	Controller Interrupt Clear (MINTCLR)	32	See section	See section
98	Controller Interrupt Mask (MINTMASKED)	32	See section	0000_0000
9C	Controller Errors and Warnings (MERRWARN)	32	See section	0000_0000
A0	Controller DMA Control (MDMACTRL)	32	See section	0000_0010
AC	Controller Data Control (MDATACTRL)	32	See section	8000_0030
B0	Controller Write Data Byte (MWDATAB)	32	See section	See section
B4	Controller Write Data Byte End (MWDATABE)	32	See section	See section
B8	Controller Write Data Halfword (MWDATAH)	32	See section	See section
BC	Controller Write Data Halfword End (MWDATAHE)	32	See section	See section
C0	Controller Read Data Byte (MRDATAB)	32	See section	0000_0000
C8	Controller Read Data Halfword (MRDATAH)	32	See section	0000_0000
CC	Controller Write Byte Data 1(to bus) (MWDATAB1)	32	See section	0000_0000
D0	Controller Write Message Control in SDR mode (MWMSG_SDR_CONTROL)	32	See section	See section
D0	Controller Write Message Data in SDR mode (MWMSG_SDR_DATA)	32	See section	0000_0000
D4	Controller Read Message in SDR mode (MRMSG_SDR)	32	See section	0000_0000
D8	Controller Write Message in DDR mode: First Control Word (MWMSG_DDR_CONTROL)	32	See section	See section
D8	Controller Write Message in DDR mode Control 2 (MWMSG_DDR_CONTROL2)	32	See section	See section

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
D8	Controller Write Message Data in DDR mode (MWMSG_DDR_DATA)	32	See section	0000_0000
DC	Controller Read Message in DDR mode (MRMSG_DDR)	32	See section	0000_0000
E4	Controller Dynamic Address (MDYNADDR)	32	See section	0000_0000
11C	Map Feature Control 0 (SMAPCTRL0)	32	See section	0000_0000
140	Extended IBI Data 1 (IBIEXT1)	32	See section	0000_0070
144	Extended IBI Data 2 (IBIEXT2)	32	RW	0000_0000
FFC	Target Module ID (SID)	32	RO	EDCB_0100

42.7.1.1 Controller Configuration (MCONFIG)

Controls all controller states when controller operation is enabled. The MCONFIG register should not be changed when an active transaction is occurring.

Offset

Register	Offset
MCONFIG	0h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	I2CBAUD				SKEW			ODHPP	ODBAUD							
Reset	0	0	0	0	0	0	0	0	0	0	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	PPLOW				PPBAUD				Reserved	ODSTOP	HKEEP		DISTO	0	MSTENA	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-28 I2CBAUD	I2C Baud Rate The I2C low and high in terms of number of ODBAUD counts. • I2CBAUD >> 1 is the main count load, and it is *count - 1* . So, for I2CBAUD >> 1: I2CBAUD = 0 for one ODBAUD beat, and I2CBAUD = 1 for two ODBAUD beats. • If I2CBAUD[28] is 1, then the low time has one extra ODBAUD beat. The I2CBAUD field is normally 3, where ODBAUD gives 200 ns. For I2CBAUD >> 1, I2CBAUD = 1, which means two ODBAUD beats. To meet the requirements for Fast Mode Plus (Fm+), (2 + 1) * 200 = 600 ns low, with 2 * 200 = 400 ns high for 1 s period. For Fast Mode (FM), I2CBAUD is normally 11 (giving 2.6 s) or 6 (giving 2.4 s).
27-25 SKEW	Skew Number of FCLK counts for an SDA change after SCL for I3C push-pull. This skew is in addition to the roundtrip of the SCL line from the pad back to the module (6 ns or more). • SKEW is normally not needed, so assign SKEW = 0. • SKEW is only used if SDA is not naturally skewing from an SCL change. • I2C automatically skews SDA (but not PUR) to match I2C rules.
24 ODHPP	Open Drain High Push-Pull See Sending a CCC to I3C targets for open-drain timing check upon the first 7Eh broadcast allowing I3C peripherals acting as I2C legacy devices to turn off their I2C 50-ns spike filters. 0 - ODHPP disabled. Open-Drain SCL High half-clock period is the same as the Open-Drain Low SCL half-period. 1 - ODHPP enabled. Open-Drain High SCL half-lock period is one PPBAUD count for I3C messages. This setting is faster (and works for I3C devices). Any legacy I2C devices on the bus will not see the SCL High at all (less than the spike filter period).
23-16 ODBAUD	Open Drain Baud Rate ODBAUD = (in terms of number of PPBAUD counts) - 1. ODBAUD should not be 0; this setting is not the same as push-pull. See Configuration initialization. This rate is usually 200 ns (see I2CBAUD for I2C counts). When used with ODHPP, this setting produces 250 ns per clock in I3C. In this case, if PPBAUD provides 12 MHz, then one PPBAUD count is half of 12 MHz, or 41.67 ns. To obtain a rate around 200 ns, use a value of 5 - 1 = 4 for ODBAUD.
15-12 PPLOW	Push-Pull Low Adder for push-pull low, to create a duty-cycle with a longer low period, with up to 15 more FCLK cycles low than high. PPLOW = 0 produces a 50/50 duty cycle.
11-8 PPBAUD	Push-Pull Baud Rate The number of FCLK counts makes each push-pull low and normally high period. PPBAUD = 0 when run at 1/2 input FCLK speed. For example, a 24 MHz FCLK produces a 12 MHz SCL, because each FCLK is SCL Low or SCL High.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	Note the effect Push-Pull Low (PPLOW) has upon the duty cycle. For example, 24 MHz with 50/50 duty cycle is 12 MHz. However, when PPLOW adds 3 more low beats, the push-pull baud rate becomes 4.8 MHz (from 24 MHz or 5 beats).
7 ——	Reserved
6 ODSTOP	Open Drain Stop 0 - Disable open-drain stop. ODSTOP must be disabled when sending an HDR exit pattern. 1 - Enable open-drain stop. STOP is emitted at open-drain speeds even for I3C messages. In legacy devices, this feature can ensure that the legacy devices see the STOP.
5-4 HKEEP	High-Keeper Indicates how High-Keeper is supported. **NOTE** Your specific device may not support any or all High-Keeper methods. Use HKEEP = 2 when I3C. Use HKEEP = 0/1 when I2C. 00 - NONE. Use PUR (Pull-Up Resistor). No separate pin_HK_SDA/pin_HK_SCL is used. Only pin_PUR_oena is used for sda. Hold pin_SCL_oena High in this mode, SCL clock is push-pull and pin_SCL_out toggles which is actual clock. 01 - WIRED_IN. High Keeper controls, use pin_HK_SDA/pin_HK_SCL (High Keeper) controls. SCL may use an HK or that signal (pin_HK_SCL) may only OR into pin_SCL_oena. Uses pin_HK_SDA as well for sda high keeper, along with pin_PUR_oena. 10 - PASSIVE_SDA. Passive on SDA, can Hi-Z (high impedance) for Bus Free (IDLE) and hold. The pins, pin_HK_SDA and pin_HK_SCL are not used, Only a combination of SDA_oena and pin_PUR_oena are used. 11 - PASSIVE_ON_SDA_SCL. Passive on SDA and SCL, can Hi-Z (high impedance) both for Bus Free (IDLE), and can Hi-Z SDA for hold. This is for I2C clock stretching mode, where both SCL and SDA are open drain.
3 DISTO	Disable Timeout Disables the timeout that produces application errors. If the controller has been left in a state other than Stopped for more than 100 s (because 10 kHz is the slowest allowed I3C speed), the timeout sends a MERRWARN interrupt. To prevent the MERRWARN interrupt during development or testing, write 1 to DISTO to disable the timeout. In systems that support timeouts, timeout is disabled automatically during debug. 0 - Timeout enabled 1 - Timeout disabled, if timeout is configured
2	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
1-0 MSTENA	Controller Enable Indicates whether the controller is enabled and what states the controller can use. 00 - CONTROLLER_OFF. Controller is disabled. The I3C module can only use Target mode. 01 - CONTROLLER_ON. Controller is enabled. When used upon start-up, this I3C module is the Main Controller by default. The module controls the bus unless the controller is handed off. If the controller is handed off, then MSTENA must become 2, so that it has the capability to become the master again. Performing the handoff means emitting GETACCMST CCC command. If the command is accepted, the module emits a STOP and sets the MSTENA field to 2 (or 0). 10 - CONTROLLER_CAPABLE. The I3C module is controller-capable, but the module is operating as a target now. When used from the start, the I3C module starts as a target, but is prepared to switch to Controller mode. To switch to Controller mode, the target emits a Controller Request (CR), or receives a GETACCMST CCC command and accepts it (to switch on the STOP). 11 - I2C_CONTROLLER_MODE. (legacy I2C) This mode uses an open-drain clock (to allow clock stretching) and relies on passive pullup-resistors on both SCL and SDA. This mode only uses I2CBAUD. START requests can only be made for I2C type, and I3C requests are not accepted. If there are only I2C targets on the bus and no clock stretching, use type 1 instead. Type 1 has a push-pull SCL clock, which is faster and uses less power.

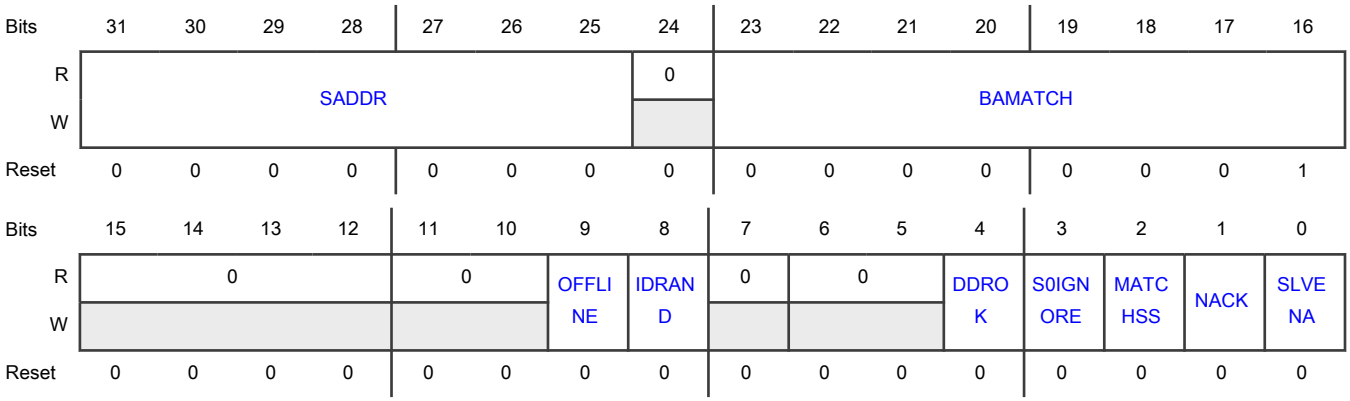
42.7.1.2 Target Configuration (SCONFIG)

Contains fields that must be configured before the module is activated.

Offset

Register	Offset
SCONFIG	4h

Diagram



Fields

Field	Description
31-25 SADDR	Static Address Sets the I2C 7-bit static address, otherwise must be 0.
24 —	Reserved
23-16 BAMATCH	Bus Available Match The Bus Available condition match value for the current slow clock. BAMATCH provides the count of the slow clock to count out 1 s (or more) to allow an In-Band Interrupt (IBI) to drive SDA low when the controller is not doing so. The maximum width and maximum values are controlled by the I3C module.
15-12 —	Reserved
11-10 —	Reserved
9 OFFLINE	Offline If OFFLINE = 1 when the target enable (SCONFIG[SLVENA]) is set to 1, then the I3C module waits for either 60 s of bus quiet or an HDR Exit Pattern. This waiting ensures that the bus is not in HDR mode, and so can safely monitor for the next activity in Single Data Rate (SDR) mode. 0 - Disable 1 - Enables wait to ensure the bus is not in HDR mode.
8 IDRAND	ID random 0 - SIDPARTNO[PARTNO] is a part number and an instance. 1 - SIDPARTNO[PARTNO] is a random value.
7 —	Reserved
6-5 —	Reserved
4 DDROK	Double Data Rate OK If DDROK = 1, write 1 to the corresponding SIDEXT[BCR] bit (Bus Characteristics Register bit in Target ID Extension Register) to indicate that High Data Rate (HDR) is available, and configure the corresponding HDRCAP HDR-DDR bit to allow using Double Data Rate (DDR) mode. **NOTE** The DDROK bit must be 1 before the target can connect to the I3C bus. The target peripheral indicates to the controller whether the target can support the feature during dynamic address assignment.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - Do not allow HDR-DDR messaging. 1 - Allow HDR-DDR messaging.
3 S0IGNORE	Ignore TE0/TE1 Errors 0 - Do not ignore TE0/TE1 errors 1 - Ignore TE0/TE1 errors. Target does not detect TE0 or TE1 errors, so it does not lock up waiting on an Exit Pattern. This setting should only be used when the bus does not use HDR mode.
2 MATCHSS	Match START or STOP 0 - Match START or STOP disable 1 - Match START or STOP enable. START and STOP sticky SSTATUS bits only become 1 when SSTATUS[MATCHED] is 1. This setting allows START and STOP to be used to detect the end of a message to/from this target.
1 NACK	Not Acknowledge 0 - Always NACK disable 1 - Always NACK enable. The target rejects all requests to it, except for a Common Command Code (CCC) broadcast. NACK = 1 should be used with caution, because the controller may decide that the target is missing, if NACK is overused.
0 SLVENA	Target Enable SLVENA must not be set before registers like SCONFIG (SIDPARTNO, SIDEXT, and others) are set, because these registers affect the data to and from the controller. Target enable is configured just once before the I3C bus comes up. If target enable is used at other times, see Hot-Join. In the case of Hot-Join, the Hot-Join bit (SCAPABILITIES[IBI_MR_HJ]) must be 1 before writing 1 to the target enable bit (SLVENA), so that the device does not see a START or STOP incorrectly. 0 - Target ignores the I2C or I3C bus 1 - Target can operate on the I2C or I3C bus

42.7.1.3 Target Status (SSTATUS)

Not all bits are used if the module only acts as a target. The Target Status register indicates sticky status for interrupts and "states" and "modes" related to the I3C bus. The fields are divided into current activity, interrupt maskable actions, then states and modes on the bus.

Offset

Register	Offset
SSTATUS	8h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TIMECTRL		ACTSTATE		HJDIS	0	MRDIS	IBIDIS	0		EVDET		0	EVENT	CHANDLED	HDRMATCH
W														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	ERRWARN	CCC	DACHG	TXNOTFU...	RX_PEND	STOP	MATCHED	START	0	STHDR	STDAA	STREQWR	STREQRD	STCCCH	STMSG	STNOTST...
W		W1C	W1C			W1C	W1C	W1C								
Reset	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-30 TIMECTRL	Time Control Indicates whether time control is enabled, and in which mode 00 - NO_TIME_CONTROL. No time control is enabled 01 - SYNC_MODE. Synchronous mode is enabled 10 - ASYNC_MODE. Asynchronous standard mode (0 or 1) is enabled 11 - BOTHSYNCASYNC. Both Synchronous and Asynchronous modes are enabled
29-28 ACTSTATE	Activity State from Common Command Codes (CCC) 00 - NO_LATENCY. Normal bus operations 01 - LATENCY_1MS. 1 ms of latency 10 - LATENCY_100MS. 100 ms of latency 11 - LATENCY_10S. 10 seconds of latency
27 HJDIS	Hot-Join Disabled While Hot-Join is disabled, CTRL requests are not responded to. 0 - Hot-Join not disabled 1 - Hot-Join disabled
26 —	Reserved
25 MRDIS	Controller Requests Are Disabled While Controller Requests are disabled, CTRL requests are not responded to. 0 - Controller Requests not disabled 1 - Controller Requests disabled

Table continues on the next page...

Table continued from the previous page...

Field	Description
24 IBIDIS	In-Band Interrupts Are Disabled While In-Band Interrupts are disabled, CTRL requests are not responded to. 0 - In-Band Interrupts not disabled 1 - In-Band Interrupts disabled
23-22 —	Reserved
21-20 EVDET	Event Details Current details of the last (SSTATUS[EVENT] = 1) or pending event. 00 - NONE. No event or no pending event 01 - NO_REQUEST. Request is not sent yet. Either there was no START yet, or is waiting for Bus-Available or Bus-Idle (HJ). 10 - NACKED. Not acknowledged (request sent and rejected). The module will try again. 11 - ACKED. Acknowledged (request sent and accepted), so done (unless the time control data is still being sent).
19 —	Reserved
18 EVENT	Event For a target, indicates that a pending In-Band Interrupt (IBI), Controller Request (CR), or Hot-Join (HJ) has been sent as requested. See the upper status register fields for details. EVENT only asserts when ACKed by a controller. 0 - No event has occurred. 1 - An IBI, CR, or HJ has occurred.
17 CHANDLED	Common Command Code Handled Indicates whether a CCC is being handled by the module. This field is a notification only, but it may result in updates to the SSTATUS register. 0 - CCC handling not in progress. 1 - CCC handling in progress.
16 HDRMATCH	High Data Rate Command Match Indicates whether HDR command matched the I3C Dynamic Address of this device. The HDR command is available as the first byte and RXPEND are set. The MSB of the command byte indicates whether it is a read or a write command. If the HDR command is a read, and there are to-bus bytes waiting, then the command is ACKed and the data is sent back. Otherwise, the HDR command is NACKED.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	NOTE When HDRMATCH is 1, ERRWARN should be checked, because the HPAR error may be encountered after signaling this HDR command. The parity is after the destination address and command. 0 - HDR command did not match. 1 - HDR command matched the I3C Dynamic Address of this device.
15 ERRWARN	Error Warning An error or warning has occurred, such as data underrun, data overrun, parity error, HDR-DDR CRC error, or other error or warning condition. See the Target Errors and Warnings (SERRWARN) .
14 CCC	Common Command Code Indicates whether a CCC has been received, and is not handled by the I3C module. There are 2 types of Common Command Codes: • Broadcast CCC. Corresponds with RXPEND, and the first byte is the CCC (command). • Direct CCC, which may never be directed to this device. If Direct CCC are directed to this device, then the TXSEND or RXPEND are triggered, and the RXPEND contains the command. 0 - No CCC received. 1 - CCC received.
13 DACHG	Dynamic Address Change Indicates occurrence of dynamic address (DA) change. Actual DA can be seen in the DYNADDR register. This field is also used when the MAP Auto feature is configured, changing one or more MAP items. See DYNADDR and MAPCTRLn. DYNAADDR for the main DA (0) indicates whether last change was due to Auto-MAP. 0 - No DA change detected. 1 - DA change detected. The target DA has been assigned, re-assigned, or reset (lost) and is now in the state of being valid or none.
12 TXNOTFULL	Transmit Buffer Is Not Full Indicates whether transmit buffer is not full. If DMA is enabled for transmitting, then it will also be signaled to provide more data. 0 - Transmit buffer full 1 - Transmit buffer not full. To-bus buffer or FIFO can accept more data to be transmitted. For all but External FIFO, this process uses SDATACTRL[TXTRIG] , which defaults to not-full.
11 RX_PEND	Received Message Pending Indicates when receiving a message from the controller that is not being handled by the I3C module (not a CCC message). Such messages are internally processed by the module. For all but External FIFO, this process uses SDATACTRL[RXTRIG] , which defaults to not-empty. If DMA is enabled for receiving, then

Table continues on the next page...

Table continued from the previous page...

Field	Description
	DMA is signaled as well. RX_PEND automatically becomes 0 if data is read (from FIFO and non-FIFO sources). 0 - No received message is pending. 1 - Received message is pending.
10 STOP	Stop Detects stopped state. The STNOTSTOP state also indicates when the I3C module is in stop mode. A fast STOP/START combination may not trigger the STOP status. In that case, START is always set. 0 - No STOP detected. 1 - Stopped state detected. A STOP state was present on the bus since the bus was last cleared.
9 MATCHED	Matched 0 - No header matched. 1 - An incoming header matched the I3C Dynamic or I2C Static address of this device (if any) since the bus was last cleared.
8 START	Start This field is not usually needed, but can be used for wake events. 0 - No START seen. 1 - A START or repeated START was seen after the START bit was last cleared.
7 —	Reserved
6 STHDR	Status High Data Rate 0 - I3C bus not in HDR-DDR mode 1 - The I3C bus is in HDR-DDR mode , regardless of whether HDR mode is supported by this module or not, and regardless of whether the message is to this module or to some other module.
5 STDAA	Status Dynamic Address Assignment 0 - Not in ENTDAA mode. 1 - I3C bus is in Enter Dynamic Address Assignment (ENTDAA) mode, regardless of whether this bus target has a Dynamic Address or not.
4 STREQWR	Status Request Write See Status High Data Rate (STHDR) for Double Data Rate (DDR) handling. 0 - REQ in process is not SDR write data from the controller. 1 - REQ in process is SDR write data from the controller to this bus target (or all I3C targets), but not in ENTDAA mode.
3	Status Request Read

Table continues on the next page...

Table continued from the previous page...

Field	Description
STREQRD	See also Status High Data Rate (STHDR) for Double Data Rate (DDR) handling. 0 - REQ in process is not an SDR read from this target. 1 - The REQ in process is an SDR read from this target, or an In-Band Interrupt (IBI) is being pushed out.
2 STCCCH	Status Common Command Code Handler 0 - No CCC message is being handled. 1 - A CCC message is being handled automatically.
1 STMSG	Status message If STNOSTOP = 1, STMSG is 0 when a non-matching address is seen, until the next repeated START or STOP occurs. 0 - Bus target not listening or responding. 1 - This bus target is listening to the bus traffic or responding.
0 STNOTSTOP	Status Not Stop Other SSTATUS bits may also be set when busy. STNOTSTOP can also be 1 after an TE0 or TE1 error, when the I3C module is waiting for an Exit Pattern. 0 - I3C module is in a STOP condition. 1 - The bus is busy (has activity).

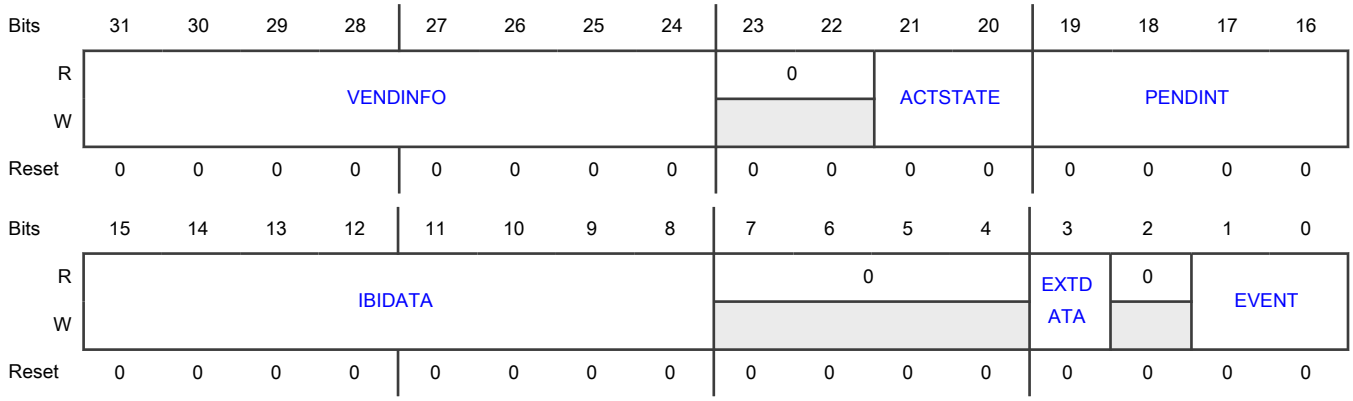
42.7.1.4 Target Control (SCTRL)

Contains controls for the active use of the I3C bus (for example, event generation such as interrupts to the controller). Only if the module is configured to support various special operations for the target, this register is used to activate those operations. These events include IBI and GETSTATUS fields (except Protocol error, which is automatically set).

Offset

Register	Offset
SCTRL	Ch

Diagram



Fields

Field	Description
31-24 VENDINFO	Vendor Information Should be set to the Vendor Reserved field that the GETSTATUS CCC returns. The vendor information should be maintained by the application, because the controller reads this field. If VENDINFO is not configured, then the GETSTATUS field always returns 0.
23-22 —	Reserved
21-20 ACTSTATE	Activity State of Target Should be set to the activity state of the target that the GETSTATUS CCC command returns as activity mode. The activity state should be maintained by the application, because the controller reads this field. If the activity state is not configured, then the GETSTATUS command always returns 0.
19-16 PENDINT	Pending Interrupt Should be set to the pending interrupt that the GETSTATUS CCC command returns. The pending interrupt should be maintained by the application, because the controller reads this field. If PENDINT = 0 and • If an IBI interrupt is pending, the GETSTATUS command returns 1. • If an IBI interrupt is not pending, the GETSTATUS field returns 0.
15-8 IBIDATA	In-Band Interrupt Data Data byte accompanying the IBI, if the module is enabled for IBI. If SCTRL[IBIDATA] is enabled, then IBI is required.
7-4 —	Reserved
3 EXTDATA	Extended Data See IBIEXT1[MAX] . If extended data is used with time control, the data follows the time information. 0 - Extended data disabled.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Extended data enabled. After IBIDATA is emitted, extended data is taken from IBIEXT1 and IBIEXT2 if configured.
2 —	Reserved
1-0 EVENT	Event • If EVENT is set to a nonzero value, it requests an event. • After the request, SSTATUS[EVENT] and SSTATUS[EVDET] show the status as it progresses. • After completion, the EVENT field automatically returns to 0. • After EVENT is a nonzero value, only 0 can be written to EVENT (to cancel) until the event processing is finished. 00 - NORMAL_MODE. If EVENT is set to 0 after it was a nonzero value and event processing has not yet started, event processing is canceled. If event processing has already started, event processing is not canceled. 01 - IBI. Start pushing an IBI onto the I3C bus. If there is data associated with the IBI, the data is read from SCTRL[IBIDATA] . If time control is enabled, this data includes any time-control-related bytes. Additionally, SCTRL[IBIDATA] bit 7 becomes 1 automatically (as is required for time control). The IBI interrupt occurs after the first (mandatory) IBIDATA, if any. 10 - CONTROLLER_REQUEST. Start a Controller Request request: the meaning depends on the Bus Characteristics Register (SIDEXT[BCR]) configured in the I3C module. 11 - HOT_JOIN_REQUEST. Start a Hot-Join request. A Hot-Join Request is used when the device is powered on after the I3C bus is already powered up. It is also used when the device is connected using hot-insertion methods (the device is powered up when it is physically inserted in the powered-up I3C bus). The HJ waits for Bus Idle, and SCTRL[EVENT] = HOT_JOIN_REQUEST must be set before the target enable (SCONFIG[SLVENA]).

42.7.1.5 Target Interrupt Set (SINTSET)

Sets interrupt enables for select Target Status (SSTATUS) fields. Reading the SINTSET register returns the status of the interrupt enables.

- To activate an interrupt enable, write 1 to its corresponding field in this register (SINTSET).
- To disable an interrupt, write 1 to its corresponding field in the Target Interrupt Clear (SINTCLR) register. Writing 0 to the interrupt enable in this register (SINTSET) does not disable the interrupt.

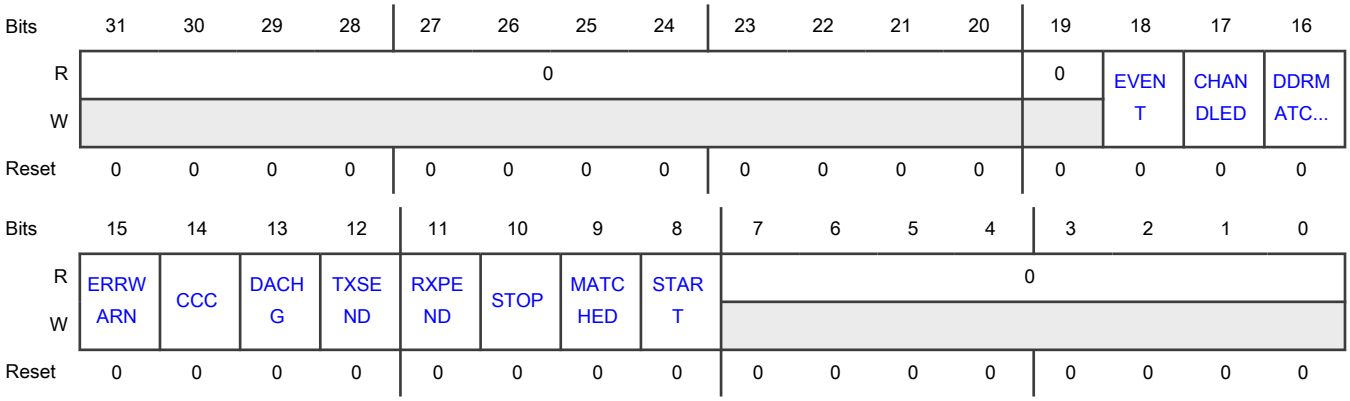
The Interrupt registers allow the masking of interrupt sources. They also allow the checking of which interrupts have activated. The normal method is to enable an interrupt, and then once the interrupt occurs, clear the interrupt either by writing the SSTATUS register or by performing action on the corresponding data register. The interrupt is level-held, meaning the interrupt stays set until the cause is cleared by some method. The module prevents races; if a new event occurs, that new event is not lost.

- SINTSET sets interrupt enables for Target Status (SSTATUS) fields. Reading the SINTSET register returns the status of the interrupt enables.
- SINTCLR clears interrupt enables for SSTATUS fields.
- SINTMASKED returns the value of the SSTATUS fields ANDed with their interrupt enables.

Offset

Register	Offset
SINTSET	10h

Diagram



Fields

Field	Description
31-20 —	Reserved
19 —	Reserved
18 EVENT	Event Interrupt Enable Interrupt when Pending IBI, CR, or Hot-Join has been sent as requested. See SSTATUS[EVDET] . Used only if configured to support events. 0 - Disable Event interrupt 1 - Enable Event interrupt
17 CHANDLED	Common Command Code (CCC) Interrupt Enable Interrupt when a CCC is received and handled by the block (is done). SSTATUS shows new results. CHANDLED can be used to track when Activity states and when masks on events (for example, IBIs) occur. Used for CCCs that are enabled. 0 - Disable CCC Handled interrupt 1 - Enable CCC Handled interrupt
16 DDRMATCHED	Double Data Rate Interrupt Enable Interrupt when DDR matched for read or write command. Used only if HDR enabled. 0 - Disable DDR interrupt

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Enable DDR interrupt
15 ERRWARN	Error or Warning Interrupt Enable Interrupt when an error or warning has occurred, such as data underrun, data overrun, parity error, or HDR-DDR CRC error. See the Target Errors and Warnings (SERRWARN) register for details of the cause. Only available for errors that are configured features. 0 - Disable error or warning interrupt 1 - Enable error or warning interrupt
14 CCC	CCC (that was not handled by I3C module) Interrupt Enable For CCCs not handled by the block, RXPEND also interrupts, and SSTATUS[STREQRD] indicates that it is a CCC sending a read request. 0 - Disable CCC interrupt 1 - Enable CCC interrupt
13 DACHG	Dynamic Address Change Interrupt Enable Interrupt on Dynamic address defined (SETDASA or ENTDAA) or lost (RSTDAA). See also Controller Dynamic Address (MDYNADDR) register. 0 - Disable DA Change interrupt 1 - Enable DA Change interrupt
12 TXSEND	Transmit Interrupt Enable Interrupt when request data by controller (read). This interrupt occurs on the first request (header) as well as when ready for more. The application indicates whether the interrupt is for more data or END. If this interrupt is for a FIFO, it triggers on the Transmit emptiness trigger. If this interrupt is for DMA, then it indicates message end (DMA end or termination). 0 - Disable Transmit interrupt 1 - Enable Transmit interrupt
11 RXPEND	Receive Interrupt Enable Interrupt when receiving a message from controller that is not being handled by the block like where data is consumed by hardware directly when it goes into the FIFO (excludes CCCs being handled automatically). If this interrupt is for a FIFO, it is a receive fullness trigger. If this interrupt is for DMA, then it indicates message end. 0 - Disable Receive interrupt 1 - Enable Receive interrupt
10 STOP	Stop Interrupt Enable Interrupt on STOP state on the bus. See SINTSET[START] as the preferred interrupt when needed. This interrupt may not trigger for quick a STOP/START combination, because it relates to the state of being stopped.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - Disable STOP interrupt 1 - Enable STOP interrupt
9 MATCHED	Match interrupt enable Interrupt on Matching header for I3C Dynamic Address. If configured and if no Dynamic Address set, this interrupt is also for matching a header on a I2C Static Address. See Controller Dynamic Address (MDYNADDR) register. 0 - Disable match interrupt 1 - Enable match interrupt
8 START	Start Interrupt Enable Interrupt on START and repeated START when needed (such as wakeup). See also SINTSET[STOP] . 0 - Disable START interrupt 1 - Enable START interrupt
7-0 —	Reserved

42.7.1.6 Target Interrupt Clear (SINTCLR)

Clears interrupt enables for select Target Status (SSTATUS) fields. To clear an interrupt enable, write 1 to the corresponding field in this register (SINTCLR). Writing 0 has no effect.

Offset

Register	Offset
SINTCLR	14h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												0	EVENT	CHANDLED	DDRMATC...
W														W1C	W1C	W1C
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	ERRWARN	CCC	DACHG	TXSEND	RXPEND	STOP	MATCHED	START	0							
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C								
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-20 —	Reserved
19 —	Reserved
18 EVENT	EVENT Interrupt Enable Clear
17 CHANDLED	CHANDLED Interrupt Enable Clear
16 DDRMATCHED	DDRMATCHED Interrupt Enable Clear
15 ERRWARN	ERRWARN Interrupt Enable Clear
14 CCC	CCC Interrupt Enable Clear
13 DACHG	DACHG Interrupt Enable Clear
12 TXSEND	TXSEND Interrupt Enable Clear
11 RXPEND	RXPEND Interrupt Enable Clear
10 STOP	STOP Interrupt Enable Clear
9 MATCHED	MATCHED Interrupt Enable Clear
8 START	START Interrupt Enable Clear
7-0 —	Reserved

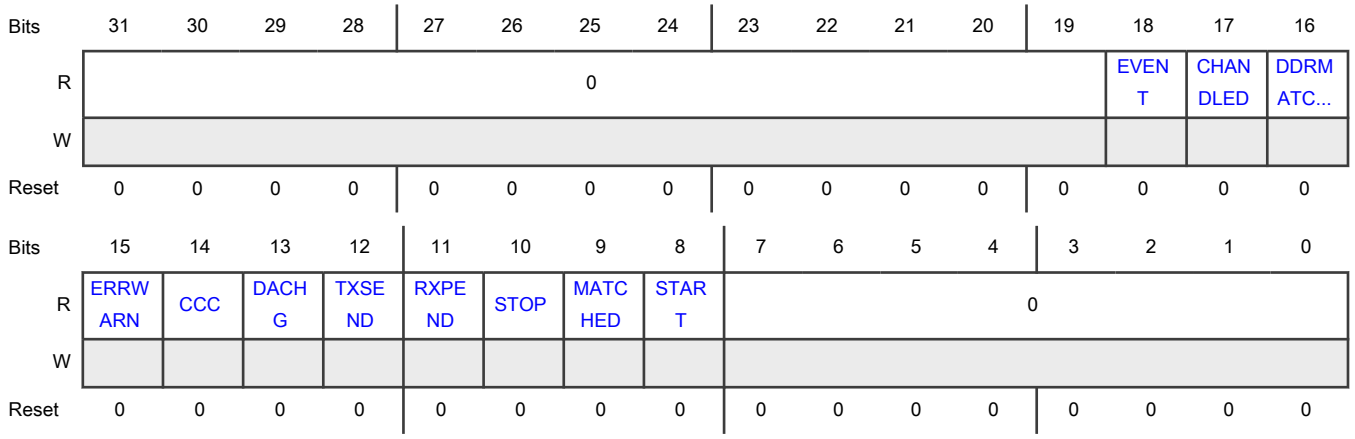
42.7.1.7 Target Interrupt Mask (SINTMASKED)

Returns the status of enabled interrupts (the value of Target Status (SSTATUS) ANDed with the value of Target Interrupt Set (SINTSET)).

Offset

Register	Offset
SINTMASKED	18h

Diagram



Fields

Field	Description
31-19 —	Reserved
18 EVENT	EVENT Interrupt Mask
17 CHANDLED	CHANDLED Interrupt Mask
16 DDRMATCHED	DDRMATCHED Interrupt Mask
15 ERRWARN	ERRWARN Interrupt Mask
14 CCC	CCC Interrupt Mask
13	DACHG Interrupt Mask

Table continues on the next page...

Table continued from the previous page...

Field	Description
DACHG	
12 TXSEND	TXSEND Interrupt Mask
11 RXPEND	RXPEND Interrupt Mask
10 STOP	STOP Interrupt Mask
9 MATCHED	MATCHED Interrupt Mask
8 START	START interrupt mask
7-0 —	Reserved

42.7.1.8 Target Errors and Warnings (SERRWARN)

Contains errors and warnings from I3C and I2C protocols, which includes internal issues such as overrun and underrun, detected errors and conditions like parity errors, CRC errors, and read terminations by the controller. Related to SSTATUS[ERRWARN] and SINTSET[ERRWARN] interrupt.

Offset

Register	Offset
SERRWARN	1Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0										0		0		OWRITE	OREAD
W															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	S0S1	HCRC	HPAR	SPAR	0			INVSTART	TERM	URUNNACK	URUN	ORUN
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	Description
31-22 —	Reserved
21 —	Reserved
20-18 —	Reserved
17 OWRITE	Over-write Error Indicates that the Target Write Data Byte (SWDATAB) or Target Write Data Byte End (SWDATABE) register was written when full. 0 - No Overwrite error 1 - Overwrite error
16 OREAD	Over-read Error Indicates that the Target Read Data Byte (SRDATAB) register was read for more bytes than were available by the application. This error also indicates over-read errors for the Target Read Data Halfword (SRDATAH) register, if it is enabled. 0 - No Over-read error 1 - Over-read error
15-14 —	Reserved
13 —	Reserved
12 —	Reserved
11 S0S1	TE0 or TE1 Error Indicates when an TE0 or TE1 error has occurred and the target is locked and waiting for an HDR Exit Pattern. Writing 1 to S0S1 causes the module to release the lock, but this method should be used with great care. S0S1 becomes 0 automatically when an Exit Pattern is detected, so writing 1 to S0S1 must be used under controlled circumstances to avoid problems. Before starting to operate normally after this error, the module waits for a START (or repeated START) or STOP. 0 - No TE0 or TE1 error 1 - TE0 or TE1 error
10 HCRC	HDR-DDR CRC Error

Table continues on the next page...

Table continued from the previous page...

Field	Description
	Indicates when an HDR-DDR Cyclic Redundancy Check (CRC) error on a message from the controller occurs. Causes include an HDR Restart and an Exit being issued before an HDR-DDR message from controller has finished. This error calls into question the data from the entire DDR command frame. 0 - No HDR-DDR CRC error 1 - HDR-DDR CRC error
9 HPAR	HDR Parity Error Indicates when HDR Parity error or framing error on a message from the controller occurs. The corresponding command or data that has the error is usually in the Rx buffer, which can be read using the Target Read Data Byte (SRDATAB) register. 0 - No HDR Parity error 1 - HDR Parity error
8 SPAR	SDR Parity Error Indicates when an SDR Parity error on a message from the controller occurs. This error also sets the GETSTATUS Protocol Error sticky bit (which becomes 0 after a GETSTATUS read). For read operations, this field becomes 1 when a Read Abort (timeout) occurs due to the controller not driving clock for more than 100 s during an I3C SDR Read. 0 - No SDR Parity error 1 - SDR Parity error
7-5 —	Reserved
4 INVSTART	Invalid Start Error Indicates an invalid condition with SCL falling before SDA falls, so there is no start. 0 - No invalid start error 1 - Invalid start error
3 TERM	Terminated Error Indicates when the controller terminates a read from a target when an END is not set (on the same read or the previous read). 0 - No terminated error 1 - Terminated error
2 URUNNACK	Underrun and Not Acknowledged (NACKED) Error Indicates when the internal to-bus buffer/FIFO is underrun in the read header and so the module NACKED the header. 0 - No underrun and not acknowledged error 1 - Underrun and not acknowledged error

Table continues on the next page...

Table continued from the previous page...

Field	Description
1 URUN	Underrun Error Indicates when the internal to-bus buffer/FIFO is underrun during data read (the application is not providing the data fast enough). The END bit or register should be used if the read was the last one. 0 - No underrun error 1 - Underrun error
0 ORUN	Overrun Error Indicates when the internal from-bus buffer or FIFO is overrun (too many characters are arriving and cannot be processed by the user application fast enough). Cutting off the RX_FIFO too close to the boundary when DUT is a target for SDR write or HDR-DDR write can cause an ORUN error when the FIFO is full. 0 - No overrun error 1 - Overrun error

42.7.1.9 Target DMA Control (SDMACTRL)

Allows DMA to be used for inbound and outbound messages. This register is limited in value for target use because the target must be reactive. Two common use models are:

- To avoid an overrun in from-bus collection. SCONFIG[MATCHSS] becomes 1, then the processor enables the interrupts for START and STOP and enables the DMA to collect the data. The START or STOP interrupt only occurs after a message is directed to the target (MATCHED is 1). The DMA copied data can then be examined.

  NOTE

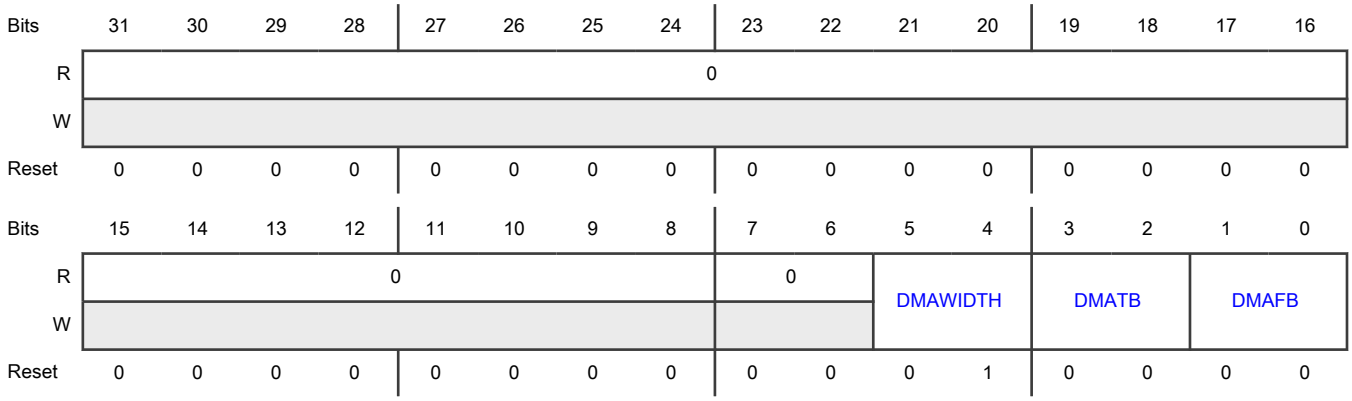
  Do not enable DMA after a transaction starts. To avoid a RX FIFO overrun, DMA must be enabled only while enabling the target and interrupts.

- To perform larger reads from the to-bus. I3C and I2C reads are preceded by a write that indicates what will be read (or in response to an IBI from the target). Because of this process, the DMA can be used to push through the data.
  - For I3C, the last value must be managed by the processor, unless the DMA moves wider words and is able to write 1 to the END field. These values are 16-bit values when in byte mode or 32-bit values when in half-word mode.
  - For I2C, the last value is determined by the controller, so the DMA may end early or may run out when the controller expects more values.

Offset

Register	Offset
SDMACTRL	20h

Diagram



Fields

Field	Description
31-8 —	Reserved
7-6 —	Reserved
5-4 DMAWIDTH	Width of DMA Operations Indicates the width of DMA operations, if configured to allow half-word data access. 00,01 - Byte 10 - Half word (16 bits). This value ensures that two bytes are available in the FIFO. 11 - Reserved
3-2 DMATB	DMA Write (To-bus) Trigger Enables DMA writes. If enabled with 1 or 2, DMATB starts a request DMA on a transmit trigger (see Target Data Control (SDATACTRL)). DMATB requests until full, unless the DMA is set up as a trigger. DMATB becomes 0 when MSTATUS[ERRWARN] . 00 - DMA not used 01 - DMA enabled for one frame (ended by DMA or terminated). DMATB automatically becomes 0 after a STOP or START. See SCONFIG[MATCHSS] . 10 - DMA enabled until turned off. This value must only be used with Controller Message mode. 11 - Reserved
1-0 DMAFB	DMA Read (From-bus) Trigger Enables DMA reads. If enabled with 1 or 2, DMAFB requests DMA on receive trigger (see SDATACTRL). It requests until empty unless the DMA is set up as a trigger. DMAFB becomes 0 when SSTATUS[ERRWARN] becomes 1. 00 - DMA not used

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01 - DMA is enabled for one frame. Automatically becomes 0 on STOP or repeated START. See SCONFIG[MATCHSS] . 10 - DMA enabled until turned off 11 - Reserved

42.7.1.10 Target Data Control (SDATACTRL)

Assists in data control when no FIFO is used. Also assists in data control of FIFO when the FIFO is available (regardless of size) allowing some control over the FIFO behavior. This register allows control of when to interrupt based on fullness or emptiness of a buffer or FIFO. It also controls behavior related to width, when the buffer or FIFO is not one byte wide.

Offset

Register	Offset
SDATACTRL	2Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RXEMPTY	TXFULL	0	RXCOUNT					0		0		TXCOUNT			
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								RXTRIG		TXTRIG			0		
W													UNLOCK		FLUSHFB	FLUSHTB
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

Fields

Field	Description
31 RXEMPTY	Receive Is Empty 0 - Not empty 1 - Empty
30 TXFULL	Transmit Is Full 0 - Not full 1 - Full

Table continues on the next page...

Table continued from the previous page...

Field	Description
29 —	Reserved
28-24 RXCOUNT	Count of Bytes in Receive
23-22 —	Reserved
21 —	Reserved
20-16 TXCOUNT	Count of Bytes in Transmit
15-8 —	Reserved
7-6 RXTRIG	Receive Trigger Level Indicates trigger level for Receive fullness when using a FIFO. Affects RXPEND interrupt. 00 - Trigger when not empty 01 - Trigger when 1/4 or more full 10 - Trigger when 1/2 or more full 11 - Trigger when 3/4 or more full
5-4 TXTRIG	Transmit Trigger Level Indicates trigger level for Transmit emptiness when using a FIFO. Affects TXNOTFULL interrupt. 00 - Trigger when empty 01 - Trigger when 1/4 full or less 10 - Trigger when 1/2 full or less 11 - Default. Trigger when 1 less than full or less
3 UNLOCK	Unlock UNLOCK must be 1 in the same cycle while writing to TXTRIG or RXTRIG. 0 - RXTRIG and TXTRIG fields cannot be changed on a write. 1 - RXTRIG and TXTRIG fields can be changed on a write.
2 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
1 FLUSHFB	Flush the From-bus Buffer or FIFO Not normally used.
0 FLUSHTB	Flush the To-bus Buffer or FIFO Used when the controller terminates a to-bus (read) message prematurely.

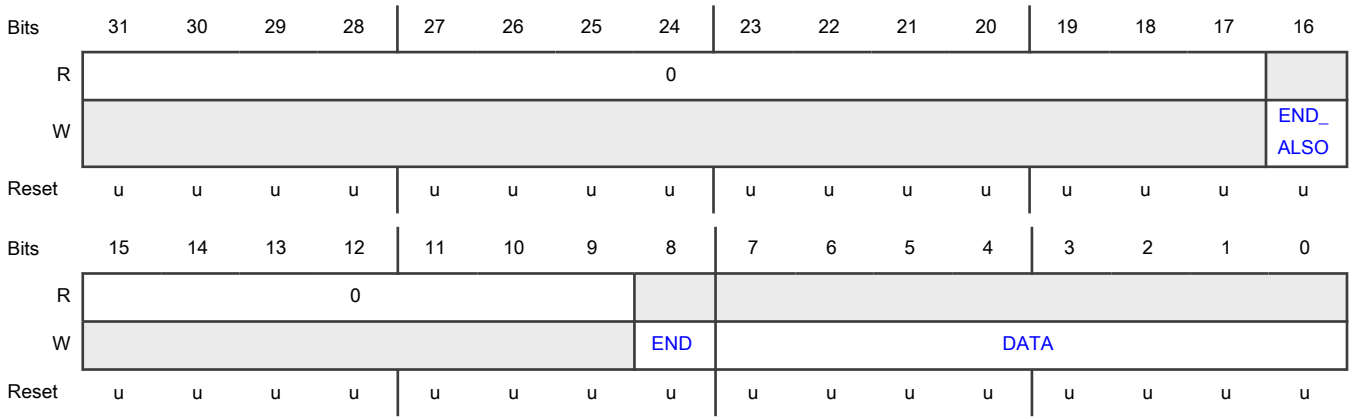
42.7.1.11 Target Write Data Byte (SWDATAB)

Allows writing a byte to the bus (to controller) unless an external FIFO is used. Writing a byte requires a byte plus an end-of-data (last) marker bit. A byte should not be written unless there is room, indicated by SSTATUS[TXNOTFULL] = 1.

Offset

Register	Offset
SWDATAB	30h

Diagram



Fields

Field	Description
31-17 —	Reserved
16 END_ALSO	End Also This field is required for I3C, but is optional for I2C. For HDR-DDR, the byte with the END_ALSO must be an even byte (second, fourth, sixth, and so on) because DDR uses byte-pairs. 0 - Not the end. There are more bytes in the message.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - End. This bit marks the last byte of the message.
15-9 —	Reserved
8 END	End This field is required for I3C, but is optional for I2C. For HDR-DDR, the byte with the END must be an even byte (second, fourth, sixth, and so on) because DDR uses byte-pairs. 0 - Not the end. There are more bytes in the message. 1 - End. This bit marks the last byte of the message.
7-0 DATA	Data Data byte to send to the controller

42.7.1.12 Target Write Data Byte End (SWDATABE)

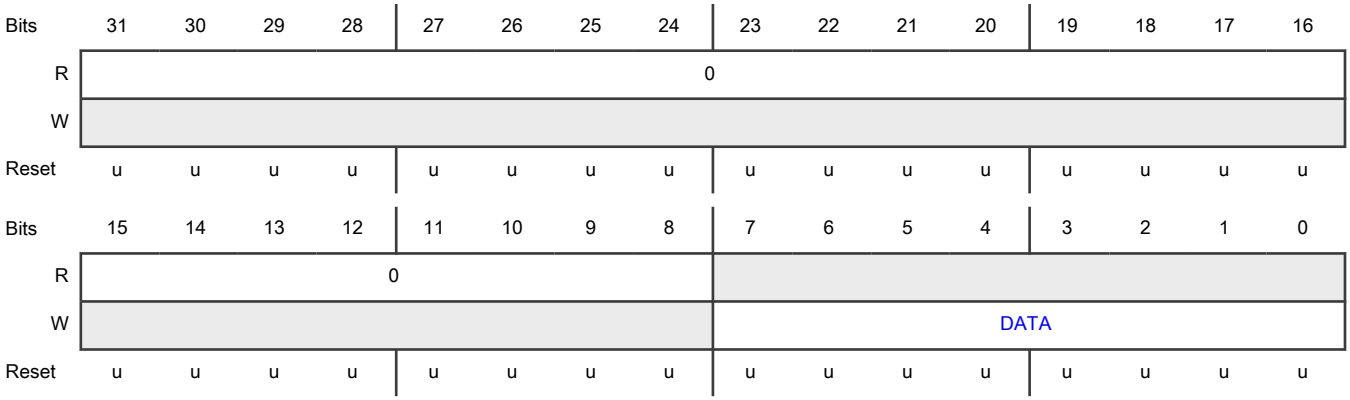
Allows writing a byte to the bus (to controller) unless an external FIFO is used. Unlike SWDATABm writing a byte only requires the byte itself, and is marked as end-of-data (last byte). A byte should not be written unless there is room, indicated by SSTATUS[TXNOTFULL] = 1.

For HDR-DDR, the byte with the END must be an even byte (second, fourth, sixth, and so on) because DDR uses byte-pairs.

Offset

Register	Offset
SWDATABE	34h

Diagram



Fields

Field	Description
31-8 —	Reserved
7-0 DATA	Data The data byte to send to the controller

42.7.1.13 Target Write Data Half-word (SWDATAH)

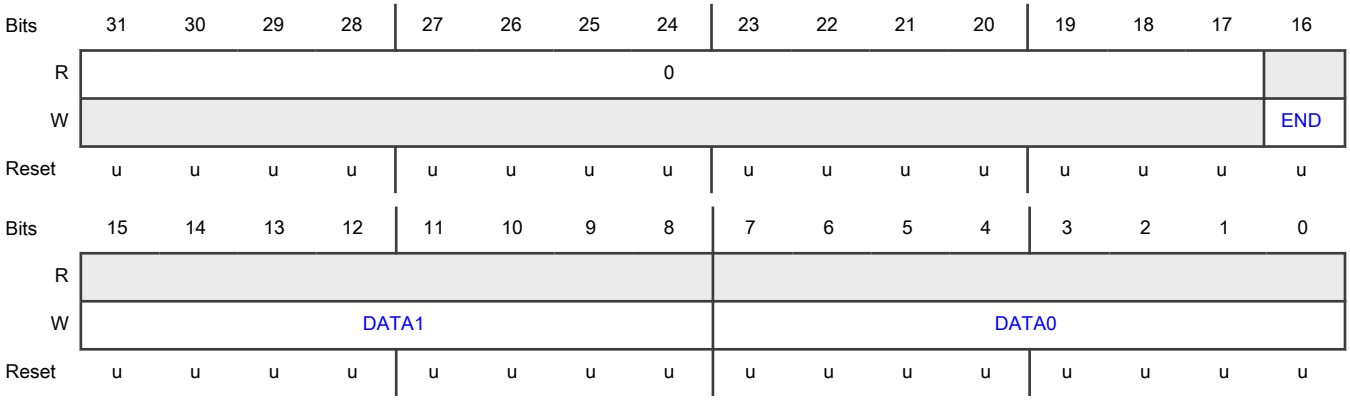
Allows writing a half-word (pair of bytes) to the bus unless an external FIFO is used. Sends the low byte followed by the high byte. The 16th bit marks the end; that is, the last byte of the half-word is the end.

An end-of-data (last) marker bit is allowed (or must be 0). A half-word should not be written unless there is room for both, as indicated by the use of Transmit FIFO level trigger or SDATACTRL[TXCOUNT] .

Offset

Register	Offset
SWDATAH	38h

Diagram



Fields

Field	Description
31-17 —	Reserved
16 END	End of message This field always marks DATA1 as the end. This field is required for I3C, but is optional for I2C.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	For HDR-DDR, the byte with the END must be an even byte (second, fourth, sixth, and so on) because DDR uses byte-pairs. 0 - Not the end. There are more bytes in the message. 1 - End. This bit marks the last byte of the message.
15-8 DATA1	Data 1 The second byte to send to the controller
7-0 DATA0	Data 0 The first byte to send to the controller

42.7.1.14 Target Write Data Half-word End (SWDATAHE)

Allows writing a half word of data, which is the end (the last byte of the half word is the end). Writes the half word (byte pair) just like Target Write Data Half-word (SWDATAH) , but marks the second byte as end-of-data (last byte).

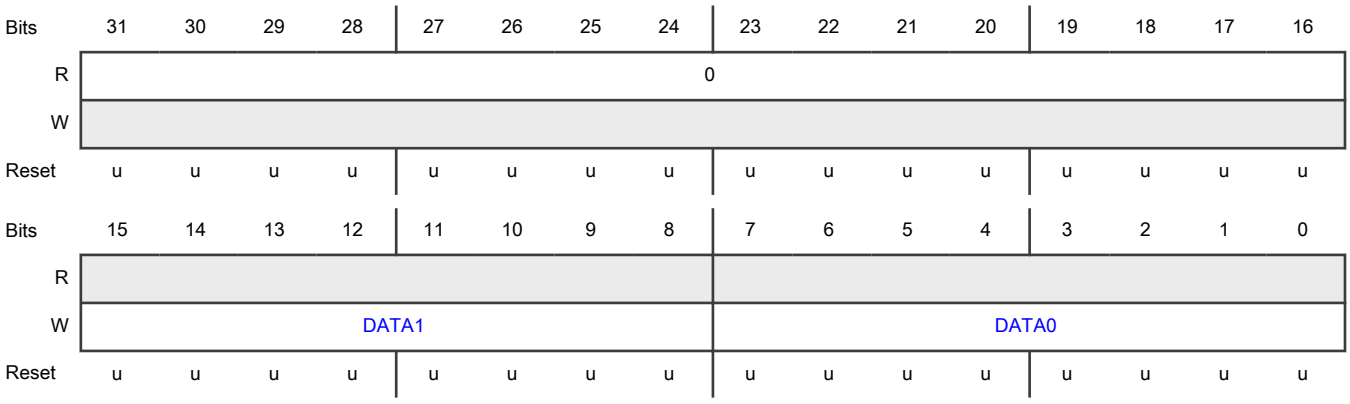
For HDR-DDR, the byte with the END must be an even (second, fourth, sixth, and so on) because DDR uses byte-pairs.

A half-word should not be written unless there is room for both, as indicated by the use of Transmit FIFO level trigger or SDATACTRL[TXCOUNT] .

Offset

Register	Offset
SWDATAHE	3Ch

Diagram



Fields

Field	Description
31-16 —	Reserved
15-8 DATA1	Data 1 The second byte to send to the controller
7-0 DATA0	Data 0 The first byte to send to the controller

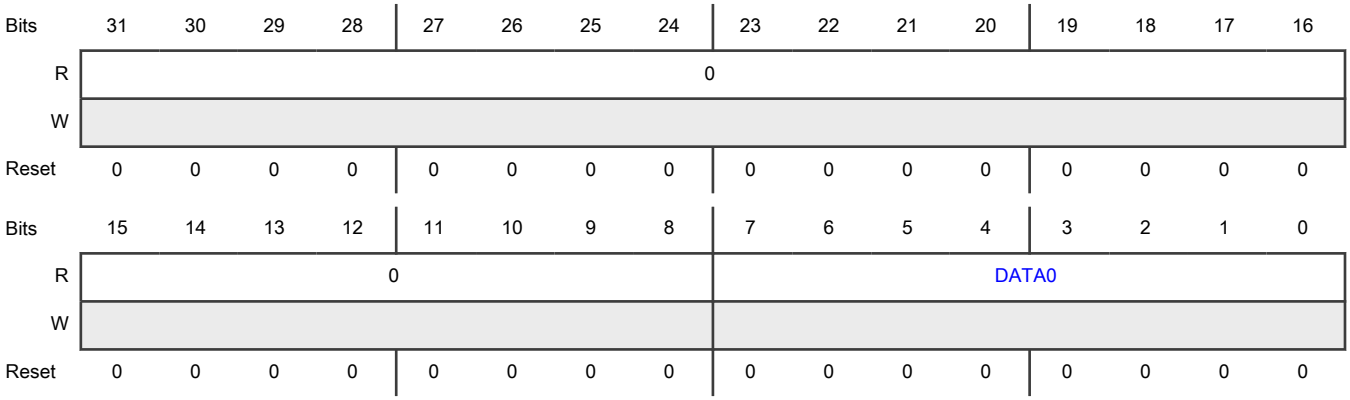
42.7.1.15 Target Read Data Byte (SRDATAB)

Allows reading a byte from the bus (controller). A byte should not be read unless there is data waiting, as indicated by SSTATUS[RX_PEND] = 1.

Offset

Register	Offset
SRDATAB	40h

Diagram



Fields

Field	Description
31-8 —	Reserved
7-0 DATA0	Data 0 Byte read from the controller

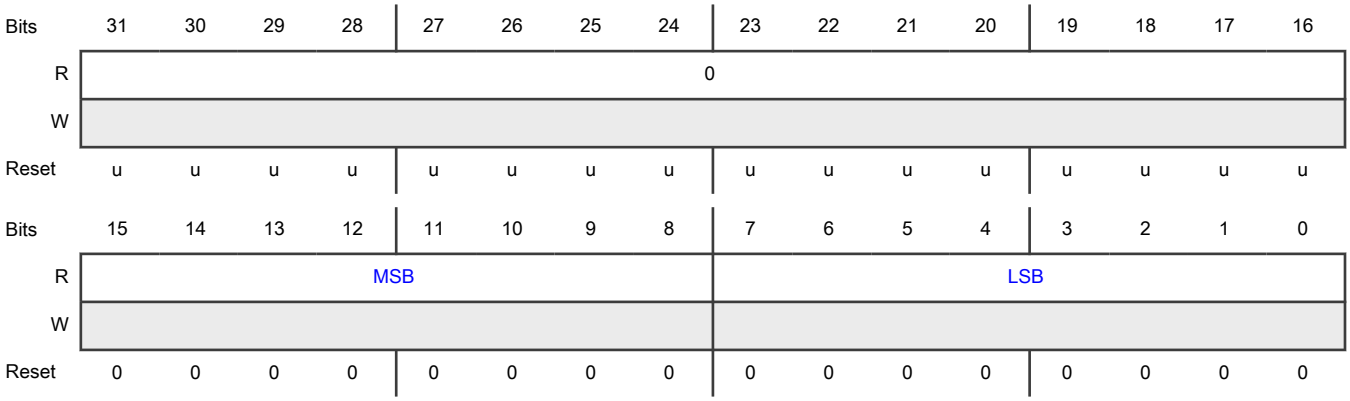
42.7.1.16 Target Read Data Halfword (SRDATAH)

Allows reading a halfword (byte pair) written by the target after an SDR Read or DAA or DDR. This register is only used when using Controller Control (MCTRL) to start the message. If MWMSG_SDR or MWMSG_DDR is used to start a message, that interface must be used exclusively. A half word should not be read unless there are at least two bytes of data waiting, as indicated the Receive FIFO level trigger or SDATACTRL[RXCOUNT] .

Offset

Register	Offset
SRDATAH	48h

Diagram



Fields

Field	Description
31-16 —	Reserved
15-8 MSB	The second byte read from the target
7-0 LSB	The first byte read from the target

42.7.1.17 Target Write Data Byte (SWDATAB1)

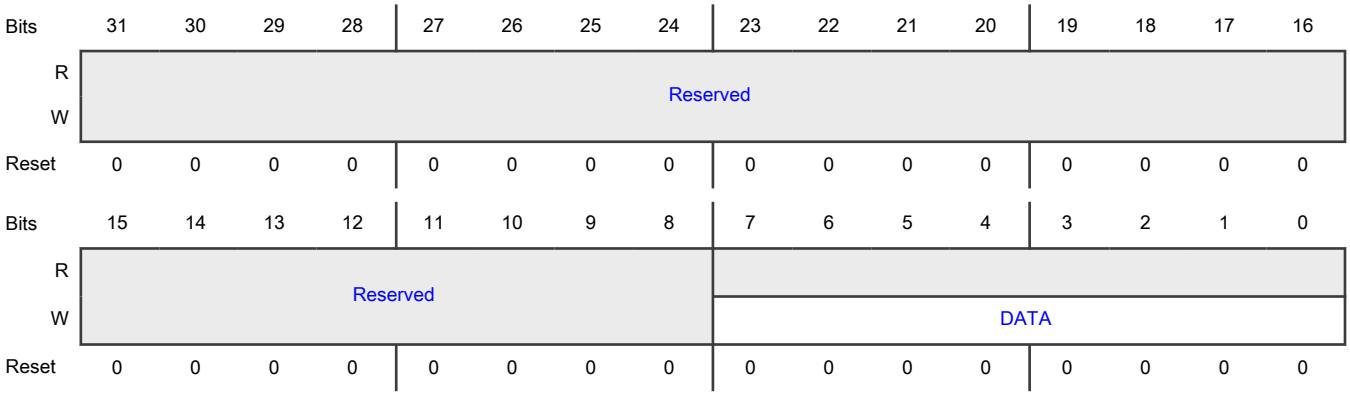
Allows writing a single byte to the bus (to controller) such that only bits 7:0 are used. Intended for use by DMAs, which do not format the upper part of the APB word.

A byte should not be written unless there is room, as indicated by SSTATUS[TXNOTFULL] = 1.

Offset

Register	Offset
SWDATAB1	54h

Diagram



Fields

Field	Description
31-8 —	Ignored field
7-0 DATA	Data Byte to send to controller.

42.7.1.18 Target Capabilities 2 (SCAPABILITIES2)

Indicates which features are available and supported in this I3C module, including controller and target capabilities, HDR modes, and others.

Offset

Register	Offset
SCAPABILITIES2	5Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0								SSTWR	SSTSUB	AASA	0	GROUP		SLVRST	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						IBIXREG	IBIEXT	0	I2CDEVID	I2CRST	I2C10B	MAPCNT			
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

Fields

Field	Description
31-24 —	Reserved
23 SSTWR	Target-Target(s)-Tunnel Write Capable 0 - Not write capable 1 - Write capable
22 SSTSUB	Target-Target(s)-Tunnel Subscriber Capable 0 - Not subscriber capable 1 - Subscriber capable
21 AASA	Supports SETAASA Supports the Set Static Address as Dynamic Address CCC feature 0 - Does not support SETAASA 1 - Supports SETAASA
20 —	Reserved
19-18 GROUP	Group Indicates whether v1.1 Group addressing is supported. Groups use mapping, so MAPCNT must be the same or greater than GROUP. 00 - Does not supports v1.1 Group addressing 01 - Supports one group 10 - Supports two groups 11 - Supports three groups

Table continues on the next page...

Table continued from the previous page...

Field	Description
17 SLVRST	Target Reset Indicates whether v1.1 Target Reset is supported 0 - Does not support Target Reset 1 - Supports Target Reset
16 —	Reserved
15-10 —	Reserved
9 IBIXREG	In-Band Interrupt Extended Register Indicates whether the Extended IBI Data 1 (IBIEXT1) register is available. The Extended IBI Data 2 (IBIEXT2) register is available if IBIEXT1[MAX] > 3. 0 - Does not support extended registers for IBIs 1 - Supports extended registers for IBIs
8 IBIEXT	In-Band Interrupt EXTDATA Supports SCTRL[EXTDATA] to allow data beyond the mandatory data byte (MDB) for IBIs. 0 - Does not support IBIEXT 1 - Supports IBIEXT
7 —	Reserved
6 I2CDEVID	I2C Device ID Indicates whether I2C Device ID is supported 0 - Does not support I2C device ID 1 - Supports I2C device ID
5 I2CRST	I2C Software Reset Is 1 if I2C Software Reset is supported 0 - Does not support I2C software reset 1 - Supports I2C software reset
4 I2C10B	I2C 10-bit Address Allows 10-bit I2C address in MAP 1. If this field is 1, MAPCNT must be 1 or greater. 0 - Does not support 10-bit I2C address 1 - Supports 10-bit I2C address

Table continues on the next page...

Table continued from the previous page...

Field	Description
3-0 MAPCNT	Map Count Indicates how many maps are allowed. If there is no mapping, this field is 0.

42.7.1.19 Target Capabilities (SCAPABILITIES)

Indicates which features are available and supported in this I3C module, including controller and/or target capabilities, HDR modes, and others.

Offset

Register	Offset
SCAPABILITIES	60h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DMA	INT	FIFORX		FIFOTX		EXTFIFO		0		TIMECTRL		IBI_MR_HJ			
W																
Reset	1	1	1	0	1	0	0	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	CCCHANDLE				SADDR		MASTER	0	HDRSUPP		IDREG				IDENA	
W																
Reset	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0

Fields

Field	Description
31 DMA	Direct Memory Access Indicates whether DMA is supported 0 - Not supported 1 - Supported
30 INT	Interrupts Indicates whether interrupts are supported 0 - Not supported 1 - Supported

Table continues on the next page...

Table continued from the previous page...

Field	Description
29-28 FIFORX	FIFO Receive Indicates size of Receive (from-bus) FIFO. **NOTE** Target and Controller use the same set of receive FIFOs. Because the module is either a Controller or Target at a time, there is no conflict. FIFO size of SDR and HDR-DDR is in bytes. 00 - Two or three 01 - Four 10 - Eight 11 - 16 or larger
27-26 FIFOTX	FIFO Transmit Indicates size of Transmit (to-bus) FIFO. **NOTE** Target and Controller use the same set of transmit FIFOs. Because the module is either a Controller or Target at a time, there is no conflict. FIFO size of SDR and HDR-DDR is in bytes. 00 - Two 01 - Four 10 - Eight 11 - 16 or larger
25-23 EXTFIFO	External FIFO Indicates whether External FIFOs are enabled. If External FIFOs are not enabled, then check FIFOTX and FIFORX for the internal FIFO. 000 - No external FIFO is available 001 - Standard available or free external FIFO 010 - Request track external FIFO All other values are reserved.
22 —	Reserved
21 TIMECTRL	Time Control Specifies whether any time-control type is supported. 0 - No time control enabled 1 - At least one time-control type supported

Table continues on the next page...

Table continued from the previous page...

Field	Description
20-16 IBI_MR_HJ	In-Band Interrupts, Controller Requests, Hot-Join Events Indicates which events (IBI, CR, and HJ) are allowed. For example, if this field is 00011b, IBI (bit 0) and IBI_HAS_DATA (bit 1) functionality are both enabled. 0_0000 - Application cannot generate IBI, CR, or HJ. 1_xxxx - Application can use SCONFIG[BAMATCH] for bus-available timing. x_1xxx - Application can generate a Hot-Join event. x_x1xx - Application can generate a Controller Request for a secondary controller. x_xx1x - When bit 0 = 1, the IBI has data from the SCTRL register. x_xxx1 - Application can generate an IBI.
15-12 CCCHANDLE	Common Command Codes Handling Indicates who manages CCC between I3C module and the user application. 0000 - All handling features below are disabled. 1xxx - GETSTATUS CCC returns SCTRL[VENDINFO] value. x1xx - GETSTATUS CCC returns SCTRL[PENDINT] and SCTRL[ACTSTATE] values. xx1x - The block manages maximum read and write lengths, and max data speed. xxx1 - The block (I3C module) manages events, activities, status, HDR, and if enabled for it, ID and static-address-related items.
11-10 SADDR	Static Address Indicates how the static address is managed. 00 - No static address 01 - Static address is fixed in hardware 10 - Hardware controls the static address dynamically (for example, from the pin strap) 11 - SCONFIG register supplies the static address
9 MASTER	Controller Specifies whether controller capability is supported. 0 - Not supported 1 - Supported
8 —	Reserved
7-6 HDRSUPP	High Data Rate Support Indicates which HDR modes are supported. 00 - No HDR modes supported 01 - Double Data Rate mode supported

Table continues on the next page...

Table continued from the previous page...

Field	Description
	All other values are reserved.
5-2 IDREG	ID Register Indicates which ID features are in the registers compared to what is in the hardware. 0000 - All ID register features below are disabled. 1xxx - A Bus Characteristics Register (BCR) is available. x1xx - A Device Characteristic Register (DCR) is available. xx1x - An ID Random field is available. xxx1 - ID Instance is a register, and is used if there is no PARTNO register.
1-0 IDENA	ID 48b Handler Indicates who handles the 48-bit ID value. 00 - Application 01 - Hardware 10 - Hardware, but the I3C module instance handles ID 48b 11 - A part number register (PARTNO)

42.7.1.20 Target Dynamic Address (SDYNADDR)

The Target Dynamic Address register is filled in with the assigned address once the controller has assigned it via SETDASA or ENTDAA CCC commands. It clears if the RESETDAA CCC is used. The current validity state is also indicated through SSTATUS[DAVALID], which can be used to interrupt the processor. The DCAUSE field can be used to determine the changes, if enabled.

NOTE

Uses for Mapped addresses, if enabled, are accessed through the SMAPCTRLn registers.

The first dynamic address (DA[0]) may be written whether the mapping is enabled or not. If configured to allow write used to restore the DA after a power-down (an ultra-low power state that loses power to peripherals but retains the DA somewhere else). This is not required if state-retention flops are used for the DA. This mechanism only allows writes when target is disabled (and it will be ignored otherwise). If the controller uses RSTDAA or SETNEWDA, then it will over-ride this mechanism and cede (yield) to the controller-assigned DA. Note that when enabling the target, the SCONFIG[OFFLINE] bit must be set. This waits for evidence that the bus is not in I3C HDR mode and exits when an HDR Exit pattern is seen or when 60 s has expired. This makes it safe to monitor START and STOP. If the application needs to do an IBI, then the application should either wait for a STOP (see STATUS) or make sure that 200 s have gone by with no activity (no START or STOP) before the application emits the IBI.

The MAPIDX/MAPSA model allows writing additional DAs (and SAs) into a list (the number in the list is pre-configured), any DA or SA with DAVALID = 0 are never matched. The additional DAs may be based on the bridge target CCC or done by a move (read current DA and then copy into the upper map, then invalidate the 0 map) when matching ENTDAA over and over. Any mapped location can be invalidated by writing the MAPIDX and DAVALID = 0. The mapped ones are not reset by the RSTDA CCC. See the SMSGMAPADDR register.

To copy the base DA to the mapped set, the configuration has to be set up to allow it, otherwise they are distinct mechanisms.

- If used in I3C mode, a base DA is required, so the last one should not be cleared (or writing one with DAVALID = 1 is necessary).

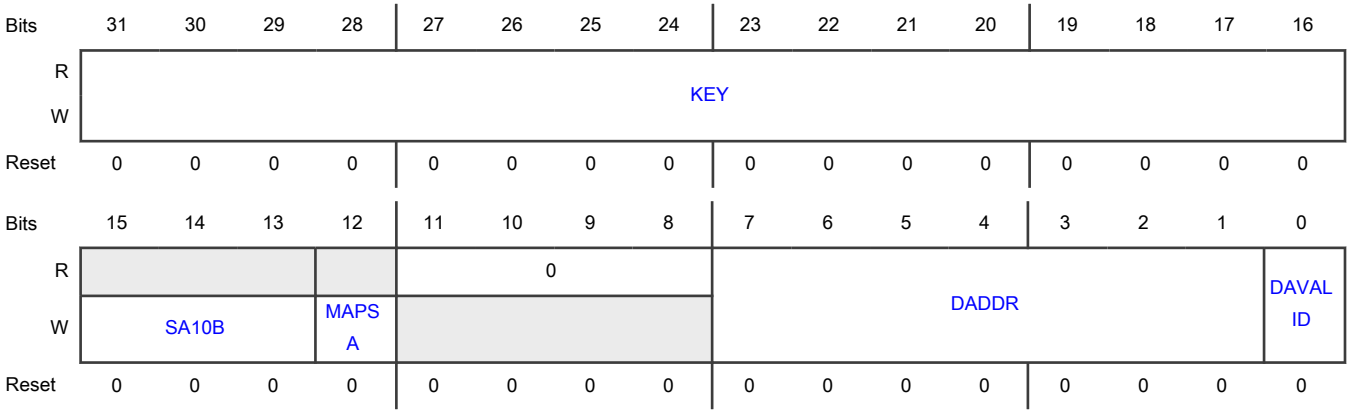
NACK/ACK of Mapped DAs/SAs: The register also permits specific DAs (above the base) to NACK or ACK writes or reads, such as for bridges when the endpoint is busy. The NACK should not be set for too long, or the controller may think the target is in an error state. The model is to write with KEY = 0xA731 and the MAPIDX and DAVALID to select ACK or NACK.

Mapping: If write with MAPIDX != 0 and KEY = 0, then next Read returns details on the Static or Dynamic address at that location. If the read does not have MAPIDX != 0, then it was not an acceptable location.

Offset

Register	Offset
SDYNADDR	64h

Diagram



Fields

Field	Description
31-16 KEY	Key Must set to 0xA4D9 to write DADDR field (and set DAVALID bit to 1). Write only when a target is not enabled (for restoring after power-down sleep with auto-restore). The mapped locations and base may be written when a target is enabled, but do not write when there are transactions on the I3C bus. KEY reads back 1 if overwritten, else KEY is 0 if assigned by the controller including when the controller changes it. If address mapping is allowed, then writing with KEY = 0xCB19 is used to clear the base DA. If Mapping is allowed, then writing with the key of 0xA731 means that MAPIDX indicates which mapped address to set to ACK or NACK, such that DAVALID = 1 if ACK, DAVALID = 0 if NACK. This allows having the address refuse write/read requests when NACK. When using read back from map index (write with MAPIDX != 0, KEY = 0, read) bit [16] = 1 if NACK, else 0.
15-13 SA10B	10bit Static Address If not 0 when MAPSA is set, stores a 10-bit static address composed of {SA10B,DADDR}. Only one static address may be stored this way, and it must be MAPIDX == 1.
12	Map a Static Address

Table continues on the next page...

Table continued from the previous page...

Field	Description
MAPSA	If MAPSA = 1 on a write with MAPIDX != 0, then this sets a static address into the list, otherwise a dynamic address is used.
11-8 —	Reserved
7-1 DADDR	Dynamic Address This is the assigned DA, when DAVALID is 1. Will be a static address if MAPSA is 1.
0 DAVALID	Dynamic Address Valid Determines if a Dynamic Address is assigned. Is 1 for ACK, 0 for NACK when using the 0xA731 KEY. 0 - DANOTASSIGNED: a Dynamic Address is not assigned 1 - DAASSIGNED: a Dynamic Address is assigned

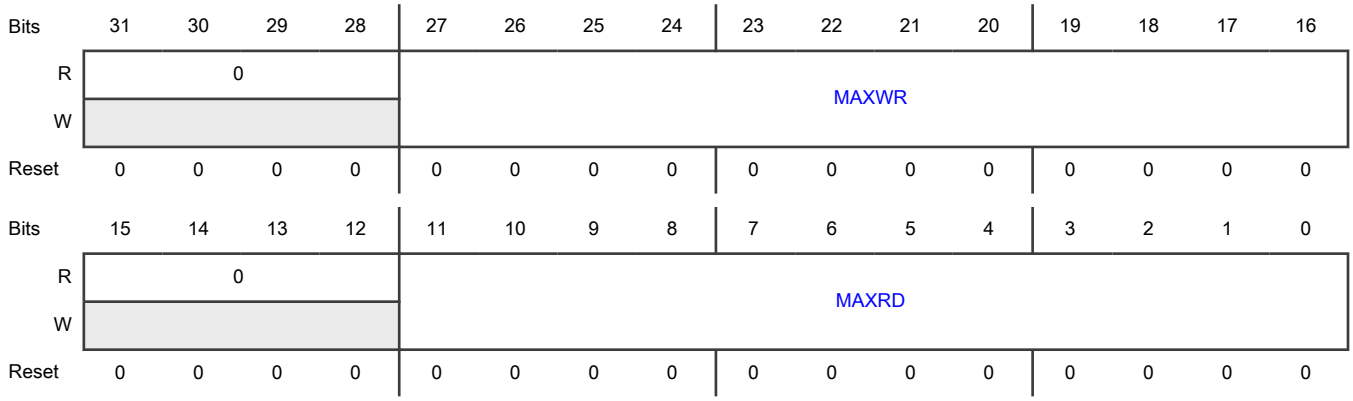
42.7.1.21 Target Maximum Limits (SMAXLIMITS)

Indicates the limits set by the controller (or the original requested limits). The maximum limits are not enabled in the hardware design, including maximum read and write lengths. If the maximum read and write lengths are enabled, then the current setting (including default request) shows up in this register (SMAXLIMITS).

Offset

Register	Offset
SMAXLIMITS	68h

Diagram



Fields

Field	Description
31-28 —	Reserved
27-16 MAXWR	Maximum Write Length Indicates the maximum write length, which must be between 8 to 4095 (saturation). The application must not set the maximum write length to a higher value than the maximum write length set by the controller.
15-12 —	Reserved
11-0 MAXRD	Maximum Read Length Indicates the maximum read length, which must be between 16 to 4095 (saturation). The application must not set the maximum read length to a higher value than the maximum read length set by the controller.

42.7.1.22 Target ID Part Number (SIDPARTNO)

Allows an application to write the ID part number. The application must write a nonzero value into the PARTNO field, because 0 is not valid.

Offset

Register	Offset
SIDPARTNO	6Ch

Diagram

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

Fields

Field	Description
31-0 PARTNO	Part number

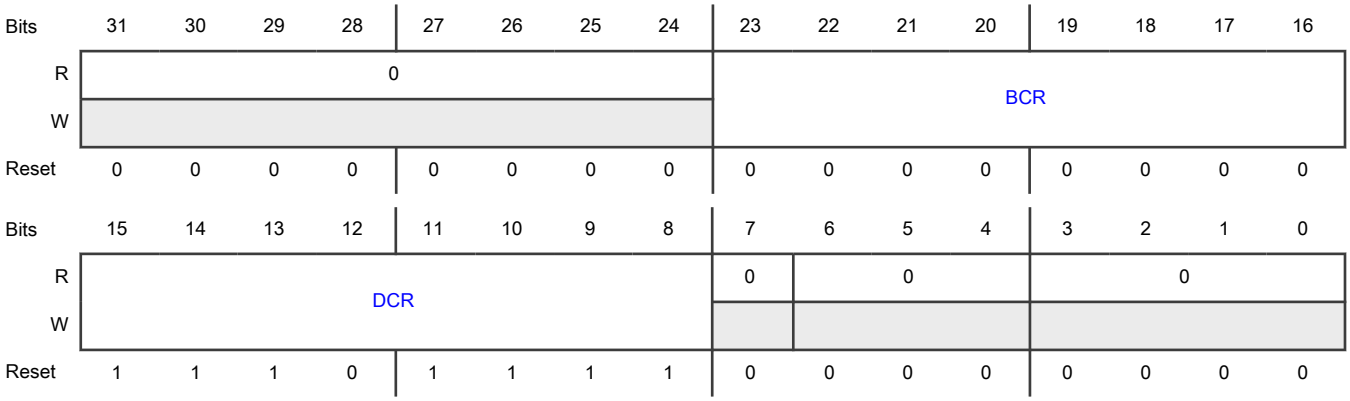
42.7.1.23 Target ID Extension (SIDEXT)

Allows an application to write the ID extension of the Device Characteristic Register (DCR) and/or the Bus Characteristics Register (BCR).

Offset

Register	Offset
SIDEXT	70h

Diagram



Fields

Field	Description
31-24 —	Reserved
23-16 BCR	Bus Characteristics Register Sets the value for the Bus Characteristics Register (BCR), if this field is configured. This field controls features such as Secondary Controller and slow speed requirements.
15-8 DCR	Device Characteristic Register Sets the value for the Device Characteristic Register (DCR), if this field is configured.
7 —	Reserved
6-4 —	Reserved
3-0 —	Reserved

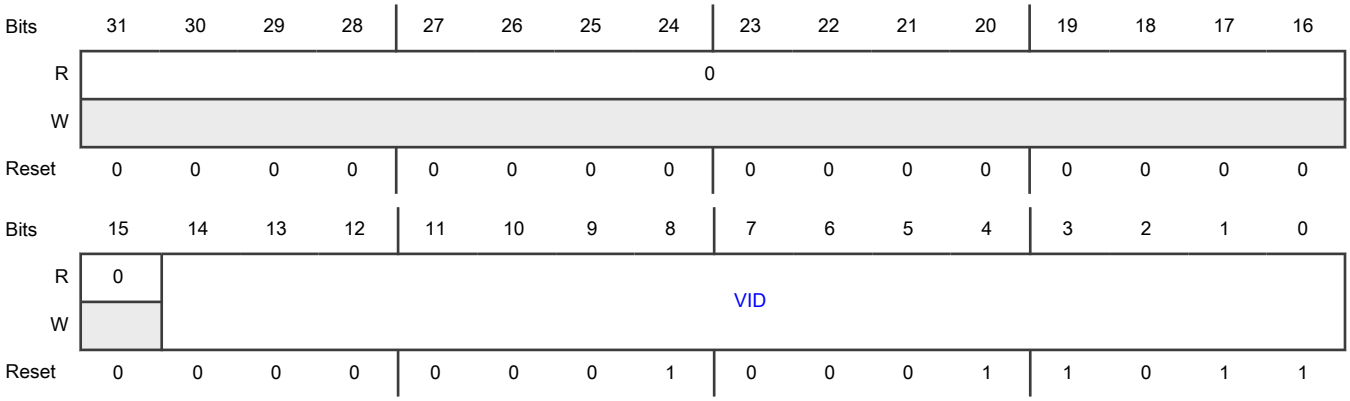
42.7.1.24 Target Vendor ID (SVENDORID)

Allows an application to write the Vendor ID. The default value is the chip vendor ID, and is set from the constant field. If using the chip vendor ID, the part number (PARTNO) does not collide with other uses. The MIPI Vendor ID is available to all companies (MIPI membership is not required). To get a vendor ID make a request at the mipi.org website.

Offset

Register	Offset
SVENDORID	74h

Diagram



Fields

Field	Description
31-15 —	Reserved
14-0 VID	Vendor ID Can be set to the 15-bit MIPI Vendor ID.

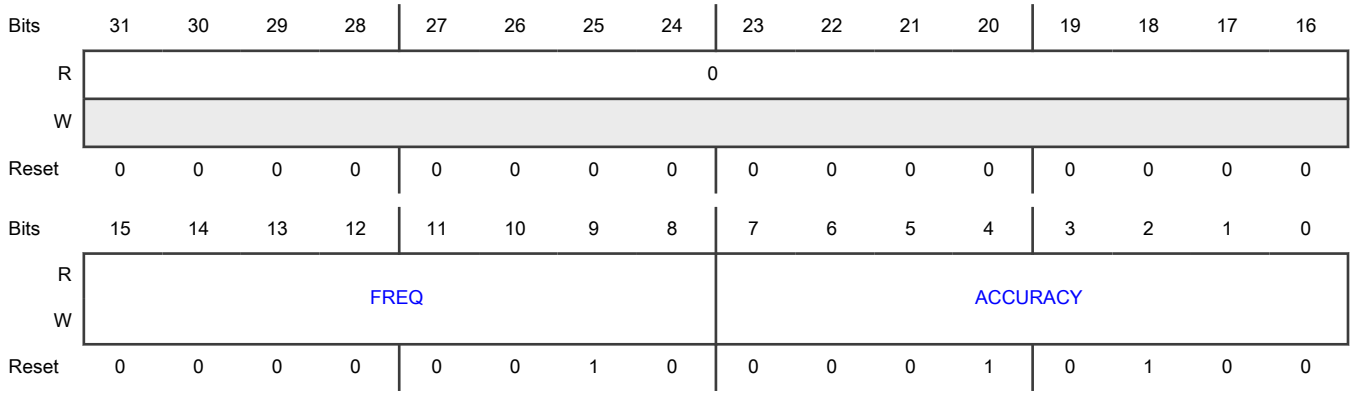
42.7.1.25 Target Time Control Clock (STCCLOCK)

Allows an application to dynamically set the time control clock and accuracy information. The clock frequency and accuracy are constants set by the hardware. If the clock can be adjusted (that is, divided) or if the accuracy could vary with knowable information, then the clock may be set via this register. This register must be updated whenever the clock source is changed.

Offset

Register	Offset
STCCLOCK	78h

Diagram



Fields

Field	Description
31-16 —	Reserved
15-8 FREQ	Clock Frequency Indicates the clock frequency in 0.5-MHz steps. For example, a value of 20 in this field indicates a frequency of 10 MHz. Default set by parameters if configured.
7-0 ACCURACY	Clock Accuracy Indicates the clock accuracy in 1/10ths of %. For example, a value of 15 indicates an accuracy of 1.5%. Default set by parameters if configured.

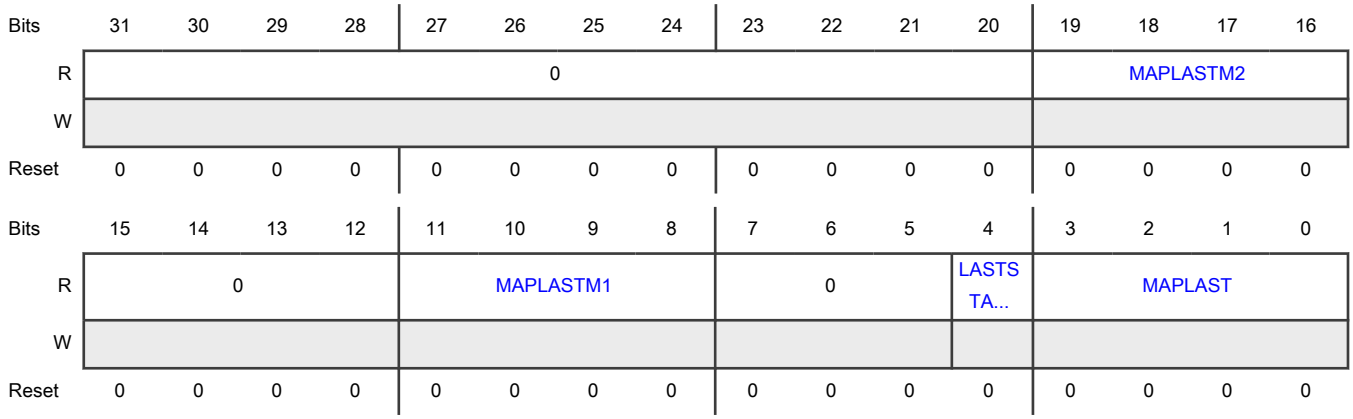
42.7.1.26 Target Message Map Address (SMSGMAPADDR)

Allows the software to determine which address is matched when STATUS[MATCHED] = 1. This register is used when the DYNADDR register builds a list of extra DAs or SAs to match. It holds the last three matches.

Offset

Register	Offset
SMSGMAPADDR	7Ch

Diagram



Fields

Field	Description
31-20 —	Reserved
19-16 MAPLASTM2	Matched Previous Index 2 Indicates Index 2 of the previous matched address. 0 for the base address.
15-12 —	Reserved
11-8 MAPLASTM1	Matched Previous Address Index 1 Indicates Index 1 of the previous matched address. 0 for the base address.
7-5 —	Reserved
4 LASTSTATIC	Last Static Address Matched Indicates whether the last matched address was an I2C static address or an I3C dynamic address. 0 - I3C dynamic address 1 - I2C static address
3-0 MAPLAST	Matched Address Index Indicates the matched address index for current or last matched message. 0 for the base address.

42.7.1.27 Controller Control (MCTRL)

Starts activities on the I3C or I2C bus. Also see the MWMSG register. A request cannot be changed when a message is in progress; the REQUEST field will become 0 automatically.

NOTE

Write MCTRL fields as per use case or as mentioned in Operating modes. Field read/writes may not work independently.

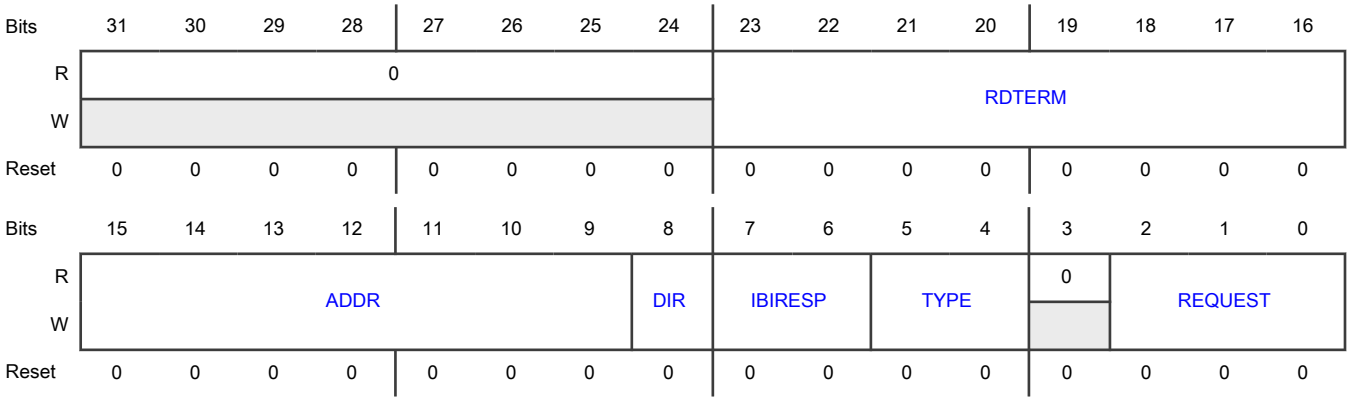
NOTE

If MCONFIG[MSTENA] is set for I2C Controller mode (legacy I2C), only REQUEST = 1 and REQUEST = 2 are accepted. Also, fields are constrained to I2C-supported fields.

Offset

Register	Offset
MCTRL	84h

Diagram



Fields

Field	Description
31-24 —	Reserved
23-16 RDTERM	Read Terminate Counter Determines when to terminate a read operation. • For I2C, controls when to NACK a read. • For I3C, can be used to terminate (end) a read. — RDTERM = 0 has no effect. — RDTERM = 1 terminates after the next character. — RDTERM = 2 terminates after the next two characters. Supports up to 255 characters. In DDR mode, RDTERM terminates the read based on word counts (for DDR) instead of byte counts (for SDR). **NOTE** The value 1 may be written any time to this field, but a number larger than 1 should be written when starting EmitStartAddress. Self clears on COMPLETE.

Table continues on the next page...

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td>15-9
ADDR</td><td>Address
I3C Dynamic Address or I2C Static Address. Some values are not allowed based on which bus is used (I3C or I2C).</td></tr>
<tr><td>8
DIR</td><td>Direction
Indicates write or read
0 - Write
1 - Read</td></tr>
<tr><td>7-6
IBIRESP</td><td>In-Band Interrupt Response
Indicates the response to use when you get an IBI from START, and when to force using a IBI ACK NACK request when completing a manual IBI. Completion of a manual IBI means that the target DA is known, and so the mandatory byte (or not) is specified by the application when acknowledging.
NOTE
Controller Request and Hot-Join always cause IBIRESP = 3 (Manual) so the application must decide.
The Mctrl[IBIRESP] field is also used when a message is emitted in Message mode using the MWMSG_SDR or MWMSG_DDR registers.
If an IBI with MDB (mandatory byte) is ACKed, the controller limits it to a max of eight more bytes after the MDB. RDTERM = 1 can be used with Request = None to terminate data from a target sooner.
00 - ACK (acknowledge). When REQUEST = 1 (EmitStartAddr) or REQUEST = 7 (AutoIBI), ACK with mandatory byte (or not), decided by the Controller In-band Interrupt Registry and Rules (MIBIRULES) register. When REQUEST = 3 (IBIAckNack), ACK with no mandatory byte.
01 - NACK (reject). Not acknowledge
10 - Acknowledge with mandatory byte. When REQUEST = 1 or REQUEST = 7, ignore the MIBIRULES register. Do not use this setting unless only targets with a mandatory byte can cause an IBI. When REQUEST = 3, ACK with mandatory byte.
11 - Manual. When REQUEST = 1 or REQUEST = 7, stop and wait for a decision using the IBI ACK NACK request. When REQUEST = 3, reserved.</td></tr>
<tr><td>5-4
TYPE</td><td>Bus Type with EmitStartAddr
<table>
<tr><th>Value</th><th>Meaning when REQUEST = 1 (EmitStartAddr)</th><th>Meaning when REQUEST = 2 (EmitStop)</th><th>Meaning when REQUEST = 6 (ForceExit)</th></tr>
<tr><td>0</td><td>I3C - SDR mode of I3C</td><td>I3C - SDR mode of I3C</td><td>Exit Pattern</td></tr>
<tr><td>1</td><td>I2C - Standard I2C protocol</td><td>I2C - Standard I2C protocol</td><td>Reserved</td></tr>
<tr><td>2</td><td>DDR - HDR-DDR mode of I3C. Enter DDR mode (7E and then ENTHDR0),</td><td>Reserved</td><td>Target Reset</td></tr>
</table>
</td></tr>
</table>

Table continued from the previous page...

<table>
<tr><th>Field</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Meaning when REQUEST = 1 (EmitStartAddr)</th><th>Meaning when REQUEST = 2 (EmitStop)</th><th>Meaning when REQUEST = 6 (ForceExit)</th></tr>
<tr><td></td><td>if the module is not in DDR mode. The 1st byte written to the Tx FIFO must be a command and already in the FIFO. To end DDR mode, use ForceExit.</td><td></td><td></td></tr>
<tr><td>3</td><td>BT - HDR-BT mode of I3C. If not already in HDR-BT, will automatically Enter BT (send 7E and ENTHDR3). When using this, you must also set the MHDRBTCFG register for multilane and CMD rules.</td><td>Reserved</td><td>Reserved</td></tr>
</table>
The meaning of the field settings change according to REQUEST value.
00 - I3C. When REQUEST = 1 (EmitStartAddr) or 2 (EmitStop), the SDR mode of I3C. When REQUEST = 6 (ForceExit), the Exit pattern.
01 - I2C. When REQUEST = 1 (EmitStartAddr) or 2 (EmitStop), the Standard I2C protocol. When REQUEST = 6, reserved.
10 - DDR. When REQUEST = 1, the HDR-DDR mode of I3C. Enter DDR mode (7E and then ENTHDR0), if the module is not in DDR mode. The 1st byte written to the Tx FIFO must be a command and already in the FIFO. To end DDR mode, use ForceExit. When REQUEST = 2, reserved. When REQUEST = 6, Target Reset.
11 - Reserved
</td></tr>
<tr><td>3
—</td><td>Reserved</td></tr>
<tr><td>2-0
REQUEST</td><td>Request
Emits the requested operation when performing in pieces, instead of performing by message. The Controller Status (MSTATUS) register should be checked because some requests can only be made in some states. For example, the system cannot enter SDR mode from DDR mode, and it cannot use an incorrect request in DAA mode.
000 - NONE. Indicates that no request is present. The REQUEST field returns to NONE when finished with any request. The MSTATUS register indicates the state of the controller. See also AutoIBI mode. NONE is only written as 0 in these cases: when writing 1 to MCTRL[RDTERM] (to stop a read in progress) or when setting MCTRL[IBIRESP] for MSG use.
001 - EMITSTARTADDR. Emit START with address and direction, either from a stopped state or in the middle of a Single Data Rate (SDR) message. If from a stopped state (IDLE), then emit start may be prevented by an event (like IBI, CR, HJ). In this case the appropriate interrupt is signaled. Emit START can be resubmitted.</td></tr>
</table>

Table continues on the next page...

Table continued from the previous page...

Field	Description
	010 - EMITSTOP. Emit a STOP on bus. Must be in SDR mode. In Dynamic Address Assignment (DAA) mode, emitting stop exits DAA mode. 011 - IBIACKNACK. Manual IBI ACK or NACK. When MCTRL[IBIRESP] has indicated a hold on an IBI to allow a manual decision, this request completes it. Uses MCTRL[IBIRESP] to provide the information. 100 - PROCESSDAA. If not currently in Dynamic Address Assignment (DAA) mode, emits START, 7E, ENTDAA sequence, then emits 7E/R to process the first target. Stops just before the new Dynamic Address (DA) is to be emitted. The DA is written using Controller Write Data Byte (MWDATAB) , then Process DAA is requested again to write the new address, and then it starts the next unless marked to STOP. An MSTATUS indicating NACK means DA was not accepted (for example, parity error). If PROCESSDAA is NACKED on the 7E/R request, meaning no more targets need a DA, then a COMPLETE is signaled (along with DONE) and a STOP issued. 101 - Reserved 110 - Force Exit and Target Reset. Emit an Exit Pattern from any state. End Double Data Rate (DDR) mode (including MSGDDR), including a STOP afterward. 111 - AUTOIBI. Hold in a stopped state, but auto-emit a START, 7E sequence when the target holds SDA low for an IBI. Actual IBI handling is defined by MCTRL[IBIRESP].

42.7.1.28 Controller Status (MSTATUS)

Status for the controller, including which events cause interrupts. The peripherals share the IRQ (called Parallel-to-Target status).

Because a peripheral can either be in Controller or Target mode, but not both at the same time, only one (Target or Controller peripheral) can be the cause of the IRQ. If there is an IRQ and the peripheral is a controller, then this register (MSTATUS) has the status.

If there is an IRQ and the peripheral is a target, then the Target Status (SSTATUS) register has the status.

Self clears on COMPLETE.

If MCONFIG is set for I2C only, some states and bits will never be active (for example, DAA or IBI).

Offset

Register	Offset
MSTATUS	88h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	IBIADDR							0				NOWMAST...	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	ERRWARN	0	IBIWON	TXNOTFU...	RXPEND	COMPLETE	MCTRLDO...	SLVSTART	IBITYPE		NACKED	BETWEEN	0	STATE		
W			W1C			W1C	W1C	W1C								
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31 —	Reserved
30-24 IBIADDR	IBI Address Indicates the address of: • The IBI, when MSTATUS[IBITYPE] = 1. • The Controller Request (CR), when MSTATUS[IBITYPE] = 2. • 7'h2 when Hot-Join, when MSTATUS[IBITYPE] = 3.
23-20 —	Reserved
19 NOWMASTER	Module Is Now Controller Indicates when the module is now a controller. That is, it was previously a target, controllership acceptance was requested from the previous controller, and controllership was accepted. The reverse operation (controller becomes a target) does not need an interrupt, because the application grants it through the GETACCMST CCC. 0 - Module has not become controller 1 - Module has become controller
18-16 —	Reserved
15 ERRWARN	Error Or Warning Indicates whether an error occurred, such as improper register use, overrun or underrun of FIFO or buffer, or invalid parity or CRC in a DDR read. See the Controller Errors and Warnings (MERRWARN) register.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - No error or warning 1 - Error or warning
14 —	Reserved
13 IBIWON	In-Band Interrupt (IBI) Won Indicates whether an IBI, CR, or HJ has won the arbitration on a header address, regardless of whether it was NACKed or ACKed. Arbitration requires manual intervention for CR and HJ, and optionally requires it for IBI if the MCTRL[IBIRESP] = 3 (Manual). 0 - No IBI arbitration won 1 - IBI arbitration won
12 TXNOTFULL	TX Buffer or FIFO Not Full Indicates whether the buffer, FIFOm or message register can accept another byte or halfword. FIFO uses trigger level. If DMA enabled for transmitting, it transfers data as long as it is not full. 0 - Receive buffer or FIFO full 1 - Receive buffer or FIFO not full
11 RXPEND	RXPEND Indicates whether a message is being received from a target and bytes are in the input buffer or FIFO. • If using a FIFO, this message is at least one FIFO trigger's worth (a minimum of one byte in the FIFO). • If DMA is enabled for receiving, the DMA is signaled. RXPEND becomes 0 when the data is read. 0 - No receive message pending 1 - Receive message pending
10 COMPLETE	Complete Indicates whether a message has completed. • With MWMSG_SDR or MWMSG_DDR, this condition occurs when MWMSG_SDR_CONTROL[LEN] or MWMSG_DDR_CONTROL[LEN] reaches 0. • When MCTRL[REQUEST] = 1 (EmitStartAddr), this condition occurs after the end of a write operation, or after a read operation has terminated or ended. • With an IBI (AutoIBI or EmitStartAddr), this condition occurs at the end of IBI data (if any). 0 - Not complete 1 - Complete

Table continues on the next page...

Table continued from the previous page...

Field	Description
9 MCTRLDONE	Controller Control Done Indicates whether the module has completed an MCTRL request. MCTRLDONE automatically becomes 0 when writing a new control. When MCTRL[REQUEST] = 1 (EmitStartAddr), MCTRLDONE becomes 1 when the address goes out (and is ACKed, NACKed, or ended in an IBI). If ACKed, MSTATUS[COMPLETE] becomes 1 when the write or read data has completed. When MCTRL[REQUEST] = 4 (ProcessDAA), MCTRLDONE becomes 1 when the module is ready to emit the Dynamic Address (DA) for the target, or when no more targets are ACKing. This condition can be determined by using the MSTATUS[BETWEEN] and MSTATUS[STATE] fields. 0 - Not done 1 - Done
8 SLVSTART	Target Start Indicates whether a target is/was requesting a START by holding SDA low. Handling starts automatically when MCTRL[REQUEST] = 7 (AutoIBI). 0 - Target not requesting START 1 - Target requesting START
7-6 IBITYPE	In-Band Interrupt (IBI) Type Indicates the type of IBI of the last event that won the arbitration, whether the interrupt is ACKED or NACKED or pending. 00 - NONE. No IBI. This status occurs when MSTATUS[IBIWON] becomes 0. 01 - In-Band Interrupt 10 - Controller Request 11 - Hot-Join
5 NACKED	Not Acknowledged Indicates whether the last Start and Address sequence was NACKed (was not ACKed by the addressed target). 0 - Not NACKed 1 - NACKed (not acknowledged)
4 BETWEEN	Between Between messages or Dynamic Address Assignments (DAA). Active when: • MSTATUS[STATE] is MSGSDR, DDR, or DAA, and the state is between messages/DAAs. It is expecting a new messages/DAAs to start (or STOP or Exit). • MSTATUS[STATE] is NORMACT. The module is waiting on the Transmit FIFO to be not empty or the Receive FIFO to be not full. 0 - Inactive. For other cases.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Active. If STATE is MSGSDR, DDR, DAA, or NORMACT.
3 —	Reserved
2-0 STATE	State Of The Controller Indicates the current controller state 000 - IDLE. Bus has stopped. 001 - SLVREQ. Target request. The bus has stopped but a target is holding SDA low. If using auto-emit IBI (MCTRL[REQUEST] = 7), the controller will not remain in this state. 010 - MSGSDR. Single Data Rate Message mode, from using MWMSG_SDR 011 - NORMACT. Normal active SDR mode, from using MCTRL and MWDATAn and MRDATAn registers. The controller remains in this state until a STOP is issued. 100 - MSGDDR. Double Data Rate Message mode, from using MWMSG_DDR or using the normal method with DDR. The controller remains in the DDR state until the controller exits using EXIT (emitting the Exit pattern). 101 - DAA. Enter Dynamic Address Assignment (ENTDAA) mode 110 - IBIACK. Waiting for an In-Band Interrupt (IBI) ACK/NACK decision 111 - IBIRCV. Receiving an In-Band Interrupt (IBI). This state is used after IBI/CR/HJ has won an arbitration. IBIRCV state is also used for IBI mandatory byte (if any) and any bytes that follow.

42.7.1.29 Controller In-band Interrupt Registry and Rules (MIBIRULES)

Contains the rules for using IBI, and keeps a registry of the targets that use the IBI byte.

Defines the IBI mandatory byte rules for the targets. Determines which targets do (or do not) have a mandatory byte.

Concerning ADDRn fields: The address is six bits. Assuming that MIBIRULES[MSB0] = 1, the most significant bit of each address is 0. In this case, each address is seven bits (usually written as A7 to A1) and A7 must be 0. If the application does not use that optimal convention, the Manual method of IBI ACK handling must be used (see MCTRL[IBIRESP]).

By default, the ADDRn values indicate the targets with a mandatory byte. If MIBIRULES[NOBYTE] = 1, the ADDRn values indicate targets that do not have a mandatory IBI byte.

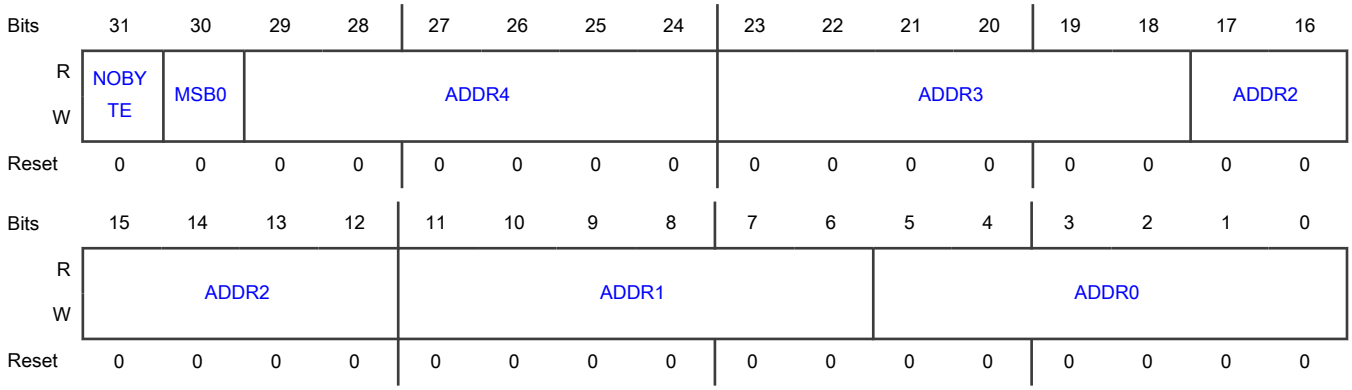
NOTE

A7 = 0 is only needed for targets that use IBI. For legacy I2C devices, A7 can use any valid value, because I2C devices cannot cause an IBI.

Offset

Register	Offset
MIBIRULES	8Ch

Diagram



Fields

Field	Description
31 NOBYTE	No IBI byte Indicates whether the ADDRn fields refer to targets with or without a mandatory IBI byte. 0 - With mandatory IBI byte 1 - Without mandatory IBI byte
30 MSB0	Most Significant Address Bit Is 0 Assigning 1 to this field allows the START header to be optimized. 0 - MSB is not 0. 1 - For all I3C dynamic addresses, MSB is 0.
29-24 ADDR4	ADDR4 Address of target with or without Mandatory IBI byte. If 0, then the address does not apply.
23-18 ADDR3	ADDR3 Address of target with or without Mandatory IBI byte. If 0, then the address does not apply.
17-12 ADDR2	ADDR2 Address of target with or without Mandatory IBI byte. If 0, then the address does not apply.
11-6 ADDR1	ADDR1 Address of target with or without Mandatory IBI byte. If 0, then the address does not apply.
5-0 ADDR0	ADDR0 Address of target with or without Mandatory IBI byte. If 0, then the address does not apply.

42.7.1.30 Controller Interrupt Set (MINTSET)

Set interrupt enables for select bits in Controller Status (MSTATUS) . Reading the MINTSET register returns the status of the interrupt enables.

- To activate an interrupt enable, write 1 to its field.

- To disable an interrupt enable, write 1 to the appropriate field in the Controller Interrupt Clear (MINTCLR) register. Writing 0 to the interrupt enable in this register (MINTSET) does not disable the interrupt.

The interrupt registers allow the masking of interrupt sources, as well as checking which interrupts have activated in the MSTATUS register. The normal method is to enable an interrupt and then once that interrupt fires, clear the interrupt either by writing the MSTATUS register or via an action on the corresponding data register. The interrupt is level-held, meaning that the interrupt remains 1 until the cause is cleared by some method. The module prevents races; if a new event comes in, that new event is not lost.

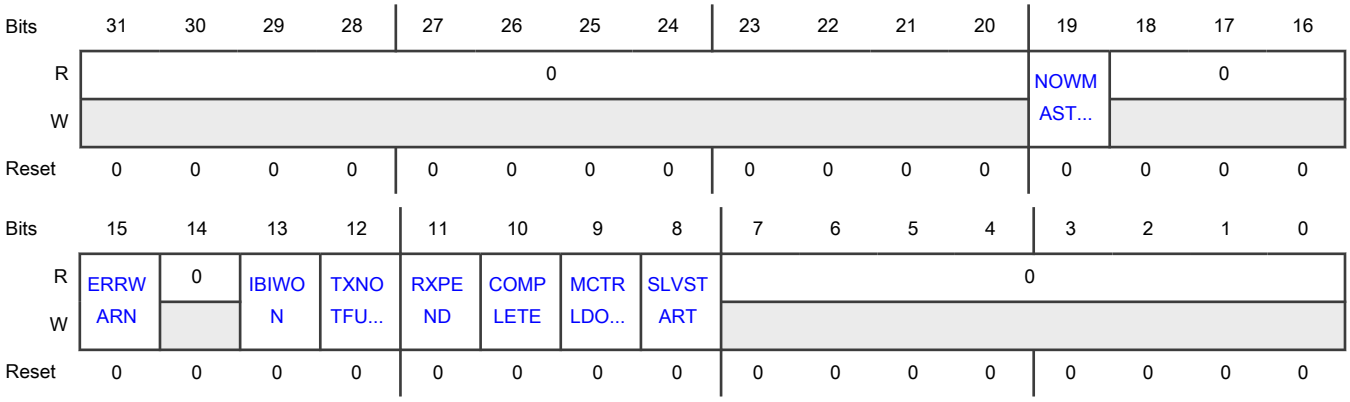
These interrupts are parallel to the Target Interrupt Set (SINTSET) / Target Interrupt Clear (SINTCLR) set, and only one interrupt set is active depending on state (controller or target operation).

- MINTSET: Sets interrupt enables for MSTATUS bits. Reading MINTSET register returns the status of the interrupt enables.
- MINTCLR: Clears interrupt enables for MSTATUS bits.
- MINTMASKED: Returns the value of the MSTATUS bits ANDed with their interrupt enables.

Offset

Register	Offset
MINTSET	90h

Diagram



Fields

Field	Description
31-20 —	Reserved
19 NOWMASTER	Now Controller (now this I3C module is a controller) Interrupt Enable 0 - Disable 1 - Enable
18-16 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
15 ERRWARN	Error or Warning (ERRWARN) Interrupt Enable 0 - Disable 1 - Enable
14 —	Reserved
13 IBIWON	In-Band Interrupt (IBI) Won Interrupt Enable 0 - Disable 1 - Enable
12 TXNOTFULL	TX buffer/FIFO is not full interrupt enable 0 - Disable 1 - Enable
11 RXPEND	Rx Pending Interrupt Enable
10 COMPLETE	Completed Message Interrupt Enable 0 - Disable 1 - Enable
9 MCTRLDONE	Controller Control Done Interrupt Enable 0 - Disable 1 - Enable
8 SLVSTART	Target Start Interrupt Enable 0 - Disable 1 - Enable
7-0 —	Reserved

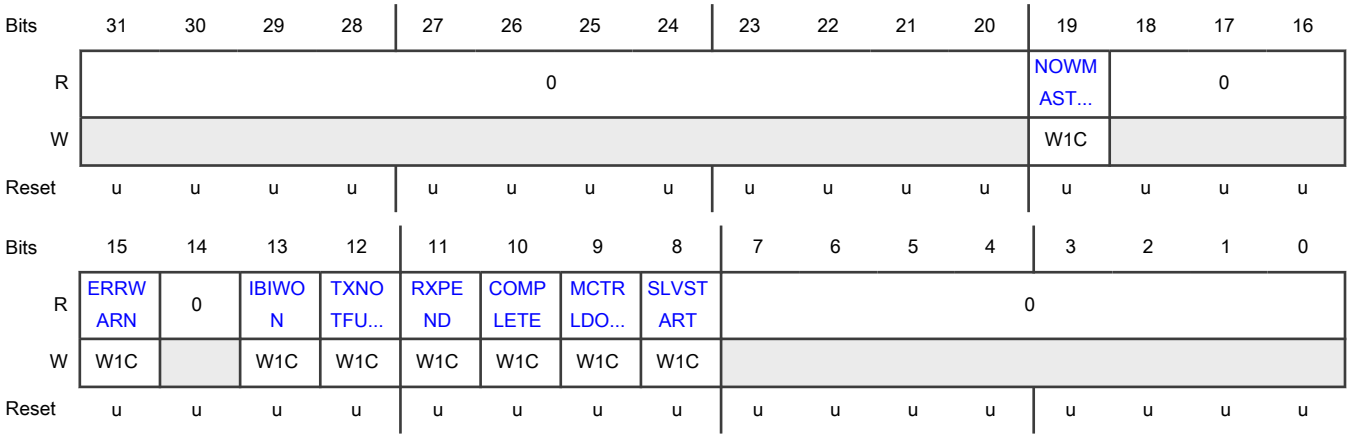
42.7.1.31 Controller Interrupt Clear (MINTCLR)

Clears interrupt enables for select Controller Status (MSTATUS) fields. Writing a 1 clears the corresponding interrupt enable. Writing a 0 has no effect.

Offset

Register	Offset
MINTCLR	94h

Diagram



Fields

Field	Description
31-20 —	Reserved
19 NOWMASTER	NOWCONTROLLER Interrupt Enable Clear 0 - No effect 1 - Corresponding interrupt enable becomes 0
18-16 —	Reserved
15 ERRWARN	ERRWARN Interrupt Enable Clear 0 - No effect 1 - Corresponding interrupt enable becomes 0
14 —	Reserved
13 IBIWON	IBIWON Interrupt Enable Clear 0 - No effect 1 - Corresponding interrupt enable becomes 0
12 TXNOTFULL	TXNOTFULL Interrupt Enable Clear 0 - No effect 1 - Corresponding interrupt enable becomes 0
11 RXPEND	RXPEND Interrupt Enable Clear 0 - No effect 1 - Corresponding interrupt enable becomes 0

Table continues on the next page...

Table continued from the previous page...

Field	Description
10 COMPLETE	COMPLETE Interrupt Enable Clear 0 - No effect 1 - Corresponding interrupt enable becomes 0
9 MCTRLDONE	MCTRLDONE Interrupt Enable Clear 0 - No effect 1 - Corresponding interrupt enable becomes 0
8 SLVSTART	SLVSTART Interrupt Enable Clear 0 - No effect 1 - Corresponding interrupt enable becomes 0
7-0 —	Reserved

42.7.1.32 Controller Interrupt Mask (MINTMASKED)

Returns the status of enabled interrupts (the value of Controller Status (MSTATUS) ANDed with the value of Controller Interrupt Set (MINTSET)).

Offset

Register	Offset
MINTMASKED	98h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												NOWMAST...	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	ERRWARN	0	IBIWON	TXNOTFU...	RXPEND	COMPLETE	MCTRLDO...	SLVSTART	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-20 —	Reserved
19 NOWMASTER	NOWCONTROLLER Interrupt Mask 0 - Interrupt not enabled and/or not active 1 - Interrupt enabled and active
18-16 —	Reserved
15 ERRWARN	ERRWARN Interrupt Mask 0 - Interrupt not enabled and/or not active 1 - Interrupt enabled and active
14 —	Reserved
13 IBIWON	IBIWON Interrupt Mask 0 - Interrupt not enabled and/or not active 1 - Interrupt enabled and active
12 TXNOTFULL	TXNOTFULL Interrupt Mask 0 - Interrupt not enabled and/or not active 1 - Interrupt enabled and active
11 RXPEND	RXPEND Interrupt Mask
10 COMPLETE	COMPLETE Interrupt Mask 0 - Interrupt not enabled and/or not active 1 - Interrupt enabled and active
9 MCTRLDONE	MCTRLDONE Interrupt Mask 0 - Interrupt not enabled and/or not active 1 - Interrupt enabled and active
8 SLVSTART	SLVSTART Interrupt Mask 0 - Interrupt not enabled and/or not active 1 - Interrupt enabled and active
7-0 —	Reserved

42.7.1.33 Controller Errors and Warnings (MERRWARN)

Contains errors and warnings from I3C/I2C protocol. When any errors or warnings are not 0, then the MSTATUS[ERRWARN] bit is 1.

Parallel-to-target ERRWARN:

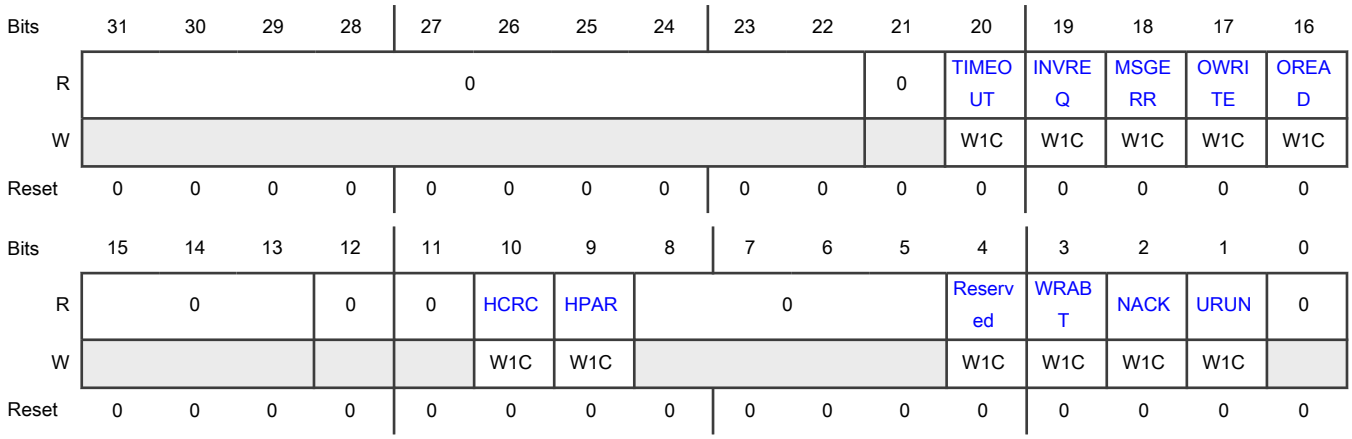
- In Controller mode, use this register (MERRWARN).
- In Target mode, use the Target Errors and Warnings (SERRWARN) register.

The error fields in both registers (MERRWARN and SERRWARN) are similar in meaning.

Offset

Register	Offset
MERRWARN	9Ch

Diagram



Fields

Field	Description
31-22 —	Reserved
21 —	Reserved
20 TIMEOUT	Timeout Error Indicates an error caused by the module stalling for too long in a frame. This stalling occurs when: • The Transmit FIFO or Receive FIFO is not handled, and the bus is stuck in the middle of a message. • No STOP is issued and between messages. • IBI manual is used and no decision has been made. The maximum stall period is 10 kHz or 100 s.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - No error 1 - Error
19 INVREQ	Invalid Request Error Indicates an error caused by an invalid use of a request: • Not using IBI ACK NACK when stopped in manual hold for IBI acknowledgment. • Using a request other than ForceStop or ForceExit while in a message. Other requests are valid when the message is done. • Other mismatched uses (for example, IBI ACK NACK while in normal states). 0 - No error 1 - Error
18 MSGERR	Message Error Indicates an error caused by : • Trying to write to or read from MWMSG_SDR register while in a DDR message. • Trying to write to or read from MWMSG_DDR register while in a SDR message. • Trying to read from HRMSG_SDR or HRMSG_DDR registers when no message has yet started. 0 - No error 1 - Error
17 OWRITE	Over-write Error Indicates an error caused by trying to write to the Controller Write Data Byte Register (MWDATAB) when the FIFO is full. 0 - No error 1 - Error
16 OREAD	Over-read Error Indicates an error caused by trying to read from the Controller Read Data Byte (MRDATAB) register when the FIFO is empty. 0 - No error 1 - Error
15-13 —	Reserved
12 —	Reserved
11	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
10 HCRC	High Data Rate CRC Error Indicates that a Cyclic Redundancy Check (CRC) error occurred from a DDR read. 0 - No error 1 - Error
9 HPAR	High Data Rate Parity Indicates a parity error from a DDR read, including a bad preamble on a read. Does not stop the read, because it is not safe to terminate; the read data may become mis-framed. Ends on a run of 1 second. 0 - No error 1 - Error
8-5 —	Reserved
4 —	Reserved
3 WRABT	Write Abort Error Indicates an error caused by the I2C target NACKing the write data, terminating the message. For example, the controller is writing in I2C and the Target NACKED the write. If the MCTRL register is written, this field automatically becomes 0. 0 - No error 1 - Error
2 NACK	Not Acknowledge Error Indicates an error caused by the target or targets NACKing (not acknowledging) the last address. If 7Eh was the address, then all targets NACKed the last address. If the MCTRL register is written, this field automatically becomes 0. **NOTE** In HDR mode, this error occurs when an address is not accepted (as opposed to NACKed). 0 - No error 1 - Error
1 URUN	Underrun error Indicates an underrun for HDR-BT. This error occurs when attempting to write from an empty transmit FIFO. 0 - No error

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Error
0 —	Reserved

42.7.1.34 Controller DMA Control (MDMACTRL)

DMA can be used with a controller in these ways:

- Push or pull data to accompany an MCTRL[REQUEST] = 1 (EmitStartAddr) request written by the processor.
- Implementing message mode completely controlled by DMA.

Offset

Register	Offset
MDMACTRL	A0h

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		DMAWIDTH		DMATB		DMAFB	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Fields

Field	Description
31-8 —	Reserved
7-6 —	Reserved
5-4 DMAWIDTH	DMA Width Specifies the data width of DMA operations.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00,01 - Byte 10 - Halfword (16 bits). This setting ensures that two bytes are free/available in FIFO. 11 - Reserved
3-2 DMATB	DMA To Bus Represents the DMA Write (to-bus) trigger. When DMAFB = 1 or DMAFB = 2, the I3C module starts request DMA when a transmit trigger occurs. See Controller Data Control (MDATACTRL) register. The I3C module requests until full, unless the DMA is set up as a trigger. DMAFB becomes 0 when MSTATUS[ERRWARN] = 1. 00 - DMA is not used 01 - Enable DMA for one frame (ended by DMA or Terminated). STOP or START causes DMATB to become 0. See SCONFIG[MATCHSS] . 10 - Enable DMA until DMA is turned off. Normally, DMA ENABLE should only be used in Controller Message mode. 11 - Reserved
1-0 DMAFB	DMA From Bus Represents the DMA Read (from-bus) trigger. When DMAFB = 1 or DMAFB = 2, the I3C module requests DMA when a receive trigger occurs. See Controller Data Control (MDATACTRL) register. The I3C module requests until empty, unless the DMA is set up as a trigger. DMAFB becomes 0 when MSTATUS[ERRWARN] = 1. 00 - DMA is not used 01 - Enable DMA for one frame. STOP or repeated START causes DMAFB to automatically become 0. See SCONFIG[MATCHSS] . 10 - Enable DMA until DMA is turned off 11 - Reserved

42.7.1.35 Controller Data Control (MDATACTRL)

Assists in data control when there is no FIFO. It also assists the FIFO when FIFO is available (regardless of size). This assistance allows some control over the FIFO behavior. In particular, it provides control over when to interrupt when a particular state of fullness or emptiness is reached. It also controls behavior related to width, when the width is not one-byte wide.

NOTE

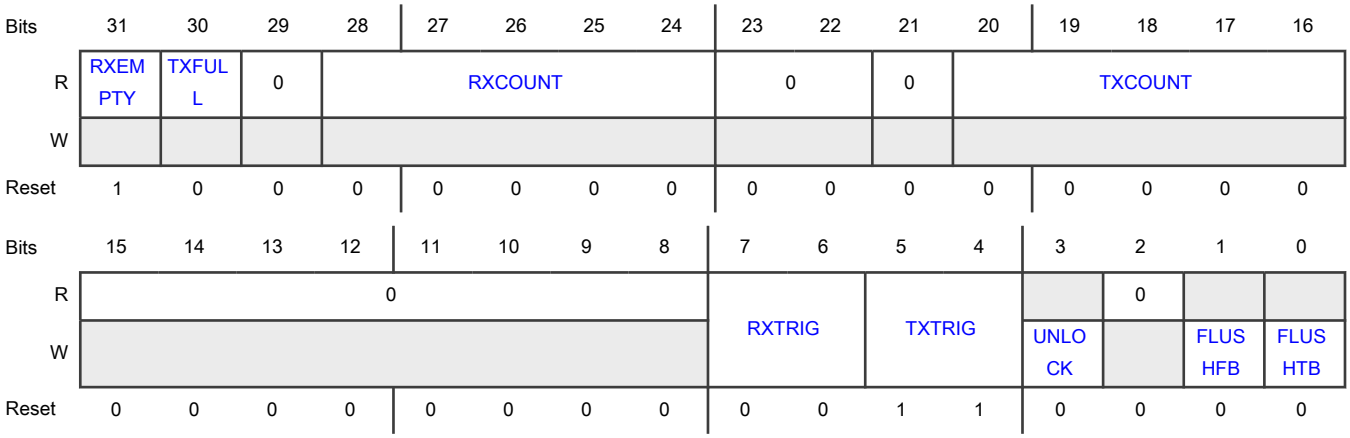
When flushing a FIFO while DMA is in use, disable the DMA channel first. FIFO flush should not be used when a message (in that direction) is in flight.

The Controller Data Control register is an alias of the Target Data Control (SDATACTRL) register.

Offset

Register	Offset
MDATACTRL	ACh

Diagram



Fields

Field	Description
31 RXEMPTY	Receive Is Empty 0 - Receive FIFO or buffer is not yet empty. 1 - Receive FIFO or buffer is empty.
30 TXFULL	Transmit Is Full 0 - Transmit FIFO or buffer is not yet full. 1 - Transmit FIFO or buffer is full.
29 —	Reserved
28-24 RXCOUNT	Receive Byte Count Contains the count of bytes in the Receive FIFO or buffer.
23-22 —	Reserved
21 —	Reserved
20-16 TXCOUNT	Transmit Byte Count

Table continues on the next page...

Table continued from the previous page...

Field	Description
	Contains the count of bytes waiting in the TXFIFO. This count is the number of bytes the application has written to the Transmit FIFO that have not yet gone onto the I3C bus.
15-8 —	Reserved
7-6 RXTRIG	Receive Trigger Level Indicates trigger level for Receive fullness when using a FIFO. Affects RXPEND interrupt. 00 - Trigger when not empty 01 - Trigger when 1/4 full or more 10 - Trigger when 1/2 full or more 11 - Trigger when 3/4 full or more
5-4 TXTRIG	Transmit Trigger Level Indicates trigger level for Transmit emptiness when using a FIFO. Affects TXNOTFULL interrupt. 00 - Trigger when empty 01 - Trigger when 1/4 full or less 10 - Trigger when 1/2 full or less 11 - Default. Trigger when 1 less than full or less
3 UNLOCK	Unlock **NOTE** UNLOCK must be 1 in the same cycle while writing to TXTRIG or RXTRIG. 0 - Locked. RXTRIG and TXTRIG fields cannot be changed on a write. 1 - Unlocked. RXTRIG and TXTRIG fields can be changed on a write.
2 —	Reserved
1 FLUSHFB	Flush From-bus Buffer or FIFO FLUSHFB is not normally used. 0 - No action 1 - Flush the buffer
0 FLUSHTB	Flush To-bus Buffer or FIFO Used when the controller terminates a to-bus message (read) prematurely. 0 - No action 1 - Flush the buffer

42.7.1.36 Controller Write Data Byte (MWDATAB)

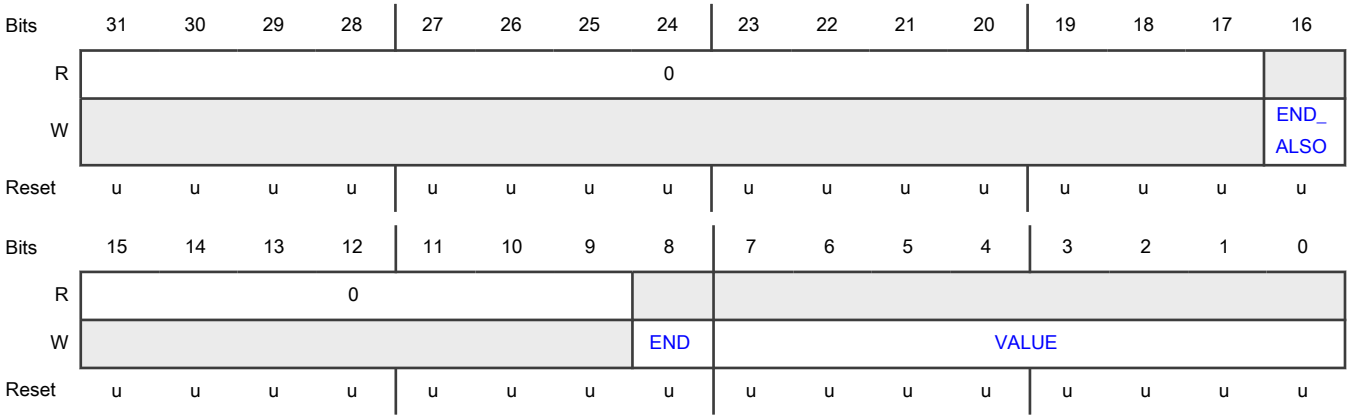
Allows writing bytes to send onto the bus. Only used when using Controller Control (MCTRL) to start the message. If MWMSG_SDR or MWMSG_DDR is used to start a message, that interface must be used exclusively.

The MWDATAB register is the alias of the Target Write Data Byte (SWDATAB) register.

Offset

Register	Offset
MWDATAB	B0h

Diagram



Fields

Field	Description
31-17 —	Reserved
16 END_ALSO	End of Message Also Indicates end of message, used to end outbound message normally. Every message must indicate when it is the last message to be sent. This method can be used with the MDATABE register. This field is required for I3C and is optional for I2C. For HDR-DDR, the byte with the END_ALSO must be an even byte (second, fourth, sixth, and so on) because DDR uses byte-pairs. 0 - Not the end. More bytes are assumed to be in the message. 1 - End. The END bit marks the last byte of the message.
15-9 —	Reserved
8 END	End of Message

Table continues on the next page...

Table continued from the previous page...

Field	Description
	Indicates the end of a message. This field is required for I3C and is optional for I2C. For HDR-DDR, the byte with the END must be an even byte (second, fourth, sixth, and so on) because DDR uses byte-pairs. 0 - Not the end. More bytes are assumed to be in the message. 1 - End. The END bit marks the last byte of the message.
7-0 VALUE	Data Byte Represents the byte written to the target (stored in Transmit FIFO). • If I3C, the block computes the parity. • If I2C, the block manages the ACK or NACK.

42.7.1.37 Controller Write Data Byte End (MWDATABE)

Allows writing the last byte to send onto the bus. Only used when using Controller Control (MCTRL) to start the message. If MWMSG_SDR or MWMSG_DDR is used to start a message, that interface must be used exclusively. The MWDATABE register is the alias of the Target Write Data Byte End (SWDATABE) register.

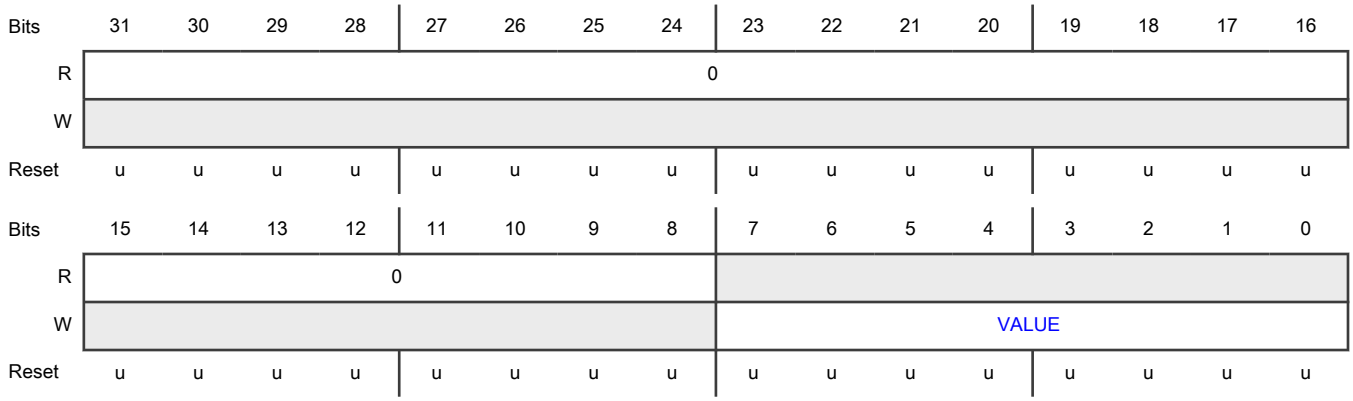
NOTE

MWDATAB can also indicate END using bit 8.

Offset

Register	Offset
MWDATABE	B4h

Diagram



Fields

Field	Description
31-8 —	Reserved
7-0 VALUE	Data Represents the last byte written to the target (stored in Transmit FIFO). • If I3C, the block computes the parity. • If I2C, the block manages the ACK or NACK.

42.7.1.38 Controller Write Data Halfword (MWDATAH)

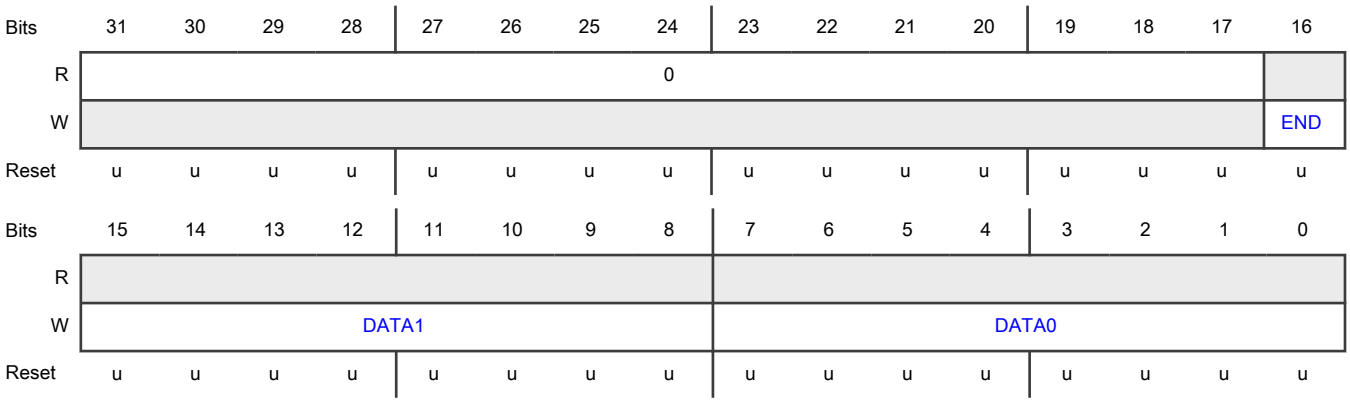
Allows writing a halfword (pair of bytes) to the bus unless an external FIFO is used. Sends the low byte followed by the high byte. An end-of-data (last) marker bit is allowed (or must be 0). A halfword should not be written unless there is room for both, as indicated by the use of Transmit FIFO level trigger or MDATACTRL[TXCOUNT] .

The MWDATAH register is the alias of the Target Write Data Half-word (SWDATAH) register.

Offset

Register	Offset
MWDATAH	B8h

Diagram



Fields

Field	Description
31-17 —	Reserved
16	End of message

Table continues on the next page...

Table continued from the previous page...

Field	Description
END	Indicates the end of a message. For this register, this field always marks DATA1 as the end. This field is required for I3C and is optional for I2C. For HDR-DDR, the byte with the END must be an even byte (second, fourth, sixth, and so on) because DDR uses byte-pairs. 0 - Not the end. More bytes are assumed to be in the message. 1 - End. The END bit marks the last byte of the message.
15-8 DATA1	Data Byte 1 Represents the second byte that is sent to the target (written to Transmit FIFO).
7-0 DATA0	Data Byte 0 Represents the first byte that is sent to the target (written to Transmit FIFO).

42.7.1.39 Controller Write Data Halfword End (MWDATAHE)

Allows writing a half word of data, which is the end (the last byte of the half word is the end). Writes the half word (byte pair) just like Controller Write Data Halfword (MWDATAH) , but marks the second byte as end-of-data (last byte).

For HDR-DDR, the byte with the END must be an even (second, fourth, sixth, and so on) because DDR uses byte-pairs.

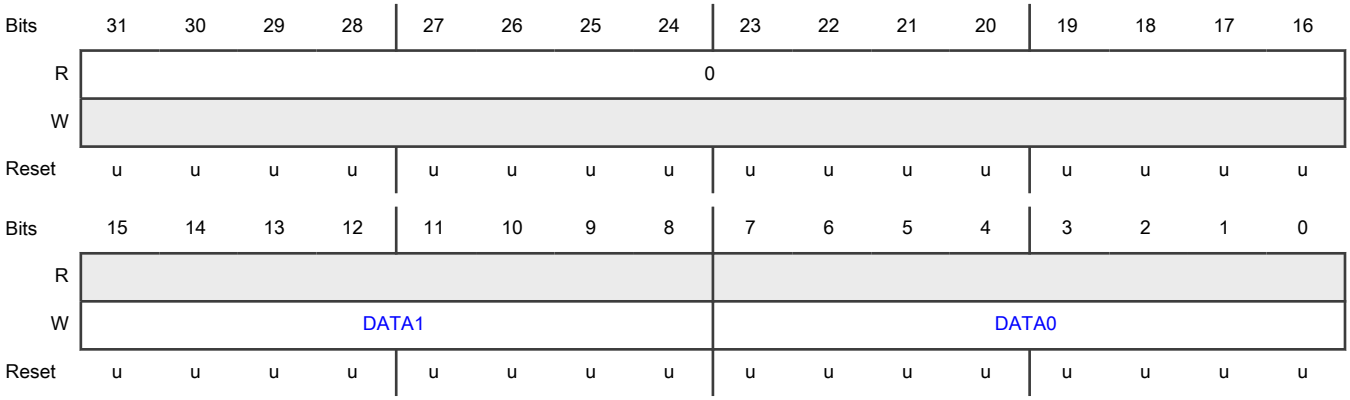
A half-word should not be written unless there is room for both, as indicated by the use of Transmit FIFO level trigger or MDATACTRL[TXCOUNT] .

The MWDATAHE register is the alias of the Target Write Data Half-word End (SWDATAHE) register.

Offset

Register	Offset
MWDATAHE	BCh

Diagram



Fields

Field	Description
31-16 —	Reserved
15-8 DATA1	Data Byte 1 Represents the second byte that is sent to the target (written to Transmit FIFO).
7-0 DATA0	Data Byte 0 Represents the first byte that is sent to the target (written to Transmit FIFO).

42.7.1.40 Controller Read Data Byte (MRDATAB)

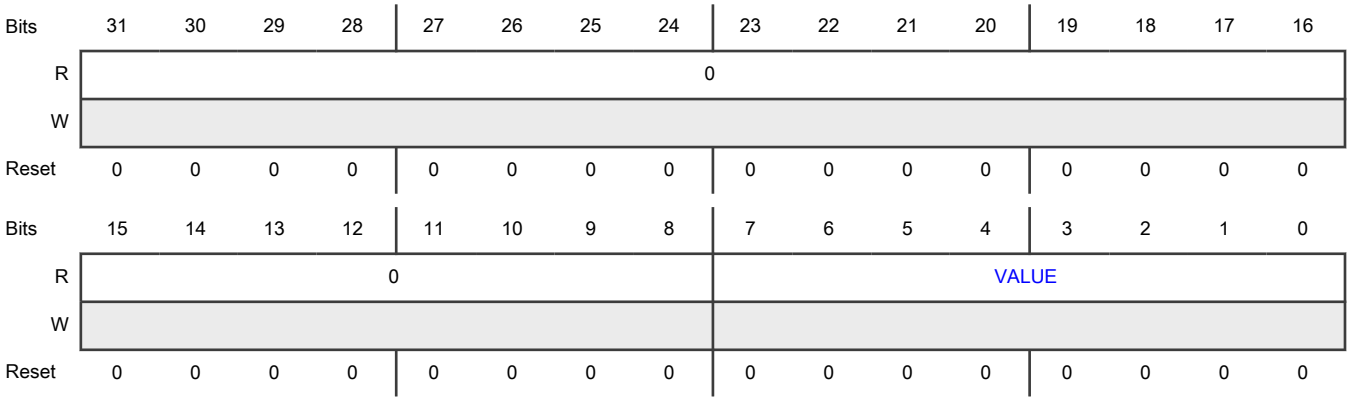
Allows reading bytes written by the target after an SDR Read, or DAA or DDR. Only used when using Controller Control (MCTRL) to start the message. If MWMSG_SDR or MWMSG_DDR is used to start a message, that interface must be used exclusively.

This register is the alias of the Target Read Data Byte (SRDATAB) register.

Offset

Register	Offset
MRDATAB	C0h

Diagram



Fields

Field	Description
31-8 —	Reserved
7-0 VALUE	Value Represents the byte read from the controller (and written by the target).

42.7.1.41 Controller Read Data Halfword (MRDATAH)

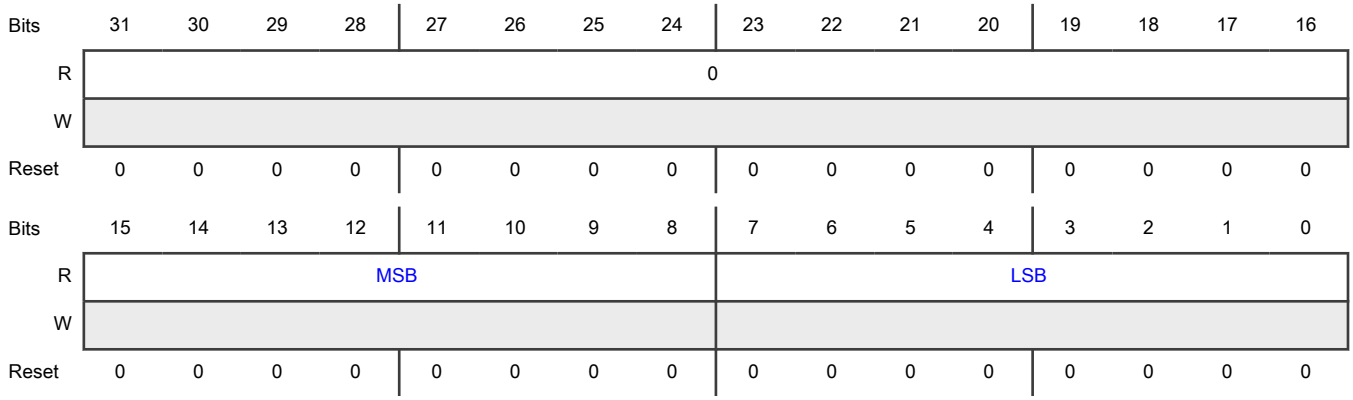
Allows reading a halfword (byte pair) written by the target after an SDR Read or DAA or DDR. This register is only used when using Controller Control (MCTRL) to start the message. If MWMSG_SDR or MWMSG_DDR is used to start a message, that interface must be used exclusively.

A halfword should not be read unless there are at least two bytes of data waiting, as indicated the Receive FIFO level trigger or MDATACTRL[RXCOUNT] .

Offset

Register	Offset
MRDATAH	C8h

Diagram



Fields

Field	Description
31-16 —	Reserved
15-8 MSB	MSB Represents the second byte read from the controller (and written by the target).
7-0 LSB	LSB Represents the first byte read from the controller (and written by the target).

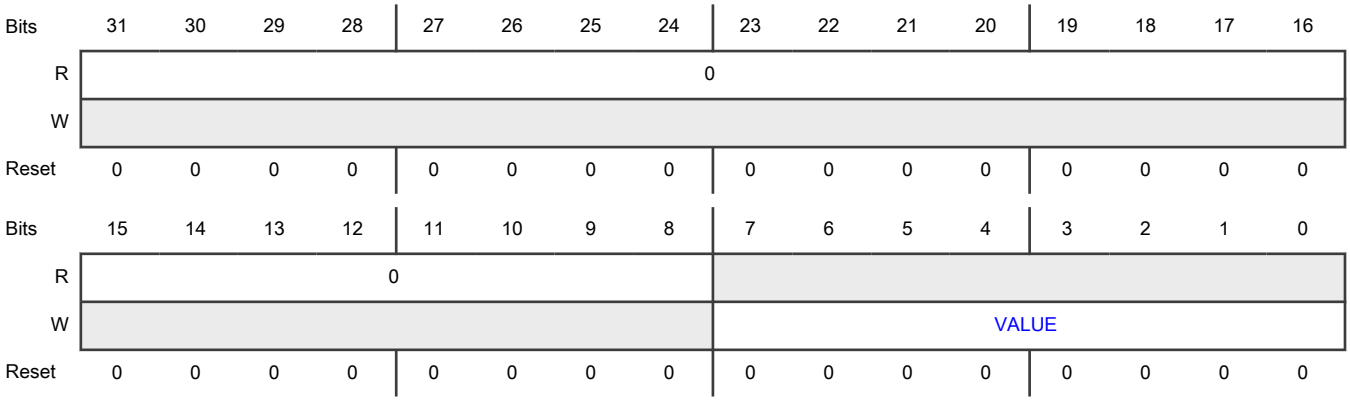
42.7.1.42 Controller Write Byte Data 1(to bus) (MWDATAB1)

Allows writing bytes to send onto the bus, and is intended for DMAs which do not clear the upper bits of the word. It does not have the END bits. This register is only used when using Controller Control (MCTRL) to start the message. If MWMSG_SDR or MWMSG_DDR is used to start a message, that interface must be used exclusively.

Offset

Register	Offset
MWDATAB1	CCh

Diagram



Fields

Field	Description
31-8 —	Reserved
7-0 VALUE	Value Represents byte to write out. The block computes the parity for I3C, or manages the ACK or NACK for I2C.

42.7.1.43 Controller Write Message Control in SDR mode (MWMSG_SDR_CONTROL)

Allows setting up and writing 16-bit words in Single Data Rate (SDR) mode. The MWMSG_SDR_ register is modal and has two modes: control and data.

- For the first write to set up a new message, this register functions as the MWSMSG_SDR_CONTROL register.
- For subsequent writes, this register functions as the MWMSG_SDR_DATA register.
- The control information is not pipelined. Write control information registers for a start and then again only when the data to be sent is done.

When not in the middle of a message, the MWMSG_SDR_CONTROL register is used to start a new message (or emit a STOP). After starting a message, the MWMSG_SDR_DATA register is used until the Length (see LEN field) counts down, or until data with END = 1 is used.

The MWMSG_SDR_CONTROL register is written with the 16-bit control word if currently stopped or after the end of the previous message. If MSTATUS[STATE] = 2 (MSGSDR) and MSTATUS[BETWEEN] = 0, the register (at this offset address) functions as the MWMSG_SDR_DATA register. Otherwise, this register (at this offset address) functions as the MWMSG_SDR_CONTRL register, as long as MSTATUS[STATE] is not in another mode.

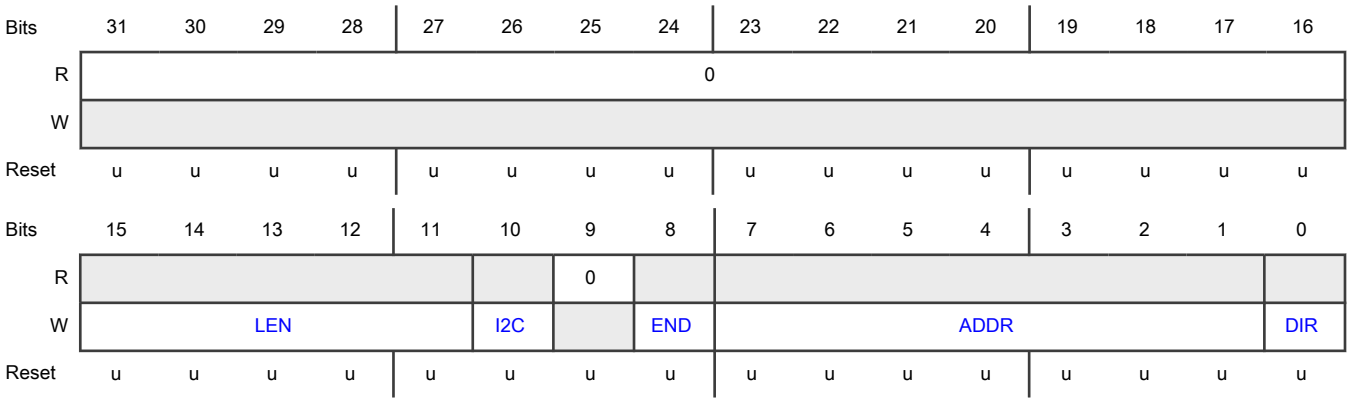
The control word contains the byte length (6-bit), the address, the direction, and how it ends (stop, ready for next, continuation with more length). If the command is START and an event (IBI, CR, HJ) occurs, MCTRL[IBIRESP] is used to determine action, and the corresponding interrupt will occur. In that case, the message is restarted.

The MWMSG_SDR_CONTROL and MWMSG_SDR_DATA registers are oriented to DMA operations, but the MWMSG_SDR_CONTROL register can also be written by the processor (instead of using MCTRL and MxDATAB).

Offset

Register	Offset
MWMSG_SDR_CONTROL	D0h

Diagram



Fields

Field	Description
31-16 —	Reserved
15-11 LEN	Length Indicates the byte length of the message. If LEN = 0, then only END is used (and it will not use the address if Stopped). LEN = 1 must not be used. The minimal LEN size is 2 bytes (LEN = 2).
10 I2C	I2C Specifies whether the message is I2C or I3C. 0 - I3C message 1 - I2C message
9 —	Reserved
8 END	End of SDR Message Indicates end of SDR message. MSTATUS[COMPLETE] = 1 when done. The end can happen: • Before LEN bytes are read in I3C mode, if a target ends sooner.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	• Before LEN bytes are written in I2C mode, if a target NACKs. 0 - Not the end. SDR message ends waiting for a new SDR message (issues a repeated START for a new message). 1 - End. SDR message ends at the STOP.
7-1 ADDR	Address Contains address to be written.
0 DIR	Direction 0 - Write 1 - Read

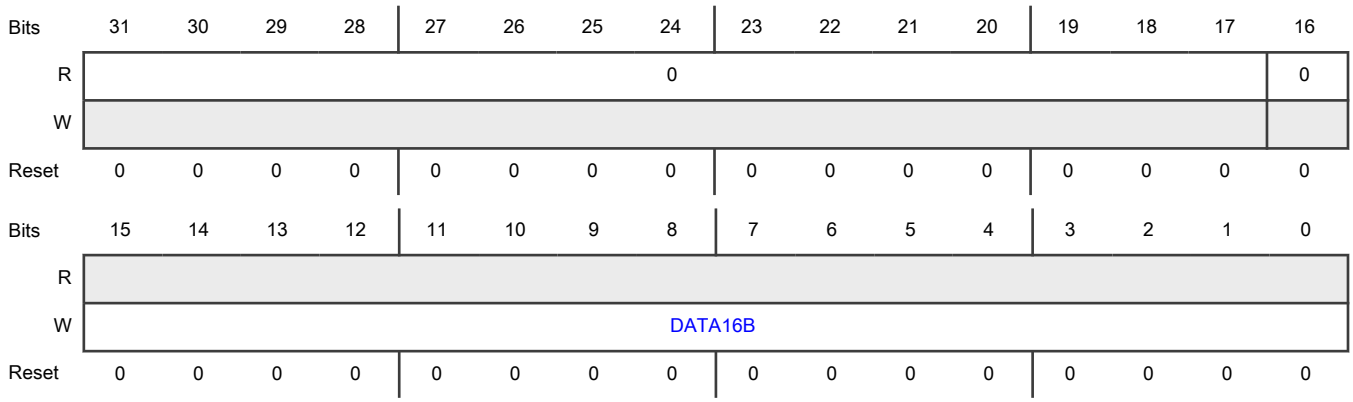
42.7.1.44 Controller Write Message Data in SDR mode (MWMSG_SDR_DATA)

Contains the 16-bit word to be written in Single Data Rate (SDR) mode. This register also functions as the MWMSG_SDR_CONTROL register.

Offset

Register	Offset
MWMSG_SDR_DATA	D0h

Diagram



Fields

Field	Description
31-17 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
16 —	Reserved
15-0 DATA16B	Data

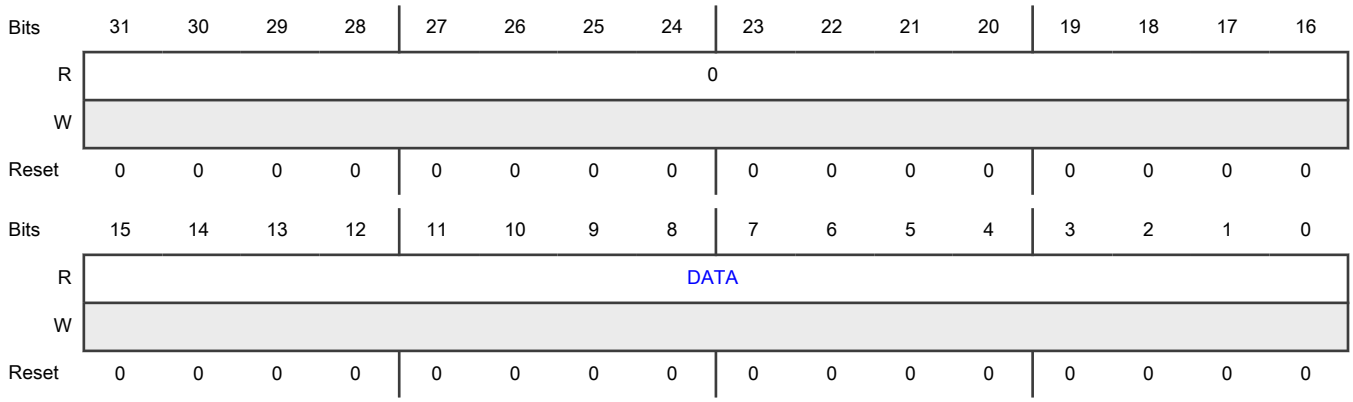
42.7.1.45 Controller Read Message in SDR mode (MRMSG_SDR)

Allows reading 16-bit words from a target in SDR message mode. The MRMSG_SDR register is used to read 16-bit words from an active message started with MWMSG_SDR. These words are intended to be read by DMA.

Offset

Register	Offset
MRMSG_SDR	D4h

Diagram



Fields

Field	Description
31-16 —	Reserved
15-0 DATA	Data Contains the 16-bit word read from the target. • If the length (LEN) was an odd number, the upper byte is 0. • If the target ends before LEN is finished, the module treats the read as completed. • If the target is not done before LEN is finished and END is not a continuation, the read is terminated.

42.7.1.46 Controller Write Message in DDR mode: First Control Word (MWMSG_DDR_CONTROL)

Allows setting up and writing 16-bit words in Double Data Rate (DDR) mode. The register has three modes.

- The first write to set up a new message, functions as the MWSMSG_DDR_CONTROL register.
- The second write contains the functions as shown in the MWMSG_DDR_CONTROL2 register.
- For subsequent writes, this register functions as the MWMSG_DDR_DATA register.
- The control information is not pipelined. Write control information registers at the start and then again only once the data to be sent is done.

When not in the middle of a message, the MWMSG_DDR_CONTROL register is used to start a new message (or emit a STOP). After starting a message, the MWMSG_DDR_DATA register is used until the Length (see LEN field) counts down or until data with END = 1 is used.

The MWMSG_DDR_CONTROL register is written with the 16-bit Control word if currently stopped or after the end of the previous message. If MSTATUS[STATE] = 4 (MSGDDR) and MSTATUS[BETWEEN] = 0, then the register (at this offset address) functions as the MWMSG_DDR_DATA register. Otherwise, this register (at this offset address) functions as the MWMSG_DDR_CONTROL register, as long as MSTATUS[STATE] is not in another mode.

The main control word contains the 16-bit word length and how it ends (stop, ready for next, continuation with more length). Then the command word contains the command and address for read or write. If the command is START and an event (IBI, CR, HJ) occurs, MCTRL[IBIRESP] is used to determine action, and the corresponding interrupt will occur. In that case, the message is restarted.

NOTE

The module handles preamble, parity, and CRC.

The MWMSG_DDR_CONTROL, MWMSG_DDR_CONTROL2, and MWMSG_DDR_DATA registers are oriented to DMA operations, but the MWMSG_DDR_CONTROL register can also be written by the processor (instead of using MCTRL and MxDATAB).

Offset

Register	Offset
MWMSG_DDR_CONTROL	D8h

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																
W	ADDRCMD															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Fields

Field	Description
31-16 —	Reserved
15-0 ADDRCMD	Address Command First Data write after control write with LEN!=0. This is formatted as: • 15:9 Target Address to read or write. • 8 Reserved, should be 0. • 7 1 if Read, 0 if Write. • 6:0 CMD as 7-bit value, always for controller.

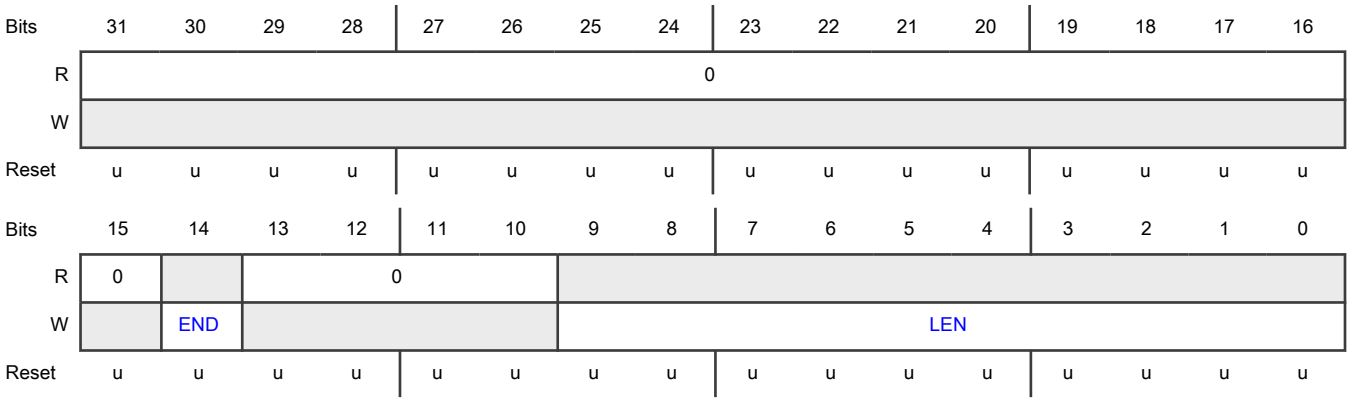
42.7.1.47 Controller Write Message in DDR mode Control 2 (MWMSG_DDR_CONTROL2)

Contains the 2nd Control word instructions with length of message and end.

Offset

Register	Offset
MWMSG_DDR_CONTROL2	D8h

Diagram



Fields

Field	Description
31-16 —	Reserved
15	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
14 END	End of message Indicates end of DDR message. MSTATUS[COMPLETE] = 1 when done. The end can happen before LEN bytes are read if the target ends sooner. 0 - Not the end. DDR message ends waiting for a new DDR message (will issue a HDR Restart for the new message). 1 - End. DDR message ends on HDR Exit.
13-10 —	Reserved
9-0 LEN	Length of Message Contains the length of the message (including the command) in halfwords, up to 2046 bytes. If LEN = 0, then END is applied only. • For reads, + 1 for the CRC. For example, to read 4 bytes (2 halfwords), use 1 + 2 + 1 for CMD + 2 halfwords + CRC. • For writes, LEN is the number of halves of data bytes + 1 (for command).

42.7.1.48 Controller Write Message Data in DDR mode (MWMSG_DDR_DATA)

Contains the 16-bit word to be written in Double Data Rate (DDR) mode. This register also functions as the MWMSG_DDR_CONTROL register .

Offset

Register	Offset
MWMSG_DDR_DATA	D8h

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																
W	DATA16B															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-17 —	Reserved
16 —	Reserved
15-0 DATA16B	Data

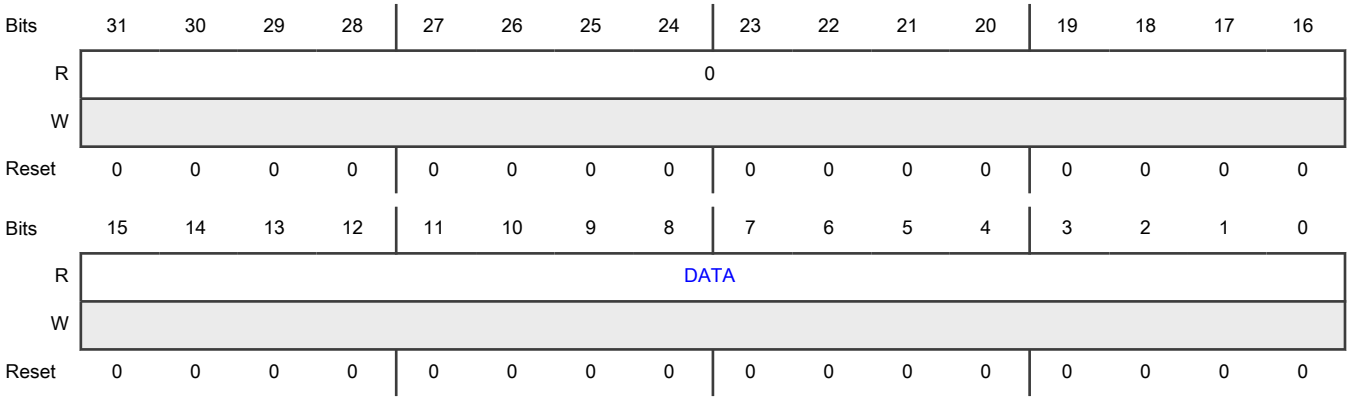
42.7.1.49 Controller Read Message in DDR mode (MRMSG_DDR)

Allows reading 16-bit words from a target in Double Data Rate (DDR) message mode from an active message started with MWMSG_DDR. These words are intended to be read by DMA.

Offset

Register	Offset
MRMSG_DDR	DCh

Diagram



Fields

Field	Description
31-16 —	Reserved
15-0 DATA	Data Contains the 16-bit word read from a target. The first byte is the LSB, and is in DATA[7:0]. The second byte is the MSB, and is in DATA[15:8].

Table continues on the next page...

Table continued from the previous page...

Field	Description
	• If the target ends before the entire length of the message (MWMSG_DDR[LEN]) is read, the module considers the DATA read as completed. In I3C mode, the target can indicate the end of message (the last byte). Otherwise, the controller terminates the message if the message is more than the controller will accept. • If the target has not yet finished sending DATA before the entire length of the message (MWMSG_DDR[LEN]) is read and END is not a continuation, the DATA read will be terminated.

42.7.1.50 Controller Dynamic Address (MDYNADDR)

Allows an I3C module to write its own Dynamic Address (DA) when the I3C module changes from Controller mode to Target mode.

If this device is the Main Controller (the controller during bus initialization), then this device may use this mechanism to assign itself its DA. When the device hands off control to a secondary controller, it becomes a target itself. This DA must be written before switching to Target mode and must not be changed once in Target mode (it is not clock-safe to do so). It must be written with a valid address value in DADDR if DAVALID = 1.

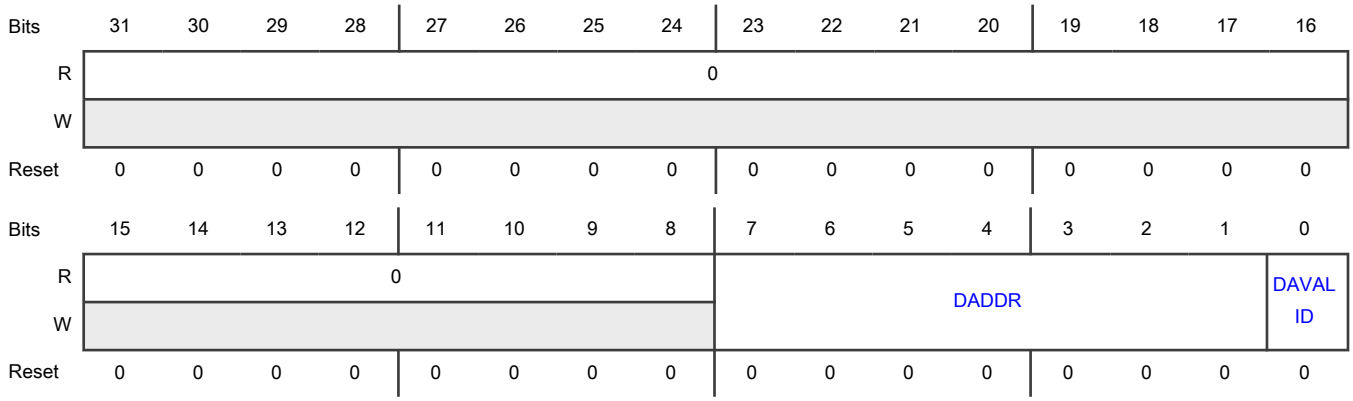
NOTE

The Main Controller also uses DEFSLVS CCC to define the target addresses, including itself; this mechanism is how secondary controllers know this address. If the controller is not the Main Controller, then this mechanism should not be used.

Offset

Register	Offset
MDYNADDR	E4h

Diagram



Fields

Field	Description
31-8	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
7-1 DADDR	Dynamic address Contains the assigned Dynamic Address when DAVALID = 1.
0 DAVALID	Dynamic address valid 0 - No valid DA assigned 1 - Valid DA assigned

42.7.1.51 Map Feature Control 0 (SMAPCTRL0)

The SMAPCTRLn registers are named SMAPCTRL0, SMAPCTRL1, and so on based on the number of mapped addresses. MAPCTRL0 represents the Primary DA or SA with SMAPCTRL1 onwards being the Mapped addresses.

The features of the SMAPCTRLn registers depend on configurations. SMAPCTRL0 acts differently, as shown below.

In general, this mechanism is intended to replace the DYNADDR register for all MAP related uses.

When using the Auto-MAP and DASA or AASA, the slot is changed from SA to DA. If the controller then issues RSTDAA, the application must rewrite the static addresses and enable them, as they will be marked disabled.

Offset

Register	Offset
SMAPCTRL0	11Ch

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				CAUSE				DA							ENA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-17	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
16 —	Reserved
15-11 —	Reserved
10-8 CAUSE	Cause Indicates the cause of the most recent DA assignment, which caused a SSTATUS[DACHG] interrupt. With MAP enabled. 000 - No information. This value occurs when not configured to write DA. 001 - Set using ENTDAA 010 - Set using SETDASA, SETAASA, or SETNEWDA 011 - Cleared using RSTDAA 100 - Auto MAP change happened last. The change may have changed this DA as well (for example, ENTDAA, and SETAASA), but at least one MAP entry automatically changed after. All other values are reserved.
7-1 DA	Dynamic Address Contains primary DA when ENA = 1. When ENA = 0, static address is used (but not shown here). With MAP enabled.
0 ENA	Enable Primary Dynamic Address With MAP enabled. 0 - Disable 1 - Enable

42.7.1.52 Extended IBI Data 1 (IBIEXT1)

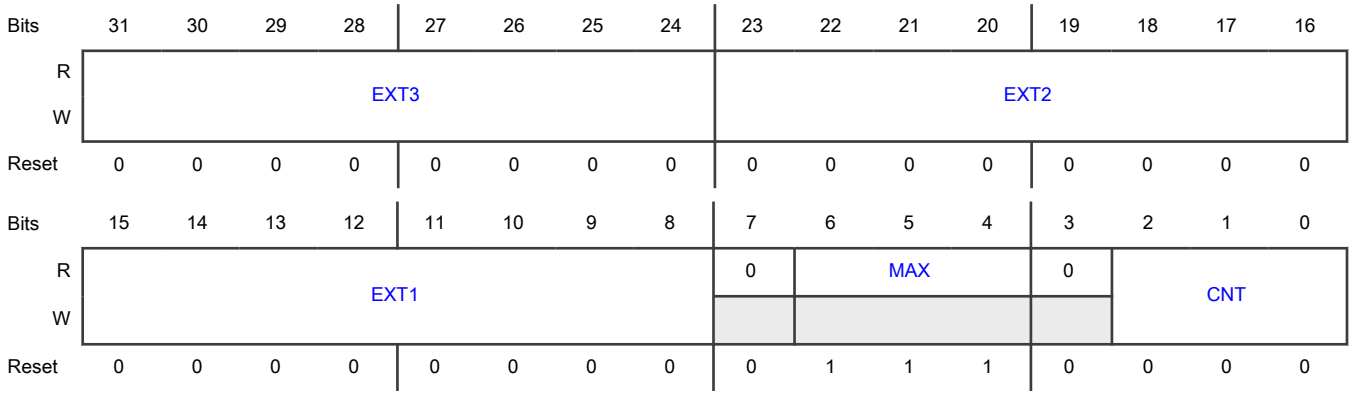
Contains extended IBI data.

The CTRL register is used to submit IBI, CR, and Hot-Join when enabled to do so. If allowed, an IBI may have additional bytes following the Mandatory Data byte (MDB). Extended IBI data is allowed when SCTRL[EXTDATA] = 1. If allowed, the extra bytes are indicated using these two registers.

Offset

Register	Offset
IBIEXT1	140h

Diagram



Fields

Field	Description
31-24 EXT3	Extra byte 3 Contains third extra byte
23-16 EXT2	Extra byte 2 Contains second extra byte
15-8 EXT1	Extra byte 1 Contains first extra byte
7 —	Reserved
6-4 MAX	Maximum Indicates the maximum number of extra bytes allowed by configuration. 0, if none.
3 —	Reserved
2-0 CNT	Count Contains the number of extra bytes beyond the MDB to be used. 0, if none.

42.7.1.53 Extended IBI Data 2 (IBIEXT2)

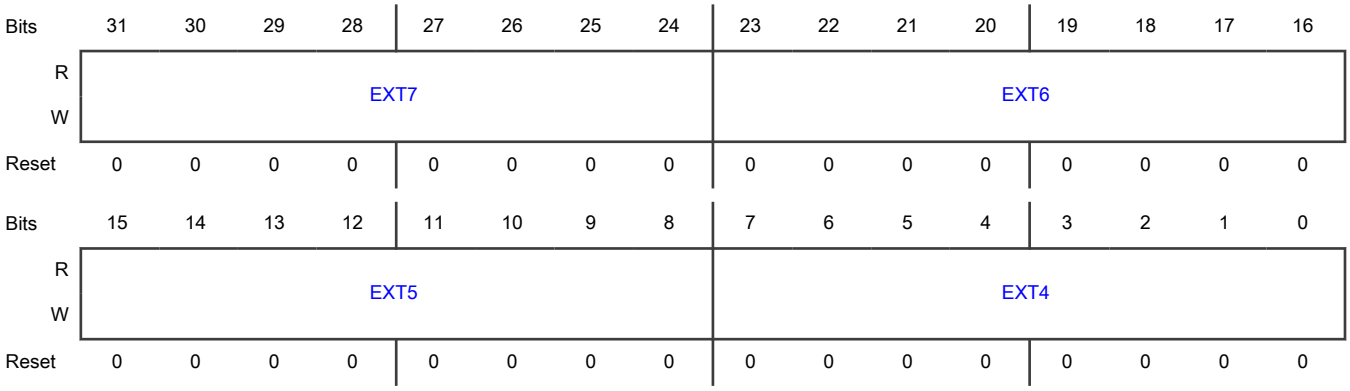
Contains extended IBI data.

The CTRL register is used to submit IBI, CR, and Hot-Join when enabled to do so. If allowed, an IBI may have additional bytes following the Mandatory Data byte (MDB). Extended IBI data is allowed when SCTRL[EXTDATA] = 1. If allowed, the extra bytes are indicated using these two registers.

Offset

Register	Offset
IBIEXT2	144h

Diagram



Fields

Field	Description
31-24 EXT7	Extra byte 7 Contains seventh extra byte
23-16 EXT6	Extra byte 6 Contains sixth extra byte
15-8 EXT5	Extra byte 5 Contains fifth extra byte
7-0 EXT4	Extra byte 4 Contains extra byte

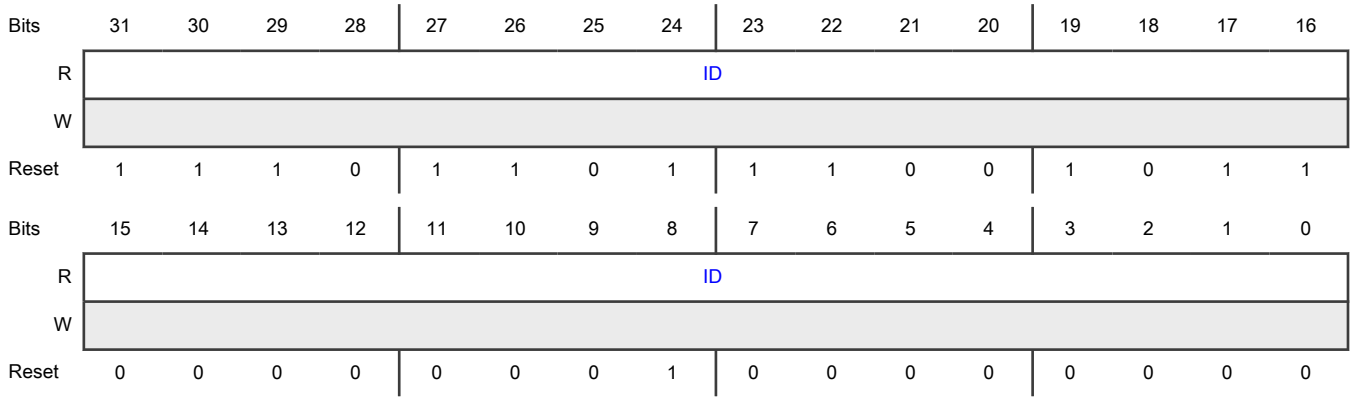
42.7.1.54 Target Module ID (SID)

The BlockID, if enabled, allows software to detect the module and its version information.

Offset

Register	Offset
SID	FFCh

Diagram



Fields

Field	Description
31-0 ID	ID The ID meaning is specific to each use of the I3C module. ID = 3989504256.

Chapter 43
Controller Area Network Flexible Data (MCAN)

43.1 Chip-specific MCAN information

Table 358. Reference links to related information

Topic	Related module	Reference
Full description	MCAN	MCAN
System memory map		System memory map
Clocking		Clock generation
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

43.1.1 Module instances

This device has one instance of the MCAN.

43.1.2 Initialization

The MCAN controller is configured using the following registers:

- Clock: Set the AHBCLKCTRL1[CAN] bit to enable the function clock to the respective MCAN block being used (see SYSCTL/SYSCON).

  NOTE

  The MCAN blocks are disabled on reset (AHBCLKCTRL1[CAN] = 0).

  The CAN function clock is the clock source used for the CAN. Use the CANCLKDIV register to divide the main clock to reach the required CAN clock frequency if needed.
- Pins: Select the MCAN pins and pin modes through the relevant IOCON registers (see IOCON).
- Interrupts: Enable in the NVIC using the appropriate interrupt set enable register.

43.2 Overview

Controller Area Network (CAN) is a high performance communication protocol that provides serial data communications. The MCAN controller is designed to provide a full implementation of the CAN protocol according to the CAN Specification Version 2.0 part A, B and to CAN FD Specification V.1.0.

The MCAN controller can be used to build powerful local networks with low-cost, multiplex wiring by supporting distributed real-time control with a very high level of security. The CAN controller consists of a CAN core, message RAM, control registers, AHB interface, and message handlers for transferring and receiving messages.

43.2.1 Features

- Conforms to Controller Area Network (CAN) protocol version 2.0 part A, B, and ISO 11898-1
- CAN FD with up to 64 data bytes supported
- CAN error logging
- AUTOSAR support

- SAE J1939 support
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signaling on reception of High Priority Messages
- Up to 64 dedicated Receive buffers
- Up to 32 dedicated Transmit buffers
- Configurable Transmit FIFO
- Configurable Transmit Queue
- Configurable Transmit Event FIFO
- Message RAM is assigned to on-chip SRAM, accessible by CPU and DMA
- Programmable loop-back test mode
- Maskable module interrupts
- Power-down support
- Debug on CAN support

43.2.2 Block Diagram

The core functionality provided by CAN is the CAN protocol controller and the transfer and receive shift register. The CAN core handles all ISO 11898-1 protocol functions and supports 11-bit and 29-bit identifiers. The Tx handler transmits messages from the message RAM to the CAN Core as well as providing transmit status information. The Rx handler manages acceptance filtering and transfers received messages from the CAN core to the message RAM as well as providing receive message status information. Acceptance filtering is implemented by a combination of up to 128 filter elements where each element can be configured as a range, bit mask, or dedicated ID filter.

Figure 187 shows the MCAN IP block diagram.

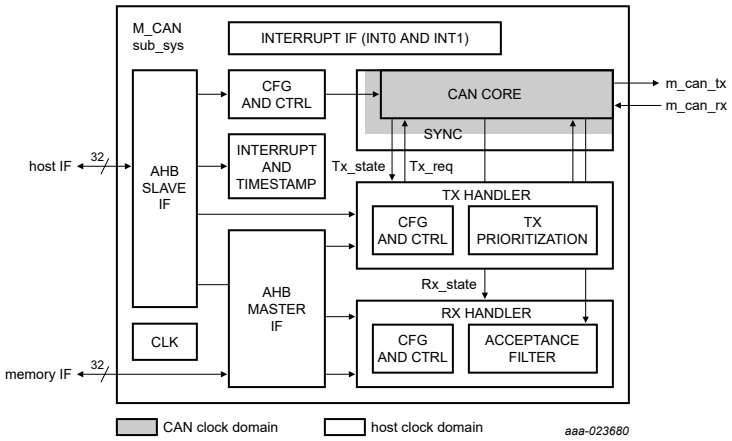


Figure 187. MCAN IP block diagram

43.3 Functional description

43.3.1 Operating modes

43.3.1.1 Normal operation

When the MCAN is initialized (see Initialization) and the INIT bit in the CCCR register is cleared, the MCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including message ID and DLC are stored into a dedicated Rx buffer or into Rx FIFO 0 or Rx FIFO 1.

To transmit messages, dedicated Tx buffers and/or a Tx FIFO or a Tx queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

43.3.1.2 CAN FD operation

There are two variants in the CAN FD frame transmission. The first is the CAN FD frame without bit rate switching. The second variant is the CAN FD frame where control field, data field, and CRC field are transmitted with a higher bit rate than the beginning and the end of the frame.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers will now be decoded as FDF bit. FDF = recessive signifies a CAN FD frame, FDF = dominant signifies a classic CAN frame. In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by res = dominant and BRS = recessive. The coding of res = recessive is reserved for future expansion of the protocol. In case the MCAN receives a frame with FDF = recessive and

res = recessive, it will signal a Protocol Exception Event by setting bit PSR.PXE. When Protocol Exception Handling is enabled in the CCCR register, this causes the operation state to change from Receiver at the next sample point, see the ACT bits in the PSR register. In case Protocol Exception Handling is disabled in the CCCR register, the MCAN will treat a recessive res bit as a form error and will respond with an error frame.

With FDOE bit cleared in the CCCR register, the setting of bits FDF and BRS is ignored and frames are transmitted in Classic CAN format. With FDOE bit set and the BRSE bit cleared in the CCCR register, only bit FDF of a Tx buffer element is evaluated. With FDOE and BRSE bit set in the CCCR register, transmission of CAN FD frames with bit rate switching is enabled. All Tx buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significantly higher than in the CAN FD arbitration phase. In this case, disable the CAN FD bit rate switching option for transmissions.
- During system startup, all nodes are transmitting classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN partial networking have to be transmitted in classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in silent mode until programming has completed. Afterwards, all nodes switch back to classic CAN communication.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in the standard CAN. Table 359 shows the DLC codes 9 to 15. In the standard CAN, codes 9 to 15 code a data field of 8 bytes.

Table 359. DLC coding in CAN FD

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD frames, the bit timing will be switched inside the frame, after the Bit Rate Switch (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 the NBTP register. In the following CAN FD data phase, the data phase bit timing is used as defined by the DBTP register. The bit timing is switched back from the data phase timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN clock frequency. For example, if the CAN clock frequency is 20 MHz and the shortest configurable bit time of 4 tq, the bit rate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD and CAN FD with bit rate switching, the value of the Error Status Indicator (ESI) bit is determined by the error state of the transmitter at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, otherwise it is transmitted dominant.

43.3.1.3 Restricted operation mode

In restricted Operation Mode, the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The REC and TEC error counters in the ECR register are frozen while the CEL bit is set. The MCAN can be put into restricted operation mode by setting the ASM bit in the CCCR register. The bit can only be set when CCE and INIT bits are set in the CCCR register. The bit can be cleared at any time.

Restricted operation mode is automatically entered when the Tx Handler was not able to read data from the message RAM in time. To leave restricted operation mode, the ASM bit in the CCCR register must be cleared.

The restricted operation mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the restricted operation mode after it has received a valid frame.

If the MCAN is connected to the clock calibration on the CAN unit, the ASM bit in the CCCR register is controlled by input m_can_cok. In case MCAN_COK switches to 0, the ASM bit is set. When the m_can_cok switches back to 1, the ASM bit returns to the previously written value. When there is no clock calibration on the CAN unit's connected input, m_can_cok is hardwired to 1.

NOTE

The restricted operation mode must not be combined with the loop back mode, whether internal or external.

43.3.1.4 Bus monitoring mode

The MCAN is set in bus monitoring mode by setting the MON bit in the CCCR registers. In bus monitoring mode (see ISO11898-1, 10.12 Bus monitoring), the MCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the MCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the MCAN monitors this dominant bit, although the CAN bus may remain in recessive state. In bus monitoring mode, the TXBRP register is held in reset state.

The bus monitoring mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. Figure 188 shows the connection of the m_can_tx and m_can_rx signals to the MCAN in bus monitoring mode.

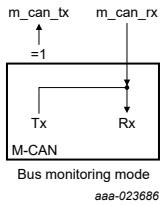


Figure 188. Pin control in bus monitoring mode

43.3.1.5 MCAN power down mode (sleep mode)

The MCAN can be set into power down mode controlled by the CSR bit in the CCCR register.

When all pending transmission requests have completed, the M_CAN waits until bus idle state is detected. Then the MCAN sets the INIT bit in the CCCR register to 1 to prevent any more CAN transfers. Now the MCAN acknowledges that it is ready for power down by setting the CSA bit to 1. In this state, before the clocks are switched off, further register accesses can be made. A write access to INIT bit will have no effect.

To leave the MCAN power down mode, the application has to turn on the module clocks before resetting the CSR bit in the CCCR register. The MCAN acknowledges this by resetting the CSA bit. Afterwards, the application can restart CAN communication by clearing the INIT bit.

43.3.1.6 Test modes

To enable write access to the test register, the TEST bit in the CCCR register must be set. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN transmit pin CAN1_TD by programming the TX bit field in the TEST register. Additionally to its default function – the serial data output – it can drive the CAN sample point signal to monitor the bit timing f the MCAN and it can drive constant dominant or recessive values. The actual value at pin CAN1_RD can be read from the RX bit field in the TEST register. Both functions can be used to check the physical layer of the CAN bus.

The synchronization mechanism in the CAN IP may cause a delay of several clock periods between writing to the TX bit field until the new configuration is visible at output pin m_can_tx. This applies also when reading input pin CAN1_RD via the RX bit field.

Note: Test modes should be used for production tests or self-tests only. The software control for the m_can_tx interferes with all CAN protocol functions. It is not recommended to use test modes for applications.

43.3.1.6.1 External loop back mode

The MCAN can be set in external loop back mode by setting the LBCK bit in the TEST register. In loop back mode, the MCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into an Rx buffer or an Rx FIFO. Figure 12 shows the connection of signals m_can_tx and CAN1_RD to the MCAN in external loop back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the MCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in loop back mode. In this mode the MCAN performs an internal feedback from its Tx output to its Rx input. The actual value of the CAN1_RD input pin is disregarded by the MCAN. The transmitted messages can be monitored at the m_can_tx pin.

43.3.1.6.2 Internal loop back mode

Internal loop back mode is entered by setting the LBCK bit in the TEST register and the MON bit in the CCCR register. This mode can be used for a "Hot Selftest", meaning, the MCAN can be tested without affecting a running CAN system connected to the pins m_can_tx and CAN1_RD. In this mode pin CAN1_RD is disconnected from the MCAN and pin m_can_tx is held recessive. Figure 189 shows the connection of m_can_tx and CAN1_RD to the MCAN in case of internal loop back mode.

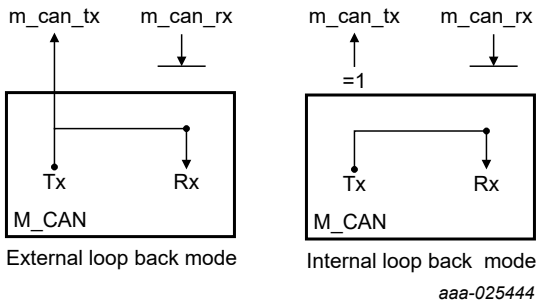


Figure 189. Pin control in loop back modes

43.3.2 Transmitter delay compensation

During the data phase of a CAN FD transmission, only one node transmits while all others are receivers. The length of the bus line has no impact. When transmitting with the CANx_TD pin, the M_CAN receives the transmitted data from its local CAN transceiver via the CANx_RD pin. The received data is delayed by the transmitter delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. In order to enable a data phase bit time that is even shorter than the transmitter delay, the delay compensation is introduced. Without transmitter delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transmitter delay.

43.3.2.1 Description

The protocol unit of the MCAN has implemented a delay compensation mechanism to compensate the transmitter delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the Secondary Sample Point SSP. If a bit error is detected, the transmitter will react on this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the new ISO11898-1. It is enabled by setting the TDC bit in the DBTP register.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the MCAN's transmit output through the transceiver to the receive input plus the transmitter delay compensation offset as configured by TDCO bit field in the TDCR register. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (for example, half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of MCAN clock periods.

The TDCV bit field in the PSR shows the actual transmitter delay compensation value. This bit field is cleared when INIT bit in the CCCR register is set and is updated at each transmission of an FD frame while TDC bit is set in the DBTP register.

The following boundary conditions have to be considered for the transmitter delay compensation implement in the MCAN:

- The sum of the measured delay from the m_can_tx to the CAN1_RD and the configured transmitter delay compensation offset bit field in the TDCR register has to be less than 6 bit times in the data phase.
- The sum of the measured delay from the m_can_tx to the CAN1_RD and the configured transmitter delay compensation offset bit field in the TDCR register has to be less or equal 127 MCAN clock periods. In case this sum exceeds 127 MCAN clock periods, the maximum value of 127 MCAN clock periods is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter that stops checking of receive bits at the SSPs.

43.3.2.2 Transmitter delay compensation measurement

If the transmitter delay compensation is enabled by setting the TDC bit in the DBTP register, the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the MCAN_RX of the transmitter. The resolution of this measurement is one MCAN clock periods.

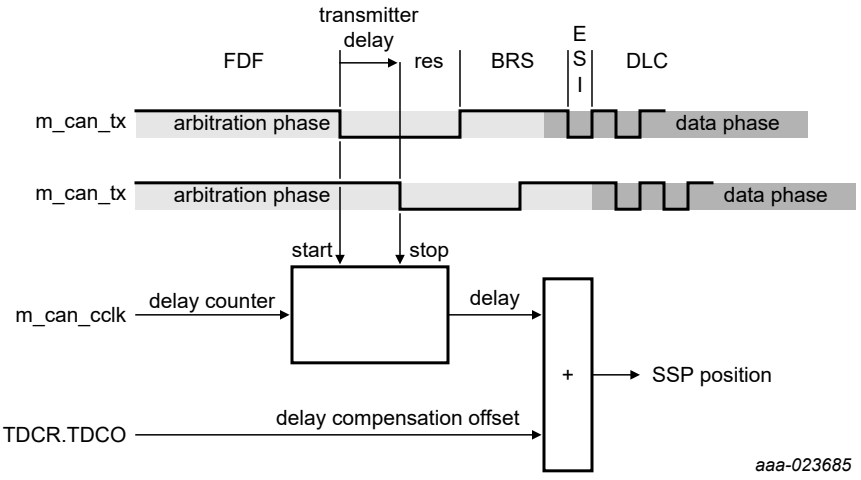


Figure 190. Transmitter delay measurement

To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit, resulting in a too early SSP position, the use of a transmitter delay compensation filter window can be enabled by programming the TDCF bit field in the TDCR register. This defines a minimum value for the SSP position. Dominant edges on CAN1_RD that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least equal to the TDCF bit field and m_can_rx is low.

43.3.3 Disabled automatic retransmission

According to the CAN Specification (see ISO11898-1, 6.3.3 Recovery Management), the MCAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled using the DAR bit field in the CCCR register.

43.3.3.1 Frame transmission in DAR mode

In DAR mode, all transmissions are automatically cancelled after they started on the CAN bus. A Tx buffer's Tx request pending bit in the TXBRP register is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  - Corresponding Tx buffer transmission occurred bit in the TXBTO register is set.
  - Corresponding Tx buffer cancellation finished bit in the TXBCF register is not set.
- Successful transmission in spite of cancellation:
  - Corresponding Tx buffer transmission occurred bit in the TXBTO register is set.
  - Corresponding Tx buffer cancellation finished bit in the TXBCF register is set.
- Arbitration lost or frame transmission disturbed:
  - Corresponding Tx buffer transmission occurred bit in the TXBTO register is not set.
  - Corresponding Tx buffer cancellation finished bit in the TXBCF register is set.

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx event FIFO element is written with event type = '10' (transmission in spite of cancellation).

43.3.4 Timestamp generation

For timestamp generation, the MCAN supplies a 16-bit wrap-around counter. The TCP bit field in the TSCC register contains a prescaler that can be configured to clock the counter in multiples of CAN bit times (1…16). The counter is readable via the TSC bit field in the TSCV register. A write access to the TSCV register resets the counter to zero.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of an Rx buffer / Rx FIFO (RXTS register) or Tx Event FIFO (TXTS register) element.

By programming the TSS bit in the TSCC register, the 16-bit external timestamp counter can be used.

43.3.5 Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO the MCAN supplies a 16-bit timeout counter. It operates as a down-counter and uses the same prescaler controlled by TCP bit field in the TSCC register. The timeout counter is configured via the TOCC register. The actual counter value can be read from TOC bit field in the TSCV. The timeout counter can only be started while the INIT bit in the CCCR register is cleared. It is stopped when the INIT bit is set. For example, when the M_CAN enters a buf_off state.

The operation mode is selected by the TOS bit field in the TOCC register. When operating in continuous mode, the counter starts when the INIT bit is cleared. A write to the TOCV register presets the counter to the value configured by TOP bit field in the TOCC register and continues down-counting.

When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP bit field in the TOCC register. Down-counting is started when the first FIFO element is stored. Writing to TOCV register has no effect.

When the counter reaches zero, the TOO interrupt flag in the IR register is set. In continuous mode, the counter is immediately restarted at the value in the TOP bit field in the TOCC register.

NOTE

The clock signal for the timeout counter is derived from the CAN core's sample point signal. Therefore, the point in time where the timeout counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN core. If the bit rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

43.3.6 Rx handling

The Rx handler controls the acceptance filtering, the transfer of received messages to the Rx buffers or to one of the two Rx FIFOs, and Rx FIFO's put and get indices.

43.3.6.1 Acceptance filtering

The MCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to an Rx buffer or to Rx FIFO 0,1. For acceptance filtering, each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as a range filter, filter for one or two dedicated IDs, or classic bit mask filter.
- Each filter element is configurable for acceptance or rejection filtering.
- Each filter element can be enabled or disabled individually.
- Filters are checked sequentially, execution stops with the first matching filter element.

The related configuration registers are:

- Global filter configuration (GFC).
- Standard ID filter configuration (SIDFC).
- Extended ID filter configuration (XIDFC.
- Extended ID AND mask (XIDAM).

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1.
- Store received frame in Rx buffer.
- Store received frame in Rx buffer and generate pulse at filter event pin.
- Reject received frame.
- Set high priority message interrupt flag (HPM bit in the IR register).
- Set high priority message interrupt flag and store the received frame in FOF 0 or FIFO 1.

Acceptance filtering is started after the complete identifier is received. After acceptance filtering has completed, and if a matching Rx buffer or Rx FIFO has been found, the message handler starts writing the received message data in 32 bit chunks to the matching Rx buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (e.g. CRC error), this message is discarded with the following impact on the affected Rx buffer or Rx FIFO:

- Rx buffer
  - New data flag of matching Rx buffer is not set, but the Rx buffer is partially overwritten with received data. For error types, see the LEC and DLEC bit field in the PSR register.
- Rx FIFO
  - Put index of matching Rx FIFO is not updated, but the related Rx FIFO element is partially overwritten with received data. For error types, see the LEC and DLEC bit field in the PSR register. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in Section X have to be considered.

NOTE

When an accepted message is written to one of the two Rx FIFOs, or into an Rx buffer, the unmodified received identifier is stored independent of the filters used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

43.3.6.1.1 Range filter

The filter matches for all received frames with message IDs in the range defined by the SFID1/SFID2 resp. EFID1/EFID2.

There are two possibilities when range filtering is used together with extended frames:

EFT = "00": the message ID of received frames is ANDed with the extended ID AND mask register before the range filter is applied.

EFT = "11": the extended ID AND mask register is not used for range filtering.

43.3.6.1.2 Filter for specific IDs

A filter element can be configured to filter for one or two specific message IDs. To filter for one specific message ID, the filter element has to be configured with SFID1= SFID2 resp. EFID1= EFID2.

43.3.6.1.3 Classic bit mask filter

Classic bit mask filtering is intended to filter groups of message IDs by masking single bits of a received message ID. With classic bit mask filtering, SFID1/EFID1 is used as message ID filter, while SFID2/EFID2 is used as filter mask.

A 0 bit at the filter mask will mask out the corresponding bit position of the configured ID filter. For example, the value of the received message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received message ID where the corresponding mask bits are 1 are relevant for acceptance filtering.

In case all mask bits are one, a match occurs only when the received message ID and the message ID filter are identical. If all mask bits are zero, all message IDs match.

43.3.6.1.4 Standard message ID filtering

Figure 191 shows the flow of standard message ID filtering (11-bit identifier).

Controlled by the GFC and SIDFC registers, the message ID, remote transmission request bit (RTR), and the identifier extension bit (IDE) of received frames are compared against the list of configured filter elements.

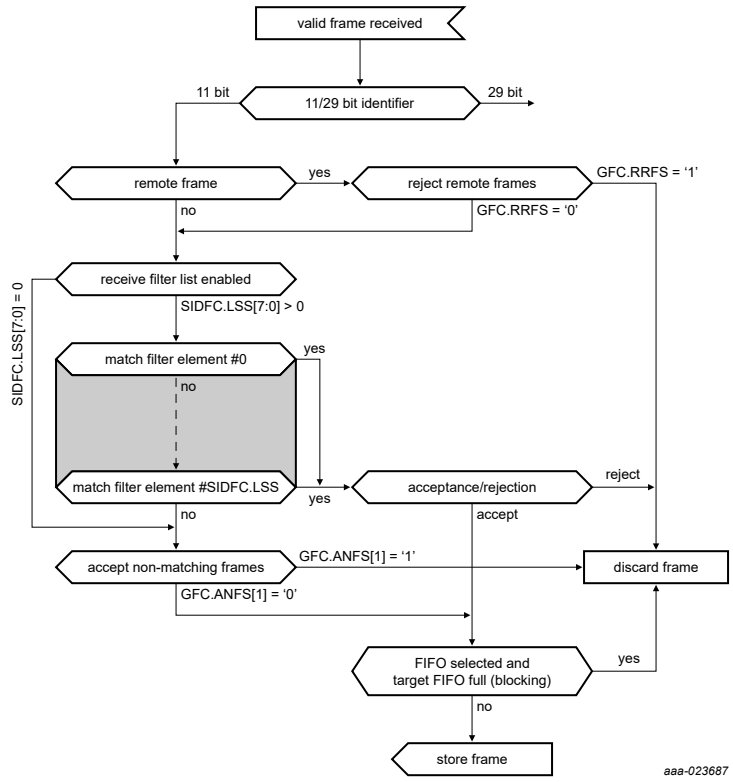


Figure 191. Standard message ID filter path

43.3.6.1.5 Extended message ID filtering

Figure 192 shows the flow of standard message ID filtering (11-bit identifier). The standard message ID filter element is described in Standard message ID filtering.

Controlled by the GFC and SIDFC registers, the message ID, remote transmission request bit (RTR), and the identifier extension bit (IDE) of received frames are compared against the list of configured filter elements.

The XIDAM register is ANDed with the received identifier before the filter list is execute.

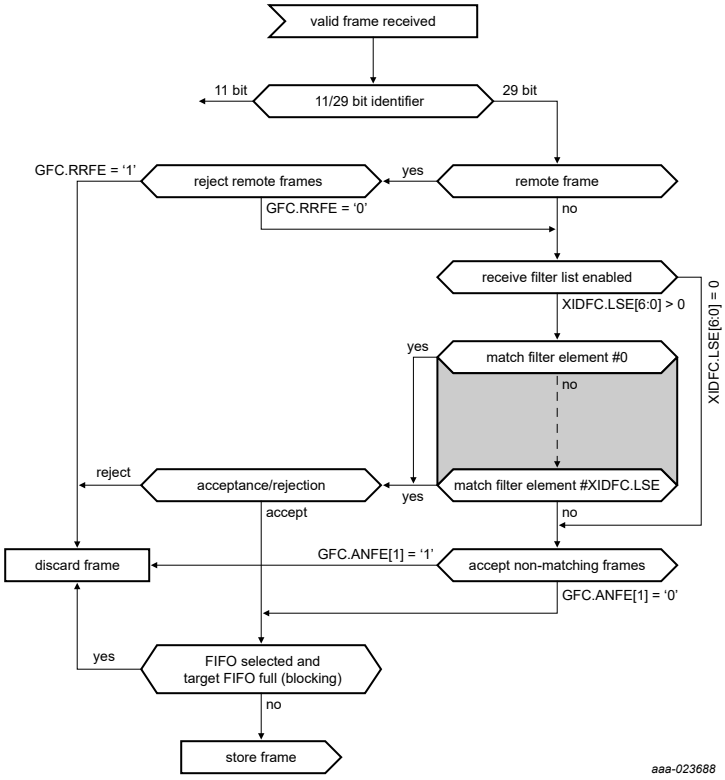


Figure 192. Extended message ID filter path

43.3.6.2 Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via the RXF0C and RXF1C registers.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1 see Acceptance filtering. The Rx FIFO element is described in Rx buffer and FIFO element.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by Rx FIFO watermark bit fields in the RX FIFO configuration registers, the Rx FIFO watermark interrupt flag in the IR register is set. When the Rx FIFO put index reaches the Rx FIFO get index an Rx FIFO full condition is signaled by Rx FIFO full bit fields in the Rx FIFO status registers. In addition, the Rx FIFO full interrupt flag in the IR register is set.

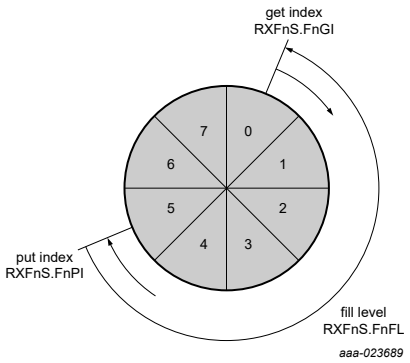


Figure 193. Rx FIFO status

When reading from an Rx FIFO, the Rx FIFO get index bit field in the Rx FIFO status register * FIFO element size has to be added to the corresponding Rx FIFO start address in the Rx FIFO configuration register.

Table 360. Rx buffer / FIFO element size

RXESC.RBDS[2:0] RXESC.FnDS[2:0]	Data Field (bytes)	FIFO element size (RAM words)
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

43.3.6.2.1 Rx FIFO blocking mode

The Rx FIFO blocking mode is configured in the RX FIFO configuration registers. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (Rx FIFO put index = the Rx FIFO get index), no more messages are written in the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO get index has been incremented. An Rx FIFO full condition is signaled when the Rx FIFO full bit fields are set. In addition the Rx FIFO full interrupt flag in the IR register is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signaled in the Rx FIFO status register. In addition, the Rx FIFO lost interrupt flag in the IR register is set.

43.3.6.2.2 Rx FIFO overwrite mode

The Rx FIFO overwrite mode is configured in the Rx FIFO configuration register.

When an Rx FIFO full condition (Rx FIFO put index = the Rx FIFO get index) is signaled by the Rx FIFO full bit fields in Rx FIFO status register, the next message accepted by the FIFO will overwrite the oldest FIFO message. The put and get indices are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signaled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for this is that a received message is written to the message RAM (put index) while the MCU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the MCU accesses the Rx FIFO. Figure 194 shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

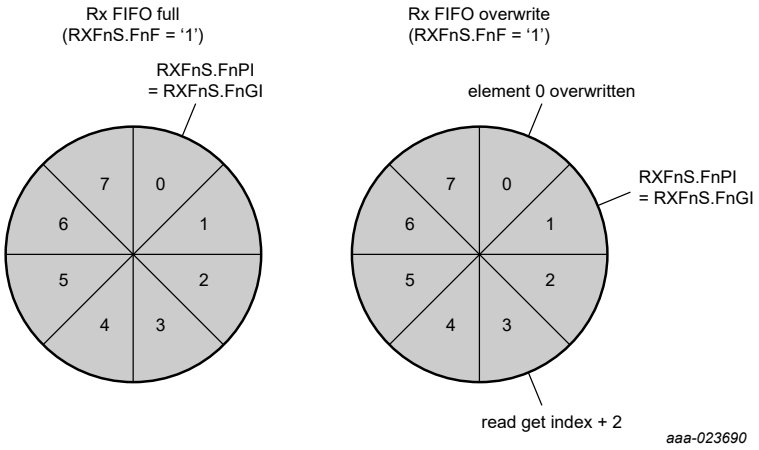


Figure 194. Rx FIFO overflow handling

43.3.6.2.3 Dedicated Rx buffers

The MCAN supports up to 64 dedicated Rx buffers. The start address of the dedicated Rx buffer section is configured via the RBSA bit in the RXBC register.

For each Rx buffer a Standard or extended message ID filter element with SFEC / EFEC = "111" and SFID2 / EFID2[10:9] = "00" has to be configured.

After a received message is accepted by a filter element, the message is stored into the Rx buffer in the message RAM that is referenced by the filter element. The format is the same as an Rx FIFO element. In addition, the DRX interrupt flag is set in the IR register.

Table 361. Example filter configuration for Rx buffers

Filter element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID message 1	00	00 0000
1	ID message 2	00	00 0001
2	ID message 3	00	00 0010

After the last word of a matching received message has been written to the message RAM, the respective new data flags are set in the NDAT registers. As long as the new data flag is set, the respective Rx buffer is locked against updates from received matching frames. The new data flags have to be reset by writing a 1 to the respective bit position.

While an Rx buffer's new data flag is set, a message ID filter element referencing this specific Rx buffer will not match, causing the acceptance filtering to continue. Subsequent message ID filter elements may cause the received message to be stored into another Rx buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

43.3.6.2.4 Rx buffer handling

- Reset the DRX interrupt flag in the IR register.
- Read data from the NDAT registers.
- Read messages from the message RAM.
- Reset the NDAT flags of processed messages.

43.3.7 Tx handling

The Tx handler handles transmission requests for the dedicated Tx buffers, the Tx FIFO, and the Tx queue. It controls the transfer of transmit messages to the CAN core, the put and get Indices, and the Tx event FIFO. Up to 32 Tx buffers can be set up for message transmission. The CAN mode for transmission (classic CAN or CAN FD) can be configured separately for each Tx buffer element. The Tx buffer element is described in Tx buffer element. The table shown below describes the possible configurations for frame transmission.

Table 362. Possible configurations for frame transmission

CCCR		Tx buffer element		Frame transmission
BRSE	FDOE	FDF	BRS	
ignored	0	ignored	ignored	Classic CAN
0	1	0	ignored	Classic CAN
0	1	1	ignored	FD without bit rate switching
1	1	0	ignored	Classic CAN
1	1	1	0	FD without bit rate switching
1	1	1	1	FD without bit rate switching

Note: AUTOSAR requires at least three Tx queue buffers and support for transmit cancellation.

The Tx handler starts a Tx scan to check for the highest priority pending Tx request (Tx buffer with lowest message ID) when the TXBRP register is updated, or when a transmission has been started.

43.3.7.1 Transmit pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU's CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

For example, if CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The transmit pause feature is controlled by the TXP bit in the CCCR register. If the bit is set, the MCAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. By default, transmit pause is disabled in the CCCR register.

This feature breaks up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

43.3.7.2 Dedicated Tx buffers

Dedicated Tx buffers are intended for message transmissions under complete control of the Host CPU. Each dedicated Tx buffer is configured with a specific message ID. In case multiple Tx buffers are configured with the same message ID, the Tx buffer with the lowest buffer number is transmitted first.

If the data section id updated, a transmission is requested by the add request bit fields in the TXBAR register. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx queue and externally with messages on the CAN bus, and are sent out according to their message ID.

A dedicated Tx buffer allocates element size 32-bit words in the message RAM. Therefore the start address of a dedicated Tx buffer in the message RAM is calculated by adding transmit buffer index * element size to the Tx buffer start address stored in the TBSA bit field in the TXBC register.

Table 363. Tx buffer / FIFO / queue element size

TXESC.TBDS[2:0]	Data Field (bytes)	Element size (RAM words)
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

43.3.7.3 Tx FIFO

Tx queue operation is configured by setting the TFQM bit in the TXBC register. Messages stored in the Tx queue are transmitted starting with the message with the lowest message ID (highest priority). In case multiple queue buffers are configured with the same message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages must be written to the Tx buffer referenced by the Tx FIFO queue put index bit field in the TXFQS register. An add request cyclically increments the put index to the next free Tx buffer. In case the Tx queue is full, the put index is not valid and no more message should be written to the Tx queue until at least one of the requested messages is sent out or a pending transmission request is cancelled.

The application may use the TXBRP register instead of the put index and may place messages to any Tx buffer without pending transmission request.

A dedicated Tx buffer allocates element size 32-bit words in the message RAM. Therefore, the start address of a dedicated Tx buffer in the message RAM is calculated by adding transmit buffer index * element size to the Tx buffer start address stored in the TBSA bit field in the TXBC register.

43.3.7.4 Tx queue

Tx queue operation is configured by programming the TFQM bit field to a value of one. Messages stored in the Tx queue are transmitted starting with the message with the lowest message ID (highest priority). In case that multiple queue buffers are configured with the same message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages must be written to the Tx buffer referenced by the put index stored in the TFQPI bit field in the TXFQS register. An add request cyclically increments the put index to the next free Tx buffer. In case the Tx queue is full (if the TFQF bit field is set in the TXFQS register), the put index is not valid and no more messages should be written to the Tx queue until at least one of the requested messages is sent out or a pending transmission request has been cancelled.

The application may use the TXBRP register instead of the put index and may place messages to any Tx buffer without pending transmission request.

A Tx queue buffer allocated element size 32-bit words in the message RAM. Therefore, the start address of the next available (free) Tx queue buffer is calculated by adding Tx FIFO/queue put index sorted in the TFQPI bit field in the TXFQS register * element size to the Tx buffer start address stored in the TBSA bit field in the TXBC register.

43.3.7.5 Mixed dedicated Tx buffers / Tx FIFO

The Tx buffers section in the message RAM is subdivided into a set of dedicated Tx buffers and a Tx FIFO. The number of dedicated Tx buffers is configured by the NDTB bit field in the TXBC register. The number of Tx buffers assigned to the Tx FIFO is configured by the TFQS bit field in the TXBC register. In case the TFQS bit field is programmed to zero, only dedicated buffers are used.

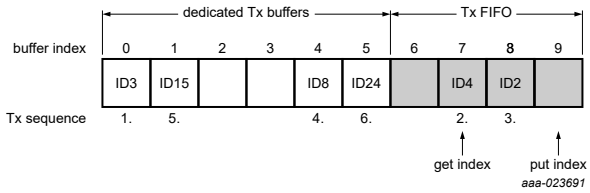


Figure 195. Example of mixed configuration dedicated Tx buffers / Tx FIFO

Tx prioritization:

- Scan dedicated Tx buffers and oldest pending Tx FIFO buffer (referenced by the TFGI bit field in the TXFS register).
- Buffer with lowest message ID gets highest priority and is transmitted next.

43.3.7.6 Mixed dedicated Tx buffers / Tx queue

The Tx buffers section in the message RAM is subdivided into a set of dedicated Tx buffers and a Tx queue. The number of dedicated Tx buffers is configured by the NDTB bit field in the TXBC register. The number of Tx queue buffers is configured by the TFQS bit field in the TXBC register. In case the TFQS bit field is programmed with a value of zero, only dedicated Tx buffers are used.

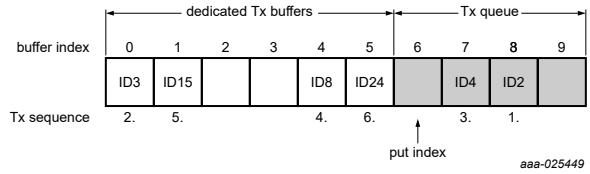


Figure 196. Example of mixed configuration dedicated Tx buffers / Tx queue

Tx prioritization:

- Scan all Tx buffers with activated transmission requests.
- Tx buffer with lowest message ID gets highest priority and is transmitted next.

43.3.7.7 Transmit cancellation

The MCAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a dedicated Tx buffer or a Tx queue buffer, set the corresponding bit in the TXBCR register. Transmit cancellation is not intended for Tx FIFO operation. Successful cancellation is signaled by setting the corresponding bit in the TXBCF register.

In case a transmit cancellation is requested while a transmission from a Tx buffer is already ongoing, the corresponding bit in the TXBRP register remains set as long as the transmission is in progress. If the transmission was successful, the corresponding bits in the XBTO and TXBCF register are set. If the transmission was not successful, it is not repeated and only the corresponding bit in the TXBCF register is set.

NOTE

In case a pending transmission is cancelled immediately before this transmission could have started, there follows a short time window where no transmission has started even if another message is also pending in this node. This may enable another node to transmit a message, which may have a lower priority than the second message in this node.

43.3.7.8 Tx event handling

To support Tx event handling the MCAN has implemented a Tx event FIFO. After the MCAN has transmitted a message on the CAN bus, message ID and timestamp are stored in a Tx event FIFO element. To link a Tx event to a Tx event FIFO element, the message marker from the transmitted Tx buffer is copied into the Tx event FIFO element.

The Tx event FIFO can be configured to a maximum of 32 elements. The Tx event FIFO element is described in Tx event FIFO element.

The purpose of the Tx event FIFO is to decouple handling transmit status information from transmit message handling, that is, a Tx buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx buffer before overwriting that Tx buffer.

When a Tx event FIFO full condition is signaled by the TEFF interrupt flag in the IR register, no more elements are written to the Tx event FIFO until at least one element has been read out and the Tx event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx event FIFO is full, this event is discarded and the TEFL interrupt flag is set in the IR register.

To avoid a Tx event FIFO overflow, the Tx event FIFO watermark can be used. When the Tx event FIFO fill level reaches the Tx event FIFO watermark configured by the EFWM bit field in the TXEFC register, the TEFW interrupt flag is set in the IR register.

When reading from the Tx event FIFO, two times the Tx event FIFO get index value stored in the EFGI bit field in the TXEFS register has to be added to the Tx event FIFO start address stored in the EFSA bit field in the TXEFC register.

43.3.8 FIFO acknowledge handling

The get indices of Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO are controlled by writing to the corresponding FIFO acknowledge. Writing to the FIFO acknowledge index will set the FIFO get index to the FIFO acknowledge index plus one and thereby updates the FIFO fill level. There are two use cases:

- When only a single element is read from the FIFO (the one being pointed to by the get index), this get index value is written to the FIFO Acknowledge Index.
- When a sequence of elements are read from the FIFO, it is sufficient to write the FIFO acknowledge index only once at the end of that read sequence (value: Index of the last element read), to update the get index of the FIFO.

Take special care when reading FIFO elements in an arbitrary order (that is, not using get index). This might be useful when reading a high priority message from one of the two Rx FIFOs. In this case the acknowledge index of the FIFO should not be written because this would set the get index to a wrong position and also alter the fill level of the FIFO. In this case some of the older FIFO elements would be lost.

NOTE

The application has to ensure that a valid value is written to the FIFO acknowledge index. The MCAN does not check for erroneous values.

43.4 Signals

Table 364. CAN signals

Signal	Type	Description
TD	O	MCAN transmit output
RD	I	MCAN receive input

43.5 Initialization

Software initialization is started by setting the INIT bit in the CCCR register, either by software or by a hardware reset, when an uncorrected bit error was detected in the message RAM, or through a Bus_Off. While the INIT bit is set, message transfer from and to the CAN bus is stopped while the status of the CAN bus output m_can_tx is recessive. The counters of the error management logic are unchanged. Setting the INIT bit does not change any configuration register. Resetting the INIT bit finishes the software initialization. Afterwards, the bit stream processor synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits, indicating that the bus is idle, before it can take part in bus activities and start the message transfer.

Access to the MCAN configuration registers is only enabled when both the INIT and CCE bits are set in the CCCR register.

The CCE bit can only be set or reset while the INIT bit is set. The CCE bit is automatically reset when the INIT bit is reset.

The following registers are reset when the CCE bit is set:

- HPMS – High priority message status.
- RXF0S – Rx FIFO 0 status.
- RXF1S – Rx FIFO 1 status.
- TXFQS – Tx FIFO/queue status.
- TXBTO – Tx buffer transmission occurred.
- TXBRP – Tx buffer request pending.
- TXBCF – Tx buffer cancellation finished.
- TXEFS – Tx event FIFO status.

The timeout counter value in the TOCV register is preset to the value configured by the TOP bits in the TOCC register, while the CCE bit is set in the CCCR register. In addition, the state machines of the Tx and Rx handlers are held in an idle state while the CCE bit is set.

The following registers are only writable while the CCE bit is cleared:

- TXBAR – Tx buffer add request.
- TXBCR – Tx buffer cancellation request.

The TEST and MON bits in the CCCR register can only be set while the INIT and CCE bits are set. Both bits may be reset at any time. The DAR bit can only be set or reset while the INIT and CCE bit are set.

43.6 Memory Map and register definition

This section includes the MCAN module memory map and detailed descriptions of all registers as well as the data structures needed to support messaging.

43.6.1 MCAN register descriptions

43.6.1.1 MCAN memory map

CAN base address: 4009_D000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
C	Data Bit Timing and Prescaler (DBTP)	32	See section	0000_0A33
10	Test (TEST)	32	See section	See section
18	CC Control (CCCR)	32	See section	0000_0001
1C	Nominal Bit Timing and Prescaler (NBTP)	32	See section	0600_0A03
20	Timestamp Counter Configuration (TSCC)	32	See section	0000_0000
24	Timestamp Counter Value (TSCV)	32	RO	0000_0000
28	Timeout Counter Configuration (TOCC)	32	See section	FFFF_0000
2C	Timeout Counter Value (TOCV)	32	RO	0000_FFFF
40	Error Counter (ECR)	32	RO	0000_0000
44	Protocol Status (PSR)	32	See section	0000_0707
48	Transmitter Delay Compensator (TDCR)	32	See section	0000_0000
50	Interrupt (IR)	32	See section	0000_0000
54	Interrupt Enable (IE)	32	See section	0000_0000
58	Interrupt Line Select (ILS)	32	See section	0000_0000
5C	Interrupt Line Enable (ILE)	32	See section	0000_0000
80	Global Filter Configuration (GFC)	32	See section	0000_0000
84	Standard ID Filter Configuration (SIDFC)	32	See section	0000_0000
88	Extended ID Filter Configuration (XIDFC)	32	See section	0000_0000
90	Extended ID AND Mask (XIDAM)	32	See section	1FFF_FFFF

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
94	High Priority Message Status (HPMS)	32	RO	0000_0000
98	New Data 1 (NDAT1)	32	RW	0000_0000
9C	New Data 2 (NDAT2)	32	RW	0000_0000
A0	Rx FIFO 0 Configuration (RXF0C)	32	See section	0000_0000
A4	Rx FIFO 0 Status (RXF0S)	32	RO	0000_0000
A8	Rx FIFO 0 Acknowledge (RXF0A)	32	See section	0000_0000
AC	Rx Buffer Configuration (RXBC)	32	See section	0000_0000
B0	Rx FIFO 1 Configuration (RXF1C)	32	See section	0000_0000
B4	Rx FIFO 1 Status (RXF1S)	32	RO	0000_0000
B8	Rx FIFO 1 Acknowledge (RXF1A)	32	See section	0000_0000
BC	Rx Buffer and FIFO Element Size Configuration (RXESC)	32	See section	0000_0000
C0	Tx Buffer Configuration (TXBC)	32	See section	0000_0000
C4	Tx FIFO/Queue Status (TXFQS)	32	See section	0000_0000
C8	Tx Buffer Element Size Configuration (TXESC)	32	See section	0000_0000
CC	Tx Buffer Request Pending (TXBRP)	32	RO	0000_0000
D0	Tx Buffer Add Request (TXBAR)	32	RW	0000_0000
D4	Tx Buffer Cancellation Request (TXBCR)	32	RW	0000_0000
D8	Tx Buffer Transmission Occurred (TXBTO)	32	RO	0000_0000
DC	Tx Buffer Cancellation Finished (TXBCF)	32	RO	0000_0000
E0	Tx Buffer Transmission Interrupt Enable (TXBTIE)	32	RW	0000_0000
E4	Tx Buffer Cancellation Finished Interrupt Enable (TXBCIE)	32	RW	0000_0000
F0	Tx Event FIFO Configuration (TXEFC)	32	See section	0000_0000
F4	Tx Event FIFO Status (TXEFS)	32	See section	0000_0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
F8	Tx Event FIFO Acknowledge (TXEFA)	32	See section	0000_0000
200	Message RAM Base Address (MRBA)	32	See section	0000_0000
400	External Timestamp Counter Configuration (ETSCC)	32	See section	0000_0000
600	External Timestamp Counter Value (ETSCV)	32	See section	0000_0000

43.6.1.1.1 Data Bit Timing and Prescaler (DBTP)

Enable write access to the Data Bit Timing and Prescaler (DBTP) register by setting CCCR[CCE] and CCCR[INIT] = 1.

The nominal bit time for CAN is determined by the number of time quanta per bit, and the time each time quantum represents. The time quantum (t_q) may be programmed in the range of 1 to 32 MCAN clock periods:

```
T_q = (DBRP + 1) MCAN clock periods
```

A single CAN bit time is made up from 4 segments:

- Segment SYNC_SEG is not programmable and is fixed at one time quantum.
- Segments PROP_SEG and PHASE_SEG1 are combined in a single bit field, and are programmable in the range of 1 to 16 time quanta. The sample point is set right after PHASE_SEG1.
- Segment PHASE_SEG2 is programmable in the range of 1 to 16 time quanta.

The length of the bit time (in time quanta) is the combined value of these 4 segments:

```
[SYNC_SEG + PROP_SEG + PHASE_SEG1 + PHASE_SEG2] t_q = [1 + PROP_SEG
+ PHASE_SEG1 + PHASE_SEG2] t_q
```

Therefore, the CAN bit time may be programmed in the range of 3 to 33 time quanta.

The combined value of segments PROP_SEG and PHASE_SEG1 can be programmed using bit field DTSEG1, and segment PHASE_SEG2 can be programmed using bit field DTSEG2.

The CAN hardware interprets these bit fields as the programmed value+1, so the length of the bit time (in time quanta) can be calculated from the programmed values as:

```
[1 + DTSEG1 + 1 + DTSEG2 + 1] t_q = [DTSEG1 + DTSEG2 + 3] t_q
```

The CAN bit time may be programmed in the range of 4 to 49 time quanta. The CAN time quantum may be programmed in the range of 1 to 32 MCAN clock periods.

```
T_q = (DBRP + 1) mtq.
```

DTSEG1 is the sum of prop_seg and phase_Seg1. DTSEG2 is phase_seg2. Therefore, the length of the bit time is:

```
[DTSEG1 + DTSEG2 + 3] t_q for programmed values, or,
[sync_seg + prop_seg + phase_seg1 + phase_seg2] t_q for functional values.
```

The information processing time (IPT) is zero, meaning that the data for the next bit is available the first clock edge after the sample point.

NOTE

With a MCAN clock of 8 MHz, the reset value of the Data Bit Timing and Prescaler (DBTP) register configures the MCAN for a data phase bit rate of 500 kb/s.

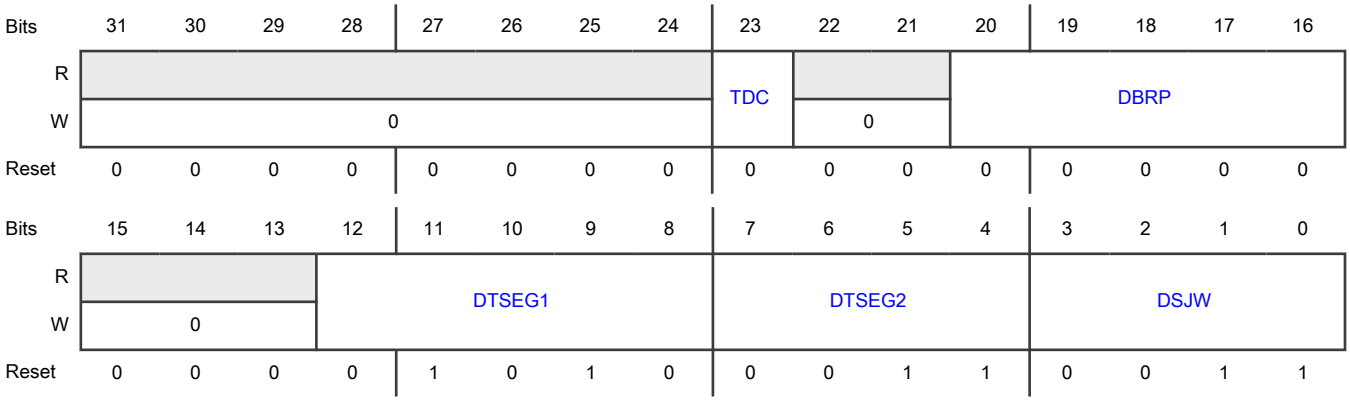
NOTE

The bit rate configured for the CAN FD data phase through the DBTP register must be higher or equal to the bit rate configured for the arbitration phase through the NBTP register.

Offset

Register	Offset
DBTP	Ch

Diagram



Fields

Field	Description
31-24 —	Reserved Read value is undefined, only zero should be written.
23 TDC	Transmitter Delay Compensation 0 - Transmitter delay compensation disabled 1 - Transmitter delay compensation enabled
22-21 —	Reserved Read value is undefined, only zero should be written.
20-16 DBRP	Data Bit Rate Prescaler The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. Valid values are 0 to 31.
15-13 —	Reserved Read value is undefined, only zero should be written.
12-8	Data Time Segment Before Sample Point

Table continues on the next page...

Table continued from the previous page...

Field	Description
DTSEG1	The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. Valid values are 0 to 15.
7-4 DTSEG2	Data Time Segment After Sample Point The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. Valid values are 0 to 15.
3-0 DSJW	Data (Re)Synchronization Jump Width The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. Valid values are 0 to 15. Limits by how many time quanta a bit period may be extended or shortened because of re-synchronization.

43.6.1.1.2 Test (TEST)

Setting CCCR[TEST] = 1 enables write access to the TEST register. All TEST register functions are set to their reset values when the CCCR[TEST] field is reset.

The different test functions can be combined, but when TX bits are programmed with a value that is not 0x0, the message transfer is disturbed.

Offset

Register	Offset
TEST	10h

Diagram

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

Fields

Field	Description
31-8 —	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
7 RX	Monitors the Actual Value of the CAN_RXD 0 - The CAN bus is dominant (CAN_RXD = 0). 1 - The CAN bus is recessive (CAN_RXD = 1).
6-5 TX	Control of Transmit Pin 00 - Loop back mode is disabled. 01 - The sample point can be monitored at the CAN_TXD. 10 - CAN_TXD pin is driven LOW/dominant. 11 - CAN_TXD is driven HIGH/recessive.
4 LBCK	Loop Back Mode 0 - Loop back mode is disabled. 1 - Loop back mode is enabled.
3-0 —	Reserved Read value is undefined, only zero should be written.

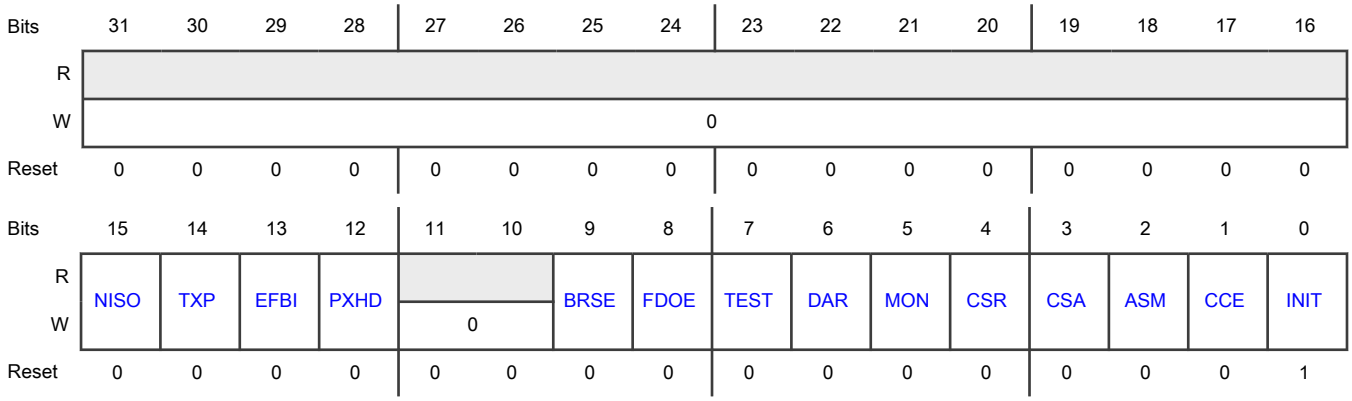
43.6.1.1.3 CC Control (CCCR)

The MCAN module has a mechanism to synchronize the two clock domains within itself, which may cause a delay before the value written to the INIT field can be read back. Because of the potential delay, it is recommended to read back the value of the INIT bit and confirm that it has been accepted before writing a new value to the INIT bit.

Offset

Register	Offset
CCCR	18h

Diagram



Fields

Field	Description
31-16 —	Reserved Read value is undefined, only zero should be written.
15 NISO	Non ISO Operation If this bit is set, the MCAN module uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0. 0 - CAN FD frame format will follow according to ISO11898-1. 1 - CAN FD frame format will follow according to Bosch CAN FD Specification V1.0.
14 TXP	Transmit Pause 0 - Transmit pause is disabled. 1 - Transmit pause is enabled.
13 EFBI	Edge Filtering During Bus Integration 0 - Edge filtering is disabled. 1 - Two consecutive dominant quanta required to detect an edge for hard synchronization.
12 PXHD	Protocol Exception Handling Disable When PXHD is disabled, the MCAN module transmits an error frame when it detects a protocol exception condition. 0 - Protocol exception handling is enabled. 1 - Protocol exception handling is disabled.
11-10 —	Reserved Read value is undefined, only zero should be written.
9 BRSE	Bit Rate Switching Enable When CAN FD operation is disabled, this bit is not evaluated. 0 - Bit rate switching for transmissions is disabled. 1 - Bit rate switching for transmission is enabled.
8 FDOE	CAN FD Operation Enable 0 - CAN FD operation is disabled. 1 - CAN FD operation is enabled.
7 TEST	Test Mode Enable 0 - Normal operation 1 - Test mode enabled
6 DAR	Disable Automatic Retransmission 0 - Automatic retransmission of messages not transmitted successfully enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Automatic retransmission disabled.
5 MON	Bus Monitoring Mode 0 - Bus Monitoring mode is disabled. 1 - Bus Monitoring mode is enabled.
4 CSR	Clock Stop Request 0 - No clock stop is requested. 1 - Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reaches idle.
3 CSA	Clock Stop Acknowledge 0 - No clock stop acknowledged. 1 - MCAN may be set in Power Down mode by stopping the internal MCAN clocks.
2 ASM	Restricted Operational Mode 0 - Normal CAN operation 1 - Restricted operation mode active
1 CCE	Configuration Change Enable 0 - No write access. The CPU has no write access to the protected configuration registers. 1 - Write access. The CPU has write access to the protected configuration registers.
0 INIT	Initialization 0 - Normal operation 1 - Initialization is started

43.6.1.1.4 Nominal Bit Timing and Prescaler (NBTP)

Setting CCCR[CCE] and CCCR[INIT] = 1 enables write access to the Nominal Bit Timing and Prescaler (NBTP) register.

The CAN bit time may be programmed in the range of 4 to 385 time quanta. The CAN time quantum may be programmed in the range of 1 to 152 MCAN clock periods.

```
T_q = (DBRP + 1) MCAN clock periods
```

NTSEG1 is the sum of prop_seg and phase_Seg1. DTSEG2 is phase_seg2. Therefore, the length of the bit time is:

```
[NTSEG1 + NTSEG2 + 3] t_q for programmed values, or,
[sync_seg + prop_seg + phase_seg1 + phase_seg2] t_q for functional values.
```

The information processing time (IPT) is zero, meaning that the data for the next bit is available upon the first clock edge after the sample point.

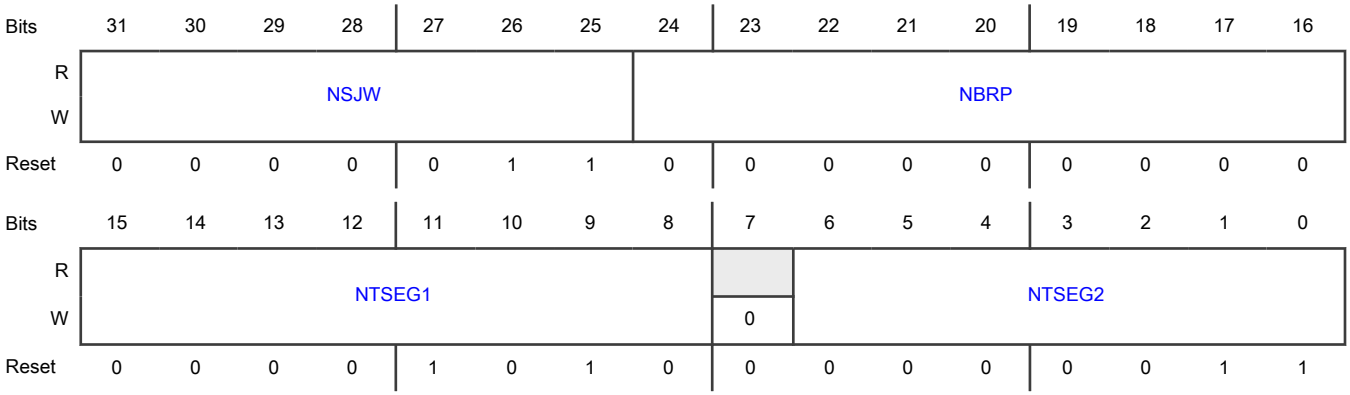
NOTE

With an MCAN clock of 8 MHz, the reset value of the NBTP register configures the MCAN for a data phase bit rate of 500 kb/s.

Offset

Register	Offset
NBTP	1Ch

Diagram



Fields

Field	Description
31-25 NSJW	Nominal (Re)Synchronization Jump Width The actual value used by hardware is one more than the value programmed here. Valid values are 0 to 127.
24-16 NBRP	Nominal Bit Rate Prescaler The value by which the oscillator frequency is divided for generating the bit time quanta. The actual value used by hardware is one more than the value programmed here. Valid values are 0 to 511.
15-8 NTSEG1	Nominal Time Segment Before Sample Point The actual value used by hardware is one more than the value programmed here. Valid values are 0 to 255.
7 —	Reserved Read value is undefined, only zero should be written.
6-0 NTSEG2	Nominal Time Segment After Sample Point The actual value used by hardware is one more than the value programmed here. Valid values are 0 to 127.

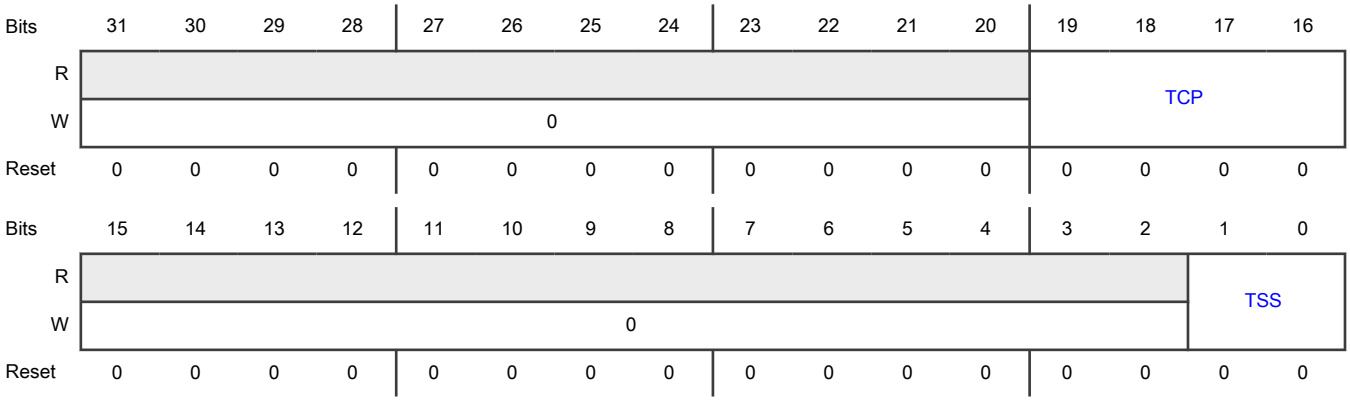
43.6.1.1.5 Timestamp Counter Configuration (TSCC)

The TSCC register configures the Timestamp and the time unit used.

Offset

Register	Offset
TSCC	20h

Diagram



Fields

Field	Description
31-20 —	Reserved Read value is undefined, only zero should be written.
19-16 TCP	Timestamp Counter Prescaler Configures the timestamp and timeout counters time unit in a multiple of CAN bit times. The actual value used by hardware is one more than the value programmed here. Valid values are 0 to 15.
15-2 —	Reserved Read value is undefined, only zero should be written.
1-0 TSS	Timestamp Select 00,11 - Timestamp counter value static at 0x0000 01 - Timestamp counter value incremented according to TCP bits 10 - External timestamp counter value used

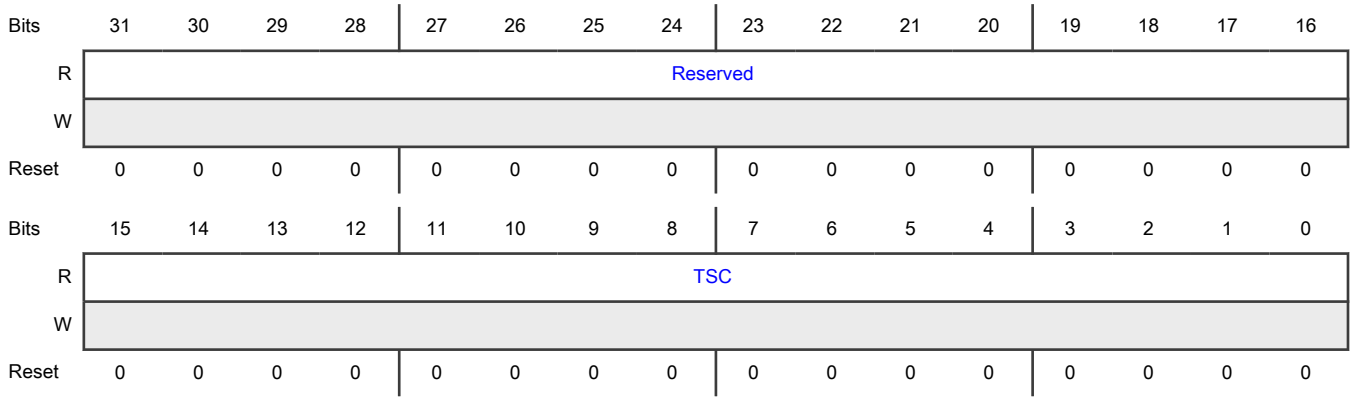
43.6.1.1.6 Timestamp Counter Value (TSCV)

The internal/external timestamp counter value is captured on start of frame (both Rx and Tx).

Offset

Register	Offset
TSCV	24h

Diagram



Fields

Field	Description
31-16 —	Reserved Read value is undefined, only zero should be written.
15-0 TSC	Timestamp Counter The timestamp counter increments depending on the TSCC[TSS] field. An overflow sets the IR[TSW] interrupt flag = 1.

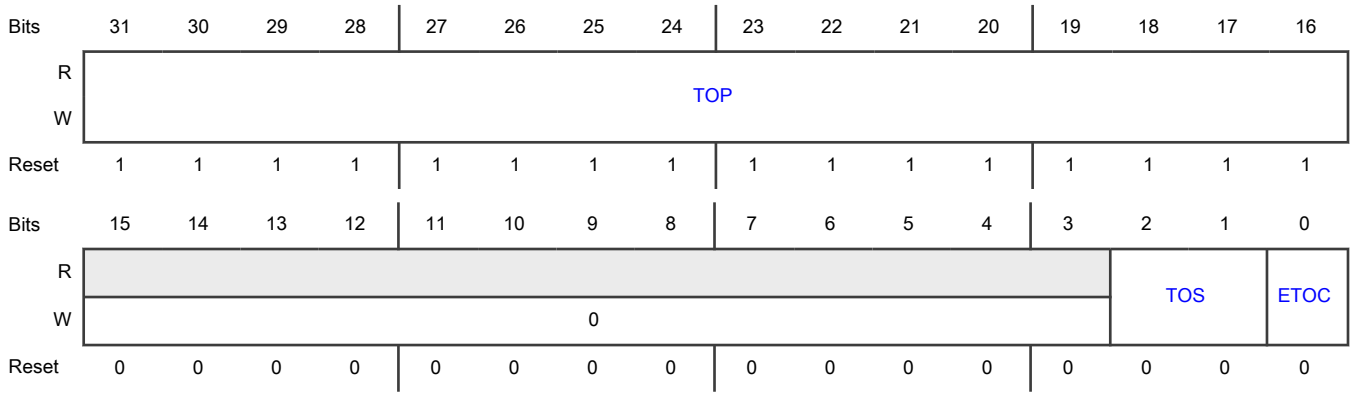
43.6.1.1.7 Timeout Counter Configuration (TOCC)

The TOCC register controls the Timeout Counter Configuration.

Offset

Register	Offset
TOCC	28h

Diagram



Fields

Field	Description
31-16 TOP	Timeout Period This register field holds the start value of the timeout counter to configure the timeout period. This counter counts down.
15-3 —	Reserved Read value is undefined, only zero should be written.
2-1 TOS	Timeout Select When the timeout counter is operating in continuous mode, a write to the TOCV register presets the counter to the value configured in the TOCC[TOP] field and continues down-counting. When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by the TOCC[TOP] field. Down-counting starts when the first FIFO element is stored. 00 - Continuous operation 01 - Timeout is controlled by Tx event FIFO. 10 - Timeout is controlled by Rx FIFO 0. 11 - Timeout is controlled by Rx FIFO 1.
0 ETOC	Enable Timeout Counter 0 - Timeout counter is disabled. 1 - Timeout counter is enabled.

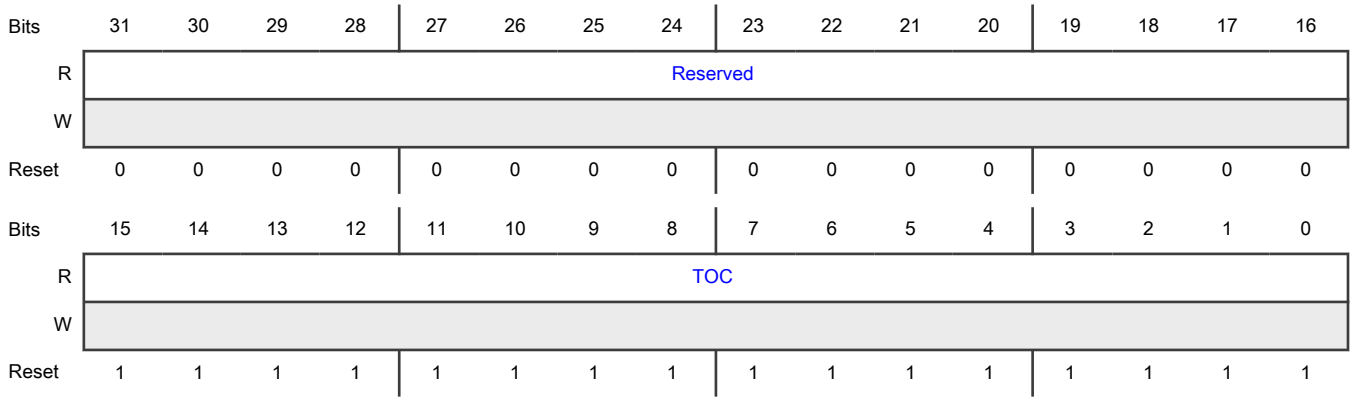
43.6.1.1.8 Timeout Counter Value (TOCV)

Start and reset conditions are configured by the TOC[TOS] field.

Offset

Register	Offset
TOCV	2Ch

Diagram



Fields

Field	Description
31-16 —	Reserved Read value is undefined, only zero should be written.
15-0 TOC	Timeout Counter The Timeout Counter is decremented in multiples of CAN bit times depending on the configuration of the TSCC[TCP] field. When decremented to zero, the IR[TOO] interrupt flag field = 1 and the timeout counter is stopped.

43.6.1.1.9 Error Counter (ECR)

The ECR register holds the TX and RX error counts and CAN error logging.

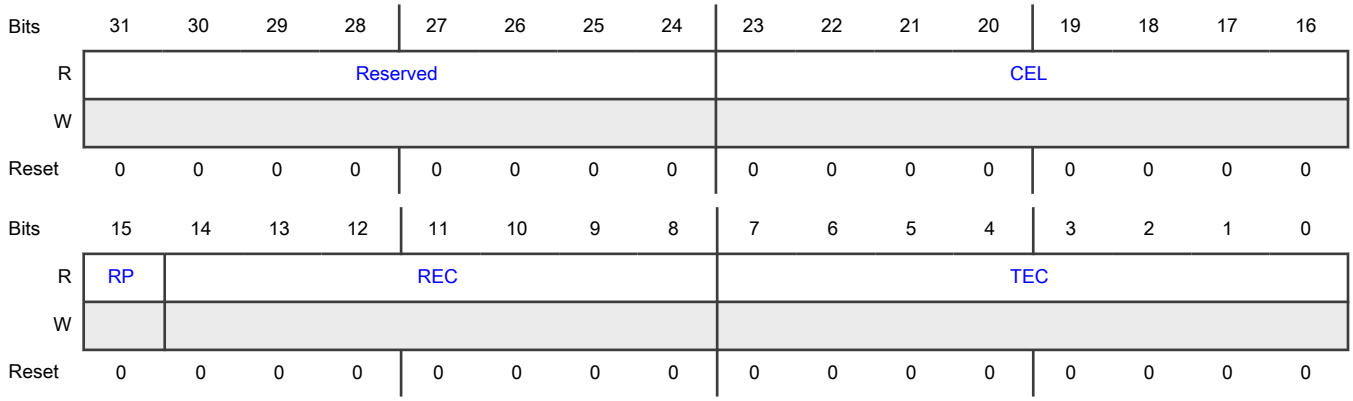
NOTE

When the CCCR[ASM] field = 1, the CAN protocol controller does not increment the TEC and REC bits when a CAN protocol error is detected, but the CEL bits are still incremented.

Offset

Register	Offset
ECR	40h

Diagram



Fields

Field	Description
31-24 —	Reserved Read value is undefined, only zero should be written.
23-16	CAN Error Logging

Table continues on the next page...

Table continued from the previous page...

Field	Description
CEL	CEL is incremented when TEC or REC is incremented. The counter stops at 0xFF and the next occurrence of a CAN protocol error sets the IR[ELO] interrupt flag = 1. This counter is reset whenever a read access to these bits happens.
15 RP	Receive Error Passive 0 - Below error level. The receive counter is below the error passive level of 128. 1 - At error level. The receive counter has reached the error passive level of 128.
14-8 REC	Receive Error Counter Current value of the receive error counter. Valid values are between 0 and 127.
7-0 TEC	Transmit Error Counter Current value of the transmit error counter. Valid values are between 0 and 255.

43.6.1.1.10 Protocol Status (PSR)

NOTE

When a frame in a CAN FD format reaches the data phase with the BRS flag set, the next CAN event (error or valid frame) will be shown in the DLEC bits instead of LEC bits. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error.

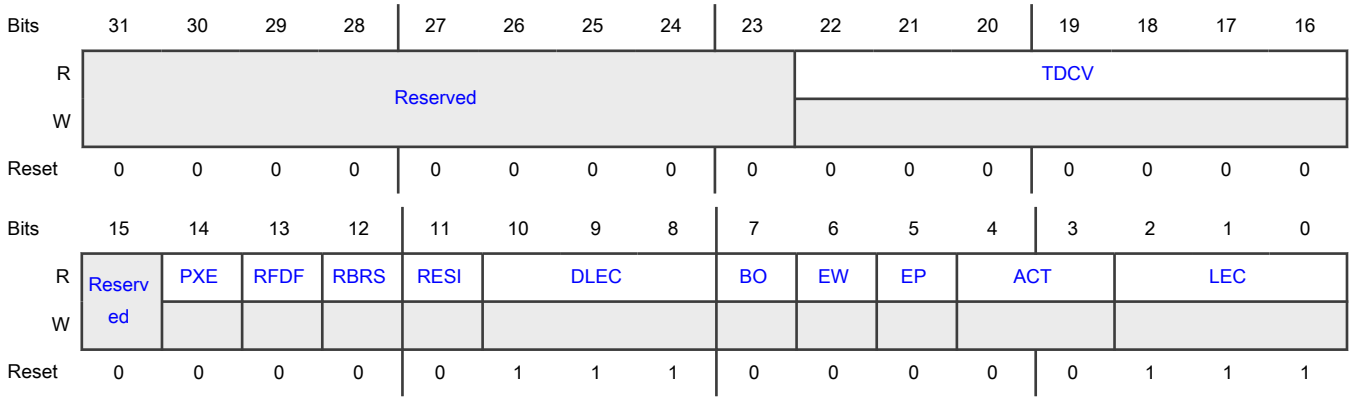
NOTE

The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting the CCCR[INIT] bit. If the device goes Bus_Off, it will set the CCCR[INIT] bit of its own accord, stopping all bus activities. After the CCCR[INIT] bit clears, the device waits for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters reset. During the waiting time after resetting CCCR[INIT], each time a sequence of 11 recessive bits has been monitored, a Bit0Error code writes to the LEC bits to check whether the CAN bus is stuck at dominant or continuously disturbed, and to monitor the Bus_Off recovery sequence. The ECR[REC] field counts these sequences.

Offset

Register	Offset
PSR	44h

Diagram



Fields

Field	Description
31-23 —	Reserved Read value is undefined, only zero should be written.
22-16 TDCV	Transmitter Delay Compensation Value Position of the secondary sample point, defined by the sum of the measured delay from m_can_tx and m_can_rx and TDCO bits in the TDCR register. The SSP position in the data phase is the number of MCAN clock periods between the start of a transmitted bit and secondary sample point. Valid values are 0 to 127 MCAN clock periods.
15 —	Reserved Read value is undefined, only zero should be written.
14 PXE	Protocol Exception Event PXE will be set upon a read access. 0 - No protocol exception event occurred since last read access. 1 - Protocol exception event occurred.
13 RFDF	Received a CAN FD Message RFDF is set independent of acceptance filtering. RFDF will be set on a read access. 0 - No CAN FD message received since the last CPU reset. 1 - Message in CAN FD format with FDF flag set has been received.
12 RBRS	BRS Flag of Last Received CAN FD Message RBRS is set together with RFDF bits, independent of acceptance filtering. RBRS will be set upon a read access. 0 - Last received CAN FD message did not have its BRS flag set. 1 - Last received CAN FD message had its BRS flag set.
11	ESI Flag of the Last Received CAN FD Message

Table continues on the next page...

Table continued from the previous page...

Field	Description
RESI	RESI is set together with RFDF bits, independent of acceptance filtering. RESI will be set upon a read access. 0 - Last received CAN FD message did not have its ESI flag set. 1 - Last received CAN FD message had its ESI flag set.
10-8 DLEC	Data Phase Last Error Code Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC bits. This field clears to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error. The bits in this bit field will be set upon a read access.
7 BO	Bus Off Status 0 - Disabled 1 - Enabled
6 EW	Warning Status 0 - Both error counters are below the Error_Warning limit of 96. 1 - At least one of error counter has reached the Error_Warning limit of 96.
5 EP	Error Passive 0 - The MCAN is in Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected. 1 - The MCAN is in the Error_Passive state.
4-3 ACT	Activity ACT monitors the MCAN communication state. 00 - Synchronizing – node is synchronizing on CAN communication. 01 - Idle – node is neither receiver nor transmitter. 10 - Receiver – node is operating as receiver. 11 - Transmitter – node is operating as transmitter.
2-0 LEC	Last Error Code The LEC field indicates the type of the last error to occur on the CAN bus. This field clears when a message transfers without error. The field sets when there is a read access. 000 - No error: No error has occurred since the LEC bits has been reset by successful reception or transmission. 001 - Stuff error: More than 5 equal bits in a sequence have occurred in a part of a received message where not allowed. 010 - Form error: A fixed format part of a received frame has the wrong format. 011 - AckError: The message transmitted by the MCAN was not acknowledged by another node.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	100 - Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant. 101 - Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the processing of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed). 110 - CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data. 111 - NoChange: Any read access to the protocol status register re-initializes the LEC bits to 0x7. When the LEC equals the value 0x7, no CAN bus event was detected since the last CPU read access to the protocol status register.

43.6.1.1.11 Transmitter Delay Compensator (TDCR)

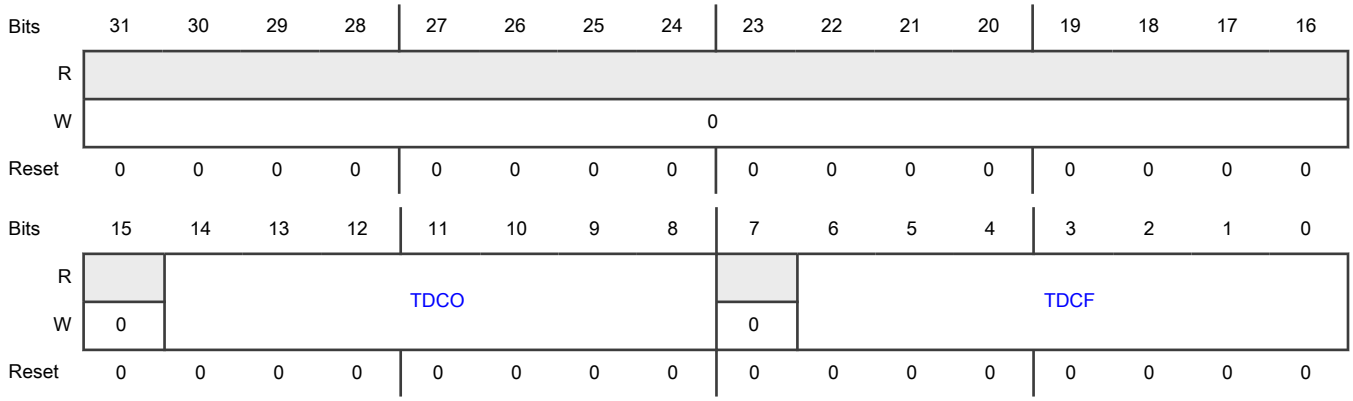
The TDCR register holds the values used to adjust the position of the Secondary Sample Point (SSP) inside the received bit.

Valid values for TDCF and TDC0 are 0 to 127 MCAN clock periods.

Offset

Register	Offset
TDCR	48h

Diagram



Fields

Field	Description
31-15	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	Read value is undefined, only zero should be written.
14-8 TDCO	Transmitter Delay Compensation Offset Offset value defining the distance between the measured delay from m_can_tx to m_can_rx and the secondary sample point.
7 —	Reserved Read value is undefined, only zero should be written.
6-0 TDCF	Transmitter Delay Compensation Filter Window Length Defines the minimum value for the SSP position, dominant edges on m_can_rx that would result in an earlier SSP position are ignored for transmitter delay measurement. The feature is enabled when TDCF bits are configured to a value greater than TDCO bits.

43.6.1.1.12 Interrupt (IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the host clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset will clear the register. The configuration of the IE register controls whether an interrupt is generated. The configuration of the ILS register controls on which interrupt line an interrupt is signaled.

Offset

Register	Offset
IR	50h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			ARA	PED	PEA	WDI	BO	EW	EP	ELO	BEU	BEC	DRX	TOO	MRAF	TSW
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	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-30 —	Reserved Read value is undefined, only zero should be written.
29 ARA	Access to Reserved Address 0 - No access to reserved address occurred. 1 - Access to reserved address occurred.
28 PED	Protocol Error in Data Phase 0 - No protocol error in data phase. 1 - Protocol error in data phase detected.
27 PEA	Protocol Error in Arbitration Phase 0 - No protocol error in arbitration phase. 1 - Protocol error in arbitration phase detected.
26 WDI	Watchdog Interrupt 0 - No message RAM watchdog event occurred. 1 - Message RAM watchdog event due to missing READY.
25 BO	Bus_Off Status 0 - Bus_Off status unchanged. 1 - Bus_Off status changed.
24 EW	Warning Status 0 - Error_Warning status unchanged. 1 - Error_Warning status changed.
23 EP	Error Passive 0 - Error_Passive status unchanged. 1 - Error_Passive status changed.
22 ELO	Error Logging Overflow 0 - CAN error logging counter did not overflow. 1 - Overflow of CAN error logging counter occurred.
21 BEU	Bit Error Uncorrected Message RAM bit error detected, uncorrected. Controlled by input signal m_can_aeim_berr[1] generated by an optional external parity / ECC logic attached to the Message RAM. An uncorrected Message RAM bit error sets INIT bit in the CCCR register to 1. This is done to avoid transmission of corrupted data. 0 - No bit error detected when reading from message RAM. 1 - Bit error detected, uncorrected (example, parity logic).

Table continues on the next page...

Table continued from the previous page...

Field	Description
20 BEC	Bit Error Corrected Message RAM bit error detected and corrected. Controlled by input signal m_can_aeim_berr[0] generated by an optional external parity / ECC logic attached to the message RAM. 0 - No bit error detected when reading from message RAM. 1 - Bit error detected and corrected (example, ECC).
19 DRX	Message Stored in Dedicated Rx Buffer 0 - No Rx buffer updated. 1 - At least one received message stored into an Rx buffer.
18 TOO	Timeout Occurred 0 - No timeout. 1 - Timeout reached.
17 MRAF	Message RAM Access Failure The flag is set when the Rx Handler meets either of the following criteria: • The Rx handler has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx Handler starts processing of the following message. • The Rx handler was not able to write a message to the Message RAM and the message storage is aborted. In both cases the FIFO put index is not updated resp. the New Data flag for a dedicated Rx buffer is not set, a partly stored message is overwritten when the next message is stored to this location. • The flag is also set when the Tx Handler was not able to read a message from the Message RAM in time. In this case message transmission is aborted. In case of a Tx Handler access failure the MCAN is switched into Restricted Operation Mode. To leave Restricted Operation Mode, CCCR[ASM] bit must be cleared (CCCR[ASM] = 0). 0 - No message RAM access failure occurred. 1 - Message RAM access failure occurred.
16 TSW	Timestamp Wraparound 0 - No timestamp counter wraparound. 1 - Timestamp counter wrapped around.
15 TEFL	Tx Event FIFO Element Lost 0 - No Tx event FIFO element lost. 1 - Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size zero.
14 TEFF	Tx Event FIFO Full 0 - Tx event FIFO not full. 1 - Tx event FIFO full.

Table continues on the next page...

Table continued from the previous page...

Field	Description
13 TEFW	Tx Event FIFO Watermark Reached 0 - Tx event FIFO fill level below watermark. 1 - Tx event FIFO fill level reached watermark.
12 TEFN	Tx Event FIFO New Entry 0 - Tx event FIFO unchanged. 1 - Tx Handler wrote Tx event FIFO element.
11 TFE	Tx FIFO Empty 0 - Tx FIFO non-empty. 1 - Tx FIFO empty.
10 TCF	Transmission Cancellation Finished 0 - No transmission cancellation finished. 1 - Transmission cancellation finished.
9 TC	Transmission Completed 0 - No transmission completed. 1 - Transmission completed.
8 HPM	High Priority Message 0 - No high priority message received. 1 - High priority message received.
7 RF1L	Rx FIFO 1 Message Lost 0 - No Rx FIFO 1 message lost. 1 - Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero.
6 RF1F	Rx FIFO 1 Full 0 - Rx FIFO 1 not full. 1 - Rx FIFO 1 full.
5 RF1W	Rx FIFO 1 Watermark Reached 0 - Rx FIFO 1 fill level below watermark. 1 - Rx FIFO 1 fill level reached watermark.
4 RF1N	Rx FIFO 1 New Message 0 - No new message written to Rx FIFO 1. 1 - New message written to Rx FIFO 1.
3 RF0L	Rx FIFO 0 Message Lost 0 - No Rx FIFO 0 message lost.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero.
2 RF0F	Rx FIFO 0 Full 0 - Rx FIFO 0 not full. 1 - Rx FIFO 0 full.
1 RF0W	Rx FIFO 0 Watermark Reached 0 - Rx FIFO 0 fill level below watermark. 1 - Rx FIFO 0 fill level reached watermark.
0 RF0N	Rx FIFO 0 New Message 0 - No new message written to Rx FIFO 0. 1 - New message written to Rx FIFO 0.

43.6.1.1.13 Interrupt Enable (IE)

The setting in the interrupt enable register determines which status changes in the interrupt register will be signaled on an interrupt line.

Offset

Register	Offset
IE	54h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	BEUE	BECE	DRXE	TOOE	MRAFE	TSWE
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	TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-30	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	Read value is undefined, only zero should be written.
29 ARAE	Access to Reserved Address Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
28 PEDE	Protocol Error in Data Phase Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
27 PEAE	Protocol Error in Arbitration Phase Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
26 WDIE	Watchdog Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
25 BOE	Bus_Off Status Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
24 EWE	Warning Status Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
23 EPE	Error Passive Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
22 ELOE	Error Logging Overflow Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
21 BEUE	Bit Error Uncorrected Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
20 BECE	Bit Error Corrected Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled

Table continues on the next page...

Table continued from the previous page...

Field	Description
19 DRXE	Message Stored in Dedicated Rx Buffer Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
18 TOOE	Timeout Occurred Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
17 MRAFE	Message RAM Access Failure Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
16 TSWE	Timestamp Wraparound Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
15 TEFLE	Tx Event FIFO Element Lost Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
14 TEFFE	Tx Event FIFO Full Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
13 TEFWE	Tx Event FIFO Watermark Reached Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
12 TEFNE	Tx Event FIFO New Entry Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
11 TFEE	Tx FIFO Empty Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
10 TCFE	Transmission Cancellation Finished Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
9 TCE	Transmission Completed Interrupt Enable 0 - Interrupt disabled

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Interrupt enabled
8 HPME	High Priority Message Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
7 RF1LE	Rx FIFO 1 Message Lost Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
6 RF1FE	Rx FIFO 1 Full Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
5 RF1WE	Rx FIFO 1 Watermark Reached Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
4 RF1NE	Rx FIFO 1 New Message Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
3 RF0LE	Rx FIFO 0 Message Lost Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
2 RF0FE	Rx FIFO 0 Full Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
1 RF0WE	Rx FIFO 0 Watermark Reached Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled
0 RF0NE	Rx FIFO 0 New Message Interrupt Enable 0 - Interrupt disabled 1 - Interrupt enabled

43.6.1.1.14 Interrupt Line Select (ILS)

The interrupt line select register assigns an interrupt generated by a specific interrupt flag from the interrupt register. For interrupt generation, the respective interrupt line has to be enabled through the interrupt line enable register.

Offset

Register	Offset
ILS	58h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R			ARAL	PEDL	PEAL	WDIL	BOL	EWL	EPL	ELOL	BEUL	BECL	DRXL	TOOL	MRAFL	TSWL
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	TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-30 —	Reserved Read value is undefined, only zero should be written.
29 ARAL	Access to Reserved Address Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
28 PEDL	Protocol Error in Data Phase Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
27 PEAL	Protocol Error in Arbitration Phase Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
26 WDIL	Watchdog Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
25 BOL	Bus_Off Status Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1

Table continues on the next page...

Table continued from the previous page...

Field	Description
24 EWL	Warning Status Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
23 EPL	Error Passive Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
22 ELOL	Error Logging Overflow Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
21 BEUL	Bit Error Uncorrected Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
20 BECL	Bit Error Corrected Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
19 DRXL	Message Stored in Dedicated Rx Buffer Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
18 TOOL	Timeout Occurred Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
17 MRAFL	Message RAM Access Failure Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
16 TSWL	Timestamp Wraparound Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
15 TEFLL	Tx Event FIFO Element Lost Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
14 TEFFL	Tx Event FIFO Full Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Interrupt assigned to interrupt line MCANx_INT1
13 TEFWL	Tx Event FIFO Watermark Reached Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
12 TEFNL	Tx Event FIFO New Entry Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
11 TFEL	Tx FIFO Empty Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
10 TCFL	Transmission Cancellation Finished Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
9 TCL	Transmission Completed Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
8 HPML	High Priority Message Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
7 RF1LL	Rx FIFO 1 Message Lost Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
6 RF1FL	Rx FIFO 1 Full Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
5 RF1WL	Rx FIFO 1 Watermark Reached Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
4 RF1NL	Rx FIFO 1 New Message Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1

Table continues on the next page...

Table continued from the previous page...

Field	Description
3 RF0LL	Rx FIFO 0 Message Lost Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
2 RF0FL	Rx FIFO 0 Full Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
1 RF0WL	Rx FIFO 0 Watermark Reached Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1
0 RF0NL	Rx FIFO 0 New Message Interrupt Line 0 - Interrupt assigned to interrupt line MCANx_INT0 1 - Interrupt assigned to interrupt line MCANx_INT1

43.6.1.1.15 Interrupt Line Enable (ILE)

The ILE register is used to enable or disable the two interrupt lines available.

Offset

Register	Offset
ILE	5Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
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															EINT1	EINT0
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-2 —	Reserved Read value is undefined, only zero should be written.
1 EINT1	Enable Interrupt Line 1 0 - Interrupt line to MCANx_INT1 is disabled. 1 - Interrupt line to MCANx_INT1 is enabled.
0 EINT0	Enable Interrupt Line 0 0 - Interrupt line to MCANx_INT0 is disabled. 1 - Interrupt line to MCANx_INT0 is enabled.

43.6.1.1.16 Global Filter Configuration (GFC)

The GFC register controls message filtering.

Offset

Register	Offset
GFC	80h

Diagram

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

Fields

Field	Description
31-6 —	Reserved Read value is undefined, only zero should be written.
5-4	Accept Non-matching Frames Standard

Table continues on the next page...

Table continued from the previous page...

Field	Description
ANFS	00 - Accept in Rx FIFO 0 01 - Accept in Rx FIFO 1 10,11 - Reject
3-2 ANFE	Accept Non-matching Frames Extended Defines how receives messages with 29-bit IDs that do not match any element of the filter list are treated. 00 - Accept in Rx FIFO 0 01 - Accept in Rx FIFO 1 10,11 - Reject
1 RRFS	Reject Remote Frames Standard 0 - Filter remote frames with 11-bit standard IDs 1 - Reject all remote frames with 11-bit standard IDs
0 RRFE	Reject Remote Frames Extended 0 - Filter remote frames with 29-bit extended IDs 1 - Reject all remote frames with 29-bit extended IDs

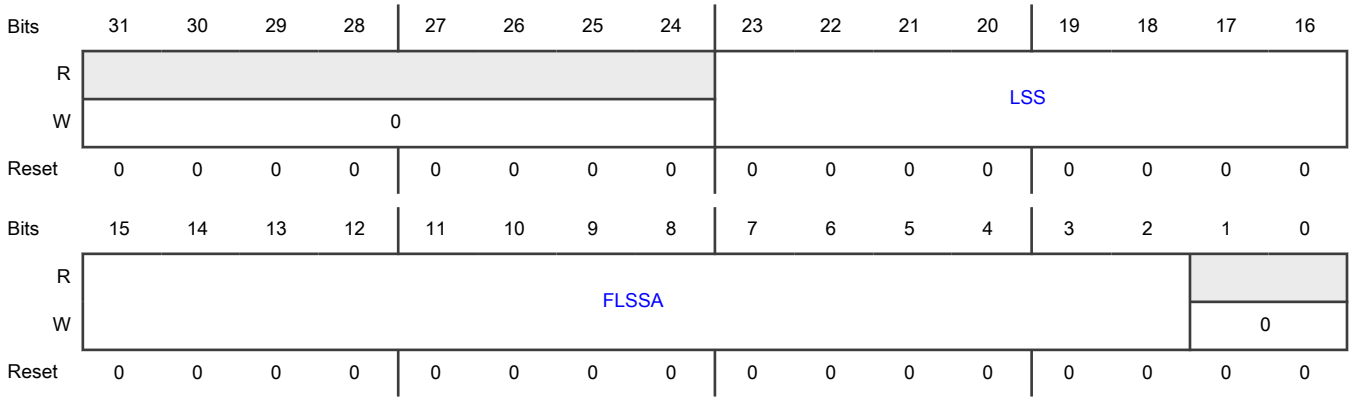
43.6.1.1.17 Standard ID Filter Configuration (SIDFC)

The SIDFC register controls the filter path for standard messages and settings for the 11-bit standard message ID filter.

Offset

Register	Offset
SIDFC	84h

Diagram



Fields

Field	Description
31-24 —	Reserved Read value is undefined, only zero should be written.
23-16 LSS	List Size Standard • 0 = No standard message ID filter. • 1 - 128 = Number of standard message ID filter elements • > 128 = Values of greater than 128 are interpreted as 128
15-2 FLSSA	Filter List Standard Start Address Start address of the standard message ID filter list
1-0 —	Reserved Read value is undefined, only zero should be written.

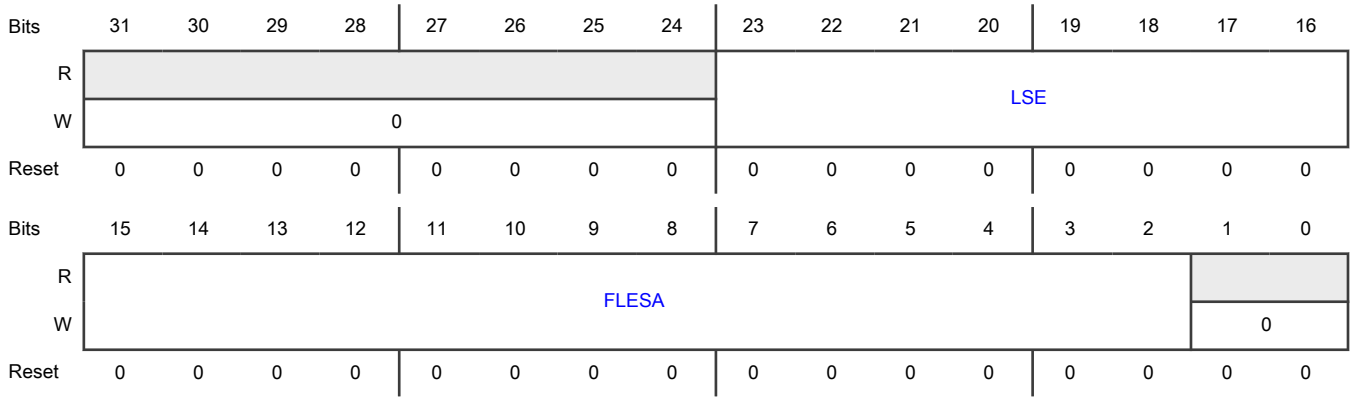
43.6.1.1.18 Extended ID Filter Configuration (XIDFC)

The XIDFC register controls the filter path for extended messages and settings for the 29-bit extended message ID filter.

Offset

Register	Offset
XIDFC	88h

Diagram



Fields

Field	Description
31-24	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	Read value is undefined, only zero should be written.
23-16 LSE	List Size Extended • 0 = No extended message ID filter. • 1 – 64 = Number of extended message ID filter elements • > 64 = Values of greater than 64 are interpreted as 64
15-2 FLESA	Filter List Extended Start Address Start address of the extended message ID filter list
1-0 —	Reserved Read value is undefined, only zero should be written.

43.6.1.1.19 Extended ID AND Mask (XIDAM)

The XIDAM register controls the acceptance filtering of extended frames. The value specified in the EIDM bits is ANDed with the message ID of a received frame. The reset value of the EIDM bits is all 1's, which makes the mask not active.

Offset

Register	Offset
XIDAM	90h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R					EIDM											
W		0														
Reset	0	0	0	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	EIDM															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Description
31-29	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	Read value is undefined, only zero should be written.
28-0 EIDM	Extended ID Mask

43.6.1.1.20 High Priority Message Status (HPMS)

The HPMS register is updated every time a message ID filter element configured to generate a priority event matches. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Offset

Register	Offset
HPMS	94h

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	FLST	FIDX							MSI		BIDX					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-16 —	Reserved Read value is undefined, only zero should be written.
15 FLST	Filter List Indicates the filter list of the matching filter element. 0 - Standard filter list 1 - Extended filter list
14-8	Filter Index

Table continues on the next page...

Table continued from the previous page...

Field	Description
FIDX	Index of matching filter element. The range is 1 minus the LSS bits in the SIDFC register resp. LSE bits in the XIDFC registers.
7-6 MSI	Message Storage Indicator 00 - No FIFO selected 01 - FIFO message lost 10 - Message stored in FIFO 0 11 - Message stored in FIFO 1
5-0 BIDX	Buffer Index Index of the Rx FIFO element to which the message was stored. This is only valid when the MSI bits are configured to store the message in either FIFO 0 or FIFO 1 (MSI = 0x2 or 0x3).

43.6.1.1.21 New Data 1 (NDAT1)

The NDAT1 register holds the new data flags of Rx buffers 0 to 31. The flags are set when the respective Rx buffer updates from a received frame. The flags remain set until they are cleared. Clear a flag by writing a 1 to the corresponding bit position.

Offset

Register	Offset
NDAT1	98h

Diagram

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

Fields

Field	Description
31-0 ND	New Data The most significant bit in this register corresponds to Rx buffer 31 while the least significant bit in this register corresponds to Rx buffer 0.

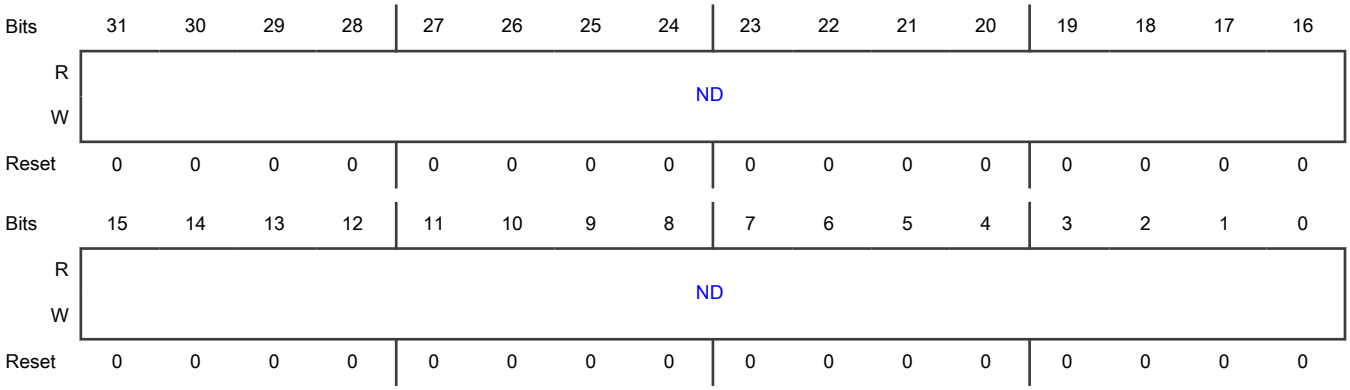
43.6.1.1.22 New Data 2 (NDAT2)

The NDAT2 register holds the new data flags of Rx buffers 32 to 63. The flags are set when the respective Rx buffer updates from a received frame. The flags remain set until they are cleared. Clear a flag by writing a 1 to the corresponding bit position.

Offset

Register	Offset
NDAT2	9Ch

Diagram



Fields

Field	Description
31-0 ND	New Data The most significant bit in this register corresponds to Rx buffer 63 while the least significant bit in this register corresponds to Rx buffer 32.

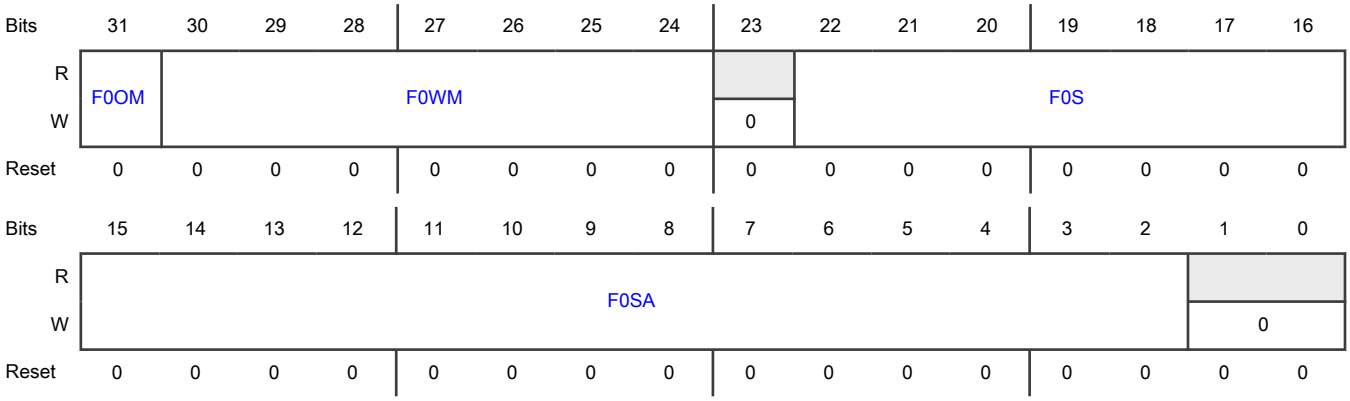
43.6.1.1.23 Rx FIFO 0 Configuration (RXF0C)

The RXFOC register contains the Rx FIFO 0 configuration information.

Offset

Register	Offset
RXF0C	A0h

Diagram



Fields

Field	Description
31 F0OM	FIFO 0 Operation Mode The FIFO can be operated in block or overwrite mode. 0 - FIFO 0 blocking mode 1 - FIFO 0 overwrite mode
30-24 F0WM	Rx FIFO 0 Watermark • 0 = Watermark interrupt disabled • 1 – 64 = Level for RF0W interrupt flag in the IR register • > 64 = Watermark interrupt disabled
23 —	Reserved Read value is undefined, only zero should be written.
22-16 F0S	Rx FIFO 0 Size • 0 = No Rx FIFO 0 • 1 – 64 = Number of Rx FIFO 0 elements • > 64 = Values greater than 64 are interpreted as 64 The maximum Rx FIFO 0 elements are the value set in this register minus 1.
15-2 F0SA	Rx FIFO 0 Start Address Start address of Rx FIFO 0 in message RAM
1-0 —	Reserved Read value is undefined, only zero should be written.

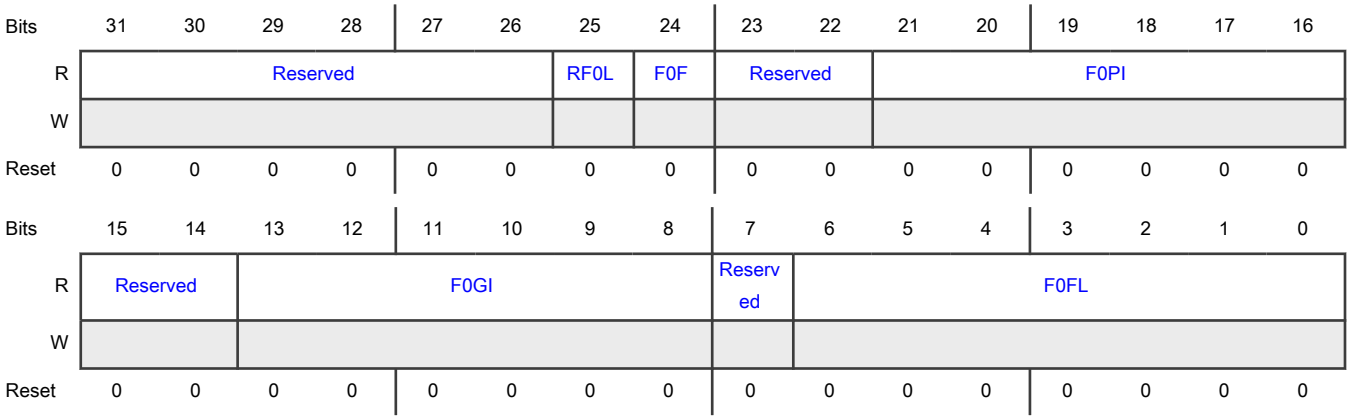
43.6.1.1.24 Rx FIFO 0 Status (RXF0S)

The RXFOS register contains the Rx FIFO 0 status information.

Offset

Register	Offset
RXF0S	A4h

Diagram



Fields

Field	Description
31-26 —	Reserved Read value is undefined, only zero should be written.
25 RF0L	Rx FIFO 0 Message Lost This bit is a copy of the RF0L interrupt flag in the IR register. **NOTE** Overwriting the oldest message when the F0OM bit in the RXF0C register is set to 1 will not set this flag.
24 F0F	Rx FIFO 0 Full 0 - Rx FIFO 0 not full 1 - Rx FIFO 0 full
23-22 —	Reserved Read value is undefined, only zero should be written.
21-16 F0PI	Rx FIFO 0 Put Index Rx FIFO 0 write index pointer, range 0 to 63.
15-14 —	Reserved Read value is undefined, only zero should be written.
13-8	Rx FIFO 0 Get Index

Table continues on the next page...

Table continued from the previous page...

Field	Description
F0GI	Rx FIFO 0 read index pointer, range 0 to 63.
7 —	Reserved Read value is undefined, only zero should be written.
6-0 F0FL	Rx FIFO 0 Fill Level Number of elements stored in Rx FIFO0, range 0 to 64.

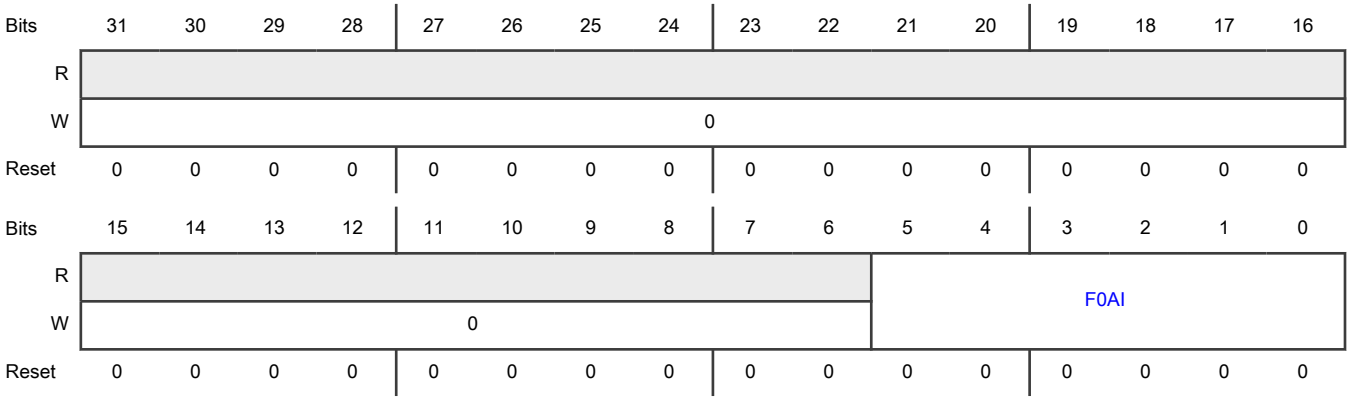
43.6.1.1.25 Rx FIFO 0 Acknowledge (RXF0A)

After a message or sequence of messages have been read from Rx FIFO 0, the buffer index of the last element read from the Rx FIFO 0 must be written to the F0AI bits. This will set the Rx FIFO 0 get index bits to the Rx FIFO 0 acknowledge index bit value + 1 and update the FIFO 0 fill level bits.

Offset

Register	Offset
RXF0A	A8h

Diagram



Fields

Field	Description
31-6 —	Reserved Read value is undefined, only zero should be written.
5-0 F0AI	Rx FIFO 0 Acknowledge Index

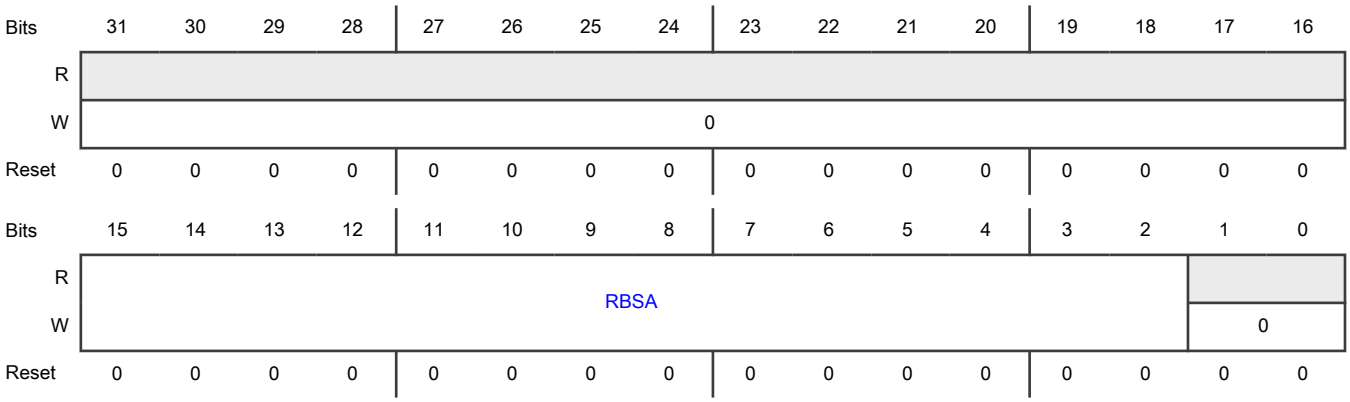
43.6.1.1.26 Rx Buffer Configuration (RXBC)

The RXBC register configures the start address of the Rx buffers section in the message RAM.

Offset

Register	Offset
RXBC	ACh

Diagram



Fields

Field	Description
31-16 —	Reserved Read value is undefined, only zero should be written.
15-2 RBSA	Rx Buffer Start Address
1-0 —	Reserved Read value is undefined, only zero should be written.

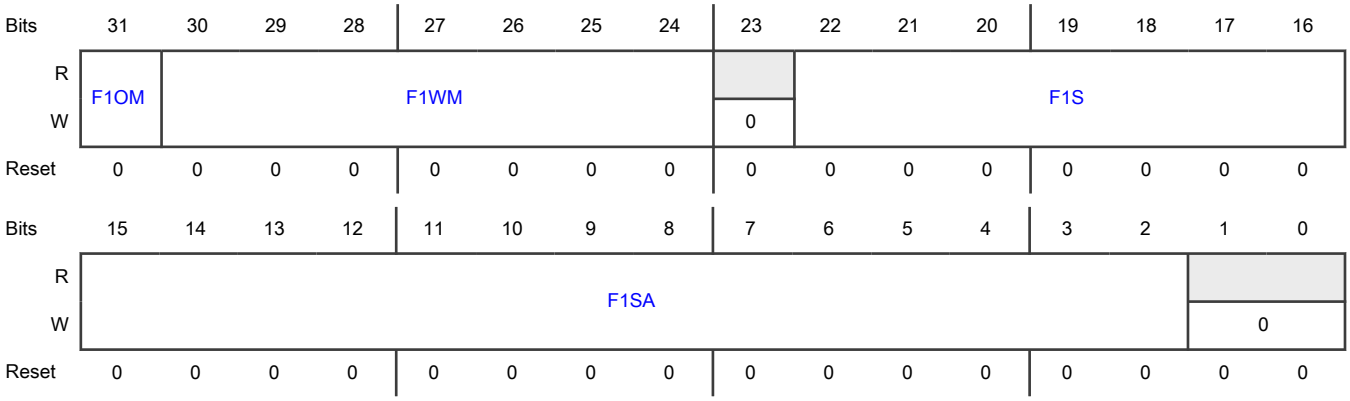
43.6.1.1.27 Rx FIFO 1 Configuration (RXF1C)

The RXF1C register contains the Rx FIFO 1 configuration information.

Offset

Register	Offset
RXF1C	B0h

Diagram



Fields

Field	Description
31 F1OM	FIFO 1 Operation Mode The FIFO can be operated in block or overwrite mode. 0 - FIFO 1 blocking mode 1 - FIFO 1 overwrite mode
30-24 F1WM	Rx FIFO 1 Watermark • 0 = Watermark interrupt disabled • 1 – 64 = Level for RF1W interrupt flag in the IR register • > 64 = Watermark interrupt disabled
23 —	Reserved Read value is undefined, only zero should be written.
22-16 F1S	Rx FIFO 1 Size • 0 = No Rx FIFO 1 • 1 – 64 = Number of Rx FIFO 1 elements • > 64 = Values greater than 64 are interpreted as 64 The maximum Rx FIFO 1 elements are the value set in this register minus 1.
15-2 F1SA	Rx FIFO 1 Start Address Start address of Rx FIFO 1 in message RAM.
1-0 —	Reserved Read value is undefined, only zero should be written.

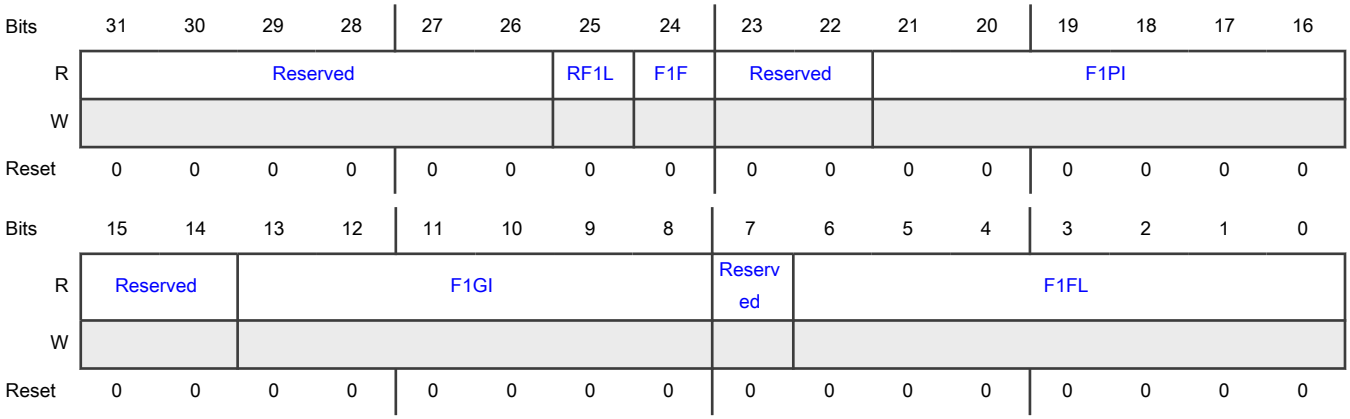
43.6.1.1.28 Rx FIFO 1 Status (RXF1S)

The RXF1S register contains the Rx FIFO 1 status information.

Offset

Register	Offset
RXF1S	B4h

Diagram



Fields

Field	Description
31-26 —	Reserved Read value is undefined, only zero should be written.
25 RF1L	Rx FIFO 1 message lost. This bit is a copy of the RF1L interrupt flag in the IR register. **NOTE** Overwriting the oldest message when the F1OM bit from the RXF1C register is set to 1 will not set this flag. 0 - No Rx FIFO 1 message lost. 1 - Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero.
24 F1F	Rx FIFO 1 Full 0 - Rx FIFO 1 not full 1 - Rx FIFO 1 full
23-22 —	Reserved Read value is undefined, only zero should be written.
21-16 F1PI	Rx FIFO 1 Put Index Rx FIFO 1 write index pointer, range 0 to 63.
15-14	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	Read value is undefined, only zero should be written.
13-8 F1GI	Rx FIFO 1 Get Index Rx FIFO 1 read index pointer, range 0 to 63.
7 —	Reserved Read value is undefined, only zero should be written.
6-0 F1FL	Rx FIFO 1 Fill Level Number of elements stored in Rx FIFO 1, range 0 to 64.

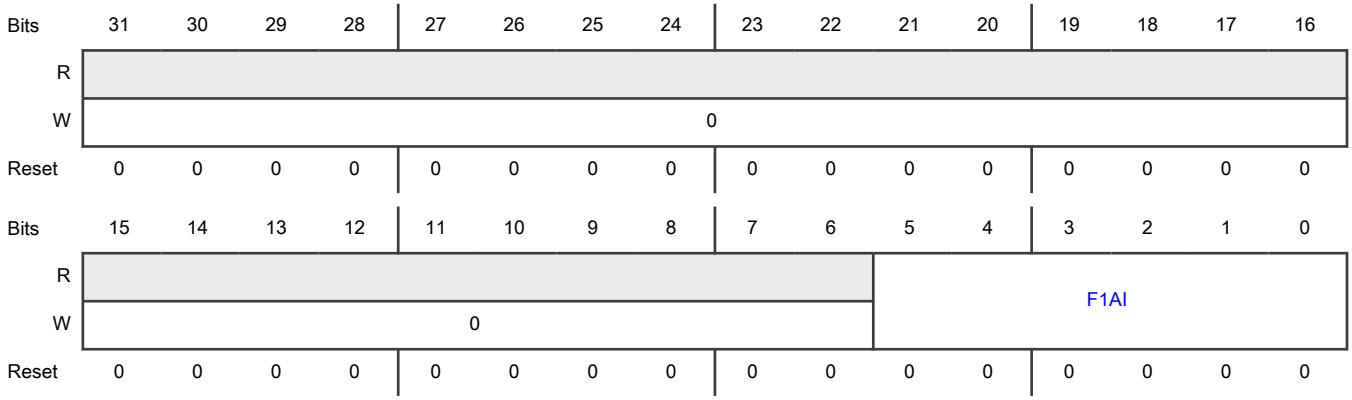
43.6.1.1.29 Rx FIFO 1 Acknowledge (RXF1A)

After a message or sequence of messages have been read from Rx FIFO 1, the buffer index of the last element read from the Rx FIFO 1 has to be written to the F1AI bits. This will set the Rx FIFO 1 get index bits to the Rx FIFO 1 acknowledge index bit value + 1 and update the FIFO 1 fill level bits.

Offset

Register	Offset
RXF1A	B8h

Diagram



Fields

Field	Description
31-6 —	Reserved Read value is undefined, only zero should be written.

Table continues on the next page...

Table continued from the previous page...

Field	Description
5-0 F1AI	Rx FIFO 1 Acknowledge Index

43.6.1.1.30 Rx Buffer and FIFO Element Size Configuration (RXESC)

The RXESC register configures the number of data bytes belonging to the Rx buffer and Rx FIFO element. Data field sizes that are greater than 8 bytes are intended for CAN FD operation only.

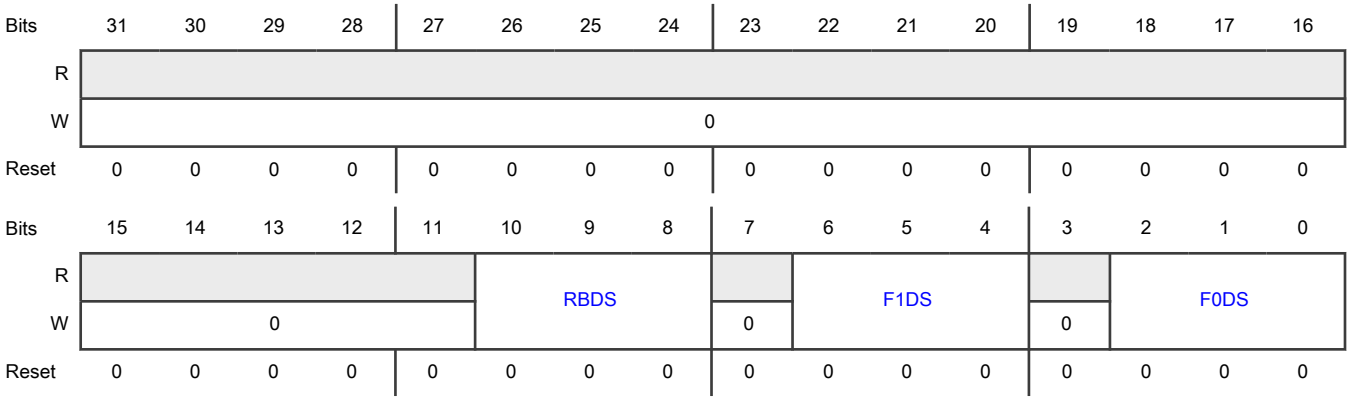
NOTE

In case the data field size of an accepted CAN frame exceeds the data field size configured for the matching Rx buffer or Rx FIFO, only the number of bytes as configured by the RXESC register are stored to the Rx buffer resp. Rx FIFO element. The data field of the rest of the frame is ignored.

Offset

Register	Offset
RXESC	BCh

Diagram



Fields

Field	Description
31-11 —	Reserved Read value is undefined, only zero should be written.
10-8 RBDS	Rx Buffer Data Field Size 000 - 8 byte data field 001 - 12 byte data field 010 - 16 byte data field

Table continues on the next page...

Table continued from the previous page...

Field	Description
	011 - 20 byte data field 100 - 24 byte data field 101 - 32 byte data field 110 - 48 byte data field 111 - 64 byte data field
7 —	Reserved Read value is undefined, only zero should be written.
6-4 F1DS	Rx FIFO 1 Data Field Size 000 - 8 byte data field 001 - 12 byte data field 010 - 16 byte data field 011 - 20 byte data field 100 - 24 byte data field 101 - 32 byte data field 110 - 48 byte data field 111 - 64 byte data field
3 —	Reserved Read value is undefined, only zero should be written.
2-0 F0DS	Rx FIFO 0 Data Field Size 000 - 8 byte data field 001 - 12 byte data field 010 - 16 byte data field 011 - 20 byte data field 100 - 24 byte data field 101 - 32 byte data field 110 - 48 byte data field 111 - 64 byte data field

43.6.1.1.31 Tx Buffer Configuration (TXBC)

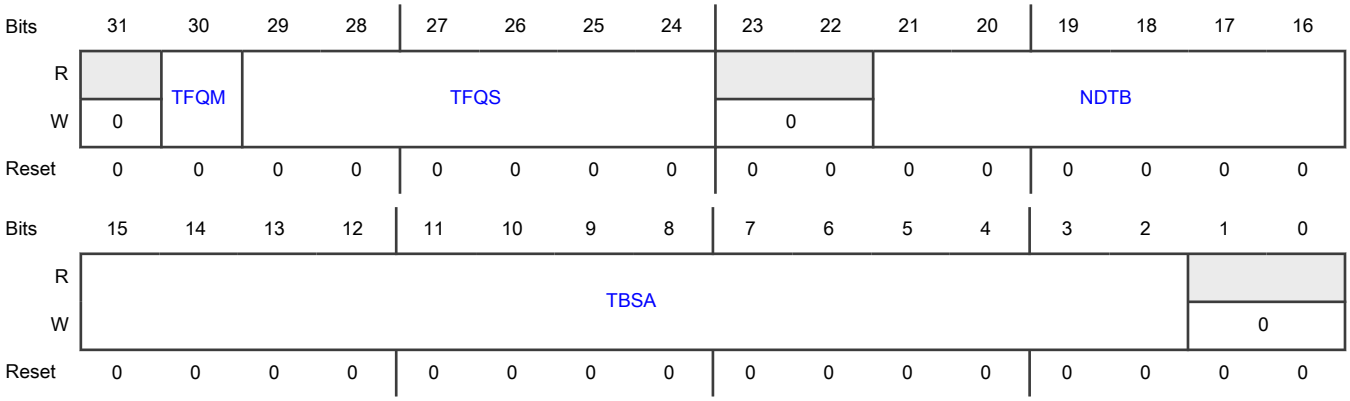
NOTE

The sum of TFQS and NDTB bits may not be greater than 32. There is no check for erroneous configurations. The Tx buffers section in the Message RAM starts with the dedicated Tx buffers.

Offset

Register	Offset
TXBC	C0h

Diagram



Fields

Field	Description
31 —	Reserved Read value is undefined, only zero should be written.
30 TFQM	Tx FIFO/Queue Mode 0 - Tx FIFO operation 1 - Tx queue operation
29-24 TFQS	Transmit FIFO/Queue Size • 0 = No Tx FIFO/Queue • 1 – 32 = Number of Tx buffers used for Tx FIFO/Queue • > 32 = Values greater than 32 are interpreted as 32
23-22 —	Reserved Read value is undefined, only zero should be written.
21-16 NDTB	Number of Dedicated Transmit Buffers • 0 = No dedicated Tx buffers • 1 – 32 = Number of dedicated Tx buffers • > 32 = Values greater than 32 are interpreted as 32
15-2 TBSA	Tx Buffers Start Address Start address of Tx buffers in message RAM.

Table continues on the next page...

Table continued from the previous page...

Field	Description
1-0 —	Reserved Read value is undefined, only zero should be written.

43.6.1.1.32 Tx FIFO/Queue Status (TXFQS)

The Tx FIFO/queue status is related to the pending Tx requests listed in the TXBRP register. Therefore the effect of add/cancellation requests may be delayed due to a running Tx scan.

NOTE

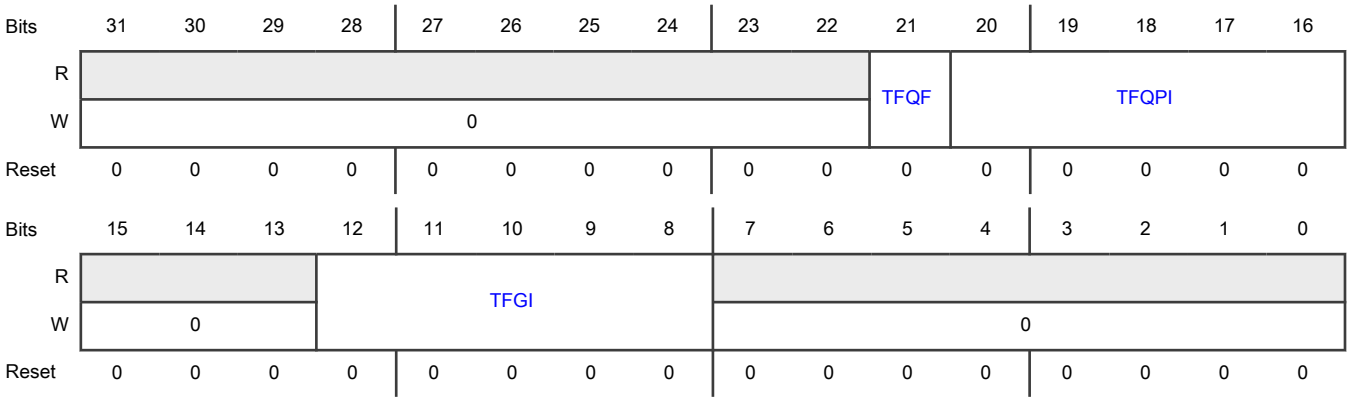
In case of mixed configurations where dedicated Tx buffers are combined with a Tx FIFO or a Tx queue, the put and get indices indicate the number of the Tx buffer starting with the first dedicated Tx buffers.

For example, a configuration of 12 dedicated Tx buffers and a Tx FIFO of 20 buffers a put index of 15 points to the fourth buffer of the Tx FIFO.

Offset

Register	Offset
TXFQS	C4h

Diagram



Fields

Field	Description
31-22 —	Reserved Read value is undefined, only zero should be written.
21 TFQF	Tx FIFO/Queue Full 0 - Tx FIFO/Queue not full 1 - Tx FIFO/Queue full

Table continues on the next page...

Table continued from the previous page...

Field	Description
20-16 TFQPI	Tx FIFO/Queue Put Index Tx FIFO/Queue write index pointer, range 0 to 31.
15-13 —	Reserved Read value is undefined, only zero should be written.
12-8 TFGI	Tx FIFO Get Index Tx FIFO read index pointer, range 0 to 31.
7-0 —	Reserved Read value is undefined, only zero should be written.

43.6.1.1.33 Tx Buffer Element Size Configuration (TXESC)

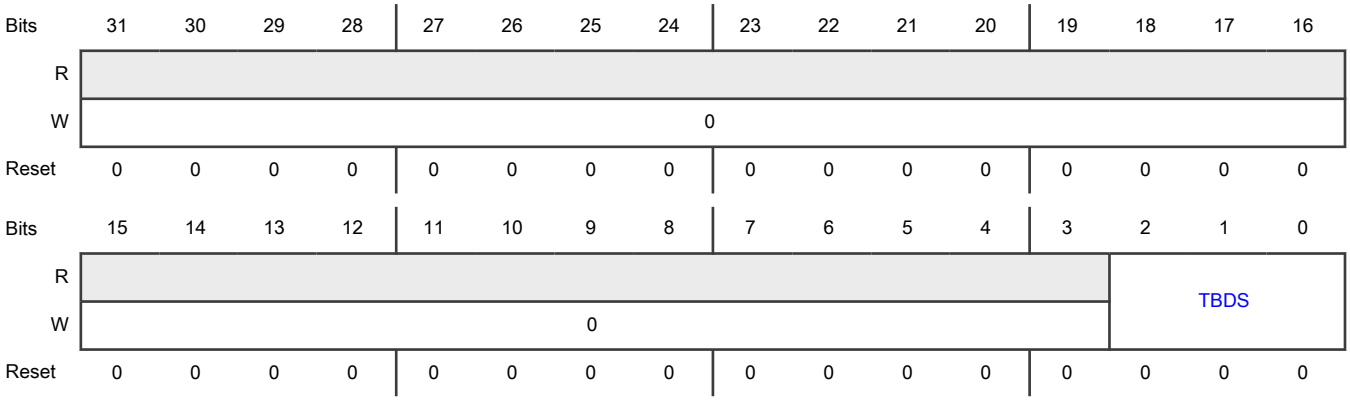
NOTE

In case the data length code of a Tx buffer element is configured to a value higher than the TBDS bits, the bytes not defined by the Tx buffer are transmitted as “0xCC” as padding.

Offset

Register	Offset
TXESC	C8h

Diagram



Fields

Field	Description
31-3	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	Read value is undefined, only zero should be written.
2-0 TBDS	Tx Buffer Data Field Size 000 - 8 byte data field 001 - 12 byte data field 010 - 16 byte data field 011 - 20 byte data field 100 - 24 byte data field 101 - 32 byte data field 110 - 48 byte data field 111 - 64 byte data field

43.6.1.1.34 Tx Buffer Request Pending (TXBRP)

Each Tx buffer has its own transmission request pending bit. The bits are set in the TXBAR register. The bits are reset after a requested transmission has completed or has been cancelled via the TXBCR register.

The TXBRP bits are set only for those Tx buffers configured in the TXBC register. After a TXBRP bit has been set, a Tx scan is started to check the pending Tx request with the highest priority.

A cancellation request resets the corresponding transmission request pending bit in the TXBRP register. In case a transmission has already been started when a cancellation is requested, it is done at the end of the transmission, regardless of the transmission success. The cancellation request bits are reset directly after the corresponding TXBRP bit has been reset.

After a cancellation has been requested, a finished cancellation is signaled using the TXBCF register:

- After a successful transmission together with the corresponding TXBTO bit.
- When a transmission has not yet been started at the point of cancellation.
- When the transmission has been aborted due to lost arbitration.
- When an error occurred during frame transmission.

In DAR mode, all transmissions are automatically cancelled if they are not successful. The corresponding TXBCF bit is set for all unsuccessful transmissions.

NOTE

TXBRP bits, which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx buffer, this add request is cancelled immediately and the corresponding TXBRP bit is reset.

Offset

Register	Offset
TXBRP	CCh

Diagram

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

Fields

Field	Description
31-0 TRP	Transmission Request Pending • 0 - No transmission request pending • 1 - Transmission request pending

43.6.1.1.35 Tx Buffer Add Request (TXBAR)

Each Tx buffer has its own add request bit. The number of Tx buffers is configured in the TXBC register. Writing a 1 will set the corresponding add request bit while writing a 0 has no impact. This allows for setting transmission requests for multiple Tx buffers with one write. The TXBAR bits are only set for those Tx buffers configured in the TXBC register. When no Tx scan is running, the bits are reset immediately, otherwise the bits remain set until the Tx scan process is completed.

NOTE

If an add request is applied for a Tx buffer with pending transmission request, the add request is ignored.

Offset

Register	Offset
TXBAR	D0h

Diagram

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

Fields

Field	Description
31-0 AR	Add Request • 0 - No transmission request added • 1 - Transmission request added

43.6.1.1.36 Tx Buffer Cancellation Request (TXBCR)

Each Tx buffer has its own add request bit. The number of Tx buffers is configured in the TXBC register. Writing a 1 will set the corresponding add request bit while writing a 0 has no impact. This allows for setting transmission requests for multiple Tx buffers with one write. The TXBCR bits are only set for those Tx buffers configured in the TXBC register. The bits remain set until the corresponding bit in the TXBRP register is reset.

Offset

Register	Offset
TXBCR	D4h

Diagram

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

Fields

Field	Description
31-0 CR	Cancellation Request • 0 - No cancellation pending • 1 - Cancellation pending

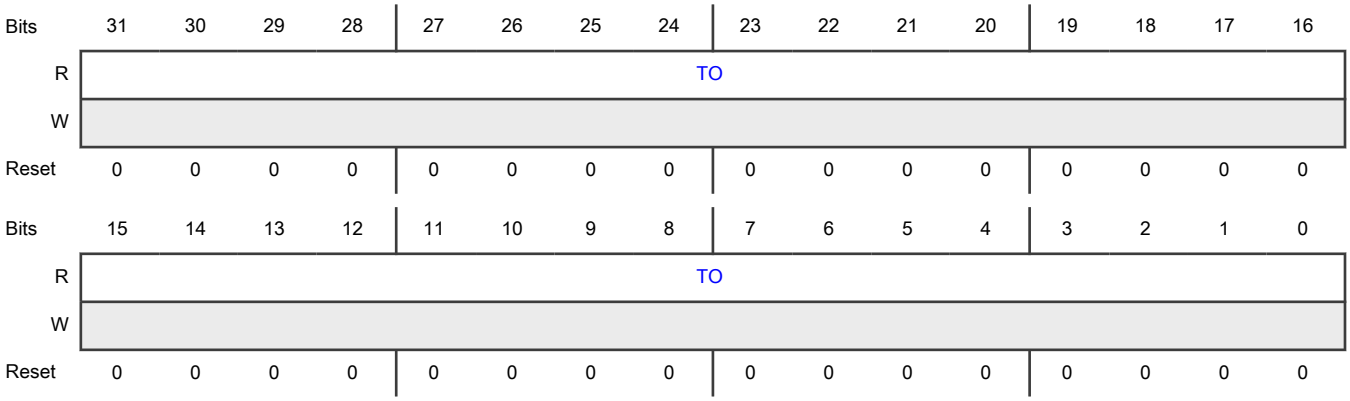
43.6.1.1.37 Tx Buffer Transmission Occurred (TXBTO)

Each Tx buffer has its own transmission occurred bit. The bits are set when the corresponding bits in the TXBRP register is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a 1 to the corresponding bit in the TXBAR register.

Offset

Register	Offset
TXBTO	D8h

Diagram



Fields

Field	Description
31-0 TO	Transmission Occurred • 0 - No transmission occurred • 1 - Transmission occurred

43.6.1.1.38 Tx Buffer Cancellation Finished (TXBCF)

Each Tx buffer has its own cancellation finished bit. The bits are set when the corresponding bits in the TXBRP register is cleared after a cancellation was requested in the TXBCR register. In case the corresponding bits in the TXBRP register was not set at the point of cancellation, the cancellation finished bits in this register will be set immediately. The bits are reset when a new transmit cancellation is requested by writing a 1 to the corresponding bit in the TXBAR register.

Offset

Register	Offset
TXBCF	DCh

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	0	0

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

Fields

Field	Description
31-0 TO	Cancellation Finished • 0 - No transmit buffer cancellation • 1 - Transmit buffer cancellation finished

43.6.1.1.39 Tx Buffer Transmission Interrupt Enable (TXBTIE)

Each Tx buffer has its own transmission interrupt enable bit.

Offset

Register	Offset
TXBTIE	E0h

Diagram

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

Fields

Field	Description
31-0 TIE	Transmission Interrupt Enable • 0 - Transmission interrupt disabled • 1 - Transmission interrupt enabled

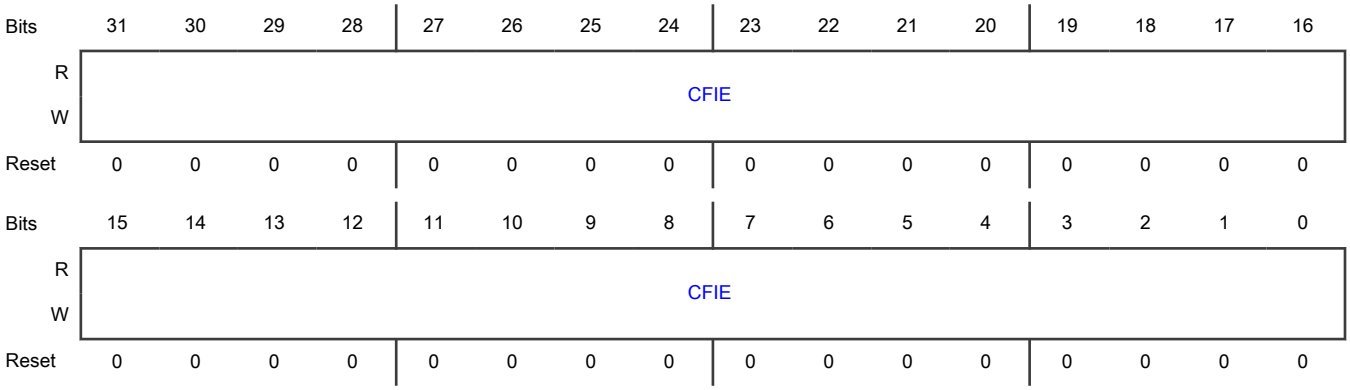
43.6.1.1.40 Tx Buffer Cancellation Finished Interrupt Enable (TXBCIE)

Each Tx buffer has its own transmission interrupt enable bit.

Offset

Register	Offset
TXBCIE	E4h

Diagram



Fields

Field	Description
31-0 CFIE	Cancellation Finished Interrupt Enable • 0 - Cancellation finished interrupt disabled • 1 - Cancellation finished interrupt enabled

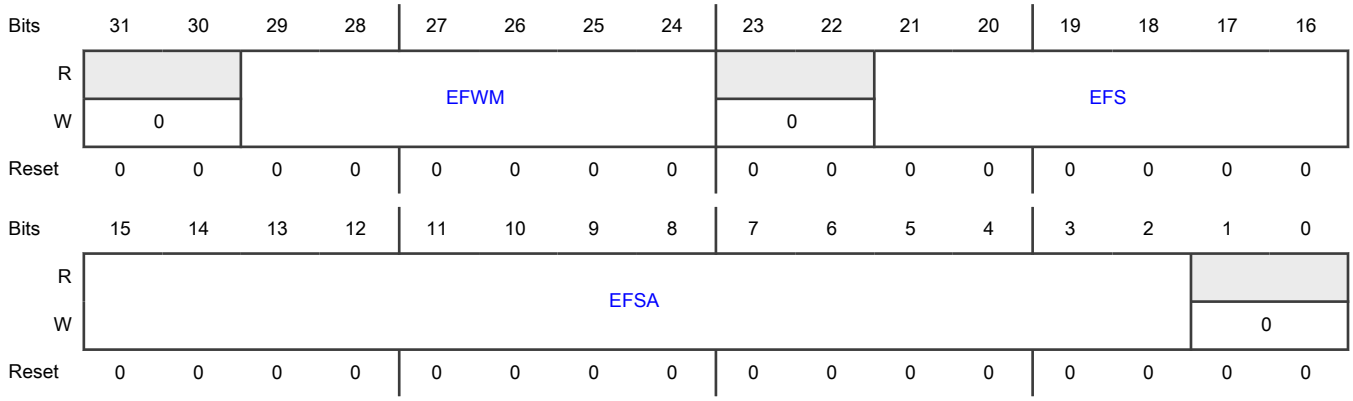
43.6.1.1.41 Tx Event FIFO Configuration (TXEFC)

The TXEFC register holds the values for the Tx Event FIFO configuration.

Offset

Register	Offset
TXEFC	F0h

Diagram



Fields

Field	Description
31-30 —	Reserved Read value is undefined, only zero should be written.
29-24 EFWM	Event FIFO Watermark • 0 = Watermark Interrupt disabled • 1 – 32 = Level for TEFW interrupt flag in the IR register • > 32 = Watermark interrupt disabled
23-22 —	Reserved Read value is undefined, only zero should be written.
21-16 EFS	Event FIFO Size • 0 = Tx event FIFO disabled. • 1 – 32 = Number of Tx event FIFO elements • > 32 = Values greater than 32 are interpreted as 32 The maximum Tx FIFO elements are the value set in this register minus 1.
15-2 EFSA	Event FIFO Start Address Start address of the Tx event FIFO in message RAM.
1-0 —	Reserved Read value is undefined, only zero should be written.

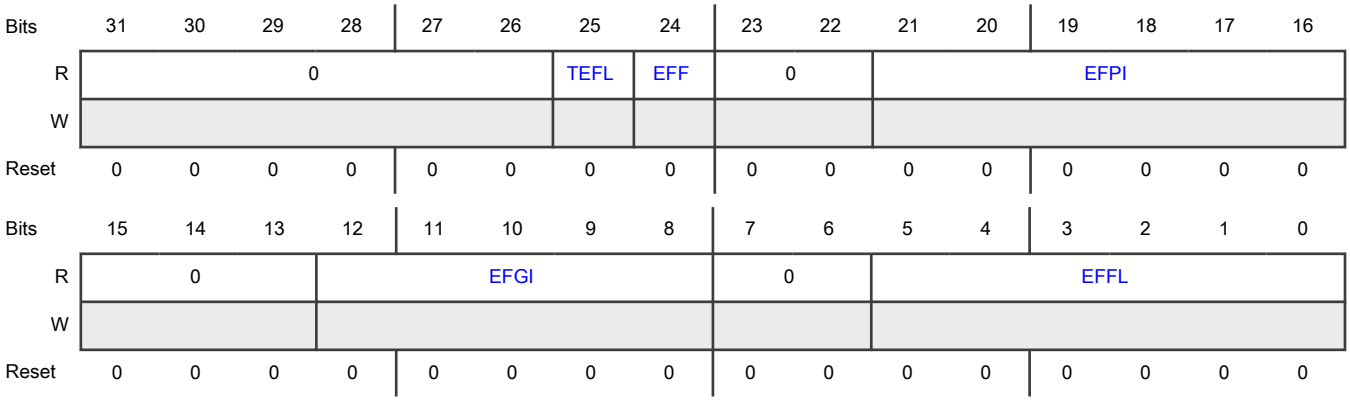
43.6.1.1.42 Tx Event FIFO Status (TXEFS)

The TXEFS register holds the values for the Tx Event FIFO status.

Offset

Register	Offset
TXEFS	F4h

Diagram



Fields

Field	Description
31-26 —	Reserved Read value is undefined, only zero should be written.
25 TEFL	Tx Event FIFO Element Lost This bit is a copy of the TEFL interrupt flag in the IR register. When the TEFL bit is reset, this bit is also reset. 0 - No Tx event FIFO element lost. 1 - Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size zero.
24 EFF	Event FIFO Full 0 - Tx event FIFO not full 1 - Tx event FIFO full
23-22 —	Reserved Read value is undefined, only zero should be written.
21-16 EFPI	Event FIFO Put Index Tx event FIFO write index pointer, range 0 to 31.
15-13 —	Reserved Read value is undefined, only zero should be written.
12-8	Event FIFO Get Index

Table continues on the next page...

Table continued from the previous page...

Field	Description
EFGI	Tx event FIFO read index pointer, range 0 to 31.
7-6 —	Reserved Read value is undefined, only zero should be written.
5-0 EFFL	Event FIFO Fill Level Number of elements stored in Tx event FIFO, range 0 to 32.

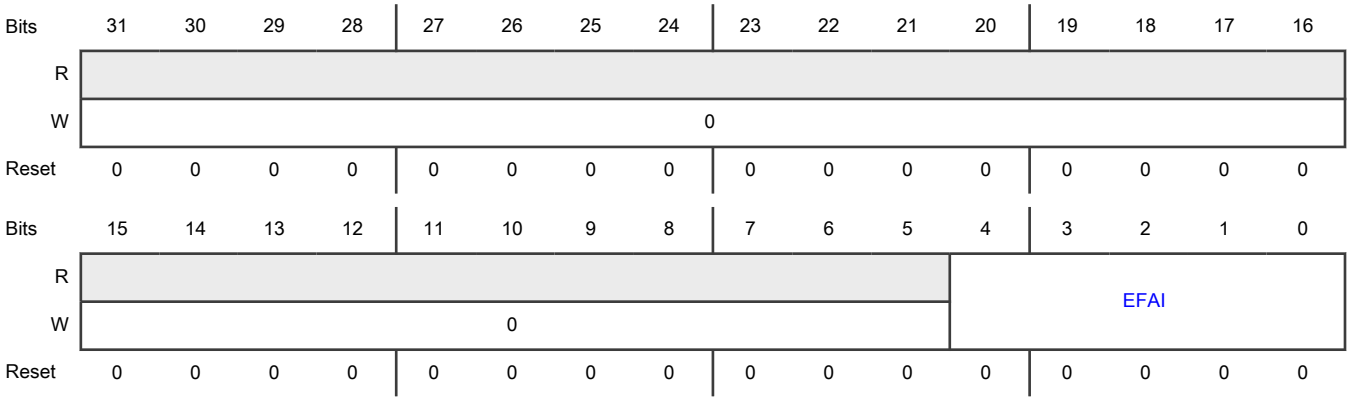
43.6.1.1.43 Tx Event FIFO Acknowledge (TXEFA)

After an element or sequence of elements have been read from the Tx event FIFO, the index of the last element read has to be written to the event FIFO acknowledge index bits in the EFAI bits. This will set the Tx event FIFO get index bits to the value stored in the EFAI +1 and update the event FIFO fill level.

Offset

Register	Offset
TXEFA	F8h

Diagram



Fields

Field	Description
31-5 —	Reserved Read value is undefined, only zero should be written.
4-0 EFAI	Event FIFO Acknowledge Index

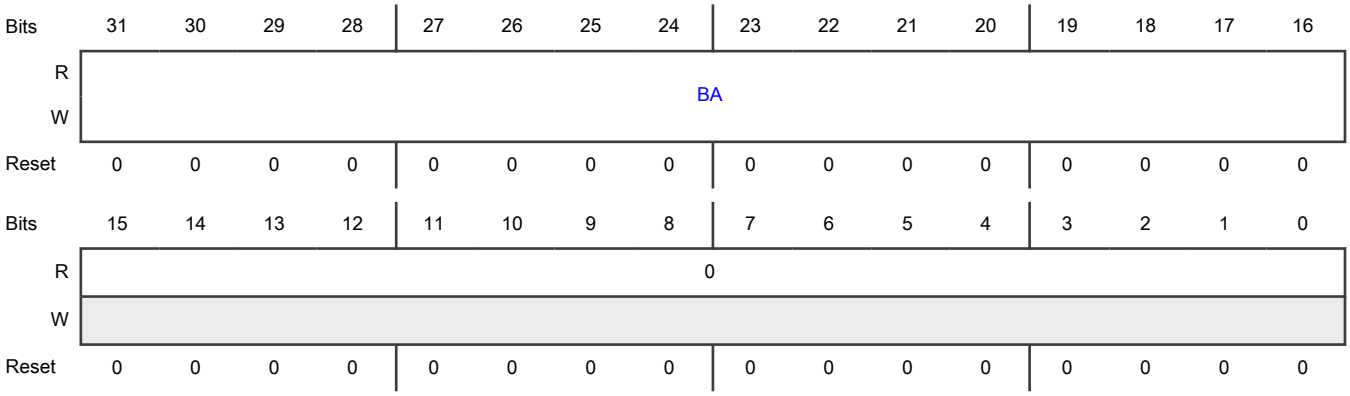
43.6.1.1.44 Message RAM Base Address (MRBA)

Base address for the message RAM in the chip memory map. Bits 0 to 15 are reserved and 0x0000 should be written to these bits.

Offset

Register	Offset
MRBA	200h

Diagram



Fields

Field	Description
31-16 BA	Base Address for the message RAM in the chip memory map.
15-0 —	Reserved Read value is undefined, only zero should be written.

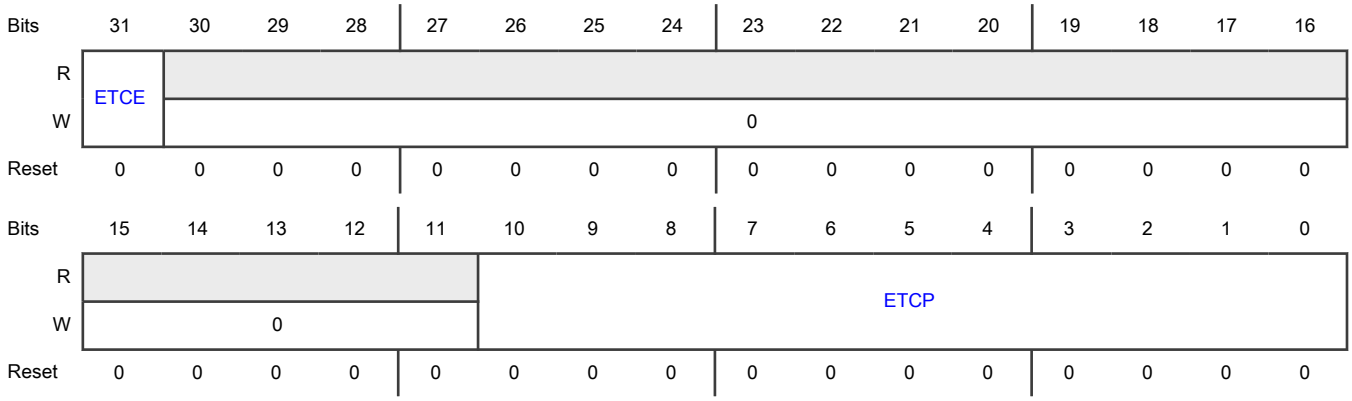
43.6.1.1.45 External Timestamp Counter Configuration (ETSCC)

The ETSCC register contains the External Timestamp Prescaler Value and External Timestamp Counter Enable fields.

Offset

Register	Offset
ETSCC	400h

Diagram



Fields

Field	Description
31 ETCE	External Timestamp Counter Enable 0 - External timestamp counter is disabled 1 - External timestamp counter is enabled
30-11 —	Reserved Read value is undefined, only zero should be written.
10-0 ETCP	External Timestamp Prescaler Value The CPUCLK is divided down by the prescaler value plus 1 to clock the external timestamp counter.

43.6.1.1.46 External Timestamp Counter Value (ETSCV)

The ETSCV register holds the current External Timestamp Counter Value.

Offset

Register	Offset
ETSCV	600h

Diagram

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

Fields

Field	Description
31-16 —	Reserved Read value is undefined, only zero should be written.
15-0 ETSC	External Timestamp Counter Read to return current counter value. Write to initialize the counter with the specified 16-bit value.

43.6.2 Message RAM

Any general purpose SRAM can be used for storage of Rx/Tx messages and for storage of the filter configuration. The base address of the SRAM used for messages is determined by the value in the MRBA register that is configurable by the application.

43.6.2.1 Message RAM configuration

The message RAM has a width of 32 bits. The MCAN module can be configured to allocate up to 4352 words in the message RAM. It is not necessary to configure each section and there is no restriction with respect to the sequence of the sections.

When operated in CAN FD mode, the required message RAM size depends on the element size configured for the Rx FIFO0, Rx FIFO1, Rx buffers, and Tx buffers via the RXESC and TXESC registers.

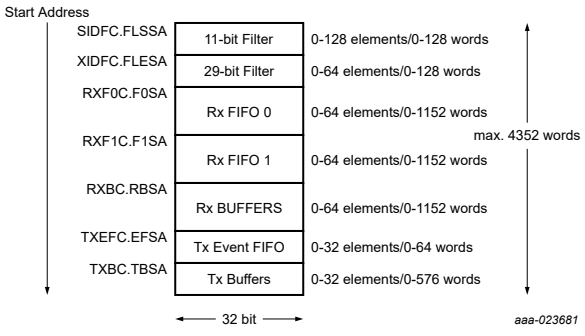


Figure 197. Message RAM configuration

When the MCAN module addresses the message RAM, it addresses 32-bit words, rather than single bytes. The configurable start addresses are 32-bit word addresses, where bits 15 to 2 are evaluated and the two least significant bits are ignored.

Note: MCAN does not check for erroneous configuration of the message RAM. The configuration of the start addresses of different sections, and the number of elements of each section, must be specified precisely to avoid falsification or loss of data.

43.6.3 Rx buffer and FIFO element

Up to 64 Rx buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of an Rx buffer/FIFO element is shown in Figure 198. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via the RXESC register.

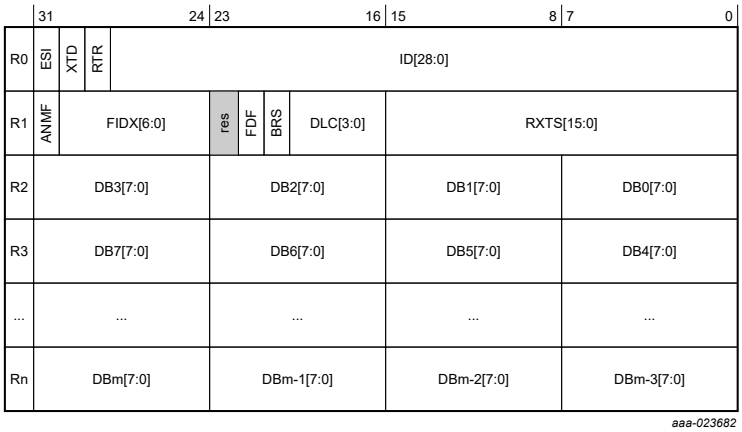


Figure 198. Rx buffer and FIFO element

Table 365. R0 description

Bit	Symbol	Value	Description
31	ESI		Error state identifier.
		0	Transmitting node is error active
		1	Transmitting node is error passive
30	XTD		Extended identifier.
		0	11-bit standard identifier
		1	29-bit extended identifier
29	RTR		Remote transmission request. This bit is set depending on whether the received frame is a data frame or a remote frame. **NOTE** There are no remote frames in CAN FD format.
		0	Received frame is a data frame
		1	Received frame is a remote frame
28:0	ID	-	Identifier. Standard or extended identifier depending on the XTD bit. A standard identifier is expected to be stored into ID bits 28:18.

Table 366. R1 description

Bit	Symbol	Value	Description
31	ANMF		Accepted non-matching frame. Acceptance of non-matching frames may be enabled via the ANFS and ANFE bits in the GFC register.
		0	Received frame matching filter index FIDX
		1	Received frame did not match any Rx filter element
30:24	FIDX	-	Filter index. 0 – 127 = Index of matching Rx acceptance filter element (invalid if the ANMF bit is set) Range is 0 to one minus the LSS bits in the SIDFC register resp. the LSE bit in the XIDFC register
23:22	-	-	Reserved.
21	FDF		FD format.
		0	Standard frame format.
		1	CAN FD frame format (new DLC-coding and CRC).
20	BRS		Bit rate switch.
		0	Frame received without bit rate switching
		1	Frame received with bit rate switching
19:16	DLC	-	Data length code. 0 – 8 = CAN + CAN FD: received frame has 0 - 8 data bytes 9 – 15 = CAN: received frame has 8 data bytes 9 – 15 = CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
15:0	RXTS	-	Rx timestamp. Timestamp counter value captures on start of frame reception. Resolution depending on configuration of timestamp counter prescaler bit in the TSCC register.

Table 367. R2 description

Bit	Symbol	Description
31:24	DB	Data byte
23:16	DB	Data byte
15:8	DB	Data byte
7:0	DB	Data byte

NOTE

Depending on the configuration of the element size in the RXESC register, between two and sixteen 32-bit words are used for storage of a CAN message's data field.

43.6.4 Tx buffer element

The Tx buffers section can be configured to hold dedicated Tx buffers as well as a Tx FIFO/Tx Queue. In case that the Tx buffers section is shared by dedicated Tx buffers and a Tx FIFO/Tx Queue, the dedicated Tx buffers start at the beginning of the Tx buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx buffers and Tx FIFO/Tx Queue by evaluating the TFQS and NDTB bits in the TXBC register. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via the TXESC register.

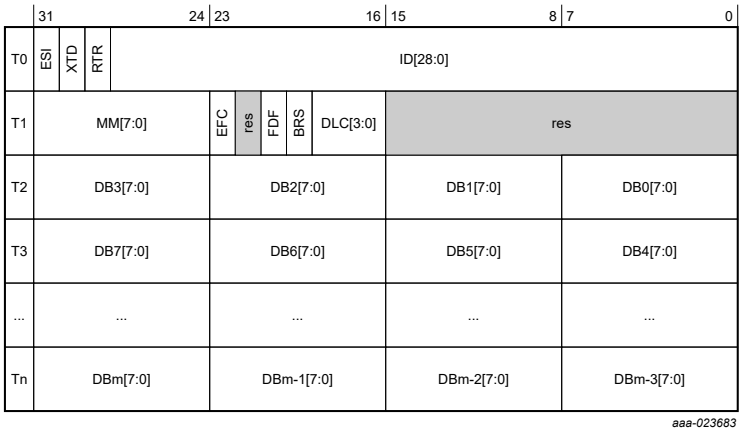


Figure 199. Tx buffer element

Table 368. T0 description

Bit	Symbol	Value	Description
31	ESI		Error state identifier **NOTE** The ESI bit of the transmit buffer is OR'd with the passive flag to decide the value of the ESI bit in the transmitted FD frame. Per the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node will always transmit ESI bit recessive.
		0	ESI bit in CAN FD format depends only on error passive flag
		1	ESI bit in CAN FD format transmitted recessive
30	XTD		Extended identifier
		0	11-bit standard identifier
		1	29-bit extended identifier
29	RTR		Remote transmission request

Table continues on the next page...

Table 368. T0 description (continued)

Bit	Symbol	Value	Description
			NOTE When this bit is set, the MCAN transmits a remote frame according to ISO11898-1, even if the FDOE bit of the CCCR register enables the transmission in CAN FD format.
		0	Transmit data frame
		1	Transmit remote frame
28:0	ID	-	Identifier. Standard or extended identifier depending on the XTD bit. A standard identifier is expected to be stored into ID bits 28:18.

Table 369. T1 description

Bit	Symbol	Value	Description
31:24	MM	-	Message Marker. Written during Tx buffer configuration. Copied into Tx event FIFO element for identification of Tx message status.
23	EFC		Event FIFO control
		0	Do not store Tx events
		1	Store Tx events
22	-		Reserved
21	FDF		FD format
		0	Frame transmitted in classic CAN format
		1	Frame transmitted in CAN FD format
20	BRS		Bit rate switch
		0	CAN FD frames transmitted without bit rate switching
		1	CAN FD frames transmitted with bit rate switching
19:16	DLC	-	Data length code. 0 – 8 = CAN + CAN FD: transmit frame has 0 - 8 data bytes 9 – 15 = CAN: transmit frame has 8 data bytes 9 – 15 = CAN FD: transmit frame has 12/16/20/24/32/48/64 data bytes
15:0	-	-	Reserved

Table 370. R2 to Rn description

Bit	Symbol	Description
31:24	DB	Data byte
23:16	DB	Data byte
15:8	DB	Data byte
7:0	DB	Data byte

NOTE

Depending on the configuration of the element size in the RXESC register, between two and sixteen 32-bit words are used to store the data field of a CAN message.

43.6.5 Tx event FIFO element

Each element stores information about transmitted messages that can be used to get information regarding the order of transmitted messages. Status information about the Tx event FIFO can be obtained from the TXEFS register.

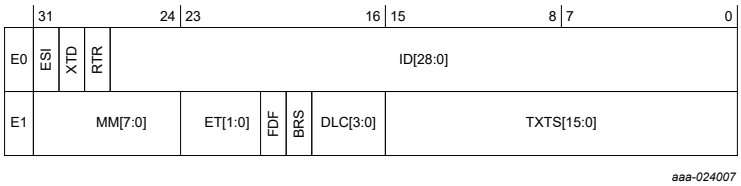


Figure 200. Tx event FIFO element

Table 371. E0 description

Bit	Symbol	Value	Description
31	ESI		Error state identifier
		0	Transmitted node is error active
		1	Transmitted node is error passive
30	XTD		Extended identifier
		0	11-bit standard identifier
		1	29-bit extended identifier
29	RTR		Remote transmission request
		0	Data frame transmitted
		1	Remote frame transmitted
28:0	ID	-	Identifier. Standard or extended identifier depending on the XTD bit. A standard identifier is expected to be stored into ID bits 28:18.

Table 372. E1 description

Bit	Symbol	Value	Description
31:24	MM	-	Message Marker. Copied from Tx buffer into Tx event FIFO element for identification of Tx message status.
23:22	ET		Event type
		0x0	Reserved
		0x1	Tx event
		0x2	Transmission in spite of cancellation (always set for transmission in DAR mode)
		0x3	Reserved
21	FDF		FD Format
		0	Standard frame format
		1	CAN FD frame format (new DLC-coding and CRC)
20	BRS		Bit rate switch
		0	Frames transmitted without bit rate switching
		1	Frames transmitted with bit rate switching
19:16	DLC	-	Data length code 0 – 8 = CAN + CAN FD: frame with 0 - 8 data bytes transmitted 9 – 15 = CAN: frame with 8 data bytes transmitted 9 – 15 = CAN FD: frame with 12/16/20/24/32/48/64 data bytes transmitted
15:0	TXTS	-	Tx timestamp. Timestamp counter value captured on start of frame transmission. Resolution depending on configuration of the TCP bit in the TSCC register.

43.6.6 Standard message ID filter element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a standard message ID filter element, its address is equal to the filter list standard start address bit field in the SIDFC register plus the index of the filter element.

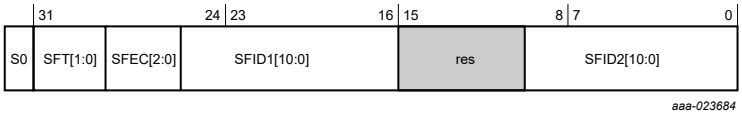


Figure 201. Standard message ID filter element

Table 373. S0 description

Bit	Symbol	Value	Description
31:30	SFT		Standard filter type
		0x0	Range filter from SFID1 to SFID2 (SFID2 ≥ SFID1)
		0x1	Dual ID filter for SFID1 or SFID2
		0x2	Classic filter: SFID1 = filter, SFID2 = mask
		0x3	Filter element disabled
29:27	SFEC		Standard filter element configuration. All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If this bit field is set to 0x4, 0x5, or 0x6, a match sets the HPM interrupt flag in the IR register, if enabled, an interrupt is generated. In this case, the HPMS register is updated with the status of the priority match.
		0x0	Disable filter element
		0x1	Store in Rx FIFO 0 if filter matches
		0x2	Store in Rx FIFO 1 if filter matches
		0x3	Reject ID if filter matches
		0x4	Set priority if filter matches
		0x5	Set priority and store in FIFO 0 if filter matches
		0x6	Set priority and store in FIFO 1 if filter matches
		0x7	Store into Rx buffer or as debug message, configuration of the SFT bit field is ignored.
26:16	SFID1	-	Standard filter ID 1. First ID of standard ID filter element. When filtering for Rx buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.
15:11	-	-	Reserved
10:0	SFID2	-	Standard filter ID 2. This bit field has a different meaning depending on the configuration of the SFEC register. 1. SFEC = 0x1 – 0x6, this bit field becomes the second ID of standard ID filter element. 2. SFEC = 0x7, this bit field is used to filter for Rx buffers or for debug messages. Bits 10:9 decides whether the received message is stored into an Rx buffer or treated as message A, B, or C of the debug message sequence. 0x0 = Store message into an Rx buffer 0x1 = Debug message A

Table continues on the next page...

Table 373. S0 description (continued)

Bit	Symbol	Value	Description
			0x2 = Debug message B 0x3 = Debug message C Bits 8:6 are used to control the filter event pins at the extension interface. A 1 at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one MCAN function clock period in case the filter matches. Bits 5:0 define the offset to the Rx buffer start address bit field in the RXBC register for storage of a matching message.

43.6.7 Extended message ID filter element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an extended message ID filter element, its address is equal to the filter list extended start address bit field in the XIDFC register plus two times the index of the filter element.

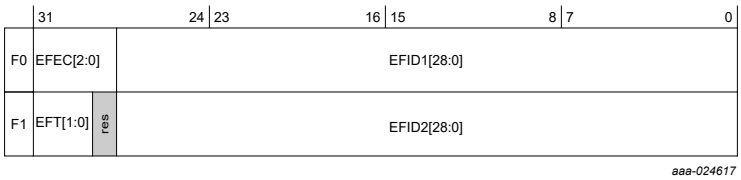


Figure 202. Extended message ID filter element

Table 374. F0 description

Bit	Symbol	Value	Description
31:29	EFEC		Extended filter element configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If this bit field is equal to 0x4, 0x5, or 0x6, a match sets the HPM interrupt flag in the IR register and, if enabled, an interrupt is generated. In this case, the HPMS register is updated with the status of the priority match.
		0x0	Disable filter element
		0x1	Store in Rx FIFO 0 if filter matches
		0x2	Store in Rx FIFO 1 if filter matches
		0x3	Reject ID if filter matches
		0x4	Set priority if filter matches
		0x5	Set priority and store in FIFO 0 if filter matches
		0x6	Set priority and store in FIFO 1 if filter matches
		0x7	Store into Rx buffer or as debug message, configuration of the EFT bit field is ignored

Table continues on the next page...

Table 374. F0 description (continued)

Bit	Symbol	Value	Description
28:0	EFID1	-	Extended filter ID 1. First ID of extended ID filter element. When filtering for Rx buffers or for debug messages, this field defines the ID of an extended message to be stored. The received identifiers must match exactly when masked with the XIDAM register.

Table 375. F1 description

Bit	Symbol	Value	Description
31:30	EFT		Extended filter type
		0x0	Range filter from EFID1 to EFID2 (EFID2 ≥ EFID1)
		0x1	Dual ID filter for EFID1 or EFID2
		0x2	Classic filter: EFID1 = filter, EFID2 = mask
		0x3	Range filter from EFID1 to EFID2 (EFID2 ≥ EFID1), the mask in the XIDAM register is not applied.
29	-	-	Reserved
28:0	EFID2	-	Extended filter ID 2 This bit field has a different meaning depending on the configuration of the EFEC bit field. 1. EFEC = 0x1 – 0x6, this bit field becomes the second ID of extended ID filter element. 2. EFEC = 0x7, this bit field is used to filter for Rx buffers or debug messages. Bits 10:9 decides whether the received message is stored into an Rx buffer or treated as message A, B, or C of the debug message sequence. 0x0 = Store message into an Rx buffer 0x1 = Debug message A 0x2 = Debug message B 0x3 = Debug message C Bits 8:6 is used to control the filter event pins at the extension interface. A 1 at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one MCAN function clock period in case the filter matches. Bits 5:0 defines the offset to the Rx buffer start address bit field in the RXBC register for storage of a matching message.

Chapter 44
Enhanced Flex Pulse Width Modulator (eFlexPWM)

44.1 Chip-specific PWM information

Table 376. Reference links to related information

Topic	Related module	Reference
Full description	PWM	PWM
System memory map		System memory map
Clocking		Clock distribution
Internal signal connections	INPUTMUX	INPUTMUX
Signal multiplexing	Port control	Signal multiplexing

44.1.1 Module instances

This device has two instances of the PWM module, PWM0 and PWM1.

NOTE

In this device, PWMx_EXTB is not applicable. Also, there is no NanoEdge placement block.

44.2 Overview

The pulse width modulator (PWM) module contains PWM submodules, each of which is set up to control a single half-bridge power stage. Fault channel support is provided.

This PWM module can generate various switching patterns, including highly sophisticated waveforms. It is ideal for controlling different Switched Mode Power Supplies (SMPS) topologies.

44.2.1 Block Diagram

The following figure is a block diagram of the PWM.

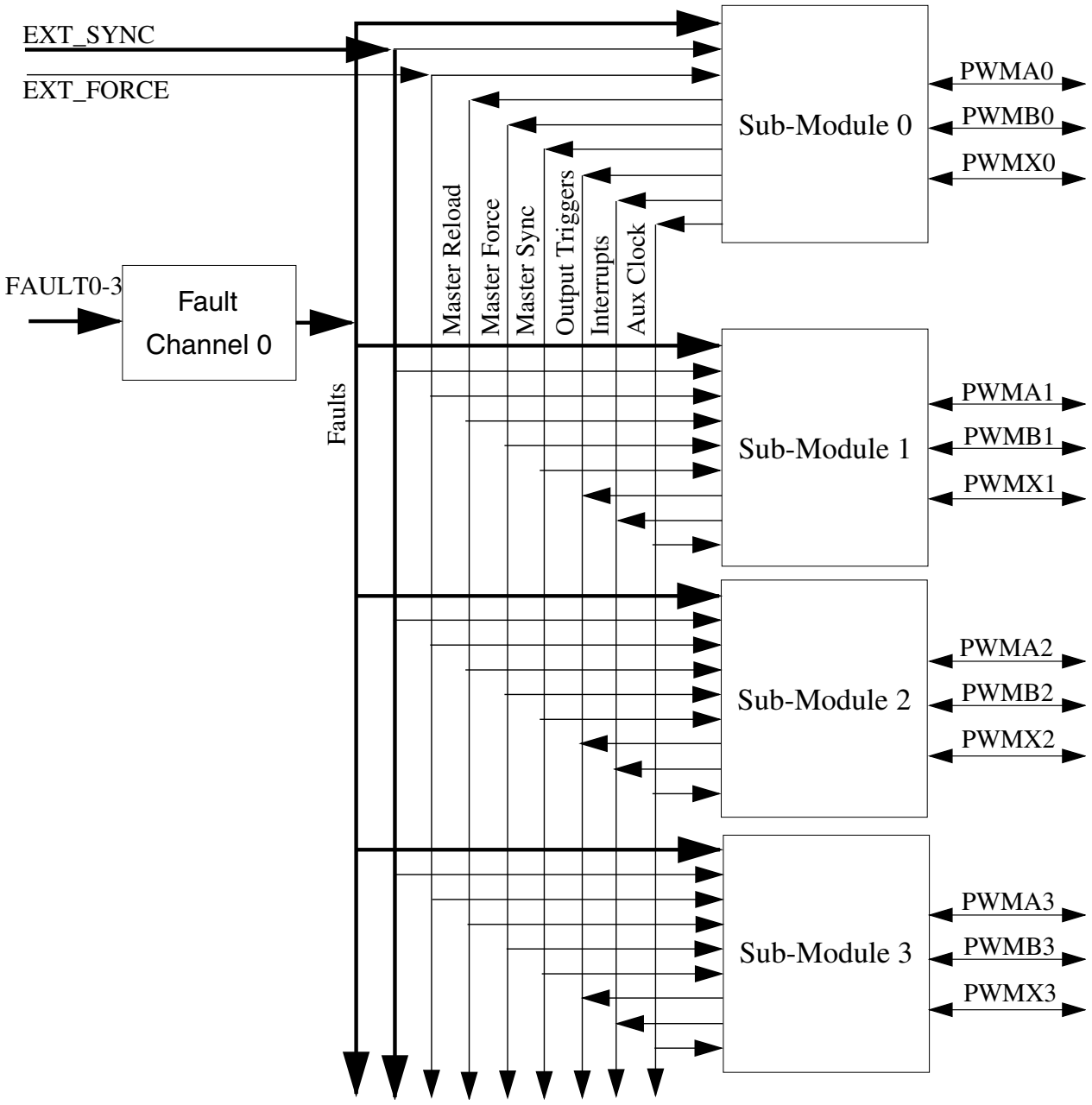


Figure 203. PWM Block Diagram

44.2.1.1 PWM Submodule

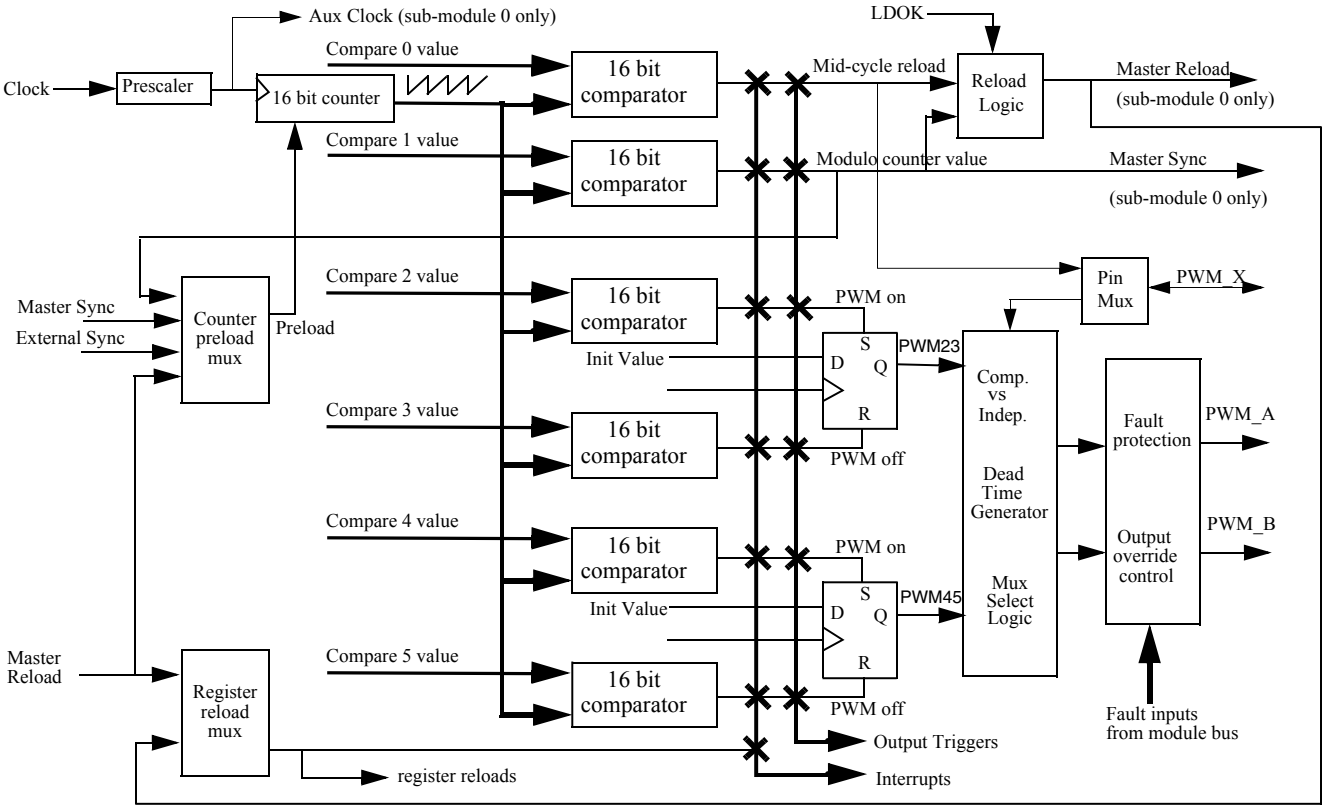


Figure 204. PWM Submodule Block Diagram

44.2.2 Features

- 16 bits of resolution for center, edge-aligned, and asymmetrical PWMs
- Dithering to simulate enhanced resolution when fine edge placement is not available
- PWM outputs that can operate as complementary pairs or independent channels
- Ability to accept signed numbers for PWM generation
- Independent control of both edges of each PWM output
- Support for synchronization to external hardware or other PWM
- Double buffered PWM registers
  - Integral reload rates from 1 to 16
  - Half cycle reload capability
- Multiple output trigger events can be generated per PWM cycle via hardware
- Support for double switching PWM outputs
- Fault inputs can be assigned to control multiple PWM outputs
- Programmable filters for fault inputs
- Independently programmable PWM output polarity
- Independent top and bottom deadtime insertion
- Each complementary pair can operate with its own PWM frequency and deadtime values

- Individual software control for each PWM output
- All outputs can be programmed to change simultaneously via a FORCE_OUT event
- PWM_X pin can optionally output a third PWM signal from each submodule
- Channels not used for PWM generation can be used for buffered output compare functions
- Channels not used for PWM generation can be used for input capture functions
- Enhanced dual edge capture functionality

44.2.3 Modes of operation

Be careful when using this module in Deep Sleep, Sleep, and Debug operating modes.

CAUTION

Some applications require regular software updates for proper operation. Failure to provide regular software updates could result in destroying the hardware setup.

To accommodate this situation, PWM outputs are placed in their inactive states in Deep Sleep mode, and they can optionally be placed in inactive states in Sleep and Debug modes. PWM outputs are reactivated (assuming they were active beforehand) when these modes are exited.

Table 377. Modes when PWM Operation is Restricted

Mode	Description
Deep Sleep	PWM outputs are inactive.
Sleep	PWM outputs are driven or inactive as a function of CTRL2[WAITEN].
Debug	PWM outputs are driven or inactive as a function of CTRL2[DBGEN].

44.3 Functional Description

44.3.1 PWM Capabilities

This section describes some capabilities of the PWM module.

44.3.1.1 Center Aligned PWMs

Each submodule has its own timer that is capable of generating PWM signals on two output pins. The edges of each of these signals are controlled independently as shown in Figure 205.

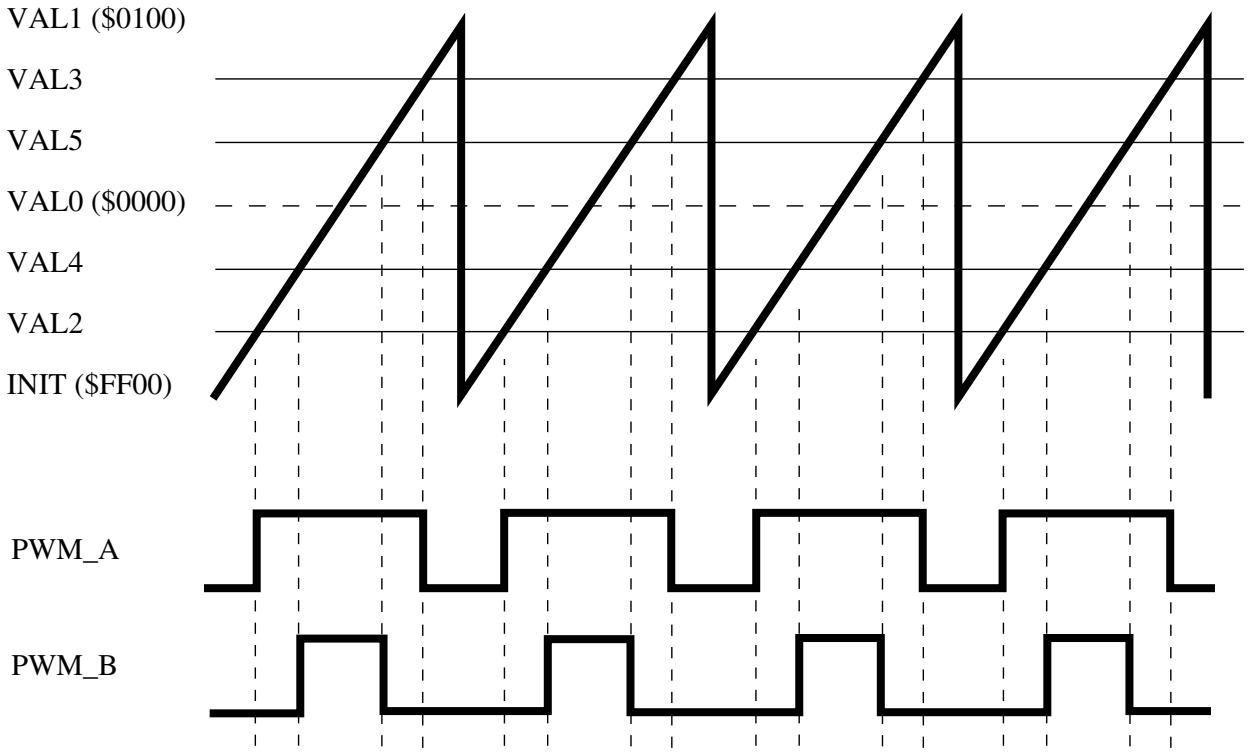


Figure 205. Center Aligned Example

The submodule timers only count in the up direction and then reset to the INIT value. Instead of having a single value that determines pulse width, there are two values that must be specified: the turn on edge and the turn off edge. This double action edge generation not only gives the user control over the pulse width, but over the relative alignment of the signal as well. As a result, there is no need to support separate PWM alignment modes since the PWM alignment mode is inherently a function of the turn on and turn off edge values.

Figure 205 also illustrates an additional enhancement to the PWM generation process. When the counter resets, it is reloaded with a user specified value, which may or may not be zero. If the value chosen happens to be the 2's complement of the modulus value, then the PWM generator operates in "signed" mode. This means that if each PWM's turn on and turn off edge values are also the same number but only different in their sign, the "on" portion of the output signal will be centered around a count value of zero. Therefore, only one PWM value needs to be calculated in software and then this value and its negative are provided to the submodule as the turn off and turn on edges respectively. This technique will result in a pulse width that always consists of an odd number of timer counts. If all PWM signal edge calculations follow this same convention, then the signals will be center aligned with respect to each other, which is the goal. Of course, center alignment between the signals is not restricted to symmetry around the zero count value, as any other number would also work. However, centering on zero provides the greatest range in signed mode and also simplifies the calculations.

44.3.1.2 Edge Aligned PWMs

When the turn on edge for each pulse is specified to be the INIT value, then edge aligned operation results, as the following figure shows. Therefore, only the turn off edge value needs to be periodically updated to change the pulse width.

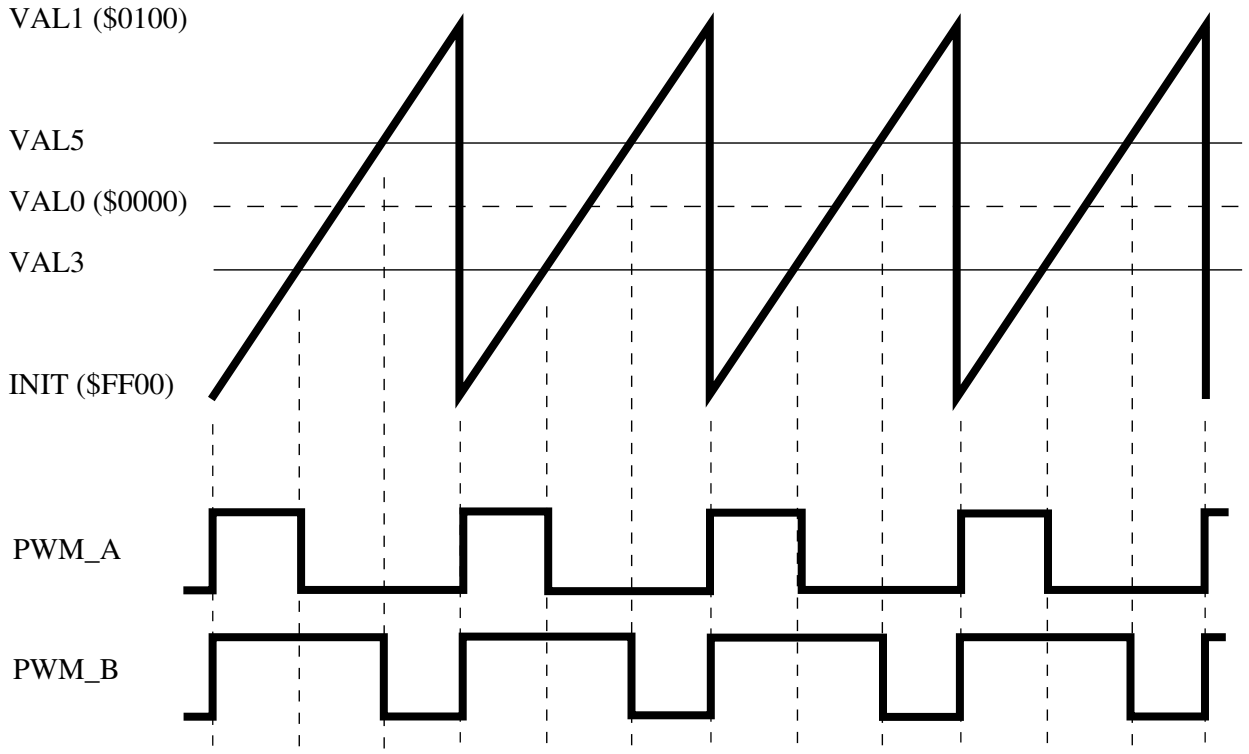


Figure 206. Edge Aligned Example (INIT=VAL2=VAL4)

With edge aligned PWMs, another example of the benefits of signed mode can be seen. A common way to drive an H-bridge is to use a technique called "bipolar" PWMs where a 50% duty cycle results in zero volts on the load. Duty cycles less than 50% result in negative load voltages and duty cycles greater than 50% generate positive load voltages. If the module is set to signed mode operation (the INIT and VAL1 values are the same number with opposite signs), then there is a direct proportionality between the PWM turn off edge value and the motor inverter voltage, INCLUDING the sign. So once again, signed mode of operation simplifies the software interface to the PWM module since no offset calculations are required to translate the output variable control algorithm to the voltage on an H-Bridge load.

44.3.1.3 Phase Shifted PWMs

In the previous sections, the benefits of signed mode of operation were discussed in the context of simplifying the required software calculations by eliminating the requirement to bias up signed variables before applying them to the module. However, if numerical biases are applied to the turn on and turn off edges of different PWM signal, the signals will be phase shifted with respect to each other, as the following figure shows. This results in certain advantages when applied to a power stage. For example, when operating a multi-phase inverter at a low modulation index, all of the PWM switching edges from the different phases occur at nearly the same time. This can be troublesome from a noise standpoint, especially if ADC readings of the inverter must be scheduled near those times. Phase shifting the PWM signals can open up timing windows between the switching edges to allow a signal to be sampled by the ADC. However, phase shifting does NOT affect the duty cycle so average load voltage is not affected.

If the outputs of submodules 1-3 need to be delayed from the output of submodule 0 (and from each other), instead of just creating a phase delay by adding an offset to the turn on and turn off times of the different submodules, another method is to use the PHASEDLY registers for submodules 1-3 to indicate their delay from the submodule 0 timing. This method can be used when the master sync signal from submodule 0 is selected as the initialization source (CTRL2[INIT_SEL]==b10). This method allows all of the submodules to be programmed with the same turn on and turn off time but submodules 1-3 can still be delayed the time from submodule 0.

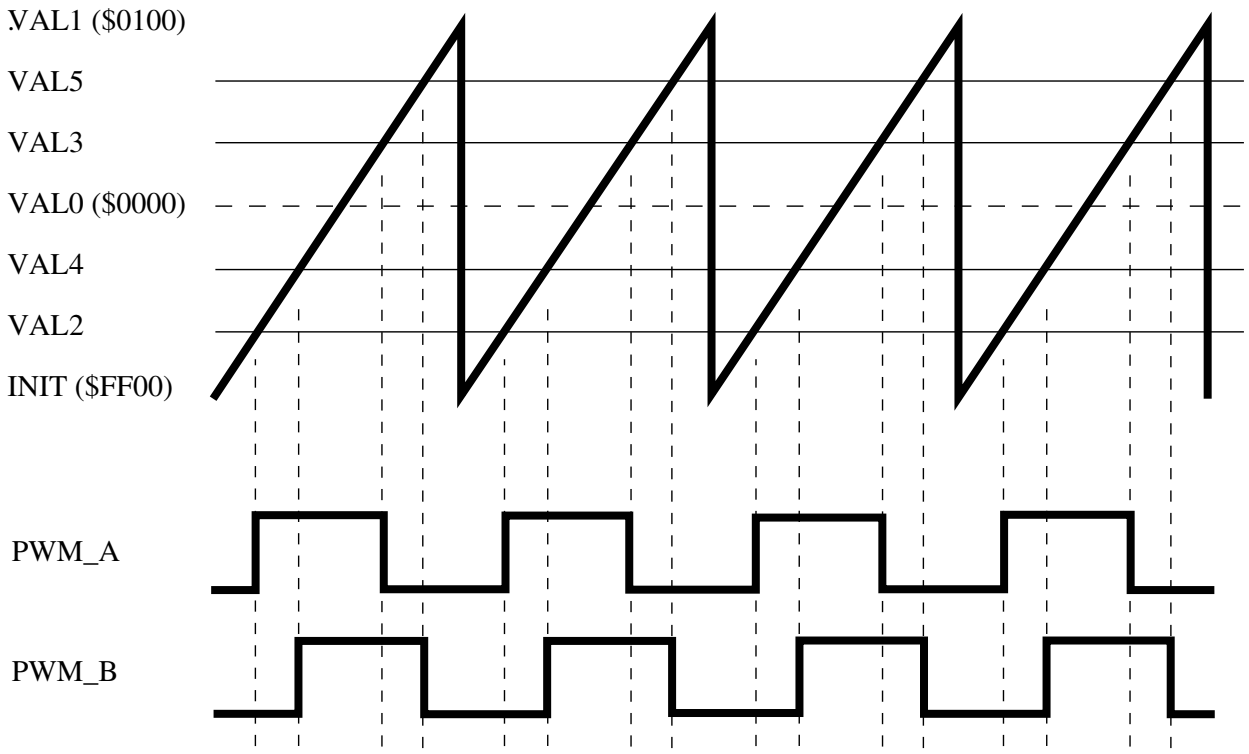


Figure 207. Phase Shifted Outputs Example

An additional benefit of phase shifted PWMs appears in Figure 208. In this case, an H-Bridge circuit is driven by 4 PWM signals to control the voltage waveform on the primary of a transformer. Both left and right side PWMs are configured to always generate a square wave with 50% duty cycle. This works out nicely for the H-Bridge since no narrow pulse widths are generated reducing the high frequency switching requirements of the transistors. Notice that the square wave on the right side of the H-Bridge is phase shifted compared to the left side of the H-Bridge. As a result, the transformer primary sees the bottom waveform across its terminals. The RMS value of this waveform is directly controlled by the amount of phase shift of the square waves. Regardless of the phase shift, no DC component appears in the load voltage as long as the duty cycle of each square wave remains at 50% making this technique ideally suited for transformer loads. As a result, this topology is frequently used in industrial welders to adjust the amount of energy delivered to the weld arc.

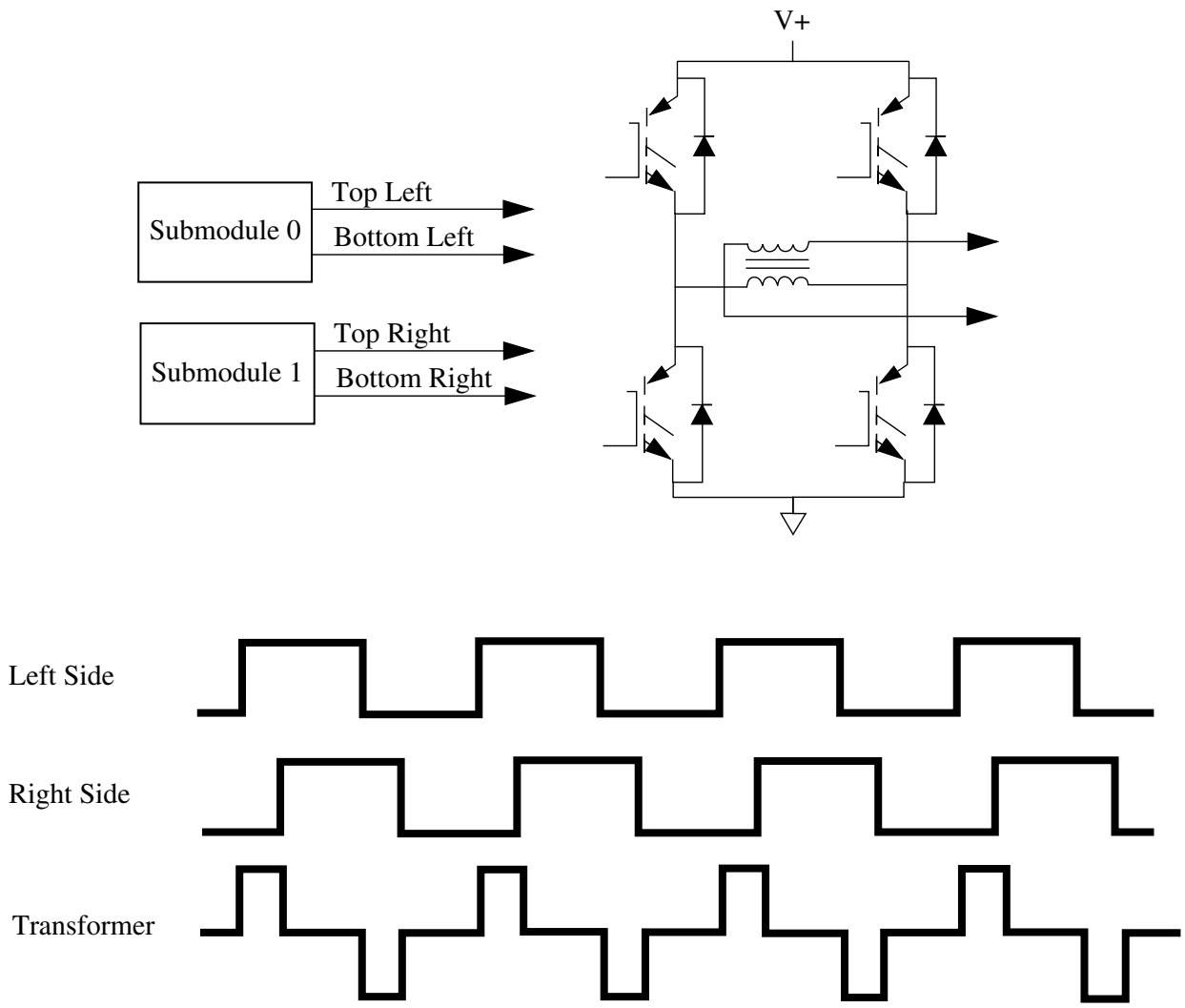


Figure 208. Phase Shifted PWMs Applied to a Transformer Primary

44.3.1.4 Double Switching PWMs

Double switching PWM output is supported to aid in single shunt current measurement and three phase reconstruction. This method support two independent rising edges and two independent falling edges per PWM cycle. The VAL2 and VAL3 registers are used to generate the even channel (labelled as PWM_A in the figure) while VAL4 and VAL5 are used to generate the odd channel. The two channels are combined using XOR logic (force out logic) as the following figure shows. The DBLPWM signal can be run through the deadtime insertion logic.

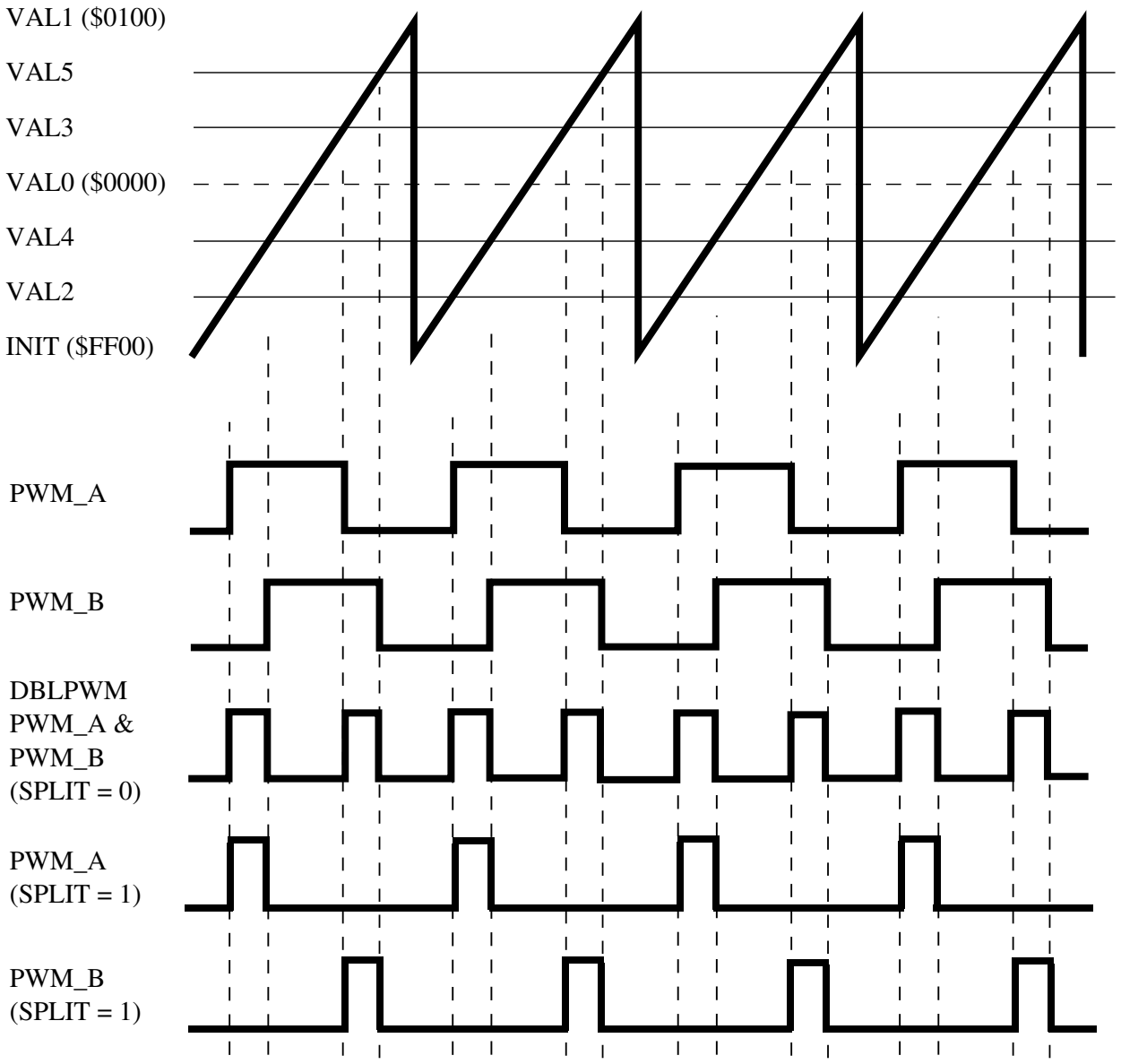


Figure 209. Double Switching Output Example

44.3.1.5 ADC Triggering

In cases where the timing of the ADC triggering is critical, it must be scheduled as a hardware event instead of software activated. With this PWM module, multiple ADC triggers can be generated in hardware per PWM cycle without the requirement of another timer module. Figure 210 shows how this is accomplished. When specifying complementary mode of operation, only two edge comparators are required to generate the output PWM signals for a given submodule. This means that the other comparators are free to perform other functions. In this example, the software does not need to quickly respond after the first conversion to set up other conversions that must occur in the same PWM cycle.

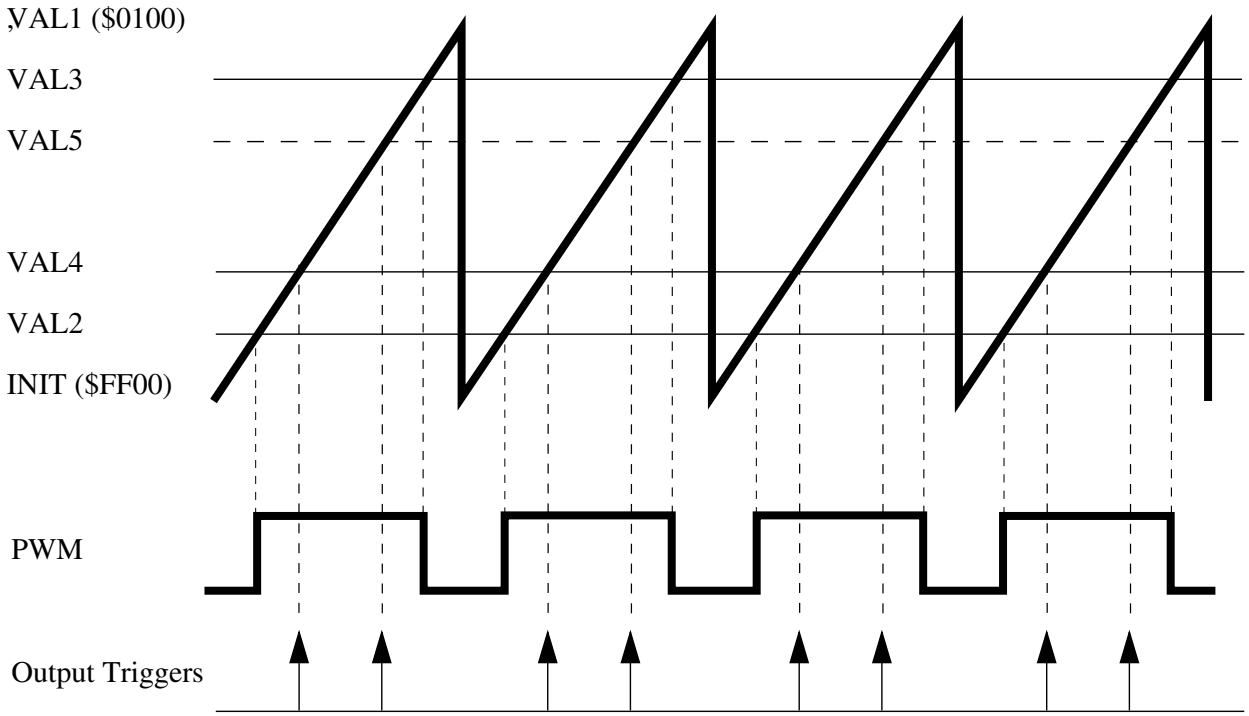


Figure 210. Multiple Output Trigger Generation in Hardware

Because each submodule has its own timer, it is possible for each submodule to run at a different frequency. One of the options possible with this PWM module is to have one or more submodules running at a lower frequency, but still synchronized to the timer in submodule0. Figure 211 shows how this feature can be used to schedule ADC triggers over multiple PWM cycles. A suggested use for this configuration would be to use the lower-frequency submodule to control the sampling frequency of the software control algorithm where multiple ADC triggers can now be scheduled over the entire sampling period. In Figure 211, *all* submodule comparators are shown being used for ADC trigger generation.

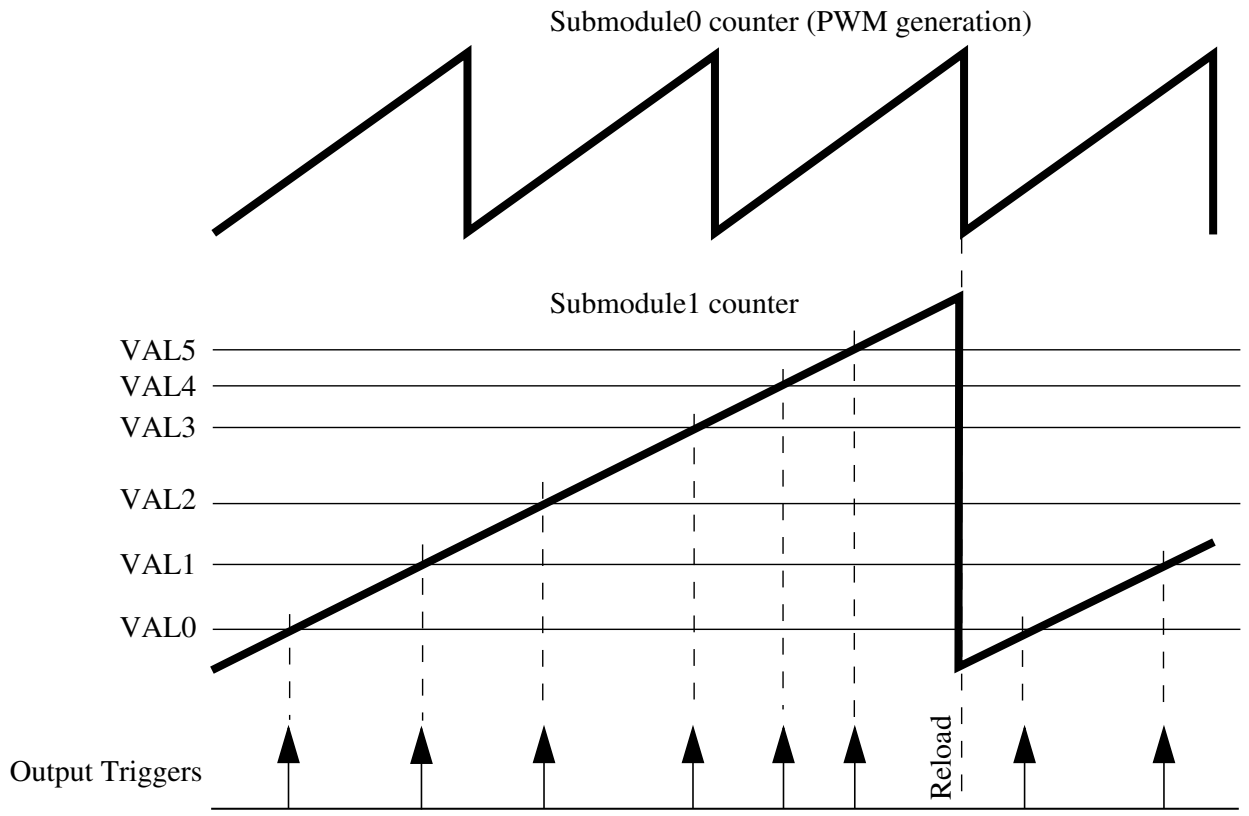


Figure 211. Multiple Output Triggers Over Several PWM Cycles

44.3.1.6 Enhanced Capture Capabilities (E-Capture)

When a PWM pin is not being used for PWM generation, it can be used to perform input captures. Recall that for PWM generation BOTH edges of the PWM signal are specified via separate compare register values. When programmed for input capture, both of these registers work on the same pin to capture multiple edges, toggling from one to the other in either a free running or one-shot fashion. By simply programming the desired edge of each capture circuit, period and pulse width of an input signal can easily be measured without the requirement to re-arm the circuit. In addition, each edge of the input signal can clock an 8 bit counter where the counter output is compared to a user specified value (EDGCMP). When the counter output equals EDGCMP, the value of the submodule timer is captured and the counter is automatically reset. This feature allows the module to count a specified number of edge events and then perform a capture and interrupt. The following figure illustrates some of the functionality of the E-Capture circuit.

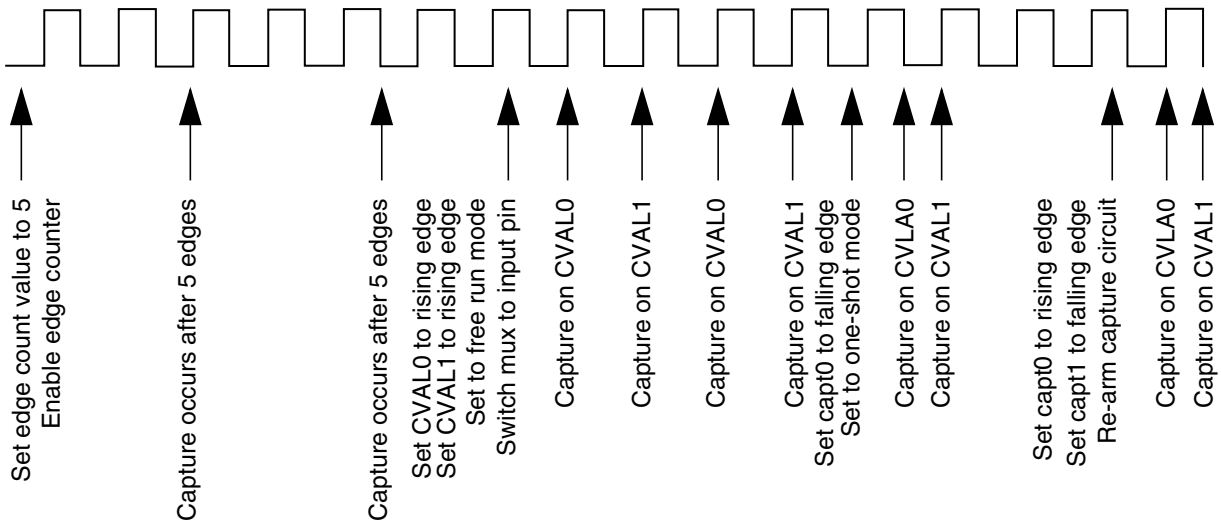


Figure 212. Capture Capabilities of the E-Capture Circuit

When a submodule is being used for PWM generation, its timer counts up to the modulus value used to specify the PWM frequency and then is re-initialized. Therefore, using this timer for input captures on one of the other pins (for example, PWM_X) has limited utility since it does not count through all of the numbers and the timer reset represents a discontinuity in the 16 bit number range. However, when measuring a signal that is synchronous to the PWM frequency, the timer modulus range is perfectly suited for the application. Consider the following figure as an example. In this application the output of a PWM power stage is connected to the PWM_X pin that is configured for free running input captures. Specifically, the CVAL0 capture circuitry is programmed for rising edges and the CVAL1 capture circuitry is set for falling edges. This will result in new load pulse width data being acquired every PWM cycle. To calculate the pulse width, simply subtract the CVAL0 register value from the CVAL1 register value. This measurement is extremely beneficial when performing dead-time distortion correction on a half bridge circuit driving an inductive load. Also, these values can be directly compared to the VALx registers responsible for generating the PWM outputs to obtain a measurement of system propagation delays. For details, refer to the separate discussion of deadtime distortion correction.

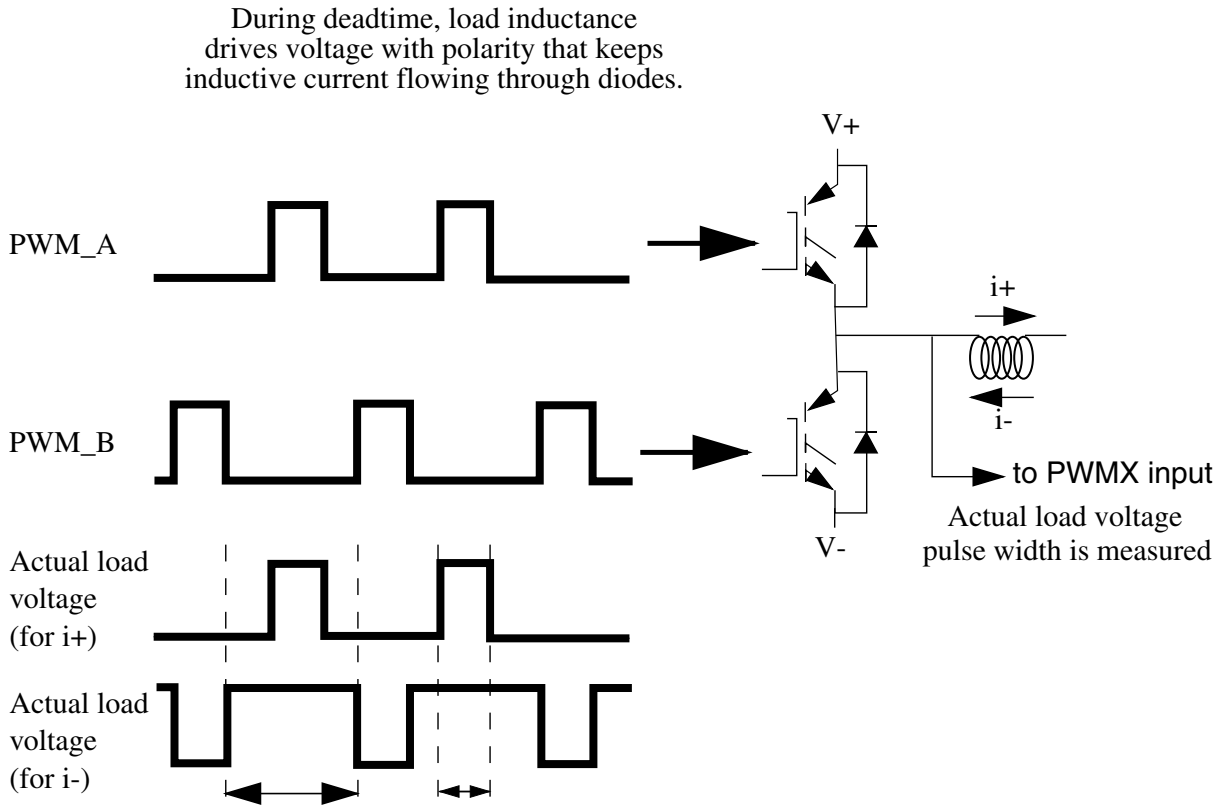


Figure 213. Output Pulse Width Measurement Possible with the E-Capture Circuit

44.3.1.7 Synchronous Switching of Multiple Outputs

Before the PWM signals are routed to the output pins, they are processed by a hardware block that permits all submodule outputs to be switched synchronously. This feature can be extremely useful in commutated motor applications where the next commutation state can be laid in ahead of time and then immediately switched to the outputs when the appropriate condition or time is reached. Not only do all the changes occur synchronously on all submodule outputs, but they occur IMMEDIATELY after the trigger event occurs eliminating any interrupt latency.

The synchronous output switching is accomplished via a signal called FORCE_OUT. This signal originates from the local FORCE bit within the submodule, from submodule0, or from external to the PWM module and, in most cases, is supplied from an external timer channel configured for output compare. In a typical application, software sets up the desired states of the output pins in preparation for the next FORCE_OUT event. This selection lays dormant until the FORCE_OUT signal transitions and then all outputs are switched simultaneously. The signal switching is performed upstream from the deadtime generator so that any abrupt changes that might occur do not violate deadtime on the power stage when in complementary mode.

Figure 214 shows a popular application that can benefit from this feature. On a brushless DC motor it is desirable on many cases to spin the motor without need of hall-effect sensor feedback. Instead, the back EMF of the motor phases is monitored and this information is used to schedule the next commutation event. The top waveforms of Figure 214 are a simplistic representation of these back EMF signals. Timer compare events (represented by the long vertical lines in the diagram) are scheduled based on the zero crossings of the back-EMF waveforms. The PWM module is configured via software ahead of time with the next state of the PWM pins in anticipation of the compare event. When it happens, the output compare of the timer drives the FORCE_OUT signal which immediately changes the state of the PWM pins to the next commutation state with no software latency.

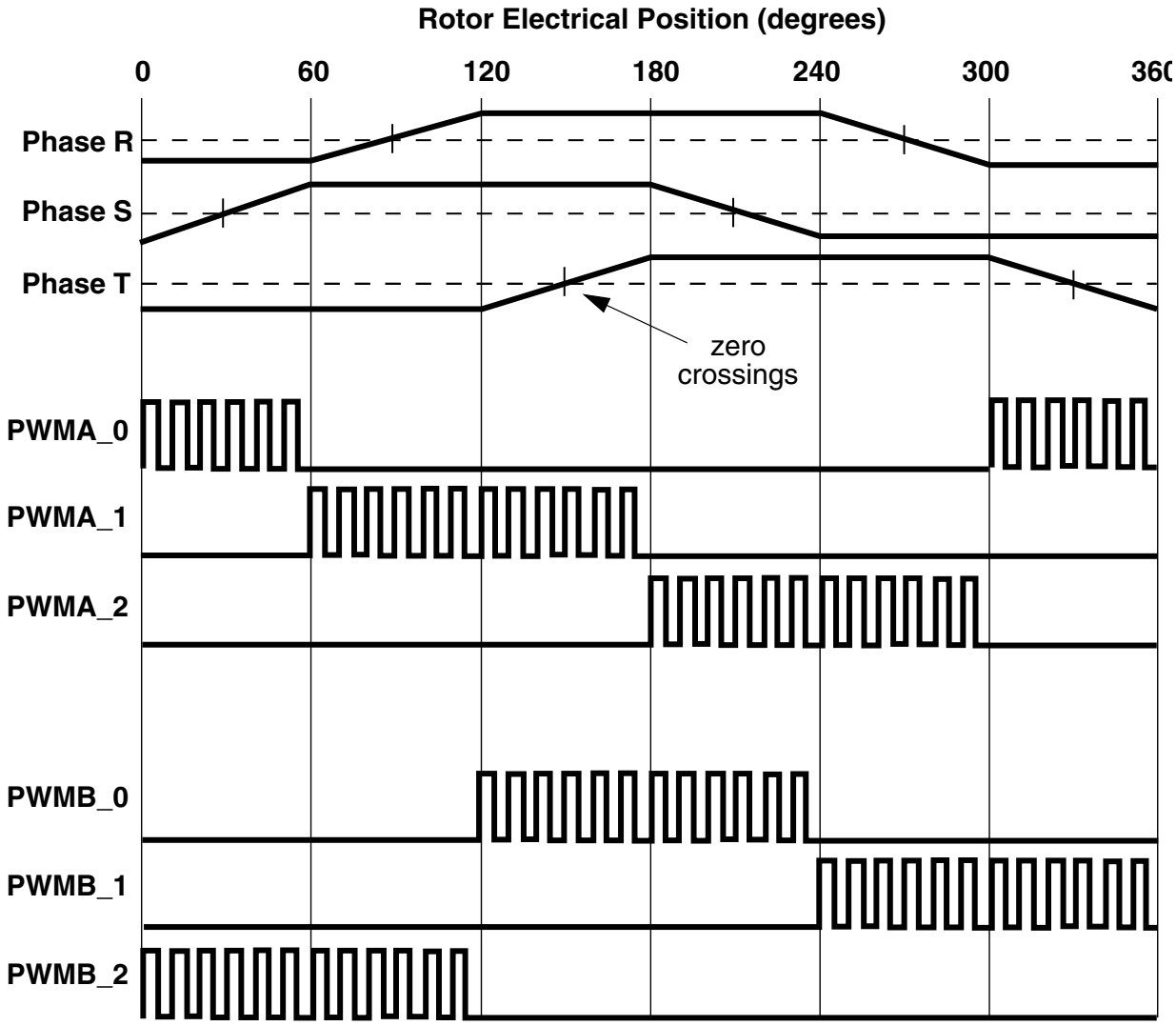


Figure 214. Sensorless BLDC Commutation Using the Force Out Function

44.3.2 Functional Details

This section describes the implementation of various sections of the PWM in greater detail.

The following figure is a high-level block diagram of output PWM generation.

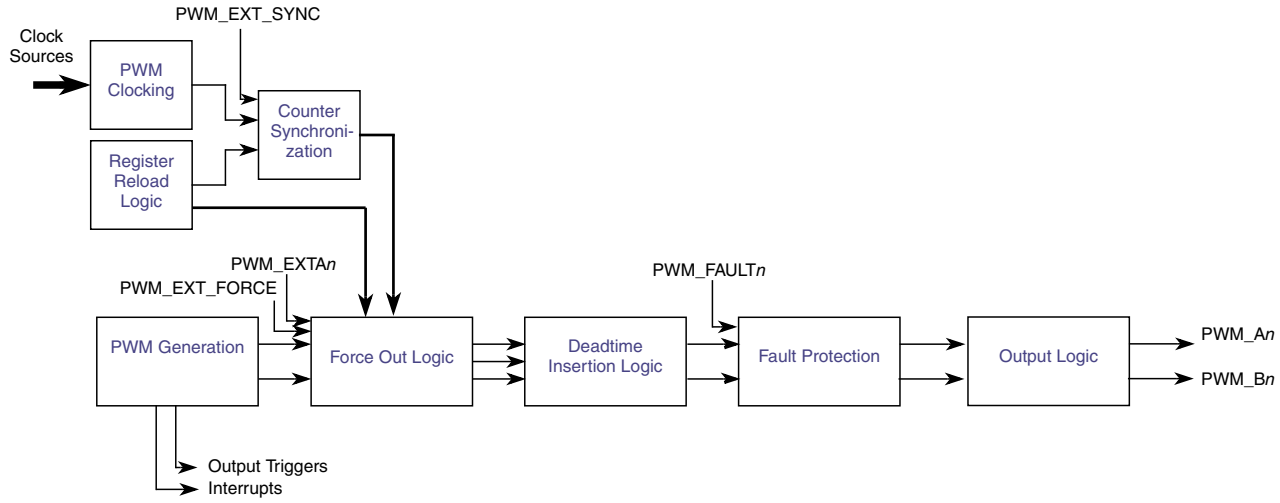


Figure 215. High-Level Output PWM Generation Block Diagram

44.3.2.1 PWM Clocking

Figure 216 shows the logic used to generate the main counter clock. Each submodule can select between three clock signals: the IPBus clock, EXT_CLK, and AUX_CLK. The EXT_CLK goes to all of the submodules. The AUX_CLK signal is broadcast from submodule0 and can be selected as the clock source by other submodules so that the 8-bit prescaler and MCTRL[RUN] from submodule0 can control all of the submodules.

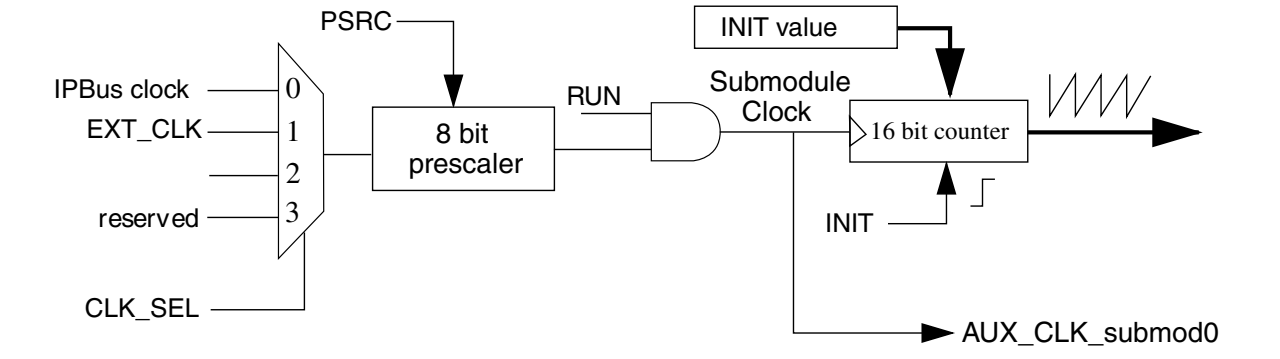


Figure 216. Clocking Block Diagram for PWM Submodule 0

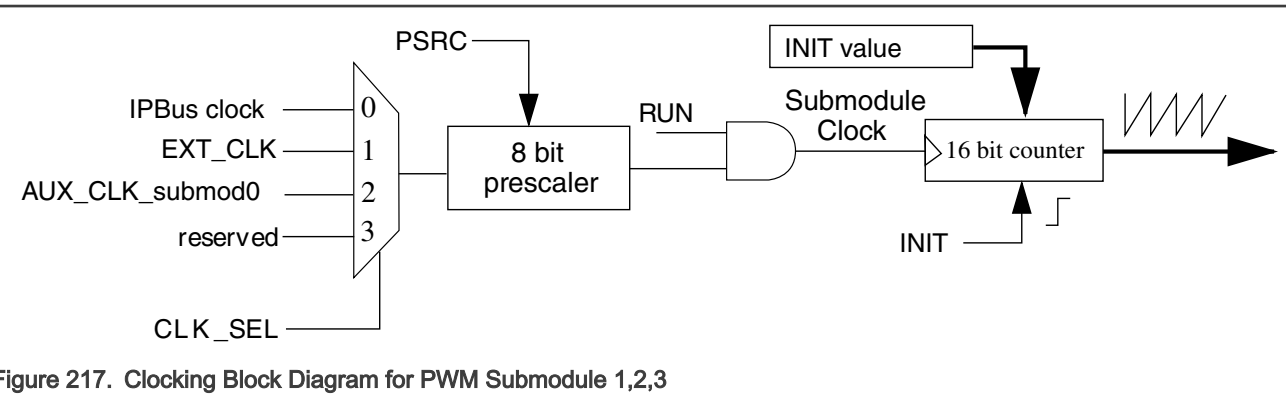


Figure 217. Clocking Block Diagram for PWM Submodule 1,2,3

To permit lower PWM frequencies, the prescaler produces the PWM clock frequency by dividing the IPBus clock frequency by 1-128. The prescaler bits, CTRL[PRSC], select the prescaler divisor. This prescaler is buffered and will not be used by the PWM generator until MCTRL[LDOK] is set and a new PWM reload cycle begins or CTRL[LDMOD] is set.

44.3.2.2 Register Reload Logic

The register reload logic is used to determine when the outer set of registers for all double buffered register pairs will be transferred to the inner set of registers. The register reload event can be scheduled to occur every "n" PWM cycles using CTRL[LDFQ] and CTRL[FULL]. A half cycle reload option is also supported (CTRL[HALF]) where the reload can take place in the middle of a PWM cycle. The half cycle point is defined by the VAL0 register and does not have to be exactly in the middle of the PWM cycle.

As illustrated in Figure 218 the reload signal from submodule0 can be broadcast as the Master Reload signal allowing the reload logic from submodule0 to control the reload of registers in other submodules.

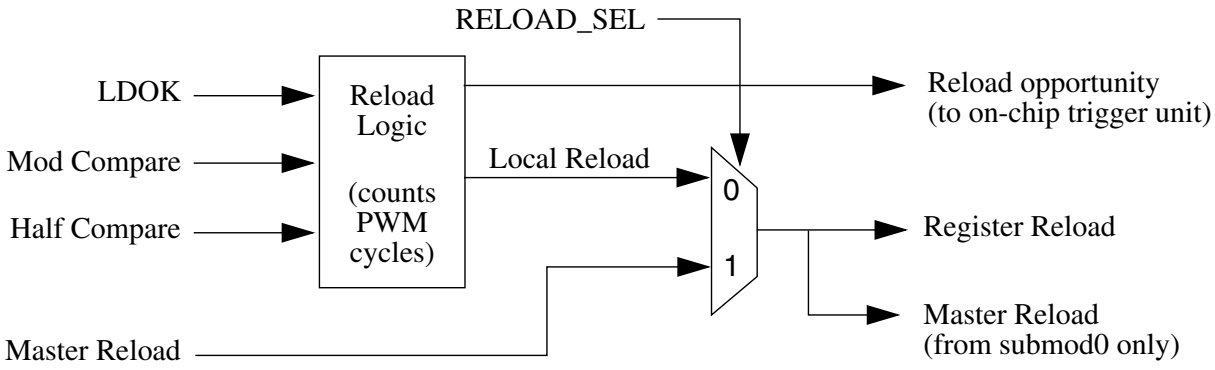


Figure 218. Register Reload Logic

44.3.2.3 Counter Synchronization

In the following figure, the 16 bit counter will count up until its output equals VAL1 which is used to specify the counter modulus value. The resulting compare causes a rising edge to occur on the Local Sync signal which is one of four possible sources used to cause the 16 bit counter to be initialized with INIT. If Local Sync is selected as the counter initialization signal, then VAL1 within the submodule effectively controls the timer period (and thus the PWM frequency generated by that submodule) and everything works on a local level.

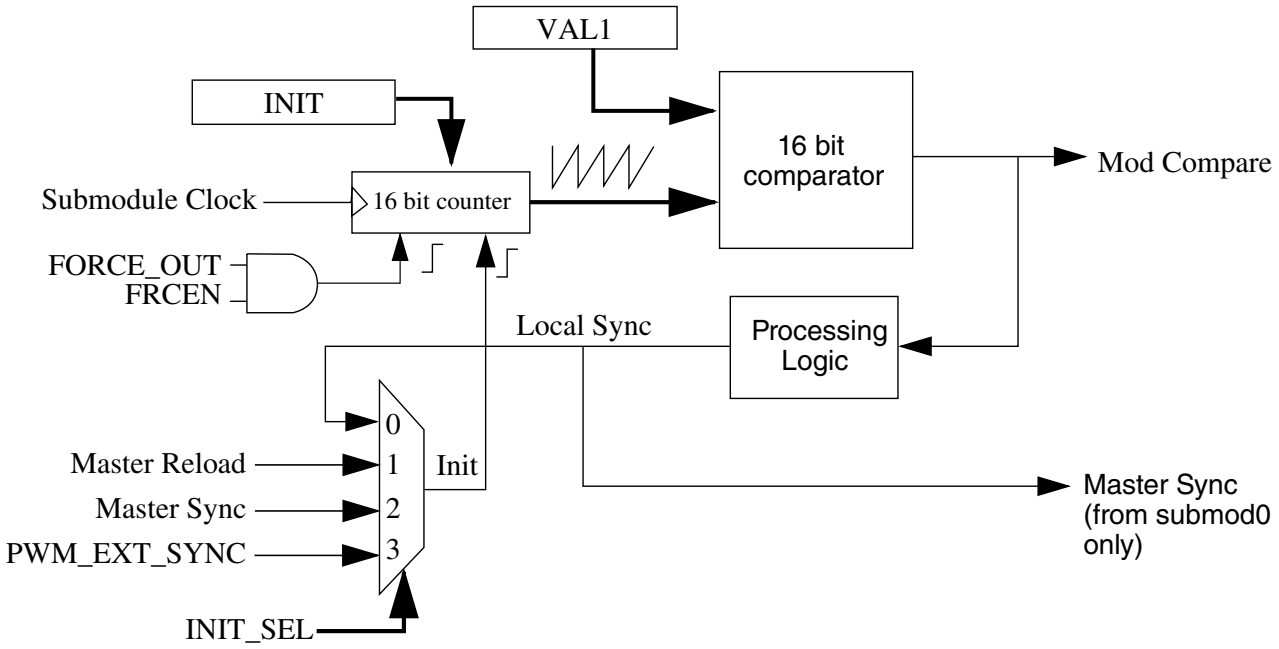


Figure 219. Submodule Timer Synchronization

The Master Sync signal originates as the Local Sync from submodule0. If configured to do so, the timer period of any submodule can be locked to the period of the timer in submodule0.

The PWM_EXT_SYNC signal originates on chip or off chip depending on the system architecture. This signal may be selected as the source for counter initialization so that an external source can control the period of all submodules.

If the Master Reload signal is selected as the source for counter initialization, then the period of the counter will be locked to the register reload frequency of submodule0. Since the reload frequency is usually commensurate to the sampling frequency of the software control algorithm, the submodule counter period will therefore equal the sampling period. As a result, this timer can be used to generate output compares or output triggers over the entire sampling period which may consist of several PWM cycles. The Master Reload signal can only originate from submodule0.

The counter can optionally initialize upon the assertion of the FORCE_OUT signal assuming that CTRL2[FRCEN] is set. As indicated by the preceding figure, this constitutes a second init input into the counter which will cause the counter to initialize regardless of which signal is selected as the counter init signal. A forced initialization will also cause a register reload if MCTRL[LDOK] is set. The FORCE_OUT signal is provided mainly for commutated applications. When PWM signals are commutated on an inverter controlling a brushless DC motor, it is necessary to restart the PWM cycle at the beginning of the commutation interval. This action effectively resynchronizes the PWM waveform to the commutation timing. Otherwise, the average voltage applied to a motor winding integrated over the entire commutation interval will be a function of the timing between the asynchronous commutation event with respect to the PWM cycle. The effect is more critical at higher motor speeds where each commutation interval may consist of only a few PWM cycles. If the counter is not initialized at the start of each commutation interval, the result will be an oscillation caused by the beating between the PWM frequency and the commutation frequency.

44.3.2.4 PWM Generation

Figure 220 illustrates how PWM generation is accomplished in each submodule. In each case, two comparators and associated VALx registers are utilized for each PWM output signal. One comparator and VALx register are used to control the turn-on edge, while a second comparator and VALx register control the turn-off edge.

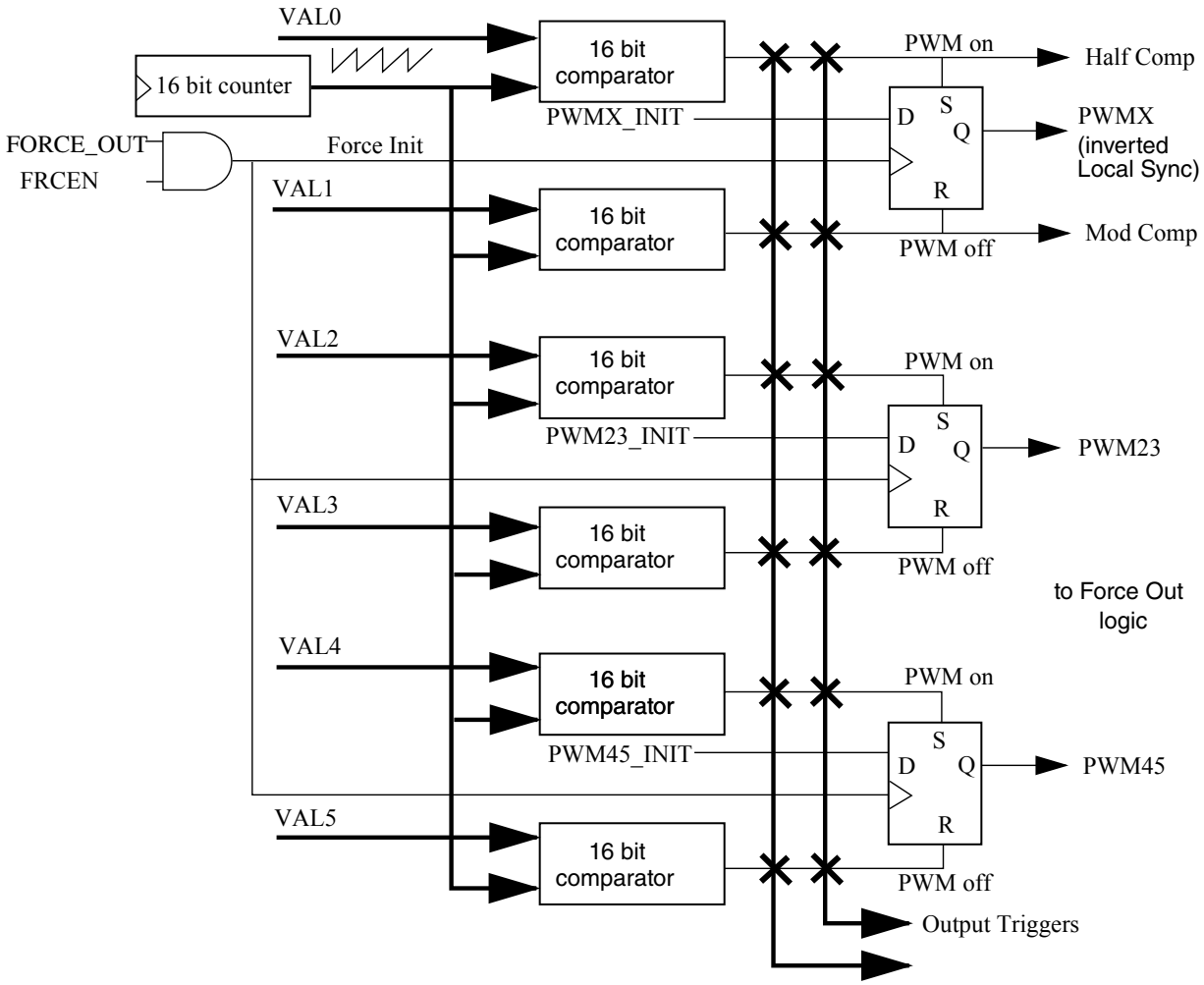


Figure 220. PWM Generation Hardware

The generation of the Local Sync signal is performed exactly the same way as the other PWM signals in the submodule. While comparator 0 causes a falling edge of the Local Sync signal, comparator 1 generates a rising edge. Comparator 1 is also hardwired to the reload logic to generate the full cycle reload indicator.

If VAL1 is controlling the modulus of the counter and VAL0 is half of the VAL1 register minus the INIT value, then the half cycle reload pulse will occur exactly half way through the timer count period and the Local Sync will have a 50% duty cycle. On the other hand, if the VAL1 and VAL0 registers are not required for register reloading or counter initialization, they can be used to modulate the duty cycle of the Local Sync signal, effectively turning it into an auxiliary PWM signal (PWM_X) assuming that the PWM_X pin is not being used for another function such as input capture or deadtime distortion correction. Including the Local Sync signal, each submodule is capable of generating three PWM signals where software has complete control over each edge of each of the signals.

If the comparators and edge value registers are not required for PWM generation, they can also be used for other functions such as output compares, generating output triggers, or generating interrupts at timed intervals.

The 16-bit comparators shown in Figure 220 are "equal to" comparators. In addition, if both the set and reset of the flip-flop are asserted, then the flop output goes to 0.

44.3.2.5 Output Compare Capabilities

By using the VALx registers in conjunction with the submodule timer and 16 bit comparators, buffered output compare functionality can be achieved with no additional hardware required. Specifically, the following output compare functions are possible:

- An output compare sets the output high
- An output compare sets the output low
- An output compare generates an interrupt
- An output compare generates an output trigger

In PWM generation, an output compare is initiated by programming a VALx register for a timer compare, which in turn causes the output of the D flip-flop to either set or reset. For example, if an output compare is desired on the PWM_A signal that sets it high, VAL2 would be programmed with the counter value where the output compare should take place. However, to prevent the D flip-flop from being reset again after the compare has occurred, the VAL3 register must be programmed to a value outside of the modulus range of the counter. Therefore, a compare that would result in resetting the D flip-flop output would never occur. Conversely, if an output compare is desired on the PWM_A signal that sets it low, the VAL3 register is programmed with the appropriate count value and the VAL2 register is programmed with a value outside the counter modulus range. Regardless of whether a high compare or low compare is programmed, an interrupt or output trigger can be generated when the compare event occurs.

44.3.2.6 Force Out Logic

For each submodule, software can select between eight signal sources for the FORCE_OUT signal: local CTRL2[FORCE], the Master Force signal from submodule0, the local Reload signal, the Master Reload signal from submodule0, the Local Sync signal, the Master Sync signal from submodule0, the EXT_SYNC signal from on- or off-chip, or the EXT_FORCE signal from on- or off-chip depending on the chip architecture. The local signals are used when the user simply wants to change the signals on the output pins of the submodule without regard for synchronization with other submodules. However, if it is required that all signals on all submodule outputs change at the same time, the Master, EXT_SYNC, or EXT_FORCE signals should be selected.

Figure 221 illustrates the Force logic. The SEL23 and SEL45 fields each choose from one of four signals that can be supplied to the submodule outputs: the PWM signal, the inverted PWM signal, a binary level specified by software via the OUT23 and OUT45 bits, or the PWM_EXTA or PWM_EXTB alternate external control signals. The selection can be determined ahead of time and, when a FORCE_OUT event occurs, these values are presented to the signal selection mux that immediately switches the requested signal to the output of the mux for further processing downstream.

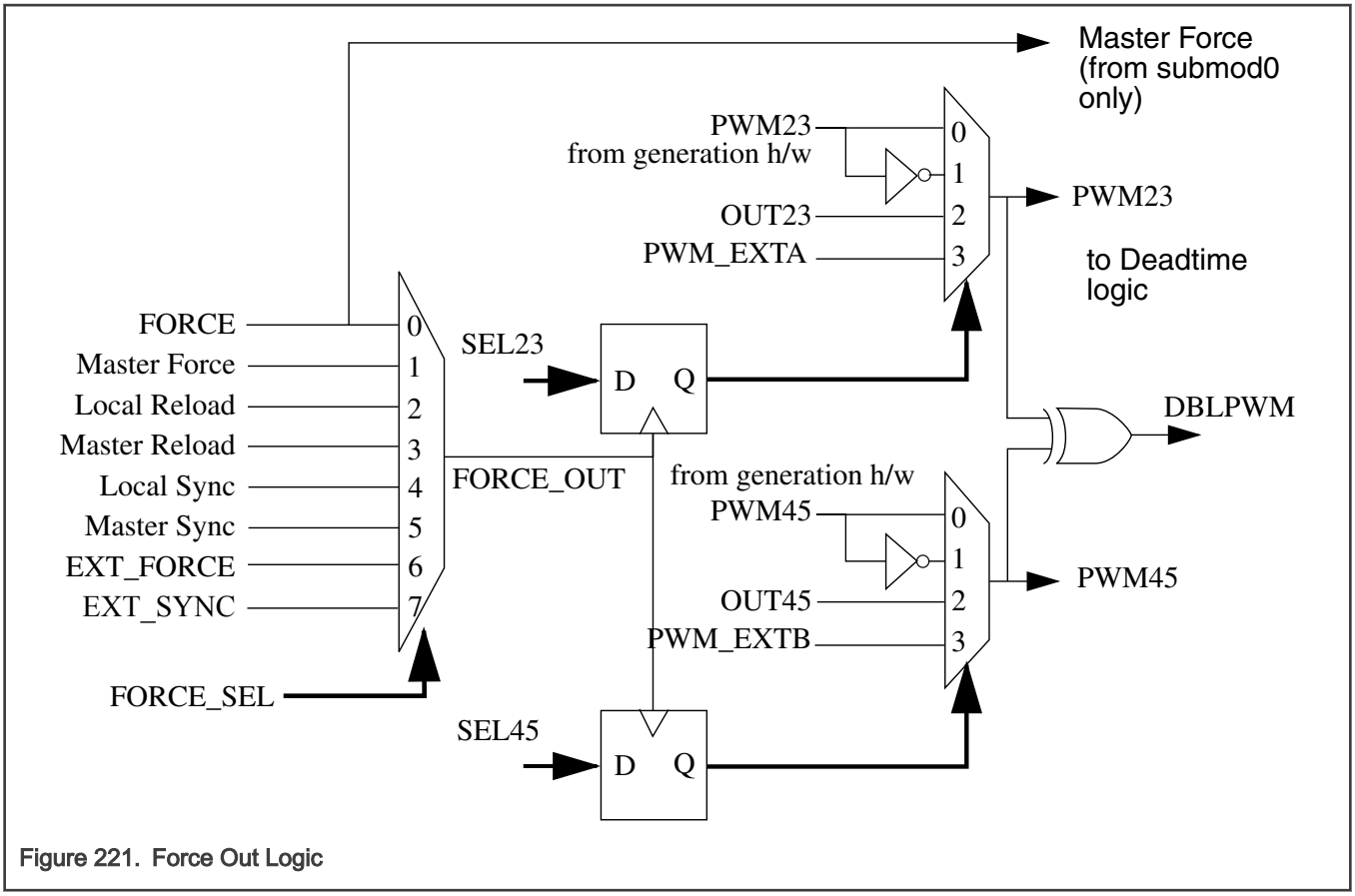


Figure 221. Force Out Logic

The local CTRL2[FORCE] signal of submodule0 can be broadcast as the Master Force signal to other submodules. This feature allows the CTRL2[FORCE] of submodule0 to synchronously update all of the submodule outputs at the same time. The EXT_FORCE signal originates from outside the PWM module from a source such as a timer or digital comparators in the Analog-to-Digital Converter.

44.3.2.7 Independent or Complementary Channel Operation

Writing a logic one to CTRL2[INDEP] configures the pair of PWM outputs as two independent PWM channels. Each PWM output is controlled by its own VALx pair operating independently of the other output.

Writing a logic zero to CTRL2[INDEP] configures the PWM output as a pair of complementary channels. The PWM pins are paired as shown in Figure 3-16 in complementary channel operation. Which signal is connected to the output pin (PWM23 or PWM45) is determined by MCTRL[IPOL].

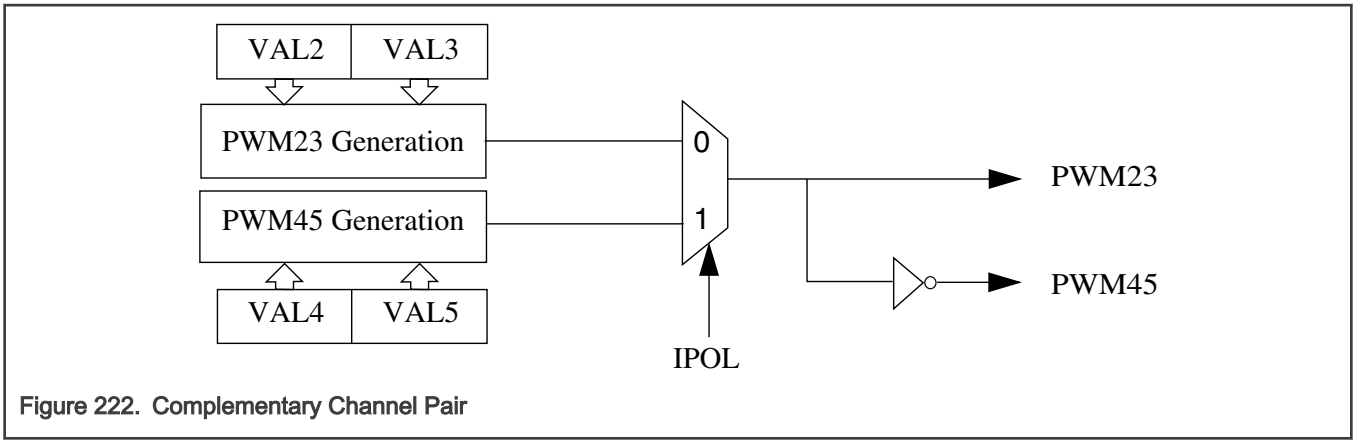


Figure 222. Complementary Channel Pair

The complementary channel operation is for driving top and bottom transistors in a motor drive circuit, such as the one in Figure 223.

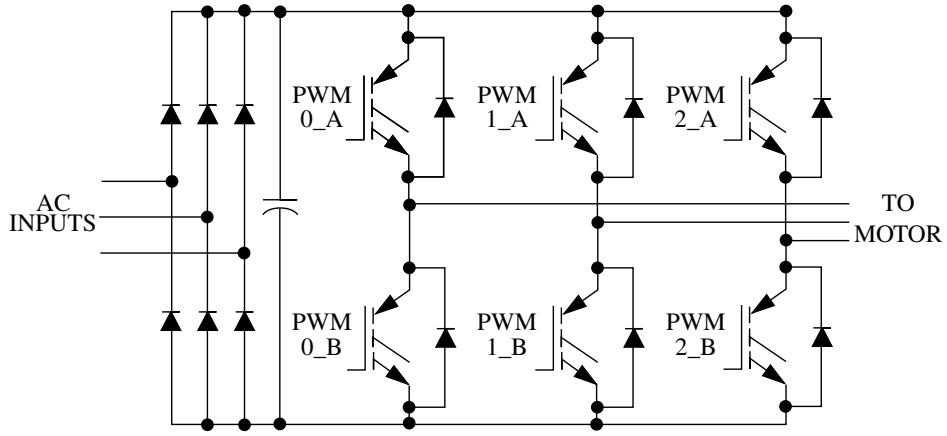


Figure 223. Typical 3 Phase AC Motor Drive

Complementary operation allows the use of the deadtime insertion feature.

44.3.2.8 Deadtime Insertion Logic

The following figure shows the deadtime insertion logic of each submodule which is used to create non-overlapping complementary signals when not in independent mode.

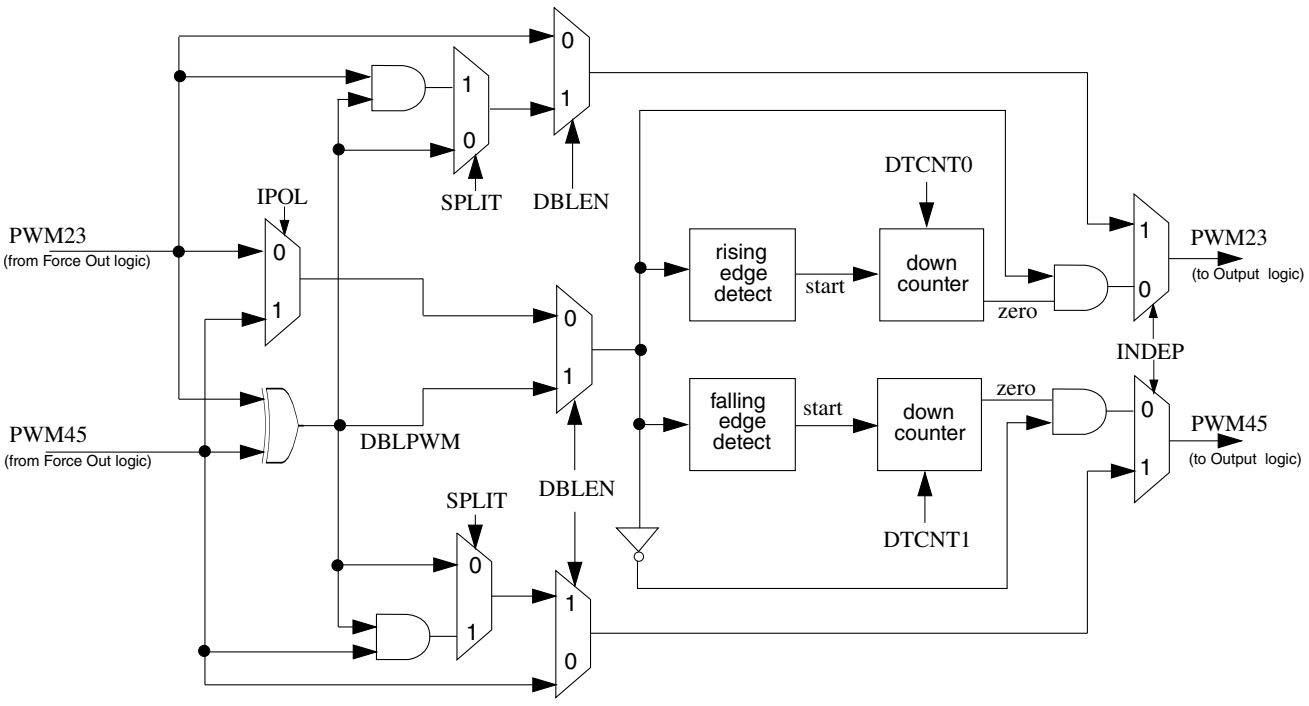


Figure 224. Deadtime Insertion Logic

While in the complementary mode, a PWM pair can be used to drive top/bottom transistors, as shown in the figure. When the top PWM channel is active, the bottom PWM channel is inactive, and vice versa.

NOTE

To avoid short circuiting the DC bus and endangering the transistor, there must be no overlap of conducting intervals between top and bottom transistor. But the transistor's characteristics may make its switching-off time longer than switching-on time. To avoid the conducting overlap of top and bottom transistors, deadtime needs to be inserted in the switching period, as illustrated in the following figure.

The deadtime generators automatically insert software-selectable activation delays into the pair of PWM outputs. The deadtime registers (DTCNT0 and DTCNT1) specify the number of IPBus clock cycles to use for deadtime delay. Every time the deadtime generator inputs change state, deadtime is inserted. Deadtime forces both PWM outputs in the pair to the inactive state.

When deadtime is inserted in complementary PWM signals connected to an inverter driving an inductive load, the PWM waveform on the inverter output will have a different duty cycle than what appears on the output pins of the PWM module. This results in a distortion in the voltage applied to the load. A method of correcting this, adding to or subtracting from the PWM value used, is discussed next.

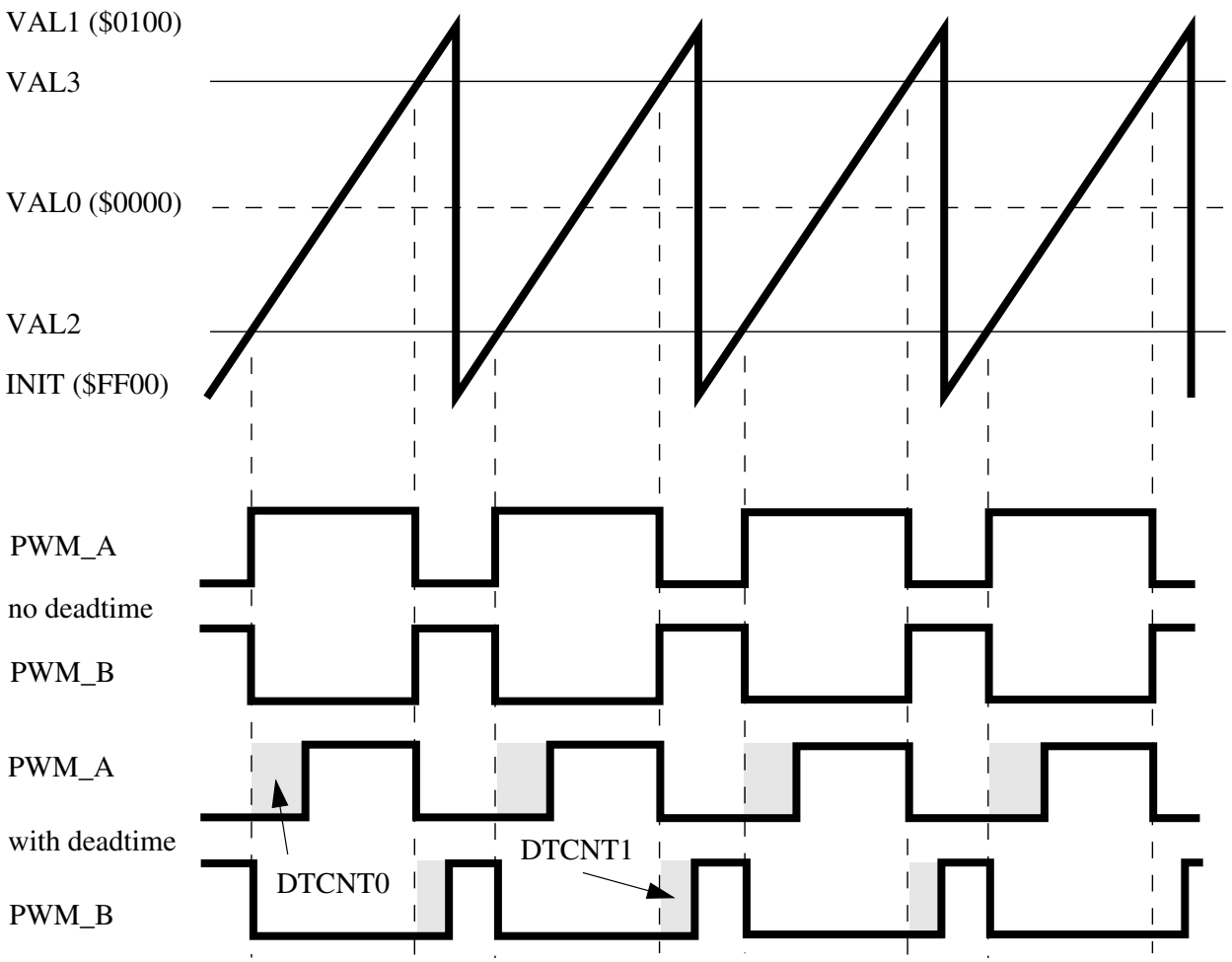


Figure 225. Deadtime Insertion

44.3.2.8.1 Top/Bottom Correction

In complementary mode, either the top or the bottom transistor controls the output voltage. However, deadtime has to be inserted to avoid overlap of conducting interval between the top and bottom transistor. Both transistors in complementary mode are off during deadtime, allowing the output voltage to be determined by the current status of load and introduce distortion in the output voltage. See the following figure. On AC induction motors running open-loop, the distortion typically manifests itself as poor low-speed performance, such as torque ripple and rough operation.

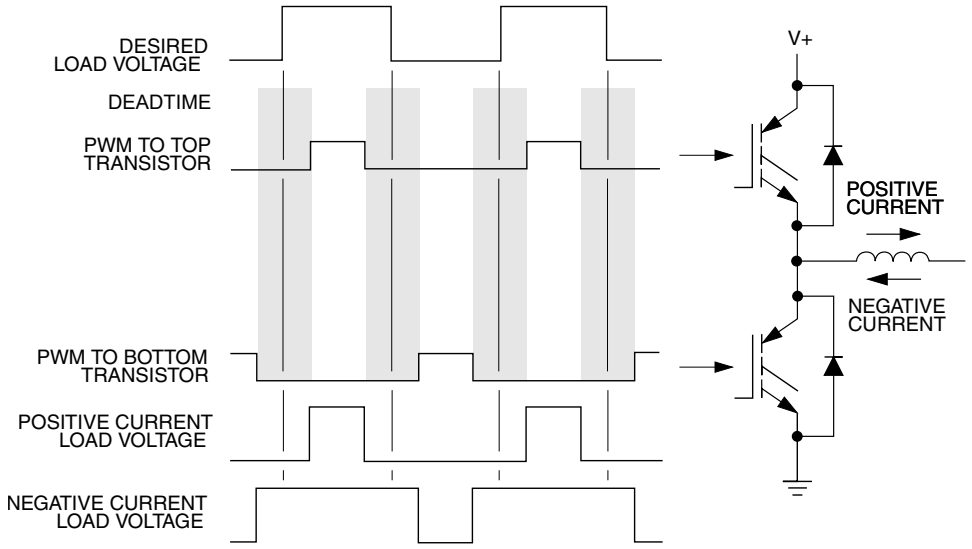


Figure 226. Deadtime Distortion

During deadtime, load inductance distorts output voltage by keeping current flowing through the diodes. This deadtime current flow creates a load voltage that varies with current direction. With a positive current flow, the load voltage during deadtime is equal to the bottom supply, putting the top transistor in control. With a negative current flow, the load voltage during deadtime is equal to the top supply putting the bottom transistor in control.

Remembering that the original PWM pulse widths where shortened by deadtime insertion, the averaged sinusoidal output will be less than the desired value. However, when deadtime is inserted, it creates a distortion in the motor current waveform inverter outputs. This distortion is aggravated by dissimilar turn-on and turn-off delays of each of the transistors. By giving the PWM module information on which transistor is controlling at a given time this distortion can be corrected.

For a typical circuit in complementary channel operation, only one of the transistors will be effective in controlling the output voltage at any given time. This depends on the direction of the motor inverter current for that pair, as the preceding figure shows. To correct distortion one of two different factors must be added to the desired PWM value, depending on whether the top or bottom transistor is controlling the output voltage. Therefore, the software is responsible for calculating both compensated PWM values prior to placing them in the VALx registers. Either the VAL2/VAL3 or the VAL4/VAL5 register pair controls the pulse width at any given time. For a given PWM pair, whether the VAL2/VAL3 or VAL4/VAL5 pair is active depends on either:

- The state of the current status pin, PWMX, for that driver
- The state of the odd/even correction bit, MCTRL[IPOL], for that driver

To correct deadtime distortion, software can decrease or increase the value in the appropriate VALx register.

- In edge-aligned operation, decreasing or increasing the PWM value by a correction value equal to the deadtime typically compensates for deadtime distortion.
- In center-aligned operation, decreasing or increasing the PWM value by a correction value equal to one-half the deadtime typically compensates for deadtime distortion.

44.3.2.8.2 Manual Correction

To detect the current status, the voltage on each PWMX pin is sampled twice in a PWM period, at the end of each deadtime. The value is stored in CTRL[DT]. CTRL[DT] is a timing marker especially indicating when to toggle between PWM value registers. Software can then set MCTRL[IPOL] to switch between VAL2/VAL3 and VAL4/VAL5 register pairs according to CTRL[DT] values.

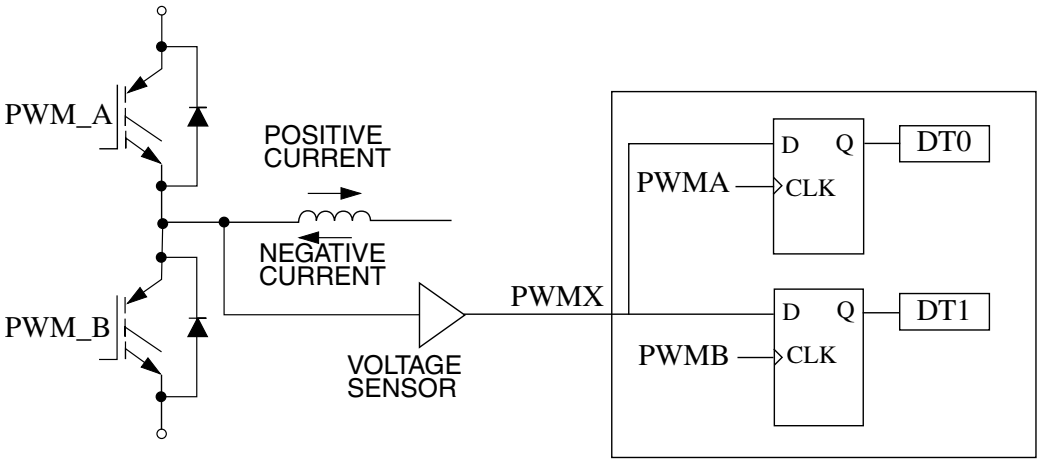


Figure 227. Current-status Sense Scheme for Deadtime Correction

Both D flip-flops latch low, CTRL[DT] = 00, during deadtime periods if current is large and flowing out of the complementary circuit. See the preceding figure. Both D flip-flops latch the high, CTRL[DT] = 11, during deadtime periods if current is also large and flowing into the complementary circuit.

However, under low-current, the output voltage of the complementary circuit during deadtime is somewhere between the high and low levels. The current cannot free-wheel through the opposition anti-body diode, regardless of polarity, giving additional distortion when the current crosses zero. **Sampled results will be CTRL[DT] = b10. Thus, the best time to change one PWM value register to another is just before the current zero crossing.**

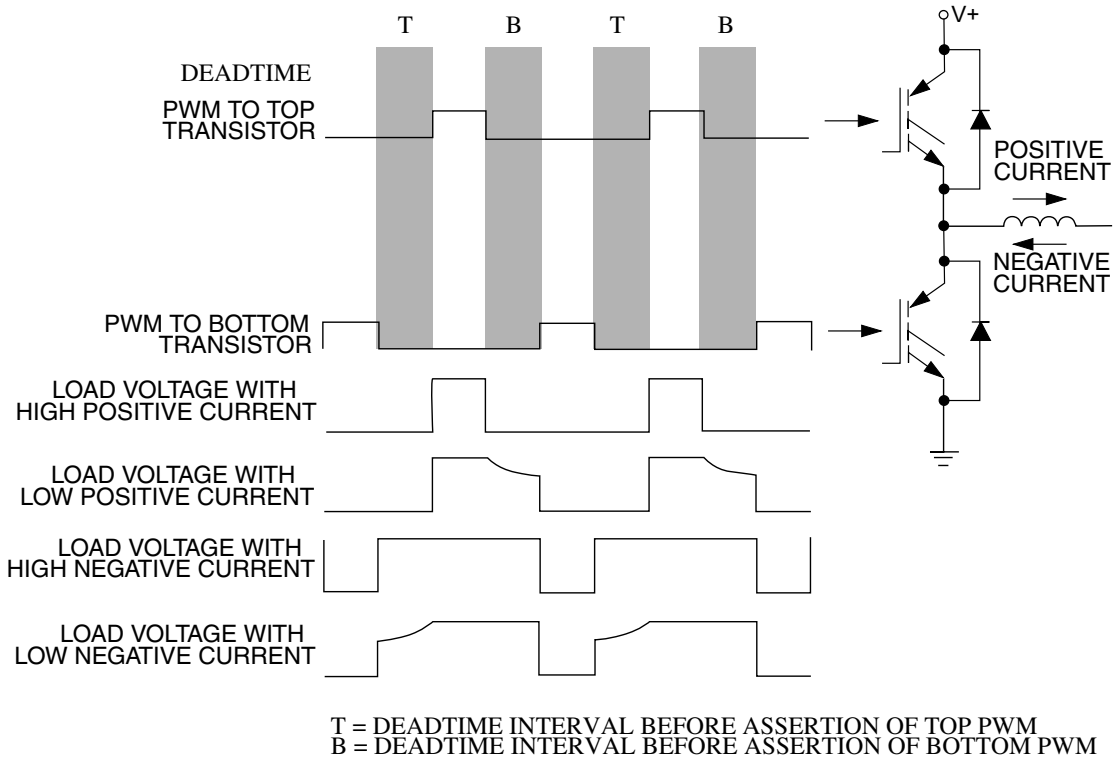


Figure 228. Output Voltage Waveforms

44.3.2.9 Fractional Delay Logic

For applications where more resolution than a single IPBus clock period is needed, the fractional delay logic can be used to achieve fine resolution on the rising and falling edges of the PWM_A and PWM_B outputs . Enable the use of the fractional delay logic by setting FRCTRL[FRACx_EN]. The FRACVALx registers act as a fractional clock cycle addition to the turn on and turn off count specified by the VAL2, VAL3, VAL4, or VAL5 registers. The FRACVAL1 register acts as a fractional increase in the PWM period as defined by VAL1. If FRACVAL1 is programmed to a non-zero value, then the largest value for the VAL1 register is 0xFFFE for unsigned usage or 0x7FFE for signed usage. This limit is needed in order to avoid counter rollovers when accumulating the fractional additional period.

Both the fractional enables (1, 23, 45) and the fractional values (1-5) are double buffered and reload at the same time as the value registers.

Each PWM cycle, the value compare point is increased by 1 if there is overflow on the double buffered value of the fractional register plus the 5-bit accumulated fractional value. The accumulated fractional value starts at zero, so it is impossible to overflow the first PWM cycle.

At the end of each PWM cycle, if the corresponding fractional enable is set then the accumulated fractional value increments by the fractional value (double buffered value). The accumulated fractional value is 5-bits, so in the case of overflow only the remainder is kept. If the corresponding fractional enable is clear, then the accumulated fractional value is reset. This is the only way to reset the accumulated fractional value.

The accumulated fractional values are not accessible to software.

As an example, assume the following conditions:

- INIT = 0x0000
- VAL1 = 0x000F
- VAL2 = 0x0000
- VAL3 = 0x0007
- FRACVAL3 = 0x00

This would cause the PWM output to have a 50% duty cycle: it would be high from a count value of 0x0 to the count value of 0x7, at which point it would go low until the counter reaches the maximum value of 0xF. If FRACVAL3 was changed to a value of 0x17, then the time the PWM output is active would be:

- 8 cycles + (23/32) cycle = 8.719 cycles
- for a high duty cycle of:
- (8.719 cycles / 16 cycles) x 100% = 54.49%

Another case involves fine tuning the PWM period using the FRACVAL1 register. If you want a period of 100.25 clock cycles, the program VAL1 with 0x0064 and FRACVAL1 with 0x4000. The fractional value will accumulate so that every 4 PWM cycles will be 1 clock cycle longer (101 instead of 100). The rising and falling edges of the PWM outputs will also use the accumulated fraction to delay their edges and maintain a consistent 100.25 cycles spacing between corresponding edges from one cycle to the next.

44.3.2.9.1 Fractional Delay Logic without NanoEdge Placement Block

For submodules that are not supported by the NanoEdge placer, the PWM can use dithering to simulate fine edge control. Enable this feature by setting the FRCTRL[FRAC1_EN], FRCTRL[FRAC23_EN], and FRCTRL[FRAC45_EN] bits. The PWM period or the PWM edges will dither from the nearest whole number values to achieve an average value that is equivalent to the programmed fractional value. The added cycles are based on the accumulation of the fractional component. For example, if you want the PWM period to be 50.25 clock cycles, then program VAL1 with 0x0032 and FRACVAL1 with 0x4000. The PWM period will be 50 cycles long most of the time, but will occasionally be 51 cycles long to achieve a long-term average of 50.25 cycles.

In submodules that are not supported by a NanoEdge placer, the clock frequency is not required to be any specific value to achieve proper operation.

44.3.2.10 Output Logic

The following figure shows the output logic of each submodule including how each PWM output has individual fault disabling, polarity control, and output enable. This allows for maximum flexibility when interfacing to the external circuitry.

The PWM23 and PWM45 signals which are output from the deadtime logic (refer to the figure) are positive true signals. In other words, a high level on these signals should result in the corresponding transistor in the PWM inverter being turned ON. The voltage level required at the PWM output pin to turn the transistor ON or OFF is a function of the logic between the pin and the transistor. Therefore, it is imperative that the user program OCTRL[POLA] and OCTRL[POLB] before enabling the output pins. A fault condition can result in the PWM output being tristated, forced to a logic 1, or forced to a logic 0 depending on the values programmed into the OCTRL[PWMxFS] fields.

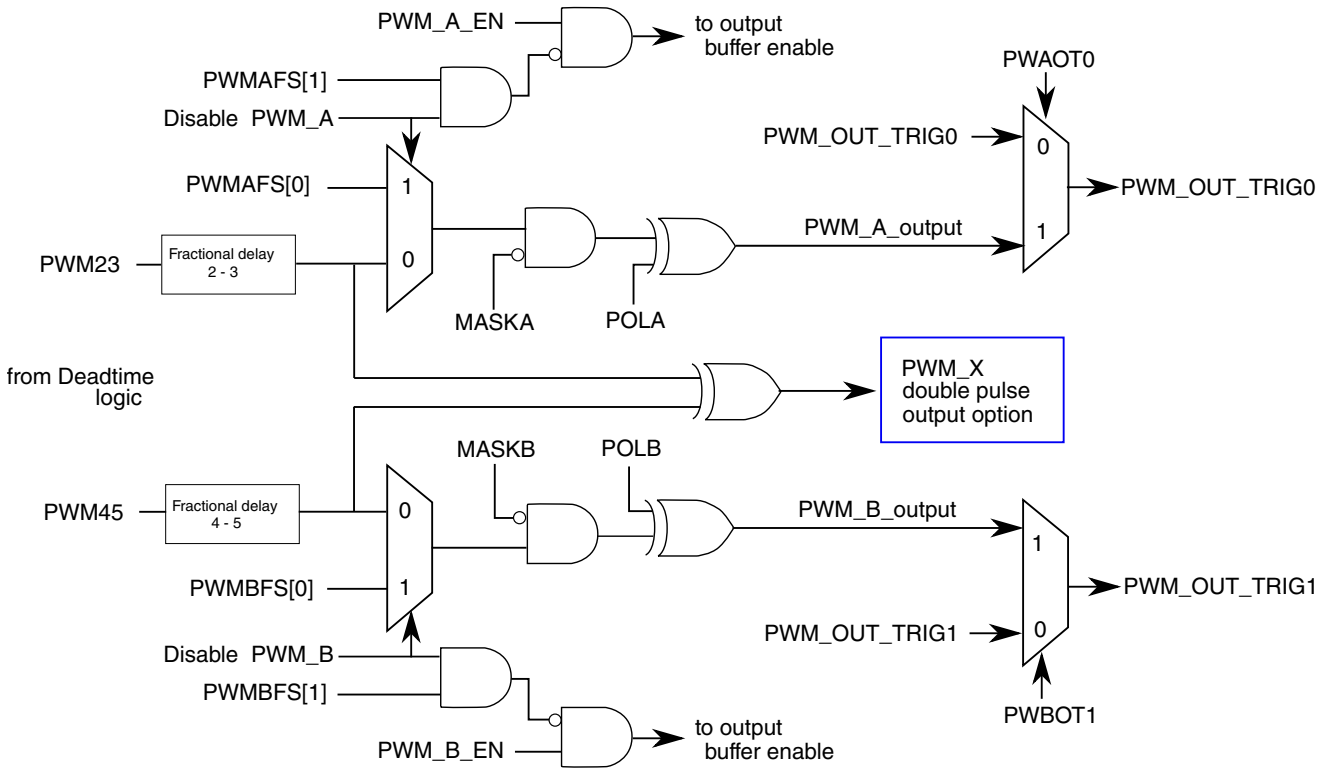


Figure 229. Output Logic

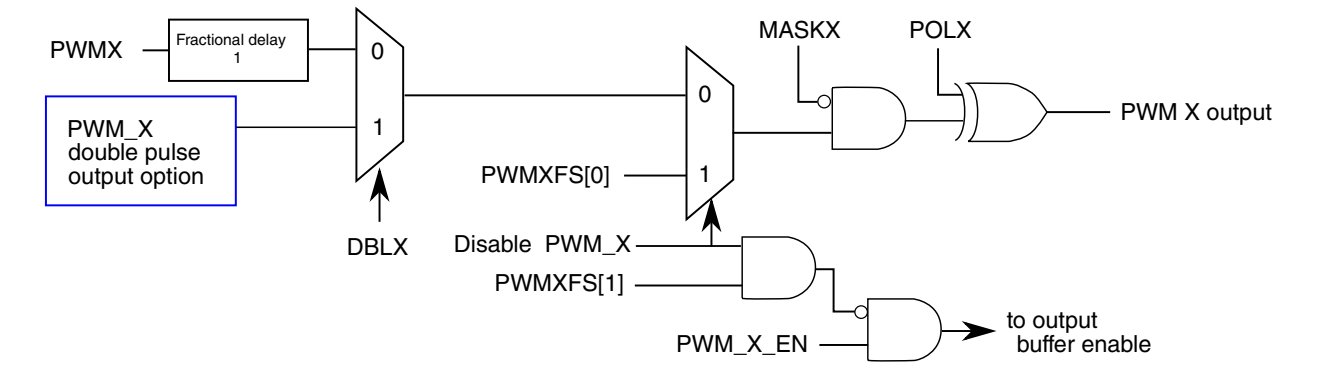


Figure 230. Output Logic continued

44.3.2.11 E-Capture

Commensurate with the idea of controlling both edges of an output signal, the Enhanced Capture (E-Capture) logic is designed to measure both edges of an input signal. As a result, when a submodule pin is configured for input capture, the CVALx registers associated with that pin are used to record the edge values.

The following figure is a block diagram of the E-Capture circuit. Upon entering the pin input, the signal is split into two paths. One goes straight to a mux input where software can select to pass the signal directly to the capture logic for processing. The other path connects the signal to an 8 bit counter which counts both the rising and falling edges of the signal. The output of this counter is compared to an 8 bit value that is specified by the user (EDGCMPx) and when the two values are equal, the comparator generates a pulse that resets the counter. This pulse is also supplied to the mux input where software can select it to be processed by the capture logic. This feature permits the E-Capture circuit to count up to 256 edge events before initiating a capture event. this feature is useful for dividing down high frequency signals for capture processing so that capture interrupts don't overwhelm the CPU. Also, this feature can be used to simply generate an interrupt after "n" events have been counted.

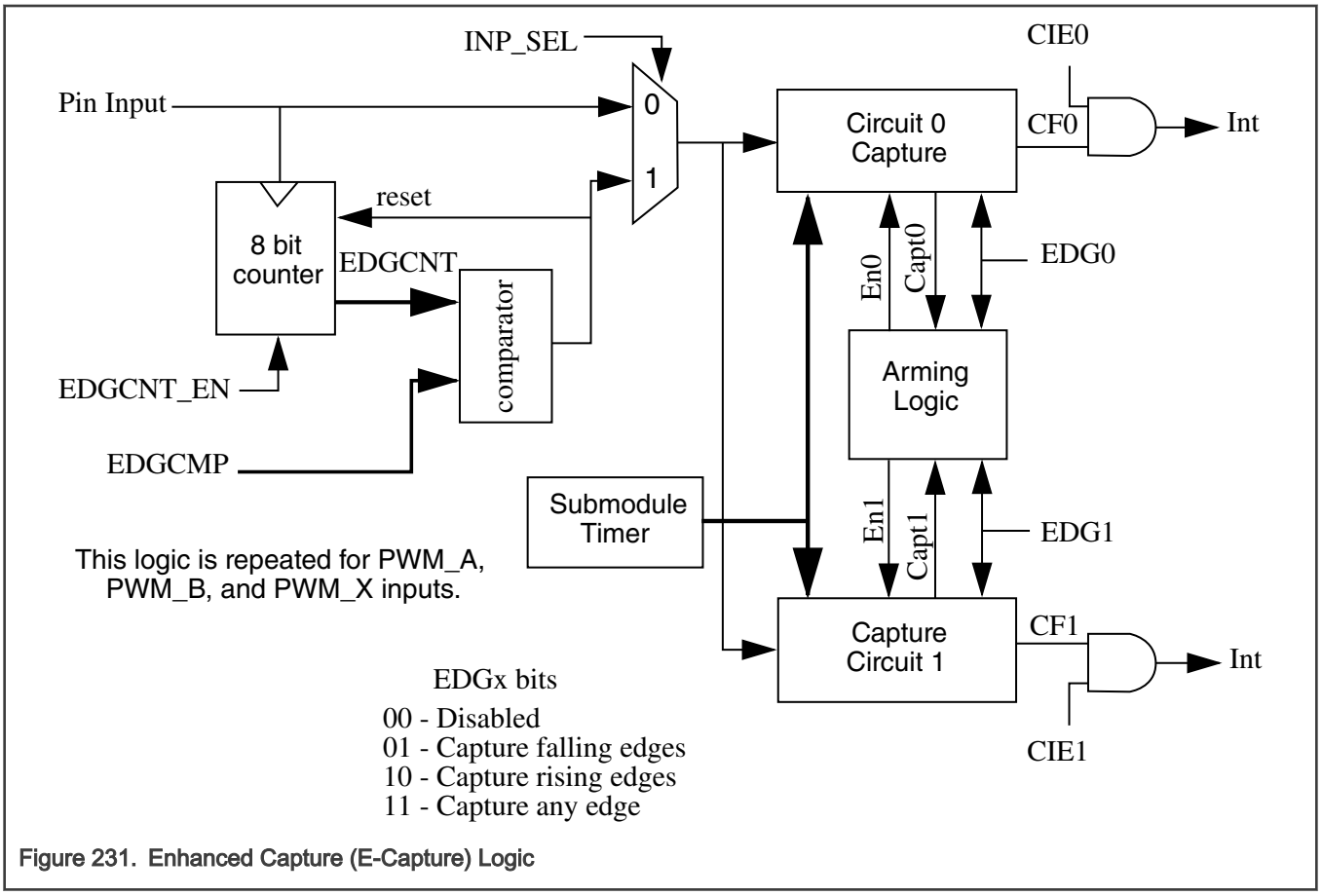


Figure 231. Enhanced Capture (E-Capture) Logic

Based on the mode selection, the mux selects either the pin input or the compare output from the count/compare circuit to be processed by the capture logic. The selected signal is routed to two separate capture circuits which work in tandem to capture sequential edges of the signal. The type of edge to be captured by each circuit is determined by CAPTCTRLx[EDGx1] and CAPTCTRLx[EDGx0], whose functionality is listed in the preceding figure. Also, controlling the operation of the capture circuits is the arming logic which allows captures to be performed in a free running (continuous) or one shot fashion. In free running mode, the capture sequences will be performed indefinitely. If both capture circuits are enabled, they will work together in a ping-pong style where a capture event from one circuit leads to the arming of the other and vice versa. In one shot mode, only one capture sequence will be performed. If both capture circuits are enabled, capture circuit 0 is first armed and when a capture event occurs, capture circuit 1 is armed. Once the second capture occurs, further captures are disabled until another capture sequence is initiated. Both capture circuits are also capable of generating an interrupt to the CPU.

44.3.2.12 Fault Protection

Fault protection can control any combination of PWM output pins. Faults are generated by a logic one on any of the FAULTx pins. This polarity can be changed via FCTRL[FLVL]. Each FAULTx pin can be mapped arbitrarily to any of the PWM outputs. When fault protection hardware disables PWM outputs, the PWM generator continues to run, only the output pins are forced to logic 0, logic 1, or tristated depending the values of OCTRL[PWMxFS].

The fault decoder disables PWM pins selected by the fault logic and the disable mapping (DISMAPn) registers. The following figure shows an example of the fault disable logic. Each bank of bits in DISMAPn control the mapping for a single PWM pin. See the following table.

The fault protection is enabled even when the PWM module is not enabled; therefore, a fault will be latched in and must be cleared in order to prevent an interrupt when the PWM is enabled.

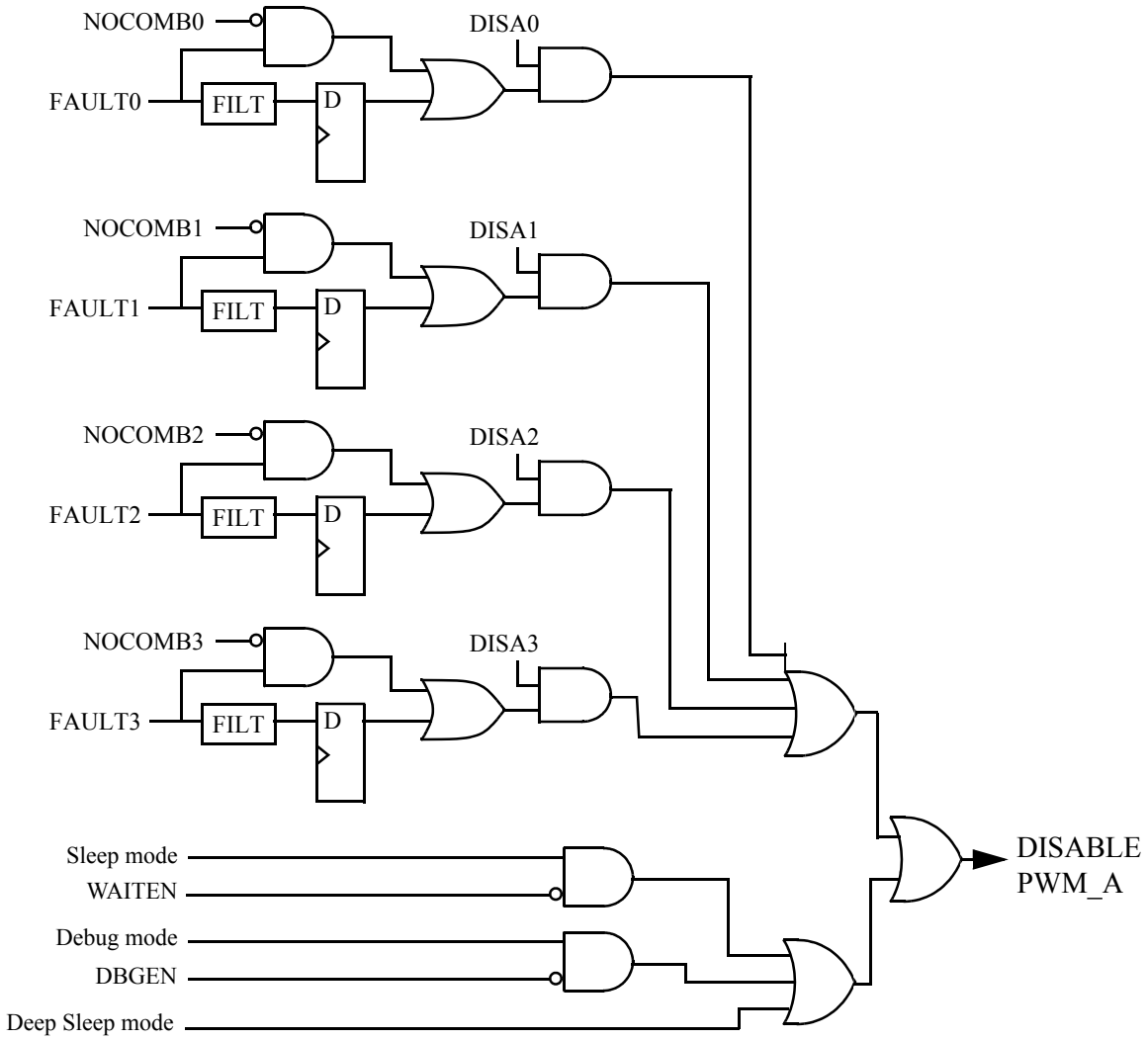


Figure 232. Fault Decoder for PWM_A

Table 378. Fault Mapping

PWM Pin	Controlling Register Bits
PWM_A	DISMAP0[DIS0A]

Table continues on the next page...

Table 378. Fault Mapping (continued)

PWM Pin	Controlling Register Bits
PWM_B	DISMAP0[DIS0B]
PWM_X	DISMAP0[DIS0X]

44.3.2.12.1 Fault Pin Filter

Each fault pin has a programmable filter that can be bypassed. The sampling period of the filter can be adjusted with FFILT[FILT_PER]. The number of consecutive samples that must agree before an input transition is recognized can be adjusted using FFILT[FILT_CNT]. Setting FFILT[FILT_PER] to all 0 disables the input filter for a given FAULTx pin.

Upon detecting a logic 0 on the filtered FAULTx pin (or a logic 1 if FCTRL[FLVLx] is set), the corresponding FSTS[FFPINx] and fault flag, FSTS[FFLAGx], bits are set. FSTS[FFPINx] remains set as long as the filtered FAULTx pin is zero. Clear FSTS[FFLAGx] by writing a logic 1 to FSTS[FFLAGx].

If the FIEx, FAULTx pin interrupt enable bit is set, FSTS[FFLAGx] generates a CPU interrupt request. The interrupt request latch remains set until:

- Software clears FSTS[FFLAGx] by writing a logic one to the bit
- Software clears the FIEx bit by writing a logic zero to it
- A reset occurs

Even with the filter enabled, there is a combinational path from the FAULTx inputs to the PWM pins. This logic is also capable of holding a fault condition in the event of loss of clock to the PWM module.

44.3.2.12.2 Automatic Fault Clearing

Setting an automatic clearing mode bit, FCTRL[FAUTOx], configures faults from the FAULTx pin for automatic clearing.

When FCTRL[FAUTOx] is set, disabled PWM pins are enabled when the FAULTx pin returns to logic one and a new PWM full or half cycle begins. See the following figure. If FSTS[FFULLx] is set, then the disabled PWM pins are enabled at the start of a full cycle. If FSTS[FHALFx] is set, then the disabled PWM pins are enabled at the start of a half cycle. Clearing FSTS[FFLAGx] does not affect disabled PWM pins when FCTRL[FAUTOx] is set.

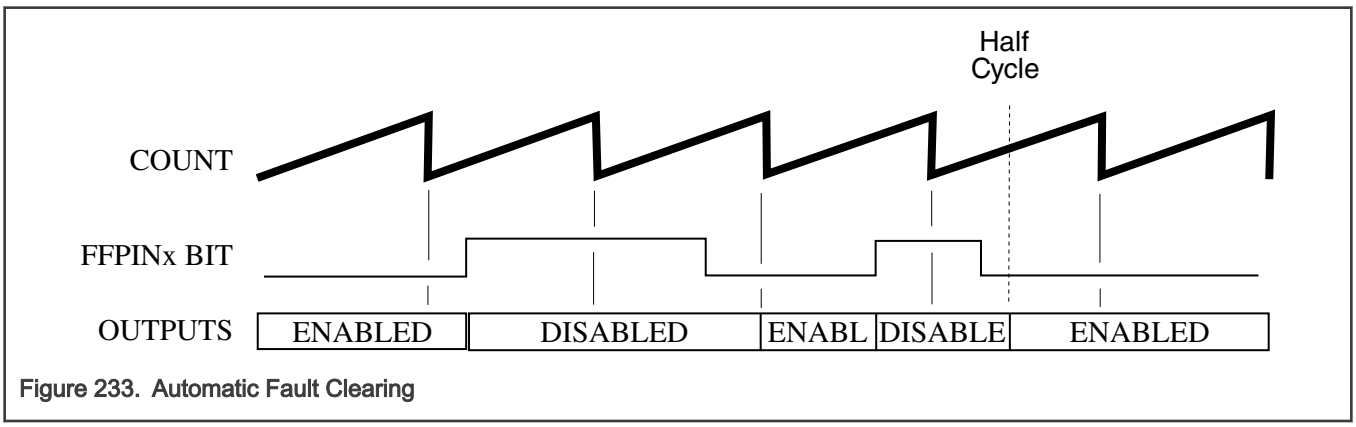


Figure 233. Automatic Fault Clearing

44.3.2.12.3 Manual Fault Clearing

Clearing the automatic clearing mode bit, FCTRL[FAUTOx], configures faults from the FAULTx pin for manual clearing:

- If the fault safety mode bits, FCTRL[FSAFEx], are clear, then PWM pins disabled by the FAULTx pins are enabled when:
  - Software clears the corresponding FSTS[FFLAGx] flag

— The pins are enabled when the next PWM full or half cycle begins regardless of the logic level detected by the filter at the FAULTx pin. See the first following figure. If FSTS[FFULLx] is set, then the disabled PWM pins are enabled at the start of a full cycle. If FSTS[FHALFx] is set, then the disabled PWM pins are enabled at the start of a half cycle.

- If the fault safety mode bits, FCTRL[FSAFEx], are set, then PWM pins disabled by the FAULTx pins are enabled when:
  — Software clears the corresponding FSTS[FFLAGx] flag
  — The filter detects a logic zero on the FAULTx pin at the start of the next PWM full or half cycle boundary. See the second following figure. If FSTS[FFULLx] is set, then the disabled PWM pins are enabled at the start of a full cycle. If FSTS[FHALFx] is set, then the disabled PWM pins are enabled at the start of a half cycle.

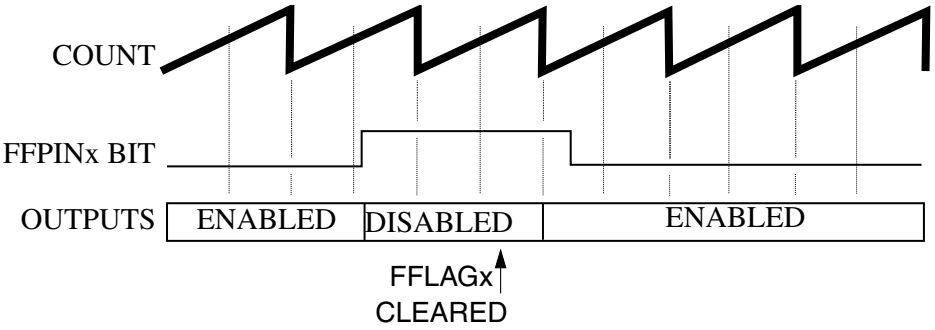


Figure 234. Manual Fault Clearing (FCTRL[FSAFE]=0)

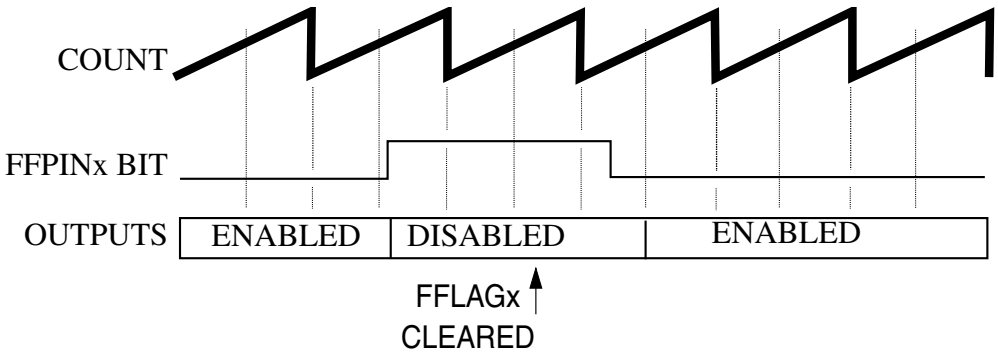


Figure 235. Manual Fault Clearing (FCTRL[FSAFE]=1)

NOTE

Fault protection also applies during software output control when the SEL23 and SEL45 fields are set to select OUT23 and OUT45 bits or PWM_EXTA and PWM_EXTB. Fault clearing still occurs at half PWM cycle boundaries while the PWM generator is engaged, MCTRL[RUN] equals one. But the OUTx bits can control the PWM pins while the PWM generator is off, MCTRL[RUN] equals zero. Thus, fault clearing occurs at IPBus cycles while the PWM generator is off and at the start of PWM cycles when the generator is engaged.

44.3.2.12.4 Fault Testing

FTST[FTEST] is used to simulate a fault condition on each of the fault inputs within that fault channel.

44.3.3 PWM Generator Loading

44.3.3.1 Load Enable

MCTRL[LDOK] enables loading of the following PWM generator parameters:

- The prescaler divisor—from CTRL[PRSC]

- The PWM period and pulse width—from the INIT and VALx registers

MCTRL[LDOK] allows software to finish calculating all of these PWM parameters so they can be synchronously updated. The CTRL[PRSC], INIT, and VALx registers are loaded by software into a set of outer buffers. When MCTRL[LDOK] is set, these values are transferred to an inner set of registers at the beginning of the next PWM reload cycle to be used by the PWM generator. These values can be transferred to the inner set of registers immediately upon setting MCTRL[LDOK] if CTRL[LDMOD] is set. Set MCTRL[LDOK] by reading it when it is a logic zero and then writing a logic one to it. After loading, MCTRL[LDOK] is automatically cleared.

44.3.3.2 Load Frequency

CTRL[LDFQ] selects an integral loading frequency of one to 16 PWM reload opportunities. CTRL[LDFQ] takes effect at every PWM reload opportunity, regardless the state of MCTRL[LDOK]. CTRL[HALF] and CTRL[FULL] control reload timing. If CTRL[FULL] is set, a reload opportunity occurs at the end of every PWM cycle when the count equals VAL1. If CTRL[HALF] is set, a reload opportunity occurs at the half cycle when the count equals VAL0. If both CTRL[HALF] and CTRL[FULL] are set, a reload opportunity occurs twice per PWM cycle when the count equals VAL1 and when it equals VAL0.

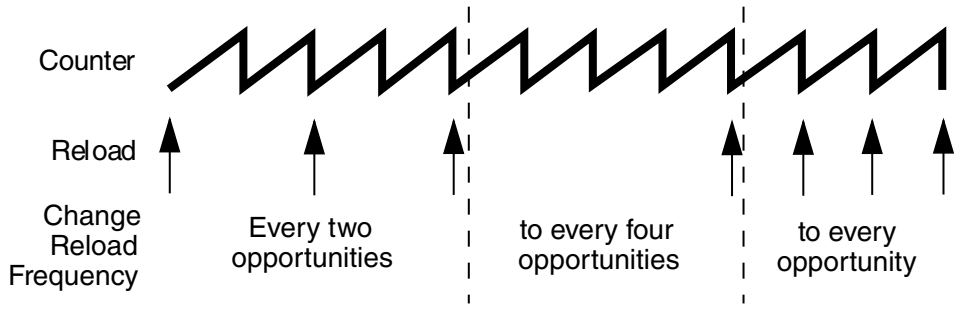


Figure 236. Full Cycle Reload Frequency Change

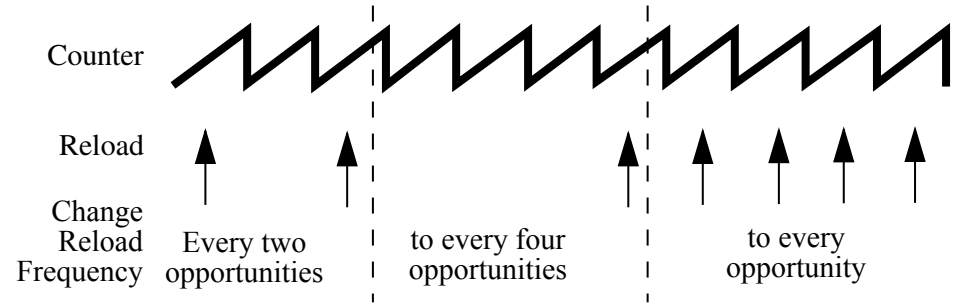


Figure 237. Half Cycle Reload Frequency Change

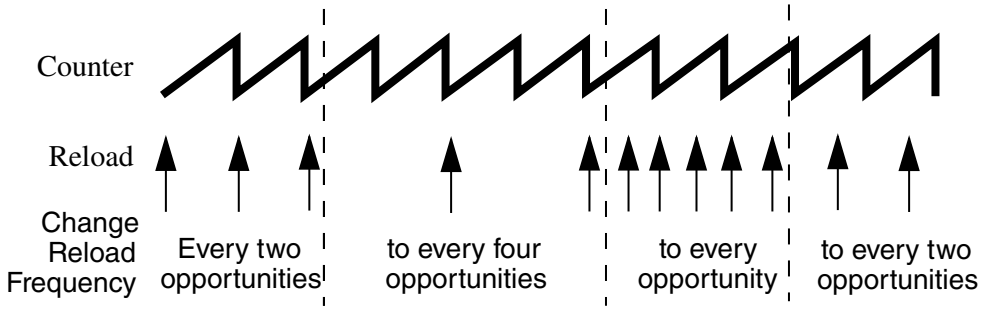


Figure 238. Full and Half Cycle Reload Frequency Change

44.3.3.3 Reload Flag

At every reload opportunity the PWM Reload Flag (STS[RF]) is set. Setting STS[RF] happens even if an actual reload is prevented by MCTRL[LDOK]. If the PWM reload interrupt enable bit, INTEN[RIE], is set, the STS[RF] flag generates CPU interrupt requests allowing software to calculate new PWM parameters in real time. When INTEN[RIE] is not set, reloads still occur at the selected reload rate without generating CPU interrupt requests.

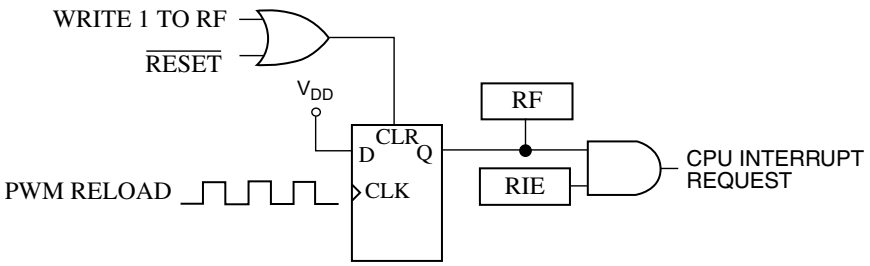


Figure 239. PWMF Reload Interrupt Request

44.3.3.4 Reload Errors

Whenever one of the VALx, FRACVALx, or CTRL[PRSC] registers is updated, the STS[RUF] flag is set to indicate that the data is not coherent. STS[RUF] will be cleared by a successful reload which consists of the reload signal while MCTRL[LDOK] is set. If STS[RUF] is set and MCTRL[LDOK] is clear when the reload signal occurs, a reload error has taken place and STS[REF] is set. If STS[RUF] is clear when a reload signal asserts, then the data is coherent and no error will be flagged.

44.3.3.5 Initialization

Initialize all registers and set MCTRL[LDOK] before setting MCTRL[RUN].

NOTE

Even if MCTRL[LDOK] is not set, setting MCTRL[RUN] also sets the STS[RF] flag. To prevent a CPU interrupt request, clear INTEN[RIE] before setting MCTRL[RUN].

The PWM generator uses the last values loaded if MCTRL[RUN] is cleared and then set while MCTRL[LDOK] equals zero.

When MCTRL[RUN] is cleared:

- The STS[RF] flag and pending CPU interrupt requests are not cleared
- All fault circuitry remains active
- Software/external output control remains active
- Deadtime insertion continues during software/external output control

44.4 External Signals

The PWM has pins named PWM_An, PWM_Bn, PWM_Xn, FAULTn, EXT_SYNC, EXT_FORCE, PWMn_EXTA, and PWMn_EXTB. The PWM also has an on-chip input called EXT_CLK and an output signal called PWMn_OUT_TRIGx.

44.4.1 PWM_An and PWM_Bn - External PWM Output Pair

These pins are the output pins of the PWM channels. These pins can be independent PWM signals or a complementary pair. When not needed as an output, they can be used as inputs to the input capture circuitry.

44.4.2 PWM_Xn - Auxiliary PWM Output signal

These pins are the auxiliary output pins of the PWM channels. They can be independent PWM signals. When not needed as an output, they can be used as inputs to the input capture circuitry or used to detect the polarity of the current flowing through the complementary circuit at deadtime correction.

44.4.3 FAULTn - Fault Inputs

These are input pins for disabling selected PWM outputs.

44.4.4 EXT_SYNC - External Synchronization Signal

These input signals allow a source external to the PWM to initialize the PWM counter. In this manner, the PWM can be synchronized to external circuitry.

44.4.5 EXT_FORCE - External Output Force Signal

This input signal allows a source external to the PWM to force an update of the PWM outputs. In this manner, the PWM can be synchronized to external circuitry.

44.4.6 PWMn_EXTA and PWMn_EXTB - Alternate PWM Control Signals

These pins allow an alternate source to control the PWM_An and PWM_Bn outputs. Typically, either the PWMn_EXTA or PWMn_EXTB input (depending on the state of MCTRL[IPOL]) is used for the generation of a complementary pair. Typical control signals include ADC conversion high/low limits, TMR outputs, GPIO inputs, and comparator outputs.

For pin input details, see eFlexPWM chip-specific information.

44.4.7 PWMn_OUT_TRIG0 and PWMn_OUT_TRIG1 - Output Triggers

These outputs allow the PWM submodules to control timing of ADC conversions. See the description of the SMnTCTRL[OUT_TRIG_EN] for information about how to enable these outputs and how the compare registers match up to the output triggers.

44.4.8 EXT_CLK - External Clock Signal

This signal allows a source external to the PWM (typically a timer or an off-chip source) to control the PWM clocking. In this manner, the PWM can be synchronized to the timer, or multiple chips can be synchronized to each other.

44.5 Resets

All PWM registers are reset to their default values upon any system reset.

The reset forces all registers to their reset states and tri-states the PWM outputs.

44.6 Interrupts

Each of the submodules within the eFlexPWM module can generate an interrupt from several sources. The fault logic can also generate interrupts. The interrupt service routine (ISR) must check the related interrupt enables and interrupt flags to determine the actual cause of the interrupt.

Table 379. Interrupt Summary

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
PWM_CMP0	SM0STS[CMPF]	SM0INTEN[CMPIE]	Submodule 0 compare interrupt	Compare event has occurred

Table continues on the next page...

Table 379. Interrupt Summary (continued)

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
PWM_CAP0	SM0STS[CFA1], SM0STS[CFA0], SM0STS[CFB1], SM0STS[CFB0], SM0STS[CFX1], SM0STS[CFX0]	SM0INTEN[CFA1IE], SM0INTEN[CFA0IE], SM0INTEN[CFB1IE], SM0INTEN[CFB0IE], SM0INTEN[CFX1IE], SM0INTEN[CFX0IE]	Submodule 0 input capture interrupt	Input capture event has occurred
PWM_RELOAD0	SM0STS[RF]	SM0INTEN[RIE]	Submodule 0 reload interrupt	Reload event has occurred
PWM_CMP1	SM1STS[CMPF]	SM1INTEN[CMPIE]	Submodule 1 compare interrupt	Compare event has occurred
PWM_CAP1	SM1STS[CFA1], SM1STS[CFA0], SM1STS[CFB1], SM1STS[CFB0], SM1STS[CFX1], SM1STS[CFX0]	SM1INTEN[CFA1IE], SM1INTEN[CFA0IE], SM1INTEN[CFB1IE], SM1INTEN[CFB0IE], SM1INTEN[CFX1IE], SM1INTEN[CFX0IE]	Submodule 1 input capture interrupt	Input capture event has occurred
PWM_RELOAD1	SM1STS[RF]	SM1INTEN[RIE]	Submodule 1 reload interrupt	Reload event has occurred
PWM_CMP2	SM2STS[CMPF]	SM2INTEN[CMPIE]	Submodule 2 compare interrupt	Compare event has occurred
PWM_CAP2	SM2STS[CFA1], SM2STS[CFA0], SM2STS[CFB1], SM2STS[CFB0], SM2STS[CFX1], SM2STS[CFX0]	SM2INTEN[CFA1IE], SM2INTEN[CFA0IE], SM2INTEN[CFB1IE], SM2INTEN[CFB0IE], SM2INTEN[CFX1IE], SM2INTEN[CFX0IE	Submodule 2 input capture interrupt	Input capture event has occurred
PWM_RELOAD2	SM2STS[RF]	SM2INTEN[RIE]	Submodule 2 reload interrupt	Reload event has occurred
PWM_CMP3	SM3STS[CMPF]	SM3INTEN[CMPIE]	Submodule 3 compare interrupt	Compare event has occurred
PWM_CAP3	SM3STS[CFA1], SM3STS[CFA0], SM3STS[CFB1],	SM3INTEN[CFA1IE], SM3INTEN[CFA0IE], SM3INTEN[CFB1IE],	Submodule 3 input capture interrupt	Input capture event has occurred

Table continues on the next page...

Table 379. Interrupt Summary (continued)

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
	SM3STS[CFB0], SM3STS[CFX1], SM3STS[CFX0]	SM3INTEN[CFB0IE], SM3INTEN[CFX1IE], SM3INTEN[CFX0IE		
PWM_RELOAD3	SM3STS[RF]	SM3INTEN[RIE]	Submodule 3 reload interrupt	Reload event has occurred
PWM_RERR	SM0STS[REF]	SM0INTEN[REIE]	Submodule 0 reload error interrupt	Reload error has occurred
	SM1STS[REF]	SM1INTEN[REIE]	Submodule 1 reload error interrupt	
	SM2STS[REF]	SM2INTEN[REIE]	Submodule 2 reload error interrupt	
	SM3STS[REF]	SM3INTEN[REIE]	Submodule 3 reload error interrupt	
PWM_FAULT	FSTS[FFLAG]	FCTRL[FIE]	Fault input interrupt	Fault condition has been detected

44.7 DMA

Each submodule can request a DMA read access for its capture FIFOs and a DMA write request for its double buffered VALx registers.

Table 380. DMA Summary

DMA Request	DMA Enable	Name	Description
Submodule 0 read request	SM0DMAEN[CX0DE]	SM0 Capture FIFO X0 read request	SM0CVAL0 contains a value to be read
	SM0DMAEN[CX1DE]	SM0 Capture FIFO X1 read request	SM0CVAL1 contains a value to be read
	SM0DMAEN[CA0DE]	SM0 Capture FIFO A0 read request	SM0CVAL2 contains a value to be read
	SM0DMAEN[CA1DE]	SM0 Capture FIFO A1 read request	SM0CVAL3 contains a value to be read
	SM0DMAEN[CB0DE]	SM0 Capture FIFO B0 read request	SM0CVAL4 contains a value to be read
	SM0DMAEN[CB1DE]	SM0 Capture FIFO B1 read request	SM0CVAL5 contains a value to be read

Table continues on the next page...

Table 380. DMA Summary (continued)

DMA Request	DMA Enable	Name	Description
	SM0DMAEN[CAPTDE]	SM0 Capture FIFO read request source select	Selects source of submodule0 read DMA request
Submodule 0 write request	SM0DMAEN[VALDE]	SM0VALx write request	SM0VALx registers need to be updated
Submodule 1 read request	SM1DMAEN[CX0DE]	SM1 Capture FIFO X0 read request	SM1CVAL0 contains a value to be read
	SM1DMAEN[CX1DE]	SM1 Capture FIFO X! read request	SM1CVAL1 contains a value to be read
	SM1DMAEN[CA0DE]	SM1Capture FIFO A0 read request	SM1CVAL2 contains a value to be read
	SM1DMAEN[CA1DE]	SM1 Capture FIFO A1 read request	SM1CVAL3 contains a value to be read
	SM1DMAEN[CB0DE]	SM1 Capture FIFO B0 read request	SM1CVAL4 contains a value to be read
	SM1DMAEN[CB1DE]	SM1 Capture FIFO B1 read request	SM!CVAL5 contains a value to be read
	SM1DMAEN[CAPTDE]	SM1 Capture FIFO read request source select	Selects source of submodule1 read DMA request
Submodule 1 write request	SM1DMAEN[VALDE]	SM1VALx write request	SM1VALx registers need to be updated
Submodule 2 read request	SM2DMAEN[CX0DE]	SM2 Capture FIFO X0 read request	SM2CVAL0 contains a value to be read
	SM2DMAEN[CX1DE]	SM2 Capture FIFO X! read request	SM2CVAL1 contains a value to be read
	SM2DMAEN[CA0DE]	SM2 Capture FIFO A0 read request	SM2CVAL2 contains a value to be read
	SM2DMAEN[CA1DE]	SM2 Capture FIFO A1 read request	SM2CVAL3 contains a value to be read
	SM2DMAEN[CB0DE]	SM2 Capture FIFO B0 read request	SM2CVAL4 contains a value to be read
	SM2DMAEN[CB1DE]	SM2 Capture FIFO B1 read request	SM2CVAL5 contains a value to be read
	SM2DMAEN[CAPTDE]	SM2 Capture FIFO read request source select	Selects source of submodule2 read DMA request

Table continues on the next page...

Table 380. DMA Summary (continued)

DMA Request	DMA Enable	Name	Description
Submodule 2 write request	SM2DMAEN[VALDE]	SM2VALx write request	SM2VALx registers need to be updated
Submodule 3 read request	SM3DMAEN[CX0DE]	SM3 Capture FIFO X0 read request	SM3CVAL0 contains a value to be read
	SM3DMAEN[CX1DE]	SM3 Capture FIFO X! read request	SM3CVAL1 contains a value to be read
	SM3DMAEN[CA0DE]	SM3 Capture FIFO A0 read request	SM3CVAL2 contains a value to be read
	SM3DMAEN[CA1DE]	SM3 Capture FIFO A1 read request	SM3CVAL3 contains a value to be read
	SM3DMAEN[CB0DE]	SM3 Capture FIFO B0 read request	SM3CVAL4 contains a value to be read
	SM3DMAEN[CB1DE]	SM3 Capture FIFO B1 read request	SM3CVAL5 contains a value to be read
	SM3DMAEN[CAPTDE]	SM3 Capture FIFO read request source select	Selects source of submodule3 read DMA request
Submodule 3 write request	SM3DMAEN[VALDE]	SM3VALx write request	SM3VALx registers need to be updated

44.8 PWM register descriptions

The address of a register is the sum of a base address and an address offset. The base address is defined at the core level, and the address offset is defined at the module level. The PWM module has a set of registers for each PWM submodule, for the configuration logic, and for each fault channel.

NOTE

While the registers are 16-bit wide, they can be accessed in pairs as 32-bit registers, if the pairs are word-aligned.

Submodule registers are repeated for each PWM submodule. To designate which submodule they are in, register names are prefixed with SMx(x=0,1,2,3). The base address of submodule 0 is the same as the base address for the PWM module as a whole. The base address of submodule 1 is offset 0x60 from the base address for the PWM module as a whole. This 0x60 offset is based on the number of registers in a submodule. The base address of submodule 2 is equal to the base address of submodule 1 plus this same 0x60 offset. The pattern repeats for the base address of submodule 3.

The base address of the configuration registers is equal to the base address of the PWM module as a whole plus an offset of 0x180.

Fault channel registers are repeated for each fault channel. To designate which fault channel they are in, register names are prefixed with F0 and F1. The base address of fault channel 0 is equal to the base address of the PWM module as a whole plus an offset of 0x18C. The base address of fault channel 1 is the base address of fault channel 0 + 0x4. This 0x4 offset is based on the number of registers in a fault channel. Each of the four fields in the fault channel registers corresponds to fault inputs 3-0.

44.8.1 PWM memory map

PWM0 base address: 400C_3000h

PWM1 base address: 400C_5000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Counter Register (SM0CNT)	16	RO	0000
2	Initial Count Register (SM0INIT)	16	RW	0000
4	Control 2 Register (SM0CTRL2)	16	See section	0000
6	Control Register (SM0CTRL)	16	See section	0400
A	Value Register 0 (SM0VAL0)	16	RW	0000
C	Fractional Value Register 1 (SM0FRACVAL1)	16	See section	0000
E	Value Register 1 (SM0VAL1)	16	RW	0000
10	Fractional Value Register 2 (SM0FRACVAL2)	16	See section	0000
12	Value Register 2 (SM0VAL2)	16	RW	0000
14	Fractional Value Register 3 (SM0FRACVAL3)	16	See section	0000
16	Value Register 3 (SM0VAL3)	16	RW	0000
18	Fractional Value Register 4 (SM0FRACVAL4)	16	See section	0000
1A	Value Register 4 (SM0VAL4)	16	RW	0000
1C	Fractional Value Register 5 (SM0FRACVAL5)	16	See section	0000
1E	Value Register 5 (SM0VAL5)	16	RW	0000
20	Fractional Control Register (SM0FRCTRL)	16	See section	0000
22	Output Control Register (SM0OCTRL)	16	See section	0000
24	Status Register (SM0STS)	16	See section	0000
26	Interrupt Enable Register (SM0INTEN)	16	See section	0000
28	DMA Enable Register (SM0DMAEN)	16	See section	0000
2A	Output Trigger Control Register (SM0TCTRL)	16	See section	0000
2C	Fault Disable Mapping Register 0 (SM0DISMAP0)	16	See section	FFFF

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
30	Deadtime Count Register 0 (SM0DTCNT0)	16	See section	07FF
32	Deadtime Count Register 1 (SM0DTCNT1)	16	See section	07FF
34	Capture Control A Register (SM0CAPTCTRLA)	16	See section	0000
36	Capture Compare A Register (SM0CAPTCOMPA)	16	See section	0000
38	Capture Control B Register (SM0CAPTCTRLB)	16	See section	0000
3A	Capture Compare B Register (SM0CAPTCOMPB)	16	See section	0000
3C	Capture Control X Register (SM0CAPTCTRLX)	16	See section	0000
3E	Capture Compare X Register (SM0CAPTCOMPX)	16	See section	0000
40	Capture Value 0 Register (SM0CVAL0)	16	RO	0000
42	Capture Value 0 Cycle Register (SM0CVAL0CYC)	16	See section	0000
44	Capture Value 1 Register (SM0CVAL1)	16	RO	0000
46	Capture Value 1 Cycle Register (SM0CVAL1CYC)	16	See section	0000
48	Capture Value 2 Register (SM0CVAL2)	16	RO	0000
4A	Capture Value 2 Cycle Register (SM0CVAL2CYC)	16	See section	0000
4C	Capture Value 3 Register (SM0CVAL3)	16	RO	0000
4E	Capture Value 3 Cycle Register (SM0CVAL3CYC)	16	See section	0000
50	Capture Value 4 Register (SM0CVAL4)	16	RO	0000
52	Capture Value 4 Cycle Register (SM0CVAL4CYC)	16	See section	0000
54	Capture Value 5 Register (SM0CVAL5)	16	RO	0000
56	Capture Value 5 Cycle Register (SM0CVAL5CYC)	16	See section	0000
5A	Capture PWMA Input Filter Register (SM0CAPTFILTA)	16	See section	0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
5C	Capture PWMB Input Filter Register (SM0CAPTFILTB)	16	See section	0000
5E	Capture PWMX Input Filter Register (SM0CAPTFILTX)	16	See section	0000
60	Counter Register (SM1CNT)	16	RO	0000
62	Initial Count Register (SM1INIT)	16	RW	0000
64	Control 2 Register (SM1CTRL2)	16	See section	0000
66	Control Register (SM1CTRL)	16	See section	0400
6A	Value Register 0 (SM1VAL0)	16	RW	0000
6C	Fractional Value Register 1 (SM1FRACVAL1)	16	See section	0000
6E	Value Register 1 (SM1VAL1)	16	RW	0000
70	Fractional Value Register 2 (SM1FRACVAL2)	16	See section	0000
72	Value Register 2 (SM1VAL2)	16	RW	0000
74	Fractional Value Register 3 (SM1FRACVAL3)	16	See section	0000
76	Value Register 3 (SM1VAL3)	16	RW	0000
78	Fractional Value Register 4 (SM1FRACVAL4)	16	See section	0000
7A	Value Register 4 (SM1VAL4)	16	RW	0000
7C	Fractional Value Register 5 (SM1FRACVAL5)	16	See section	0000
7E	Value Register 5 (SM1VAL5)	16	RW	0000
80	Fractional Control Register (SM1FRCTRL)	16	See section	0000
82	Output Control Register (SM1OCTRL)	16	See section	0000
84	Status Register (SM1STS)	16	See section	0000
86	Interrupt Enable Register (SM1INTEN)	16	See section	0000
88	DMA Enable Register (SM1DMAEN)	16	See section	0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
8A	Output Trigger Control Register (SM1TCTRL)	16	See section	0000
8C	Fault Disable Mapping Register 0 (SM1DISMAP0)	16	See section	FFFF
90	Deadtime Count Register 0 (SM1DTCNT0)	16	See section	07FF
92	Deadtime Count Register 1 (SM1DTCNT1)	16	See section	07FF
94	Capture Control A Register (SM1CAPTCTRLA)	16	See section	0000
96	Capture Compare A Register (SM1CAPTCOMPA)	16	See section	0000
98	Capture Control B Register (SM1CAPTCTRLB)	16	See section	0000
9A	Capture Compare B Register (SM1CAPTCOMPB)	16	See section	0000
9C	Capture Control X Register (SM1CAPTCTRLX)	16	See section	0000
9E	Capture Compare X Register (SM1CAPTCOMPX)	16	See section	0000
A0	Capture Value 0 Register (SM1CVAL0)	16	RO	0000
A2	Capture Value 0 Cycle Register (SM1CVAL0CYC)	16	See section	0000
A4	Capture Value 1 Register (SM1CVAL1)	16	RO	0000
A6	Capture Value 1 Cycle Register (SM1CVAL1CYC)	16	See section	0000
A8	Capture Value 2 Register (SM1CVAL2)	16	RO	0000
AA	Capture Value 2 Cycle Register (SM1CVAL2CYC)	16	See section	0000
AC	Capture Value 3 Register (SM1CVAL3)	16	RO	0000
AE	Capture Value 3 Cycle Register (SM1CVAL3CYC)	16	See section	0000
B0	Capture Value 4 Register (SM1CVAL4)	16	RO	0000
B2	Capture Value 4 Cycle Register (SM1CVAL4CYC)	16	See section	0000
B4	Capture Value 5 Register (SM1CVAL5)	16	RO	0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
B6	Capture Value 5 Cycle Register (SM1CVAL5CYC)	16	See section	0000
B8	Phase Delay Register (SM1PHASEDLY)	16	RW	0000
BA	Capture PWMA Input Filter Register (SM1CAPTFILTA)	16	See section	0000
BC	Capture PWMB Input Filter Register (SM1CAPTFILTB)	16	See section	0000
BE	Capture PWMX Input Filter Register (SM1CAPTFILTX)	16	See section	0000
C0	Counter Register (SM2CNT)	16	RO	0000
C2	Initial Count Register (SM2INIT)	16	RW	0000
C4	Control 2 Register (SM2CTRL2)	16	See section	0000
C6	Control Register (SM2CTRL)	16	See section	0400
CA	Value Register 0 (SM2VAL0)	16	RW	0000
CC	Fractional Value Register 1 (SM2FRACVAL1)	16	See section	0000
CE	Value Register 1 (SM2VAL1)	16	RW	0000
D0	Fractional Value Register 2 (SM2FRACVAL2)	16	See section	0000
D2	Value Register 2 (SM2VAL2)	16	RW	0000
D4	Fractional Value Register 3 (SM2FRACVAL3)	16	See section	0000
D6	Value Register 3 (SM2VAL3)	16	RW	0000
D8	Fractional Value Register 4 (SM2FRACVAL4)	16	See section	0000
DA	Value Register 4 (SM2VAL4)	16	RW	0000
DC	Fractional Value Register 5 (SM2FRACVAL5)	16	See section	0000
DE	Value Register 5 (SM2VAL5)	16	RW	0000
E0	Fractional Control Register (SM2FRCTRL)	16	See section	0000
E2	Output Control Register (SM2OCTRL)	16	See section	0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
E4	Status Register (SM2STS)	16	See section	0000
E6	Interrupt Enable Register (SM2INTEN)	16	See section	0000
E8	DMA Enable Register (SM2DMAEN)	16	See section	0000
EA	Output Trigger Control Register (SM2TCTRL)	16	See section	0000
EC	Fault Disable Mapping Register 0 (SM2DISMAP0)	16	See section	FFFF
F0	Deadtime Count Register 0 (SM2DTCNT0)	16	See section	07FF
F2	Deadtime Count Register 1 (SM2DTCNT1)	16	See section	07FF
F4	Capture Control A Register (SM2CAPTCTRLA)	16	See section	0000
F6	Capture Compare A Register (SM2CAPTCOMPA)	16	See section	0000
F8	Capture Control B Register (SM2CAPTCTRLB)	16	See section	0000
FA	Capture Compare B Register (SM2CAPTCOMPB)	16	See section	0000
FC	Capture Control X Register (SM2CAPTCTRLX)	16	See section	0000
FE	Capture Compare X Register (SM2CAPTCOMPX)	16	See section	0000
100	Capture Value 0 Register (SM2CVAL0)	16	RO	0000
102	Capture Value 0 Cycle Register (SM2CVAL0CYC)	16	See section	0000
104	Capture Value 1 Register (SM2CVAL1)	16	RO	0000
106	Capture Value 1 Cycle Register (SM2CVAL1CYC)	16	See section	0000
108	Capture Value 2 Register (SM2CVAL2)	16	RO	0000
10A	Capture Value 2 Cycle Register (SM2CVAL2CYC)	16	See section	0000
10C	Capture Value 3 Register (SM2CVAL3)	16	RO	0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
10E	Capture Value 3 Cycle Register (SM2CVAL3CYC)	16	See section	0000
110	Capture Value 4 Register (SM2CVAL4)	16	RO	0000
112	Capture Value 4 Cycle Register (SM2CVAL4CYC)	16	See section	0000
114	Capture Value 5 Register (SM2CVAL5)	16	RO	0000
116	Capture Value 5 Cycle Register (SM2CVAL5CYC)	16	See section	0000
118	Phase Delay Register (SM2PHASEDLY)	16	RW	0000
11A	Capture PWMA Input Filter Register (SM2CAPTFILTA)	16	See section	0000
11C	Capture PWMB Input Filter Register (SM2CAPTFILTB)	16	See section	0000
11E	Capture PWMX Input Filter Register (SM2CAPTFILTX)	16	See section	0000
120	Counter Register (SM3CNT)	16	RO	0000
122	Initial Count Register (SM3INIT)	16	RW	0000
124	Control 2 Register (SM3CTRL2)	16	See section	0000
126	Control Register (SM3CTRL)	16	See section	0400
12A	Value Register 0 (SM3VAL0)	16	RW	0000
12C	Fractional Value Register 1 (SM3FRACVAL1)	16	See section	0000
12E	Value Register 1 (SM3VAL1)	16	RW	0000
130	Fractional Value Register 2 (SM3FRACVAL2)	16	See section	0000
132	Value Register 2 (SM3VAL2)	16	RW	0000
134	Fractional Value Register 3 (SM3FRACVAL3)	16	See section	0000
136	Value Register 3 (SM3VAL3)	16	RW	0000
138	Fractional Value Register 4 (SM3FRACVAL4)	16	See section	0000
13A	Value Register 4 (SM3VAL4)	16	RW	0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
13C	Fractional Value Register 5 (SM3FRACVAL5)	16	See section	0000
13E	Value Register 5 (SM3VAL5)	16	RW	0000
140	Fractional Control Register (SM3FRCTRL)	16	See section	0000
142	Output Control Register (SM3OCTRL)	16	See section	0000
144	Status Register (SM3STS)	16	See section	0000
146	Interrupt Enable Register (SM3INTEN)	16	See section	0000
148	DMA Enable Register (SM3DMAEN)	16	See section	0000
14A	Output Trigger Control Register (SM3TCTRL)	16	See section	0000
14C	Fault Disable Mapping Register 0 (SM3DISMAP0)	16	See section	FFFF
150	Deadtime Count Register 0 (SM3DTCNT0)	16	See section	07FF
152	Deadtime Count Register 1 (SM3DTCNT1)	16	See section	07FF
154	Capture Control A Register (SM3CAPTCTRLA)	16	See section	0000
156	Capture Compare A Register (SM3CAPTCOMPA)	16	See section	0000
158	Capture Control B Register (SM3CAPTCTRLB)	16	See section	0000
15A	Capture Compare B Register (SM3CAPTCOMPB)	16	See section	0000
15C	Capture Control X Register (SM3CAPTCTRLX)	16	See section	0000
15E	Capture Compare X Register (SM3CAPTCOMPX)	16	See section	0000
160	Capture Value 0 Register (SM3CVAL0)	16	RO	0000
162	Capture Value 0 Cycle Register (SM3CVAL0CYC)	16	See section	0000
164	Capture Value 1 Register (SM3CVAL1)	16	RO	0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
166	Capture Value 1 Cycle Register (SM3CVAL1CYC)	16	See section	0000
168	Capture Value 2 Register (SM3CVAL2)	16	RO	0000
16A	Capture Value 2 Cycle Register (SM3CVAL2CYC)	16	See section	0000
16C	Capture Value 3 Register (SM3CVAL3)	16	RO	0000
16E	Capture Value 3 Cycle Register (SM3CVAL3CYC)	16	See section	0000
170	Capture Value 4 Register (SM3CVAL4)	16	RO	0000
172	Capture Value 4 Cycle Register (SM3CVAL4CYC)	16	See section	0000
174	Capture Value 5 Register (SM3CVAL5)	16	RO	0000
176	Capture Value 5 Cycle Register (SM3CVAL5CYC)	16	See section	0000
178	Phase Delay Register (SM3PHASEDLY)	16	RW	0000
17A	Capture PWMA Input Filter Register (SM3CAPTFILTA)	16	See section	0000
17C	Capture PWMB Input Filter Register (SM3CAPTFILTB)	16	See section	0000
17E	Capture PWMX Input Filter Register (SM3CAPTFILTX)	16	See section	0000
180	Output Enable Register (OUTEN)	16	See section	0000
182	Mask Register (MASK)	16	See section	0000
184	Software Controlled Output Register (SWCOUT)	16	See section	0000
186	PWM Source Select Register (DTSRCSEL)	16	RW	0000
188	Master Control Register (MCTRL)	16	See section	0000
18A	Master Control 2 Register (MCTRL2)	16	See section	0000
18C	Fault Control Register (FCTRL0)	16	RW	0000
18E	Fault Status Register (FSTS0)	16	See section	0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
190	Fault Filter Register (FFILT0)	16	See section	0000
192	Fault Test Register (FTST0)	16	See section	0000
194	Fault Control 2 Register (FCTRL20)	16	See section	0000

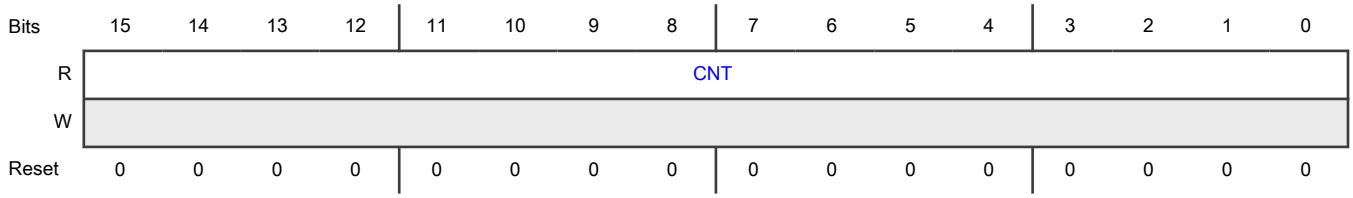
44.8.1.1 Counter Register (SM0CNT - SM3CNT)

This read-only register displays the state of the signed 16-bit submodule counter. This register is not byte accessible. Writing this register generates bus transfer error.

Offset

Register	Offset
SM0CNT	0h
SM1CNT	60h
SM2CNT	C0h
SM3CNT	120h

Diagram



Fields

Field	Description
15-0 CNT	Counter Register Bits

44.8.1.2 Initial Count Register (SM0INIT - SM3INIT)

The 16-bit signed value in this buffered, read/write register defines the initial count value for the PWM in PWM clock periods. This is the value loaded into the submodule counter when local sync, master sync, or master reload is asserted (based on the value of CTRL2[INIT_SEL]) or when CTRL2[FORCE] is asserted and force init is enabled. For PWM operation, the buffered contents of this register are loaded into the counter at the start of every PWM cycle. This register is not byte accessible.

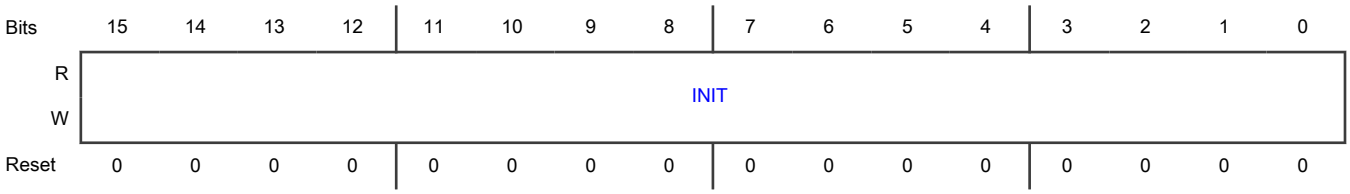
NOTE

The INIT register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. This register cannot be written when MCTRL[LDOK] is set. Reading INIT reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Offset

Register	Offset
SM0INIT	2h
SM1INIT	62h
SM2INIT	C2h
SM3INIT	122h

Diagram



Fields

Field	Description
15-0 INIT	Initial Count Register Bits

44.8.1.3 Control 2 Register (SM0CTRL2 - SM3CTRL2)

Contains control fields for clock select and forcing output and initialization.

Offset

Register	Offset
SM0CTRL2	4h
SM1CTRL2	64h
SM2CTRL2	C4h
SM3CTRL2	124h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DBGEN	WAITEN	INDEP	PWM23_I...	PWM45_I...	PWMX_IN...	INIT_SEL		FRCEN	0	FORCE_SEL			RELOAD_...	CLK_SEL	
W										FORCE						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
15 DBGEN	Debug Enable When set to one, the PWM will continue to run while the chip is in debug mode. If the device enters debug mode and this bit is zero, then the PWM outputs will be disabled until debug mode is exited. At that point the PWM pins will resume operation as programmed in the PWM registers.
14 WAITEN	Sleep Enable When set to one, the PWM will continue to run while the chip is in Sleep mode. In this mode, the peripheral clock continues to run but the CPU clock does not. If the device enters Sleep mode and this bit is zero, then the PWM outputs will be disabled until Sleep mode is exited. At that point the PWM pins will resume operation as programmed in the PWM registers.
13 INDEP	Independent or Complementary Pair Operation This bit determines if the PWM_A and PWM_B channels will be independent PWMs or a complementary PWM pair. 0 - PWM_A and PWM_B form a complementary PWM pair. 1 - PWM_A and PWM_B outputs are independent PWMs.
12 PWM23_INIT	PWM23 Initial Value This read/write bit determines the initial value for PWM23 and the value to which it is forced when FORCE_INIT is asserted.
11 PWM45_INIT	PWM45 Initial Value This read/write bit determines the initial value for PWM45 and the value to which it is forced when FORCE_INIT is asserted.
10 PWMX_INIT	PWM_X Initial Value This read/write bit determines the initial value for PWM_X and the value to which it is forced when FORCE_INIT is asserted.
9-8 INIT_SEL	Initialization Control Select These read/write bits control the source of the INIT signal which goes to the counter. 00 - Local sync (PWM_X) causes initialization. 01 - Master reload from submodule 0 causes initialization. This setting should not be used in submodule 0 as it will force the INIT signal to logic 0. The submodule counter will only reinitialize when a master reload occurs.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	10 - Master sync from submodule 0 causes initialization. This setting should not be used in submodule 0 as it will force the INIT signal to logic 0. 11 - EXT_SYNC causes initialization.
7 FRCEN	FRCEN This bit allows the CTRL2[FORCE] signal to initialize the counter without regard to the signal selected by CTRL2[INIT_SEL]. This is a software controlled initialization. A forced initialization will also assert the register reload if MCTRL[LDOK] is set. 0 - Initialization from a FORCE_OUT is disabled. 1 - Initialization from a FORCE_OUT is enabled.
6 FORCE	Force Initialization If CTRL2[FORCE_SEL] is set to 000, writing a 1 to this bit results in a FORCE_OUT event. This causes the following actions to be taken: • The PWM_A and PWM_B output pins will assume values based on DTSRCSEL[SMxSEL23] and DTSRCSEL[SMxSEL45]. • If CTRL2[FRCEN] is set, the counter value will be initialized with the INIT register value.
5-3 FORCE_SEL	This read/write bit determines the source of the FORCE OUTPUT signal for this submodule. 000 - The local force signal, CTRL2[FORCE], from this submodule is used to force updates. 001 - The master force signal from submodule 0 is used to force updates. This setting should not be used in submodule 0 as it will hold the FORCE OUTPUT signal to logic 0. 010 - The local reload signal from this submodule is used to force updates without regard to the state of LDOK. 011 - The master reload signal from submodule0 is used to force updates if LDOK is set. This setting should not be used in submodule0 as it will hold the FORCE OUTPUT signal to logic 0. 100 - The local sync signal from this submodule is used to force updates. 101 - The master sync signal from submodule0 is used to force updates. This setting should not be used in submodule0 as it will hold the FORCE OUTPUT signal to logic 0. 110 - The external force signal, EXT_FORCE, from outside the PWM module causes updates. 111 - The external sync signal, EXT_SYNC, from outside the PWM module causes updates.
2 RELOAD_SEL	Reload Source Select This read/write bit determines the source of the RELOAD signal for this submodule. When this bit is set, MCTRL[LDOK[0]] for submodule 0 should be used since the local MCTRL[LDOK] will be ignored. 0 - The local RELOAD signal is used to reload registers. 1 - The master RELOAD signal (from submodule 0) is used to reload registers. This setting should not be used in submodule 0 as it will force the RELOAD signal to logic 0.
1-0	Clock Source Select

Table continues on the next page...

Table continued from the previous page...

Field	Description
CLK_SEL	These read/write bits determine the source of the clock signal for this submodule. 00 - The IPBus clock is used as the clock for the local prescaler and counter. 01 - EXT_CLK is used as the clock for the local prescaler and counter. 10 - Submodule 0's clock (AUX_CLK) is used as the source clock for the local prescaler and counter. This setting should not be used in submodule 0 as it will force the clock to logic 0. 11 - reserved

44.8.1.4 Control Register (SM0CTRL - SM3CTRL)

Includes control settings for timing, loading, and buffering.

Offset

Register	Offset
SM0CTRL	6h
SM1CTRL	66h
SM2CTRL	C6h
SM3CTRL	126h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LDFQ				HALF	FULL	DT		COMP MODE	PRSC			SPLIT	LDMOD	DBLX	DBLEN
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
15-12 LDFQ	Load Frequency These buffered read/write bits select the PWM load frequency. Reset clears LDFQ, selecting loading every PWM opportunity. A PWM opportunity is determined by HALF and FULL. **NOTE** LDFQ takes effect when the current load cycle is complete, regardless of the state of MCTRL[LDOK]. Reading LDFQ reads the buffered values and not necessarily the values currently in effect. 0000 - Every PWM opportunity

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0001 - Every 2 PWM opportunities 0010 - Every 3 PWM opportunities 0011 - Every 4 PWM opportunities 0100 - Every 5 PWM opportunities 0101 - Every 6 PWM opportunities 0110 - Every 7 PWM opportunities 0111 - Every 8 PWM opportunities 1000 - Every 9 PWM opportunities 1001 - Every 10 PWM opportunities 1010 - Every 11 PWM opportunities 1011 - Every 12 PWM opportunities 1100 - Every 13 PWM opportunities 1101 - Every 14 PWM opportunities 1110 - Every 15 PWM opportunities 1111 - Every 16 PWM opportunities
11 HALF	Half Cycle Reload This read/write bit enables half-cycle reloads. A half cycle is defined by when the submodule counter matches the VAL0 register and does not have to be half way through the PWM cycle. 0 - Half-cycle reloads disabled. 1 - Half-cycle reloads enabled.
10 FULL	Full Cycle Reload This read/write bit enables full-cycle reloads. A full cycle is defined by when the submodule counter matches the VAL1 register. Either CTRL[HALF] or CTRL[FULL] must be set in order to move the buffered data into the registers used by the PWM generators or CTRL[LDMOD] must be set. If both CTRL[HALF] and CTRL[FULL] are set, then reloads can occur twice per cycle. 0 - Full-cycle reloads disabled. 1 - Full-cycle reloads enabled.
9-8 DT	Deadtime These read only bits reflect the sampled values of the PWM_X input at the end of each deadtime. Sampling occurs at the end of deadtime 0 for DT[0] and the end of deadtime 1 for DT[1]. Reset clears these bits.
7 COMPMODE	Compare Mode This bit controls how comparisons are made between the VAL* registers and the PWM submodule counter. This bit can only be written one time after which it requires a reset to release the bit for writing again.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - The VAL* registers and the PWM counter are compared using an "equal to" method. This means that PWM edges are only produced when the counter is equal to one of the VAL* register values. This implies that a PWMA output that is high at the end of a period will maintain this state until a match with VAL3 clears the output in the following period. 1 - The VAL* registers and the PWM counter are compared using an "equal to or greater than" method. This means that PWM edges are produced when the counter is equal to or greater than one of the VAL* register values. This implies that a PWMA output that is high at the end of a period could go low at the start of the next period if the starting counter value is greater than (but not necessarily equal to) the new VAL3 value.
6-4 PRSC	Prescaler These buffered read/write bits select the divide ratio of the PWM clock frequency selected by CTRL2[CLK_SEL]. **NOTE** Reading CTRL[PRSC] reads the buffered values and not necessarily the values currently in effect. CTRL[PRSC] takes effect at the beginning of the next PWM cycle and only when the load okay bit, MCTRL[LDOK], is set or CTRL[LDMOD] is set. This field cannot be written when MCTRL[LDOK] is set. 000 - Prescaler 1. PWM clock frequency = f_{clk} 001 - Prescaler 2. PWM clock frequency = f_{clk} /2 010 - Prescaler 4. PWM clock frequency = f_{clk} /4 011 - Prescaler 8. PWM clock frequency = f_{clk} /8 100 - Prescaler 16. PWM clock frequency = f_{clk} /16 101 - Prescaler 32. PWM clock frequency = f_{clk} /32 110 - Prescaler 64. PWM clock frequency = f_{clk} /64 111 - Prescaler 128. PWM clock frequency = f_{clk} /128
3 SPLIT	Split the DBLPWM signal to PWMA and PWMB This read/write bit is only used in independent mode when DBLEN is set. This bit allows the two PWM pulses generated by DBLEN to be split with one pulse on PWMA and one on PWMB. The two pulses within the same PWM period are created by an XOR function of the PWMA and PWMB sources. The splitting function causes PWMA to output the pulse that occurs when the PWMA source is 1 and the PWMB source is 0. The PWMB output occurs when the PWMB source is 1 and the PWMA source is 0. (See Double Switching PWMs .) 0 - DBLPWM is not split. PWMA and PWMB each have double pulses. 1 - DBLPWM is split to PWMA and PWMB.
2 LDMOD	Load Mode Select This read/write bit selects the timing of loading the buffered registers for this submodule. 0 - Buffered registers of this submodule are loaded and take effect at the next PWM reload if MCTRL[LDOK] is set.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Buffered registers of this submodule are loaded and take effect immediately upon MCTRL[LDOK] being set. In this case it is not necessary to set CTRL[FULL] or CTRL[HALF].
1 DBLX	PWMX Double Switching Enable This read/write bit enables the double switching behavior on PWMX. When this bit is set, the PWMX output shall be the exclusive OR combination of PWMA and PWMB prior to polarity and masking considerations. 0 - PWMX double pulse disabled. 1 - PWMX double pulse enabled.
0 DBLEN	Double Switching Enable This read/write bit enables the double switching PWM behavior(See Double Switching PWMs). 0 - Double switching disabled. 1 - Double switching enabled.

44.8.1.5 Value Register 0 (SM0VAL0 - SM3VAL0)

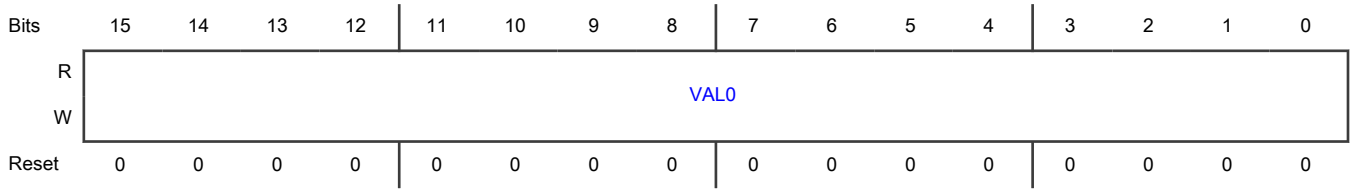
NOTE

The VAL0 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL0 cannot be written when MCTRL[LDOK] is set. Reading VAL0 reads the value in a buffer. It is not necessarily the value the PWM generator is currently using.

Offset

Register	Offset
SM0VAL0	Ah
SM1VAL0	6Ah
SM2VAL0	CAh
SM3VAL0	12Ah

Diagram



Fields

Field	Description
15-0 VAL0	Value Register 0 The 16-bit signed value in this buffered, read/write register defines the mid-cycle reload point for the PWM in PWM clock periods. This value also defines when the PWM_X signal is set and the local sync signal is reset. This register is not byte accessible. **NOTE** The actual behavior takes affect when counter equal VAL0+1.

44.8.1.6 Fractional Value Register 1 (SM0FRACVAL1 - SM3FRACVAL1)

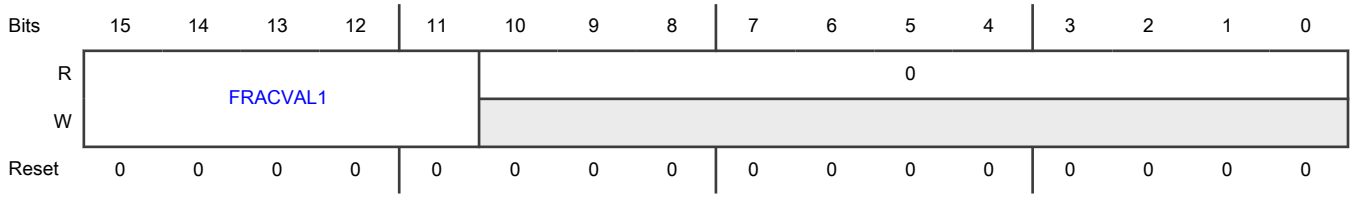
NOTE

The FRACVAL1 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL1 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL1 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Offset

Register	Offset
SM0FRACVAL1	Ch
SM1FRACVAL1	6Ch
SM2FRACVAL1	CCh
SM3FRACVAL1	12Ch

Diagram



Fields

Field	Description
15-11 FRACVAL1	Fractional Value 1 Register These bits act as a fractional addition to the value in the VAL1 register which controls the PWM period width. The PWM period is computed in terms of IPBus clock cycles. This fractional portion is accumulated at the end of every cycle until an additional whole IPBus cycle is reached. At this time the value being used for VAL1 is temporarily incremented and the PWM cycle is extended by one clock period to compensate for the accumulated fractional values.
10-0 —	RESERVED

44.8.1.7 Value Register 1 (SM0VAL1 - SM3VAL1)

NOTE

The VAL1 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL1 cannot be written when MCTRL[LDOK] is set. Reading VAL1 reads the value in a buffer. It is not necessarily the value the PWM generator is currently using.

NOTE

When using FRACVAL1, limit the maximum value of VAL1 to 0xFFFE for unsigned applications or to 0x7FFE for signed applications, to avoid counter rollovers caused by accumulating the fractional period defined by FRACVAL1.

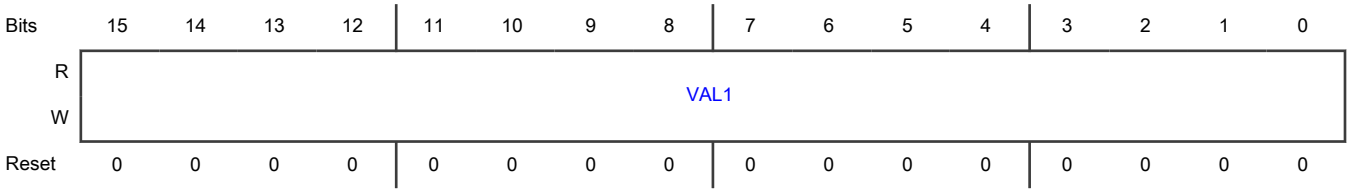
NOTE

If the VAL1 register defines the timer period (Local Sync is selected as the counter initialization signal), a 100% duty cycle cannot be achieved on the PWMX output. After the count reaches VAL1, the PWMX output is low for a minimum of one count every cycle. When the Master Sync signal (only originated by the Local Sync from sub-module 0) is used to control the timer period, the VAL1 register can be free for other functions such as PWM generation without the duty cycle limitation.

Offset

Register	Offset
SM0VAL1	Eh
SM1VAL1	6Eh
SM2VAL1	CEh
SM3VAL1	12Eh

Diagram



Fields

Field	Description
15-0 VAL1	Value Register 1 The 16-bit signed value written to this buffered, read/write register defines the modulo count value (maximum count) for the submodule counter. Upon reaching this count value, the counter reloads itself with the contents of the INIT register and asserts the local sync signal while resetting PWM_X. This register is not byte accessible. **NOTE** The actual behavior takes affect when counter equal VAL1+1.

44.8.1.8 Fractional Value Register 2 (SM0FRACVAL2 - SM3FRACVAL2)

NOTE

The FRACVAL2 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL2 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL2 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

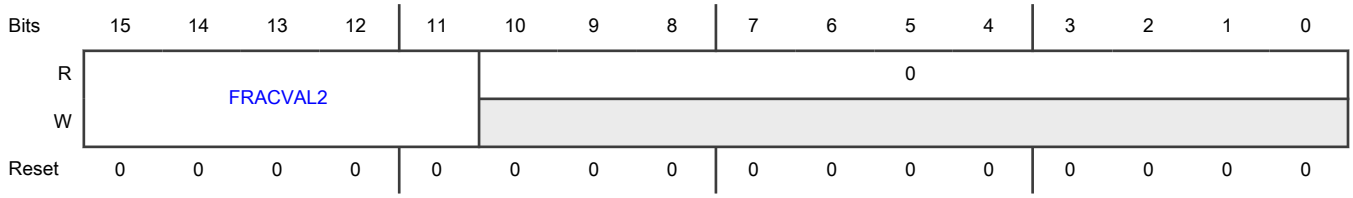
NOTE

FRCTRL[FRAC23_EN] should be set to 0 when the values of VAL2 and VAL3 cause the high or low time of the PWM output to be 3 cycles or less.

Offset

Register	Offset
SM0FRACVAL2	10h
SM1FRACVAL2	70h
SM2FRACVAL2	D0h
SM3FRACVAL2	130h

Diagram



Fields

Field	Description
15-11 FRACVAL2	Fractional Value 2 These bits act as a fractional addition to the value in the VAL2 register which controls the PWM_A turn on timing. It is also used to control the fractional addition to the turn off delay of PWM_B when MCTRL[IPOLx]=0 in complementary mode, CTRL2[INDEP]=0.
10-0 —	RESERVED

44.8.1.9 Value Register 2 (SM0VAL2 - SM3VAL2)

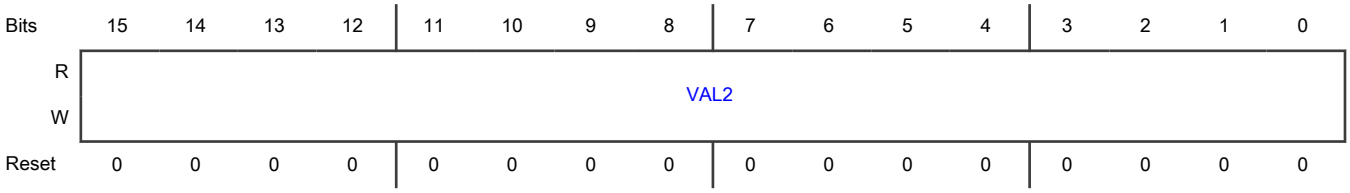
NOTE

The VAL2 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL2 cannot be written when MCTRL[LDOK] is set. Reading VAL2 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Offset

Register	Offset
SM0VAL2	12h
SM1VAL2	72h
SM2VAL2	D2h
SM3VAL2	132h

Diagram



Fields

Field	Description
15-0 VAL2	Value Register 2 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM23 high. This register is not byte accessible. **NOTE** The actual behavior takes affect when counter equal VAL2+1.

44.8.1.10 Fractional Value Register 3 (SM0FRACVAL3 - SM3FRACVAL3)

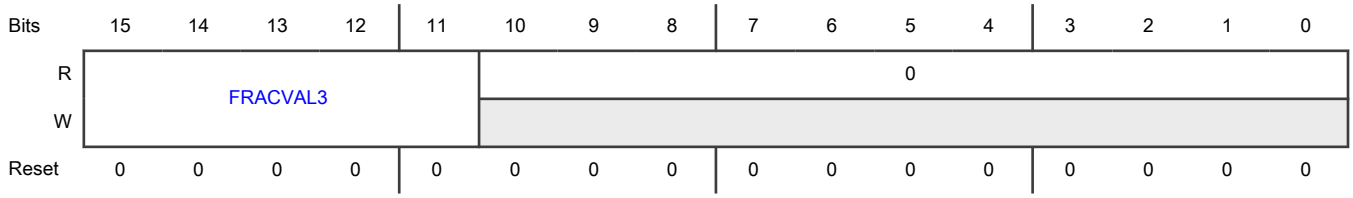
NOTE

The FRACVAL3 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL3 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL3 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Offset

Register	Offset
SM0FRACVAL3	14h
SM1FRACVAL3	74h
SM2FRACVAL3	D4h
SM3FRACVAL3	134h

Diagram



Fields

Field	Description
15-11 FRACVAL3	Fractional Value 3 These bits act as a fractional addition to the value in the VAL3 register which controls the PWM_A turn off timing. It is also used to control the fractional addition to the turn on delay of PWM_B when MCTRL[IPOLx]=0 in complementary mode, CTRL2[INDEP]=0. **NOTE** FRCTRL[FRAC23_EN] should be set to 0 when the values of VAL2 and VAL3 cause the high or low time of the PWM output to be 3 cycles or less.
10-0 —	RESERVED

44.8.1.11 Value Register 3 (SM0VAL3 - SM3VAL3)

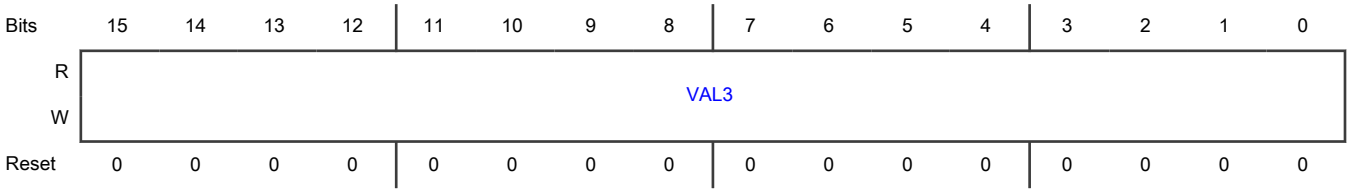
NOTE

The VAL3 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL3 cannot be written when MCTRL[LDOK] is set. Reading VAL3 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Offset

Register	Offset
SM0VAL3	16h
SM1VAL3	76h
SM2VAL3	D6h
SM3VAL3	136h

Diagram



Fields

Field	Description
15-0 VAL3	Value Register 3 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM23 low. This register is not byte accessible. **NOTE** The actual behavior takes affect when counter equal VAL3+1.

44.8.1.12 Fractional Value Register 4 (SM0FRACVAL4 - SM3FRACVAL4)

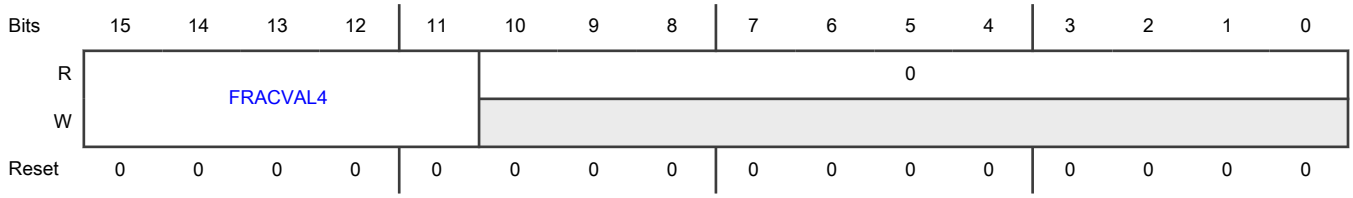
NOTE

The FRACVAL4 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL4 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL4 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Offset

Register	Offset
SM0FRACVAL4	18h
SM1FRACVAL4	78h
SM2FRACVAL4	D8h
SM3FRACVAL4	138h

Diagram



Fields

Field	Description
15-11 FRACVAL4	Fractional Value 4 These bits act as a fractional addition to the value in the VAL4 register which controls the PWM_B turn on timing. It is also used to control the fractional addition to the turn off delay of PWM_A when MCTRL[IPOLx]=1 in complementary mode, CTRL2[INDEP]=0. **NOTE** FRCTRL[FRAC45_EN] should be set to 0 when the values of VAL4 and VAL5 cause the high or low time of the PWM output to be 3 cycles or less.
10-0 —	RESERVED

44.8.1.13 Value Register 4 (SM0VAL4 - SM3VAL4)

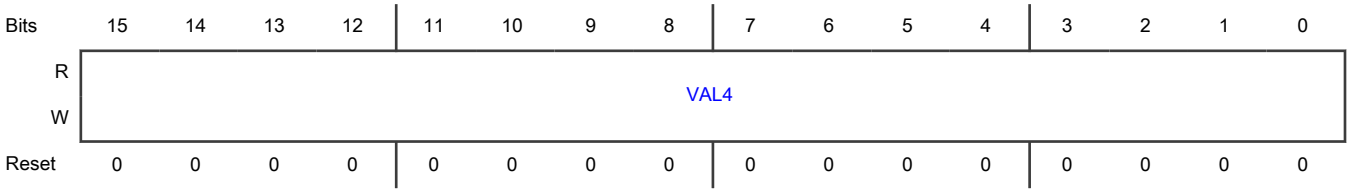
NOTE

The VAL4 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL4 cannot be written when MCTRL[LDOK] is set. Reading VAL4 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Offset

Register	Offset
SM0VAL4	1Ah
SM1VAL4	7Ah
SM2VAL4	DAh
SM3VAL4	13Ah

Diagram



Fields

Field	Description
15-0 VAL4	Value Register 4 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM45 high. This register is not byte accessible. **NOTE** The actual behavior takes affect when counter equal VAL4+1.

44.8.1.14 Fractional Value Register 5 (SM0FRACVAL5 - SM3FRACVAL5)

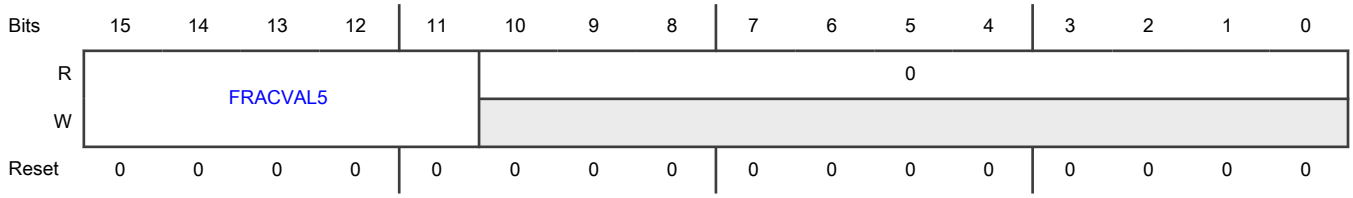
NOTE

The FRACVAL5 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL5 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL5 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Offset

Register	Offset
SM0FRACVAL5	1Ch
SM1FRACVAL5	7Ch
SM2FRACVAL5	DCh
SM3FRACVAL5	13Ch

Diagram



Fields

Field	Description
15-11 FRACVAL5	Fractional Value 5 These bits act as a fractional addition to the value in the VAL5 register which controls the PWM_B turn off timing. It is also used to control the fractional addition to the turn on delay of PWM_A when MCTRL[IPOLx]=1 in complementary mode, CTRL2[INDEP]=0. **NOTE** FRCTRL[FRAC45_EN] should be set to 0 when the values of VAL4 and VAL5 cause the high or low time of the PWM output to be 3 cycles or less.
10-0 —	RESERVED

44.8.1.15 Value Register 5 (SM0VAL5 - SM3VAL5)

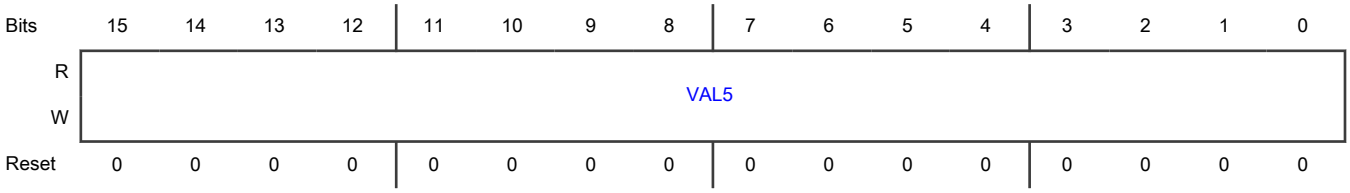
NOTE

The VAL5 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL5 cannot be written when MCTRL[LDOK] is set. Reading VAL5 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Offset

Register	Offset
SM0VAL5	1Eh
SM1VAL5	7Eh
SM2VAL5	DEh
SM3VAL5	13Eh

Diagram



Fields

Field	Description
15-0 VAL5	Value Register 5 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM45 low. This register is not byte accessible. **NOTE** The actual behavior takes affect when counter equal VAL5+1.

44.8.1.16 Fractional Control Register (SM0FRCTRL - SM3FRCTRL)

NOTE

The FRAC1_EN bit is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRAC1_EN cannot be written when MCTRL[LDOK] is set. Reading FRAC1_EN reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Offset

Register	Offset
SM0FRCTRL	20h
SM1FRCTRL	80h
SM2FRCTRL	E0h
SM3FRCTRL	140h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TEST	0							0			FRAC45_...	0	FRAC23_...	FRAC1_EN	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
15 TEST	Test Status Bit This is a read only test bit for factory use. This bit will reset to 0 but may be either 0 or 1 during PWM operation.
14-8 —	RESERVED
7-5 —	RESERVED
4 FRAC45_EN	Fractional Cycle Placement Enable for PWM_B This bit is used to enable the fractional cycle edge placement of PWM_B using the FRACVAL4 and FRACVAL5 registers. When disabled, the fractional cycle edge placement of PWM_B is bypassed. **NOTE** The FRAC45_EN bit is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRAC45_EN cannot be written when MCTRL[LDOK] is set. Reading FRAC45_EN reads the value in a buffer and not necessarily the value the PWM generator is currently using. 0 - Disable fractional cycle placement for PWM_B. 1 - Enable fractional cycle placement for PWM_B.
3 —	RESERVED
2 FRAC23_EN	Fractional Cycle Placement Enable for PWM_A This bit is used to enable the fractional cycle edge placement of PWM_A using the FRACVAL2 and FRACVAL3 registers. When disabled, the fractional cycle edge placement of PWM_A is bypassed. **NOTE** The FRAC23_EN bit is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRAC23_EN cannot be written when MCTRL[LDOK] is set. Reading FRAC23_EN reads the value in a buffer and not necessarily the value the PWM generator is currently using. 0 - Disable fractional cycle placement for PWM_A.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Enable fractional cycle placement for PWM_A.
1 FRAC1_EN	Fractional Cycle PWM Period Enable This bit is used to enable the fractional cycle length of the PWM period using the FRACVAL1 register. When disabled, the fractional cycle length of the PWM period is bypassed. 0 - Disable fractional cycle length for the PWM period. 1 - Enable fractional cycle length for the PWM period.
0 —	RESERVED

44.8.1.17 Output Control Register (SM0OCTRL - SM3OCTRL)

Contains output controls for fault states.

Offset

Register	Offset
SM0OCTRL	22h
SM1OCTRL	82h
SM2OCTRL	E2h
SM3OCTRL	142h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PWMA_IN	PWMB_IN	PWMX_IN	0		POLA	POLB	POLX	0		PWMAFS		PWMBFS		PWMXFS	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
15 PWMA_IN	PWM_A Input This read only bit shows the logic value currently being driven into the PWM_A input. The bit's reset state is undefined.
14	PWM_B Input

Table continues on the next page...

Table continued from the previous page...

Field	Description
PWMB_IN	This read only bit shows the logic value currently being driven into the PWM_B input. The bit's reset state is undefined.
13 PWMX_IN	PWM_X Input This read only bit shows the logic value currently being driven into the PWM_X input. The bit's reset state is undefined.
12-11 —	RESERVED
10 POLA	PWM_A Output Polarity This bit inverts the PWM_A output polarity. 0 - PWM_A output not inverted. A high level on the PWM_A pin represents the "on" or "active" state. 1 - PWM_A output inverted. A low level on the PWM_A pin represents the "on" or "active" state.
9 POLB	PWM_B Output Polarity This bit inverts the PWM_B output polarity. 0 - PWM_B output not inverted. A high level on the PWM_B pin represents the "on" or "active" state. 1 - PWM_B output inverted. A low level on the PWM_B pin represents the "on" or "active" state.
8 POLX	PWM_X Output Polarity This bit inverts the PWM_X output polarity. 0 - PWM_X output not inverted. A high level on the PWM_X pin represents the "on" or "active" state. 1 - PWM_X output inverted. A low level on the PWM_X pin represents the "on" or "active" state.
7-6 —	RESERVED
5-4 PWMAFS	PWM_A Fault State These bits determine the fault state for the PWM_A output during fault conditions and Deep Sleep mode. It may also define the output state during Sleep and DEBUG modes depending on the settings of CTRL2[WAITEN] and CTRL2[DBGEN]. 00 - Output is forced to logic 0 state prior to consideration of output polarity control. 01 - Output is forced to logic 1 state prior to consideration of output polarity control. 10,11 - Output is tristated.
3-2 PWMBFS	PWM_B Fault State These bits determine the fault state for the PWM_B output during fault conditions and Deep Sleep mode. It may also define the output state during Sleep and DEBUG modes depending on the settings of CTRL2[WAITEN] and CTRL2[DBGEN]. 00 - Output is forced to logic 0 state prior to consideration of output polarity control.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	01 - Output is forced to logic 1 state prior to consideration of output polarity control. 10,11 - Output is tristated.
1-0 PWMXFS	PWM_X Fault State These bits determine the fault state for the PWM_X output during fault conditions and Deep Sleep mode. It may also define the output state during Sleep and DEBUG modes depending on the settings of CTRL2[WAITEN] and CTRL2[DBGEN]. 00 - Output is forced to logic 0 state prior to consideration of output polarity control. 01 - Output is forced to logic 1 state prior to consideration of output polarity control. 10,11 - Output is tristated.

44.8.1.18 Status Register (SM0STS - SM3STS)

Contains Compare and Capture flag status.

Offset

Register	Offset
SM0STS	24h
SM1STS	84h
SM2STS	E4h
SM3STS	144h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	RUF	REF	RF	CFA1	CFA0	CFB1	CFB0	CFX1	CFX0	CMPF					
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	Description
15 —	RESERVED
14 RUF	Registers Updated Flag

Table continues on the next page...

Table continued from the previous page...

Field	Description
	This read-only flag is set when one of the INIT, VALx, FRACVALx, or CTRL[PRSC] registers has been written, which indicates potentially non-coherent data in the set of double buffered registers. Clear this bit by a proper reload sequence consisting of a reload signal while MCTRL[LDOK] = 1. Reset clears this bit. 0 - No register update has occurred since last reload. 1 - At least one of the double buffered registers has been updated since the last reload.
13 REF	Reload Error Flag This read/write flag is set when a reload cycle occurs while MCTRL[LDOK] is 0 and the double buffered registers are in a non-coherent state (STS[RUF] = 1). Clear this bit by writing a logic one to this location. Reset clears this bit. 0 - No reload error occurred. 1 - Reload signal occurred with non-coherent data and MCTRL[LDOK] = 0.
12 RF	Reload Flag This read/write flag is set at the beginning of every reload cycle regardless of the state of MCTRL[LDOK]. Clear this bit by writing a logic one to this location when DMAEN[VALDE] is clear (non-DMA mode). This flag can also be cleared by the DMA done signal when DMAEN[VALDE] is set (DMA mode) . Reset clears this bit. 0 - No new reload cycle since last STS[RF] clearing 1 - New reload cycle since last STS[RF] clearing
11 CFA1	Capture Flag A1 This bit is set when a capture event occurs on the Capture A1 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CA1DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CA1DE] is set (DMA mode) . Reset clears this bit.
10 CFA0	Capture Flag A0 This bit is set when a capture event occurs on the Capture A0 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CA0DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CA0DE] is set (DMA mode) . Reset clears this bit.
9 CFB1	Capture Flag B1 This bit is set when a capture event occurs on the Capture B1 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CB1DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CB1DE] is set (DMA mode) . Reset clears this bit.
8 CFB0	Capture Flag B0 This bit is set when a capture event occurs on the Capture B0 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CB0DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CB0DE] is set (DMA mode) . Reset clears this bit.
7 CFX1	Capture Flag X1

Table continues on the next page...

Table continued from the previous page...

Field	Description
	This bit is set when a capture event occurs on the Capture X1 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CX1DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CX1DE] is set (DMA mode) . Reset clears this bit.
6 CFX0	Capture Flag X0 This bit is set when a capture event occurs on the Capture X0 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CX0DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CX0DE] is set (DMA mode) . Reset clears this bit.
5-0 CMPF	Compare Flags These bits are set when the submodule counter value matches the value of one of the VALx registers. Clear these bits by writing a 1 to a bit position. 00_0000 - No compare event has occurred for a particular VALx value. 00_0001 - A compare event has occurred for a particular VALx value.

44.8.1.19 Interrupt Enable Register (SM0INTEN - SM3INTEN)

Contains Compare and Capture interrupt enables.

Offset

Register	Offset
SM0INTEN	26h
SM1INTEN	86h
SM2INTEN	E6h
SM3INTEN	146h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		REIE	RIE	CA1IE	CA0IE	CB1IE	CB0IE	CX1IE	CX0IE	CMPIE					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
15-14 —	RESERVED

Table continues on the next page...

Table continued from the previous page...

Field	Description
13 REIE	Reload Error Interrupt Enable This read/write bit enables the reload error flag, STS[REF], to generate CPU interrupt requests. Reset clears this bit. 0 - STS[REF] CPU interrupt requests disabled 1 - STS[REF] CPU interrupt requests enabled
12 RIE	Reload Interrupt Enable This read/write bit enables the reload flag, STS[RF], to generate CPU interrupt requests. Reset clears this bit. 0 - STS[RF] CPU interrupt requests disabled 1 - STS[RF] CPU interrupt requests enabled
11 CA1IE	Capture A 1 Interrupt Enable This bit allows the STS[CFA1] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CA1DE]. 0 - Interrupt request disabled for STS[CFA1]. 1 - Interrupt request enabled for STS[CFA1].
10 CA0IE	Capture A 0 Interrupt Enable This bit allows the STS[CFA0] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CA0DE]. 0 - Interrupt request disabled for STS[CFA0]. 1 - Interrupt request enabled for STS[CFA0].
9 CB1IE	Capture B 1 Interrupt Enable This bit allows the STS[CFB1] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CB1DE]. 0 - Interrupt request disabled for STS[CFB1]. 1 - Interrupt request enabled for STS[CFB1].
8 CB0IE	Capture B 0 Interrupt Enable This bit allows the STS[CFB0] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CB0DE]. 0 - Interrupt request disabled for STS[CFB0]. 1 - Interrupt request enabled for STS[CFB0].
7 CX1IE	Capture X 1 Interrupt Enable This bit allows the STS[CFX1] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CX1DE]. 0 - Interrupt request disabled for STS[CFX1].

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Interrupt request enabled for STS[CFX1].
6 CX0IE	Capture X 0 Interrupt Enable This bit allows the STS[CFX0] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CX0DE]. 0 - Interrupt request disabled for STS[CFX0]. 1 - Interrupt request enabled for STS[CFX0].
5-0 CMPIE	Compare Interrupt Enables These bits enable the STS[CMPF] flags to cause a compare interrupt request to the CPU. 00_0000 - The corresponding STS[CMPF] bit will not cause an interrupt request. 00_0001 - The corresponding STS[CMPF] bit will cause an interrupt request.

44.8.1.20 DMA Enable Register (SM0DMAEN - SM3DMAEN)

Contains controls for DMA.

Offset

Register	Offset
SM0DMAEN	28h
SM1DMAEN	88h
SM2DMAEN	E8h
SM3DMAEN	148h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0						VALDE	FAND	CAPTDE		CA1DE	CA0DE	CB1DE	CB0DE	CX1DE	CX0DE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
15-10 —	RESERVED
9	Value Registers DMA Enable

Table continues on the next page...

Table continued from the previous page...

Field	Description
VALDE	This read/write bit enables DMA write requests for the VALx and FRACVALx registers when STS[RF] is set. Reset clears this bit. 0 - DMA write requests disabled 1 - Enabled. DMA write requests for the VALx and FRACVALx registers enabled
8 FAND	FIFO Watermark AND Control This read/write bit works in conjunction with the DMAEN[CAPTDE] field when it is set to watermark mode (DMAEN[CAPTDE] = 01). While DMAEN[CAxDE], DMAEN[CBxDE], and DMAEN[CXxDE] determine which FIFO watermarks the DMA read request is sensitive to, this bit determines if the selected watermarks are AND'ed together or OR'ed together in order to create the request. 0 - Selected FIFO watermarks are OR'ed together. 1 - Selected FIFO watermarks are AND'ed together.
7-6 CAPTDE	Capture DMA Enable Source Select These read/write bits select the source of enabling the DMA read requests for the capture FIFOs. Reset clears these bits. 00 - Read DMA requests disabled. 01 - Exceeding a FIFO watermark sets the DMA read request. This requires at least one of DMAEN[CA1DE], DMAEN[CA0DE], DMAEN[CB1DE], DMAEN[CB0DE], DMAEN[CX1DE], or DMAEN[CX0DE] to also be set in order to determine to which watermark(s) the DMA request is sensitive. 10 - A local sync (VAL1 matches counter) sets the read DMA request. 11 - A local reload (STS[RF] being set) sets the read DMA request.
5 CA1DE	Capture A1 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture A1 FIFO data when STS[CFA1] is set. Reset clears this bit. Do not set both this bit and INTEN[CA1IE].
4 CA0DE	Capture A0 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture A0 FIFO data when STS[CFA0] is set. Reset clears this bit. Do not set both this bit and INTEN[CA0IE].
3 CB1DE	Capture B1 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture B1 FIFO data when STS[CFB1] is set. Reset clears this bit. Do not set both this bit and INTEN[CB1IE].
2 CB0DE	Capture B0 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture B0 FIFO data when STS[CFB0] is set. Reset clears this bit. Do not set both this bit and INTEN[CB0IE].
1 CX1DE	Capture X1 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture X1 FIFO data when STS[CFX1] is set. Reset clears this bit. Do not set both this bit and INTEN[CX1IE].

Table continues on the next page...

Table continued from the previous page...

Field	Description
0 CX0DE	Capture X0 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture X0 FIFO data when STS[CFX0] is set. Reset clears this bit. Do not set both this bit and INTEN[CX0IE].

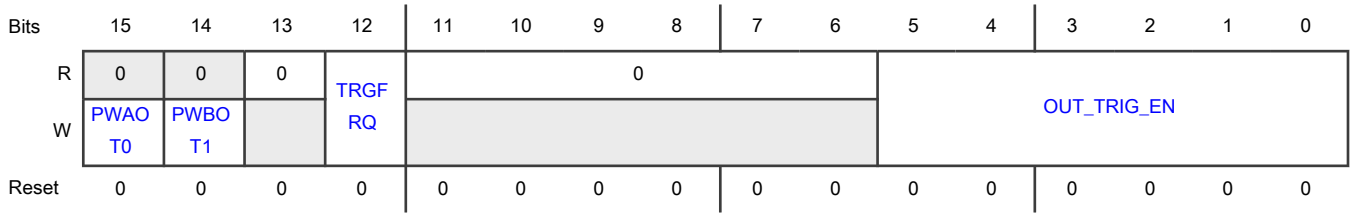
44.8.1.21 Output Trigger Control Register (SM0TCTRL - SM3TCTRL)

Contains trigger controls.

Offset

Register	Offset
SM0TCTRL	2Ah
SM1TCTRL	8Ah
SM2TCTRL	EAh
SM3TCTRL	14Ah

Diagram



Fields

Field	Description
15 PWAOT0	Mux Output Trigger 0 Source Select This bit selects which signal to bring out on the PWM's PWM_MUX_TRIG0 port. 0 - Route the PWM_OUT_TRIG0 signal to PWM_MUX_TRIG0 port. 1 - Route the PWMA output to the PWM_MUX_TRIG0 port.
14 PWBOT1	Mux Output Trigger 1 Source Select This bit selects which signal to bring out on the PWM's PWM_MUX_TRIG1 port. 0 - Route the PWM_OUT_TRIG1 signal to PWM_MUX_TRIG1 port. 1 - Route the PWMB output to the PWM_MUX_TRIG1 port.
13	RESERVED

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
12 TRGFRQ	Trigger frequency This read/write bit allows control over the frequency of the trigger outputs when using non-zero values of CTRL[LDFQ]. 0 - Trigger outputs are generated during every PWM period even if the PWM is not reloaded every period due to CTRL[LDFQ] being non-zero. 1 - Trigger outputs are generated only during the final PWM period prior to a reload opportunity when the PWM is not reloaded every period due to CTRL[LDFQ] being non-zero.
11-6 —	RESERVED
5-0 OUT_TRIG_EN	Output Trigger Enables These bits enable the generation of PWM_OUT_TRIG0 and PWM_OUT_TRIG1 outputs based on the counter value matching the value in one or more of the VAL0-5 registers. **NOTE** Due to delays in creating the PWM outputs, the output trigger signals will lead the PWM output edges by 2-3 clock cycles depending on the fractional cycle value being used. 1x_xxxx - PWM_OUT_TRIG1 will set when the counter value matches the VAL5 value. x1_xxxx - PWM_OUT_TRIG0 will set when the counter value matches the VAL4 value. xx_1xxx - PWM_OUT_TRIG1 will set when the counter value matches the VAL3 value. xx_x1xx - PWM_OUT_TRIG0 will set when the counter value matches the VAL2 value. xx_xx1x - PWM_OUT_TRIG1 will set when the counter value matches the VAL1 value. xx_xxx1 - PWM_OUT_TRIG0 will set when the counter value matches the VAL0 value.

44.8.1.22 Fault Disable Mapping Register 0 (SM0DISMAP0 - SM3DISMAP0)

This register determines which PWM pins are disabled by the fault protection inputs. Reset sets all of the bits in the fault disable mapping register.

Offset

Register	Offset
SM0DISMAP0	2Ch
SM1DISMAP0	8Ch
SM2DISMAP0	ECh
SM3DISMAP0	14Ch

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	1				DIS0X				DIS0B				DIS0A			
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Description
15-12 —	RESERVED
11-8 DIS0X	PWM_X Fault Disable Mask 0 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 0. The PWM_X output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.
7-4 DIS0B	PWM_B Fault Disable Mask 0 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 0. The PWM_B output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.
3-0 DIS0A	PWM_A Fault Disable Mask 0 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 0. The PWM_A output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.

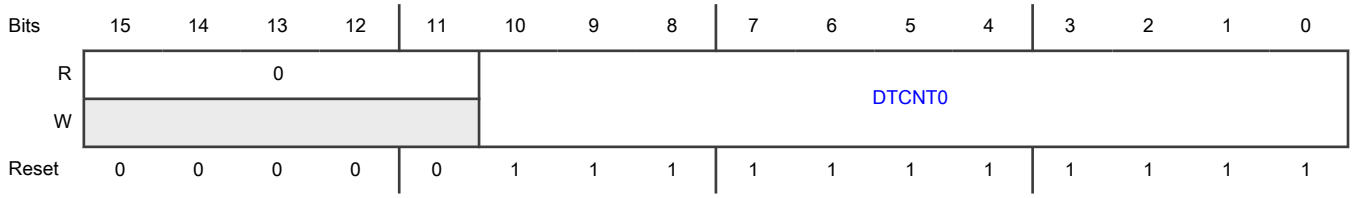
44.8.1.23 Deadtime Count Register 0 (SM0DTCNT0 - SM3DTCNT0)

Deadtime operation applies only to complementary channel operation. The values written to the DTCNTx registers are in terms of IPBus clock cycles regardless of the setting of CTRL[PRSC] and/or CTRL2[CLK_SEL]. Reset sets the deadtime count registers to a default value of 0x07FF, selecting a deadtime of 2047 IPBus clock cycles. The DTCNTx registers are not byte accessible.

Offset

Register	Offset
SM0DTCNT0	30h
SM1DTCNT0	90h
SM2DTCNT0	F0h
SM3DTCNT0	150h

Diagram



Fields

Field	Description
15-11 —	RESERVED
10-0 DTCNT0	Deadtime Count Register 0 The DTCNT0 field is used to control the deadtime during 0 to 1 transitions of the PWM_A output (assuming normal polarity).

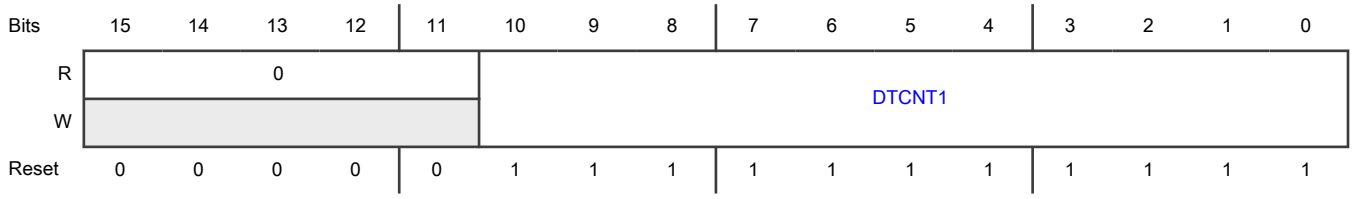
44.8.1.24 Deadtime Count Register 1 (SM0DTCNT1 - SM3DTCNT1)

Deadtime operation applies only to complementary channel operation. The values written to the DTCNTx registers are in terms of IPBus clock cycles regardless of the setting of CTRL[PRSC] and/or CTRL2[CLK_SEL]. Reset sets the deadtime count registers to a default value of 0x07FF, selecting a deadtime of 2047 IPBus clock cycles. The DTCNTx registers are not byte accessible.

Offset

Register	Offset
SM0DTCNT1	32h
SM1DTCNT1	92h
SM2DTCNT1	F2h
SM3DTCNT1	152h

Diagram



Fields

Field	Description
15-11	RESERVED

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
10-0 DTCNT1	Deadtime Count Register 1 The DTCNT1 field is used to control the deadtime during 0 to 1 transitions of the complementary PWM_B output.

44.8.1.25 Capture Control A Register (SM0CAPTCTRLA - SM3CAPTCTRLA)

Contains capture controls for mode A.

Offset

Register	Offset
SM0CAPTCTRLA	34h
SM1CAPTCTRLA	94h
SM2CAPTCTRLA	F4h
SM3CAPTCTRLA	154h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CA1CNT			CA0CNT			CFAWM		EDGCNTA...	INP_SELA	EDGA1		EDGA0		ONESHOTA	ARMA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
15-13 CA1CNT	Capture A1 FIFO Word Count This field reflects the number of words in the Capture A1 FIFO. (FIFO depth is 1.)
12-10 CA0CNT	Capture A0 FIFO Word Count This field reflects the number of words in the Capture A0 FIFO. (FIFO depth is 1.)
9-8 CFAWM	Capture A FIFOs Water Mark This field represents the water mark level for capture A FIFOs. The capture flags, STS[CFA1] and STS[CFA0], are not set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7	Edge Counter A Enable

Table continues on the next page...

Table continued from the previous page...

Field	Description
EDGCNTA_EN	This bit enables the edge counter which counts rising and falling edges on the PWM_A input signal. 0 - Edge counter disabled and held in reset 1 - Edge counter enabled
6 INP_SELA	Input Select A This bit selects between the raw PWM_A input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0 - Raw PWM_A input signal selected as source. 1 - Edge Counter. Output of edge counter/compare selected as source. Note that when this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLA[EDGA0] and CAPTCTRLA[EDGA1] fields are ignored. The software must still place a value other than 00 in either or both of the CAPTCTLRA[EDGA0] and/or CAPTCTRLA[EDGA1] fields in order to enable one or both of the capture registers.
5-4 EDGA1	Edge A 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00 - Disabled 01 - Capture falling edges 10 - Capture rising edges 11 - Capture any edge
3-2 EDGA0	Edge A 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00 - Disabled 01 - Capture falling edges 10 - Capture rising edges 11 - Capture any edge
1 ONESHOTA	One Shot Mode A This bit selects between free running and one shot mode for the input capture circuitry. 0 - Free Running. Free running mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLA[ARMA] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1 - One Shot. One shot mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLA[ARMA] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and CAPTCTRLA[ARMA] is cleared. No further captures will be performed until CAPTCTRLA[ARMA] is set again. If only one

Table continues on the next page...

Table continued from the previous page...

Field	Description
	of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and CAPTCTRLA[ARMA] is then cleared.
0 ARMA	Arm A Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0 - Input capture operation is disabled. 1 - Input capture operation as specified by CAPTCTRLA[EDGAx] is enabled.

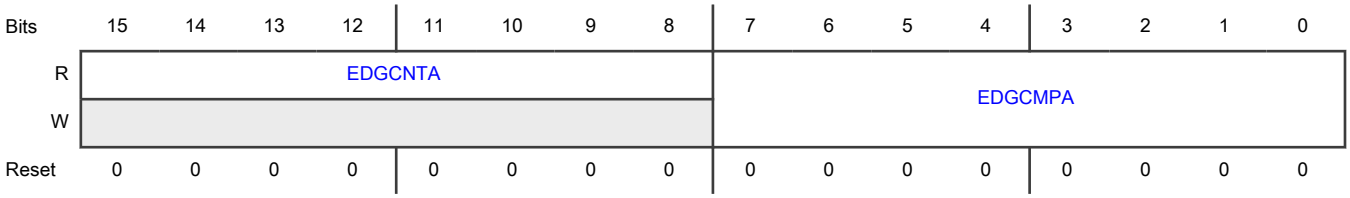
44.8.1.26 Capture Compare A Register (SM0CAPTCOMPA - SM3CAPTCOMPA)

Contains capture and compare values for mode A.

Offset

Register	Offset
SM0CAPTCOMPA	36h
SM1CAPTCOMPA	96h
SM2CAPTCOMPA	F6h
SM3CAPTCOMPA	156h

Diagram



Fields

Field	Description
15-8 EDGCNTA	Edge Counter A This read-only field contains the edge counter value for the PWM_A input capture circuitry.
7-0 EDGCMPA	Edge Compare A This read/write field is the compare value associated with the edge counter for the PWM_A input capture circuitry.

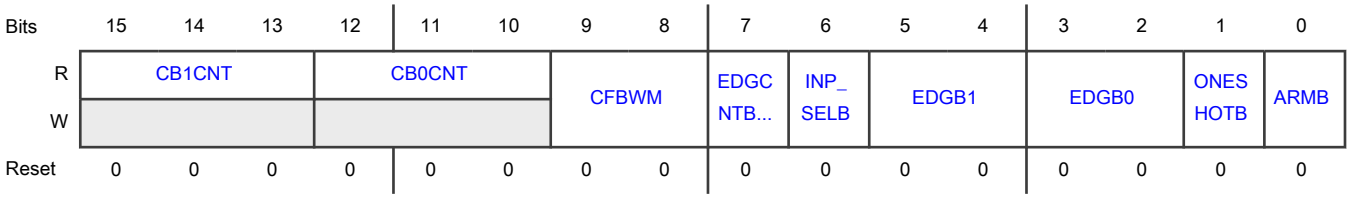
44.8.1.27 Capture Control B Register (SM0CAPTCTRLB - SM3CAPTCTRLB)

Contains capture controls for mode B.

Offset

Register	Offset
SM0CAPTCTRLB	38h
SM1CAPTCTRLB	98h
SM2CAPTCTRLB	F8h
SM3CAPTCTRLB	158h

Diagram



Fields

Field	Description
15-13 CB1CNT	Capture B1 FIFO Word Count This field reflects the number of words in the Capture B1 FIFO. (FIFO depth is 1.)
12-10 CB0CNT	Capture B0 FIFO Word Count This field reflects the number of words in the Capture B0 FIFO. (FIFO depth is 1.)
9-8 CFBWM	Capture B FIFOs Water Mark This field represents the water mark level for capture B FIFOs. The capture flags, STS[CFB1] and STS[CFB0], won't be set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7 EDGCNTB_EN	Edge Counter B Enable This bit enables the edge counter which counts rising and falling edges on the PWM_B input signal. 0 - Edge counter disabled and held in reset 1 - Edge counter enabled
6 INP_SELB	Input Select B This bit selects between the raw PWM_B input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0 - Raw PWM_B input signal selected as source.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Edge Counter. Output of edge counter/compare selected as source. Note that when this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLB[EDGB0] and CAPTCTRLB[EDGB1] fields are ignored. The software must still place a value other than 00 in either or both of the CAPTCTLRB[EDGB0] and/or CAPTCTRLB[EDGB1] fields in order to enable one or both of the capture registers.
5-4 EDGB1	Edge B 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00 - Disabled 01 - Capture falling edges 10 - Capture rising edges 11 - Capture any edge
3-2 EDGB0	Edge B 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00 - Disabled 01 - Capture falling edges 10 - Capture rising edges 11 - Capture any edge
1 ONESHOTB	One Shot Mode B This bit selects between free running and one shot mode for the input capture circuitry. 0 - Free Running. Free running mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLB[ARMB] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1 - One Shot. One shot mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLB[ARMB] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and CAPTCTRLB[ARMB] is cleared. No further captures will be performed until CAPTCTRLB[ARMB] is set again. If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and CAPTCTRLB[ARMB] is then cleared.
0 ARMB	Arm B Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0 - Input capture operation is disabled. 1 - Input capture operation as specified by CAPTCTRLB[EDGBx] is enabled.

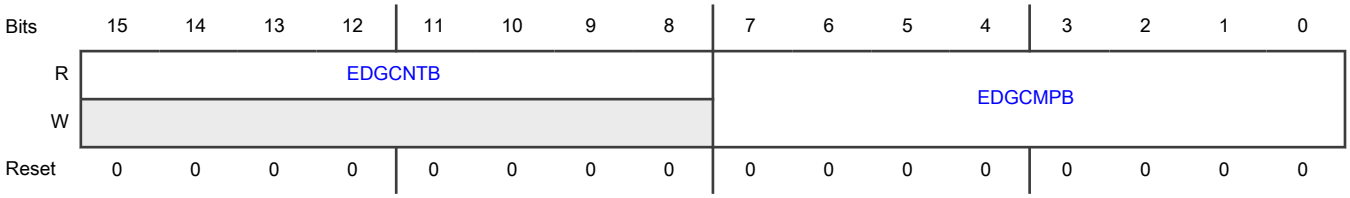
44.8.1.28 Capture Compare B Register (SM0CAPTCOMPB - SM3CAPTCOMPB)

Contains capture and compare values for mode B.

Offset

Register	Offset
SM0CAPTCOMPB	3Ah
SM1CAPTCOMPB	9Ah
SM2CAPTCOMPB	FAh
SM3CAPTCOMPB	15Ah

Diagram



Fields

Field	Description
15-8 EDGCNTB	Edge Counter B This read-only field contains the edge counter value for the PWM_B input capture circuitry.
7-0 EDGCMPB	Edge Compare B This read/write field is the compare value associated with the edge counter for the PWM_B input capture circuitry.

44.8.1.29 Capture Control X Register (SM0CAPTCTRLX - SM3CAPTCTRLX)

Contains capture controls for mode X.

Offset

Register	Offset
SM0CAPTCTRLX	3Ch
SM1CAPTCTRLX	9Ch
SM2CAPTCTRLX	FCh
SM3CAPTCTRLX	15Ch

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CX1CNT				CX0CNT				EDGCNTX...	INP_SELX	EDGX1		EDGX0		ONESHOTX	ARMX
W							CFXWM									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
15-13 CX1CNT	Capture X1 FIFO Word Count This field reflects the number of words in the Capture X1 FIFO. (FIFO depth is 1.)
12-10 CX0CNT	Capture X0 FIFO Word Count This field reflects the number of words in the Capture X0 FIFO. (FIFO depth is 1.)
9-8 CFXWM	Capture X FIFOs Water Mark This field represents the water mark level for capture X FIFOs. The capture flags, STS[CFX1] and STS[CFX0], won't be set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7 EDGCNTX_EN	Edge Counter X Enable This bit enables the edge counter which counts rising and falling edges on the PWM_X input signal. 0 - Edge counter disabled and held in reset 1 - Edge counter enabled
6 INP_SELX	Input Select X This bit selects between the raw PWM_X input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0 - Raw PWM_X input signal selected as source. 1 - Edge Counter. Output of edge counter/compare selected as source. Note that when this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLX[EDGX0] and CAPTCTRLX[EDGX1] fields are ignored. The software must still place a value other than 00 in either or both of the CAPTCTLRX[EDGX0] and/or CAPTCTRLX[EDGX1] fields in order to enable one or both of the capture registers.
5-4 EDGX1	Edge X 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00 - Disabled 01 - Capture falling edges 10 - Capture rising edges 11 - Capture any edge
3-2	Edge X 0

Table continues on the next page...

Table continued from the previous page...

Field	Description
EDGX0	These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00 - Disabled 01 - Capture falling edges 10 - Capture rising edges 11 - Capture any edge
1 ONESHOTX	One Shot Mode Aux This bit selects between free running and one shot mode for the input capture circuitry. 0 - Free Running. Free running mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1 - One Shot. One shot mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and the ARMX bit is cleared. No further captures will be performed until the ARMX bit is set again.If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and the ARMX bit is then cleared.
0 ARMX	Arm X Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0 - Input capture operation is disabled. 1 - Input capture operation as specified by CAPTCTRLX[EDGXx] is enabled.

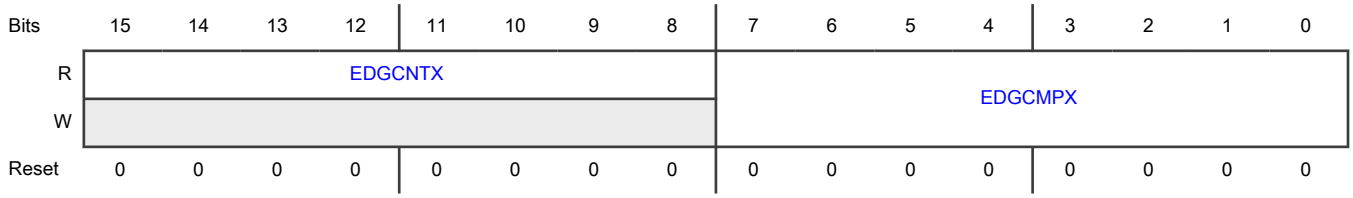
44.8.1.30 Capture Compare X Register (SM0CAPTCOMPX - SM3CAPTCOMPX)

Contains capture and control values for mode X.

Offset

Register	Offset
SM0CAPTCOMPX	3Eh
SM1CAPTCOMPX	9Eh
SM2CAPTCOMPX	FEh
SM3CAPTCOMPX	15Eh

Diagram



Fields

Field	Description
15-8 EDGCNTX	Edge Counter X This read-only field contains the edge counter value for the PWM_X input capture circuitry.
7-0 EDGCMPX	Edge Compare X This read/write field is the compare value associated with the edge counter for the PWM_X input capture circuitry.

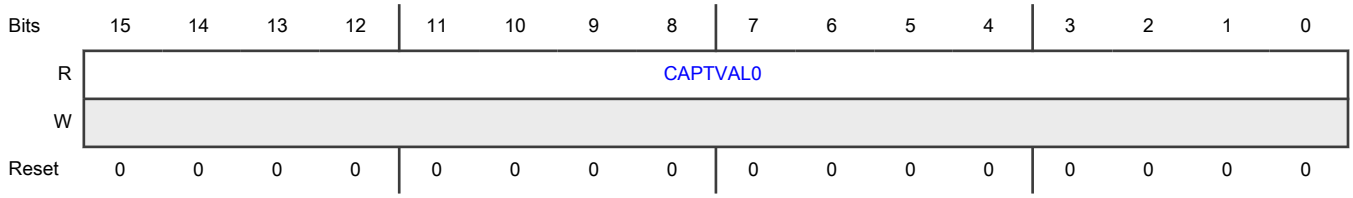
44.8.1.31 Capture Value 0 Register (SM0CVAL0 - SM3CVAL0)

Writing this register generates bus transfer error.

Offset

Register	Offset
SM0CVAL0	40h
SM1CVAL0	A0h
SM2CVAL0	100h
SM3CVAL0	160h

Diagram



Fields

Field	Description
15-0 CAPTVAL0	CAPTVAL0 This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLX[EDGX0]. Each capture will increase the value of

Table continues on the next page...

Field	Description
	CAPTCTRLX[CX0CNT] by 1 until the maximum value is reached. Each read of this register will decrease the value of CAPTCTRLX[CX0CNT] by 1 until 0 is reached. This register is not byte accessible.

44.8.1.32 Capture Value 0 Cycle Register (SM0CVAL0CYC - SM3CVAL0CYC)

Writing this register generates bus transfer error.

Offset

Register	Offset
SM0CVAL0CYC	42h
SM1CVAL0CYC	A2h
SM2CVAL0CYC	102h
SM3CVAL0CYC	162h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												CVAL0CYC			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
15-4 —	RESERVED
3-0 CVAL0CYC	CVAL0CYC This read-only register stores the cycle number corresponding to the value captured in CVAL0. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

44.8.1.33 Capture Value 1 Register (SM0CVAL1 - SM3CVAL1)

Writing this register generates bus transfer error.

Offset

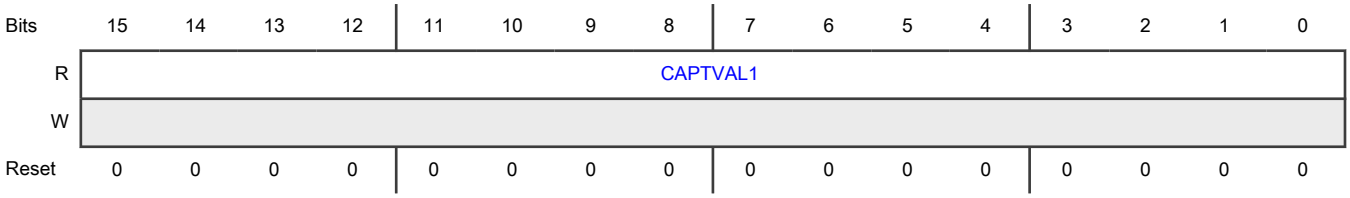
Register	Offset
SM0CVAL1	44h

Table continues on the next page...

Table continued from the previous page...

Register	Offset
SM1CVAL1	A4h
SM2CVAL1	104h
SM3CVAL1	164h

Diagram



Fields

Field	Description
15-0 CAPTVAL1	CAPTVAL1 This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLX[EDGX1]. Each capture increases the value of CAPTCTRLX[CX1CNT] by 1 until the maximum value is reached. Each read of this register decreases the value of CAPTCTRLX[CX1CNT] by 1 until 0 is reached. This register is not byte accessible.

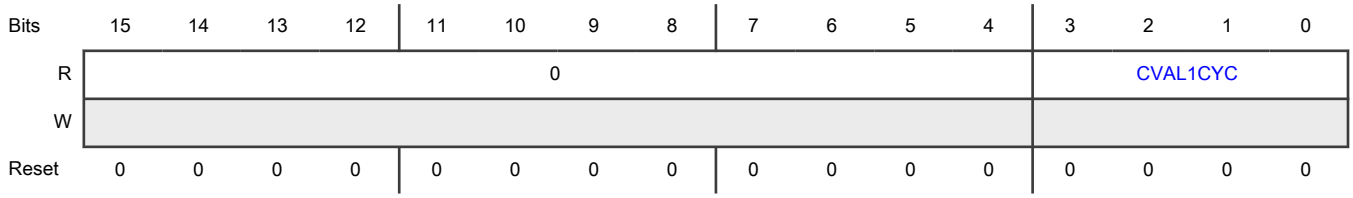
44.8.1.34 Capture Value 1 Cycle Register (SM0CVAL1CYC - SM3CVAL1CYC)

Writing this register generates bus transfer error.

Offset

Register	Offset
SM0CVAL1CYC	46h
SM1CVAL1CYC	A6h
SM2CVAL1CYC	106h
SM3CVAL1CYC	166h

Diagram



Fields

Field	Description
15-4 —	RESERVED
3-0 CVAL1CYC	CVAL1CYC This read-only register stores the cycle number corresponding to the value captured in CVAL1. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

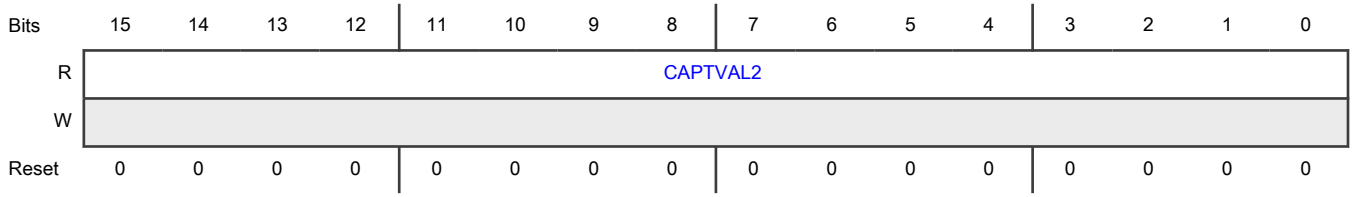
44.8.1.35 Capture Value 2 Register (SM0CVAL2 - SM3CVAL2)

Writing this register generates bus transfer error.

Offset

Register	Offset
SM0CVAL2	48h
SM1CVAL2	A8h
SM2CVAL2	108h
SM3CVAL2	168h

Diagram



Fields

Field	Description
15-0 CAPTVAL2	CAPTVAL2

Table continues on the next page...

Field	Description
	This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLA[EDGA0]. Each capture increases the value of CAPTCTRLA[CA0CNT] by 1 until the maximum value is reached. Each read of this register decreases the value of CAPTCTRLA[CA0CNT] by 1 until 0 is reached. This register is not byte accessible.

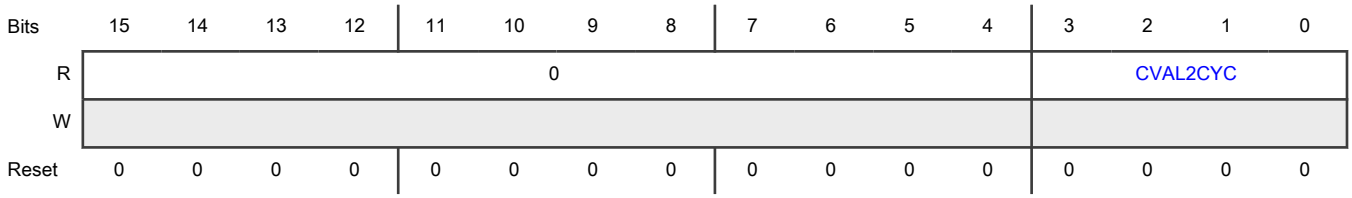
44.8.1.36 Capture Value 2 Cycle Register (SM0CVAL2CYC - SM3CVAL2CYC)

Writing this register generates bus transfer error.

Offset

Register	Offset
SM0CVAL2CYC	4Ah
SM1CVAL2CYC	AAh
SM2CVAL2CYC	10Ah
SM3CVAL2CYC	16Ah

Diagram



Fields

Field	Description
15-4 —	RESERVED
3-0 CVAL2CYC	CVAL2CYC This read-only register stores the cycle number corresponding to the value captured in CVAL2. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

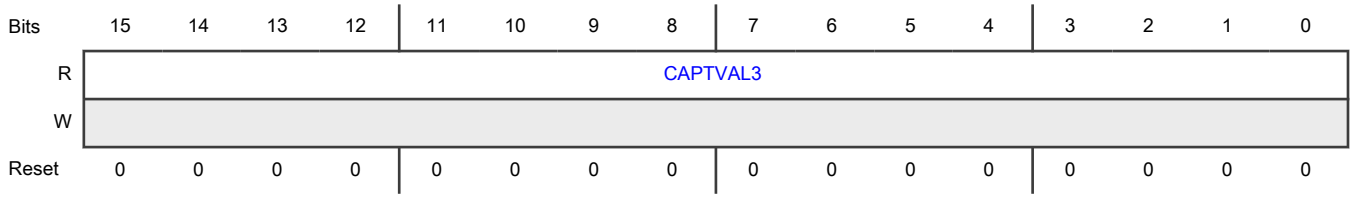
44.8.1.37 Capture Value 3 Register (SM0CVAL3 - SM3CVAL3)

Writing this register generates bus transfer error.

Offset

Register	Offset
SM0CVAL3	4Ch
SM1CVAL3	ACh
SM2CVAL3	10Ch
SM3CVAL3	16Ch

Diagram



Fields

Field	Description
15-0 CAPTVAL3	CAPTVAL3 This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLA[EDGA1]. Each capture increases the value of CAPTCTRLA[CA1CNT] by 1 until the maximum value is reached. Each read of this register decreases the value of CAPTCTRLA[CA1CNT] by 1 until 0 is reached. This register is not byte accessible.

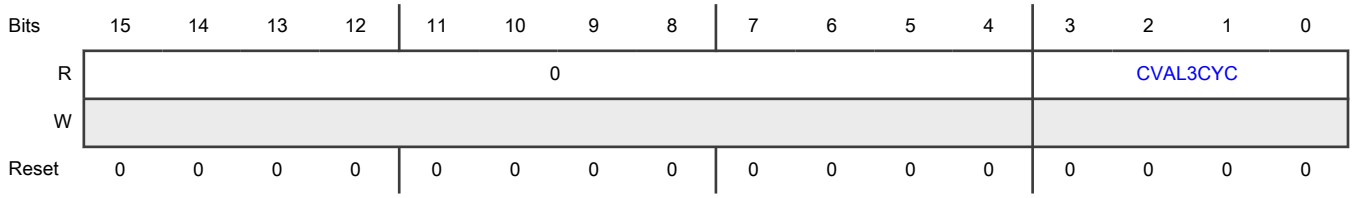
44.8.1.38 Capture Value 3 Cycle Register (SM0CVAL3CYC - SM3CVAL3CYC)

Writing this register generates bus transfer error.

Offset

Register	Offset
SM0CVAL3CYC	4Eh
SM1CVAL3CYC	AEh
SM2CVAL3CYC	10Eh
SM3CVAL3CYC	16Eh

Diagram



Fields

Field	Description
15-4 —	RESERVED
3-0 CVAL3CYC	CVAL3CYC This read-only register stores the cycle number corresponding to the value captured in CVAL3. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

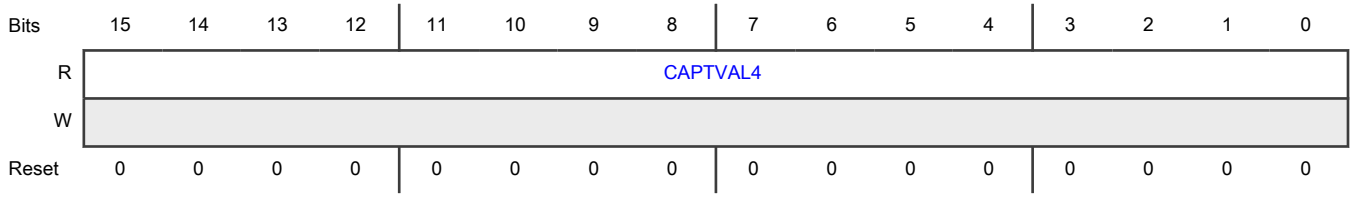
44.8.1.39 Capture Value 4 Register (SM0CVAL4 - SM3CVAL4)

Writing this register generates bus transfer error.

Offset

Register	Offset
SM0CVAL4	50h
SM1CVAL4	B0h
SM2CVAL4	110h
SM3CVAL4	170h

Diagram



Fields

Field	Description
15-0 CAPTVAL4	CAPTVAL4

Table continues on the next page...

Field	Description
	This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLB[EDGB0]. Each capture increases the value of CAPTCTRLB[CB0CNT] by 1 until the maximum value is reached. Each read of this register decreases the value of CAPTCTRLB[CB0CNT] by 1 until 0 is reached. This register is not byte accessible.

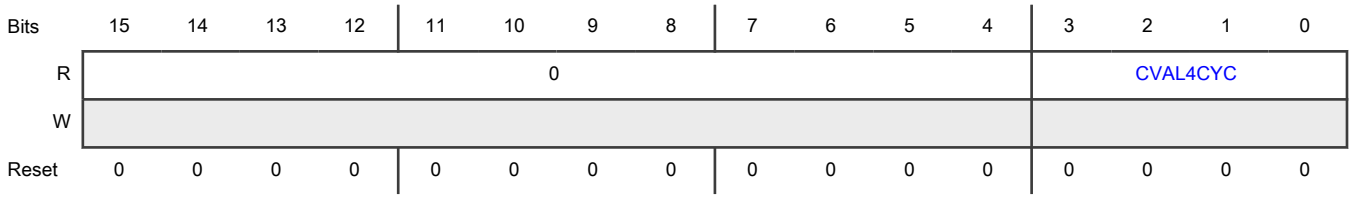
44.8.1.40 Capture Value 4 Cycle Register (SM0CVAL4CYC - SM3CVAL4CYC)

Writing this register generates bus transfer error.

Offset

Register	Offset
SM0CVAL4CYC	52h
SM1CVAL4CYC	B2h
SM2CVAL4CYC	112h
SM3CVAL4CYC	172h

Diagram



Fields

Field	Description
15-4 —	RESERVED
3-0 CVAL4CYC	CVAL4CYC This read-only register stores the cycle number corresponding to the value captured in CVAL4. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

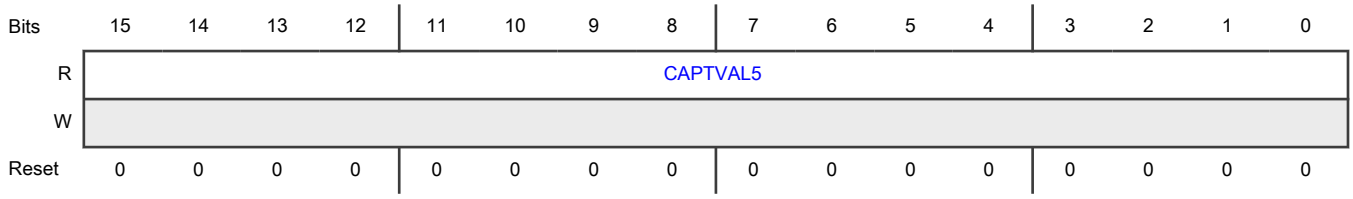
44.8.1.41 Capture Value 5 Register (SM0CVAL5 - SM3CVAL5)

Writing this register generates bus transfer error.

Offset

Register	Offset
SM0CVAL5	54h
SM1CVAL5	B4h
SM2CVAL5	114h
SM3CVAL5	174h

Diagram



Fields

Field	Description
15-0 CAPTVAL5	CAPTVAL5 This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLB[EDGB1]. Each capture increases the value of CAPTCTRLB[CB1CNT] by 1 until the maximum value is reached. Each read of this register decreases the value of CAPTCTRLB[CB1CNT] by 1 until 0 is reached. This register is not byte accessible.

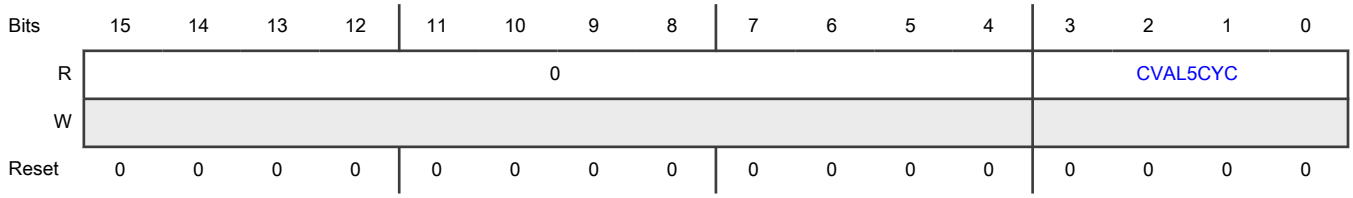
44.8.1.42 Capture Value 5 Cycle Register (SM0CVAL5CYC - SM3CVAL5CYC)

Writing this register generates bus transfer error.

Offset

Register	Offset
SM0CVAL5CYC	56h
SM1CVAL5CYC	B6h
SM2CVAL5CYC	116h
SM3CVAL5CYC	176h

Diagram



Fields

Field	Description
15-4 —	RESERVED
3-0 CVAL5CYC	CVAL5CYC This read-only register stores the cycle number corresponding to the value captured in CVAL5. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

44.8.1.43 Capture PWMA Input Filter Register (SM0CAPTFILTA - SM3CAPTFILTA)

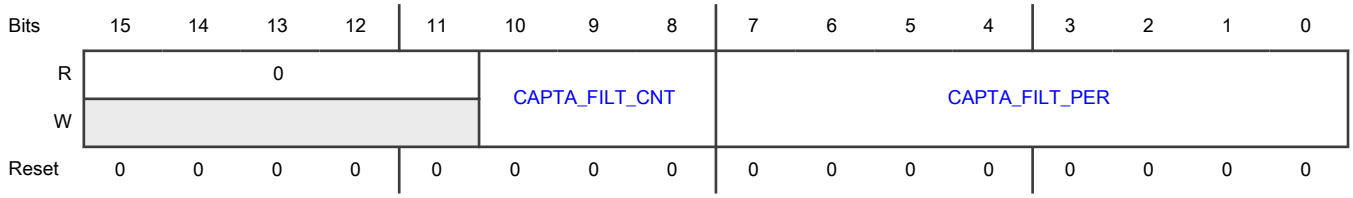
Input filter considerations include:

- The CAPTA_FILT_PER value should be set such that the sampling period is larger than the period of the expected noise. This way a noise spike will only corrupt one sample. The CAPTA_FILT_CNT value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the CAPTA_FILT_CNT+3 power.
- The values of CAPTA_FILT_PER and CAPTA_FILT_CNT must also be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting CAPTA_FILT_PER to a non-zero value) introduces a latency of ((CAPTA_FILT_CNT+4) x CAPTA_FILT_PER x IPBus clock period). Note that even when the filter is enabled, there is a combinational path to disable the PWM outputs. This is to ensure rapid response to fault conditions and also to ensure fault response if the PWM module loses its clock.

Offset

Register	Offset
SM0CAPTFILTA	5Ah
SM1CAPTFILTA	BAh
SM2CAPTFILTA	11Ah
SM3CAPTFILTA	17Ah

Diagram



Fields

Field	Description
15-11 —	RESERVED
10-8 CAPTA_FILT_CNT	Fault Filter Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. The number of samples is the decimal value of this field plus three: the bit field value of 0-7 represents 3-10 samples, respectively. The value of CAPTA_FILT_CNT affects the input latency.
7-0 CAPTA_FILT_PER	Fault Filter Period This 8-bit field applies universally to all fault inputs. These bits represent the sampling period (in IPBus clock cycles) of the fault pin input filter. Each input is sampled multiple times at the rate specified by this field. If CAPTA_FILT_PER is 0x00 (default), then the input filter is bypassed. The value of CAPTA_FILT_PER affects the input latency. **NOTE** When changing values for CAPTA_FILT_PER from one non-zero value to another non-zero value, first write a value of zero to clear the filter.

44.8.1.44 Capture PWMB Input Filter Register (SM0CAPTFILTB - SM3CAPTFILTB)

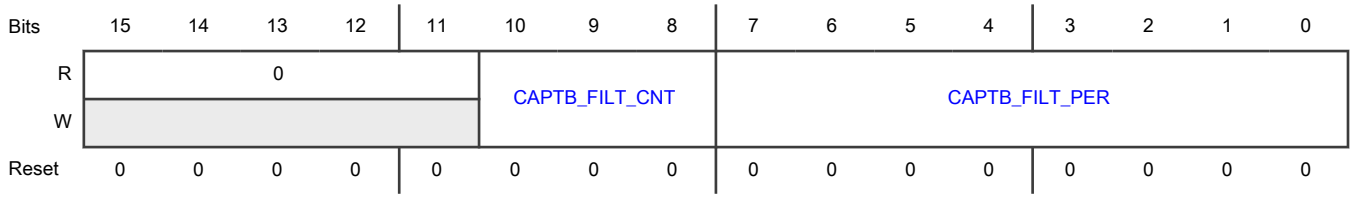
Input filter considerations include:

- The CAPTB_FILT_PER value should be set such that the sampling period is larger than the period of the expected noise. This way a noise spike will only corrupt one sample. The CAPTB_FILT_CNT value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the CAPTB_FILT_CNT+3 power.
- The values of CAPTB_FILT_PER and CAPTB_FILT_CNT must also be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting CAPTB_FILT_PER to a non-zero value) introduces a latency of ((CAPTB_FILT_CNT+4) x CAPTB_FILT_PER x IPBus clock period). Note that even when the filter is enabled, there is a combinational path to disable the PWM outputs. This is to ensure rapid response to fault conditions and also to ensure fault response if the PWM module loses its clock. The latency induced by the filter will be seen in the time to set FSTS[FFLAG] and FSTS[FFPIN].

Offset

Register	Offset
SM0CAPTFILTB	5Ch
SM1CAPTFILTB	BCh
SM2CAPTFILTB	11Ch
SM3CAPTFILTB	17Ch

Diagram



Fields

Field	Description
15-11 —	RESERVED
10-8 CAPTB_FILT_CNT	Fault Filter Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. The number of samples is the decimal value of this field plus three: the bit field value of 0-7 represents 3-10 samples, respectively. The value of CAPTB_FILT_CNT affects the input latency.
7-0 CAPTB_FILT_PER	Fault Filter Period This 8-bit field applies universally to all fault inputs. These bits represent the sampling period (in IPBus clock cycles) of the fault pin input filter. Each input is sampled multiple times at the rate specified by this field. If CAPTB_FILT_PER is 0x00 (default), then the input filter is bypassed. The value of CAPTB_FILT_PER affects the input latency. **NOTE** When changing values for CAPTB_FILT_PER from one non-zero value to another non-zero value, first write a value of zero to clear the filter.

44.8.1.45 Capture PWMX Input Filter Register (SM0CAPTFILTX - SM3CAPTFILTX)

Input filter considerations include:

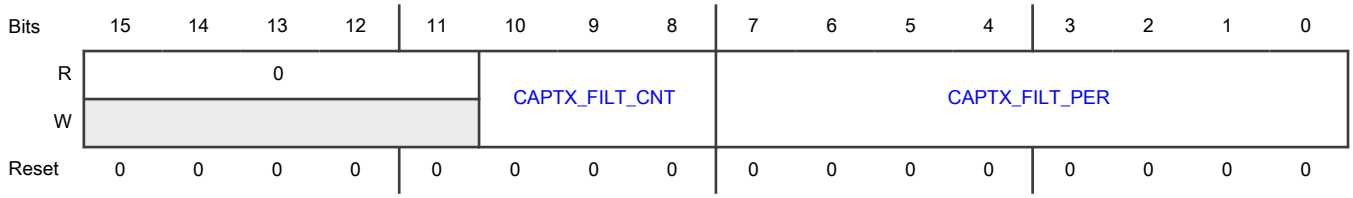
- The CAPTX_FILT_PER value should be set such that the sampling period is larger than the period of the expected noise. This way a noise spike will only corrupt one sample. The CAPTX_FILT_CNT value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the CAPTX_FILT_CNT+3 power.
- The values of FILT_PER and CAPTX_FILT_CNT must also be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting CAPTX_FILT_PER to a non-zero value) introduces a latency of

((CAPTX_FILT_CNT+4) x CAPTX_FILT_PER x IPBus clock period). Note that even when the filter is enabled, there is a combinational path to disable the PWM outputs. This is to ensure rapid response to fault conditions and also to ensure fault response if the PWM module loses its clock. The latency induced by the filter will be seen in the time to set FSTS[FFLAG] and FSTS[FFPIN].

Offset

Register	Offset
SM0CAPTFILTX	5Eh
SM1CAPTFILTX	BEh
SM2CAPTFILTX	11Eh
SM3CAPTFILTX	17Eh

Diagram



Fields

Field	Description
15-11 —	RESERVED
10-8 CAPTX_FILT_CNT	Fault Filter Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. The number of samples is the decimal value of this field plus three: the bit field value of 0-7 represents 3-10 samples, respectively. The value of CAPTX_FILT_CNT affects the input latency.
7-0 CAPTX_FILT_PER	Fault Filter Period This 8-bit field applies universally to all fault inputs. These bits represent the sampling period (in IPBus clock cycles) of the fault pin input filter. Each input is sampled multiple times at the rate specified by this field. If CAPTX_FILT_PER is 0x00 (default), then the input filter is bypassed. The value of CAPTX_FILT_PER affects the input latency. **NOTE** When changing values for CAPTX_FILT_PER from one non-zero value to another non-zero value, first write a value of zero to clear the filter.

44.8.1.46 Phase Delay Register (SM1PHASEDLY - SM3PHASEDLY)

The 16-bit unsigned value in this buffered, read/write register defines the delay from the master sync signal of submodule 0 to the time that this submodule recognizes the master sync in PWM clock periods. CTRL2[INIT_SEL] must be set to 10b in order to select the master sync signal as the source for initialization when using this register. Setting this register with a non-zero value

and using the master sync signal as the initialization source, allows the output of this submodule to be a fixed number of cycles delayed from submodule 0. For PWM operation, the buffered contents of this register are updated at the start of every PWM cycle. This register is not byte accessible.

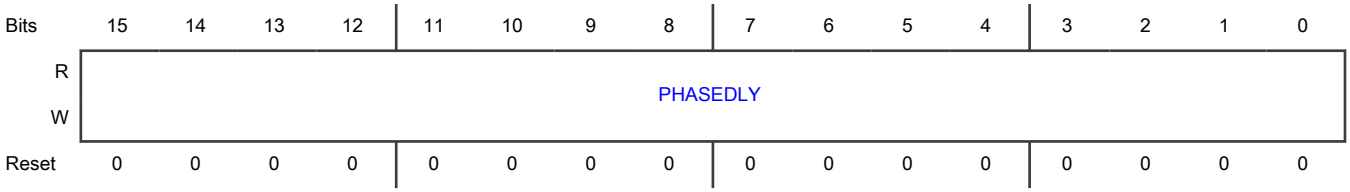
NOTE

The PHASEDLY register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. This register cannot be written when MCTRL[LDOK] is set. Reading PHASEDLY reads the value in a buffer and not necessarily the value the PWM generator is currently using. Also note, the value of this register should not be set to a value larger than the period defined in submodule 0.

Offset

Register	Offset
SM1PHASEDLY	B8h
SM2PHASEDLY	118h
SM3PHASEDLY	178h

Diagram



Fields

Field	Description
15-0 PHASEDLY	Initial Count Register Bits

44.8.1.47 Output Enable Register (OUTEN)

Contains PWM output enables.

Offset

Register	Offset
OUTEN	180h

Diagram

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

Fields

Field	Description
15-12 —	RESERVED
11-8 PWMA_EN	PWM_A Output Enables The four bits of this field enable the PWM_A outputs of submodules 3-0, respectively. These bits should be set to 0 (output disabled) when a PWM_A pin is being used for input capture. 0000 - PWM_A output disabled. 0001 - PWM_A output enabled.
7-4 PWMB_EN	PWM_B Output Enables The four bits of this field enable the PWM_B outputs of submodules 3-0, respectively. These bits should be set to 0 (output disabled) when a PWM_B pin is being used for input capture. 0000 - PWM_B output disabled. 0001 - PWM_B output enabled.
3-0 PWMX_EN	PWM_X Output Enables The four bits of this field enable the PWM_X outputs of submodules 3-0, respectively. These bits should be set to 0 (output disabled) when a PWM_X pin is being used for input capture or deadtime correction. 0000 - PWM_X output disabled. 0001 - PWM_X output enabled.

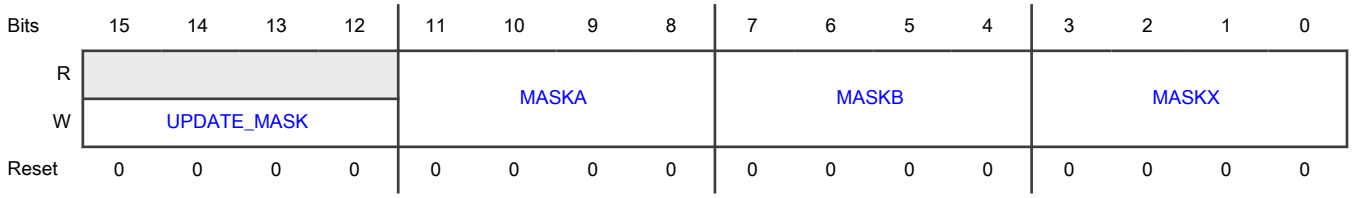
44.8.1.48 Mask Register (MASK)

MASK is double buffered and does not take effect until a FORCE_OUT event occurs within the appropriate submodule. Reading MASK reads the buffered values and not necessarily the values currently in effect. This double buffering can be overridden by setting the UPDATE_MASK bits.

Offset

Register	Offset
MASK	182h

Diagram



Fields

Field	Description
15-12 UPDATE_MASK	Update Mask Bits Immediately The four bits mask the PWM_X outputs of submodules 3-0, respectively, The four bits of this field force the MASK* bits to be immediately updated within submodules 3-0, respectively, without waiting for a FORCE_OUT event. These self-clearing bits always read as zero. Software may write to any or all of these bits and may set these bits in the same write operation that updates the MASKA, MASKB, and MASKX fields of this register. 0000 - Normal operation. MASK* bits within the corresponding submodule are not updated until a FORCE_OUT event occurs within the submodule. 0001 - Immediate operation. MASK* bits within the corresponding submodule are updated on the following clock edge after setting this bit.
11-8 MASKA	PWM_A Masks The four bits of this field mask the PWM_A outputs of submodules 3-0, respectively, forcing the output to logic 0 prior to consideration of the output polarity. 0000 - PWM_A output normal. 0001 - PWM_A output masked.
7-4 MASKB	PWM_B Masks The four bits of this field mask the PWM_B outputs of submodules 3-0, respectively, forcing the output to logic 0 prior to consideration of the output polarity. 0000 - PWM_B output normal. 0001 - PWM_B output masked.
3-0 MASKX	PWM_X Masks The four bits of this field mask the PWM_X outputs of submodules 3-0, respectively, forcing the output to logic 0 prior to consideration of the output polarity. 0000 - PWM_X output normal. 0001 - PWM_X output masked.

44.8.1.49 Software Controlled Output Register (SWCOUT)

These bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Reading these bits reads the buffered value and not necessarily the value currently in effect.

Offset

Register	Offset
SWCOUT	184h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								SM3OUT23	SM3OUT45	SM2OUT23	SM2OUT45	SM1OUT23	SM1OUT45	SM0OUT23	SM0OUT45
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
15-8 —	RESERVED
7 SM3OUT23	Submodule 3 Software Controlled Output 23 This bit is only used when DTSRCSEL[SM3SEL23] is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 - A logic 0 is supplied to the deadtime generator of submodule 3 instead of PWM23. 1 - A logic 1 is supplied to the deadtime generator of submodule 3 instead of PWM23.
6 SM3OUT45	Submodule 3 Software Controlled Output 45 This bit is only used when DTSRCSEL[SM3SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 - A logic 0 is supplied to the deadtime generator of submodule 3 instead of PWM45. 1 - A logic 1 is supplied to the deadtime generator of submodule 3 instead of PWM45.
5 SM2OUT23	Submodule 2 Software Controlled Output 23 This bit is only used when DTSRCSEL[SM2SEL23] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 - A logic 0 is supplied to the deadtime generator of submodule 2 instead of PWM23. 1 - A logic 1 is supplied to the deadtime generator of submodule 2 instead of PWM23.
4 SM2OUT45	Submodule 2 Software Controlled Output 45 This bit is only used when DTSRCSEL[SM2SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 - A logic 0 is supplied to the deadtime generator of submodule 2 instead of PWM45. 1 - A logic 1 is supplied to the deadtime generator of submodule 2 instead of PWM45.
3	Submodule 1 Software Controlled Output 23

Table continues on the next page...

Table continued from the previous page...

Field	Description
SM1OUT23	This bit is only used when DTSRCSEL[SM1SEL23] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 - A logic 0 is supplied to the deadtime generator of submodule 1 instead of PWM23. 1 - A logic 1 is supplied to the deadtime generator of submodule 1 instead of PWM23.
2 SM1OUT45	Submodule 1 Software Controlled Output 45 This bit is only used when DTSRCSEL[SM1SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 - A logic 0 is supplied to the deadtime generator of submodule 1 instead of PWM45. 1 - A logic 1 is supplied to the deadtime generator of submodule 1 instead of PWM45.
1 SM0OUT23	Submodule 0 Software Controlled Output 23 This bit is only used when DTSRCSEL[SM0SEL23] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 - A logic 0 is supplied to the deadtime generator of submodule 0 instead of PWM23. 1 - A logic 1 is supplied to the deadtime generator of submodule 0 instead of PWM23.
0 SM0OUT45	Submodule 0 Software Controlled Output 45 This bit is only used when DTSRCSEL[SM0SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0 - A logic 0 is supplied to the deadtime generator of submodule 0 instead of PWM45. 1 - A logic 1 is supplied to the deadtime generator of submodule 0 instead of PWM45.

44.8.1.50 PWM Source Select Register (DTSRCSEL)

The PWM source select bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Reading these bits reads the buffered value and not necessarily the value currently in effect.

Offset

Register	Offset
DTSRCSEL	186h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SM3SEL23		SM3SEL45		SM2SEL23		SM2SEL45		SM1SEL23		SM1SEL45		SM0SEL23		SM0SEL45	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
15-14 SM3SEL23	Submodule 3 PWM23 Control Select This field selects possible over-rides to the generated SM3PWM23 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 - Generated SM3PWM23 signal is used by the deadtime logic. 01 - Inverted generated SM3PWM23 signal is used by the deadtime logic. 10 - SWCOUT[SM3OUT23] is used by the deadtime logic. 11 - PWM3_EXTA signal is used by the deadtime logic.
13-12 SM3SEL45	Submodule 3 PWM45 Control Select This field selects possible over-rides to the generated SM3PWM45 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 - Generated SM3PWM45 signal is used by the deadtime logic. 01 - Inverted generated SM3PWM45 signal is used by the deadtime logic. 10 - SWCOUT[SM3OUT45] is used by the deadtime logic. 11 - PWM3_EXTB signal is used by the deadtime logic.
11-10 SM2SEL23	Submodule 2 PWM23 Control Select This field selects possible over-rides to the generated SM2PWM23 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 - Generated SM2PWM23 signal is used by the deadtime logic. 01 - Inverted generated SM2PWM23 signal is used by the deadtime logic. 10 - SWCOUT[SM2OUT23] is used by the deadtime logic. 11 - PWM2_EXTA signal is used by the deadtime logic.
9-8 SM2SEL45	Submodule 2 PWM45 Control Select This field selects possible over-rides to the generated SM2PWM45 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that sudeadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 - Generated SM2PWM45 signal is used by the deadtime logic. 01 - Inverted generated SM2PWM45 signal is used by the deadtime logic. 10 - SWCOUT[SM2OUT45] is used by the deadtime logic.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	11 - PWM2_EXTB signal is used by the deadtime logic.
7-6 SM1SEL23	Submodule 1 PWM23 Control Select This field selects possible over-rides to the generated SM1PWM23 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 - Generated SM1PWM23 signal is used by the deadtime logic. 01 - Inverted generated SM1PWM23 signal is used by the deadtime logic. 10 - SWCOUT[SM1OUT23] is used by the deadtime logic. 11 - PWM1_EXTA signal is used by the deadtime logic.
5-4 SM1SEL45	Submodule 1 PWM45 Control Select This field selects possible over-rides to the generated SM1PWM45 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 - Generated SM1PWM45 signal is used by the deadtime logic. 01 - Inverted generated SM1PWM45 signal is used by the deadtime logic. 10 - SWCOUT[SM1OUT45] is used by the deadtime logic. 11 - PWM1_EXTB signal is used by the deadtime logic.
3-2 SM0SEL23	Submodule 0 PWM23 Control Select This field selects possible over-rides to the generated SM0PWM23 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 - Generated SM0PWM23 signal is used by the deadtime logic. 01 - Inverted generated SM0PWM23 signal is used by the deadtime logic. 10 - SWCOUT[SM0OUT23] is used by the deadtime logic. 11 - PWM0_EXTA signal is used by the deadtime logic.
1-0 SM0SEL45	Submodule 0 PWM45 Control Select This field selects possible over-rides to the generated SM0PWM45 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00 - Generated SM0PWM45 signal is used by the deadtime logic. 01 - Inverted generated SM0PWM45 signal is used by the deadtime logic. 10 - SWCOUT[SM0OUT45] is used by the deadtime logic. 11 - PWM0_EXTB signal is used by the deadtime logic.

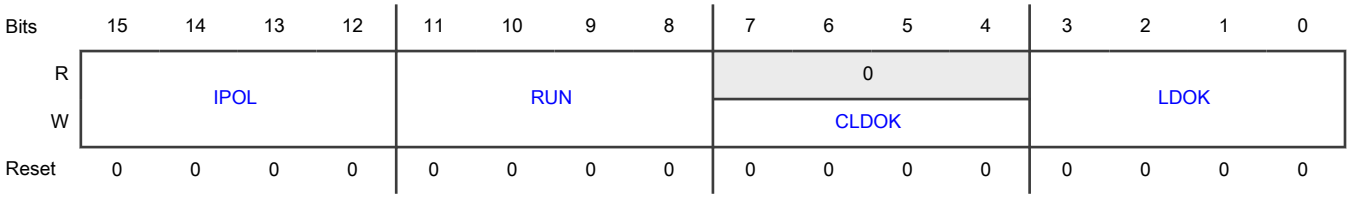
44.8.1.51 Master Control Register (MCTRL)

In every 4-bit field in this register, each bit acts on a separate submodule. Accordingly, the description of every bit field refers to the effect of an individual bit.

Offset

Register	Offset
MCTRL	188h

Diagram



Fields

Field	Description
15-12 IPOL	Current Polarity The four buffered read/write bits of this field correspond to submodules 3-0, respectively. Each bit selects between PWM23 and PWM45 as the source for the generation of the complementary PWM pair output for the corresponding submodule. MCTRL[IPOL] is ignored in independent mode. MCTRL[IPOL] does not take effect until a FORCE_OUT event takes place in the appropriate submodule. Reading MCTRL[IPOL] reads the buffered value and not necessarily the value currently in effect. 0000 - PWM23 is used to generate complementary PWM pair in the corresponding submodule. 0001 - PWM45 is used to generate complementary PWM pair in the corresponding submodule.
11-8 RUN	Run The four read/write bits of this field enable the clocks to the PWM generator of submodules 3-0, respectively. The corresponding MCTRL[RUN] bit must be set for each submodule that is using its input capture functions or is using the local reload as its reload source. When this bit equals zero, the submodule counter is reset and PWM outputs are held. A reset clears this field. 0000 - PWM counter is stopped, but PWM outputs will hold the current state. 0001 - PWM counter is started in the corresponding submodule.
7-4 CLDOK	Clear Load Okay The 4 bits of CLDOK field correspond to submodules 3-0, respectively. Each write-only bit is used to clear the corresponding bit of MCTRL[LDOK]. Write a 1 to CLDOK to clear the corresponding MCTRL[LDOK] bit. If a reload occurs within a submodule with the corresponding MCTRL[LDOK] bit set at the same time that MCTRL[CLDOK] is written, then the reload in that submodule will not be performed and MCTRL[LDOK] will be cleared. CLDOK bit is self-clearing and always reads as a 0.
3-0 LDOK	Load Okay The 4 bits of LDOK field correspond to submodules 3-0, respectively. Each read/set bit loads CTRL[PRSC] and the INIT, FRACVALx, and VALx registers of the corresponding submodule into a set of buffers. The buffered prescaler divisor, submodule counter modulus value, and PWM pulse width take effect at the next PWM reload if CTRL[LDMOD] is clear or immediately if CTRL[LDMOD] is set. Set the corresponding

Table continues on the next page...

Table continued from the previous page...

Field	Description
	MCTRL[LDOK] bit by reading it when it is logic zero and then writing a logic one to it. The VALx, FRACVALx, INIT, and CTRL[PRSC] registers of the corresponding submodule cannot be written while the corresponding MCTRL[LDOK] bit is set. In Master Reload Mode (CTRL2[RELOAD_SEL]=1), it is only necessary to set the LDOK bit corresponding to submodule0; however, it is recommended to also set the LDOK bit of the slave submodules, to prevent unwanted writes to the registers in the slave submodules. The MCTRL[LDOK] bit is automatically cleared after the new values are loaded, or it can be manually cleared before a reload by writing a logic 1 to the appropriate MCTRL[CLDOK] bit. LDOK bits cannot be written with a zero. MCTRL[LDOK] can be set in DMA mode when the DMA indicates that it has completed the update of all CTRL[PRSC], INIT, FRACVALx, and VALx registers in the corresponding submodule. Reset clears LDOK field. 0000 - Do not load new values. 0001 - Load prescaler, modulus, and PWM values of the corresponding submodule.

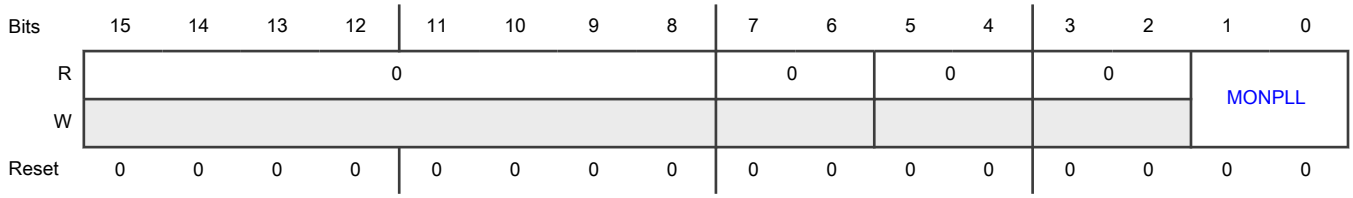
44.8.1.52 Master Control 2 Register (MCTRL2)

Includes control for monitoring the PLL state and write protection of some configuration registers.

Offset

Register	Offset
MCTRL2	18Ah

Diagram



Fields

Field	Description
15-8 —	RESERVED
7-6 —	RESERVED
5-4	RESERVED

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
3-2 —	RESERVED
1-0 MONPLL	Monitor PLL State These bits are used to control disabling of the fractional delay block when the chip PLL is unlocked and/or missing its input reference. The fractional delay block requires a continuous 200 MHz clock from the PLL. If this clock turns off when the fractional delay block is being used, then the output of the fractional delay block can be stuck high or low even if the PLL restarts. When this control bit is set, PLL problems cause the fractional delay block to be disabled until the PLL returns to a locked state. Once the PLL is receiving a proper reference and is locked, the fractional delay block requires a 25 µs startup time just as if the FRCTRL[FRAC*_EN] bits had been turned off and turned on again. If PLL monitoring is disabled, then software should manually clear and then set the FRCTRL[FRAC*_EN] bits when the PLL loses its reference or loses lock. This will cause the fractional delay block to be disabled and restarted. If the fractional delay block is not being used, then the value of these bits do not matter. 00 - Not locked. Do not monitor PLL operation. Resetting of the fractional delay block in case of PLL losing lock will be controlled by software. 01 - Not locked. Monitor PLL operation to automatically disable the fractional delay block when the PLL encounters problems. 10 - Locked. Do not monitor PLL operation. Resetting of the fractional delay block in case of PLL losing lock will be controlled by software. These bits are write protected until the next reset. 11 - Locked. Monitor PLL operation to automatically disable the fractional delay block when the PLL encounters problems. These bits are write protected until the next reset.

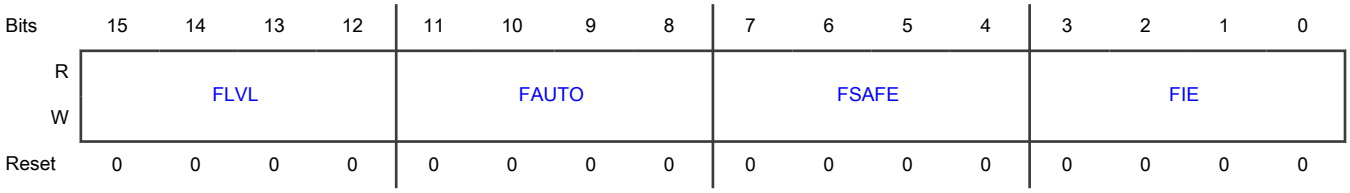
44.8.1.53 Fault Control Register (FCTRL0)

For every 4-bit field in this register, the bits act on the fault inputs in order. For example, FLVL bits 15-12 act on faults 3-0, respectively.

Offset

Register	Offset
FCTRL0	18Ch

Diagram



Fields

Field	Description
15-12 FLVL	Fault Level The four read/write bits of this field select the active logic level of the individual fault inputs 3-0, respectively. A reset clears this field. 0000 - A logic 0 on the fault input indicates a fault condition. 0001 - A logic 1 on the fault input indicates a fault condition.
11-8 FAUTO	Automatic Fault Clearing The four read/write bits of this field select automatic or manual clearing of faults 3-0, respectively. A reset clears this field. 0000 - Manual fault clearing. PWM outputs disabled by this fault are not enabled until FSTS[FFLAGx] is clear at the start of a half cycle or full cycle depending on the states of FSTS[FHALF] and FSTS[FFULL]. If neither FFULL nor FHALF is set, then the fault condition cannot be cleared. This is further controlled by FCTRL[FSAFE]. 0001 - Automatic fault clearing. PWM outputs disabled by this fault are enabled when FSTS[FFPINx] is clear at the start of a half cycle or full cycle depending on the states of FSTS[FHALF] and FSTS[FFULL] without regard to the state of FSTS[FFLAGx]. If neither FFULL nor FHALF is set, then the fault condition cannot be cleared.
7-4 FSAFE	Fault Safety Mode These read/write bits select the safety mode during manual fault clearing. A reset clears this field. FSTS[FFPINx] may indicate a fault condition still exists even though the actual fault signal at the FAULTx pin is clear due to the fault filter latency. 0000 - Normal mode. PWM outputs disabled by this fault are not enabled until FSTS[FFLAGx] is clear at the start of a half cycle or full cycle depending on the states of FSTS[FHALF] and FSTS[FFULL] without regard to the state of FSTS[FFPINx]. If neither FHALF nor FFULL is set then the fault condition cannot be cleared. The PWM outputs disabled by this fault input will not be re-enabled until the actual FAULTx input signal de-asserts since the fault input will combinationally disable the PWM outputs (as programmed in DISMAPn). 0001 - Safe mode. PWM outputs disabled by this fault are not enabled until FSTS[FFLAGx] is clear and FSTS[FFPINx] is clear at the start of a half cycle or full cycle depending on the states of FSTS[FHALF] and FSTS[FFULL]. If neither FHLAF nor FFULL is set, then the fault condition cannot be cleared.
3-0 FIE	Fault Interrupt Enables This read/write field enables CPU interrupt requests generated by the FAULTx pins. A reset clears this field.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	NOTE The fault protection circuit is independent of the FIEx bit and is always active. If a fault is detected, the PWM outputs are disabled according to the disable mapping register. 0000 - FAULTx CPU interrupt requests disabled. 0001 - FAULTx CPU interrupt requests enabled.

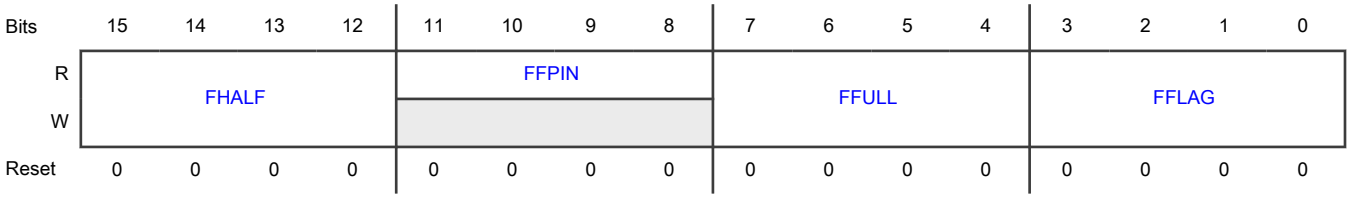
44.8.1.54 Fault Status Register (FSTS0)

Includes controls related to fault conditions.

Offset

Register	Offset
FSTS0	18Eh

Diagram



Fields

Field	Description
15-12 FHALF	Half Cycle Fault Recovery These read/write bits are used to control the timing for re-enabling the PWM outputs after a fault condition. These bits apply to both automatic and manual clearing of a fault condition. **NOTE** Both FHALF and FFULL can be set so that the fault recovery occurs at the start of a full cycle and at the start of a half cycle (as defined by VAL0). If neither FHALF nor FFULL is set, then no fault recovery is possible. 0000 - PWM outputs are not re-enabled at the start of a half cycle. 0001 - PWM outputs are re-enabled at the start of a half cycle (as defined by VAL0).
11-8 FFPIN	Filtered Fault Pins These read-only bits reflect the current state of the filtered FAULTx pins converted to high polarity. A logic 1 indicates a fault condition exists on the filtered FAULTx pin. A reset has no effect on this field.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	After the system reset de-asserts, these are the possible values of FFPIN: If FCTRL[FLVL] = 0 and FAULTx = 0, then FFPIN is set to 1. If FCTRL[FLVL] = 0 and FAULTx = 1, then FFPIN is kept as 0. If FCTRL[FLVL] = 1 and FAULTx = 0, then FFPIN is kept as 0. If FCTRL[FLVL] = 1 and FAULTx = 1, then FFPIN is set to 1.
7-4 FFULL	Full Cycle These read/write bits are used to control the timing for re-enabling the PWM outputs after a fault condition. These bits apply to both automatic and manual clearing of a fault condition. **NOTE** Both FHALF and FFULL can be set so that the fault recovery occurs at the start of a full cycle and at the start of a half cycle (as defined by VAL0). If neither FHALF nor FFULL is set, then no fault recovery is possible. 0000 - PWM outputs are not re-enabled at the start of a full cycle 0001 - PWM outputs are re-enabled at the start of a full cycle
3-0 FFLAG	Fault Flags These read-only flags are set within two CPU cycles after a transition to active on the FAULTx pin. Clear this bit by writing a logic one to it. A reset clears this field. While the reset value is 0, these bits may be set to 1 by the time they can be read depending on the state of the fault input signals. After the system reset de-asserts, these are the possible values of FFLAG: If FCTRL[FLVL] = 0 and FAULTx = 0, then FFLAG is set to 1. If FCTRL[FLVL] = 0 and FAULTx = 1, then FFLAG is kept as 0. If FCTRL[FLVL] = 1 and FAULTx = 0, then FFLAG is kept as 0. If FCTRL[FLVL] = 1 and FAULTx = 1, then FFLAG is set to 1. 0000 - No fault on the FAULTx pin. 0001 - Fault on the FAULTx pin.

44.8.1.55 Fault Filter Register (FFILT0)

The settings in this register are shared among each of the fault input filters within the fault channel.

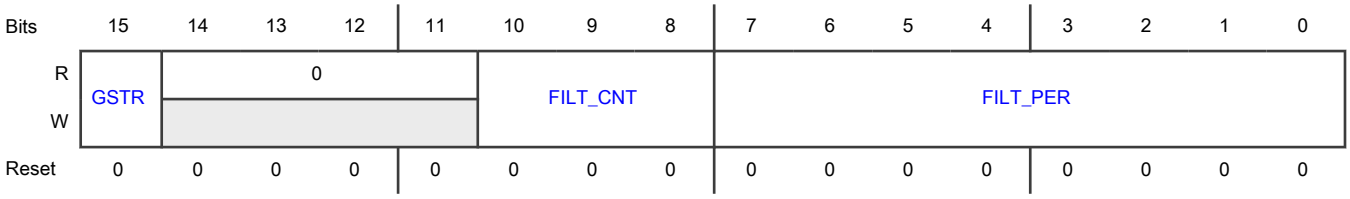
Input filter considerations include:

- The FILT_PER value should be set such that the sampling period is larger than the period of the expected noise. This way a noise spike will only corrupt one sample. The FILT_CNT value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the FILT_CNT+3 power.
- The values of FILT_PER and FILT_CNT must also be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting FILT_PER to a non-zero value) introduces a latency of ((FILT_CNT+4) x FILT_PER x IPBus clock period). Note that even when the filter is enabled, there is a combinational path to disable the PWM outputs. This is to ensure rapid response to fault conditions and also to ensure fault response if the PWM module loses its clock. The latency induced by the filter will be seen in the time to set FSTS[FFLAG] and FSTS[FFPIN].

Offset

Register	Offset
FFILT0	190h

Diagram



Fields

Field	Description
15 GSTR	Fault Glitch Stretch Enable This bit is used to enable the fault glitch-stretching logic. This logic ensures that narrow fault glitches are stretched to be at least 2 IPBus clock cycles wide. In some cases a narrow fault input can cause problems due to the short PWM output shutdown/re-activation time. The stretching logic ensures that a glitch on the fault input, when the fault filter is disabled, will be registered in the fault flags. 0 - Fault input glitch stretching is disabled. 1 - Input fault signals will be stretched to at least 2 IPBus clock cycles.
14-11 —	RESERVED
10-8 FILT_CNT	Fault Filter Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. The number of samples is the decimal value of this field plus three: the bit field value of 0-7 represents 3-10 samples, respectively. The value of FILT_CNT affects the input latency.
7-0 FILT_PER	Fault Filter Period This 8-bit field applies universally to all fault inputs. These bits represent the sampling period (in IPBus clock cycles) of the fault pin input filter. Each input is sampled multiple times at the rate specified by this field. If FILT_PER is 0x00 (default), then the input filter is bypassed. The value of FILT_PER affects the input latency. **NOTE** When changing values for FILT_PER from one non-zero value to another non-zero value, first write a value of zero to clear the filter.

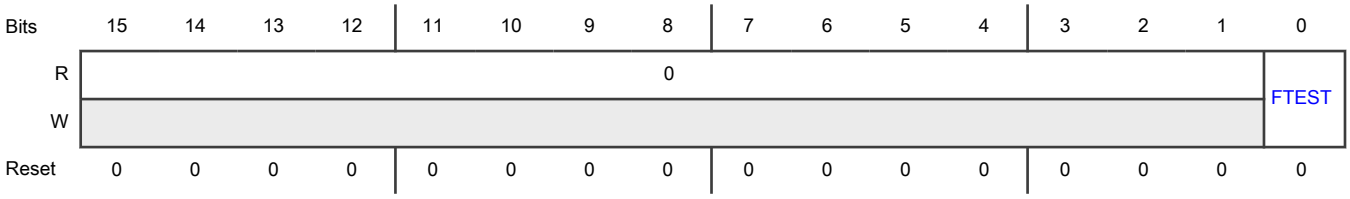
44.8.1.56 Fault Test Register (FTST0)

Contains FTEST field for fault simulation.

Offset

Register	Offset
FTST0	192h

Diagram



Fields

Field	Description
15-1 —	RESERVED
0 FTEST	Fault Test This read/write bit is used to simulate a fault condition. Setting this bit causes a simulated fault to be sent into all of the fault filters. The condition propagates to the fault flags and possibly the PWM outputs depending on the DISMAPn settings. Clearing this bit removes the simulated fault condition. 0 - No fault 1 - Cause a simulated fault

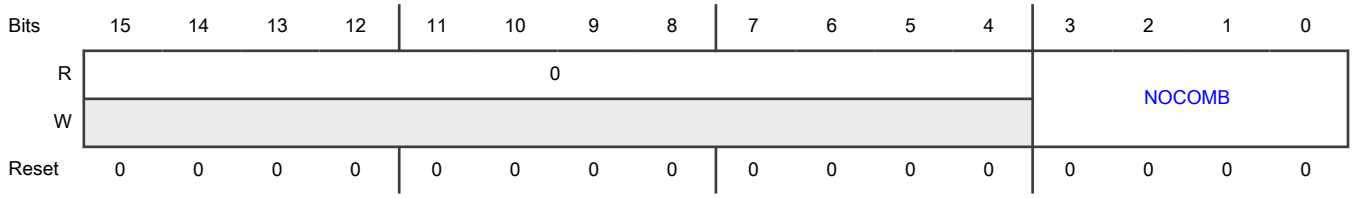
44.8.1.57 Fault Control 2 Register (FCTRL20)

Controls combinational link from fault inputs to PWM outputs.

Offset

Register	Offset
FCTRL20	194h

Diagram



Fields

Field	Description
15-4 —	RESERVED
3-0 NOCOMB	No Combinational Path From Fault Input To PWM Output This read/write field is used to control the combinational path from the fault inputs to the PWM outputs. When these bits are low (default), the corresponding fault inputs have a combinational path to the PWM outputs that are sensitive to these fault inputs (as defined by DISMAP0). This combinational path is a safety feature that ensures the output is disabled even if the SOC has a failure of its clocking system. The combinational path also means that a pulse on the fault input can cause a brief disable of the PWM output even if the fault pulse is not wide enough to get through the input filter and be latched in the fault logic. Setting these bits removes the combinational path and uses the filtered and latched fault signals as the fault source to disable the PWM outputs. This eliminates fault glitches from creating PWM output glitches but also increases the latency to respond to a real fault. 0000 - There is a combinational link from the fault inputs to the PWM outputs. The fault inputs are combined with the filtered and latched fault signals to disable the PWM outputs. 0001 - The direct combinational path from the fault inputs to the PWM outputs is disabled and the filtered and latched fault signals are used to disable the PWM outputs.

Chapter 45
Quadrature Decoder (ENC)

45.1 Chip-specific ENC information

Table 381. Reference links to related information

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

45.1.1 Module instances

This device has two instance of the ENC module, ENC0 and ENC1.

45.2 Overview

The enhanced quadrature decoder module interfaces to position/speed sensors that are used in industrial motor control applications. Using 5 input signals (PHASEA, PHASEB, INDEX, TRIGGER, and HOME) from those position/speed sensors, the quadrature decoder module decodes shaft position, revolution count, and speed.

45.2.1 System Block Diagram

The next figure shows the block diagram of the quadrature decoder module integrated into an SoC.

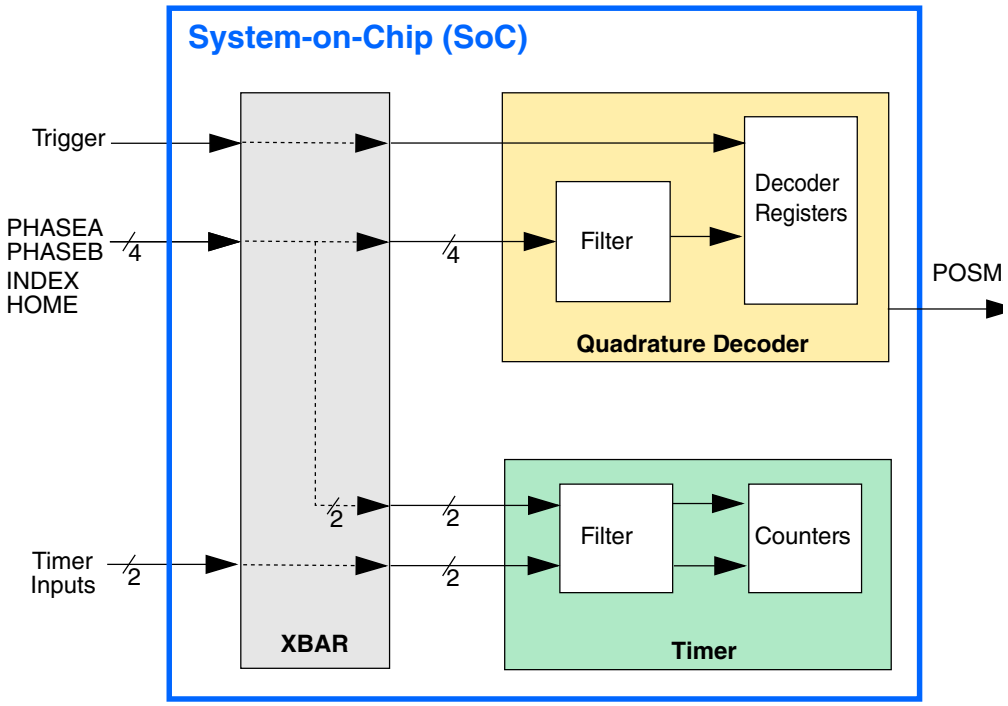


Figure 240. System Block Diagram

45.2.2 Features

- Includes logic to decode quadrature signals
- Inputs can be connected to a general purpose timer to make low speed velocity measurements
- Configurable digital filter for inputs
- Quadrature decoder filter can be bypassed
- 32-bit position counter capable of modulo counting
- Position counter can be initialized by software or external events
- 16-bit position difference register
- Compare function can indicate when shaft has reached a defined position
- A watchdog timer can detect a non-rotating shaft condition
- Preloadable 16-bit revolution counter
- Maximum count frequency equals the peripheral clock rate
- Optional interrupt when both PHASEA and PHASEB inputs change in the same cycle

45.3 Functional Description

The next timing diagram shows the basic operation of a quadrature incremental position quadrature decoder.

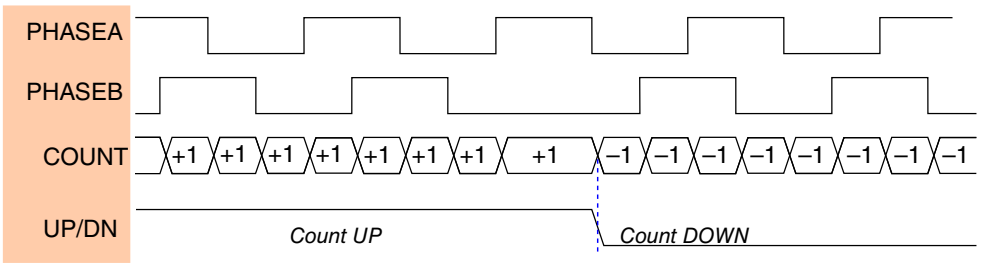


Figure 241. Quadrature Decoder Signals

45.3.1 Decoder Block Diagram

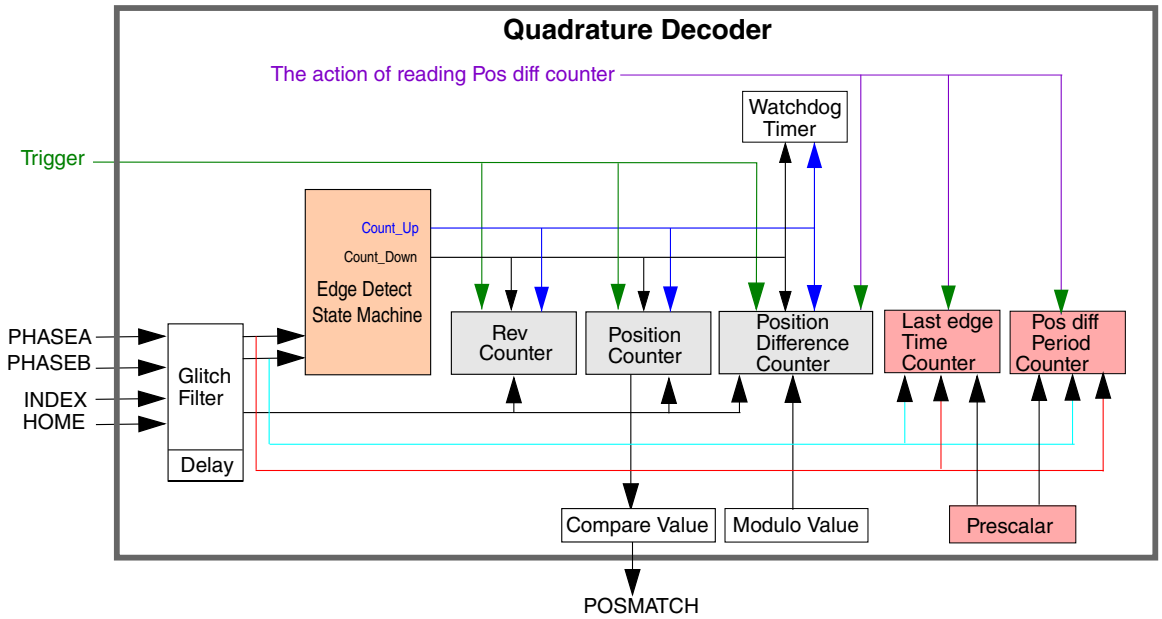


Figure 242. Quadrature Decoder Block Diagram

45.3.1.1 Glitch Filter

Because the quadrature decoder logic must sense signal transitions, the signal inputs are first run through a glitch filter. This glitch filter has a digital delay line, which samples multiple time points on the signal and verifies a stable new signal state, before outputting this new signal state to the internal quadrature decoder logic. To adapt to a variety of signal bandwidths, the sample rate of this delay line is programmable.

45.3.1.2 Edge Detect State Machine

The edge detect state machine looks for changes in the 4 possible states of the filtered PHASEA and PHASEB inputs, which are used to calculate the direction of motion. This information is formatted as Count_Up and Count_Down signals. Each counter has its own hold register. These signals are routed into up to 3 up/down counters:

- Position counter
- Revolution counter
- Position difference counter

45.3.1.3 Position Counter

The 32-bit position counter calculates up or down on every count pulse, generated by the difference of PHASEA and PHASEB. The position counter acts as an integration_info, whose count value is proportional to position. The direction of the count is determined by the Count_Up and Count_Down signals. Position counters may be initialized to a predetermined value by one of 4 different methods:

- Software-triggered event
- INDEX signal transition
- HOME signal transition
- Rising-edge of trigger event

The INDEX and HOME signals can be programmed to interrupt the processor. Whenever the position counters (UPOS or LPOS) are read, a snapshot of the

- position counter
- position difference counter (only if CTRL3[PMEN] is clear)
- revolution counter

are each placed into their respective hold registers.

45.3.1.4 Position Difference Counter

The 16-bit position difference counter contains the position difference value occurring between each read of the position register. The position register counts up or down on every count pulse. The position difference counter acts as a differentiator, whose count value is proportional to the change in position since the last time the position counter was read.

When the position registers, the position difference counter, or the revolution counter is read, the position difference counter's contents are copied into the position difference hold register (POSDH) and the position difference counter is cleared.

If CTRL3[PMEN] is set, then the position difference counter and its hold register are only updated by a read of the postion difference counter itself and not by a read of the revolution counter or position counters.

45.3.1.5 Position Difference Counter Hold

The Position Difference Counter Hold register stores a copy of the position difference counter at the time that the position register was read. When the position register, the position difference counter, or the revolution counter is read, the position difference counter's contents are copied into the position difference hold register (POSDH), and the position difference counter is cleared.

If CTRL3[PMEN] is set, then the position difference hold register is only updated by a read of the postion difference counter itself and not by a read of the revolution counter or position counters.

45.3.1.6 Revolution Counter

The 16-bit up/down revolution counter is intended to count or integrate revolutions by counting index pulses. The direction of the count is determined by the Count_Up and Count_Down signals, determined by the Phase A and B inputs. A different count direction on the rising and falling edges of the index pulse indicates that the quad encoder changed direction on the index pulse.

45.3.1.7 Watchdog Timer

The watchdog timer ensures that the algorithm is indicating motion in the shaft; 2 successive counts indicate proper operation and will reset the timer.

- The timeout value is programmable.
- When a timeout occurs, an interrupt to the processor can be generated.

45.3.1.8 Pulse Accumulator Functionality

The logic can be programmed to integrate only selected transitions of the PHASEA signal. In this way, the position counter can be used as a pulse accumulator.

- The count direction is up.
- The pulse accumulator can also optionally be initialized by the INDEX input.

45.3.1.9 Last Edge Time Counter

The 16-bit last edge time counter contains the time since the last edge on PHASEA or PHASEB. The last edge time counter counts up on every prescaled clock pulse.

When the position difference counter is read, the LASTEDGE counter's contents are copied into the LASTEDGEH hold register.

45.3.1.10 Position Difference Period Counter

The 16-bit position difference period counter contains the accumulated time from the last time the position difference counter was read. The position difference period counter counts up on every prescaled clock pulse. The position difference period counter is loaded from the last edge time counter whenever the position difference register is read.

45.3.2 Positive versus Negative Direction

A typical quad encoder has 3 outputs: PHASEA signal, PHASEB signal, INDEX pulse (not shown).

- If PHASEA leads PHASEB, then motion is in the positive direction.
- If PHASEA trails PHASEB, then motion is in the negative direction.

Transitions on these phases can be integrated to yield position or differentiated to yield velocity. The quadrature decoder is designed to perform these functions in hardware.

45.3.3 Prescaler for Slow or Fast Speed Measurement

- For applications with a fast moving shaft encoder, the speed can be computed by calculating the change in the position counter per unit time, or by reading the position difference counter register (POSD) and calculating speed.
- For applications with slow motor speeds and low line count quad encoders, the timer module enables high resolution velocity measurements by measuring the time period between quad phases.
  - The timer module uses a 16-bit free running counter operated from a prescaled version of the IPBus clock.
  - The prescaler divides the IPBus clock by values ranging from 1 to 128. A 60 MHz IPBus clock frequency would yield a resolution of from 17 ns to 2.1 µs and a maximum count period of from 1.09 ms to 140 ms. For example, with a 1000-tooth decoder, speeds could be calculated down to 0.11 rpm using a prescaler.

45.3.4 Holding Registers and Initializing Registers

Hold registers are associated with 4 types of counters:

- Position
- Position difference
- Revolution
- Rising edge of trigger event when UPDHLD bit is set

When any of the counter registers are read, the contents of each counter register is written to the corresponding hold register. Taking a snapshot of the counters' values allows a consistent view of a system's position and the velocity to be attained. If CTRL3[PMEN] is set, then the POSDH register will only be updated when the POSD counter is read and not when other counters are read. To capture a time stamp of when these registers are read, use the POSMATCH output with a timer channel.

The position counter is 32 bits wide. To ensure that the position counter can be reliably initialized with two 16-bit accesses, two registers (an upper and a lower initialization register) are provided. The upper initialization register (UINIT) and lower initialization register (LINIT) should be loaded with the desired value. Next, the position counter can be loaded by writing 1 to the Software-Triggered Initialization of Position Counters UPOS and LPOS bit (CTRL[SWIP]). Alternatively, either the INDEX-Triggered Initialization of Position Counters UPOS and LPOS bit (CTRL[XIP]) or the Enable HOME to Initialize Position Counters UPOS and LPOS bit (CTRL[HIP]) can enable the position counter to be initialized in response to a HOME or INDEX signal transition.

45.3.5 Speed Measurement Method

This section explains speed measurement method. It includes speed measurement algorithm.

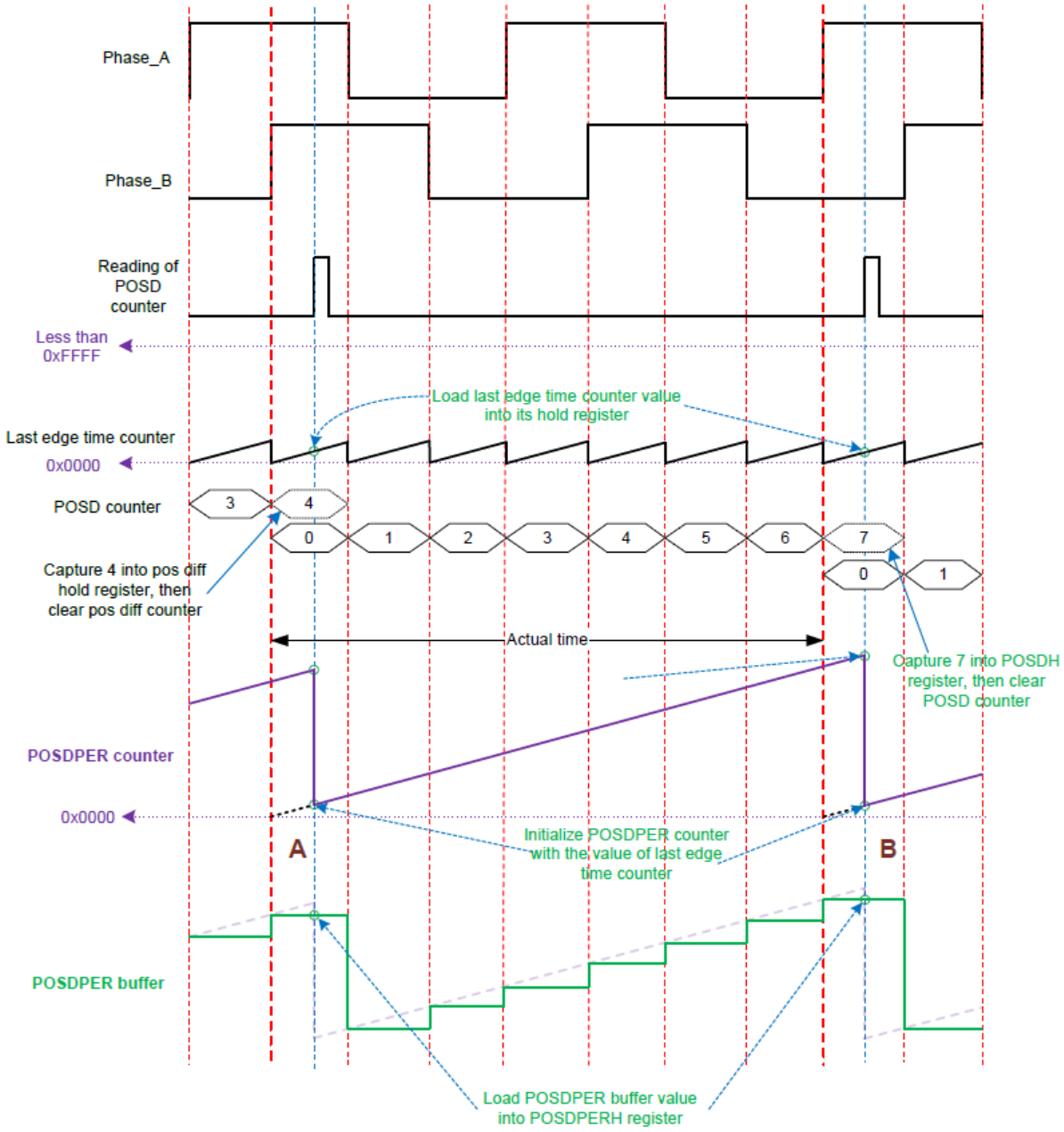


Figure 243. Counters behavior and updating mechanism

A reading of POSD occurs at time point A and it also occurs at time point B. “Actual time” represents the time duration between the first phase_a/b edges right before time point A and B. At time point B, after reading POSD register, POSDPERH contains the time length of “Actual time” and POSDH contains the value of phase_a/b pulses within that “Actual time”, speed can be calculated just based on POSDPERH and POSDH. A visualized explanation of speed measurement with this method is shown in figure below.

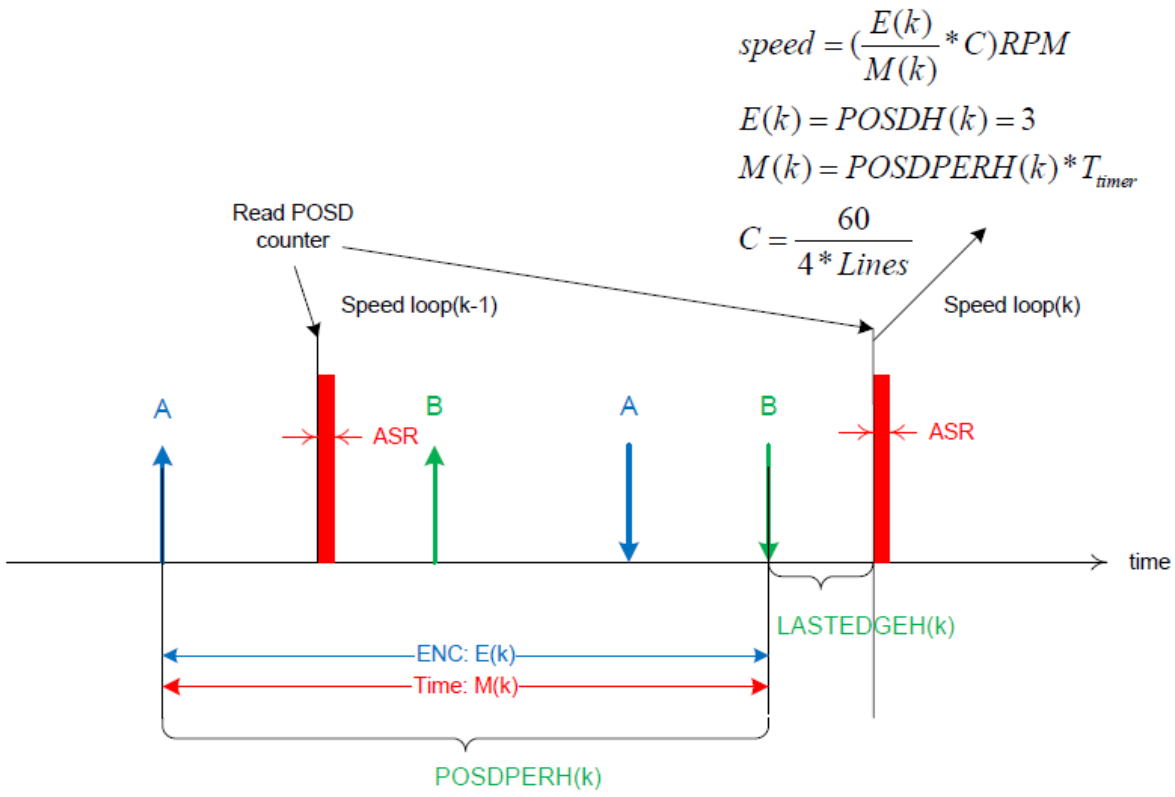
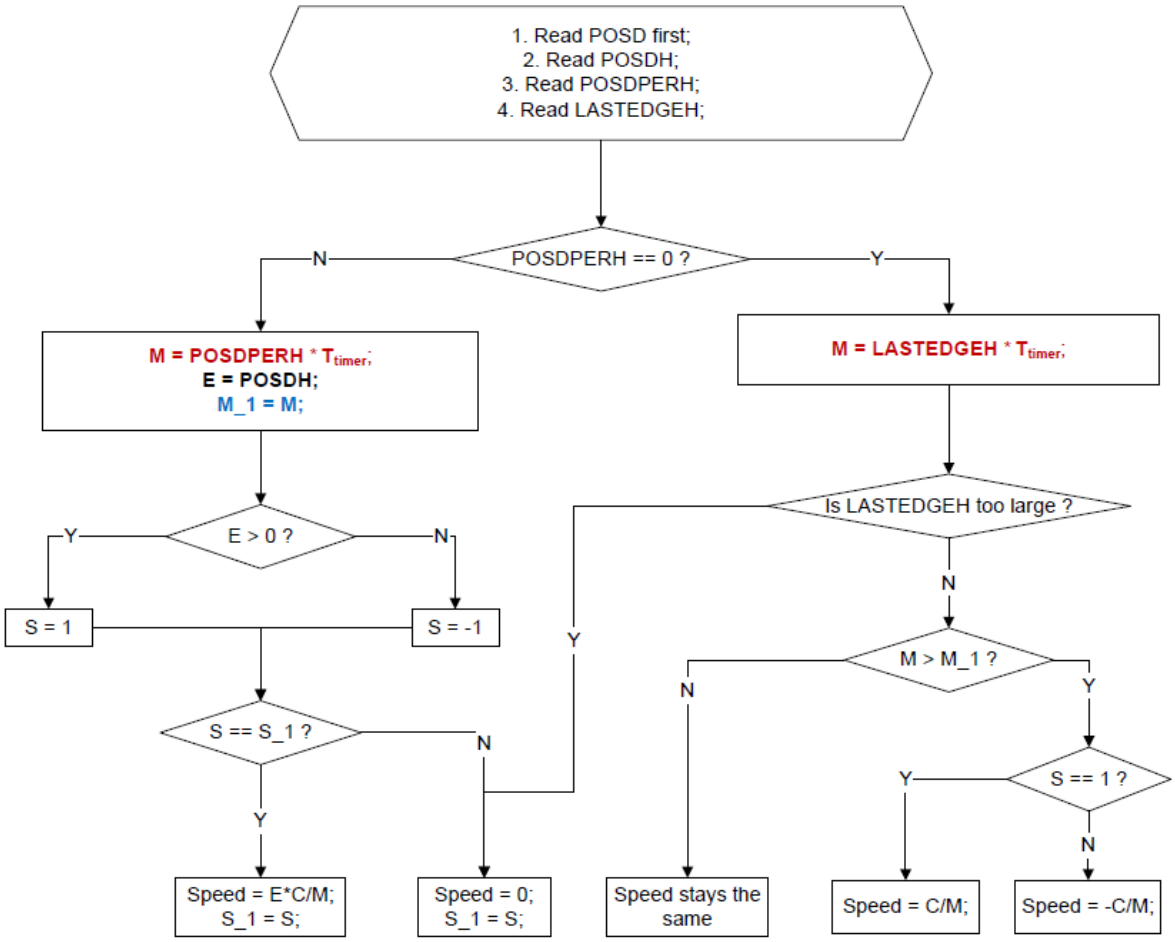


Figure 244. High speed measurement with method

Speed calculation algorithm

However, in order to cover all the cases in real applications, a simple speed measurement algorithm is necessary. Below is the algorithm.



S: Tells if E=0 or not in this POSD reading operation, if E=0, then S=-1, if E>0 S=1
S_1:Tells if E=0 or not in last POSD reading operation, if E=0, then S=-1, if E>0 S=1
M: Depicts M value in this POSD reading operation
M_1: Depicts M value in last POSD reading operation

Figure 245. Flowchart of speed calculation with method

Speed calculation of motor's starting from standstill (Chip is out of reset)

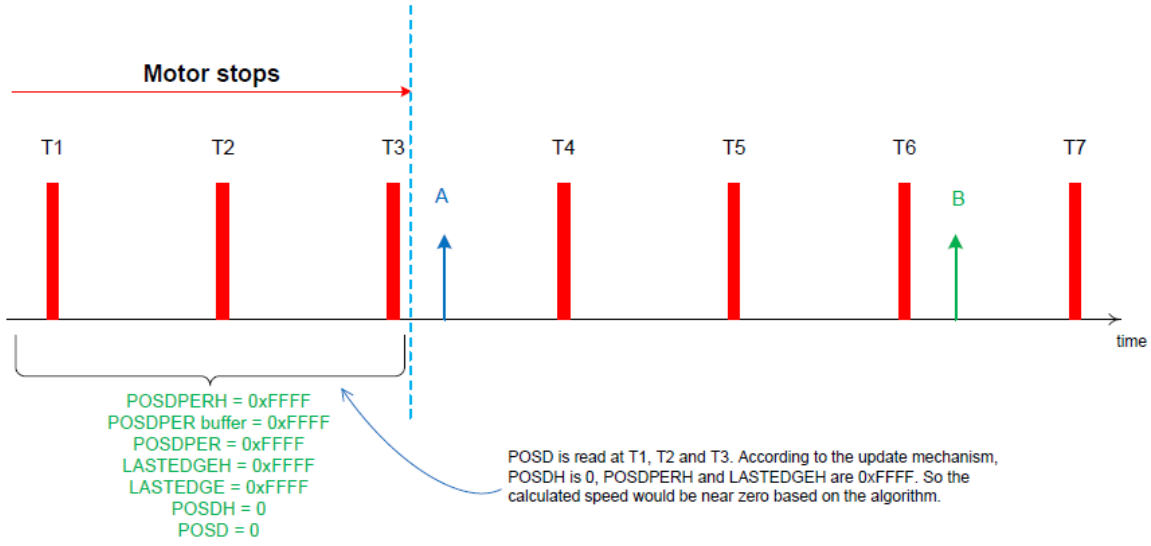


Figure 246. Speed measurement at time point T1~T3

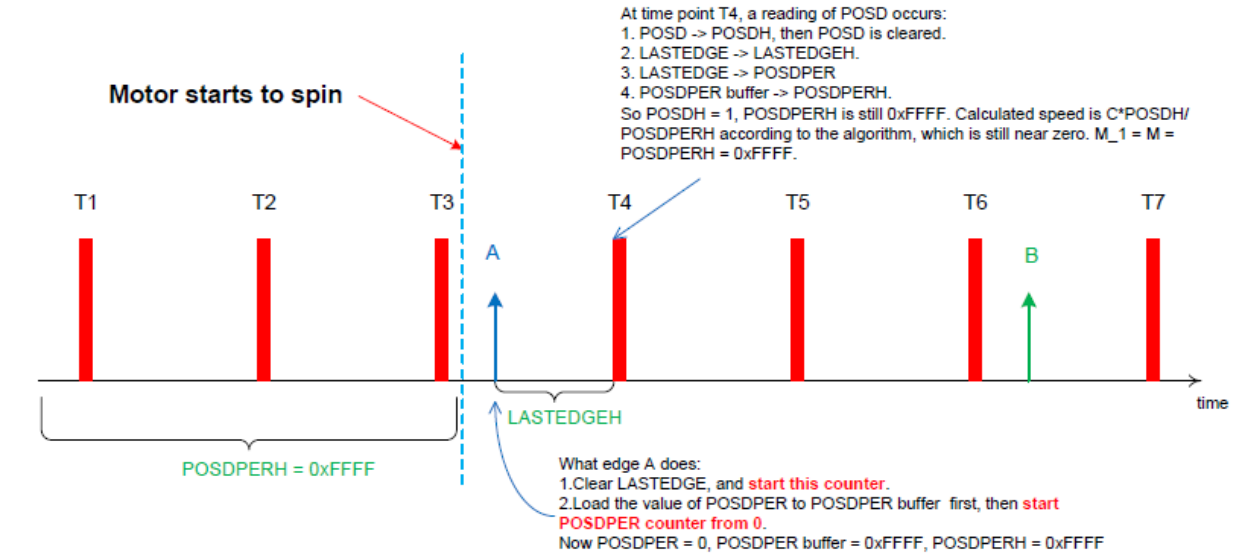


Figure 247. Speed measurement at time point T4

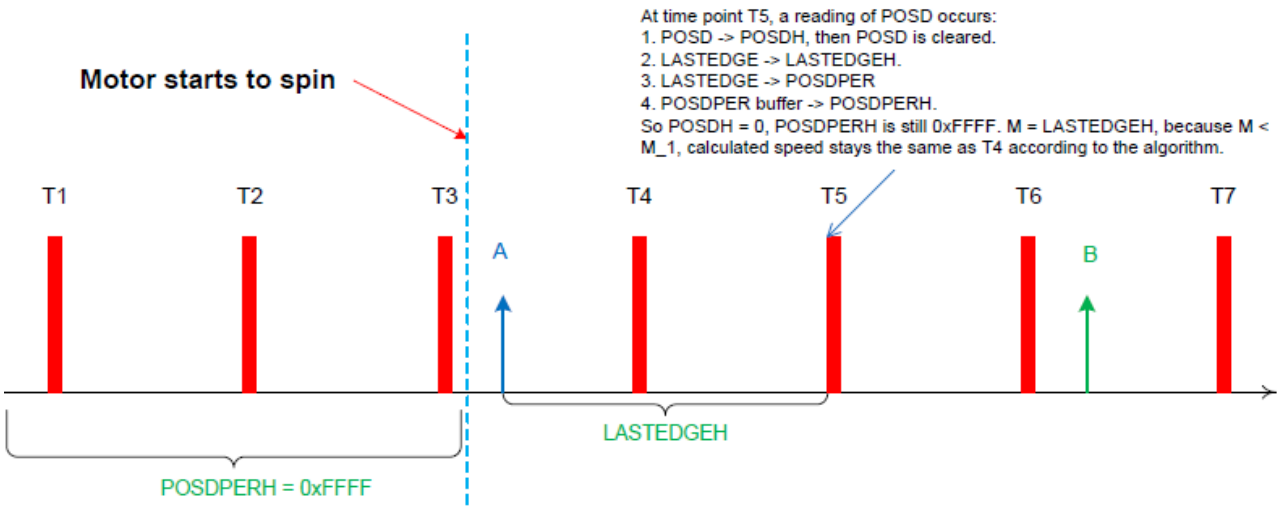


Figure 248. Speed measurement at time point T5

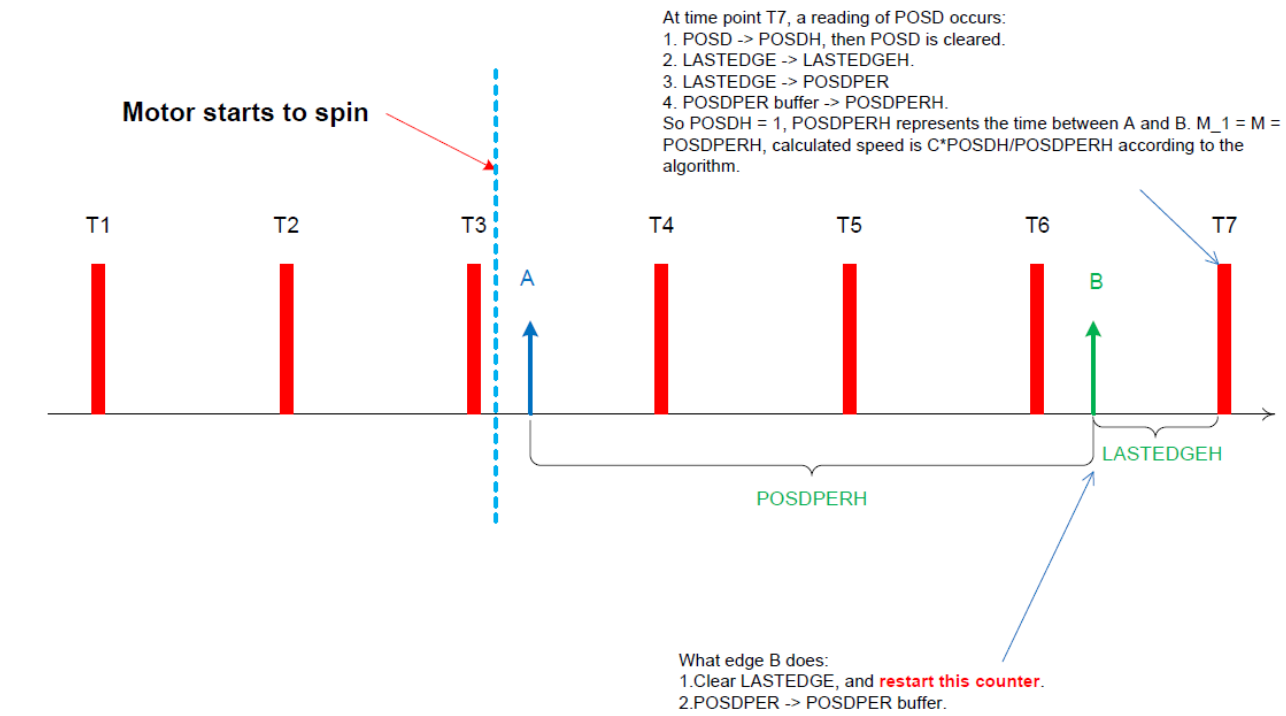


Figure 249. Speed measurement at time point T7

Speed calculation when motor stops

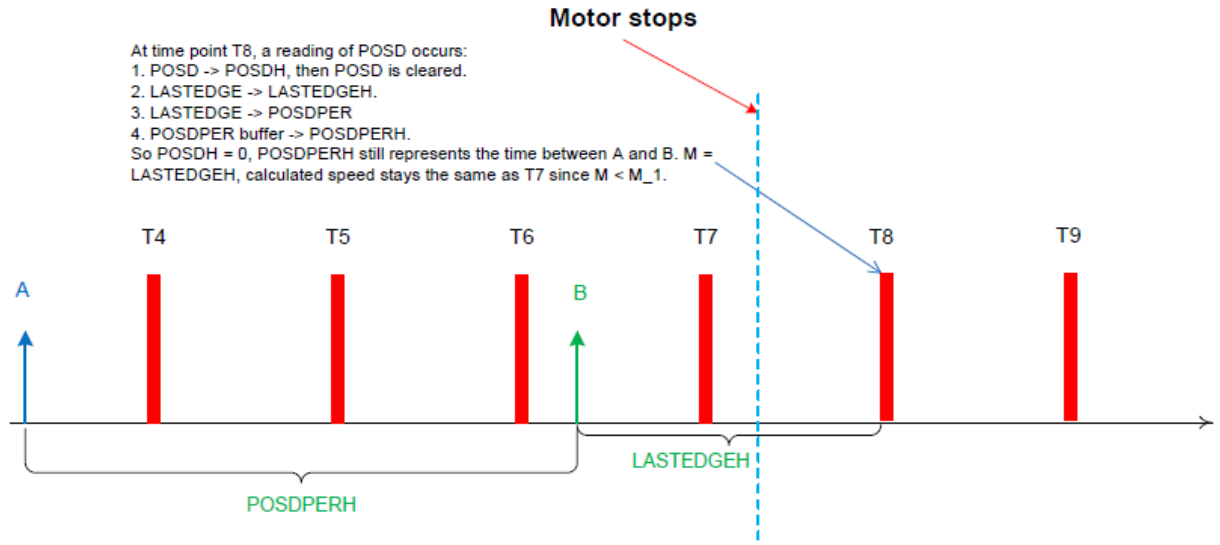


Figure 250. Speed measurement at time point T8

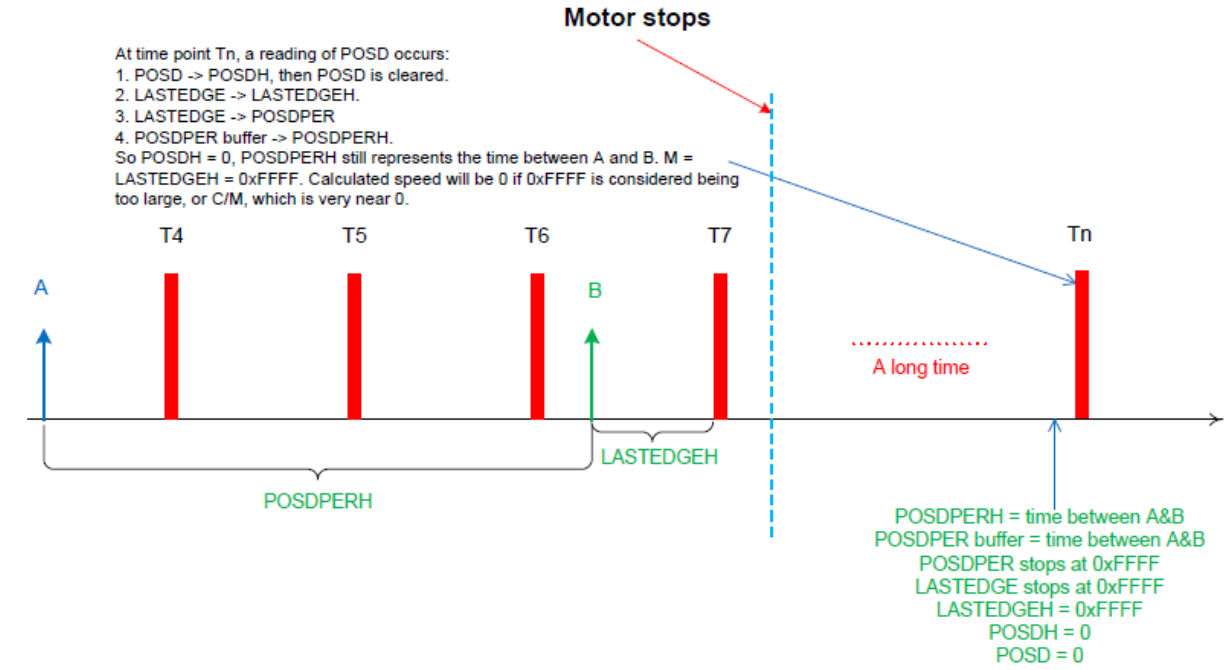


Figure 251. Speed measurement at time point Tn

Speed calculation when motor starts again

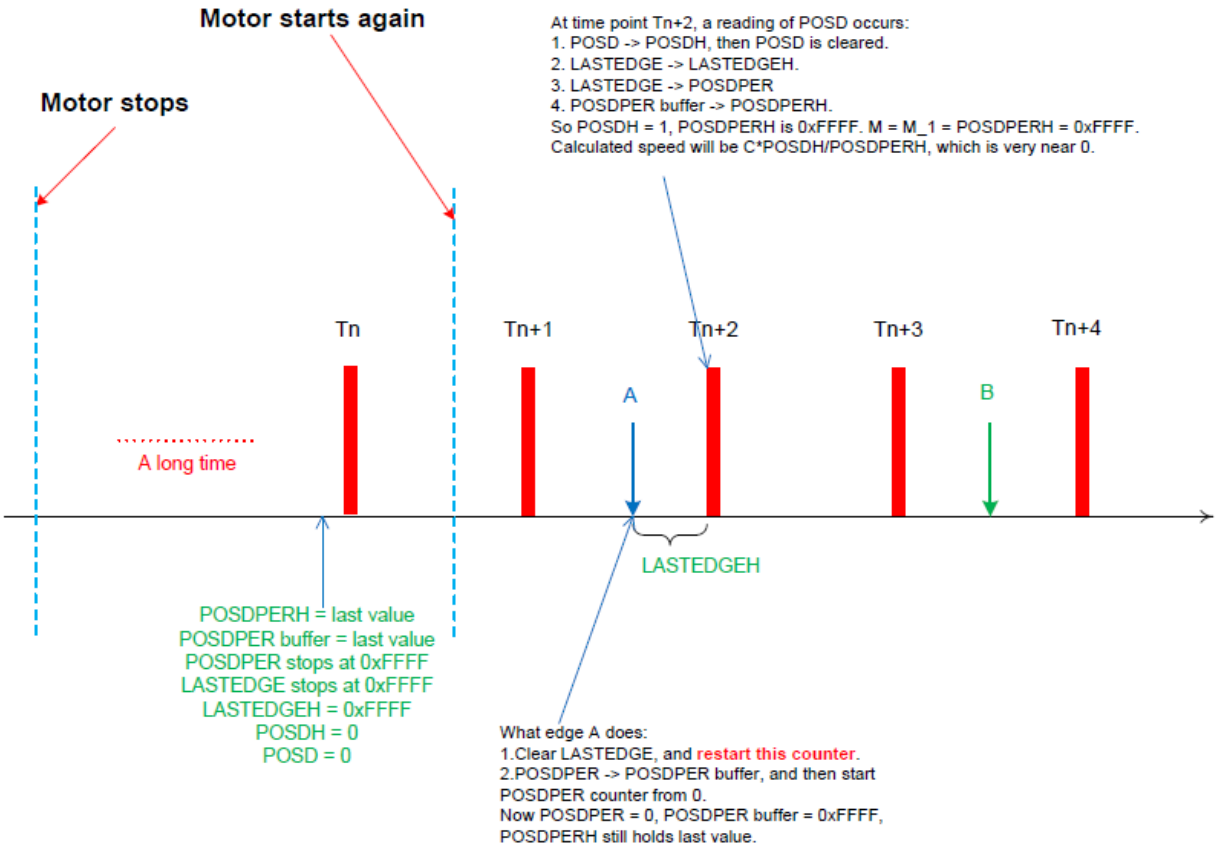


Figure 252. Speed measurement at time point Tn+2

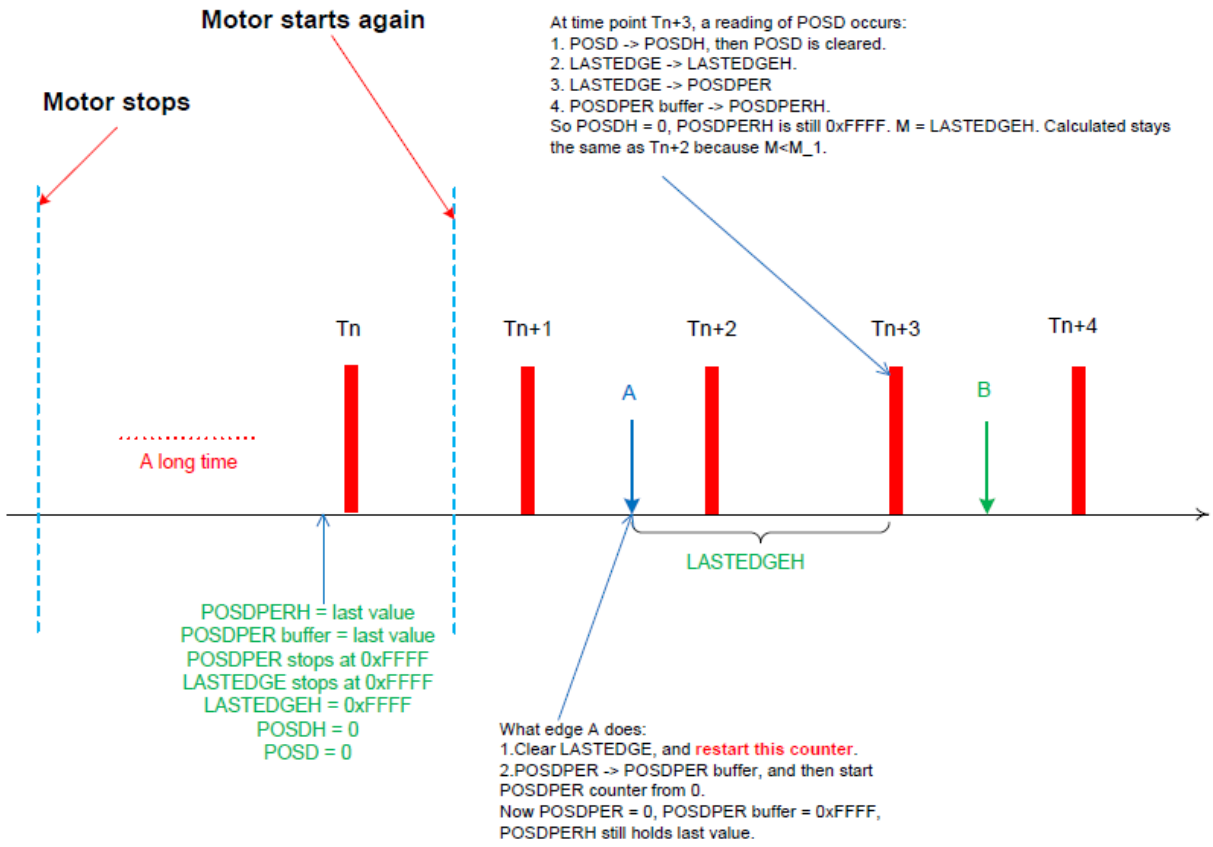


Figure 253. Speed measurement at time point Tn+3

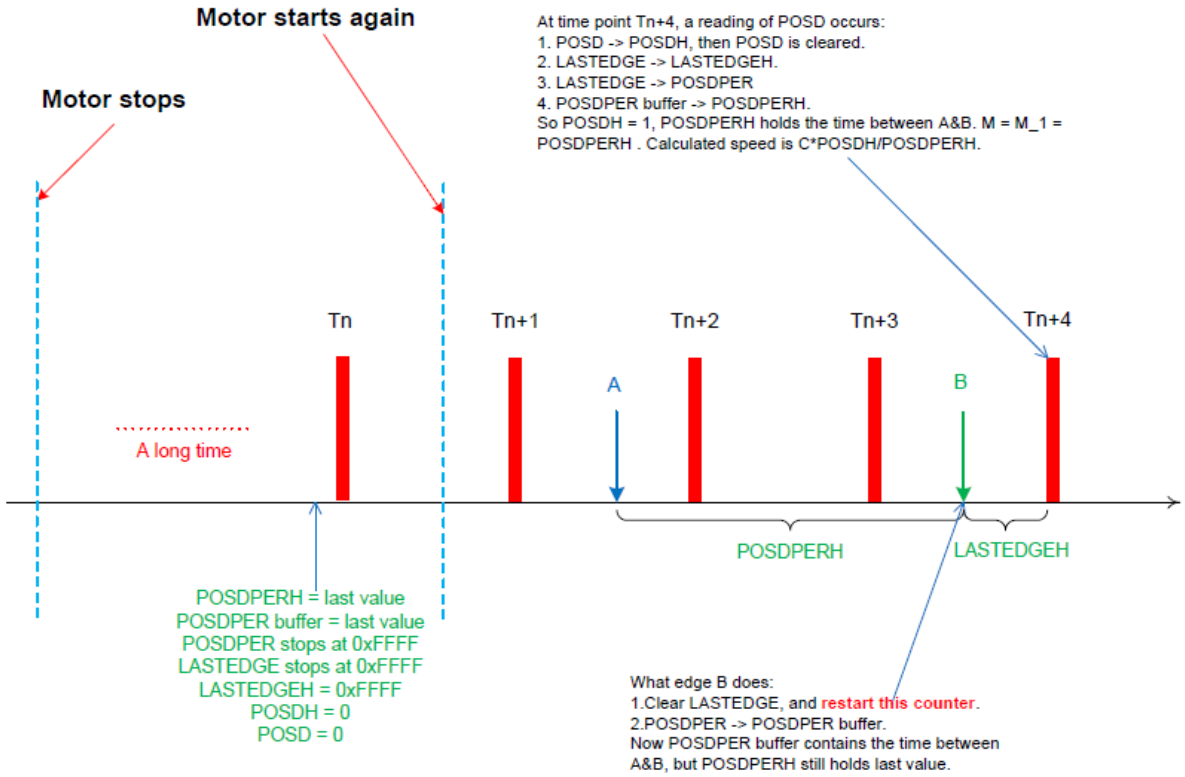


Figure 254. Speed measurement at time point Tn+4

45.3.6 Clocks

The IPBus clock is the only clock required by this module in normal operation.

45.3.7 Resets

There are no special requirements. This module is reset by any system reset.

45.3.8 Interrupts

The following table lists the module interrupts.

Table 382. Interrupt Summary

Core Interrupt	Interrupt Flag	Interrupt Enable	Description
ipi_int_home	CTRL[HIRQ]	CTRL[HIE]	HOME signal transition interrupt
ipi_int_index	CTRL[XIRQ]	CTRL[XIE]	INDEX signal transition interrupt or roll-over/ under interrupt
	CTRL2[ROIRQ]	CTRL2[ROIE]	
	CTRL2[RUIRQ]	CTRL2[RUIE]	

Table continues on the next page...

Table 382. Interrupt Summary (continued)

Core Interrupt	Interrupt Flag	Interrupt Enable	Description
ipi_int_wdog	CTRL[DIRQ]	CTRL[DIE]	Watchdog timeout interrupt
ipi_int_cmp	CTRL[CMPIRQ]	CTRL[CMPIE]	Compare match interrupt
ipi_int_sab	CTRL2[SABIRQ]	CTRL2[SABIE]	Simultaneous PHASEA and PHASEB change interrupt

45.4 Signal Descriptions

Four external signals are multiplexed to the four pins (PHASEA, PHASEB, INDEX, HOME) of a quad timer module. Two internal signals (TRIGGER, POSMATCH) can be connected to other SoC resources.

Table 383. Signals

Signal		Description
PHASEA	Phase A Input	The PHASEA input can be connected to one of the phases from a two-phase shaft quad encoder output. PHASEA and PHASEB are used by the quadrature decoder module to indicate a decoder increment has passed, and to calculate its direction. • When the quadrature decoder module is used as a single phase pulse accumulator, PHASEA can also be used as the single input. • The PHASEA input can be an input capture channel for one of the timer modules (in the SoC), which can be connected to using a crossbar switch. Direction for two-phase shaft quad encoder output: • PHASEA is the leading phase for a shaft rotating in the positive direction. • PHASEA is the trailing phase for a shaft rotating in the negative direction.
PHASEB	Phase B Input	The PHASEB input can be connected to the other phase from a two-phase shaft quad encoder output. • The PHASEB input can be an input capture channel for one of the timer modules (in the SoC), which can be connected to using a crossbar switch. Direction for two-phase shaft quad encoder output: • PHASEB is the trailing phase for a shaft rotating in the positive direction. • PHASEB is the leading phase for a shaft rotating in the negative direction.
INDEX	Index Input	• Normally connected to the index pulse output of a quad encoder, the INDEX pulse can optionally reset the position counter and the pulse accumulator of the quadrature decoder module. The INDEX pulse also causes a change of state on the revolution counter. The direction of this state change (increment or decrement) is calculated from the PHASEA and PHASEB inputs. • The INDEX input can be an input capture channel for one of the timer modules (in the SoC), which can be connected to using a crossbar switch.
HOME	Home Switch Input	The HOME input can be used by the quadrature decoder and the timer module.

Table continues on the next page...

Table 383. Signals (continued)

Signal		Description
		• The HOME input can be used to trigger the initialization of the position counters (UPOS and LPOS). Often the HOME signal is connected to a sensor on the motor or machine, sending notification that it has reached a defined home position. • The HOME signal can be connected to the timer module (in the SoC) using a crossbar switch.
TRIGGER	Trigger Input	The TRIGGER input can be used • to clear the position counters (UPOS and LPOS) • to take a snapshot of the POS, REV, and POSD registers • to indicate an elapsed time period, by connecting the TRIGGER input signal to a periodic pulse generator or timer
POSMATCH	Position Match Output	• To record the time at which the position of the shaft matches a user-defined compare value (COMP), the POSMATCH output can be used to trigger a timer channel. • Alternatively, to record the time at which the position information was read, (when the position counters (UPOS and LPOS), revolution counter (REV), or position difference counter (POSD) registers are read) the POSMATCH output can be used to trigger a timer channel.

45.5 Memory Map and Registers

This section describes the module memory map and registers.

45.5.1 ENC register descriptions

The address of a register is the sum of a base address and an address offset.

- Base address is defined at the MCU level
- Address offset is defined at the module level

For the base address, see the specific chip documentation. All memory locations base and offsets are given in hex.

45.5.1.1 ENC memory map

QEI0 base address: 400C_4000h

QEI1 base address: 400C_6000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Control Register (CTRL)	16	See section	0000
2	Input Filter Register (FILT)	16	See section	0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
4	Watchdog Timeout Register (WTR)	16	RW	0000
6	Position Difference Counter Register (POSD)	16	RW	0000
8	Position Difference Hold Register (POSDH)	16	RO	0000
A	Revolution Counter Register (REV)	16	RW	0000
C	Revolution Hold Register (REVH)	16	RO	0000
E	Upper Position Counter Register (UPOS)	16	RW	0000
10	Lower Position Counter Register (LPOS)	16	RW	0000
12	Upper Position Hold Register (UPOSH)	16	RO	0000
14	Lower Position Hold Register (LPOSH)	16	RO	0000
16	Upper Initialization Register (UINIT)	16	RW	0000
18	Lower Initialization Register (LINIT)	16	RW	0000
1A	Input Monitor Register (IMR)	16	See section	0000
1C	Test Register (TST)	16	RW	0000
1E	Control 2 Register (CTRL2)	16	See section	0000
20	Upper Modulus Register (UMOD)	16	RW	0000
22	Lower Modulus Register (LMOD)	16	RW	0000
24	Upper Position Compare Register (UCOMP)	16	RW	FFFF
26	Lower Position Compare Register (LCOMP)	16	RW	FFFF
28	Last Edge Time Register (LASTEDGE)	16	RO	FFFF
2A	Last Edge Time Hold Register (LASTEDGEH)	16	RO	FFFF
2C	Position Difference Period Counter Register (POSDPER)	16	RO	FFFF
2E	Position Difference Period Buffer Register (POSDPERBFR)	16	RO	FFFF
30	Position Difference Period Hold Register (POSDPERH)	16	RO	FFFF
32	Control 3 Register (CTRL3)	16	See section	0000

45.5.1.1.1 Control Register (CTRL)

ENC controller register covers basic function on HOME and INDEX signals, and part of basic functions.

Offset

Register	Offset
CTRL	0h

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HIRQ	HIE	HIP	HNE	0	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE	CMPIRQ	CMPIE
W	W1C				SWIP			W1C				W1C			W1C	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
15 HIRQ	HOME Signal Transition Interrupt Request HOME Signal Transition Interrupt Request is set when a transition on the HOME signal occurs, according to the Use Negative Edge of HOME Input bit (CTRL[HNE]). If HOME Signal Transition Interrupt Request bit is set and HOME Interrupt Enable bit (CTRL[HIE]) is set, then a HOME interrupt occurs. • HOME Signal Transition Interrupt Request bit will remain set until it is cleared by software • Write a one to this bit (HIRQ) to clear it 0 - No transition on the HOME signal has occurred 1 - A transition on the HOME signal has occurred
14 HIE	HOME Interrupt Enable Enables/disables HOME signal interrupts. 0 - Disabled 1 - Enabled
13 HIP	Enable HOME to Initialize Position Counters UPOS and LPOS Enables the position counter to be initialized by the HOME signal. 0 - No action 1 - HOME signal initializes the position counter
12 HNE	Use Negative Edge of HOME Input Selects using the positive/negative edge of the HOME input. 0 - Use positive-going edge-to-trigger initialization of position counters UPOS and LPOS 1 - Use negative-going edge-to-trigger initialization of position counters UPOS and LPOS
11	Software-Triggered Initialization of Position Counters UPOS and LPOS

Table continues on the next page...

Table continued from the previous page...

Field	Description
SWIP	Writing 1 to Software-Triggered Initialization of Position Counters bit will transfer the UINIT and LINIT contents to UPOS and LPOS (thereby initializing those counters). • Software-Triggered Initialization of Position Counters bit is always read as 0. 0 - No action 1 - Initialize position counter (using upper and lower initialization registers, UINIT and LINIT)
10 REV	Enable Reverse Direction Counting Selects the direction of the count (how the quadrature signal is interpreted). 0 - Count normally 1 - Count in the reverse direction
9 PH1	Enable Signal Phase Count Mode Bypass/use the quadrature decoder logic. 0 - Use the standard quadrature decoder, where PHASEA and PHASEB represent a two-phase quadrature signal. 1 - Bypass the quadrature decoder. A positive transition of the PHASEA input generates a count signal. The PHASEB input and the REV bit control the counter direction. If the value of CTRL[REV] and PHASEB is same, then count is up. If the value of CTRL[REV] and PHASEB is different, then count is down.
8 XIRQ	INDEX Pulse Interrupt Request INDEX Pulse Interrupt Request is set when a transition on the INDEX signal occurs, according to the Use Negative Edge of INDEX Pulse bit (CTRL[XNE]). If INDEX Pulse Interrupt Request bit is set and INDEX Pulse Interrupt Enable bit (CTRL[XIE]) is set, then an INDEX interrupt occurs. • INDEX Pulse Interrupt Request will remain set until cleared by software • Write a one to this bit (XIRQ) to clear it 0 - INDEX pulse has not occurred 1 - INDEX pulse has occurred
7 XIE	INDEX Pulse Interrupt Enable Enables/disables INDEX pulse interrupts. 0 - Disabled 1 - Enabled
6 XIP	INDEX Triggered Initialization of Position Counters UPOS and LPOS Enables/disables the position counter to be initialized by the INDEX pulse. 0 - INDEX pulse does not initialize the position counter 1 - INDEX pulse initializes the position counter
5	Use Negative Edge of INDEX Pulse Selects the positive/negative edge of the INDEX pulse used to initialize the position counter.

Table continues on the next page...

Table continued from the previous page...

Field	Description
XNE	0 - Use positive edge of INDEX pulse 1 - Use negative edge of INDEX pulse
4 DIRQ	Watchdog Timeout Interrupt Request Watchdog Timeout Interrupt Request is set when a watchdog timeout interrupt occurs. • Watchdog Timeout Interrupt Request will remain set until cleared by software • Write a one to this bit (DIRQ) to clear it • Watchdog Timeout Interrupt Request bit is also cleared when Watchdog Enable (CTRL[WDE]) is disabled (=0) 0 - No Watchdog timeout interrupt has occurred 1 - Watchdog timeout interrupt has occurred
3 DIE	Watchdog Timeout Interrupt Enable Enables/disables watchdog timeout interrupts. 0 - Disabled 1 - Enabled
2 WDE	Watchdog Enable Enables/disables the watchdog timer that is monitoring the PHASEA and PHASEB inputs for motor movement. 0 - Disabled 1 - Enabled
1 CMPIRQ	Compare Interrupt Request Compare Interrupt Request is set when the counter matches the COMP value. • Compare Interrupt Request remains set until cleared by software • Write 1 to this bit (CMPIRQ) to clear it 0 - No match has occurred (the counter does not match the COMP value) 1 - COMP match has occurred (the counter matches the COMP value)
0 CMPIE	Compare Interrupt Enable Enables/disables compare interrupt. 0 - Disabled 1 - Enabled

45.5.1.1.2 Input Filter Register (FILT)

The Input Filter Register sets the values of the input filter sample period (FILT_PER),the filter prescaler (FILT_PRSC) and the input filter sample count (FILT_CNT).

- >The Input Filter prescaler value (FILT_PRSC) should be set to prescale the IPBus clock. (frequency of FILT clock) = (frequency of IPBus clock) / 2^FILT_PRSC
- The Input Filter Sample Period (FILT_PER) should be set so that the sampling period is larger than the period of the expected noise. Doing this means that a noise spike will only corrupt one sample.
- The Input Filter Sample Count (FILT_CNT) should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the power of FILT_CNT+3.

The values of the Input Filter Sample Period (FILT_PER) and the Input Filter Sample Count (FILT_CNT) must also be traded off against the desire for minimal latency in recognizing valid input transitions. Turning on the input filter (setting the Input Filter Sample Period (FILT_PER) to a non-zero value) introduces a latency of ((FILT_CNT+3)*FILT_PER) FILT clock cycles +2 IPBus clock periods.

The filter latency can be measured:

- Drive the quadrature decoder inputs (PHASEA, PHASEB, INDEX, HOME)
- Monitor the filtered output in the Input Monitor Register (IMR)
- Determine how many IPBus clock cycles that it takes before the output shows up, by using the following equations, where f is FILT_PER and s is FILT_CNT.
  - DELAY = f * (s+3)(FILT clock cycles) +1 (IPBus clock cycles)(to read the filtered output)
  - DELAY = f * (s+3)(FILT clock cycles) +2 (IPBus clock cycles) (to monitor the output in the IMR)

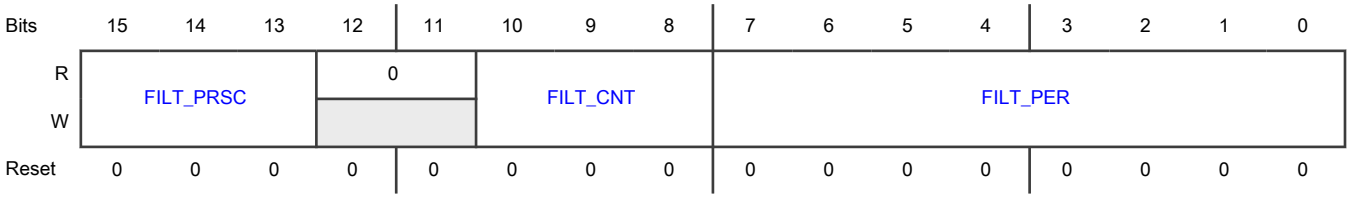
One more additional IPBus clock cycle is needed to read the filtered output, and 2 more IPBus clock cycles are needed to monitor the filtered output in the IMR. The sample rate is set when it reaches the number f. The following examples use the preceding equations:

- Example: when f = 0, the filter is bypassed: DELAY = 1 or 2 clock cycles.
- Example: when f = 5 and s = 2: DELAY = 5 * (2+3)FILT clock cycles + (1 or 2)IPBus clock cycles

Offset

Register	Offset
FILT	2h

Diagram



Fields

Field	Description
15-13 FILT_PRSC	prescaler divide IPbus clock to FILT clk

Table continues on the next page...

Table continued from the previous page...

Field	Description
	The Prescaler field is used to prescale the IPbus clock to FILT clock. The IPbus clock is prescaled by a value of 2^FILT_PRSC which means that the prescaler logic can divide the clock by a minimum of 1 and a maximum of 128. So FILTER can filt lower freq noise.
12-11 —	Reserved
10-8 FILT_CNT	Input Filter Sample Count The Input Filter Sample Count represents the number of consecutive samples that must agree, before the input filter accepts an input transition. • A value of 0x0 represents 3 samples • A value of 0x7 represents 10 samples The value of the Input Filter Sample Count (FILT_CNT) affects the input latency.
7-0 FILT_PER	Input Filter Sample Period The Input Filter Sample Period represents the sampling period (in IPBus clock cycles) of the decoder input signals. Each input is sampled multiple times at the rate specified by the Input Filter Sample Period. • If FILT_PER is 0x00 (default), then the input filter is bypassed. Bypassing the digital filter enables the position/position difference counters to operate with count rates up to the IPBus frequency. • The value of the Input Filter Sample Period (FILT_PER) affects the input latency. • When changing the Input Filter Sample Period (FILT_PER) from a non-zero value to another non-zero value, write a value of 0 first, to clear the filter.

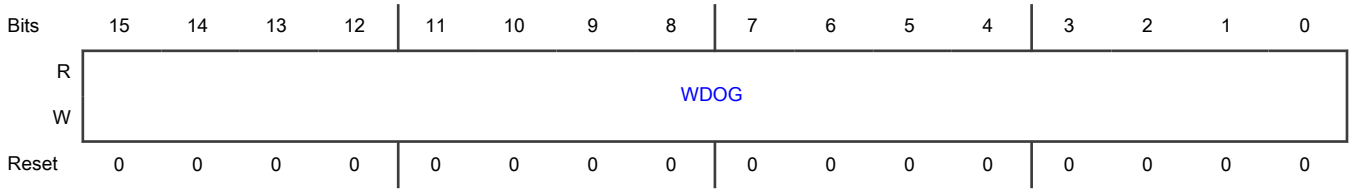
45.5.1.1.3 Watchdog Timeout Register (WTR)

The Watchdog Timeout Register stores the timeout count for the quadrature decoder module watchdog timer. This quadrature decoder module watchdog timer is separate from any other watchdog timer(s) that may also be in the device.

Offset

Register	Offset
WTR	4h

Diagram



Fields

Field	Description
15-0 WDOG	WDOG WDOG[15:0] is a binary representation of the number of clock cycles, plus one clock cycle that the watchdog timer counts before timing out (and optionally generating an interrupt).

45.5.1.1.4 Position Difference Counter Register (POSD)

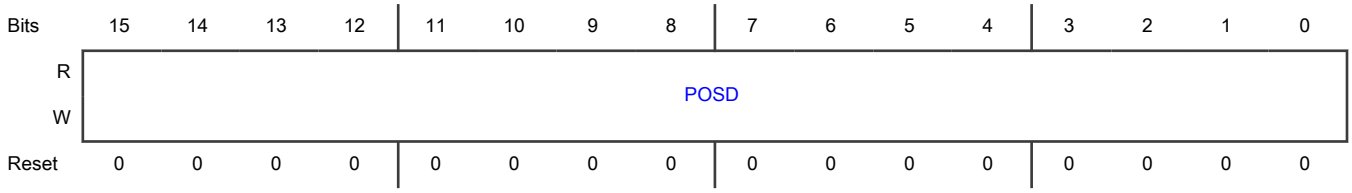
The Position Difference Counter Register contains the position change in value occurring between each read of the Position Register. The value of the Position Difference Counter Register can be used to calculate velocity.

- The 16-bit Position Difference Counter computes up or down on every count pulse.
- This Position Difference Counter acts as a differentiator, whose count value is proportional to the change in position since the last time that the Position Counter was read.
- When the Position Difference Counter (POSD) is read, its contents are copied into the Position Difference Hold Register (POSDH) and the Position Difference Counter is cleared.

Offset

Register	Offset
POSD	6h

Diagram



Fields

Field	Description
15-0 POSD	POSD The position change in value between each read of the Position Register.

45.5.1.1.5 Position Difference Hold Register (POSDH)

The Position Difference Hold Register register contains a snapshot of the value of the Position Difference Counter Register (POSD). The value of the Position Difference Hold Register (POSDH) can be used to calculate velocity.

Offset

Register	Offset
POSDH	8h

Diagram

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

Fields

Field	Description
15-0 POSDH	POSDH The value of the POSD register

45.5.1.1.6 Revolution Counter Register (REV)

Contains the current value of the Revolution Counter.

Offset

Register	Offset
REV	Ah

Diagram

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

Fields

Field	Description
15-0	REV

Table continues on the next page...

Field	Description
REV	The current value of the Revolution Counter

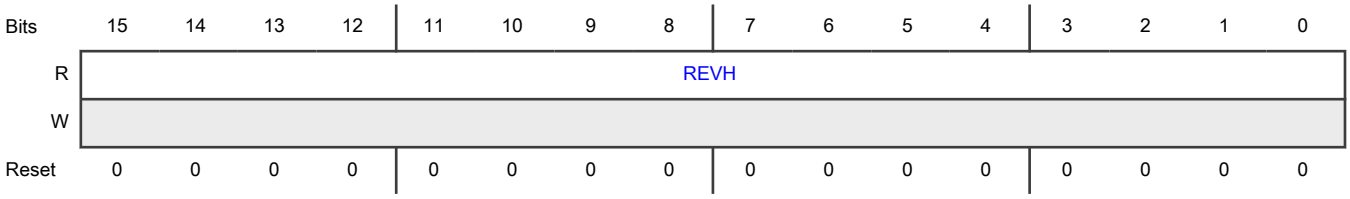
45.5.1.1.7 Revolution Hold Register (REVH)

Contains a snapshot of the value of the Revolution Counter Register (REV).

Offset

Register	Offset
REVH	Ch

Diagram



Fields

Field	Description
15-0 REVH	REVH The value of the Revolution Counter Register (REV)

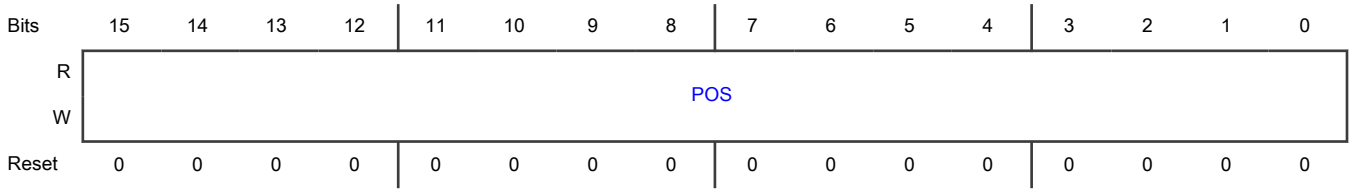
45.5.1.1.8 Upper Position Counter Register (UPOS)

Contains the upper (most significant) half of the Position Counter. This is the binary count from the Position Counter.

Offset

Register	Offset
UPOS	Eh

Diagram



Fields

Field	Description
15-0 POS	POS The upper (most significant) half of the Position Counter

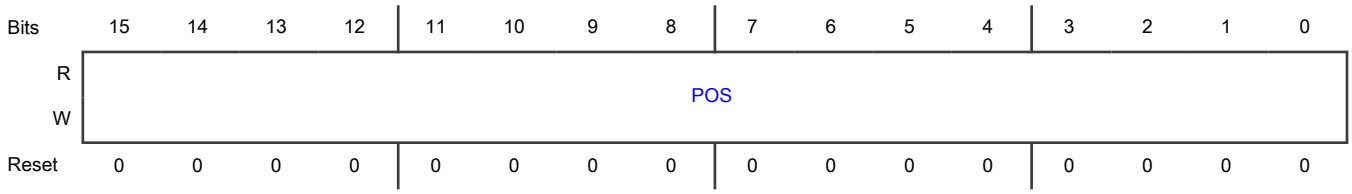
45.5.1.1.9 Lower Position Counter Register (LPOS)

Contains the lower (least significant) half of the Position Counter. This is the binary count from the Position Counter.

Offset

Register	Offset
LPOS	10h

Diagram



Fields

Field	Description
15-0 POS	POS The lower (least significant) half of the Position Counter

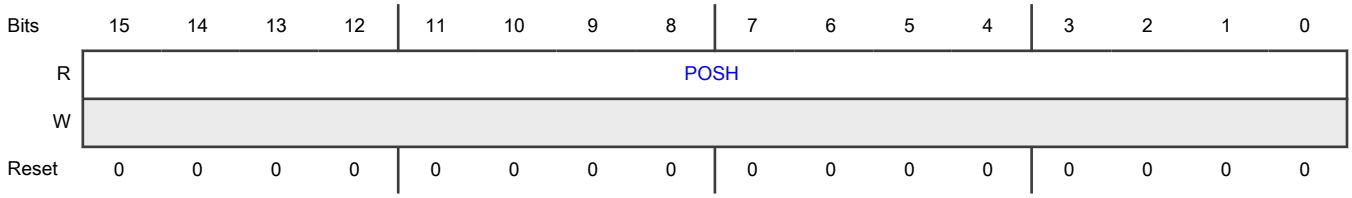
45.5.1.1.10 Upper Position Hold Register (UPOSH)

Contains a snapshot of the Upper Position Counter Register (UPOS register).

Offset

Register	Offset
UPOSH	12h

Diagram



Fields

Field	Description
15-0 POSH	POSH The value of the Upper Position Counter Register (UPOS)

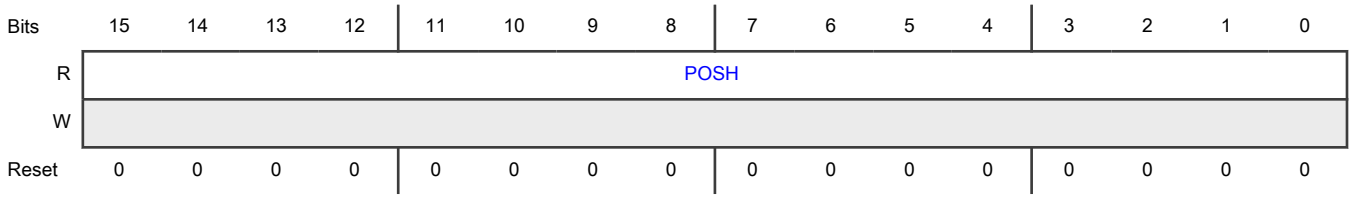
45.5.1.1.11 Lower Position Hold Register (LPOSH)

Contains a snapshot of the Lower Position Counter Register (LPOS).

Offset

Register	Offset
LPOSH	14h

Diagram



Fields

Field	Description
15-0 POSH	POSH The value of the Lower Position Counter Register (LPOS)

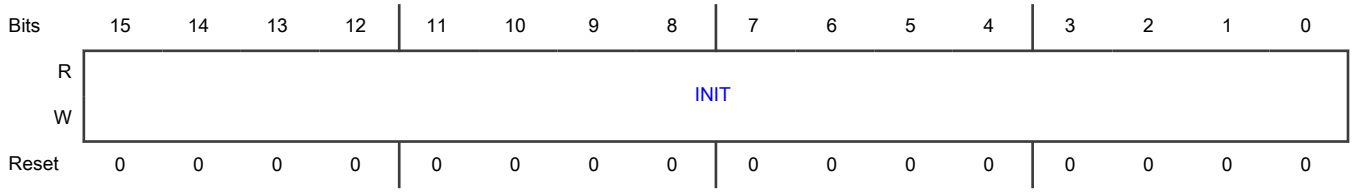
45.5.1.1.12 Upper Initialization Register (UINIT)

Contains the value to be used to initialize the upper half of the position counter (UPOS).

Offset

Register	Offset
UINIT	16h

Diagram



Fields

Field	Description
15-0 INIT	INIT The value to be used to initialize the upper half of the position counter (UPOS)

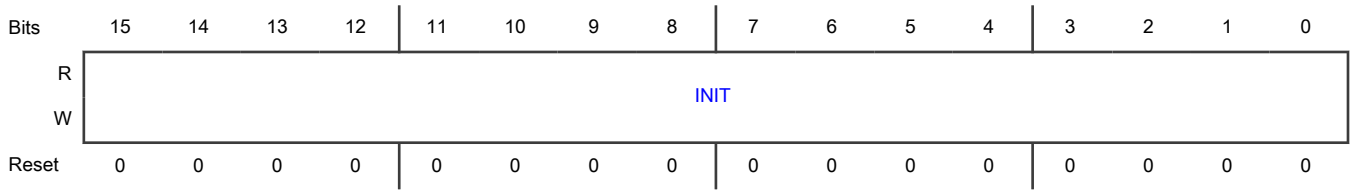
45.5.1.1.13 Lower Initialization Register (LINIT)

Contains the value to be used to initialize the lower half of the position counter (LPOS).

Offset

Register	Offset
LINIT	18h

Diagram



Fields

Field	Description
15-0 INIT	INIT The value to be used to initialize the lower half of the position counter (LPOS)

45.5.1.1.14 Input Monitor Register (IMR)

The Input Monitor Register contains the values of the raw and filtered PHASEA, PHASEB, INDEX, and HOME input signals. The reset value depends on the values of the raw and filtered values of PHASEA, PHASEB, INDEX, and HOME input signals:

- If these input pins are connected to a pull-up, then bits 0–7 of the IMR are all ones.
- If these input pins are connected to a pull-down device, then bits 0–7 of the Input Monitor Register (IMR) are all zeros.

Offset

Register	Offset
IMR	1Ah

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								FPHA	FPHB	FIND	FHOM	PHA	PHB	INDEX	HOME
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
15-8 —	Reserved
7 FPHA	FPHA Filtered version of PHASEA input
6 FPHB	FPHB Filtered version of PHASEB input
5 FIND	FIND Filtered version of INDEX input
4 FHOM	FHOM Filtered version of HOME input
3 PHA	PHA Raw PHASEA input
2 PHB	PHB Raw PHASEB input
1 INDEX	INDEX Raw INDEX input
0 HOME	HOME The raw HOME input

45.5.1.1.15 Test Register (TST)

The Test Register controls and sets the frequency of a quadrature signal generator; it provides a quadrature test signal to the inputs of the quadrature decoder module.

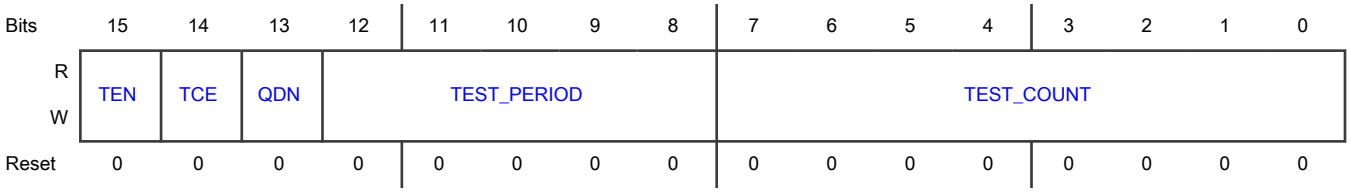
- The TEST_COUNT value is counted down to zero when the test module is enabled (TEN = 1) and the count is enabled (TCE = 1).
- Each count value of one represents a single quadrature cycle interpreted as a count of one by the position counter (UPOS and LPOS) if it is enabled (TEN = 1, TCE = 1).
- Repeated writing of new values to TEST_COUNT can cause an extra phase transition and therefore an extra count by the Position Counter.
- The period field determines the length (in IPBus clock cycles) of each quadrature cycle phase.

The Test Register is a factory test feature; however, it can also be useful in customer software development and testing.

Offset

Register	Offset
TST	1Ch

Diagram



Fields

Field	Description
15 TEN	Test Mode Enable Connects the test module to inputs of the quadrature decoder module. 0 - Disabled 1 - Enabled
14 TCE	Test Counter Enable Connects the test counter to inputs of the quadrature decoder module. 0 - Disabled 1 - Enabled
13 QDN	Quadrature Decoder Negative Signal Selects whether a negative or positive Quadrature Decoder signal is generated. 0 - Generates a positive quadrature decoder signal 1 - Generates a negative quadrature decoder signal
12-8 TEST_PERIOD	TEST_PERIOD Period of quadrature phase in IPBus clock cycles

Table continues on the next page...

Table continued from the previous page...

Field	Description
7-0 TEST_COUNT	TEST_COUNT The number of quadrature advances to generate

45.5.1.1.16 Control 2 Register (CTRL2)

ENC Controller register 2 covers the status of interrupts, interrupt enablem and part of basic functions

Offset

Register	Offset
CTRL2	1Eh

Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				SABIRQ	SABIE	OUTCTL	REVMOD	ROIRQ	ROIE	RUIRQ	RUIE	DIR	MOD	UPDPOS	UPDHLD
W					W1C				W1C		W1C					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
15-12 —	Reserved
11 SABIRQ	Simultaneous PHASEA and PHASEB Change Interrupt Request Simultaneous PHASEA and PHASEB Change Interrupt Request indicates that the PHASEA and PHASEB inputs changed simultaneously (within a single clock period). This event typically indicates an error condition, because quadrature coding requires only one of these inputs to change at a time. • Simultaneous PHASEA and PHASEB Change Interrupt Request bit remains set until it is cleared by software or a reset • Write 1 to this bit (SABIRQ) to clear it 0 - No simultaneous change of PHASEA and PHASEB has occurred 1 - A simultaneous change of PHASEA and PHASEB has occurred
10 SABIE	Simultaneous PHASEA and PHASEB Change Interrupt Enable Simultaneous PHASEA and PHASEB Change Interrupt Enable bit enables simultaneous PHASEA and PHASEB change interrupts, based on the Simultaneous PHASEA and PHASEB Change Interrupt Request (SABIRQ) bit being set.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - Disabled 1 - Enabled
9 OUTCTL	Output Control Output Control controls the pulsing of the POSMATCH output signal. This can control when a timer channel captures a timestamp. 0 - POSMATCH pulses when a match occurs between the position counters (POS) and the corresponding compare value (COMP) 1 - POSMATCH pulses when the UPOS, LPOS, REV, or POSD registers are read
8 REVMOD	Revolution Counter Modulus Enable Revolution Counter Modulus Enable selects how the revolution counter (REV) is incremented/ decremented. By default, the revolution counter is controlled based on the count direction and the INDEX pulse. As an option, the revolution counter can be controlled using the roll-over/under detection during modulo counting. 0 - Use INDEX pulse to increment/decrement revolution counter (REV) 1 - Use modulus counting roll-over/under to increment/decrement revolution counter (REV)
7 ROIRQ	Roll-over Interrupt Request Roll-over Interrupt Request bit is set (=1) when the position counter (POS) rolls over from the MOD value to the INIT value, or from 0xFFFF_FFFF to 0x0000_0000. • Roll-over Interrupt Request will remain set until cleared by software • Write a one to this bit (ROIRQ) to clear it 0 - No roll-over has occurred 1 - Roll-over has occurred
6 ROIE	Roll-over Interrupt Enable Roll-over Interrupt Enable read/write bit enables roll-over interrupts, based on Roll-under Interrupt Request (CTRL2[ROIRQ]) being set. This roll-over interrupt is combined with the index interrupt signal. 0 - Disabled 1 - Enabled
5 RUIRQ	Roll-under Interrupt Request Roll-under Interrupt Request bit is set (=1) when the position counter (POS) rolls under from the INIT value to the MOD value, or from 0x0000_0000 to 0xFFFF_FFFF. • Roll-under Interrupt Request will remain set until cleared by software • Write a one to this bit (RUIRQ) to clear it 0 - No roll-under has occurred 1 - Roll-under has occurred

Table continues on the next page...

Table continued from the previous page...

Field	Description
4 RUIE	Roll-under Interrupt Enable Roll-under Interrupt Enable read/write bit enables roll-under interrupts, based on the Roll-over Interrupt Request bit (CTRL2[RUIRQ]) being set. This roll-under interrupt is combined with the index interrupt signal. 0 - Disabled 1 - Enabled
3 DIR	Count Direction Flag The Count Direction Flag indicates the direction of the last count. 0 - Last count was in the down direction 1 - Last count was in the up direction
2 MOD	Enable Modulo Counting • When Enable Modulo Counting is set (=1), it allows the position counters (UPOS and LPOS) to count in a modulo fashion, using MOD and INIT as the upper and lower bounds of the counting range. During modulo counting: — When a "count up" is indicated and the position counter is equal to MOD, then the postion counter will be reloaded with the value of INIT. — When a "count down" is indicated and the position counter is equal to INIT, then the position counter will be reloaded with the value of MOD. • When Enable Modulo Counting is clear (=0), then the values of MOD and INIT are ignored, and the position counter wraps to zero when counting up from 0xFFFF_FFFF, or wraps to 0xFFFF_FFFF when counting down from 0. 0 - Disable modulo counting 1 - Enable modulo counting
1 UPDPOS	Update Position Registers • When Update Position Registers is set (=1), it allows the TRIGGER input to clear the POSD, REV, UPOS and LPOS registers. • When Update Position Registers is clear (=0), the POSD, REV, UPOS and LPOS registers ignore the TRIGGER input. • Do not set this bit when using the LASTEDGE and POSDPER registers to measure speed. 0 - No action for POSD, REV, UPOS and LPOS registers on rising edge of TRIGGER 1 - Clear POSD, REV, UPOS and LPOS registers on rising edge of TRIGGER
0 UPDHLD	Update Hold Registers • When Update Hold Registers is set (=1), it allows the TRIGGER input to cause an update of the POSDH, REVH, UPOSH, and LPOSH registers • When Update Hold Registers is clear (=0), the hold registers (POSDH, REVH, UPOSH, LPOSH) are not updated by the TRIGGER input

Table continues on the next page...

Table continued from the previous page...

Field	Description
	• Do not set this bit when using the LASTEDGE and POSDPER registers to measure speed. Updating the Position Difference Hold Register (POSDH) will also cause the Position Difference Counter Register (POSD) to be cleared. 0 - Disable updates of hold registers on the rising edge of TRIGGER input signal 1 - Enable updates of hold registers on the rising edge of TRIGGER input signal

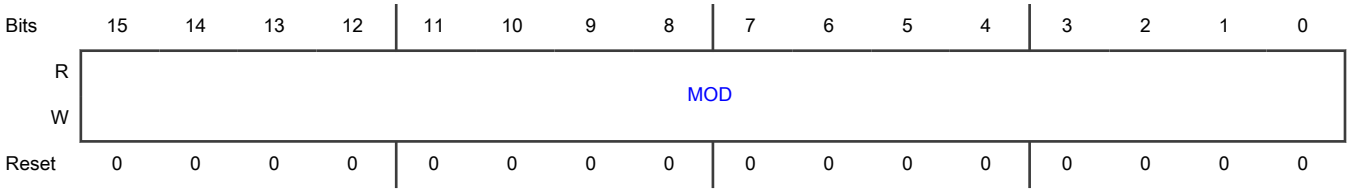
45.5.1.1.17 Upper Modulus Register (UMOD)

The Upper Modulus Register contains the upper (most significant) half of the Modulus register. MOD acts as the upper bound during modulo counting, and as the upper reload value when rolling over from the lower bound.

Offset

Register	Offset
UMOD	20h

Diagram



Fields

Field	Description
15-0 MOD	MOD Upper (most significant) half of the Modulus register

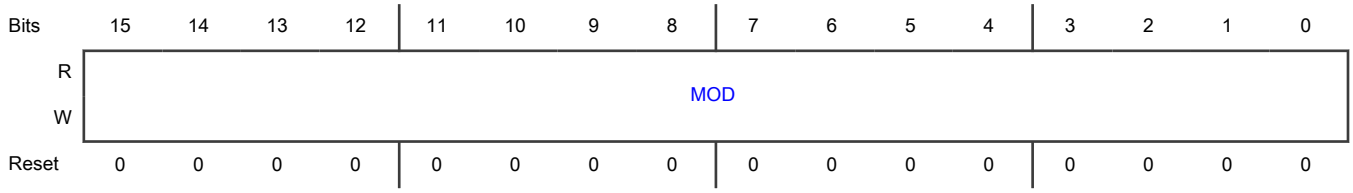
45.5.1.1.18 Lower Modulus Register (LMOD)

The Lower Modulus Register contains the lower (least significant) half of the Modulus register. MOD acts as the upper bound during modulo counting, and as the upper reload value when rolling over from the lower bound.

Offset

Register	Offset
LMOD	22h

Diagram



Fields

Field	Description
15-0 MOD	MOD Lower (least significant) half of the Modulus register

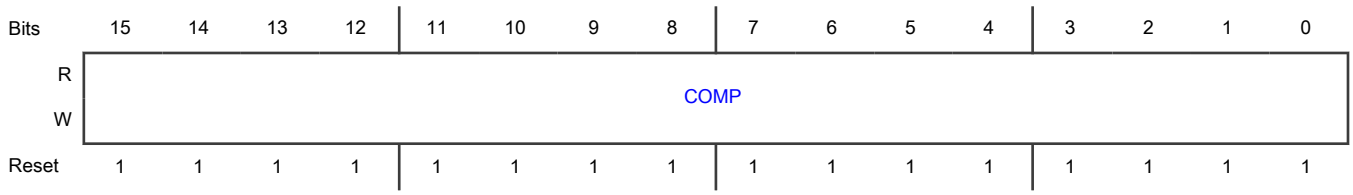
45.5.1.1.19 Upper Position Compare Register (UCOMP)

The Upper Position Compare Register contains the upper (most significant) half of the Position Compare register. When the value of the Position counter (POS) matches the value of the Position Compare register (COMP), the Compare Interrupt Request flag (CTRL[CMPIRQ]) is set and the POSMATCH output is asserted.

Offset

Register	Offset
UCOMP	24h

Diagram



Fields

Field	Description
15-0 COMP	COMP Upper (most significant) half of the Position Compare register

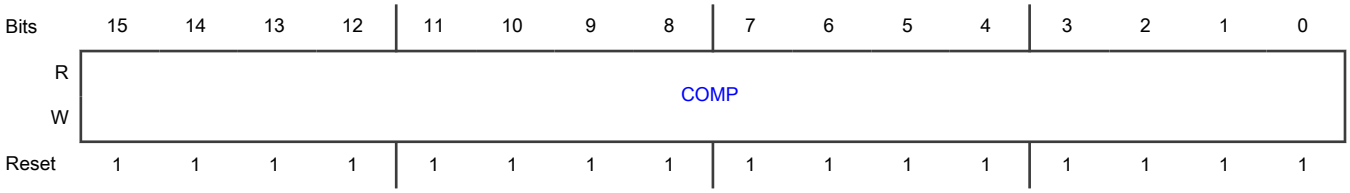
45.5.1.1.20 Lower Position Compare Register (LCOMP)

The Lower Position Compare Register contains the lower (least significant) half of the Position Compare register. When the value of the Position counter (POS) matches the value of the Position Compare register (COMP), the Compare Interrupt Request flag (CTRL[CMPIRQ]) is set and the POSMATCH output is asserted.

Offset

Register	Offset
LCOMP	26h

Diagram



Fields

Field	Description
15-0 COMP	COMP Lower (least significant) half of the Position Compare register

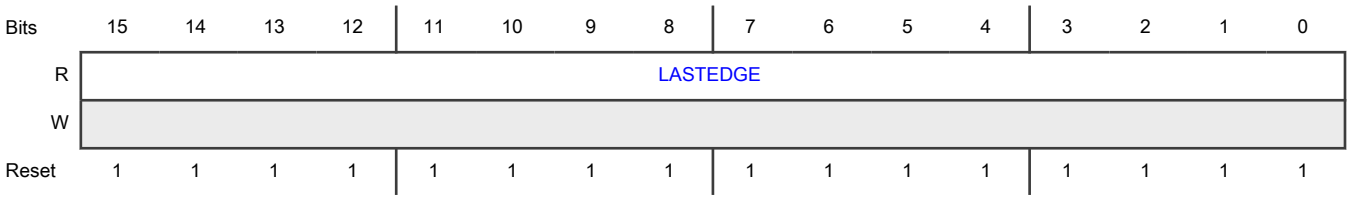
45.5.1.1.21 Last Edge Time Register (LASTEDGE)

This register represents the time since the last edge occurred on PHASEA or PHASEB.

Offset

Register	Offset
LASTEDGE	28h

Diagram



Fields

Field	Description
15-0 LASTEDGE	Last Edge Time Counter The Last Edge Time counter represents the time since the last edge occurred on PHASEA or PHASEB. LASTEDGE counts up using the peripheral clock that is prescaled by CTRL3[PRSC]. Any edge on PHASEA or PHASEB will reset this register to 0 and will start counting. If the LASTEDGE count reaches 0xffff, the counting will stop in order to prevent an overflow. Counting will continue when an edge occurs on PHASEA or PHASEB.

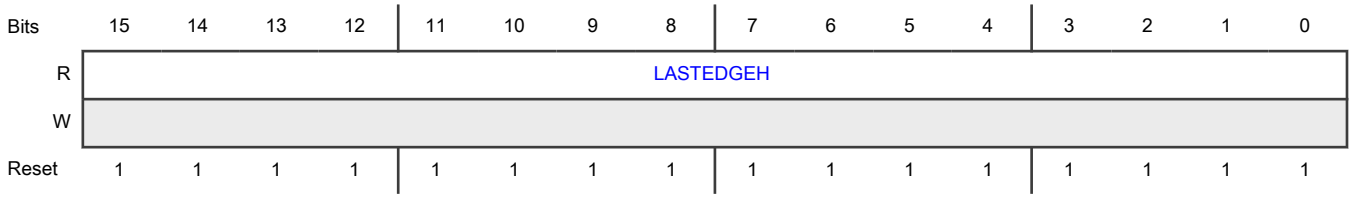
45.5.1.1.22 Last Edge Time Hold Register (LASTEDGEH)

This register holds register for the LASTEDGE value.

Offset

Register	Offset
LASTEDGEH	2Ah

Diagram



Fields

Field	Description
15-0 LASTEDGEH	Last Edge Time Hold The Last Edge Time Hold register is a hold register for the LASTEDGE value. LASTEDGEH is updated with the value of LASTEDGE when the POSD register is read.

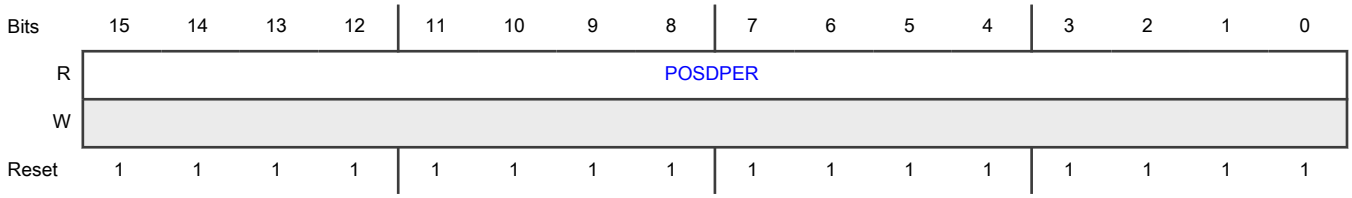
45.5.1.1.23 Position Difference Period Counter Register (POSDPER)

This register contains the information of reading position difference period counter

Offset

Register	Offset
POSDPER	2Ch

Diagram



Fields

Field	Description
15-0	Position difference period

Table continues on the next page...

Field	Description
POSDPER	The Position Difference Period counter counts up using the peripheral clock that is prescaled by CTRL3[PRSC]. Reading the POSD register will also cause the value of the LASTEDGE register to be loaded into POSDPER. If the POSDPER count reaches 0xffff, the counting will stop in order to prevent an overflow. Counting will continue when an edge occurs on PHASEA or PHASEB.

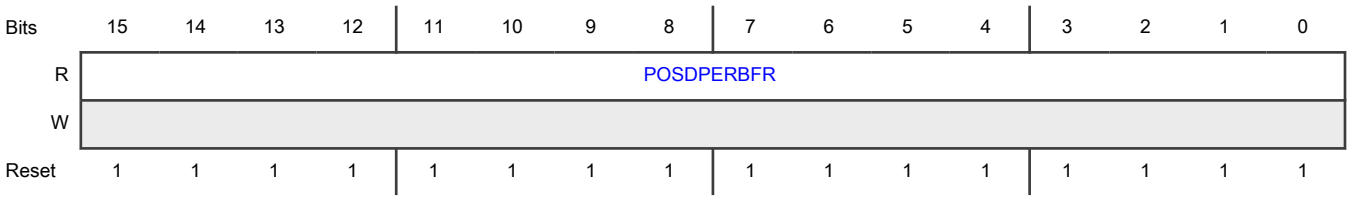
45.5.1.1.24 Position Difference Period Buffer Register (POSDPERBFR)

This register buffers value of Position Difference Period Counter.

Offset

Register	Offset
POSDPERBFR	2Eh

Diagram



Fields

Field	Description
15-0 POSDPERBFR	Position difference period buffer The Position Difference Period buffer register is a buffer register for the POSDPER value. POSDPERBFR is updated with the value of POSDPER when any edge occurs on PHASEA or PHASEB.

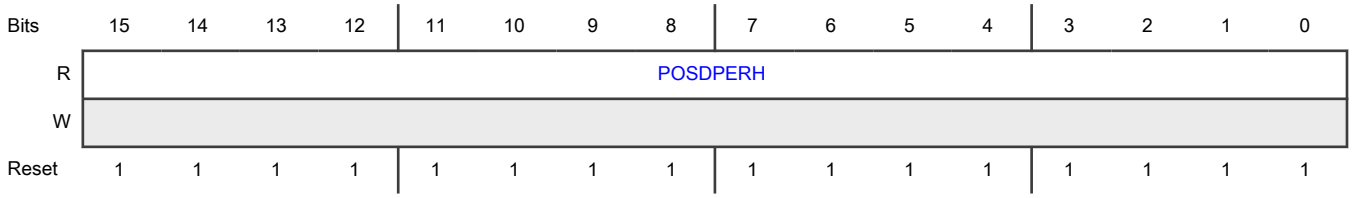
45.5.1.1.25 Position Difference Period Hold Register (POSDPERH)

This is hold register of Position Difference Period Counter value.

Offset

Register	Offset
POSDPERH	30h

Diagram



Fields

Field	Description
15-0 POSDPERH	Position difference period hold The Position Difference Period Hold register is a hold register for the POSDPER value. POSDPERH is updated with the value of POSDPERBFR when the POSD register is read.

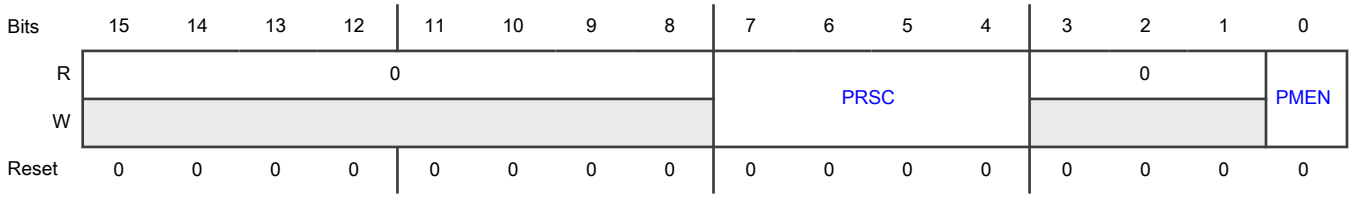
45.5.1.1.26 Control 3 Register (CTRL3)

ENC Control 3 register covers Prescaler values for Period clock divider and period measurement function.

Offset

Register	Offset
CTRL3	32h

Diagram



Fields

Field	Description
15-8 —	Reserved
7-4 PRSC	Prescaler The Prescaler field is used to prescale the peripheral clock that is used by the LASTEDGE and POSDPER counters. The clock is prescaled by a value of 2^PRSC which means that the prescaler logic can divide the clock by a minimum of 1 and a maximum of 32,768.
3-1 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
0 PMEN	Period measurement function enable This bit is used to indicate the period measurement functions of LASTEDGE, LASTEDGEH, POSDPER, POSDPERBFR, and POSDPERH are being used. 0 - Period measurement functions are not used. POSD is loaded to POSDH and then cleared whenever POSD, UPOS, LPOS, or REV is read. 1 - Period measurement functions are used. POSD is loaded to POSDH and then cleared only when POSD is read.

Chapter 46
Digital Microphone Interface Subsystem (DMIC)

46.1 Chip-specific DMIC information

This section shows how instances of the DMIC module connect with other modules in the device.

Table 384. Reference links to related information

Topic	Related module	Reference
Full description	DMIC	DMIC
System memory map		System memory map
Clocking		Clock distribution
Signal multiplexing	Port control	Signal multiplexing
DMA Controller	DMA	DMA Controller
Flexcomm Serial Communication	Flexcomm	I^2S configured for DMIC output

46.1.1 Module instances

This device contains one instance of DMIC.

46.1.2 Initialization

Initial configuration of the DMIC can be accomplished as follows:

- Enable the clock to the DMIC by setting AHBCLKCTRL1[DMIC] bitfield in SYSCON register section. This enables the register interface and the peripheral function clock.
- Select a clock source for the DMIC using the DMICFCLKSEL register in SYSCON register section.
- Select a clock divide for the DMIC using the DMICFCLKDIV register in SYSCON register section.
- Clear the DMIC peripheral reset in the PRESETCTRL1 register by writing to the PRESETCTRLCLR1 register in SYSCON registers.
- The DMIC provides interrupts to the NVIC. To allow interrupts to wake-up the device from Deep-sleep mode, enable the interrupts in the STARTEN0 register.
- Use the IOPCTL registers to connect the DMIC inputs and outputs to external pins.
- The DMIC provides DMA requests to the DMA controller.
- PDM internal setup:
  - Enable DMIC PDM channels via the EN_CH0/1 bits in the CHANEN register.
  - Set up the internal clock dividers for the PDM channels used via the DIVHFCLK0/1 registers.
  - If interrupts are used with this peripheral, enable them in the NVIC.
  - If DMA will be used with the PDM data flow, the related channels of the DMA controller must be set up. DMA must also be enabled in the FIFOCTRL[DMAEN] field for each channel using DMA.
  - Set up other functional configurations and controls for this peripheral as needed.
- HWVAD internal setup:
  - The HWVAD is active when the DMIC interface is active.

— Reset the filters with a '1' pulse of the HWVADRSTT[RSTT] field.

— Wait for a few milliseconds to let the filter converge.

— Enable the HWVAD interrupt with the appropriate NVIC bit.

— Start the HWVAD process by toggling the HWVADST10[ST10] field from '0' to '1' and back. This also clears the interrupt flag inside the HWVAD block.

— In the HWVAD interrupt service routine take appropriate action.

— Restart the HWVAD.

Wake-up for DMA only

The device can optionally be woken up from deep-sleep only far enough to perform needed DMA before returning to deep-sleep mode without the CPU waking up at all. This applies only if some interrupt requests DMA in deep-sleep mode. See the description of the HWWAKE register in SYSCTL

- Configure the DMA controller appropriately, including a transfer complete interrupt.
- Disable the related DMIC interrupt in the NVIC.
- Enable the DMA interrupt in the NVIC.
- Enable WAKEDMA in the HWWAKE register in SYSCTL.

46.1.3 Chip-specific Signals information

The DMIC subsystem PDM signals description table gives a brief summary of each of the PDM signals used by the DMIC subsystem.

NOTE

In order to output 2 channels of PDM Data, PDM_CLK01 must be used.

Table 385. DMIC subsystem PDM signals description

Signal	I/O	Description
PDM_CLK0, PDM_CLK1	O	Clock output to digital microphone(s) on the related PDM interface.
PDM_DATA0, PDM_DATA1	I	Data input from digital microphone(s) on the related PDM interface.

Recommended IOPCTL settings are shown here.

Table 386. Suggested PDM signal settings for the audio input

IOPCTL bit(s)	Name	Comment
3:0	FUNC	Select a function for this peripheral.
4	PUPDENA	Set to 0 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input).
5	PUPDSEL	Set to 0 unless PUPDENA = 1, then select appropriate value for pull-up or pull-down.
6	IBENA	Set to 1 (input buffer enabled).
7	SLEWRATE	Generally, set to 0 (standard mode).

Table continues on the next page...

Table 386. Suggested PDM signal settings for the audio input (continued)

IOPCTL bit(s)	Name	Comment
8	FULLDRIVE	Generally, set to 0 (normal output drive).
9	AMENA	Set to 0 (analog input mux, if any, disabled).
10	ODENA	Set to 0 unless simulated open-drain output is desired.
11	IIENA	Set to 0 (input function not inverted).

46.1.4 Clocking and DMIC data rates

The source for the DMIC base clock can be set in the DMICFCLKSEL register in SYSCON. Note that all of these clock sources may be divided by a factor of up to 256 by the DMIC clock divider, controlled by the DMICFCLKDIV register in SYSCON.

- FRO Clock
- Audio PLL Clock
- Master Clock In
- LPOSC
- 32 KHz Wake Clock

For the DMIC clock, the base clock divider values can be set in DIVHFCLK registers.

However, for power consumption reasons, it is preferable that the division to the required DMIC clock be done outside of the DMIC interface block (for example using register DMIC0FCLKDIV register in CLKCTL1 block).

46.2 Overview

The DMIC subsystem, includes the dual-channel digital PDM microphone interface (DMIC) and hardware voice activity detector (HWVAD).

46.2.1 Features

- DMIC (dual/stereo digital microphone interface)
  - PDM (Pulse-Density Modulation) data input for 2 channels
  - Flexible decimation.
  - 16 entry FIFO for each channel
  - DC blocking or unaltered DC bias can be selected
  - DMA supported on a channel-by-channel basis
  - Data can be transferred using DMA from Deep-sleep mode without waking up the CPU, then automatically returning to Deep-sleep mode.
  - Data from DMIC channels 0 and 1 can optionally be sent directly to a Flexcomm I^2S function for output.
- HWVAD (Hardware-based voice activity detector):
  - Optimized for PCM signals with 16 kHz sampling frequency
  - Configurable detection levels
  - Noise envelope estimator register output for further software analysis

46.3 Functional description

46.3.1 DMIC

The DMIC interface receives PDM data from multiple digital microphones and processes it to produce 24-bit PCM data. This data can be read by the CPU or DMA, and/or can be sent to a Flexcomm I^{2}S module for output (see chip-specific information). Many aspects of DMIC operation can be controlled. A block diagram of one DMIC channel is shown below.

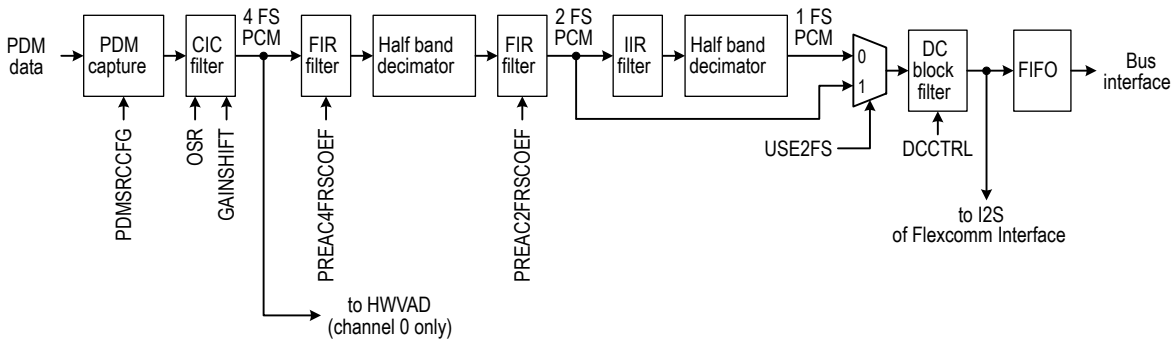


Figure 255. DMIC channel block diagram

See chip-specific information.

46.3.1.1 Clocking and DMIC data rates

The DMIC interface operation is determined by 3 clock domains:

- DMIC interface base clock: supply clock for the peripheral block
- DMIC clock: sample clock for the digital microphone
- PCM sample rate: sample rate of the PCM data resulting from the PDM to PCM conversion

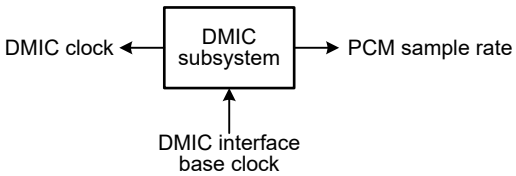


Figure 256. DMIC interface clock domains

For the DMIC clock, the base clock divider values can be set in DIVHFCLKn registers.

DIVHFCLKn divides the clock to PDM from the original clock supplied to the DMIC subsystem block.

If the supplied clock speed is 1.536MHz, a DIVHFCLKn register PDM clock divider value = 2, gives 768kHz on the microphone.

For example, to use the microphone in Low Power mode: Use 1FS mode, DMIC clock (1.536Mhz), DIVHFCLKn register PDM clock divider value = 2 (PDM clock :768Khz), sample rate = 16khz (768/(OSR=12)/ 4 (1FS mode)).

NOTE

For power consumption reasons, it is preferable that the division to the required DMIC clock be done outside of the DMIC interface block (for example, using the system configuration block registers).

The DMIC peripheral block is designed to run at a DMIC clock speed no faster than 6.144 MHz and with an input frequency no faster than 4 x 6.144 MHz = 24.576 MHz. With regards to power consumption the lowest possible frequency should be selected. This frequency very much depends on the application requirements.

- For a good quality voice tag recognition the DMIC should be clocked at least with 800 kHz.

Depending on the current operating mode of the application, the clocks can be set dynamically from one sample rate to the other. For a glitch free reduction of the DMIC clock rate by factor 2 the DMIC interface contains dedicated circuitry. By setting PHY_CTRLn[PHY_HALF] = 1, the DMIC clock is divided to half the frequency internally used for the filters. This enables an on-the-fly switching of the DMIC clock without affecting the operation of the filters. As long as the sample quality on half of the frequency is good enough for the application (for example in listening mode only) this reduction in frequency helps to decrease the power consumption of the external digital microphone.

If the PDM interface operates during Deep-sleep mode (always listening), then the presence of the clock source in this mode must be taken into account as well. For example, the PLL output is not present during Deep-sleep mode, but the 12 MHz FRO is there.

In general, other clocks such as the 48 MHz FRO or the watchdog oscillator are available in Deep-sleep mode. The clock choice depends on the use case and whether a faster or slower clock provides any advantage to the system. At high PDM data rates, for example at 6 MHz, a 48 MHz clock shortens the 'awake' periods compared to 12 MHz operation. A trade-off between the sleep and the active periods, and the internal voltage required at the chosen clock rate, that determines which of the clocks perform better in terms of average power consumption.

The watchdog oscillator low power operation can help to reduce the power consumption in simple voice detection setups. By running the DMIC interface on the slow watchdog oscillator frequency, the HWVAD feature can provide a first audio detection trigger signal to the system. After wake-up, the sample rate as well as the processor performance is increased in order to run more sophisticated voice detection and/or voice recognition algorithms.

46.3.1.2 PDM to PCM conversion

The filter block for PDM to PCM conversion consists of four stages. For more details on filters, refer to AN12590 *PDM MEMS Microphone Audio Path Optimal Settings*.

1. It begins with a CIC (Cascaded-Integrator Comb) filter which is an optimized finite impulse response (FIR) filter combined with a decimator.The CIC filter converts the PDM stream from the digital microphone into PCM data with a given oversampling rate, set in the OSR registers for each channel.
2. The second block performs a decimation by 2 and compensates for a roll-off at the upper limit of the audio band.
3. The third block decimates the signal again by half, resulting in a PCM signal with the desired sample rate.
4. A final DC filter removes any unwanted DC component in the audio signal.

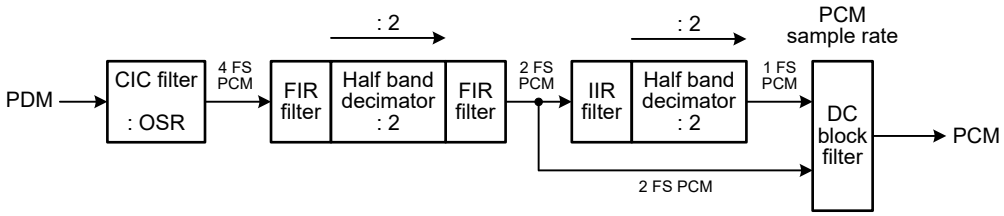


Figure 257. Principle structure of the PDM to PCM conversion

Choosing 2 FS reduces the required DMIC base clock by a factor of 2. Set this in the USE2FS[USE2FS] bit.

The PDM to PCM conversion block is designed for providing best results for PCM output signals with a 16 kHz sample rate, covering the enhanced 8 kHz speech band widely used in communication systems. However, other sample rates can be realized as well.

The final relation between the DMIC clock rate and the PCM audio sample rate is:

Table 387. DMIC input and output clock rates

2 FS mode	1 FS mode
PCM sample rate = DMIC clock rate / (2 x OSR)	PCM sample rate = DMIC clock rate / (4 x OSR)

Example: DMIC clock = 800 kHz, 2 FS mode used, OSR = 25

PCM sample rate = 800 kHz / (2 x 25)

PCM sample rate = 16 kHz

The 800 kHz DMIC data is downsampled by 25 times to 32 kHz. With the following half-band filter (2 FS) the final PCM sample rate is 16 kHz.

46.3.1.3 FIFO and DMA operation

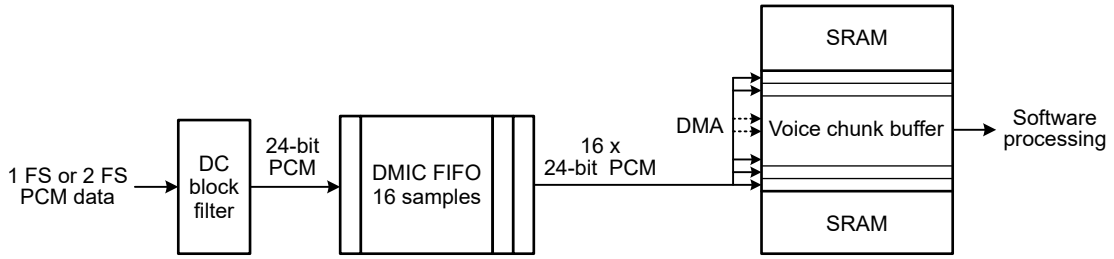


Figure 258. DMIC FIFO and DMA

The 24-bit wide FIFO of the DMIC interface consists of 16 entries for each of the channels. The trigger level for the FIFO can be set in FIFO_CTRLn register individually for each channel.

The trigger level interrupt for the DMIC interface needs to be enabled in the NVIC, and with the related FIFO_CTRLn[INTEN] bit. The FIFO_CTRLn[DMAEN] bit enables DMA operation. With each FIFO trigger level event the DMA performs a copy of the data from the FIFO into SRAM. This data batching works without contribution of the core. When reaching the defined chunk buffer size, the DMA issues an interrupt to the Arm core for further processing of the data.

This also works when the device is in Deep-sleep mode, as the FIFO event is able to wake up the required part of the hardware. After the DMA finishes the job, the device will return to Deep-sleep mode. See the DMA chapter for DMIC channel DMA requests and trigger muxes.

Since each DMIC channel provides a separate DMA request, the most obvious configuration of DMA is to have left and right data in separate memory buffers. However, it is possible to configure the DMA controller to interleave left and right data if that is preferable in the application. To do this, the DMA is set up with a data size of halfword, but the next address written to is a word address distance away. The two descriptors would be started on consecutive halfwords. Data is delivered by the DMIC as left channel followed by right channel for each PCM stereo sample.

If more history data is required for a software algorithm, another DMA request can be set up, which copies the current chunk into a larger ring buffer structure. For algorithms like voice detection or voice recognition this is key, in order to converge the software filters to the current background noise situation.

For operation without DMA the dedicated DMIC FIFO interrupt can be enabled in order to inform the Arm core about the FIFO status.

Example:

- PCM output sample rate is 16 kHz
- The FIFO gets full every 1 ms
- The DMA copies every 1 ms the content of the DMIC FIFO to SRAM
- The DMA is configured to move 512 bytes (= 256 PCM samples) from the DMIC FIFO to SRAM before issuing a DMA interrupt
- Every 16 ms the 256 PCM samples are processed by the Arm core

The FIFO_CTRLn[DMAEN] field can be used to synchronize the start of multiple channels, allowing a single DMA completion interrupt from one channel to signal that DMA data from multiple channels may be collected.

46.3.1.4 Usage of the DMIC interface in reduced power modes

The DMIC interface can batch the serial PDM stream from a digital microphone in Deep-sleep mode. This requires an appropriate base clock which is active in the respective power saving mode. The best fit for this base clock is the 12 MHz FRO, which provides a good trade-off between power consumption and performance. For lower power operation the watchdog oscillator can be used, taking into account that the clock is relatively inaccurate and the DMIC sample rate is rather low. At any time an external low power clock, connected to pin MCLK, can be used.

In combination with the HWVAD this provides lowest power consumption in listening mode. Except for the short periods with DMA activity, the MCU can remain in Deep-sleep mode until the wave envelope detector of the HWVAD identifies an energy change event and issues an interrupt. With the DMA set to larger transfer sizes (maximum is 1024 transfers), there is history data available for any type of software-based analysis of the data causing the HWVAD event.

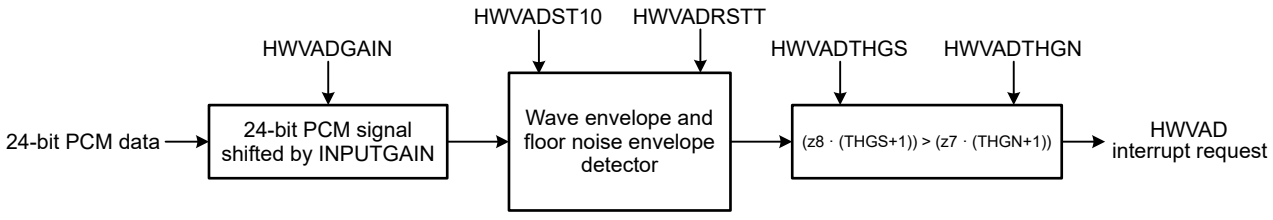
This history data also enables the system to realize different strategies for dealing with an HWVAD event. A concrete analysis of the data could for example just be started when the HWVAD detected events over a longer time frame. Analyzing the data to to understand known spurious noise could lead to strategies to avoid activating the Arm core for unnecessary events. In case the decision has been taken to take the next step in data analysis, the history buffer still contains the complete PCM data sampled since the first event, nothing is lost. The system can stay in power save mode longer if spurious events can be filtered out.

46.3.2 HWVAD

The hardware voice activity detector (HWVAD) implements a wave envelope detector and a floor noise envelope detector. It provides an interrupt when the delta between the two detectors is larger than a predefined value. The input signal for the HWVAD can come from DMIC channel 0. The basic detection of a voice activity can be the starting point for a more sophisticated task like voice recognition. As with the DMA for the DMIC subsystem, the HWVAD can be active during Deep-sleep mode and provide lower power operation, compared with a software based implementation. The DMIC receives PDM data and produces a data stream that can be read by the CPU or DMA.

The hardware voice activity detector (HWVAD) analyzes the PCM data from the DMIC channel 0 by means of a filter block. Both the noise floor and the signal wave are examined and result in separate filter outputs. The HWVAD interrupt is issued when a specific delta between the signal and the noise result is detected

Gain levels for the input signal as well as for the signal and noise filter outputs can be set independently from each other, in order to adapt the HWVAD to different acoustic situations.



HWVAD block diagram

Because of the non-uniqueness of the input signal, which includes noise and voice with various frequency components and different volumes, there is no one-and-only operation mode for the HWVAD. A few parameters as well as the chronology can play an important role for a good performance.

46.3.2.1 Basic operations

There are some basic operations for the HWVAD, which can be combined in various ways to achieve different behavior.

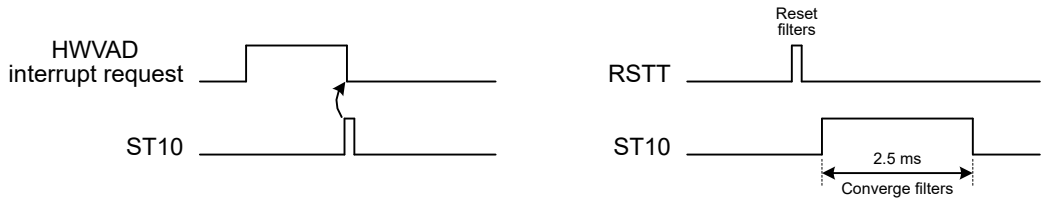


Figure 259. Use of ST10 and RSTT controls (in the HWVADST10 and HWVADRSTT registers)

1. The internal HWVAD interrupt flag is reset with a short pulse on ST10
2. A pulse on RSTT resets all detection filters
3. Keeping ST10 high for a period causes a special filter convergence. The 2.5 ms period is valid for 800 kHz DMIC sample rate. For 1 MHz sample rate the period is 2 ms.

Use the HWVADST10[ST10] bit to prepare HWVAD for an interrupt.

The internal flag is reset with the rising edge of ST10 and HWVAD waits for the next event. In case the application involves some post-processing after a HWVAD event (outside of the interrupt service routine), the flag should only be cleared at the end of this processing. The interrupt status on NVIC level is not affected by this bit setting

Use the HWVADRSTT[RSTT] bit to reset all filters.

After this reset the HWVAD filters need to converge, so for the first few milliseconds the result is not reliable. The HWVAD interrupt should be masked on the NVIC level during this time frame. The wait period depends on the sample rate of the incoming data.

- For a 1 MHz DMIC sample rate the filters need about 2ms to converge.
- For an 800 kHz DMIC sample rate the filters need about 2.5ms to converge.

Depending on the use case, it might makes sense to reset the filters before starting into a new detection process. For a voice application the filters can adapt continuously to the background environment, between the voice events there is normally enough time to let the filters converge to a changed background noise situation.

Keeping ST10 on high level during the convergence period enables a special mode. If the filters should adapt to a current background noise floor (without voice), then this can be done during this period. With HWVADST10[ST10] returning to a low level, the filter calculation is then based on a different filter pre-setting. This could be an advantage in special types of applications, where the signal is not continuously delivered to the HWVAD. In a DMIC system with continuous sampling, this convergence period is not required. In that case, the HWVADST10[ST10] bit is just used to clear the interrupt flag.

A complete setup sequence for standard operation looks like this:

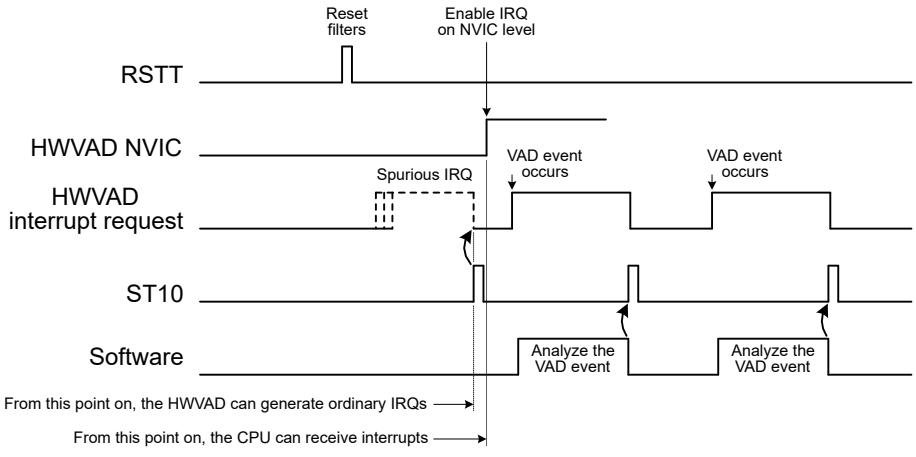


Figure 260. Complete HWVAD setup

1. Reset filters with the HWVADRSTT[RSTT] bit and provide some time to let the filter converge to the signal conditions.

2. Pulse bit HWVADST10[ST10] to clear any spurious interrupts which were generated during bad filter conditions.
3. Enable HWVAD interrupts on the NVIC level.
4. Process the VAD event in case of an interrupt, when finished clear the interrupt flag with a high pulse of HWVADST10[ST10].

46.3.2.2 Extended operation

There are a few parameters which can be set to influence the behavior of the HWVAD. There is also an intermediate filter result value available, which can be used for proprietary software-based analysis.

46.3.2.2.1 Input gain setting

The 24-bit PCM input signal can be shifted left or right with the gain setting in the HWVADGAIN register. This increases or decreases the volume of the input signal for the HWVAD processing. Note that the reset value 0x05 equals a gain factor of 1, where the signal is not shifted in either direction.

46.3.2.2.2 Filter result gain setting

The output values for the final equation can also have a gain factor.

```
[ z8 x (THGS + 1) ] > [ z7 x (THGN + 1) ]
```

These gain factors determine the proportion between the results of the signal and the noise filters. The values depend on the audio signal and noise environment.

- For low noise environments, the reset values THGN = 0 and THGS = 4 are more suitable.
- For noisy environments, the gain for THGN and THGS needs to be increased.
- In a typical voice recognition application, THGN = 3 and THGS = 6 is a good starting point.

46.3.2.2.3 High pass filter setting

The setting in the HWVADHPFS register can be used to adapt the filters to different background noise situations. In order to find the best setting, software could perform a rough spectral analysis of the audio signal.

- For a background with more low-frequency content, HWVADHPFS[HPFS] should be set to 0x1. This is the standard use case.
- For environments where the low-frequency content is small, HWVADHPFS[HPFS] can be set to 0x2.

46.3.2.2.4 Noise floor evaluation

The register HWVADLOWZ contains 2 bytes of the output of filter stage z7, which computes the noise floor. The characteristic of the filter block for voice applications is best for a value of 500 - 1000 in LOWZ. Software can tune the input gain to get the HWVADLOWZ[LOWZ] value into this region.

NOTE

For power saving reasons the HWVADLOWZ register is not synchronized to the AHB bus clock domain. To ensure correct data is read, the register should be read twice. If the data is the same, then the data is correct, if not, the register should be read one more time. The noise floor is a slowly moving calculation, so several reads in a row can guarantee that register value being read is not in the middle of a transition.

46.4 Signals

This table describes the PDM signals used by the DMIC module.

Signal	I/O	Description
PDM_CLKn	O	Clock output to digital microphone(s) on the related PDM interface.
PDM_DATAn	I	Data input from digital microphone(s) on the related PDM interface.

The PDM interface provides options to support 2 single-channel microphones or a single stereo microphone.

These figures show specific use case examples.

The first example is the simple case of one independent microphone.

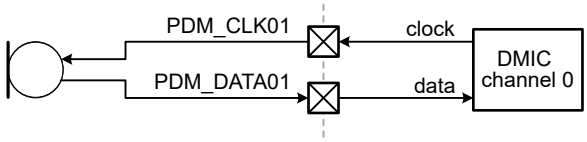


Figure 261. Example connection for a single independent microphone

This example shows two independent microphones.

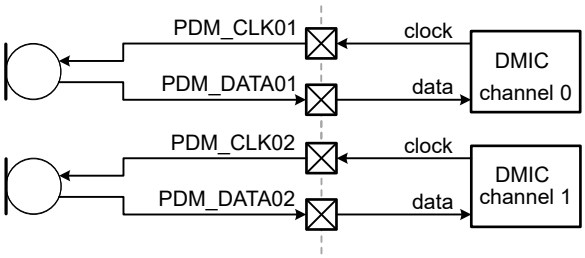


Figure 262. Example connection to two independent microphones

For the example with two microphones sharing the same clock and data lines, the DMIC channel 0 and DMIC channel 1 are configured to use the same source clock and frequency. Both microphones use the same clock from channel 0.

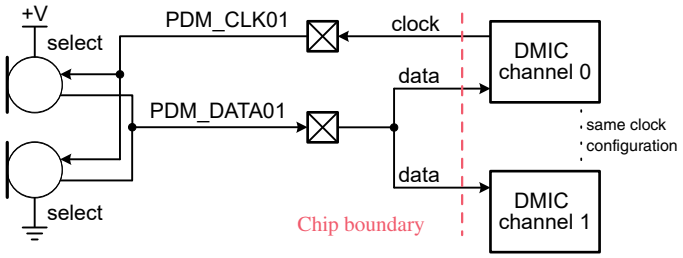


Figure 263. Example connection to two microphones sharing clock and data lines

46.5 Initialization

This section describes the general initialization of the DMIC module.

- Enable the clock to the DMIC by in the clock control registers. This enables the register interface and the peripheral function clock.
- Select a clock source for the DMIC in the clock control registers.
- Select a clock divide for the DMIC in the clock control registers.
- Clear the DMIC peripheral reset in the reset registers.
- The DMIC provides interrupts to the NVIC. To allow interrupts to wake-up the device from Deep-sleep mode, enable interrupts in the system configuration registers.
- Use the IO Pad controller (IOPCTL) registers to connect the DMIC inputs and outputs to external pins.
- The DMIC provides DMA requests to the DMA controller.
- PDM internal setup:
  - Enable DMIC PDM channels via the CHANEN register.
  - Set up the internal clock dividers for the PDM channels used via the DIVHFCLK0/1 registers.
  - If interrupts will be used with this peripheral, enable them in the NVIC.
  - If DMA will be used with the PDM data flow, the related channels of the DMA controller must be set up. DMA must also be enabled in the FIFO_CTRLn[DMAEN] field for each channel using DMA.
  - Set up other functional configurations and controls for this peripheral as needed.
- HWVAD internal setup:
  - The HWVAD is active when the DMIC interface is active.
  - Reset the filters with '1' pulse of HWVADRSTT[RSTT].
  - Wait for a few milliseconds to let the filter converge.
  - Enable the HWVAD interrupt with the appropriate NVIC bit.
  - Start the HWVAD process by toggling the HWVADST10[ST10] field from '0' to '1' and back. This also clears the interrupt flag inside the HWVAD block.
  - In the HWVAD interrupt service routine take appropriate action.
  - Restart the HWVAD.

Wake-up for DMA only

The device can optionally wake-up from Deep-sleep to perform a needed DMA before returning to Deep-sleep mode without the CPU. This applies only if some interrupt requests DMA in Deep-sleep mode.

- Configure the DMA controller appropriately, including a transfer complete interrupt.
- Disable the related DMIC interrupt in the NVIC.
- Enable the DMA interrupt in the NVIC.
- Enable the associated hardware wake-up register in the system configuration registers.

46.6 Memory map and register definition

This section includes the DMIC module memory map and detailed descriptions of all registers.

Each of the 2 DMIC channels has a set of configuration registers with the channel noted with a suffix 0 - 1. The n at the end of the register name stands for the channel 0 - 1.

- OSRn
- DIVHFCLKn
- PREAC2FSCOEFn

- PREAC4FSCOEFn
- GAINSHIFTn
- FIFO_CTRLn
- FIFO_STATUSn
- FIFO_DATAn
- PHY_CTRLn
- DC_CTRLn

Other DMIC registers control the HWVAD, FS ouput choice (USE2FS), and multiple channel enable (CHANEN).

46.6.1 DMIC register descriptions

46.6.1.1 DMIC memory map

DMIC0 base address: 4009_0000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Oversample Rate (OSR0)	32	See section	0000_0000
4	DMIC Clock (DIVHFCLK0)	32	RW	0000_0000
8	Compensation Filter for 2 FS (PREAC2FSCOEF0)	32	See section	0000_0000
C	Compensation Filter for 4 FS (PREAC4FSCOEF0)	32	See section	0000_0000
10	Decimator Gain Shift (GAINSHIFT0)	32	See section	0000_0000
80	FIFO Control (FIFO_CTRL0)	32	See section	0000_0000
84	FIFO Status (FIFO_STATUS0)	32	See section	0000_0000
88	FIFO Data (FIFO_DATA0)	32	RO	0000_0000
8C	Physical Control (PHY_CTRL0)	32	See section	0000_0000
90	DC Filter Control (DC_CTRL0)	32	See section	0000_0000
100	Oversample Rate (OSR1)	32	See section	0000_0000
104	DMIC Clock (DIVHFCLK1)	32	RW	0000_0000
108	Compensation Filter for 2 FS (PREAC2FSCOEF1)	32	See section	0000_0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
10C	Compensation Filter for 4 FS (PREAC4FSCOEF1)	32	See section	0000_0000
110	Decimator Gain Shift (GAINSHIFT1)	32	See section	0000_0000
180	FIFO Control (FIFO_CTRL1)	32	See section	0000_0000
184	FIFO Status (FIFO_STATUS1)	32	See section	0000_0000
188	FIFO Data (FIFO_DATA1)	32	RO	0000_0000
18C	Physical Control (PHY_CTRL1)	32	See section	0000_0000
190	DC Filter Control (DC_CTRL1)	32	See section	0000_0000
F00	Channel Enable (CHANEN)	32	See section	0000_0000
F10	Use 2 FS register (USE2FS)	32	See section	0000_0000
F14	Global Channel Synchronization Enable (GLOBAL_SYNC_EN)	32	See section	0000_0000
F18	Global channel synchronization counter value (GLOBAL_COUNT_VAL)	32	RW	0000_0000
F1C	DMIC decimator reset (DECRESET)	32	See section	0000_0000
F80	HWVAD Input Gain (HWVADGAIN)	32	See section	0000_0005
F84	HWVAD Filter Control (HWVADHPFS)	32	See section	0000_0001
F88	HWVAD Control (HWVADST10)	32	See section	0000_0000
F8C	HWVAD Filter Reset (HWVADRSTT)	32	See section	0000_0000
F90	HWVAD Noise Estimator Gain (HWVADTHGN)	32	See section	0000_0000
F94	HWVAD Signal Estimator Gain (HWVADTHGS)	32	See section	0000_0004
F98	HWVAD Noise Envelope Estimator (HWVADLOWZ)	32	RO	0000_0000

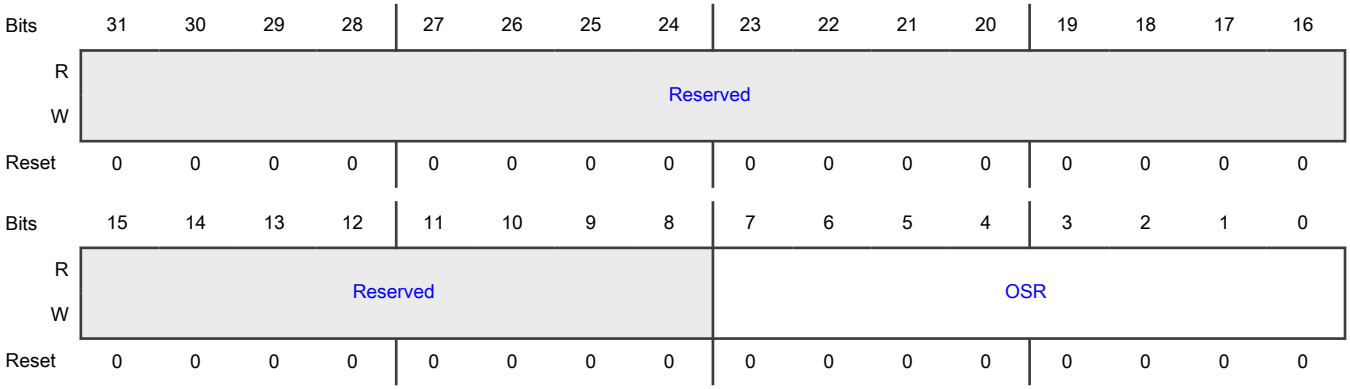
46.6.1.1.1 Oversample Rate (OSR0 - OSR1)

The OSRn register selects the oversample rate (CIC decimation rate) for the related input channel.

Offset

Register	Offset
OSR0	0h
OSR1	100h

Diagram



Fields

Field	Description
31-8 —	Reserved Read value is undefined, only zero should be written.
7-0 OSR	Oversample Rate Selects the oversample rate for the related input channel.

46.6.1.1.2 DMIC Clock (DIVHFCLK0 - DIVHFCLK1)

The DIVHFCLKn register controls the clock pre-divider for the related input channel, using the divider to generate the PDM clock from DMIC clock input.

Offset

Register	Offset
DIVHFCLK0	4h
DIVHFCLK1	104h

Diagram

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

Fields

Field	Description
31-4 —	Reserved Read value is undefined, only zero should be written.
3-0 PDMDIV	PDM Clock Divider Value Divide by factor to create PDM Clock (enumerated type) 0000 - Divide by 1 0001 - Divide by 2 0010 - Divide by 3 0011 - Divide by 4 0100 - Divide by 6 0101 - Divide by 8 0110 - Divide by 12 0111 - Divide by 16 1000 - Divide by 24 1001 - Divide by 32 1010 - Divide by 48 1011 - Divide by 64 1100 - Divide by 96 1101 - Divide by 128 1110-1111 - Reserved

46.6.1.1.3 Compensation Filter for 2 FS (PREAC2FSCOEF0 - PREAC2FSCOEF1)

The PREAC2FSCOEFn register selects the pre-emphasis filter coefficient for the related input channel when 2 FS mode is used.

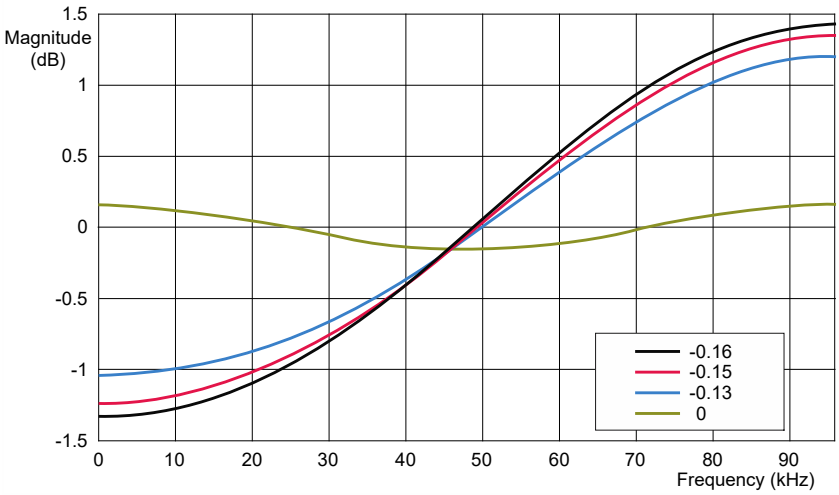


Figure 264. Pre-emphasis filter quantized response at 96 kHz

Offset

Register	Offset
PREAC2FSCOEF0	8h
PREAC2FSCOEF1	108h

Diagram

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

Fields

Field	Description
31-2 —	Reserved Read value is undefined, only zero should be written.
1-0 COMP	Compensation value Pre-emphasis filter coefficient for 2 FS mode. Co-efficient choice for CIC droop compensation droop filter.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00 - Compensation = 0. This is the recommended setting. 01 - Compensation = -0.16 10 - Compensation = -0.15 11 - Compensation = -0.13

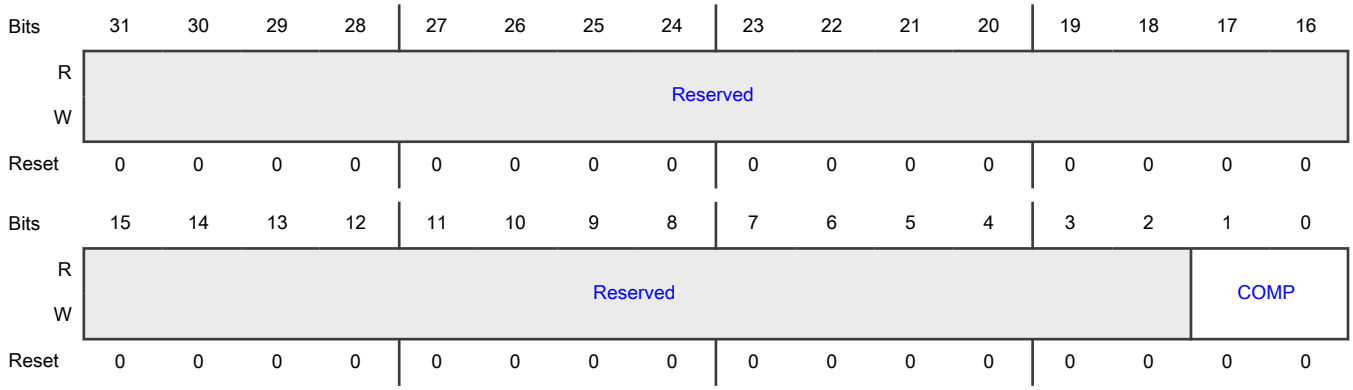
46.6.1.1.4 Compensation Filter for 4 FS (PREAC4FSCOEF0 - PREAC4FSCOEF1)

The PREAC4FSCOEFn register selects the pre-emphasis filter coefficient for the related input channel when 4 FS mode is used. See the diagram in PREAC2FSCOEFn.

Offset

Register	Offset
PREAC4FSCOEF0	Ch
PREAC4FSCOEF1	10Ch

Diagram



Fields

Field	Description
31-2 —	Reserved Read value is undefined, only zero should be written.
1-0 COMP	Compensation value Pre-emphasis filter coefficient for 4 FS mode. Co-efficient choice for CIC droop compensation droop filter. 00 - Compensation = 0. This is the recommended setting. 01 - Compensation = -0.16

Table continues on the next page...

Table continued from the previous page...

Field	Description
	10 - Compensation = -0.15 11 - Compensation = -0.13

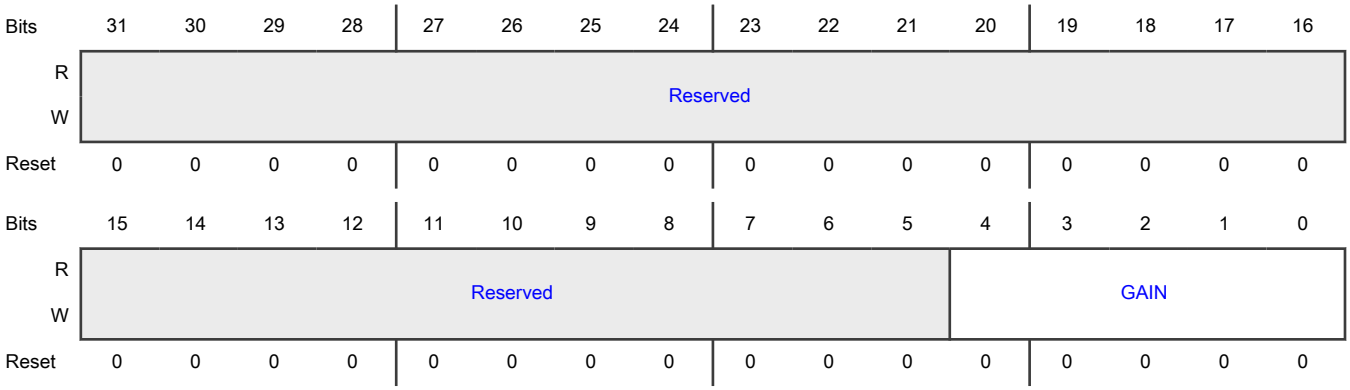
46.6.1.1.5 Decimator Gain Shift (GAINSHIFT0 - GAINSHIFT1)

The GAINSHIFTn register adjusts the gain of the 4 FS PCM data from the input filter.

Offset

Register	Offset
GAINSHIFT0	10h
GAINSHIFT1	110h

Diagram



Fields

Field	Description
31-5 —	Reserved Read value is undefined, only zero should be written.
4-0 GAIN	Gain Gain shift for decimator output can be positive or negative number. Gain control, as a positive or negative (two's complement) number of bits to shift.

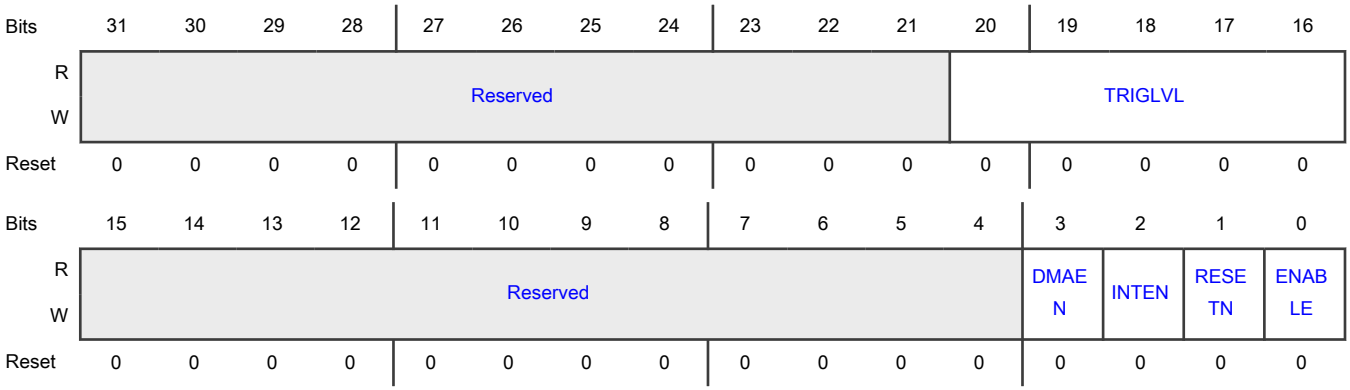
46.6.1.1.6 FIFO Control (FIFO_CTRL0 - FIFO_CTRL1)

FIFO_CTRLn configures FIFO usage.

Offset

Register	Offset
FIFO_CTRL0	80h
FIFO_CTRL1	180h

Diagram



Fields

Field	Description
31-21 —	Reserved Read value is undefined, only zero should be written.
20-16 TRIGLVL	FIFO Trigger Level for Interrupt Selects the data trigger level for interrupt or DMA operation. If enabled to do so, the FIFO level can wake up the device just enough to perform DMA, then return to the reduced power mode. TRIGLVL value + 1 = the number of received entries. Representative bit settings: 0_0000 - Trigger when the FIFO has received one entry (is no longer empty). 0_0001 - Trigger when the FIFO has received two entries. 0_1110 - Trigger when the FIFO has received 15 entries. 0_1111 - Trigger when the FIFO has received 16 entries (has become full).
15-4 —	Reserved Read value is undefined, only zero should be written.
3 DMAEN	DMA Enable 0 - DMA requests are not enabled. 1 - DMA requests based on FIFO level are enabled.
2 INTEN	Interrupt Enable. 0 - FIFO level interrupts are not enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - FIFO level interrupts are enabled.
1 RESETN	FIFO Reset 0 - Reset the FIFO. This must be cleared before resuming operation. 1 - Normal operation
0 ENABLE	FIFO Enable. 0 - Disabled. FIFO is not enabled. Enabling a DMIC channel with the FIFO disabled could be useful while data is being streamed to a Flexcomm I^2S module, or in order to avoid a filter settling delay when a channel is re-enabled after a period when the data was not needed. 1 - FIFO is enabled. The FIFO must be enabled in order for the CPU or DMA to read data from the DMIC via the FIFODATA register.

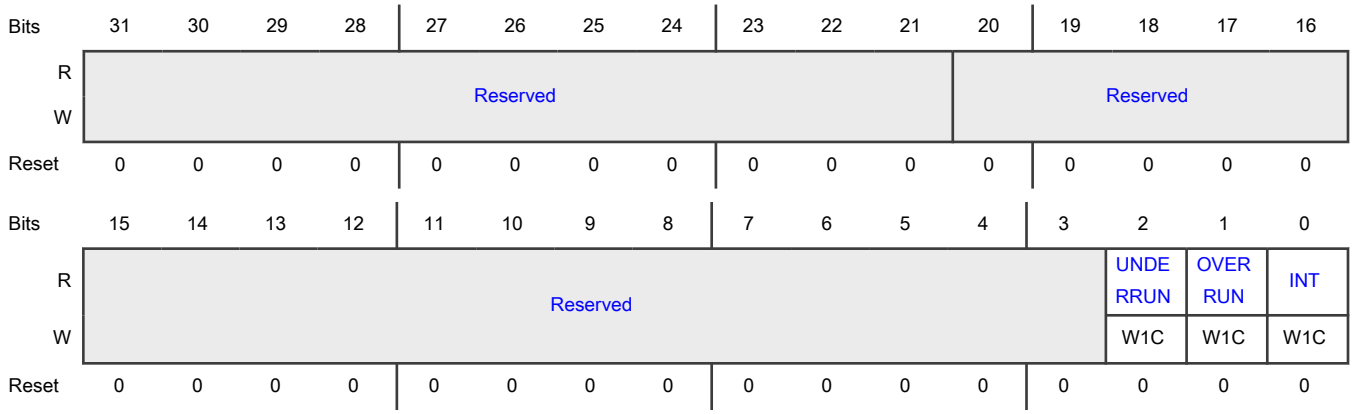
46.6.1.1.7 FIFO Status (FIFO_STATUS0 - FIFO_STATUS1)

The FIFO_STATUSn register provides status information for the FIFO and also indicates an interrupt from the peripheral function.

Offset

Register	Offset
FIFO_STATUS0	84h
FIFO_STATUS1	184h

Diagram



Fields

Field	Description
31-21	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	Read value is undefined, only zero should be written.
20-16 —	Reserved
15-3 —	Reserved Read value is undefined, only zero should be written.
2 UNDERRUN	Underrun Detected (write 1 to clear)
1 OVERRUN	Overrun Detected (write 1 to clear)
0 INT	Status of Interrupt (write 1 to clear)

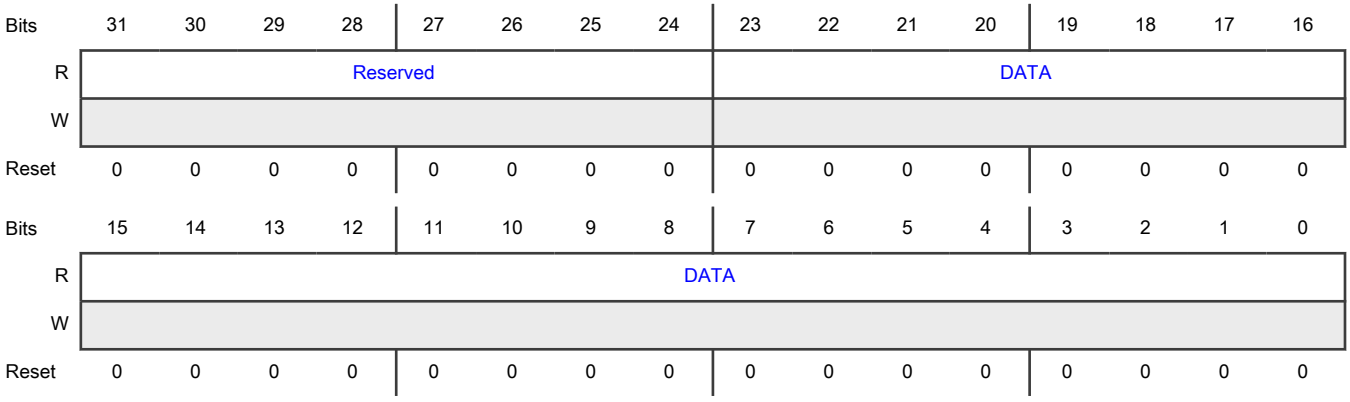
46.6.1.1.8 FIFO Data (FIFO_DATA0 - FIFO_DATA1)

The FIFODATAn register is used to read values that have been received by the via the PDM stream.

Offset

Register	Offset
FIFO_DATA0	88h
FIFO_DATA1	188h

Diagram



Fields

Field	Description
31-24 —	Reserved Read value is undefined, only zero should be written.
23-0 DATA	PCM Data Data from the top of the input filter FIFO.

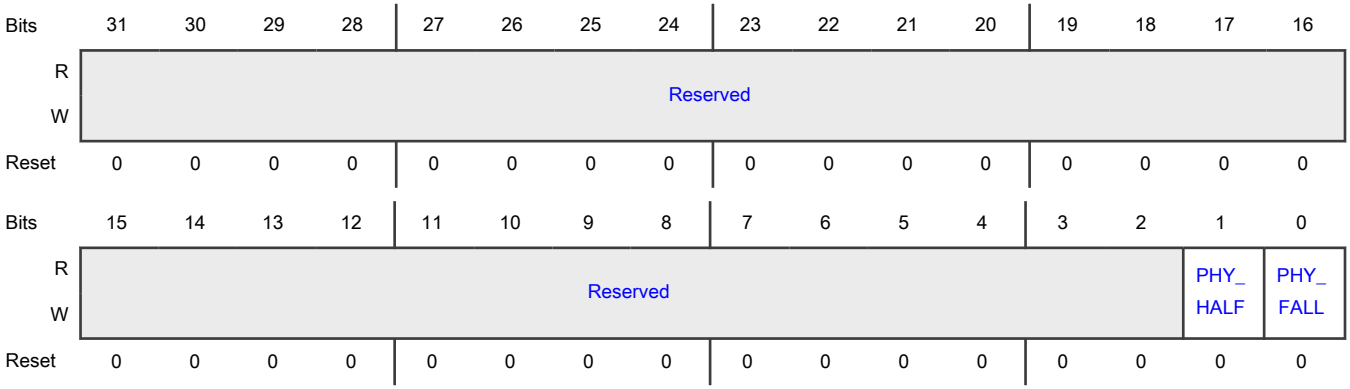
46.6.1.1.9 Physical Control (PHY_CTRL0 - PHY_CTRL1)

The PHY_CTRLn register configures how the PDM source signals are interpreted.

Offset

Register	Offset
PHY_CTRL0	8Ch
PHY_CTRL1	18Ch

Diagram



Fields

Field	Description
31-2 —	Reserved Read value is undefined, only zero should be written.
1 PHY_HALF	Use Half rate sampling (ie Clock to dmic is sent at half the speed than the decimator is providing) 0 - Standard half rate sampling. The clock to the DMIC is sent at the same rate as the decimator is providing. 1 - Use half rate sampling. The clock to the DMIC is sent at half the rate that the decimator is providing.
0	Capture DMIC on Falling edge (0 means on rising)

Table continues on the next page...

Table continued from the previous page...

Field	Description
PHY_FALL	0 - Capture PDM_DATA on the rising edge of PDM_CLK. 1 - Capture PDM_DATA on the falling edge of PDM_CLK.

46.6.1.1.10 DC Filter Control (DC_CTRL0 - DC_CTRL1)

The DC_CTRLn register controls the DC filter. The frequencies noted for DCPOLE in the table assume a PCM output frequency of 16 kHz. If the actual PCM output frequency is 8 kHz, for example, the noted frequencies would be divided by 2.

Offset

Register	Offset
DC_CTRL0	90h
DC_CTRL1	190h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	Reserved						SIGNEXT...	SATURAT...	DCGAIN				Reserved		DCPOLE	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-10 —	Reserved Read value is undefined, only zero should be written.
9 SIGNEXTEND	Sign Extend SIGNEXTEND allows processing of 24-bit audio data on 32-bit machines. 0 - Disabled. The top byte of the FIFODATA register is always 0. 1 - Enabled. The top byte of the FIFODATA register is sign extended.
8	Saturate at 16 Bit Selects 16-bit saturation.

Table continues on the next page...

Table continued from the previous page...

Field	Description
SATURATEAT16BIT	0 - Do not Saturate. Results roll over if out range and do not saturate. 1 - Saturate. If the result overflows, it saturates at 0xFFFF for positive overflow and 0x8000 for negative overflow.
7-4 DCGAIN	DC Gain Fine gain adjustment in the form of a number of bits to downshift.
3-2 —	Reserved Read value is undefined, only zero should be written.
1-0 DCPOLE	DC Block Filter 00 - Flat Response, no filter 01 - 155 Hz 10 - 78 Hz 11 - 39 Hz

46.6.1.1.11 Channel Enable (CHANEN)

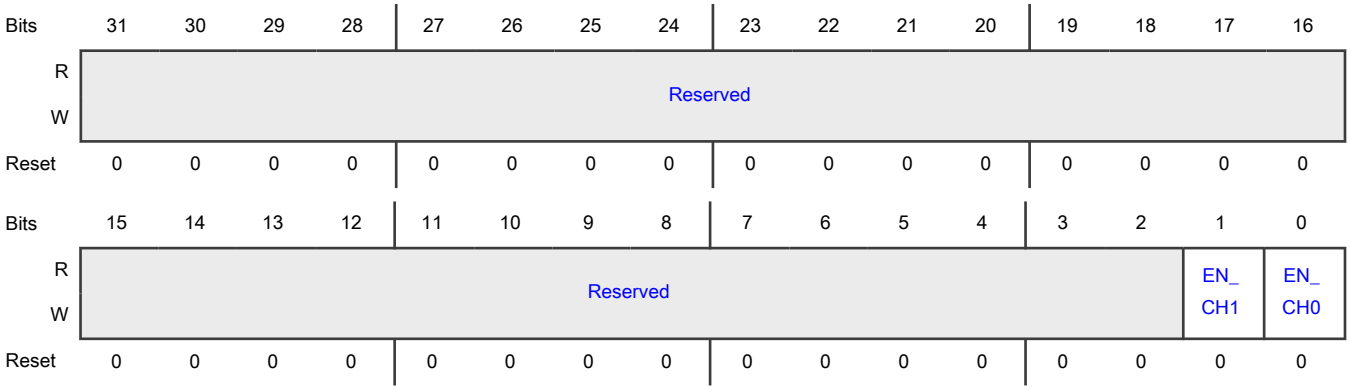
The CHANEN register allows enabling selected PDM channels.

The CHANEN register can be used to start multiple channels at the same time by setting more than one channel enable bit with a single write. In addition to synchronizing multiple channels, this allows a DMA completion interrupt from one channel to signal that DMA data from multiple channels may be collected.

Offset

Register	Offset
CHANEN	F00h

Diagram



Fields

Field	Description
31-2 —	Reserved Read value is undefined, only zero should be written.
1-0 EN_CHn	Enable Channel n 0 - PDM channel n is disabled. 1 - PDM channel n is enabled.

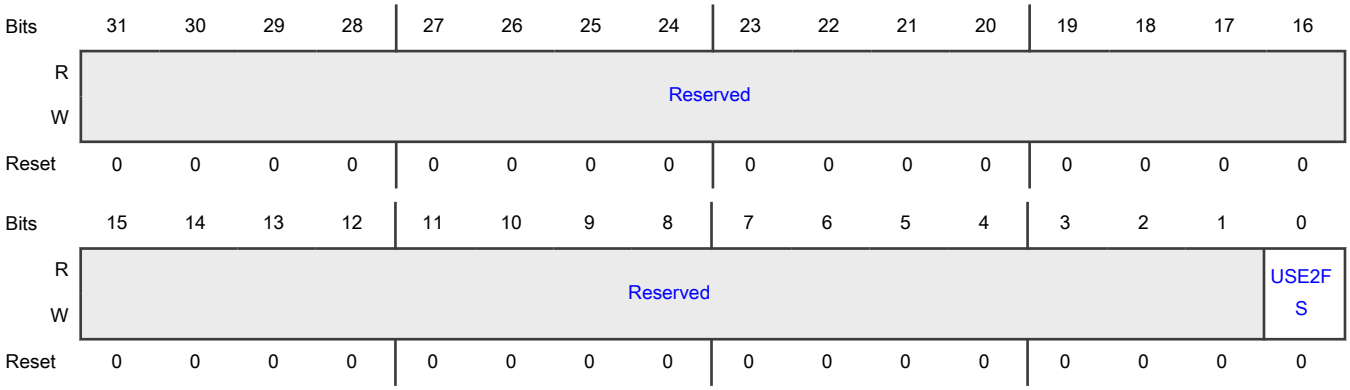
46.6.1.1.12 Use 2 FS register (USE2FS)

The USE2FS register allows selecting 2 FS output rather than 1 FS output.

Offset

Register	Offset
USE2FS	F10h

Diagram



Fields

Field	Description
31-1 —	Reserved Read value is undefined, only zero should be written.
0 USE2FS	Use 2FS register 0 - Use 1 FS output for PCM data. 1 - Use 2 FS output for PCM data.

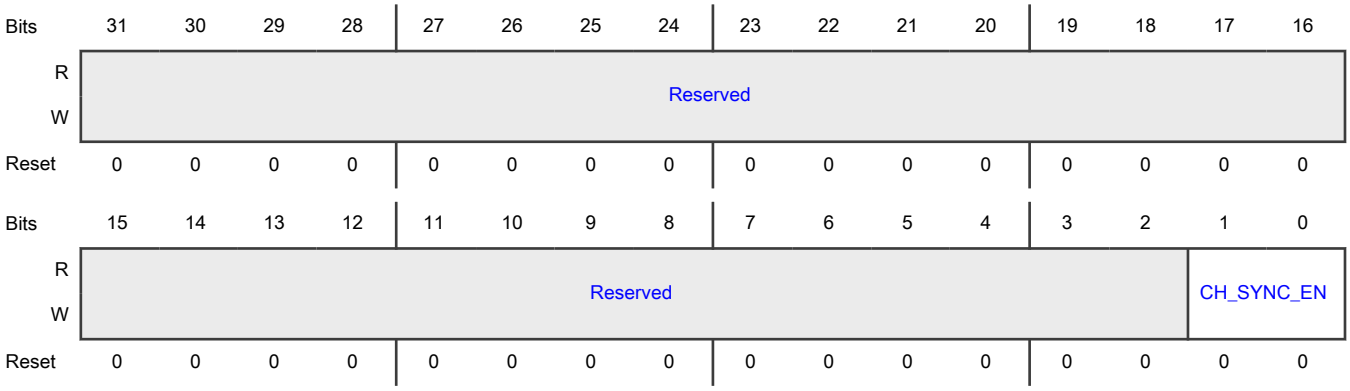
46.6.1.1.13 Global Channel Synchronization Enable (GLOBAL_SYNC_EN)

This register chooses which channels to synchronize to global sync.

Offset

Register	Offset
GLOBAL_SYNC_EN	F14h

Diagram



Fields

Field	Description
31-2 —	Reserved Read value is undefined, only zero should be written.
1-0 CH_SYNC_EN	Channel synch enable Each bit controls the corresponding DMIC channel. Note that the channels should always work in pairs, if both channels in a decimator pair (for instance channels 0 and 1) are used, the same choice should be made for the both of them.

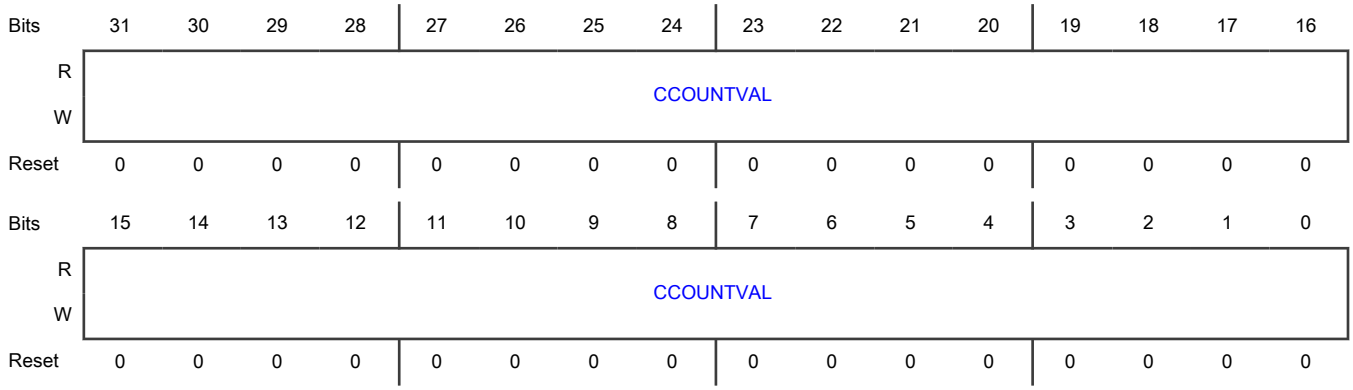
46.6.1.1.14 Global channel synchronization counter value (GLOBAL_COUNT_VAL)

This register determines when global channel synchronization occurs.

Offset

Register	Offset
GLOBAL_COUNT_VAL	F18h

Diagram



Fields

Field	Description
31-0 CCOUNTVAL	Channel Counter Value The global sync counter will trigger a pulse whenever count reaches CCOUNTVAL. If CCOUNTVAL is set to 0, there will be a pulse on every cycle.

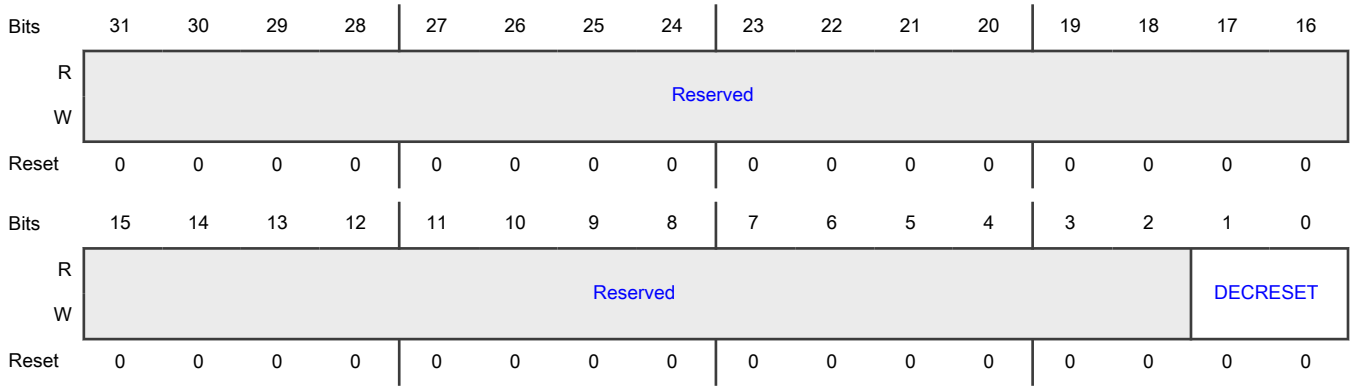
46.6.1.1.15 DMIC decimator reset (DECRESET)

This register allows resetting DMIC decimators.

Offset

Register	Offset
DECRESET	F1Ch

Diagram



Fields

Field	Description
31-2 —	Reserved
1-0 DECRESET	Decimator reset Allows resetting DMIC decimators. Resets are applied in pairs. Bit 0 controls the decimator for channels 0 and 1. Bits 1 to 7 are not used. 00 - Disable. Release reset to decimator 01 - Enable. Assert reset to decimator

46.6.1.1.16 HWVAD Input Gain (HWVADGAIN)

The HWVADGAIN register controls the input gain of the HWVAD.

Offset

Register	Offset
HWVADGAIN	F80h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	Reserved												INPUTGAIN			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Fields

Field	Description
31-4 —	Reserved Read value is undefined, only zero should be written.
3-0	Input Gain Shift value for input bits

Table continues on the next page...

Table continued from the previous page...

Field	Description
INPUTGAIN	0000 - -10 bits 0001 - -8 bits 0010 - -6 bits 0011 - -4 bits 0100 - -2 bits 0101 - 0 bits (default) 0110 - +2 bits 0111 - +4 bits 1000 - +6 bits 1001 - +8 bits 1010 - +10 bits 1011 - +12 bits 1100 - +14 bits 1101-1111 - Reserved

46.6.1.1.17 HWVAD Filter Control (HWVADHPFS)

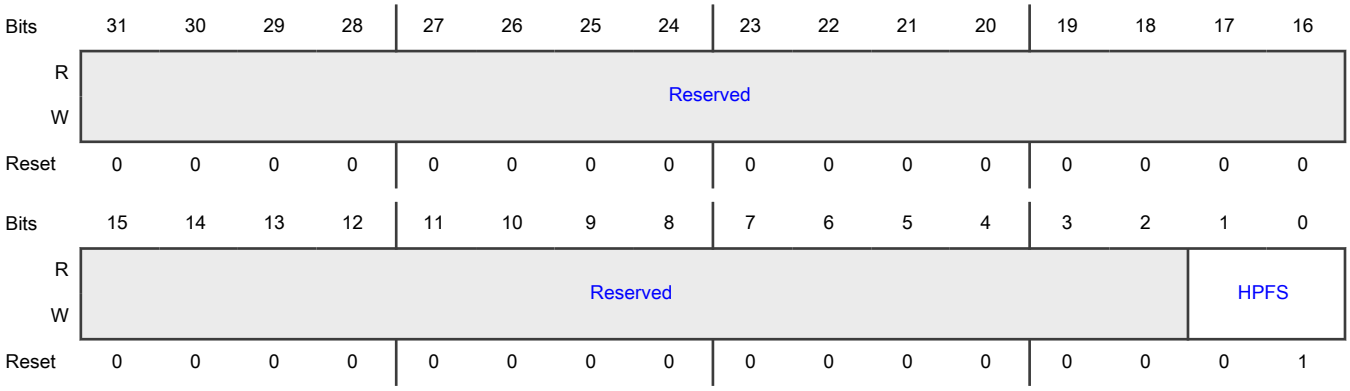
This HPFS filter setting parameter can be used to optimize performance for different background noise situations. To find the best setting, software needs to perform a rough spectral analysis of the audio signal.

Rule of thumb: If the amount of low-frequency content in the background noise is small, then HPFS = 0x2, else HPRS = 0x1.

Offset

Register	Offset
HWVADHPFS	F84h

Diagram



Fields

Field	Description
31-2 —	Reserved Read value is undefined, only zero should be written.
1-0 HPFS	The HPFS field chooses the High Pass filter in first part of HWVAD. 00 - Bypass. BYPASS. First filter by-pass. 01 - High Pass 1750 Hz. HIGH_PASS_1750HZ. High pass filter with -3dB cut-off at 1750 Hz. 10 - High Pass 215 Hz. HIGH_PASS_215HZ. High pass filter with -3dB cut-off at 215 Hz. 11 - Reserved

46.6.1.1.18 HWVAD Control (HWVADST10)

The HWVADST10 register controls the operation of the filter block and resets the internal interrupt flag. After the HWVAD triggers an interrupt, a short '1' pulse on bit ST10 clears the interrupt.

Keeping the ST10 bit set to 1 for some time has a special function for filter convergence, see more information in HWVAD

Offset

Register	Offset
HWVADST10	F88h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R 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 W	Reserved															ST10
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-1 —	Reserved Read value is undefined, only zero should be written.
0	STAGE 1

Table continues on the next page...

Table continued from the previous page...

Field	Description
ST10	'1' means enter stage 1 of VAD, that is, a sound change has been detected and the HWVAD is being allowed to settle. Use 0 when changing back to detection mode. Allow several milliseconds in stage 1 for settling. 0 - Normal operation, waiting for HWVAD trigger event (stage 0). 1 - Reset internal interrupt flag by writing a '1' (stage 1) pulse.

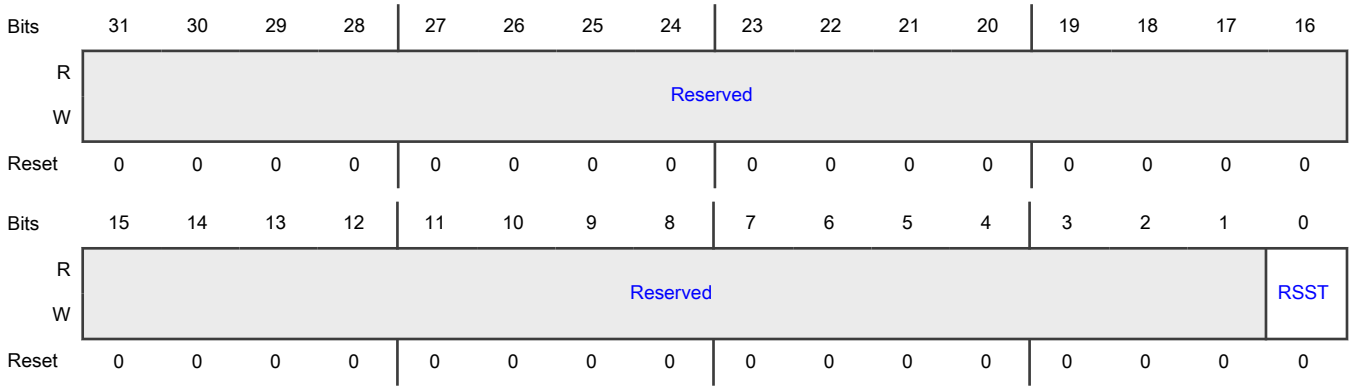
46.6.1.1.19 HWVAD Filter Reset (HWVADRSTT)

Setting bit RSTT = 1 causes a synchronous reset of all filters inside the HWVAD. The RSTT bit must be cleared to allow HWVAD operation. See HWVAD

Offset

Register	Offset
HWVADRSTT	F8Ch

Diagram



Fields

Field	Description
31-1 —	Reserved Read value is undefined, only zero should be written.
0 RSST	Reset HWVAD Write back to 0 to release reset.

46.6.1.1.20 HWVAD Noise Estimator Gain (HWVADTHGN)

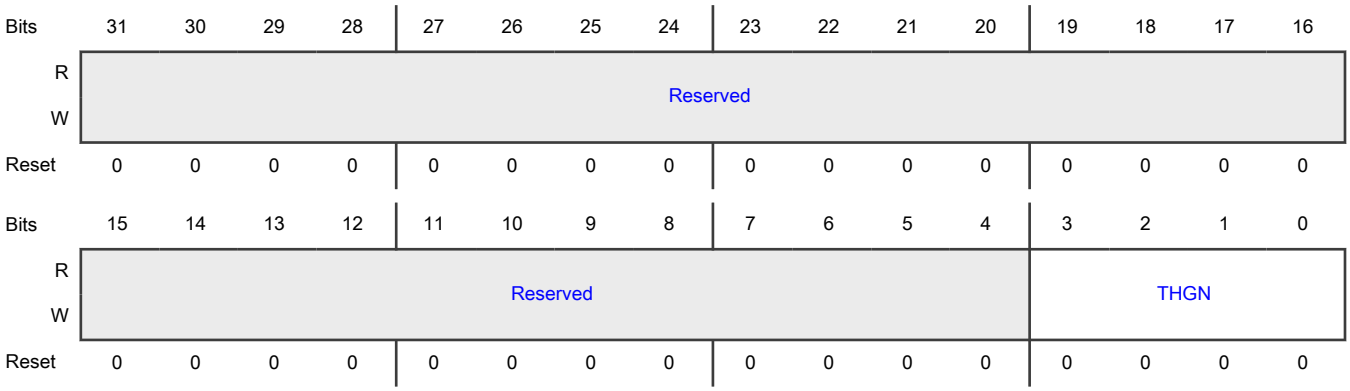
The HWVADTHGN register shows the gain value for the noise estimator value; the THGN parameter is used in the following calculation (implemented in hardware):

```
if z8 * (THGS+1) > z7 * (THGN+1)
    HWVAD_RESULT = 1;
else
    HWVAD_RESULT = 0;
```

Offset

Register	Offset
HWVADTHGN	F90h

Diagram



Fields

Field	Description
31-4 —	Reserved Read value is undefined, only zero should be written.
3-0 THGN	Gain Factor for Noise Estimator 0 to 14: 0 corresponds to a gain of 1.

46.6.1.1.21 HWVAD Signal Estimator Gain (HWVADTHGS)

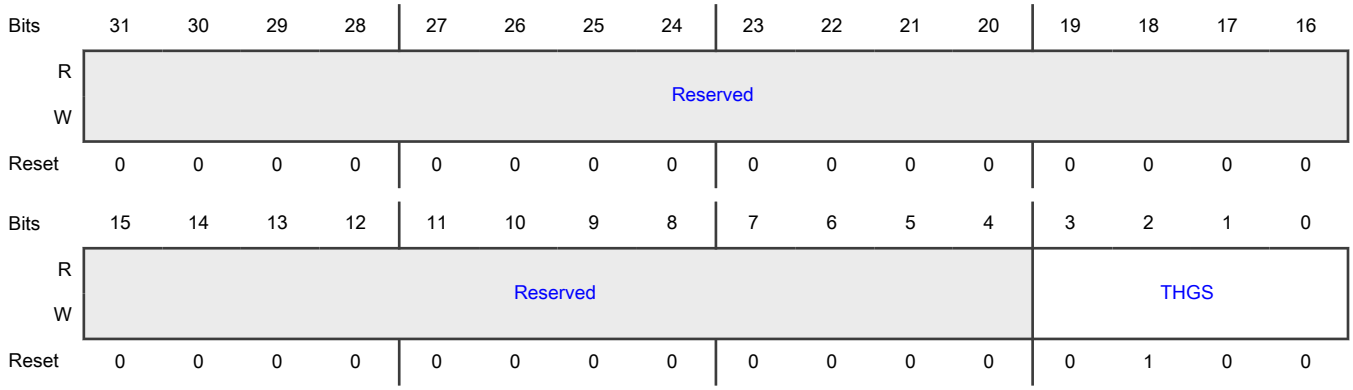
The HWVADTHGS register shows the gain value for the signal estimator value; the THGS parameter used in the following calculation (implemented in hardware):

```
if z8 * (THGS+1) > z7 * (THGN+1)
    HWVAD_RESULT = 1;
else
    HWVAD_RESULT = 0;
```

Offset

Register	Offset
HWVADTHGS	F94h

Diagram



Fields

Field	Description
31-4 —	Reserved Read value is undefined, only zero should be written.
3-0 THGS	Signal Gain Factor 0 to 14: 0 corresponds to a gain of 1.

46.6.1.1.22 HWVAD Noise Envelope Estimator (HWVADLOWZ)

The HWVADLOWZ register contains 2 bytes of the output of filter stage z7. It can be used as an indication for the noise floor and must be evaluated by software. See HWVAD

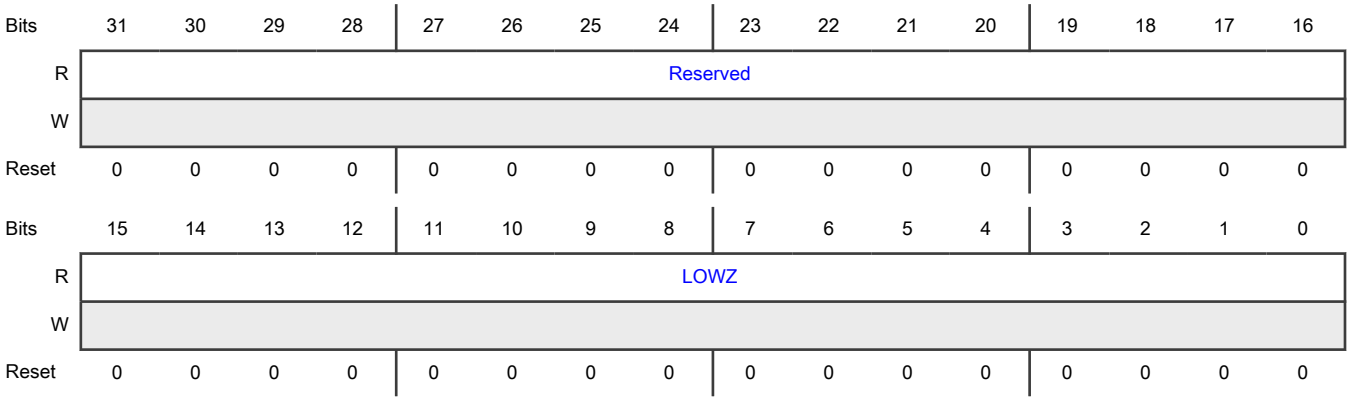
NOTE

For power saving reasons the HWVADLOWZ register is not synchronized to the AHB bus clock domain. To ensure correct data is read, the HWVADLOWZ register should be read twice. If the data is the same, then the data is correct, if not, the register should be read one more time. The noise floor is a slowly moving calculation, so several reads in a row can guarantee that register value being read is not be in the middle of a transition.

Offset

Register	Offset
HWVADLOWZ	F98h

Diagram



Fields

Field	Description
31-16 —	Reserved Read value is undefined, only zero should be written.
15-0 LOWZ	Average Noise-floor Value

Chapter 47
Analog-to-Digital Converter (ADC)

47.1 Chip-specific ADC information

Table 388. Reference links to related information

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

47.1.1 ADC subsystem Introduction

This family of devices contain various analog modules. These modules are internally connected to each other which helps to implement applications. This family of devices contains following analog peripherals.

- Two 16-bit, 2.0 Msamples/sec (two 12-bit 3.3 Msamples/sec) with eight differential channel pairs, (or 16 single-ended channels), with multiple internal and external trigger inputs and sample rates of up to 2.0 MSamples/sec. The ADC supports four independent simultaneous conversions sequences.
- Integrated temperature sensor connected to both the ADCs.
- One comparator in always-on domain with up to four input pins and internal reference voltage. Can be used as a wake-up source from low power modes.
- Three High Speed Comparators with up to five input pins and internal reference voltage.
- Three 12-bit DACs with sample rates of up to 1.0 MSample/sec.
- Three OpAmps with programmable VREF.
- For proper ADC operation, the Low Power Bandgap enable (bit 1, LPBGEN) in the VREF Control and Status Register (CSR) must be enabled prior to using the ADC peripheral.

47.1.2 ADC subsytem and connectivity

The following figure shows the Analog subsytem and connectivity for the chip.

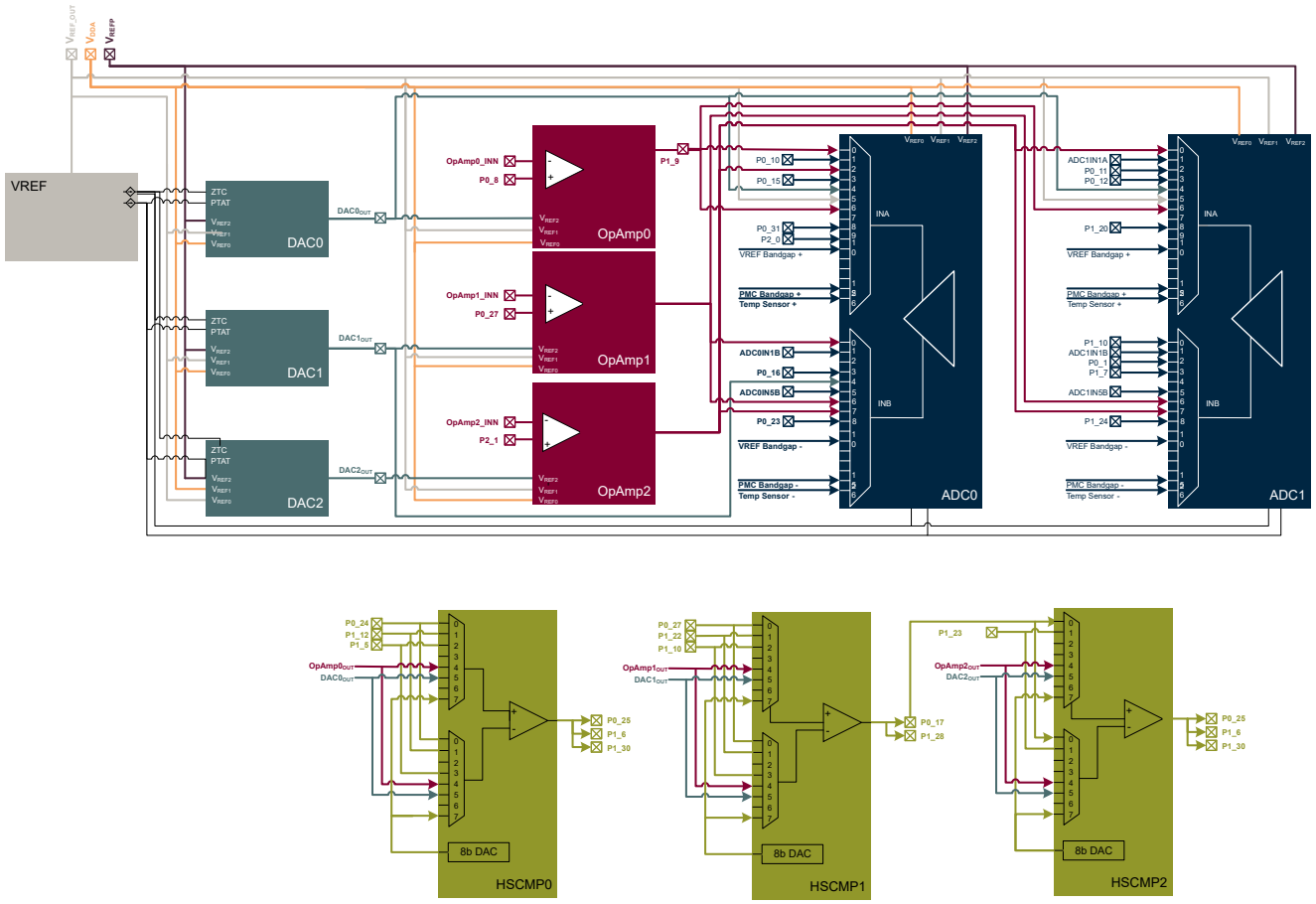


Figure 265. Analog subsystem connectivity

47.1.3 Module instances

This device has two instances of the ADC module, ADC0 and ADC1.

47.1.4 ADC input connections

Table 389. ADC number of channels

ADC	QFN48	QFP64	QFP100
Single Ended	10	13	23
Differential	4	5	9

Table 390. ADC analog channel input connections

ADC Channel	Analog Channel Input	QFN48	QFP64 AFE	QFP100	Input Type	ADC ch muxed	IO
ADC0 Channel A							

Table continues on the next page...

Table 390. ADC analog channel input connections (continued)

ADC Channel	Analog Channel Input	QFN48	QFP64 AFE	QFP100	Input Type	ADC ch muxed	IO
0	ADC0IN0A OPAMP0_OUT			25	Fast/Muxed/ Internal		PIO1_9
1	ADC0IN1A	11	14	21	Fast/Muxed		PIO0_10
2	OPAMP2_OUT				Internal	ADC0IN2B ADC1IN0A	
3	ADC0IN3A	12	15	22	Muxed		PIO0_15
4	ADC0IN4A DAC0_OUT			31	Standard/ Dedicated	ADC1IN4A	
5	ADC0IN5A VREF_OUT	7	9	14	Standard/ Dedicated	ADC1IN5A	
6	OPAMP0_VREFH_ADC				Internal		
7	OPAMP2_VREFH_ADC				Internal		
8	ADC0IN8A		16	23	Standard/ Muxed		PIO0_31
9	ADC0IN9A			24	Standard/ Muxed		PIO2_0
10	VREF Bandgap+				Internal		
26	Temperature+				Internal		
27	PMC Bandgap+				Internal		
ADC0 Channel B							
0	OPAMP1_OUT				Internal		
2	OPAMP2_OUT				Internal	ADC0IN2A ADC1IN0A	
3	ADC0IN3B	10	13	20	Fast/Muxed		PIO0_16
4	ADC0IN4B DAC1_OUT			30	Fast/Muxed		PIO1_19

Table continues on the next page...

Table 390. ADC analog channel input connections (continued)

ADC Channel	Analog Channel Input	QFN48	QFP64 AFE	QFP100	Input Type	ADC ch muxed	IO
5	ADC0IN5B VREFN	9	11	16	Standard/ Dedicated	ADC1IN5B	
6	OPAMP1_VREFH_ADC				Internal		
7	OPAMP2_VREFH_ADC				Internal		
8	ADC0IN8B			19	Standard/ Muxed		PIO0_23
10	VREF Bandgap-				Internal (VSSA)		
26	Temperature-				Internal		
27	PMC Bandgap-				Internal (VSS)		
ADC1 Channel A							
0	OPAMP2_OUT				Internal	ADC0IN2A ADC0IN2B	
1	ADC1IN1A		6	8	Fast/Dedicated		
2	ADC1IN2A	4	7	9	Fast/Muxed		PIO0_11
3	ADC1IN3A			10	Fast/Muxed		PIO0_12
4	ADC1IN4A DAC0_OUT			31	Standard/ Dedicated	ADC0IN4A	
5	ADC1IN5A VREF_OUT	7	9	14	Standard/ Dedicated	ADC0IN5A	
6	OPAMP0_VREFH_ADC				Internal		
8	ADC1IN8A			11	Standard/ Muxed		PIO1_20
10	VREF Bandgap+				Internal		
26	Temperature+				Internal		
27	PMC Bandgap+				Internal		
ADC1 Channel B							

Table continues on the next page...

Table 390. ADC analog channel input connections (continued)

ADC Channel	Analog Channel Input	QFN48	QFP64 AFE	QFP100	Input Type	ADC ch muxed	IO
0	ADC1IN0B	3	4	6	Fast/Muxed		PIO1_0
1	ADC1IN1B		5	7	Fast/Dedicated		
2	ADC1IN2B	2	3	5	Fast/Muxed		PIO0_1
3	ADC1IN3B			4	Fast/Muxed		PIO1_7
4	DAC2_OUT				Internal		
5	ADC1IN5B VREFN	9	11	16	Standard/ Dedicated	ADC0IN5B	
6	OPAMP1_VREFH_ADC				Internal		
7	OPAMP2_VREFH_ADC				Internal		
8	ADC1IN8B			3	Standard/ Muxed		PIO1_24
10	VREF Bandgap-				Internal (VSSA)		
26	Temperature-				Internal		
27	PMC Bandgap-				Internal (VSS)		

47.1.5 ADC voltage reference options

Table 391. ADC voltage reference selection

REFSEL[7:6]	Description
0	VREFH = voltage on VREFP pin
1	VREF_OUT – 1.0v to 2.1v Vref-module
2	VREFH = voltage on VDDA pin
3	Reserved

47.1.6 ADC trigger inputs

There are four trigger sources for each ADC module. ADC Trigger sources get routed through the Input Multiplexer (INPUTMUX). See the INPUTMUX chapter for available trigger sources. See registers ADC0_TRIG[0-3] and ADC1_TRIG[0-3].

47.2 Overview

The 16-bit analog-to-digital converter (ADC) is a dual successive approximation ADC designed for operation within an integrated microcontroller system-on-a-chip.

NOTE

For chip-specific modes of operation, see the power management information for the device.

47.2.1 Block diagram

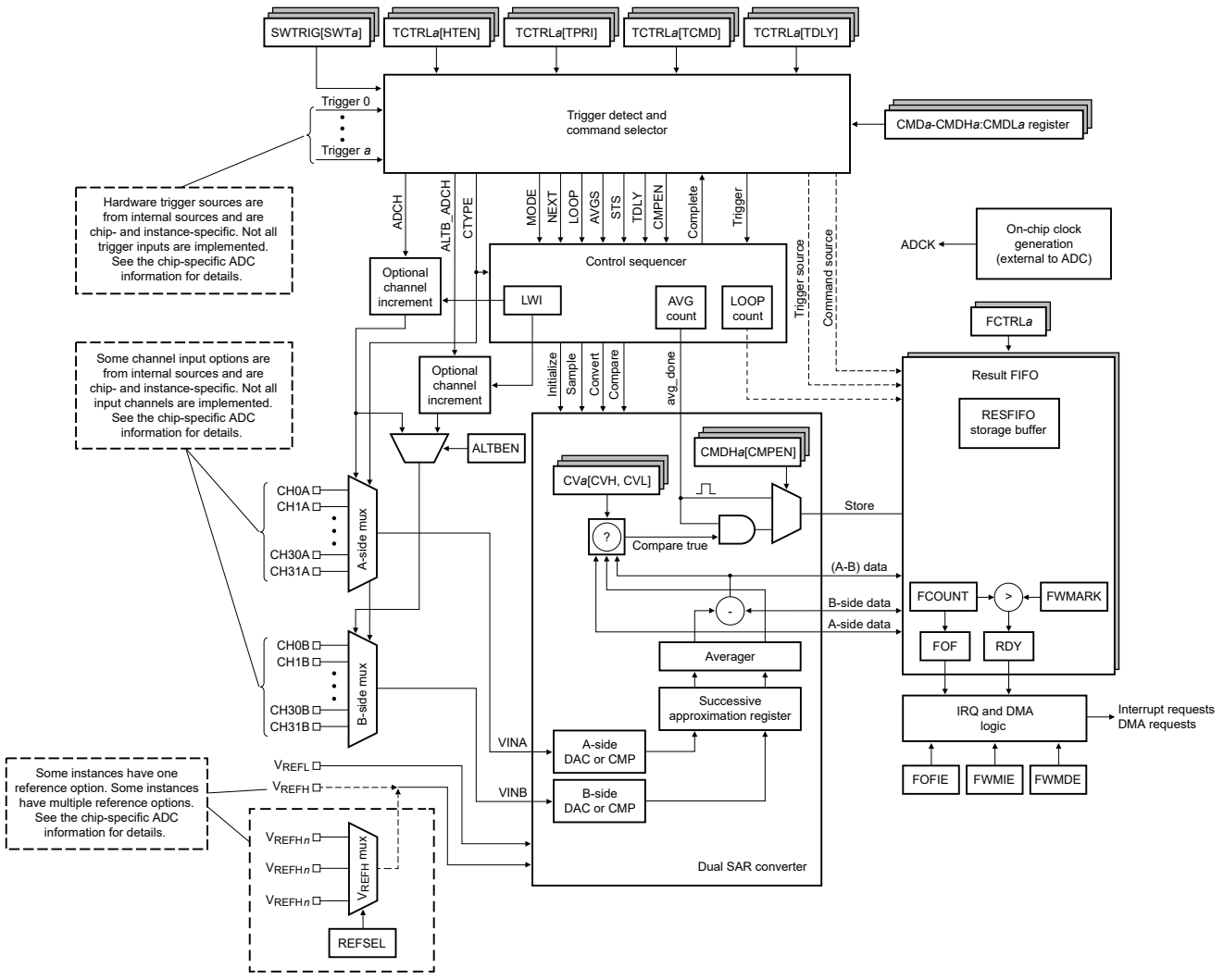


Figure 266. ADC block diagram

47.2.2 Features

ADC has these features:

- Linear successive approximation algorithm
  - Differential operation with 16-bit or 13-bit resolution
  - Single-ended operation with 16-bit or 12-bit resolution
  - Support for two simultaneous single-ended conversions
- Configurable analog input sample time
- Configurable speed options to accommodate operation in low-power modes of SoC
- Trigger detection with up to 4 trigger sources with priority level configuration. Software or hardware trigger option for each.

- 15 command buffers, to allow independent options selection and channel sequence scanning
- Automatic comparisons for less-than, greater-than, within range, or out-of-range with "store on true" and "repeat until true" options
- 2 independent result FIFOs, each containing 16 entries. Each FIFO has configurable watermark and overflow detection.
- Interrupt, DMA, or polled operation
- Linearity and gain adjustment calibration logic

47.3 Functional description

ADC performs analog-to-digital conversions on any of the software-selectable analog input channels via a successive approximation algorithm.

The ADC module can average the result of multiple conversions on a channel before storing the calculated result. The hardware average function is enabled by setting CMDHn[AVGS] to a non-zero value. The function operates in any conversion mode or configuration.

ADC can compare the result of a conversion with the contents of two value registers for less-than, greater-than, inside-range, or outside-range detection. The compare function operates in any conversion mode or configuration.

When the conversion and averaging loops finish, the resulting data is placed in one of 2 available FIFO data buffers . The data includes tag information associated with the result. When the number of stored data words exceeds the setting, a configurable watermark level supports interrupts or DMA requests. Interrupts can also be enabled to indicate when FIFO overflow errors occur.

The module initializes to its lowest power state during reset.

The ADC analog circuits can be pre-enabled to begin conversions sooner at the expense of higher idle currents. Conversions are initiated by selectable trigger events from software or hardware sources.

The trigger-detection logic includes a configurable enable and priority scheme for available trigger sources.

ADC includes multiple command buffers to provide flexibility for channel scanning and independent channel selections for different trigger sources. These command buffers can be configured for:

- Differential or single-ended conversion
- Sample time
- Averaging on a per-channel basis

ADC includes offset and linearity calibration logic. A request for calibration should be made upon reset or power up. Each successive approximation register (SAR) conversion uses calibration data calculated during the calibration routine.

The sequencing of an ADC command is summarized in the flow diagram below.

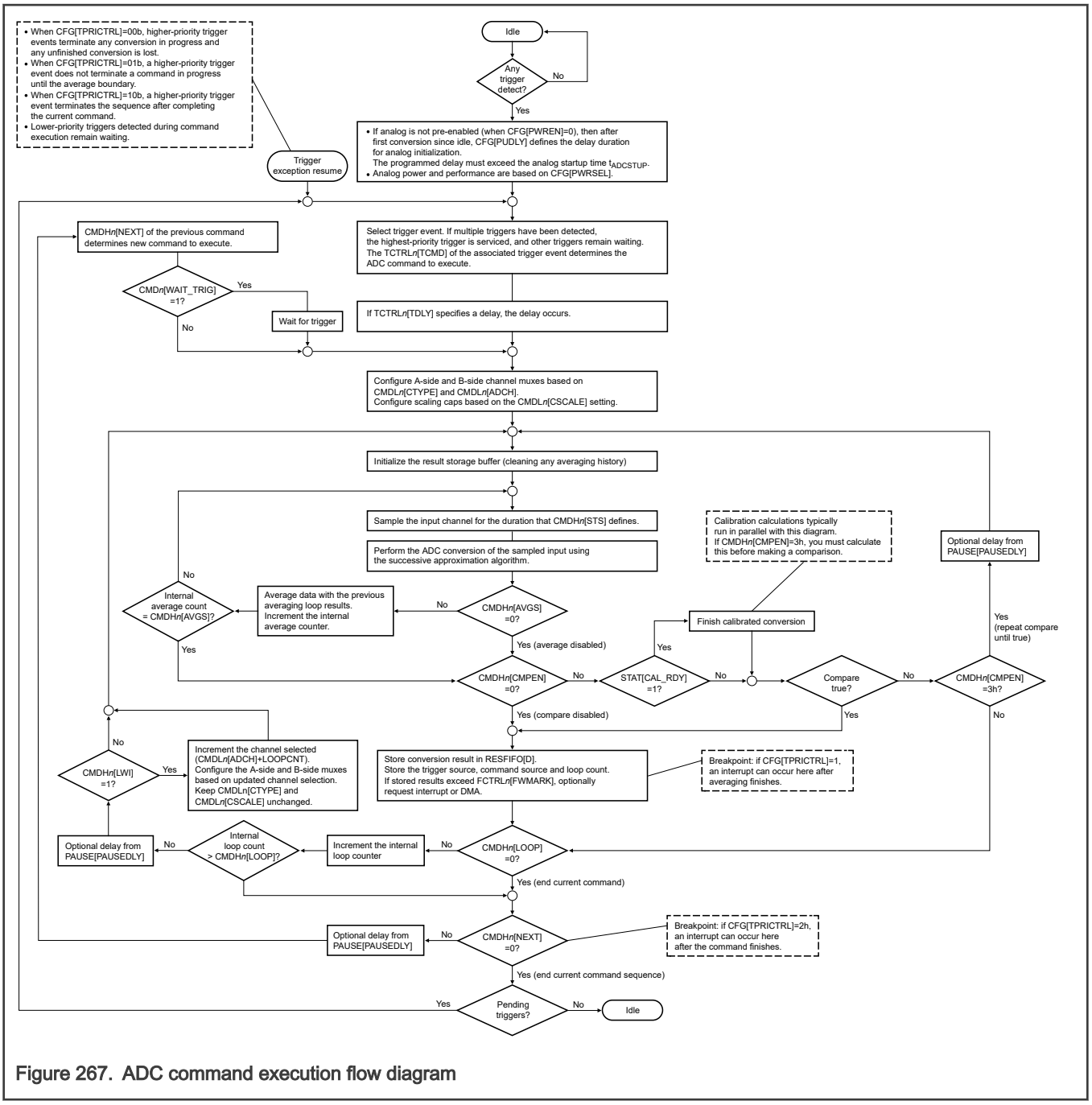


Figure 267. ADC command execution flow diagram

47.3.1 Power control mode

By default, the ADC analog circuits are disabled while ADC is in its idle state. When a trigger is detected and ADC command processing is initiated, the analog circuits are enabled. These circuits require a period of initialization before the first conversion cycle.

The value of CFG[PUDLY] should be set to incur a delay longer than $t_{ADCSTUP}$. The accuracy of initial conversions after activation is degraded if the value of CFG[PUDLY] is too low.

You can achieve faster conversion startup times by setting CFG[PWREN] to pre-enable the analog circuits of ADC. This faster conversion consumes extra power, even while ADC is in an idle state. When CFG[PWREN] is 1, the Power Enable timer is activated. The timer enforces the minimum time required (configured by CFG[PUDLY]) before detected triggers can initiate ADC conversions.

ADC also has options for controlling power and performance summarized in the table below. See the device data sheet for specification of the available power modes.

Table 392. Power option settings

CFG[PWRSEL]	Description
0xb (Default)	Slow speed and low power
1xb	Fast speed and high power

47.3.2 ADC modes of operation

The ADC module supports the chip low power modes described in the following table. See section Clock operation for more information.

Table 393. Chip modes supported by the ADC module

Chip mode	ADC Operation
Run	Normal operation
Deepsleep/Sleep	When the Doze Enable bit (CTRL[DOZEN]) is 0, the ADC can continue to operate and the module is using an external or internal clock source which remains operating during Deepsleep/Sleep modes. ADC is allowed to continue operation while the system is transitioned to the low power state. Any conversion in progress is not disrupted. External hardware trigger detect and active conversions are operational. When the DOZEN bit is 1, the module waits for the current averaging iteration/FIFO storage to complete before acknowledging Deepsleep or Sleep mode entry. There is no associated ack/handshake and the system transitions to low power state without ADC interaction. ADC operation should terminate after completion of any conversion in progress.
Deep Powerdown	The Doze Enable (CTRL[DOZEN]) bit is ignored and the module waits for the current transfer to complete any pending operation before acknowledging Deep Powerdown mode entry.

47.3.3 Voltage reference

The voltage reference high (V_{REFH}) used by ADC is supplied from either an on-chip voltage reference source or from an off-chip source via external pins. V_{REFL} is always from an external pin and must be at the same voltage as V_{SSA}.

This block supports a programmable selection of the Voltage Reference High used for ADC conversions (via CFG[REFSEL]).

See the chip configuration information on the voltage reference options specific to this packaged device.

47.3.4 Trigger detect and command execution

See Figure 267 for a flow diagram of command execution sequencing.

ADC command execution is initiated from up to 4 trigger sources. Each trigger can be software-generated by writing 1b to the corresponding SWTRIG[SWTn] field. Alternatively, asynchronous input sources at the periphery of the module can generate hardware triggers. The number and sources of hardware triggers implemented is device-specific. See the chip-specific ADC information for descriptions of available hardware trigger sources for this device.

Each hardware trigger source is enabled by setting the associated enable field (TCTRLn[HTEN]). Each trigger source is assigned a priority via the associated priority control field (TCTRLn[TPRI]). Each of the trigger sources is associated with a command buffer via the associated command select field (TCTRLn[TCMD]).

When a hardware trigger input is enabled, hardware trigger events are detected on the rising edge of the associated hardware trigger source.

Each trigger source has an associated priority field TCTRLn[TPRI] which allows arbitration between trigger sources. Arbitration selects two things: which trigger sequence to execute next, and how to handle a trigger exception. Trigger exceptions are defined as allowing a higher-priority trigger sequence to interrupt operation of a lower-priority sequence. When a trigger exception occurs, programmable arbitration allows the configurable stop and resume points for low-priority sequences. The fields affecting arbitration are CFG[HPT_EXDI], CFG[TCMDRES], CFG[TRES], and CFG[TPRICTRL].

1. When CFG[HPT_EXDI] = 1b, trigger exceptions are disabled and any higher priority triggers are left pending until the current sequence completes. New triggers are accepted based on priority.
2. When CFG[HPT_EXDI] = 0b (default), exceptions are enabled and the higher-priority sequence begins executing at a user-specified breakpoint.

Breakpoint locations are determined by CFG[TPRICTRL], which affects latency for accepting trigger exceptions.

1. When CFG[TPRICTRL] = 0h, a higher-priority trigger causes an immediate command abort and the new command specified by the trigger is immediately started.
2. When CFG[TPRICTRL] = 1h, the current conversion is allowed to complete (including averaging) before the higher-priority exception starts. In this mode, if the command is running through a series of averages, this series completes. However, there is no requirement to finish the entire command before being interrupted. For example, if the command consists of four loop iterations, there is no requirement to complete all four iterations before the interrupt occurs.
3. When CFG[TPRICTRL] = 2h, a higher-priority trigger begins once the current command is completed. If a command consists of five loop iterations each containing eight averages, then all 40 conversions must be completed before accepting the trigger exception.

CFG[TCMDRES] and CFG[TRES] determine what ADC does after accepting a trigger exception. The module can be programmed to resume commands after returning from a trigger exception.

1. When CFG[TRES] = 0h, commands are not automatically resumed after being stopped by an exception. However, an interrupt is set to indicate that this case has occurred. The flag TSTAT[TEXC_NUM] can be used to resolve which trigger was stopped by the exception.
2. When CFG[TRES] = 1h, ADC automatically resumes commands after they were stopped by an exception.

Using CFG[TRES] with CFG[TCMDRES], the module can be programmed to resume commands at one of two possible locations.

1. When CFG[TCMDRES] = 0h, the trigger which was stopped by an exception is resumed from the beginning of its associated command sequence. Triggers that are waiting to be resumed take the same priority programmed to TCTRLn[TPRI].
2. When CFG[TCMDRES] = 1h, the trigger is resumed from the command that it was executing before being interrupted by an exception.

If a trigger occurs with priority lower than the priority of the currently executing command, trigger detection is left pending until the current command sequence finishes. Lower-priority trigger events cannot be serviced until a higher-priority triggered command (or command sequence) completes.

When a conversion is completed (including hardware averaging when CMDHn[AVGS] is non-zero), the result is placed in a RESFIFO buffer. When an ADC command selects looping (when CMDHn[LOOP] is non-zero) a command stores multiple conversion results to the FIFO during execution of that command.

At the end of command execution, CMDHn[NEXT] selects the next command to be executed. Multiple commands can be executed sequentially by configuring the CMDHn[NEXT] field of each command. Setting the next command to 0h causes conversions to terminate at the completion of the current command. Unending circular command execution is allowed by setting the CMDHn[NEXT] field of the last command in a sequence to the first command in the sequence.

By default, command sequences execute automatically in the order in which CMDHn[NEXT] fields are programmed. However, by using CMDHn[WAIT_TRIG], command execution can be stalled and launched based on trigger inputs. For example, if TRIGGER2 is programmed to start the command sequence CMD1, CMD2, CMD3, then this sequence is run once unconditionally to completion upon receiving TRIGGER2. If CMDH2[WAIT_TRIG] = 1h, however, the sequence pauses after CMD1 until TRIGGER2 is received again. In this way, sequences can be stalled until a trigger assertion is received.

Disabling ADC by writing 0b to CTRL[ADCEN] terminates any active ADC command processing. Writing 0b to CTRL[ADCEN] causes the current command (or command sequence) to terminate, clears any pending triggers, and sends the ADC module to an idle state.

47.3.5 Pause option

When an application does not require the maximum conversion rate, the effective conversion rate can be reduced:

- By implementing periodic trigger events to initiate ADC conversions, or
- By selecting a reduced-frequency clock as the ADCK source.

Both options are chip-specific and are dependent on ADC triggering and clocking options external to the ADC module. The latency associated with ADC analog power-up delays limits the maximum conversion rate when using periodic triggering.

Conversion rates can also be reduced by inserting a pause of a programmable duration:

- Between loop iterations
- Between commands in a sequence
- Between conversions, when a command is executing in the Compare Until True configuration

When PAUSE[PAUSEEN] = 1, PAUSE[PAUSEDLY] controls the duration of pausing during command execution sequencing. The pause delay is a count of (PAUSE[PAUSEDLY] * 4) ADCK cycles.

NOTE

Do not change the PAUSE register while CTRL[ADCEN] = 1. Writes to the PAUSE register while CTRL[ADCEN] is 1 can lead to metastable operation.

See Figure 267 for the places during command execution sequencing where the pause can be inserted.

47.3.6 Resync functionality

Any software or hardware trigger source can be configured to act as a resync trigger. Trigger-based resync interrupts a running trigger (resync target) and clears the FIFO to which the trigger is writing. Resync can be used either to abort a running sequence, or to restart a running sequence depending on the value of CFG[TRES]. If CFG[TRES] = 1, the target sequence is aborted, the FIFO is cleared, and the sequence is restarted after the resync occurs. If CFG[TRES] = 0, the target sequence is aborted and the FIFO is cleared after the resync occurs. The FIFOs that are cleared are based on the resync targets TCTRLn[FIFO_SEL_A] and TCTRLn[FIFO_SEL_B]. To only clear one FIFO, set TCTRLn[FIFO_SEL_A] = TCTRLn[FIFO_SEL_B]. When the resync occurs, any results are stored in TCTRLn[FIFO_SEL_A] or TCTRLn[FIFO_SEL_B] that are not associated with the resync target are lost.

A resync trigger must have a specific target. The resync only occurs if the resync target is running at the time of the trigger. In the following example, n is the resync trigger number and m is the resync target number. According to these variables, trigger n should resync trigger m. To enable a trigger source to act as a resync trigger, these conditions must be satisfied:

1. The resync trigger TCTRLn[RSYNC] must be set to 1.
2. The resync trigger must have higher priority than the resync target (TCTRLn[TPRI] must be less than TCTRLm[TPRI]).
3. The resync target is specified using TCTRLn[TCMD]. In this case the resync target, m, must be equal to TCTRLn[TCMD].
4. The resync target, m, must be executing commands when the resync trigger, n, is asserted.
5. Trigger m must have at least one conversion remaining to start when trigger n is received.

If a trigger source n has TCTRLn[RSYNC] set to 1, but some of the above conditions are not met, the trigger source n is ignored.

The following figure illustrates a resync trigger sequence. In this example, trigger source 1 is configured to resync trigger source 0.

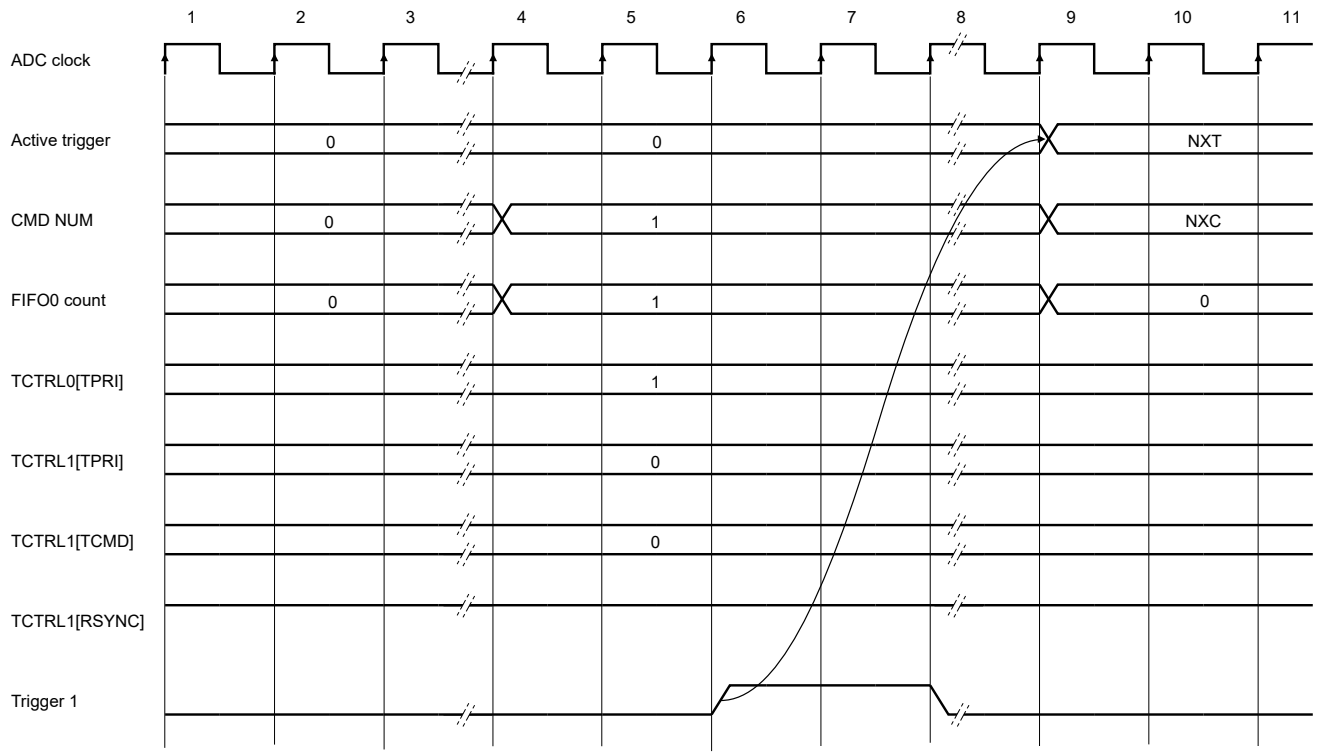


Figure 268. ADC Resync Example

In this figure, trigger source 0 is executing a sequence of commands when trigger source 1 is asserted. Trigger source 0 is stopped when trigger source 1 is asserted after some synchronization delay. In addition, the FIFO being written to by trigger source 0 is cleared (FIFO0 in this example). After the trigger 0 sequence is stopped, ADC runs the next trigger pending with the highest priority. This trigger is marked as NXT, for next trigger. If resume functionality is enabled, and trigger source 0 has the highest priority pending, then NXT = 00h.

47.3.7 Temperature sensor

ADC has a dedicated input channel for an on-chip temperature sensor. See the chip-specific channel definition to determine which channel is connected to the on-chip temperature sensor.

To calculate the temperature, you must first execute a conversion of the temperature sensor channel with configuration requirements. The sequence to convert the temperature sensor channel is:

1. Configure a command buffer register to convert the temperature sensor channel (CMDLn[ADCH]).
2. Program the command buffer with the following parameters:
   - Differential: CMDLn[CTYPE] = 2h
   - Max averaging: CMDHn[AVGS] = 7h
   - Max sample time: CMDHn[STS] = 7h
   - LOOP set to 1: CMDHn[LOOP] = 1h
   - Loop with increment disabled: CMDHn[LWI] = 0h
   - Compare function disabled: CMDHn[CMPEN] = 0h
3. Configure a trigger control register TCTRLn with the following parameters:
   - TCTRLn[TCMD] = command buffer used in steps 2 and 3.

- TCTRLn[FIFO_SEL_A] directs conversion results to FIFO0 orFIFO1

4. Trigger a conversion of the temperature sensor channel. You trigger the conversion by writing 1 to the associated bit in Software Trigger Register (SWTRIG) (that is, the trigger configured in the previous step).

After completing the conversion of the temperature sensor channel, two results are stored in the FIFO selected by TCTRLn[FIFO_SEL_A]. Each result corresponds to a component of the overall temperature value. Read these values from the FIFO and use the following equation to calculate the ambient temperature.

$$\text{Temp} = A\left[\frac{\alpha(\text{Vbe8}-\text{Vbe1})}{\text{Vbe8} + (\alpha(\text{Vbe8}-\text{Vbe1}))}\right] - B$$

Equation 2. Temperature sensor function

In the preceding equation:

- Vbe1 is the first value stored to the FIFO as a result of the temperature sensor channel conversion.
- Vbe8 is the second value stored to the FIFO as a result of the temperature sensor channel conversion.
- A is the slope factor.
- B is the offset factor.
- α is the bandgap coefficient.

A, B, and α are specified constant values from the ADC electrical information in the chip data sheet.

47.3.8 Result FIFO operation

ADC includes 2 16-entry FIFOs in which the result of ADC conversions are stored. In addition, a valid indicator bit, the trigger source, the source command, and the loop count are also stored with the data. FCTRLn[FCOUNT] indicates how many valid data words are stored in each RESFIFO.

A programmable watermark threshold supports configurable notification of data availability. When FCTRLn[FCOUNT] is greater than FCTRLn[FWMARK], the associated RDY flag is asserted. When IE[FWMIEn] = 1, a watermark interrupt request is issued. When DE[FWMDEn] = 1, a DMA request is issued. Reading RESFIFO provides the oldest unread data word entry in the FIFO and decrements FCTRLn[FCOUNT]. When FCTRLn[FCOUNT] falls equal to or below FCTRLn[FWMARK], the RDY flag is cleared.

Each FIFO can be emptied by successive reads of RESFIFOn. When RESFIFOn[VALID] is 1, the associated FIFO entry is valid. Reading RESFIFOn when the FIFO is empty (when RESFIFOn[VALID] = 0 and FCTRLn[FCOUNT] = 0h) provides an undefined data word. All FIFOs are reset by writing 1b to CTRL[RSTFIFOn].

If ADC attempts to store a data word to the FIFO when the FIFO is full, the FIFO overflow flag (FCTRLn[FOF]) is set. When IE[FOFIEn] = 1, an overflow interrupt request is issued. The FOF flag is cleared by writing 1 to STAT[FOFn]. When overflow events occur, no new data is stored and the data associated with the storage event that triggered the overflow is lost.

Conversion results can be steered to any FIFO in the design. TCTRLn[FIFO_SEL_A] and TCTRLn[FIFO_SEL_B] determine into which FIFO the final result is written. Depending on which trigger is executing, the results can be steered to different locations. Depending on the type of conversion selected, the FIFO destination register fields are interpreted differently. During either differential or single-ended mode (CMDLn[CTYPE] != 3h) only one result is produced. The destination during these modes is determined from TCTRLn[FIFO_SEL_A]. In dual-single-ended mode, both TCTRLn[FIFO_SEL_A] and TCTRLn[FIFO_SEL_B] determine the Channel A and Channel B destinations respectively.

47.3.9 Sampling Modes

The ADC module supports three different sampling modes: dual single-ended, single-ended, and differential. The sampling mode is determined by the currently executing command using CMDLn[CTYPE].

When executing a command in dual single-ended mode, two independent conversion results are calculated and stored in selectable FIFO destinations. Command processing, however, is not individually controlled for each independent channel being sampled. When operating in dual single-ended mode both channels are sampled and processed simultaneously.

The selection of Channel B is controlled by CMDLn[ALTBEN].

- When CMDLn[ALTBEN] = 1, the channel for the B-side is selected independently (via the value of CMDLn[ALTB_ADCH]).
- When CMDLn[ALTBEN] = 0, the Channel B selection is controlled by CMDLn[ADCH] and Channel A and Channel B are paired (CH0A/CH0B, CH1A/CH1B, and so on).

If comparisons are enabled in dual single-ended mode, only the A-side channels (CH0A, CH1A, and so on) are used for the comparison. The A-side comparison determines whether both the A-side and B-side results are written to the FIFOs.

- If the A-side comparison passes, both the A-side and B-side results are stored.
- If the A-side comparison fails, A-side and B-side results are not written to the FIFOs.

Single-ended mode is configurable to allow either A-side or B-side channels to be sampled. In single-ended mode, the results from each conversion can be written to a selectable FIFO using TCTRLn[FIFO_SEL_A].

In differential mode, the final SAR calculation is equivalent to V(CHA) - V(CHB). If the result is negative, then the value is stored in sign-extended two's complement format. TCTRLn[FIFO_SEL_A] also determines where differential conversion results are stored.

In dual single-ended mode, however, two independent results are produced. Individual control is provided by using TCTRLn[FIFO_SEL_A] and TCTRLn[FIFO_SEL_B] to select a FIFO destination for both results during this mode. Both single-ended results may be written to the same destination by programming TCTRLn[FIFO_SEL_A] = TCTRLn[FIFO_SEL_B]. In this case, the CH_A result is always stored before the CH_B result.

47.3.10 Compare Function

After the input is sampled and converted and any averaging iterations are performed, CMDHn[CMPEN] determines:

- Whether to use the automatic compare function, storing the conversion result when true.
- Whether to repeat the channel acquisition until the automatic compare function returns a true result.

The command-sequencing options related to the compare function are summarized in the table below.

NOTE

Latency is added to the end of a compare-until-true conversion to resolve the next command or loop in a sequence. This latency is necessary to calibrate the SAR data before resolving the result of a comparison. Delay for this feature is only added when resolving the result of a conversion. Intermediate samples during averaging do not include extra latency; only loop and command boundaries experience this delay. The latency is always less than or equal to five ADC clock cycles.

Table 394. Compare modes

CMDHn[CMPEN]	Compare function	Description
00b	Compare disabled	Do not perform compare operation. Always store the conversion result to the FIFO.
01b	Reserved	
10b	Store on true	Perform compare operation. Store conversion result to FIFO after averaging only when the result of the comparison is true. Regardless of the comparison result, the LOOP setting is considered. The LOOP counter is incremented before deciding whether the current command has completed or additional LOOP iterations are required.
11b	Repeat compare until true	Perform compare operation. Store conversion result to FIFO after averaging only when the result of the comparison is true. Once a comparison is true, the LOOP setting is considered. The LOOP counter is incremented before deciding whether the current command has completed or additional LOOP iterations are required. When a comparison is false, the conversion is repeated without considering the LOOP setting, and the LOOP counter is not incremented.

The compare operation checks the result based on the values of CVn[CVH] and CVn[CVL], as shown in the following table.

Table 395. Compare operations

CVn[CVL] vs. CVn[CVH]	Operation	Description
set CVn[CVL] < CVn[CVH]	Outside range (General form)	True if the result is less than CVn[CVL] **or** greater than CVn[CVH].
set CVn[CVH] to maximum value set CVn[CVL] to compare point	Less than	True if the result is less than CVn[CVL].
set CVn[CVL] to minimum value set CVn[CVH] to compare point	Greater than	True if the result is greater than CVn[CVH].
set CVn[CVL] > CVn[CVH]	Inside range	True if the result is less than CVn[CVL] **and** greater than CVn[CVH].

NOTE

When ADC continues operating in a low-power mode, the compare function can monitor the voltage and wake the device only when the compare condition is met.

47.3.11 Cycles per conversion

The number of ADCK cycles needed to complete a conversion, varies based on the selected resolution (CMDLa[MODE]), the sample time configured, and several other configuration options discussed below.

To calculate the cycle count, first, determine the Base Cycles Count from Table 396. Next, add the Sample Time Adder based on CMDHa[STS] setting as summarized in Table 397. If averaging is enabled (CMDHa[AVGS] does not equal to 0), then the total cycle (Base Cycles Count + Sample Time Adder) should be multiplied by the number of averages configured as summarized in Table 398.

$$CycleCount\ /\ Conversion = [(BaseCycleCount + SampleTimeAdder) * AverageMultiplier]$$

Equation 3. Cycle Count/Conversion

Note that there is latency associated with trigger detection and starting an initial conversion. There is additional latency associated with offset and gain error adjustment of a raw conversion result and storage of a final result to the FIFO. Due to the pipe-lining of conversions when looping or command chaining is configured, the next conversion is immediately started while a raw conversion result is being adjusted and stored and thus the conversion rate is maximized.

The base cycles per conversion are variable depending on CMDLa[MODE] setting as summarized in the following table:

Table 396. Base cycles per conversion

Base ADCK cycles/conversion[1]	
16-bit CMDLa[MODE] = 1	12-bit CMDLa[MODE] = 0
24	19

1. This cycle count includes min sample time of 3.5 ADCK cycles.

In addition to the base cycles per conversion, the configured sample time needs to be considered. The sample time adder is variable depending on CMDHa[STS] setting and is summarized in the following table:

Table 397. Sample time cycle adder

CMDHa[STS] (default 000)	Add ADCK cycles/conversion
000	0
001	2
010	4
011	8
100	16
101	32
110	64
111	128

Table 398. Averaging multiplier

CMDHa[AVGS] (default 000)	Averaging multiplier
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

When LOOPing is used:

CMDHa[LOOP] allows iteration of a command (that is a channel can be converted multiple times). This feature adds 3 additional cycles between conversions.

NOTE

When commands are chained using CMDHa[NEXT] to execute back-to-back conversions the ADC does not have the 3 stall cycles between commands.

For a command that executes multiple conversions of the same channel using LOOPing the total cycle count is:

$$CycleCount\ /\ Command = [(BaseCycleCount + SampleTimeAdder) * AverageMultiplier * (CMDHa[LOOP] + 1)] + (3 * CMDHa[LOOP])$$

Equation 4. Cycle Count/Command

47.3.12 Clock operation

The ADC operates from the ADCK clock input provided from an on-chip clock select block and is used by the SAR conversion control sequencing logic and the FIFO storage buffer. The ADCK frequency must be within the specified frequency range for ADCK and varies based on configuration of CFG[PWRSEL]. Refer to the device datasheet for supported frequency ranges.

The ADC continues operating in Deepsleep and Sleep modes provided the Doze Enable bit (CTRL[DOZEN]) is clear and the on-chip clock select block continues to supply an ADCK clock source. Note, in Deepsleep mode with CTRL[DOZEN] == 0b0, the bus clock can be shut off, and asynchronous interrupts and DMA requests can be configured. In addition, the ADC continues processing commands and writing data to the internal FIFO. The ADC has four sources for async interrupts during Deepsleep mode: watermark, FIFO overflow, TCOMP, and TEXC. To enable them, properly configure the bits IE[FWMIEx], IE[FOFIEx], IE[TCOMP_IE], and IE[TEXC_IE] before entering Deepsleep mode.

When the DOZEN bit is set in Deepsleep and Sleep modes, the ADC waits for the current averaging iteration/FIFO storage to complete before acknowledging Deepsleep or Sleep mode entry. Any pending triggers are dropped when a Deepsleep/Sleep mode request is made with DOZEN set. The ADC is forced into its lowest power setting after acknowledging the DOZEN Deepsleep/Sleep mode request. The same behavior is observed when entering a Deep Powerdown Mode.

47.3.13 Resets

Table 399. ADC Resets

Reset source	Description
Chip reset	The logic and registers for ADC are reset to their default states on a chip reset.
Software reset	ADC implements a software reset field in its control register. CTRL[RST] resets all logic and registers to their default states, except for the CTRL register itself.
FIFO reset	ADC implements write-only control that resets FIFO0 (CTRL[RSTFIFO0]) and FIFO1 (CTRL[RSTFIFO1]). After a FIFO is reset, that FIFO is empty.

47.3.14 Interrupts and DMA requests

ADC includes several sources for interrupts and DMA requests. The table below summarizes these sources.

A programmable watermark threshold supports configurable notification of data availability and can generate either an interrupt exception or a DMA request. When FCTRLn[COUNT] is greater than FCTRLn[FWMARK], the associated STAT[RDYn] flag is asserted. The IE[FWMIEn] and DE[FWMDEn] control fields control the masking for this exception. When STAT[RDYn] becomes 1 and IE[FWMIEn] = 1, a watermark interrupt request is issued. When STAT[RDYn] becomes 1 and DE[FWMDEn] = 1, a DMA request is issued.

The other exception sources can only generate interrupts and do not have a DMA request option. Each source has a mask control field in the IE register and no corresponding field in the DE register.

Table 400. ADC Interrupts and DMA Requests

Status Register (STAT)		Description	Can generate		
Flag	Name		Interrupt?	DMA request?	Low-power wake-up?
RDYn	Result FIFO n Ready Flag	Conversion result data is written to Result FIFO and has a watermark configurable trigger level to generate an exception request as controlled by FCTRLn[FWMARK].	Y	Y	Y
FOFn	Result FIFO n Overflow Flag	Attempting to store data to the FIFO when the FIFO is full is an error condition.	Y	N	Y
TCOMP_INT	Trigger Completion Flag	A trigger sequence has been completed. All associated commands have been run.	Y	N	Y
TEXC_INT	High Priority Trigger Flag	A high priority trigger exception has occurred.	Y	N	Y

47.4 External signals

The ADC module supports analog channel inputs with differential and single-ended conversion options for all channels. ADC requires supply and ground connections.

Table 401. ADC signal descriptions

Signal	Description	I/O
V_{DDA}	Analog Power Supply	I
V_{SSA}	Analog Ground	I
CHnA - CH0A[1]	A-side Analog Channel Inputs	I
CHnB - CH0B[1]	B-side Analog Channel Inputs	I

1. where n is the maximum channel number supported in the chip. See the chip-specific information for the number of channels supported on your device.

47.4.1 Analog channel inputs (CHnA and CHnB)

CMDLn[ADCH] and CMDLn[CTYPE] control selection of paired or individual input channels. Each ADC command independently makes a channel and conversion type selection. Each CMDLn[ADCH] channel selection has an associated A side and an associated B side input. Each CMDLn[ADCH] pair can be converted in differential mode, but only limited pairs should be converted as differential channels (for example, adjacent pins designed with matched impedance). For the pin pairings available for differential conversions for your device, see the Chip Configuration details.

NOTE

Some input channels from on-chip sources, such as temperature sensors and reference voltage sources, may only be connected to individual instances of ADC. Some input channel options in the field-setting descriptions may not be available for your device. See chip-specific information for the channels supported on this device.

47.5 Initialization

47.5.1 Calibration functions

The ADC module has multiple calibration functions that must be executed as part of ADC setup to achieve the specified accuracy. Calibration must be run after any reset and before a conversion is initiated. Before offset calibration or calibration, the user must configure the clock source and frequency of ADC for the clock source availability and needs of the application. ADC must be enabled (CTRL[ADCEN] = 1h) before a calibration function runs. If calibration is requested while ADC is actively converting, that sequence completes before starting calibration.

Averaging multiple conversions can achieve improved accuracy during the calibration routines. CTRL[CAL_AVGS] is used during a calibration routine to control how many samples are averaged together. If the application uses ADC in a wide variety of configurations, the configuration for which the highest accuracy is required should be selected. Alternatively, multiple calibrations can be done for the different configurations.

47.5.1.1 Offset calibration

The Offset Trim Register (OFSTRIM) is used to trim for ADC comparator offset voltage. The ADC supports an offset calibration function in which the OFSTRIM register is automatically updated. To perform offset calibration:

1. Configure for the desired averaging via CTRL[CAL_AVGS].
2. Initiate Offset Calibration by setting CTRL[CALOFS].
3. Poll the STAT[CAL_RDY] flag. When STAT[CAL_RDY] is asserted, the offset calibration function has completed, and the OFSTRIM register has updated.

47.5.1.2 ADC Calibration

ADC includes hardware calibration logic in which the A-side and B-side converters are calibrated for gain error and linearity error correction of the raw conversion result. GCR0[GCALR] and the CAL_GAR registers control calibration for the A-side converter. GCR1[GCALR] and the CAL_GBR registers control calibration for the B-side converter. ADC supports a calibration function in which the CAL_GAR and CAL_GBR registers are automatically updated. The calibration function also updates GCC0[GAIN_CAL] (associated with A-side calibration) and GCC1[GAIN_CAL] (associated with B-side calibration). For A-side calibration, a software calculation is required to derive GCR0[GCALR] from GCC0[GAIN_CAL]. For B-side calibration, a corresponding calculation is required to derive GCR1[GCALR] from GCC1[GAIN_CAL].

To complete calibration setup:

1. Execute Offset calibration steps. The Offset Trim Register (OFSTRIM) is used during calibration to trim for comparator offset voltage.
2. Configure the averaging via CTRL[CAL_AVGS].
3. Initiate the calibration routine by writing 1 to CTRL[CAL_REQ]. CTRL[CAL_REQ] remains 1 until the CAL routine has been accepted by the ADC. After acceptance, CTRL[CAL_REQ] automatically becomes 0.
4. The A-side calibration data is updated first. Poll the GCR0[RDY] flag. When it is asserted, the hardware controlled A-side calibration operation is complete, and CAL_GAR and GCC0[GAIN_CAL] registers are updated. The updated value in GCC0[GAIN_CAL] is required for further software processing described in the following steps.
5. Read GCC0[GAIN_CAL] and store for use in the gain_adjustment calculation.
6. Calculate the A-side gain_adjustment: (131072)/(131072-GCC0[GAIN_CAL]). GCC0[GAIN_CAL] is a 16-bit unsigned value. This calculation results in a floating point value between 1 and 2.
7. The integer value of the gain_adjustment calculation is always 1 and can be discarded. The fractional component must be stored to GCR0[GCALR]. Write the fractional component value to GCR0[GCALR].
8. Execute the same calculation for the B-side converter. Poll the GCC1[RDY] flag. When it is asserted, the hardware controlled B-side calibration operation is complete, and CAL_GBR and GCC1[GAIN_CAL] registers are updated. The updated value in GCC1[GAIN_CAL] is needed for further software processing.
9. Read the GCC1[GAIN_CAL] register and store for use in the gain_adjustment calculation.
10. Calculate the B-side gain_adjustment = (131072)/(131072-GCC1[GAIN_CAL]). Formatting for the value is the same as the A-side.
11. Like the A-side, round the fractional component of the B-side gain_adjustment to 16-bits and store it in GCR1[GCALR].
12. Once GCR0[GCALR] and GCR1[GCALR] contain the results from the gain_adjustment calculations, set the GCR0[RDY] and GCR1[RDY] flags to indicate they are valid. It is acceptable for the GCRn[GCALR] and corresponding GCRn[RDY] field to be updated on the same write cycle.

After completing the steps above, the calibration sequence is complete and the STAT[CAL_RDY] flag is set. The STAT[CAL_RDY] flag remains set until the user resets the system or requests a new calibration sequence.

When STAT[CAL_RDY] is set, ADC is configured to run in calibrated mode. Each conversion uses a combination of linearity and gain calibration results to correct SAR data. Calibration conversion latency is required to process each sample. However, due to the pipelined nature of data and control sequences, each conversion can still be initiated without experiencing this calibration delay.

47.5.1.3 Calibration General A-Side and B-Side Widths

The general calibration value registers CAL_GARn and CAL_GBRn have non-uniform widths. The following table defines the bit widths of each register.

Table 402. Calibration General Widths

Element CAL_GxR[N]	Width (Bits)
N = 00h, 20h	11
N = 01h	12
N = 02h–03h	13
N = 0x04–07h	14
N = 0x08–0Fh	15
N = 0x10–1Fh	16

These registers are typically updated automatically during the self-calibration sequence. To reduce the latency associated with ADC setup, the CAL_GxR values from a calibration sequence can be stored in non-volatile memory after an initial calibration. These values can then be written to the CAL_GxR registers via software prior to the first ADC conversion. If these registers are set to values not generated by the calibration function, the linearity error specifications may not be met.

NOTE

These values can only be written in a single access, byte accesses are not supported.

47.6 ADC register descriptions

This section describes the ADC registers.

47.6.1 ADC memory map

ADC0 base address: 400A_0000h

ADC1 base address: 400B_1000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Version ID Register (VERID)	32	See section	0200_2C0B
4	Parameter Register (PARAM)	32	RO	0F0F_1004
10	Control Register (CTRL)	32	See section	0000_0000
14	Status Register (STAT)	32	See section	0000_0000
18	Interrupt Enable Register (IE)	32	See section	0000_0000
1C	DMA Enable Register (DE)	32	See section	0000_0000
20	Configuration Register (CFG)	32	See section	0080_0000
24	Pause Register (PAUSE)	32	See section	0000_0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
34	Software Trigger Register (SWTRIG)	32	See section	0000_0000
38	Trigger Status Register (TSTAT)	32	See section	0000_0000
40	Offset Trim Register (OFSTRIM)	32	See section	0000_0000
A0 - AC	Trigger Control Register (TCTRL0 - TCTRL3)	32	See section	0000_0000
E0 - E4	FIFO Control Register (FCTRL0 - FCTRL1)	32	See section	0000_0000
F0 - F4	Gain Calibration Control (GCC0 - GCC1)	32	See section	0000_0000
F8 - FC	Gain Calculation Result (GCR0 - GCR1)	32	See section	0000_0000
100	Command Low Buffer Register (CMDL1)	32	See section	0000_0000
104	Command High Buffer Register (CMDH1)	32	See section	0000_0000
108	Command Low Buffer Register (CMDL2)	32	See section	0000_0000
10C	Command High Buffer Register (CMDH2)	32	See section	0000_0000
110	Command Low Buffer Register (CMDL3)	32	See section	0000_0000
114	Command High Buffer Register (CMDH3)	32	See section	0000_0000
118	Command Low Buffer Register (CMDL4)	32	See section	0000_0000
11C	Command High Buffer Register (CMDH4)	32	See section	0000_0000
120	Command Low Buffer Register (CMDL5)	32	See section	0000_0000
124	Command High Buffer Register (CMDH5)	32	See section	0000_0000
128	Command Low Buffer Register (CMDL6)	32	See section	0000_0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
12C	Command High Buffer Register (CMDH6)	32	See section	0000_0000
130	Command Low Buffer Register (CMDL7)	32	See section	0000_0000
134	Command High Buffer Register (CMDH7)	32	See section	0000_0000
138	Command Low Buffer Register (CMDL8)	32	See section	0000_0000
13C	Command High Buffer Register (CMDH8)	32	See section	0000_0000
140	Command Low Buffer Register (CMDL9)	32	See section	0000_0000
144	Command High Buffer Register (CMDH9)	32	See section	0000_0000
148	Command Low Buffer Register (CMDL10)	32	See section	0000_0000
14C	Command High Buffer Register (CMDH10)	32	See section	0000_0000
150	Command Low Buffer Register (CMDL11)	32	See section	0000_0000
154	Command High Buffer Register (CMDH11)	32	See section	0000_0000
158	Command Low Buffer Register (CMDL12)	32	See section	0000_0000
15C	Command High Buffer Register (CMDH12)	32	See section	0000_0000
160	Command Low Buffer Register (CMDL13)	32	See section	0000_0000
164	Command High Buffer Register (CMDH13)	32	See section	0000_0000
168	Command Low Buffer Register (CMDL14)	32	See section	0000_0000
16C	Command High Buffer Register (CMDH14)	32	See section	0000_0000
170	Command Low Buffer Register (CMDL15)	32	See section	0000_0000

Table continues on the next page...

Table continued from the previous page...

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
174	Command High Buffer Register (CMDH15)	32	See section	0000_0000
200 - 238	Compare Value Register (CV1 - CV15)	32	RW	0000_0000
300 - 304	Data Result FIFO Register (RESFIFO0 - RESFIFO1)	32	See section	0000_0000
400	Calibration General A-Side Registers (CAL_GAR0)	32	See section	0000_0000
404	Calibration General A-Side Registers (CAL_GAR1)	32	See section	0000_0000
408 - 40C	Calibration General A-Side Registers (CAL_GAR2 - CAL_GAR3)	32	See section	0000_0000
410 - 41C	Calibration General A-Side Registers (CAL_GAR4 - CAL_GAR7)	32	See section	0000_0000
420 - 43C	Calibration General A-Side Registers (CAL_GAR8 - CAL_GAR15)	32	See section	0000_0000
440 - 47C	Calibration General A-Side Registers (CAL_GAR16 - CAL_GAR31)	32	See section	0000_0000
480	Calibration General A-Side Registers (CAL_GAR32)	32	See section	0000_0000
500	Calibration General B-Side Registers (CAL_GBR0)	32	See section	0000_0000
504	Calibration General B-Side Registers (CAL_GBR1)	32	See section	0000_0000
508 - 50C	Calibration General B-Side Registers (CAL_GBR2 - CAL_GBR3)	32	See section	0000_0000
510 - 51C	Calibration General B-Side Registers (CAL_GBR4 - CAL_GBR7)	32	See section	0000_0000
520 - 53C	Calibration General B-Side Registers (CAL_GBR8 - CAL_GBR15)	32	See section	0000_0000
540 - 57C	Calibration General B-Side Registers (CAL_GBR16 - CAL_GBR31)	32	See section	0000_0000
580	Calibration General B-Side Registers (CAL_GBR32)	32	See section	0000_0000

47.6.1.1 Version ID Register (VERID)

Indicates the version integrated for this instance on the device and the inclusion of optional features.

Offset

Register	Offset
VERID	0h

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	NUM_FIFO			NUM_SEC	CALOFSI	IADCKI	VR1RNGI	0	CSW			MVI	0	DIFFEN	RES
W																
Reset	0	0	1	0	1	1	0	0	0	0	0	0	1	0	1	1

Fields

Field	Description
31-24 MAJOR	Major Version Number Returns the major version number for the module specification.
23-16 MINOR	Minor Version Number Returns the minor version number for the module specification.
15 —	Reserved
14-12 NUM_FIFO	Number of FIFOs Indicates the number of result FIFOs implemented in the design. 000 - N/A 001 - One 010 - Two 011 - Three 100 - Four
11 NUM_SEC	Number of Single-Ended Outputs Supported Indicates the number of single-ended channels which can be processed simultaneously. 0 - One 1 - Two

Table continues on the next page...

Table continued from the previous page...

Field	Description
10 CALOFSI	Calibration Function Implemented Indicates whether ADC contains hardware calibration functions. When supported, CTRL[CAL_REQ] can be used to request the calibration routine. 0 - Not implemented 1 - Implemented
9 IADCKI	Internal ADC Clock Implemented Indicates whether this implementation of the ADC block includes an internal clock source. 0 - Not implemented 1 - Implemented
8 VR1RNGI	Voltage Reference 1 Range Control Bit Implemented Indicates whether a control bit is implemented to select the input voltage range on Voltage Reference Option 1. 0 - Range control not required. 1 - Range control required.
7 —	Reserved
6-4 CSW	Channel Scale Width Indicates whether channel scaling is supported. When supported, each command buffer has a control field (CMDLn[CSCALE]) for setting input scaling. 000 - Not supported. 001 - Supported with one-bit CSCALE control field. 110 - Supported with six-bit CSCALE control field.
3 MVI	Multiple Vref Implemented Indicates whether multiple voltage reference high (VREFH) inputs are supported. When multiple voltage references are supported, CFG[REFSEL] selects voltage reference high options. 0 - Single VREFH input supported. 1 - Multiple VREFH inputs supported.
2 —	Reserved
1 DIFFEN	Differential Supported Indicates whether differential operation is supported. When supported, each command buffer has control fields (CMDLn[CTYPE]) for configuring for differential operation and expanding the number of supported channels from 32 to 64.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - Not supported 1 - Supported. CMDLn[CTYPE] controls fields implemented.
0 RES	Resolution Indicates the maximum accuracy supported. 0 - Up to 13-bit differential or 12-bit single-ended resolution supported. 1 - Up to 16-bit differential or 16-bit single-ended resolution supported. CMDLn[MODE] available for selecting the resolution of conversions for the associated command.

47.6.1.2 Parameter Register (PARAM)

The Parameter register indicates the size of several variable integration options for this instance on the device.

Offset

Register	Offset
PARAM	4h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CMD_NUM								CV_NUM							
W																
Reset	0	0	0	0	1	1	1	1	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	FIFOSIZE								TRIG_NUM							
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0

Fields

Field	Description
31-24 CMD_NUM	Command Buffer Number Indicates number of command buffers implemented.
23-16 CV_NUM	Compare Value Number Indicates number of compare value registers implemented.

Table continues on the next page...

Table continued from the previous page...

Field	Description
15-8 FIFOSIZE	Result FIFO Depth Indicates the maximum number of conversion data words that can be stored in the result FIFO before an overflow occurs. 0000_0001 - 2 0000_0100 - 4 0000_1000 - 8 0001_0000 - 16 0010_0000 - 32 0100_0000 - 64
7-0 TRIG_NUM	Trigger Number Number of Triggers implemented.

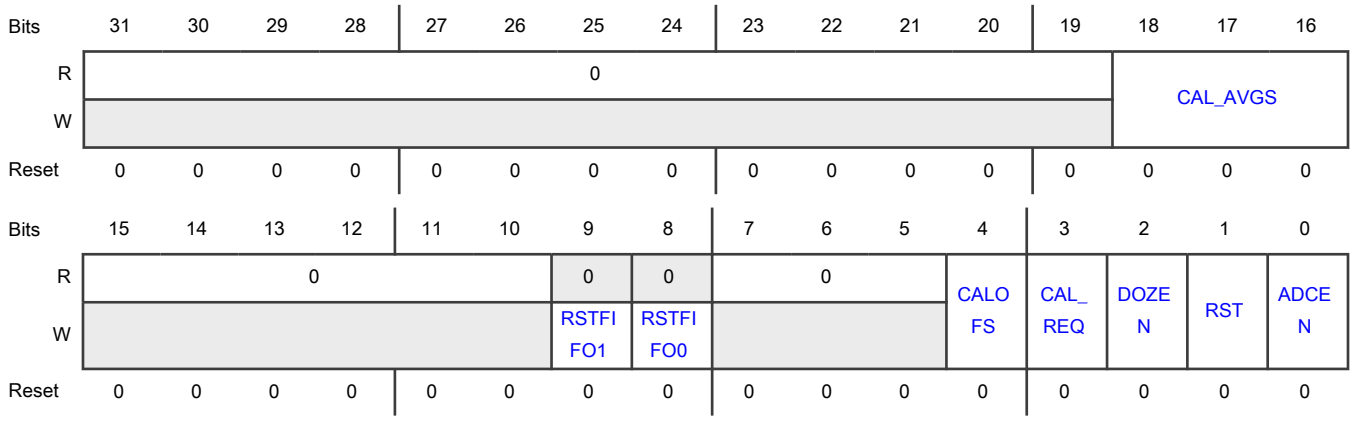
47.6.1.3 Control Register (CTRL)

Includes primary control bits.

Offset

Register	Offset
CTRL	10h

Diagram



Fields

Field	Description
31-19	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
18-16 CAL_AVGS	Auto-Calibration Averages Selects how many ADC conversions are averaged to calculate each calibration value. Selecting a higher number of averages will lead to more accurate conversions after completing calibration. The CAL_AVGS bit field applies to both CALOFS and CAL_REQ calibration types. This value should be fixed when requesting and running calibration with CTRL[CAL_REQ] or CTRL[CALOFS]. 000 - Single conversion. 001 - 2 conversions averaged. 010 - 4 conversions averaged. 011 - 8 conversions averaged. 100 - 16 conversions averaged. 101 - 32 conversions averaged. 110 - 64 conversions averaged. 111 - 128 conversions averaged.
15-10 —	Reserved
9 RSTFIFO1	Reset FIFO 1 0 - No effect. 1 - FIFO 1 is reset.
8 RSTFIFO0	Reset FIFO 0 0 - No effect. 1 - FIFO 0 is reset.
7-5 —	Reserved
4 CALOFS	Offset Calibration Request Indicates whether a request for a hardware calibration calculation has been made. Writing 1 to this field initiates a hardware offset calibration calculation. Any conversions in progress are completed before launching the offset calibration function. After being accepted, ADC calculates the OFSTRIM[OFSTRIM] value and updates the OFSTRIM register automatically. This field becomes 0 upon completion of the offset calibration calculation. STAT[CAL_RDY] indicates when the offset calibration routine has completed. 0 - Calibration function disabled 1 - Request for offset calibration function
3	Auto-Calibration Request

Table continues on the next page...

Table continued from the previous page...

Field	Description
CAL_REQ	Indicates whether a request for the hardware calibration routine has been made. Automatically becomes 0 when the system accepts the hardware calibration request. STAT[CAL_RDY] indicates when this calibration routine has been completed. 0 - No request made. 1 - Request has been made.
2 DOZEN	Doze Enable Controls system transition to Deep-sleep/ and Sleep power modes while ADC is converting. When 0, immediate entries to Deep-sleep or Sleep modes are allowed and ADC conversion functions remain enabled. Any conversion in progress is not disrupted. The selected clock source provided from the on-chip clock source must be able to continue operating. When 1, the ADC waits for the current averaging iteration or FIFO storage to complete before acknowledging the low-power entry request. After entering the low-power state, ADC is kept inactive until the system exits the low-power state. When the system enters Deep Power-down mode, the DOZEN bit is ignored and ADC waits for the current transfer to complete any pending operation. 0 - ADC is enabled in low-power mode. 1 - ADC is disabled in low-power mode.
1 RST	Software Reset Resets all internal logic and registers, except the CTRL register. Remains 1 until cleared by software. 0 - ADC logic is not reset. 1 - ADC logic is reset.
0 ADCEN	ADC Enable 0 - Disabled 1 - Enabled

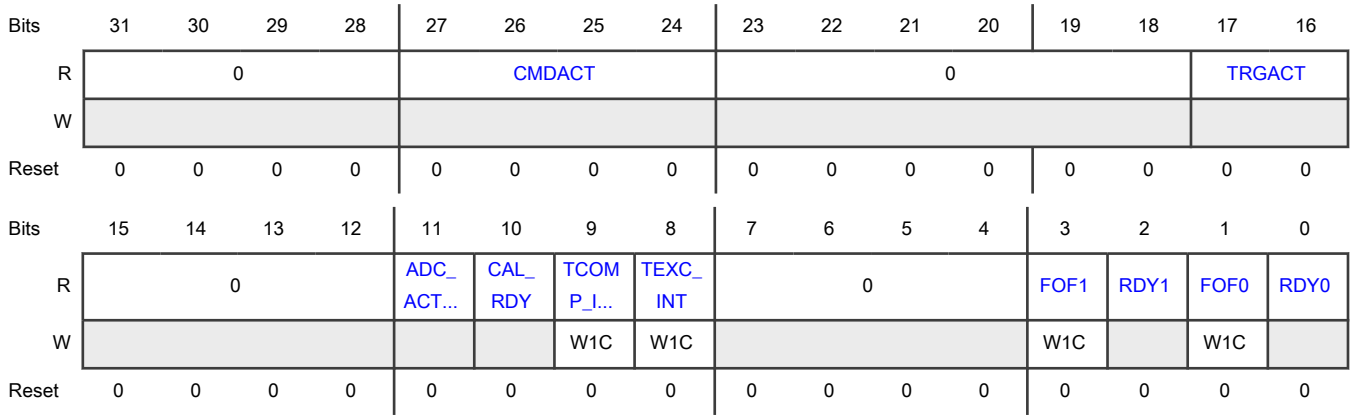
47.6.1.4 Status Register (STAT)

Provides the status of the ADC module.

Offset

Register	Offset
STAT	14h

Diagram



Fields

Field	Description
31-28 —	Reserved
27-24 CMDACT	Command Active Indicates the command actively being processed. Commands are only shown here when they are actively being processed by the SAR routine. Use with STAT[ADC_ACTIVE] to determine which conversion is running. 0000 - No command currently in progress. 0001 - Command 1 currently being executed. 0010 - Command 2 currently being executed. 0011-1111 - Associated command number currently being executed.
23-18 —	Reserved
17-16 TRGACT	Trigger Active Indicates the trigger actively being processed. This field can be used to determine which trigger is running through a trigger delay or being converted by the SAR routine. The command associated with this field is running through a conversion when STAT[CMDACT] is not zero. 00 - Command (sequence) associated with Trigger 0 currently being executed. 01 - Command (sequence) associated with Trigger 1 currently being executed. 10 - Command (sequence) associated with Trigger 2 currently being executed. 11 - Command (sequence) associated with Trigger 3 currently being executed.
15-12 —	Reserved
11	ADC Active

Table continues on the next page...

Table continued from the previous page...

Field	Description
ADC_ACTIVE	Indicates whether the module is processing a conversion, or has pending triggers to service. 0 - ADC is idle. There are no pending triggers to service and no active commands are being processed. 1 - ADC is processing a conversion, running through the power-up delay, or servicing a trigger.
10 CAL_RDY	Calibration Ready Indicates whether the ADC request for calibration (CTRL[CAL_REQ] or CTRL[CALOFS]) has been completed and the results are ready. This flag is automatically cleared when a new hardware calibration or OFSTRIM request is made. 0 - Calibration is incomplete or has not been run. 1 - ADC is calibrated.
9 TCOMP_INT	Interrupt Flag For Trigger Completion Indicates when a trigger sequence has been completed (all associated commands have been run). IE[TCOMP_IE] must be 1 for the specific trigger source to flag an interrupt upon completion. 0 - Either IE[TCOMP_IE] = 0, or no trigger sequences have run to completion. 1 - Trigger sequence has been completed and all data is stored in the associated FIFO.
8 TEXC_INT	Interrupt Flag For High-Priority Trigger Exception Indicates when a high-priority trigger exception has occurred and CFG[TRES] = 0. This flag only asserts if trigger exception interrupts are enabled (IE[TEXC_IE] = 1h). This flag can be used with TSTAT[TEXC_NUM] to resolve which trigger source was interrupted. 0 - No trigger exceptions have occurred. 1 - A trigger exception has occurred and is pending acknowledgment.
7-4 —	Reserved
3 FOF1	Result FIFO1 Overflow Flag Indicates that more data has been written to the Result FIFO 1 than it can hold. The newer data is not stored and the FIFO holds the original contents. This flag asserts regardless of the value of IE[FOFIE1]. However, an interrupt request is issued only if IE[FOFIE1] is 1. 0 - No result FIFO1 overflow has occurred since the last time that the flag was cleared. 1 - At least one result FIFO1 overflow has occurred since the last time that the flag was cleared.
2 RDY1	Result FIFO1 Ready Flag Indicates when the number of valid data words in the result FIFO 1 is greater than the level set in FCTRL0[FWMARK]. This flag asserts regardless of the value of IE[FWMIE1]. However, an interrupt request or DMA request occurs only when the associated control bit (IE[FWMIE1] and DE[FWMDE1]) is set. This flag is cleared when FCTRL0[FCOUNT] (which decrements on each read of the RESFIFO register) is less than or equal to the level set in FCTRL0[FWMARK]. 0 - Not above watermark

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - Above watermark
1 FOF0	Result FIFO 0 Overflow Flag Indicates that more data has been written to the Result FIFO 0 than it can hold. The newer data is not stored and the FIFO holds the original contents. This flag asserts regardless of the value of IE[FOFIE0]. However, an interrupt request is issued only if IE[FOFIE0] is 1. 0 - No result FIFO 0 overflow has occurred since the last time that the flag was cleared. 1 - At least one result FIFO 0 overflow has occurred since the last time that the flag was cleared.
0 RDY0	Result FIFO 0 Ready Flag Indicates when the number of valid data words in the result FIFO 0 is greater than the watermark set in FCTRL0[FWMARK]. This flag asserts regardless of the value of IE[FWMIE0]. However, an interrupt request or DMA request occurs only when the associated control field (IE[FWMIE0] and DE[FWMDE0]) is 1. This flag is cleared when the FCTRL0[FCOUNT] (which decrements on each read of the RESFIFO register) is less than or equal to the level set in FCTRL0[FWMARK]. 0 - Not above watermark 1 - Above watermark

47.6.1.5 Interrupt Enable Register (IE)

Includes system interrupt masking control bits.

Offset

Register	Offset
IE	18h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												TCOMP_IE			
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							TEXC_IE	0				FOFIE1	FWMIE1	FOFIE0	FWMIE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-20 —	Reserved
19-16 TCOMP_IE	Trigger Completion Interrupt Enable Enables generation of interrupt requests to indicate when complete trigger command sequences are executed. Each bit in TCOMP_IE corresponds to a trigger source. (TCOMP_IE[0] corresponds to trigger 0, and so on.) All results are stored in the FIFO when a sequence is considered complete. 0000 - All disabled 0001 - Trigger completion interrupts are enabled for trigger source 0 only. 0010 - Trigger completion interrupts are enabled for trigger source 1 only. 0011-1110 - Associated trigger completion interrupts are enabled. 1111 - All enabled
15-9 —	Reserved
8 TEXC_IE	Trigger Exception Interrupt Enable Enables ADC to assert an interrupt request when a high-priority trigger exception occurs. TSTAT[TEXC_NUM] contains the value of the corresponding trigger affected by the exception. 0 - Disabled 1 - Enabled
7-4 —	Reserved
3 FOFIE1	Result FIFO1 Overflow Interrupt Enable Enables generation of overflow interrupt requests when FOF1 flag is asserted. 0 - Disabled 1 - Enabled
2 FWMIE1	FIFO1 Watermark Interrupt Enable Enables generation of watermark interrupt requests when STAT[RDY1] flag is asserted. 0 - Disabled 1 - Enabled
1 FOFIE0	Result FIFO 0 Overflow Interrupt Enable Enables generation of overflow interrupt requests when FOF flag is asserted. 0 - Disabled 1 - Enabled

Table continues on the next page...

Table continued from the previous page...

Field	Description
0 FWMIE0	FIFO 0 Watermark Interrupt Enable Enables generation of watermark interrupt requests when STAT[RDY0] flag is asserted. 0 - Disabled 1 - Enabled

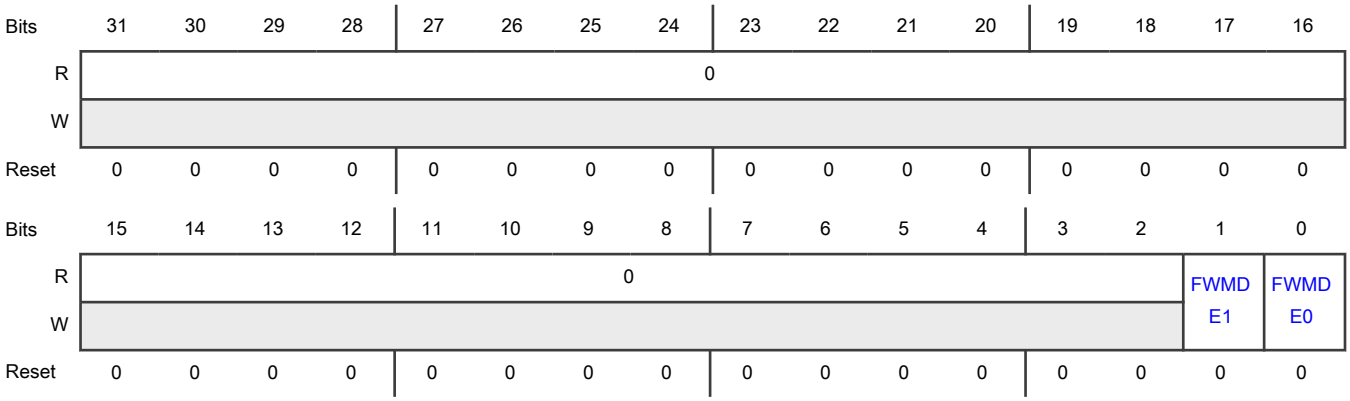
47.6.1.6 DMA Enable Register (DE)

Includes DMA request-masking control bits.

Offset

Register	Offset
DE	1Ch

Diagram



Fields

Field	Description
31-2 —	Reserved
1 FWMDE1	FIFO1 Watermark DMA Enable Enables generation of DMA requests when STAT[RDY1] flag is asserted. 0 - Disabled 1 - Enabled
0	FIFO 0 Watermark DMA Enable

Table continues on the next page...

Table continued from the previous page...

Field	Description
FWMDE0	Enables generation of DMA requests when STAT[RDY0] flag is asserted. 0 - Disabled 1 - Enabled

47.6.1.7 Configuration Register (CFG)

Controls ADC functions common to all commands. When CTRL[ADCEN] = 1, this register cannot be changed and writes to this register are ignored.

Offset

Register	Offset
CFG	20h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	PWREN	0				PUDLY							
W																
Reset	0	0	0	0	0	0	0	0	1	0	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				HPT_EXDI	TCMDRES	TRES	REFSEL		PWRSEL		0		TPRICTRL	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31 —	Reserved
30 —	Reserved
29 —	Reserved
28	ADC Analog Pre-Enable

Table continues on the next page...

Table continued from the previous page...

Field	Description
PWREN	Enables the ADC analog circuits. When setting this field, user code should delay for a period exceeding the analog startup time of $t_{ADCSTUP}$ before enabling ADC for operation. The module is still operational even when this field is 0, but command execution start is delayed for a period defined by CFG[PUDLY]. See the device data sheet for ADC idle and active power consumption parameters. 0 - ADC analog circuits are only enabled while conversions are active. Analog startup delays affect performance. 1 - ADC analog circuits are pre-enabled and ready to execute conversions without startup delays, at the cost of higher DC current consumption. A single power-up delay (CFG[PUDLY]) is executed immediately once PWREN is set. No detected triggers begin ADC operation until the power-up delay time has passed. After this initial delay expires, the analog circuits remain pre-enabled, and no additional delays are executed.
27-24 —	Reserved
23-16 PUDLY	Power-up Delay Defines the power-up delay executed after an initial trigger wakes ADC from its idle state. Must be programmed to a non-zero value to enable the ADC. Depending on the value of CFG[PWREN], the delay is executed in one of two modes: • When CFG[PWREN] = 0, the ADC analog circuits are only powered on while the module is active. The power-up delay allows time for the analog circuits to stabilize after being powered on. The startup delay count of (PUDLY * 4) ADCK cycles must result in a longer delay than the analog startup time of $t_{ADCSTUP}$. Accuracy of the initial conversions after activation is degraded if CFG[PUDLY] is set to too small a value. After active conversions, if no subsequent conversions are pending, the ADC analog circuits are automatically reverted to their low-power idle state. When ADC is awakened with a later trigger, the power-up delay runs again. • When CFG[PWREN] = 1 prior to ADC activation, the analog circuits are pre-enabled and the activation delay defined by CFG[PUDLY] is executed immediately. After the delay has been executed, the analog circuits remain enabled regardless of ADC activity. This configuration begins conversions immediately after trigger event detection, at the cost of increased DC power consumption in the analog circuits. **NOTE** ADC does not begin an initial conversion until the power-up delay has executed, regardless of the value of CFG[PWREN].
15 —	Reserved
14-11 —	Reserved
10 HPT_EXDI	High-Priority Trigger Exception Disable Disables high-priority trigger exceptions. See Trigger detect and command execution for a detailed description on trigger exception handling. When 1, exceptions are disabled and CFG[TCMDRES], CFG[TRES], and CFG[TPRICTRL] are ignored.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0 - Enabled 1 - Disabled
9 TCMDRES	Trigger Command Resume Determines where a trigger sequence resumes when interrupted by a high-priority trigger exception. To use this field, CFG[TRES] must be 1. 0 - Trigger sequence automatically restarted. 1 - Trigger sequence resumed from the command that was executed prior to the exception.
8 TRES	Trigger Resume Enable Determines whether trigger sequences interrupted by a high-priority exception are automatically resumed or restarted. If 1, the sequence can be resumed along command boundaries or restarted from the beginning of a sequence, as determined by CFG[TCMDRES]. If 0, interrupted triggers can be resumed via software by monitoring STAT[TEXC_INT] (or via ISR handling for trigger exception interrupt when IE[TEXC_IE] = 1). Software determines which trigger was interrupted by reading TSTAT[TEXC_NUM] and restarts the trigger by writing 1 to the corresponding field in the SWTRIG register. 0 - Not automatically resumed or restarted 1 - Automatically resumed or restarted
7-6 REFSEL	Voltage Reference Selection Selects the voltage reference high used for conversions. **NOTE** See the chip-specific ADC information for voltage reference options specific to this packaged device. 00 - Option 1 01 - Option 2 10 - Option 3 11 - Reserved
5-4 PWRSEL	Power Configuration Select Configures the module for power and performance. In the high-power setting, the highest conversion rates are possible. See the device data sheet for power and performance capabilities for each setting. 0x - Low power 1x - High power
3-2 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
1-0 TPRICTRL	ADC Trigger Priority Control Controls how higher-priority trigger exceptions are handled when they are received during command processing. See Trigger detect and command execution for a detailed explanation of trigger event handling. 00 - Current conversion is aborted and the new command specified by the trigger is started. 01 - Current command is stopped after completing the current conversion. If averaging is enabled, the averaging loop is completed. CMDHn[LOOP] is ignored and the higher-priority trigger is serviced. 10 - Current command is completed (averaging, looping, compare) before servicing the higher-priority trigger. 11 - Reserved

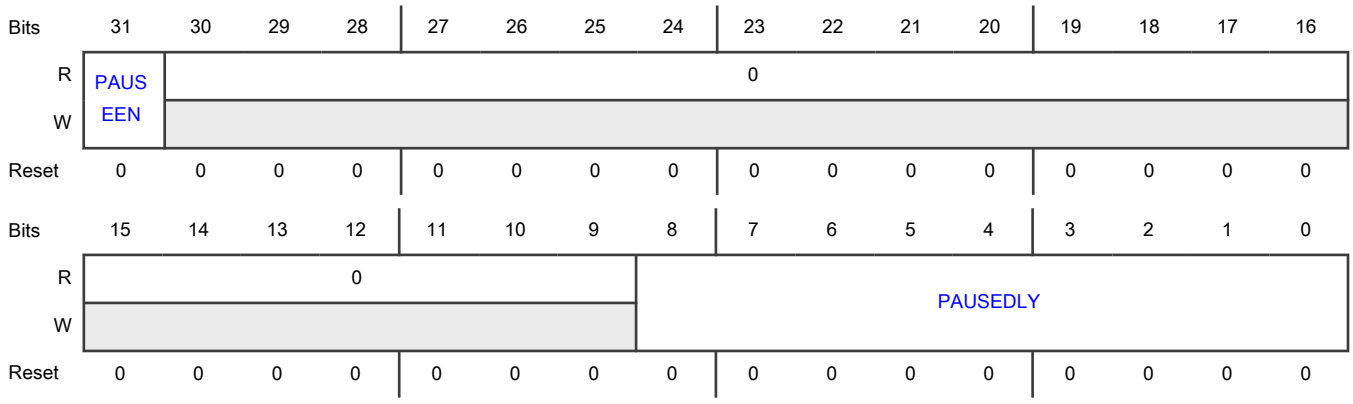
47.6.1.8 Pause Register (PAUSE)

Controls the optional inserted delay between conversions. When CTRL[ADCEN] = 1, this register should not be modified.

Offset

Register	Offset
PAUSE	24h

Diagram



Fields

Field	Description
31 PAUSEEN	Pause Enable Enables the ADC pausing function. When enabled, a programmable delay is inserted during command execution sequencing:

Table continues on the next page...

Table continued from the previous page...

Field	Description
	• Between LOOP iterations. • Between commands in a sequence. • Between conversions, when a command is executing in compare-until-true configuration. **NOTE** CMDHn[WAIT_TRIG] and PAUSE are mutually exclusive. CMDHn[WAIT_TRIG] takes priority over PAUSE between commands when both are enabled. 0 - Disabled 1 - Enabled
30-9 —	Reserved
8-0 PAUSEDLY	Pause Delay Controls the duration of pauses during command execution sequencing when PAUSE[PAUSEEN] = 1. The pause delay is a count of (PAUSEDLY * 4) ADCK cycles.

47.6.1.9 Software Trigger Register (SWTRIG)

Initiates software-triggered conversions when written to. Writes to this register are ignored while CTRL[ADCEN] = 0. There is an approximately three ADC-clock-cycle synchronization delay between asserting CTRL[ADCEN] and when SWTRIG can be accepted.

Offset

Register	Offset
SWTRIG	34h

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	0	0
W													SWT3	SWT2	SWT1	SWT0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-4 —	Reserved
3 SWT3	Software Trigger 3 Write 1 to SWT3 generates a trigger 3 event. Writing 1 to SWT3 is ignored while the trigger 3 event is being serviced or is pending. 0 - No trigger 3 event generated. 1 - Trigger 3 event generated.
2 SWT2	Software Trigger 2 Write 1 to SWT2 generates a trigger 2 event. Writing 1 to SWT2 is ignored while the trigger 2 event is being serviced or is pending. 0 - No trigger 2 event generated. 1 - Trigger 2 event generated.
1 SWT1	Software Trigger 1 Write 1 to SWT1 generates a trigger 1 event. Writing 1 to SWT1 is ignored while the trigger 1 event is being serviced or is pending. 0 - No trigger 1 event generated. 1 - Trigger 1 event generated.
0 SWT0	Software Trigger 0 Generates a trigger 0 event. Writing 1 to this field is ignored while the trigger 0 event is being serviced or is pending. 0 - No trigger 0 event generated. 1 - Trigger 0 event generated.

47.6.1.10 Trigger Status Register (TSTAT)

Contains status flags to indicate when trigger sequences have been completed or interrupted by a high-priority trigger exception. Each field in this register is set by hardware and cleared by software.

Offset

Register	Offset
TSTAT	38h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												TCOMP_FLAG			
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												TEXC_NUM			
W													W1C			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-20 —	Reserved
19-16 TCOMP_FLAG	Trigger Completion Flag Indicates which triggers have been completed. Each bit in this field corresponds to a trigger. When a trigger sequence has completed, the corresponding bit in TCOMP_FLAG is set. For example, if trigger 3 has been completed, then bit 19 is set in the TSTAT register. This register is active only if the corresponding bit in IE[TCOMP_IE] = 1. **NOTE** A synchronization delay may be added to the end of the final conversion in a sequence, prior to this flag being set. This delay can range from two to four ADC CLK cycles. 0000 - No triggers have been completed. Trigger completion interrupts are disabled. 0001 - Trigger 0 has been completed and trigger 0 has enabled completion interrupts. 0010 - Trigger 1 has been completed and trigger 1 has enabled completion interrupts. 0011-1110 - Associated trigger sequence has completed and has enabled completion interrupts. 1111 - Every trigger sequence has been completed and every trigger has enabled completion interrupts.
15-4 —	Reserved
3-0 TEXC_NUM	Trigger Exception Number Indicates which triggers have been interrupted by exceptions. Each bit in this field corresponds to a trigger. When the corresponding trigger sequence is interrupted by a high-priority trigger exception, the corresponding bit is set. For example, if trigger 3 has been interrupted, then bit 3 is set in this register. This register is active regardless of the value in IE[TEXC_IE]. 0000 - No triggers have been interrupted by a high-priority exception. Or CFG[TRES] = 1. 0001 - Trigger 0 has been interrupted by a high-priority exception.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0010 - Trigger 1 has been interrupted by a high-priority exception. 0011-1110 - Associated trigger sequence has interrupted by a high-priority exception. 1111 - Every trigger sequence has been interrupted by a high-priority exception.

47.6.1.11 Offset Trim Register (OFSTRIM)

Used to trim for offset. ADC supports a calibration step in which the ADC determines the value needed in the OFSTRIM register.

To determine the value to put in the OFSTRIM register, write 1 to CTRL[CALOFS]. This setting automatically begins a sequence to calculate this value. Once the sequence has completed, the OFSTRIM register is updated with a signed value between -16 and 15. This value is used to minimize offset during normal operation.

Offset

Register	Offset
OFSTRIM	40h

Diagram

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

Fields

Field	Description
31-21 —	Reserved
20-16 OFSTRIM_B	Trim for Offset OFSTRIM is a 5-bit signed value between -16 and 15. This value applies to the ADC B-side conversion results.
15-5	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
4-0 OFSTRIM_A	Trim for Offset OFSTRIM_A is a 5-bit signed value between -16 and 15. This value applies to the ADC A-side conversion results.

47.6.1.12 Trigger Control Register (TCTRL0 - TCTRL3)

Implements control fields associated with each trigger source. When ADC is executing commands, only one TCTRLn register controls ADC conversions.

Do not update the controlling TCTRLn register while ADC is active. Writing to a TCTRLn register while that trigger control register controls the ADC operation may cause unpredictable behavior.

Offset

Register	Offset
TCTRL0	A0h
TCTRL1	A4h
TCTRL2	A8h
TCTRL3	ACh

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				TCMD				0				TDLY			
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	RSYNC	0			TPRI				0				FIFO_SE...	FIFO_SE...	HTEN	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-28 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
27-24 TCMD	Trigger Command Select Selects the command from the command buffer to execute upon detection of the associated trigger event. When a higher-priority trigger is received while ADC is converting, the command in this register associated with the new trigger is executed. This execution is done under the control of CFG[TPRICTRL]. See Trigger detect and command execution for details about the relationship between CFG[TPRICTRL] and TCMD. 0000 - Not a valid selection from the command buffer. Trigger event is ignored. 0001 - CMD1 0010-1110 - Corresponding CMD is executed 1111 - CMD15
23-20 —	Reserved
19-16 TDLY	Trigger Delay Select Selects the trigger delay duration for the start of servicing a trigger event. Each trigger source has an associated programmable delay prior to beginning an initial conversion. When this field is 0, no delay is incurred. When this field is set to a non-zero value, the duration of the delay is 2^{TDLY} ADCK cycles.
15 RSYNC	Trigger Resync Enables trigger resync. If 1, this trigger source is used as a resync trigger. A resync trigger can be configured to restart or abort a running trigger sequence (the target). Each resync trigger is configured to have a resync target. When a resync trigger is asserted: • The target sequence is aborted. • The target FIFO destinations are cleared. • The target sequence can be restarted depending on CFG[TRES]. Multiple conditions must be met for a resync trigger to execute properly: • The resync trigger must have higher priority than the resync target. • The resync trigger TCTRLn[RSYNC] must be set to 1b. • The resync target, specified by TCTRLn[TCMD], must be executing when the resync trigger is asserted. See Resync functionality for more information. 0 - Disable 1 - Enable
14-10 —	Reserved
9-8	Trigger Priority Setting

Table continues on the next page...

Table continued from the previous page...

Field	Description
TPRI	Sets the priority of the associated trigger source. If two or more triggers have the same priority level, the lower-order trigger event has the higher priority. If Trigger 0 and Trigger 1 are pending triggers and TCTRL0[TPRI] is configured the same as TCTRL1[TPRI], the Trigger 0 command is serviced first. 00 - Highest priority, Level 1 01-10 - Set to corresponding priority level. 11 - Lowest priority, Level 4
7-3 —	Reserved
2 FIFO_SEL_B	SAR Result Destination for Channel B Indicates the FIFO to which SAR results are written for Channel B. This field is only used in dual single-ended mode (CMDLn[CTYPE] = 11b). In this mode, the SAR result from Channel B is written to the FIFO number specified by this field. See Sampling Modes for more information. 0 - FIFO 0 1 - FIFO 1
1 FIFO_SEL_A	SAR Result Destination for Channel A Indicates the FIFO to which SAR results are written for Channel A. This field provides the destination for all single-ended and differential conversions and for all A-side conversions in dual single-ended mode. Conversion results are stored in the FIFO number specified, under the following conditions: • Single-ended mode: CMDLn[CTYPE] = 00b or CMDLn[CTYPE] = 01b. • Differential mode: CMDLn[CTYPE] = 10b. • Dual single-ended mode: CMDLn[CTYPE] = 11b. 0 - FIFO 0 1 - FIFO 1
0 HTEN	Trigger Enable Enables hardware trigger source to initiate conversion on the rising edge of the input trigger source. **NOTE** Enabling the hardware trigger does not disable software triggers. 0 - Disabled 1 - Enabled

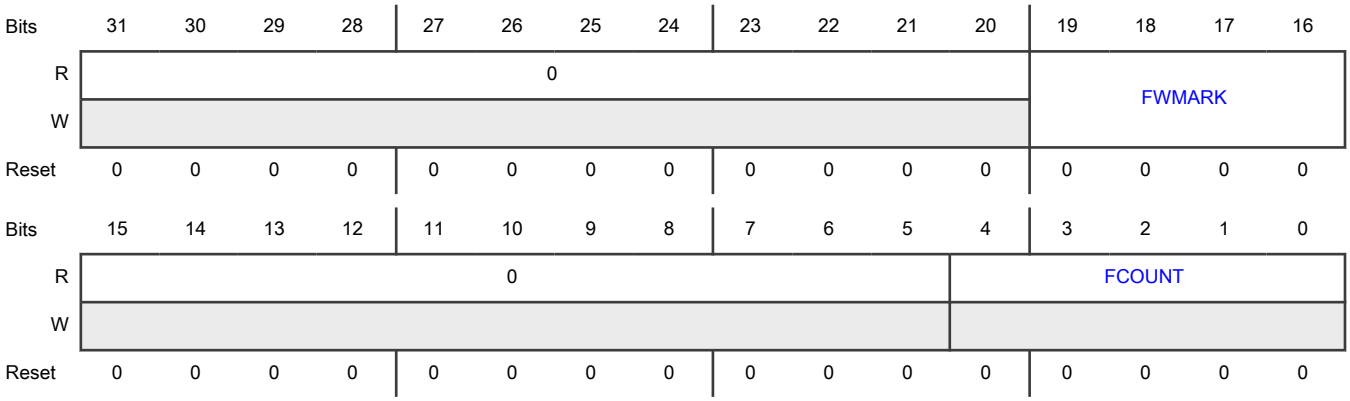
47.6.1.13 FIFO Control Register (FCTRL0 - FCTRL1)

Contains control and status fields for each FIFO in the design. A programmable watermark can be set for each FIFO, which can be used to trigger an interrupt. In addition, the number of entries stored in each FIFO can be monitored by reading FCTRLn[FCOUNT].

Offset

Register	Offset
FCTRL0	E0h
FCTRL1	E4h

Diagram



Fields

Field	Description
31-20 —	Reserved
19-16 FWMARK	Watermark Level Selection Selects the storage threshold for the ADC Result FIFO. When the number of data words stored in the FIFO is greater than this value, the STAT[RDY0] flag is asserted. When IE[FWMIEn] = 1, an interrupt request is generated. When DE[FWMDEn] = 1, a DMA request is generated.
15-5 —	Reserved
4-0 FCOUNT	Result FIFO Counter Indicates the number of data words stored in the result FIFO. This value may be used with PARAM[FIFOSIZE] to calculate how much room is left in the result FIFO. This field is incremented with each storage of new data into the result FIFO and decrements with each read of the result FIFO. The FIFO is reset by writing to CTRL[RSTFIFOn], which initializes FCTRLn[FCOUNT] to 0h.

47.6.1.14 Gain Calibration Control (GCC0 - GCC1)

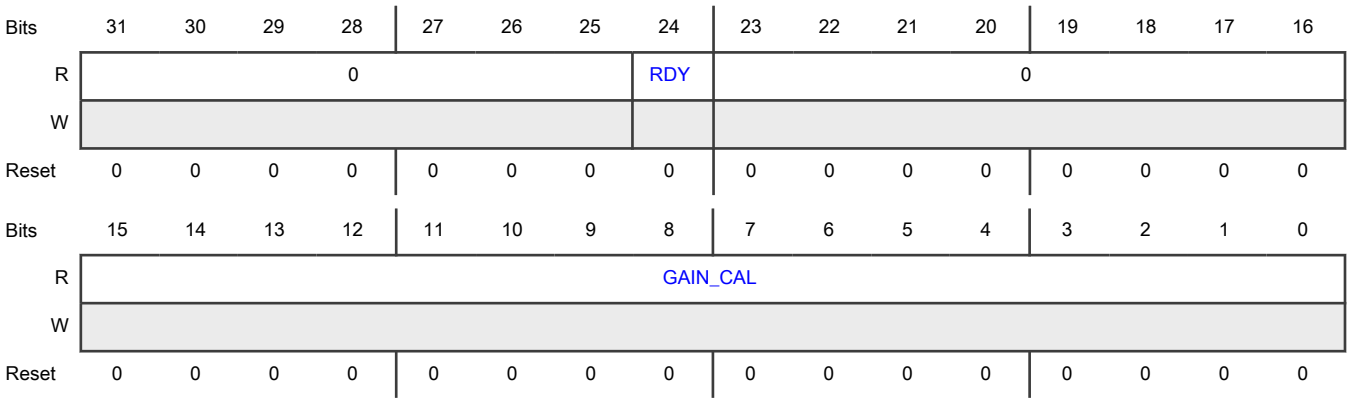
Holds intermediate values used as part of the calibration steps. GCC0 is associated with the A-side converter and GCC1 is associated with the B-side converter. GCCn[GAIN_CAL] is calculated with hardware and automatically updated during the calibration steps. Once the hardware calibration sequence has updated, GCCn[RDY] is asserted automatically.

Requesting a calibration routine automatically clears the GCCn[RDY] flag until the new GCCn[GAIN_CAL] value is calculated. To complete calibration, further software processing is needed, where GCCn[GAIN_CAL] is used to calculate the gain adjustment. The result is stored to GCRn[GCALR]. See Calibration functions.

Offset

Register	Offset
GCC0	F0h
GCC1	F4h

Diagram



Fields

Field	Description
31-25 —	Reserved
24 RDY	Gain Calibration Value Valid Indicates whether the data stored in GCCn[GAIN_CAL] is valid and should be used to derive GCRn[GCALR]. If the data in GCCn[GAIN_CAL] is invalid, you can run the hardware calibration routine to correct this issue. 0 - Invalid 1 - Valid
23-16 —	Reserved
15-0 GAIN_CAL	Gain Calibration Value Indicates the calculated value of the gain calibration. As part of the hardware calibration steps, this field is automatically updated with a 16-bit unsigned number. It is used in the gain adjustment calculations to derive the value written to GCRn[GCALR]. See Calibration functions.

47.6.1.15 Gain Calculation Result (GCR0 - GCR1)

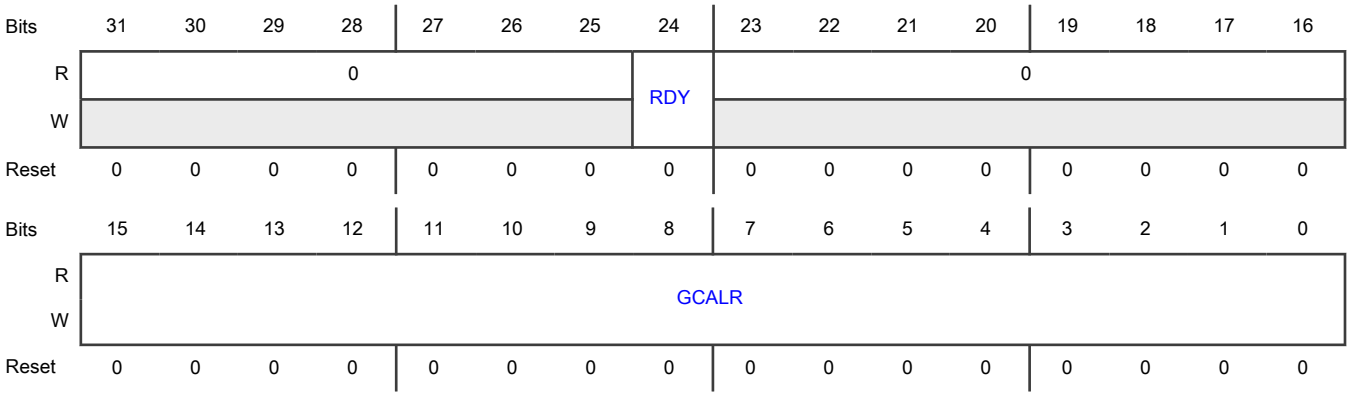
Used during ADC conversions for automated gain error adjustment. GCR0 is associated with the A-side converter and GCR1 is associated with the B-side converter. Determining the values to store to GCR0[GCALR] and GCR1[GCALR] is the result of software calculations described in setup steps in Calibration functions. Upon writing the GCRn[GCALR] value, the user should also write 1 to GCRn[RDY] to indicate that the gain adjustment result is valid.

A calibration sequence begins by writing 1 to CTRL[CAL_REQ], and the calibration sequence is not complete until GCRn[GCALR] is calculated and GCRn[RDY] is 1. The gain adjustment calculation produces a floating-point value between 1 and 2. GCRn[GCALR] holds the 16-bit fractional component of the gain adjustment calculation. To convert the GCRn[GCALR] value to the gain adjustment value in decimal format, use this formula: 1 + 0.5 * GCRn[15] + 0.25 * GCRn[14] + 0.125 * GCRn[13] + (and so on)

Offset

Register	Offset
GCR0	F8h
GCR1	FCh

Diagram



Fields

Field	Description
31-25 —	Reserved
24 RDY	Gain Calculation Ready Indicates whether the data stored in GCRn[GCALR] is valid and is used for gain adjustment. GCRn[GCALR] must be written from memory or calculated each time the calibration routine is run. The user must write 1 to this GCRn[RDY] after writing valid data to GCRn[GCALR]. This field becomes 0 automatically when requesting a new calibration sequence. 0 - Invalid 1 - Valid
23-16 —	Reserved
15-0 GCALR	Gain Calculation Result Holds the 16-bit fractional component generated from the gain adjustment calculation during the auto-calibration routine.

47.6.1.16 Command Low Buffer Register (CMDL1 - CMDL15)

Controls channel selection and conversion options. There are 15 command buffers (CMDn), each constructed from two 32-bit registers (CMDHn:CMDLn) that can be configured for different channel selection and varying conversion options. Any command buffer is selected and used as the controlling command by association with a trigger event via configuration of TCTRLn[TCMD]. When ADC is executing commands, only one of the CMD buffers controls ADC conversions. Do not update the controlling CMD buffer while ADC is active. A write to a CMD buffer while that CMD buffer controls the ADC operation may cause unpredictable behavior.

NOTE

The 15 command buffers are numbered [CMDH1:CMDL1] through [CMDH15:CMDL15]. In NXP-supplied header files, these buffers are likely to be defined as two 15-element arrays that are indexed from 0. For example, the type declaration would be:

```
unsigned int CMDH[15];
unsigned int CMDL[15];
```

and software would access ADC0 CMDH1 as `ADC0->CMDH[0]` and ADC0 CMDL15 as `ADC0->CMDL[14]`.

The CMDLn[ADCH] and CMDLn[CTYPE] fields control selection of paired or individual input channels. Each ADC command independently makes a channel and conversion type selection. Each ADCH channel selection has an associated A-side and B-side input. Each ADCH pair can be converted in differential mode, but only limited pairs should be converted as differential channels (for example, adjacent pins designed with matched impedance). For the pin pairings available for differential conversions for your device, see the Chip Configuration details.

NOTE

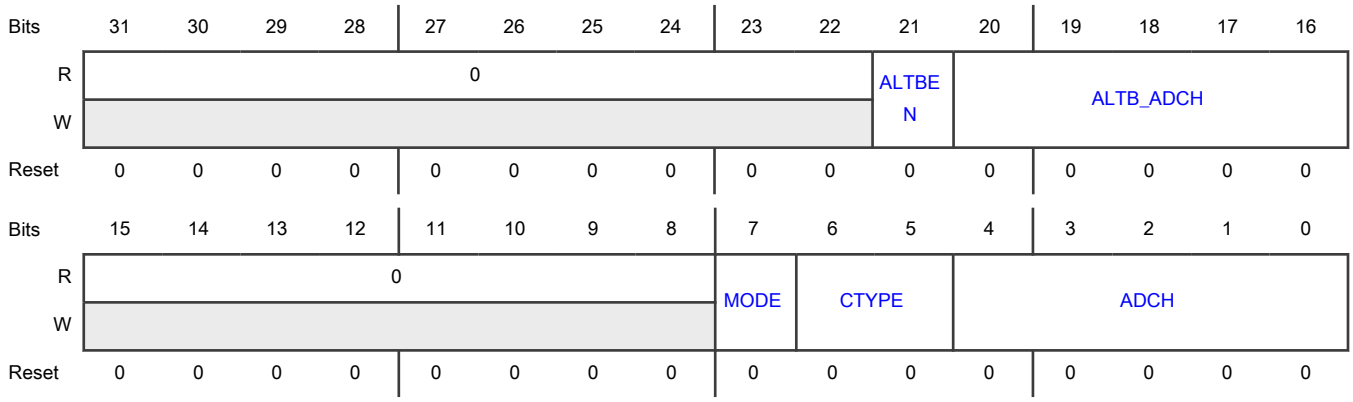
Some input channels are from on-chip sources such as temperature sensors and reference voltage sources and may only be connected to individual instances of ADC. Some input channel options in the field descriptions may not be available for your device. See the chip-specific information for the channels supported on this device.

Offset

For a = 1 to 15:

Register	Offset
CMDLa	F8h + (a × 8h)

Diagram



Fields

Field	Description
31-22 —	Reserved
21 ALTBEN	Alternate Channel B Select Enable Enables the ALTB_ADCH to select the input for Channel B independent of ADCH register settings when CTYPE is configured to 01b or 11b. **NOTE** Setting CTYPE to differential (10b) still works but is not recommended. 0 - ALTBEN_ADCH disabled. Channel-A and Channel-B inputs are selected based on ADCH settings. 1 - ALTBEN_ADCH enabled. Channel-A inputs are selected by ADCH setting and Channel-B inputs are selected by ALTB_ADCH setting.
20-16 ALTB_ADCH	Alternate Channel B Input Channel Select When ALTBEN is set, ALTB_ADCH selects the Channel B input when CTYPE is configured to 01b or 11b. **NOTE** Some input channels are from internal resources such as temperature sensors and band gap voltage sources and may only be connected to individual instances of ADC. Some input channel options in field descriptions might not be available for your device. See the chip-specific information for the ADC channel assignments for your device. 0_0000 - Select CH0B 0_0001 - Select CH1B 0_0010 - Select CH2B 0_0011 - Select CH3B 0_0100-1_1101 - Select corresponding channel CHnB 1_1110 - Select CH30B 1_1111 - Select CH31B
15-8 —	Reserved
7 MODE	Select Resolution of Conversions Selects the ADC resolution. 0 - Standard resolution. Single-ended 12-bit conversion; differential 13-bit conversion with 2's complement output. 1 - High resolution. Single-ended 16-bit conversion; differential 16-bit conversion with 2's complement output.
6-5 CTYPE	Conversion Type Chooses how a pair of A and B channels are converted.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	00 - Single-Ended mode. Only A-side channel is converted. 01 - Single-Ended mode. Only B-side channel is converted. 10 - Differential mode. A-B. 11 - Dual-Single-Ended mode. Both A-side and B-side channels are converted independently.
4-0 ADCH	Input Channel Select Selects the input from one of the channel inputs or one of the input pairs, with CMDLn[CTYPE]. Each ADCH channel selection has an associated A-side and B-side input. **NOTE** Not all pairs should be converted as differential channels. See the chip-specific information for the pin pairings available for differential conversions for your device. **NOTE** Some input channels are from internal resources such as temperature sensors and band gap voltage sources and may only be connected to individual instances of ADC. Some input channel options in field descriptions might not be available for your device. See the chip-specific information for the ADC channel assignments for your device. 0_0000 - CH0A or CH0B or CH0A/CH0B pair. 0_0001 - CH1A or CH1B or CH1A/CH1B pair. 0_0010 - CH2A or CH2B or CH2A/CH2B pair. 0_0011 - CH3A or CH3B or CH3A/CH3B pair. 0_0100-1_1101 - Select corresponding channel CHnA or CHnB or CHnA/CHnB pair. 1_1110 - CH30A or CH30B or CH30A/CH30B pair. 1_1111 - CH31A or CH31B or CH31A/CH31B pair.

47.6.1.17 Command High Buffer Register (CMDH1 - CMDH15)

Controls channel selection and conversion options. There are 15 command buffers (CMDn), each constructed from two 32-bit registers (CMDHn:CMDLn) that can be configured for different channel selection and conversion options. Any command buffer is selected and used as the controlling command by association with a trigger event via configuration of TCTRLn[TCMD]. When ADC is executing commands, only one of the CMD buffers controls ADC conversions. Do not update the controlling CMD buffer while ADC is active. A write to a CMD buffer while that CMD buffer controls the ADC operation may cause unpredictable behavior.

NOTE

The 15 command buffers are numbered [CMDH1:CMDL1] through [CMDH15:CMDL15]. In NXP-supplied header files, these buffers are likely to be defined as two 15-element arrays that are indexed from 0. For example, the type declaration would be:

```
unsigned int CMDH[15];
unsigned int CMDL[15];
```

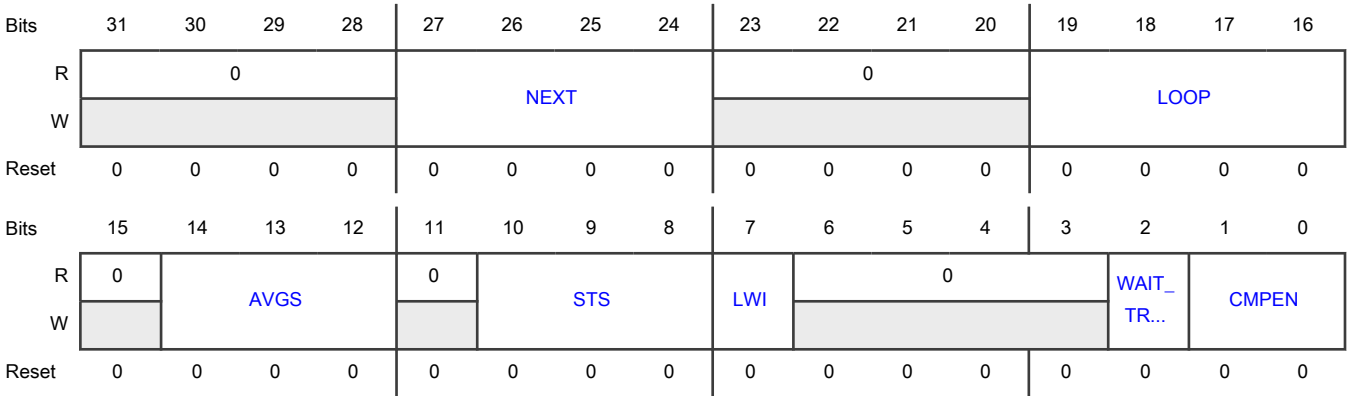
and software would access ADC0 CMDH1 as `ADC0->CMDH[0]` and ADC0 CMDL15 as `ADC0->CMDL[14]`.

Offset

For a = 1 to 15:

Register	Offset
CMDHa	FCh + (a × 8h)

Diagram



Fields

Field	Description
31-28 —	Reserved
27-24 NEXT	Next Command Select Selects the next command to be executed after this command completes. Multiple commands can be configured in a scan configuration by linking the next command in a daisy-chain sequence. The command buffer number is not indicative of any particular order. The order of execution is strictly controlled by this field. For example, a sequence of commands could be CMD2 to CMD1 to CMD3. Unending circular command execution can be configured by setting the NEXT field in the last command in a sequence to the first command in the sequence. It is also allowed for a command to set the next command to itself, resulting in a continuous conversion configuration. Setting the next command to 0h causes conversions to terminate at the completion of the command. Lower-priority trigger events cannot be serviced until a higher-priority triggered command (or sequence of commands) completes. 0000 - No next command defined. Terminate conversions at completion of current command. If lower priority trigger pending, begin command associated with lower priority trigger. 0001 - CMD1 0010-1110 - Select corresponding CMD command buffer register as next command 1111 - CMD15
23-20 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
19-16 LOOP	Loop Count Select Selects the number of times that this command executes (and stores conversion result to RESFIFO) before finishing and transitioning to the next command or idle state. CMDHn[LWI] controls whether a single channel is converted on each iteration or an automatic channel increment results in channel scanning functionality. 0000 - Looping not enabled. Command executes one time. 0001 - Loop one time. Command executes two times. 0010 - Loop two times. Command executes three times. 0011-1110 - Loop corresponding number of times. Command executes LOOP + 1 times. 1111 - Loop 15 times. Command executes 16 times.
15 —	Reserved
14-12 AVGS	Hardware Average Select Selects how many ADC conversions are averaged to create the ADC result (2^{AVGS}). An internal storage buffer captures temporary results while the averaging iterations are executed. Hardware averaging is a nested loop control and does not extend across LOOP boundaries. See Functional description for usage of AVGS, LOOP, and NEXT fields in command execution sequencing. 000 - Single conversion 001 - 2 010 - 4 011 - 8 100 - 16 101 - 32 110 - 64 111 - 128
11 —	Reserved
10-8 STS	Sample Time Select Selects the total sampling time. When this field contains a non-zero value, the sample time is ($3.5 + 2^{STS}$) ADCK cycles. The shortest sample time maximizes conversion speed for lower-impedance inputs. Extending sample time allows higher-impedance inputs to be sampled accurately. Longer sample times can lower overall power consumption when command looping and sequencing are configured and high conversion rates are not required. 000 - Minimum sample time of 3.5 ADCK cycles. 001 - 5.5 ADCK cycles. ($3.5 + 2^1$ ADCK cycles) 010 - 7.5 ADCK cycles. ($3.5 + 2^2$ ADCK cycles)

Table continues on the next page...

Table continued from the previous page...

Field	Description
	011 - 11.5 ADCK cycles. (3.5 + ^{3} ADCK cycles) 100 - 19.5 ADCK cycles. (3.5 + 2^4 ADCK cycles) 101 - 35.5 ADCK cycles. (3.5 + 2^5 ADCK cycles) 110 - 67.5 ADCK cycles. (3.5 + 2^6 ADCK cycles) 111 - 131.5 ADCK cycles. (3.5 + 2^7 ADCK cycles)
7 LWI	Loop with Increment Enables automatic channel incrementing. When 0, the LOOP field selects the number of times the selected channel is converted consecutively. When 1, automatic channel incrementing is enabled and the LOOP field defines how many consecutive channels are converted as part of the command execution. Example 1: LOOP = 8h, LWI = 0b, CMDLn[CTYPE] = 0h, CMDLn[ADCH] = Dh. Convert on channel 13A 9 times. Example 2: LOOP = 8h, LWI = 1b, CMDLn[CTYPE] = 0h, CMDLn[ADCH] = Dh. Run channels 13A-21A each one time. Maximum channel scanning using a single command buffer is defined by the maximum value of the LOOP field (16). 0 - Disabled 1 - Enabled
6-3 —	Reserved
2 WAIT_TRIG	Wait for Trigger Assertion Before Execution Selects whether commands execute automatically or a trigger must be received before execution. When 0, wait states are added before the command until the active trigger is asserted again. When WAIT_TRIG = 0, each command is automatically executed when called. 0 - Command executes automatically. 1 - Active trigger must be asserted again before executing this command.
1-0 CMPEN	Compare Function Enable Enables the automatic compare function. Selects whether to store only when the comparison operation is true, following ADC channel input sampling, conversion, and averaging. When the compare function is enabled, the conversion result is compared to the compare value registers (CVn[CVH] and CVn[CVL]). See Compare Function for details about the options for command sequencing related to the compare function. 00 - Disabled 01 - Reserved 10 - Enabled. Store on true. 11 - Enabled. Repeat channel acquisition (sample, convert, and compare) until true.

47.6.1.18 Compare Value Register (CV1 - CV15)

Contains values used to compare the conversion result when the compare function is enabled. This register is formatted like the D field in Data Result FIFO Register (RESFIFO0 - RESFIFO1), which has in different definitions for bit position and value format in different modes. There is a direct association of each compare value register to a specific command buffer register. For example, CV1 is only used during execution of CMD1 command.

When ADC is executing commands, do not update the CVn register associated with the active command (CMDn). Writes to associated CVn register during this time may result in unpredictable behavior.

NOTE

The 15 compare value registers are numbered [CV1] through [CV15]. In NXP-supplied header files, these registers are likely to be defined as a 15-element array indexed from 0. For example, the type declaration would be:

```
unsigned int CV[15];
```

and software would access ADC0 CV1 as `ADC0->CV[0]` and ADC0 CV15 as `ADC0->CV[14]`.

Offset

For a = 1 to 15:

Register	Offset
CVa	1FCh + (a × 4h)

Diagram

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

Fields

Field	Description
31-16 CVH	Compare Value High Determines the high compare value. The compare function can be configured to check whether the result: • Is less than comparison values • Is greater than comparison values • Falls within a range of two comparison values

Table continues on the next page...

Table continued from the previous page...

Field	Description
	• Falls outside a range of two comparison values. After the input is sampled and converted and any averaging iterations are performed, CVL and CVH can be used in a compare operation on the result. See Compare Function for a description of CVH usage.
15-0 CVL	Compare Value Low Determines the low compare value. The compare function can be configured to check whether the result: • Is less than comparison values • Is greater than comparison values • Falls within a range of two comparison values • Falls outside a range of two comparison values. After the input is sampled and converted and any averaging iterations are performed, CVL and CVH can be used in a compare operation on the result. See Compare Function for a description of CVL usage.

47.6.1.19 Data Result FIFO Register (RESFIFO0 - RESFIFO1)

Stores the data result of ADC conversions in a 16-entry FIFO. Several tag fields of source command and trigger information are stored with the data. FCTRLn[FCOUNT] indicates how many valid data words are stored in the RESFIFO. Reading RESFIFO provides the oldest unread data word entry in the FIFO and decrements FCTRLn[FCOUNT]. The FIFO can be emptied by successive reads of RESFIFO. The FIFO is reset by writing 0b1 to CTRL[RSTFIFOn].

The following table describes the format of data in the result FIFO in different modes of operation. The sign bit is the MSB in signed 2's complement modes. For example, when configured for 12-bit single-ended mode, D[15] and D[2:0] become 0. When configured for 13-bit differential mode, D[15] is the sign bit, and D[2:0] becomes 0.

Table 403. Data result register format description

Conversion mode	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	Format
16-bit differential	S	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	Signed 2's complement
16-bit single-ended	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	Unsigned, 16-bit magnitude
13-bit differential	S	D	D	D	D	D	D	D	D	D	D	D	D	0	0	0	Signed 2's complement, left justified, zero extended
12-bit single-ended	0	D	D	D	D	D	D	D	D	D	D	D	D	0	0	0	Unsigned, zero in D[15] and D[2:0]

NOTE

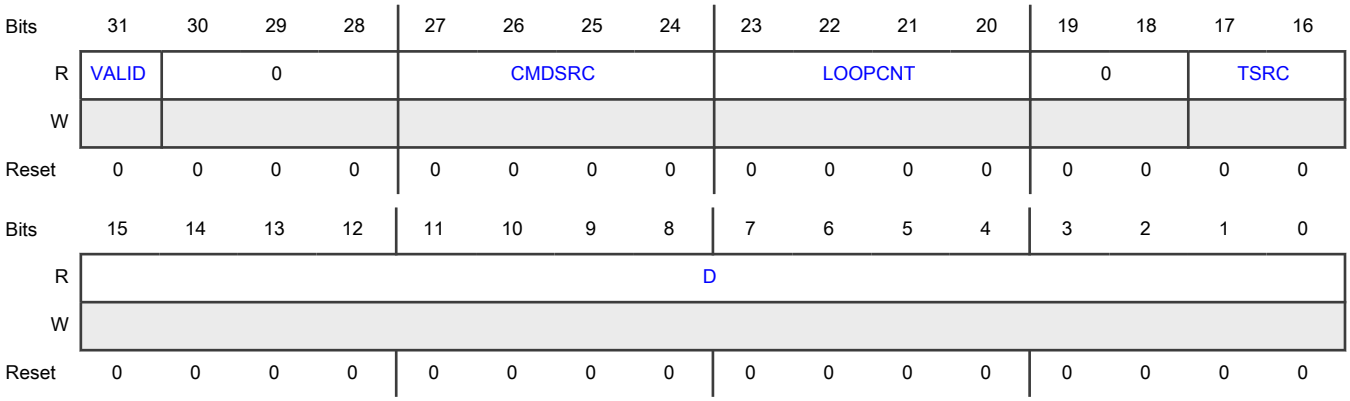
S: Sign bit;

D: Data, 2's complement data when indicated

Offset

Register	Offset
RESFIFO0	300h
RESFIFO1	304h

Diagram



Fields

Field	Description
31 VALID	FIFO Entry is Valid Indicates whether the FIFO entry is valid, which determines what happens to reads from RESFIFO. 0 - FIFO is empty. Discard any read from RESFIFO. 1 - FIFO contains data. FIFO record read from RESFIFO is valid.
30-28 —	Reserved
27-24 CMDSRC	Command Buffer Source Indicates the executed command buffer that generated this result. 0000 - Not a valid value CMDSRC value for a data word in RESFIFO. 0h is only found in the initial FIFO state, prior to the storage of an ADC conversion result into a RESFIFO buffer. 0001 - CMD1 0010-1110 - Corresponding command buffer used as control settings for this conversion. 1111 - CMD15
23-20 LOOPCNT	Loop Count Value Indicates the loop count value during the command that generated this result. When CMDHn[LOOP] is non-zero, results are stored multiple times during command execution at the loop boundary. 0000 - Result is from initial conversion in command.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	0001 - Result is from second conversion in command. 0010-1110 - Result is from (LOOPCNT + 1) conversion in command. 1111 - Result is from 16th conversion in command.
19-18 —	Reserved
17-16 TSRC	Trigger Source Indicates the trigger source that initiated a conversion and generated this result. When multiple commands are chained together using CMDHn[NEXT], this field indicates the trigger source that started the command sequence. 00 - Trigger source 0 01 - Trigger source 1 10-10 - Corresponding trigger source initiated this conversion. 11 - Trigger source 3
15-0 D	Data Result Contains the result of an ADC conversion. The formatting for the data in D is summarized in Table 403.

47.6.1.20 Calibration General A-Side Registers (CAL_GAR0)

Corrects for linearity errors of the A-side converter. All 33 A-side general calibration value registers (CAL_GAR0-CAL_GAR32) contain calibration information that is automatically updated during the self-calibration sequence. See Calibration functions for more information on completing ADC calibration steps.

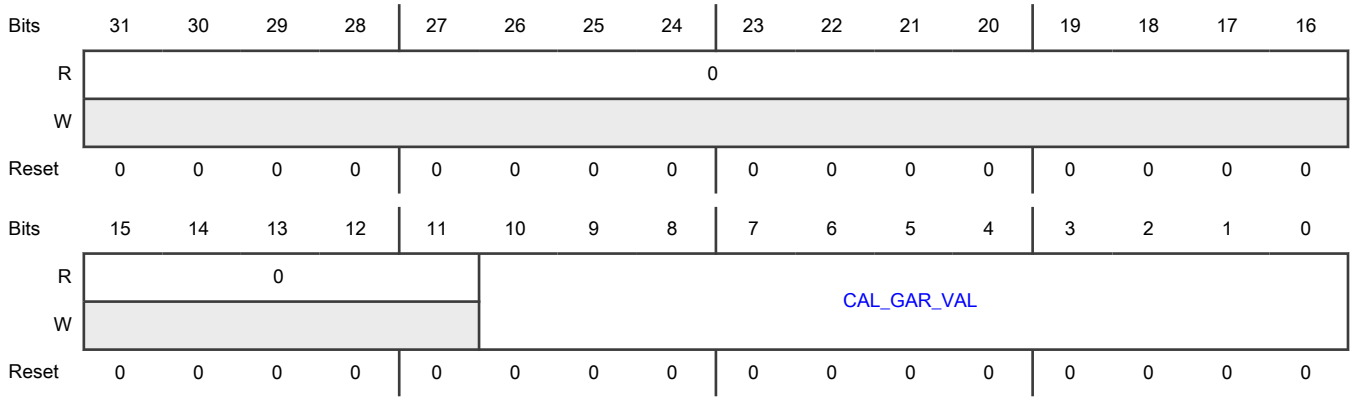
The calibration values in the CAL_GAR registers affect the conversion result by conditionally being subtracted from the conversion before the result is transferred into the FIFOs. Calibration must be run each time ADC is powered down or a hard reset is issued. To reduce the latency required to run calibration, the CAL_GAR values can be stored in non-volatile memory after an initial calibration. These values can be recovered prior to the first ADC conversion. If these registers are set to values not generated by the calibration function, the linearity error specifications may not be met.

The CAL_GAR registers are only read-and-write accessible when ADC is disabled with CTRL[ADCEN] = 0. Access time when writing to these registers is larger than three ADC clock cycles. Wait states are inserted on the bus to meet synchronization timing to the associated CAL_GAR register. The width of each register in this array is non-uniform. The exact width of each register is summarized in Calibration General A-Side and B-Side Widths.

Offset

Register	Offset
CAL_GAR0	400h

Diagram



Fields

Field	Description
31-11 —	Reserved
10-0 CAL_GAR_VAL	Calibration General A Side Register Element

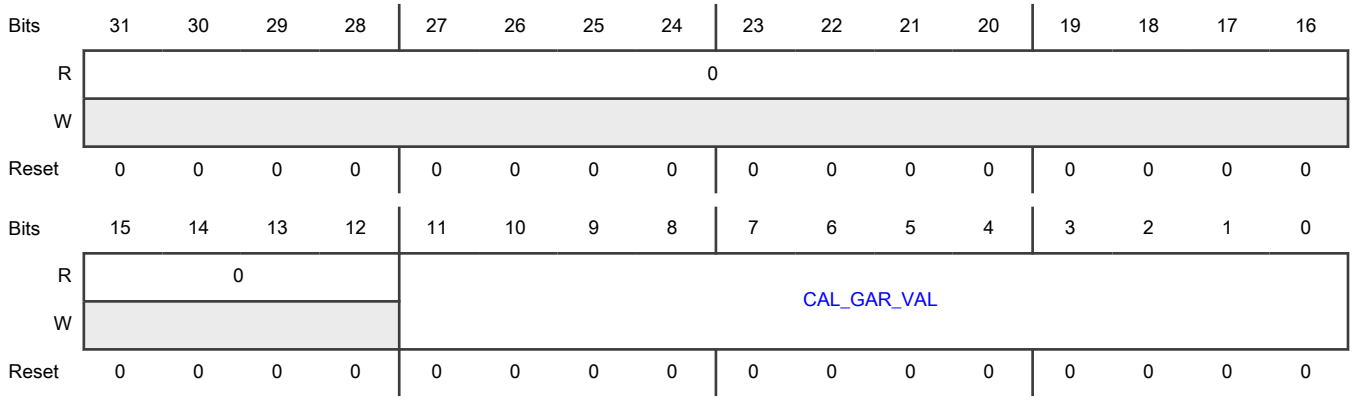
47.6.1.21 Calibration General A-Side Registers (CAL_GAR1)

Calibration register CAL_GAR1. For more detail, see Calibration General A-Side Registers (CAL_GAR0).

Offset

Register	Offset
CAL_GAR1	404h

Diagram



Fields

Field	Description
31-12 —	Reserved
11-0 CAL_GAR_VAL	Calibration General A Side Register Element

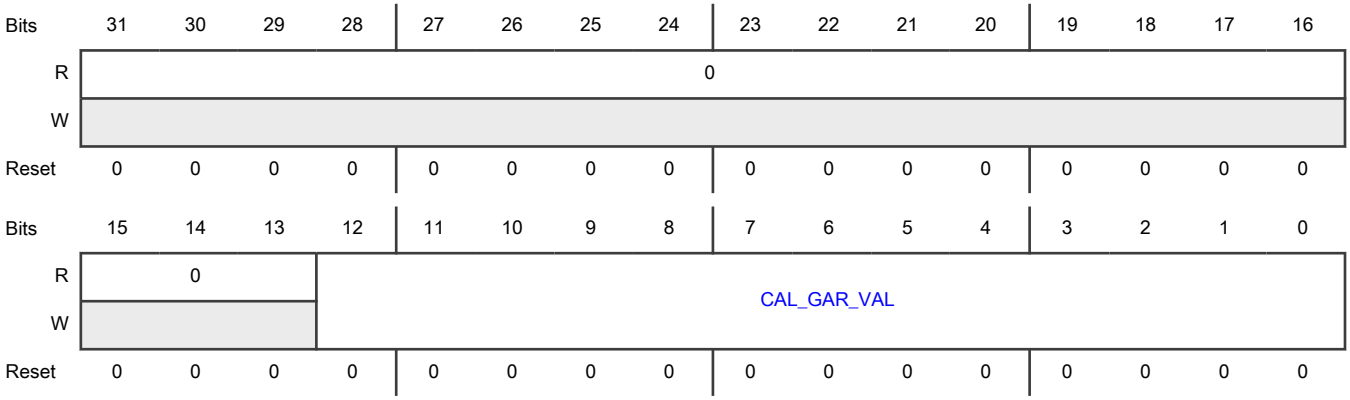
47.6.1.22 Calibration General A-Side Registers (CAL_GAR2 - CAL_GAR3)

Calibration registers CAL_GAR2 and CAL_GAR3. For more detail, see Calibration General A-Side Registers (CAL_GAR0).

Offset

Register	Offset
CAL_GAR2	408h
CAL_GAR3	40Ch

Diagram



Fields

Field	Description
31-13 —	Reserved
12-0 CAL_GAR_VAL	Calibration General A Side Register Element

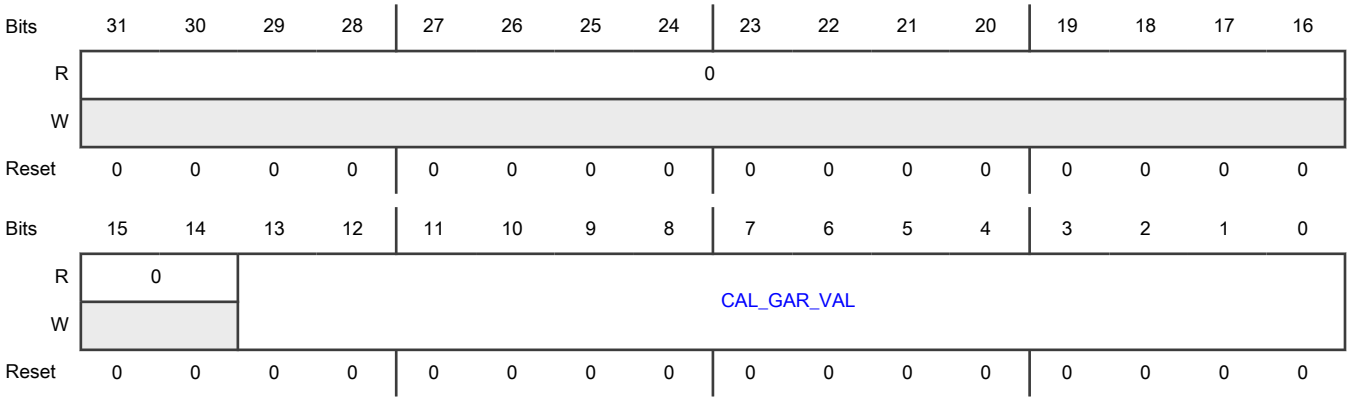
47.6.1.23 Calibration General A-Side Registers (CAL_GAR4 - CAL_GAR7)

Calibration registers CAL_GAR4 through CAL_GAR7. For more detail, see Calibration General A-Side Registers (CAL_GAR0).

Offset

Register	Offset
CAL_GAR4	410h
CAL_GAR5	414h
CAL_GAR6	418h
CAL_GAR7	41Ch

Diagram



Fields

Field	Description
31-14 —	Reserved
13-0 CAL_GAR_VAL	Calibration General A Side Register Element

47.6.1.24 Calibration General A-Side Registers (CAL_GAR8 - CAL_GAR15)

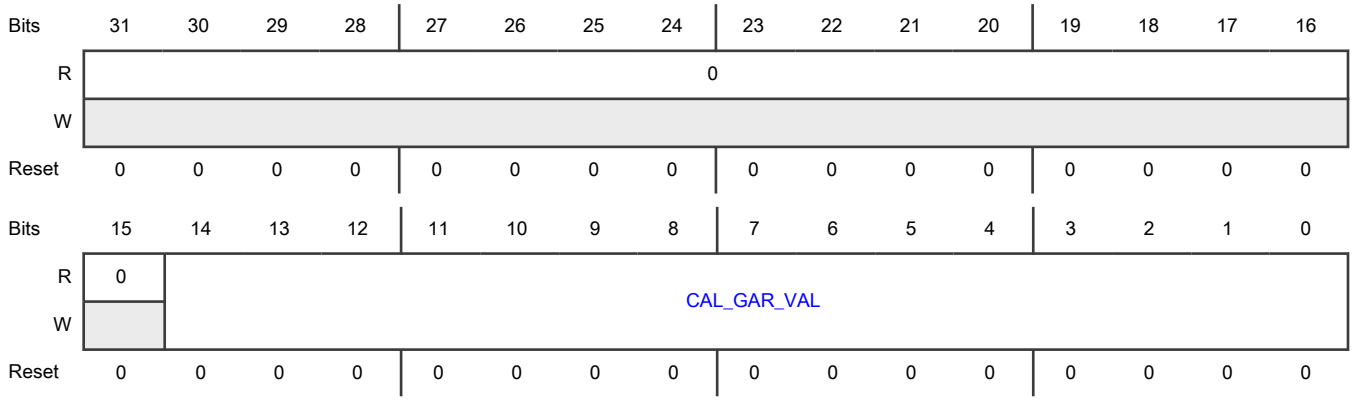
Calibration registers CAL_GAR8 through CAL_GAR15. For more detail, see Calibration General A-Side Registers (CAL_GAR0).

Offset

For a = 8 to 15:

Register	Offset
CAL_GARa	400h + (a × 4h)

Diagram



Fields

Field	Description
31-15 —	Reserved
14-0 CAL_GAR_VAL	Calibration General A Side Register Element

47.6.1.25 Calibration General A-Side Registers (CAL_GAR16 - CAL_GAR31)

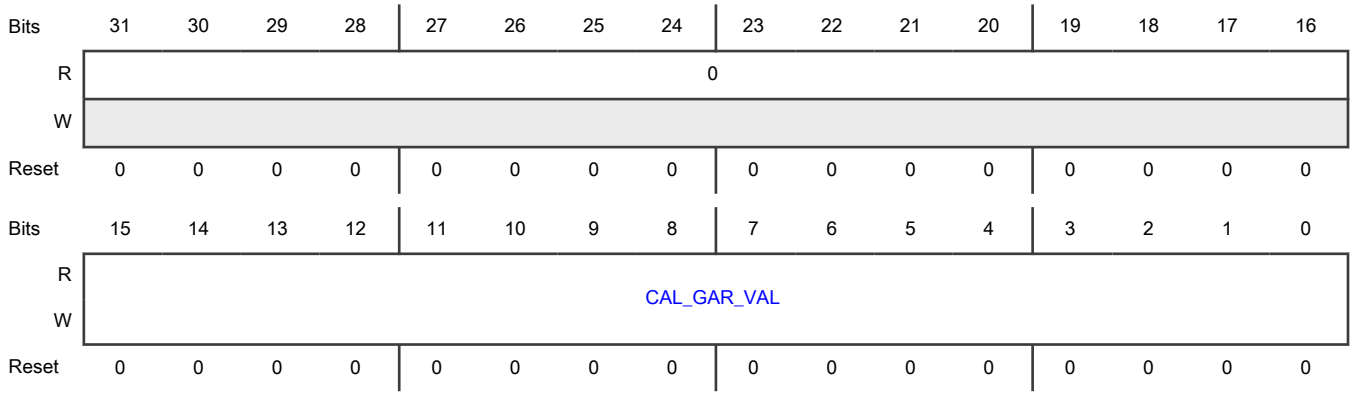
Calibration registers CAL_GAR16 through CAL_GAR31. For more detail, see Calibration General A-Side Registers (CAL_GAR0).

Offset

For a = 16 to 31:

Register	Offset
CAL_GARa	400h + (a × 4h)

Diagram



Fields

Field	Description
31-16 —	Reserved
15-0 CAL_GAR_VAL	Calibration General A Side Register Element

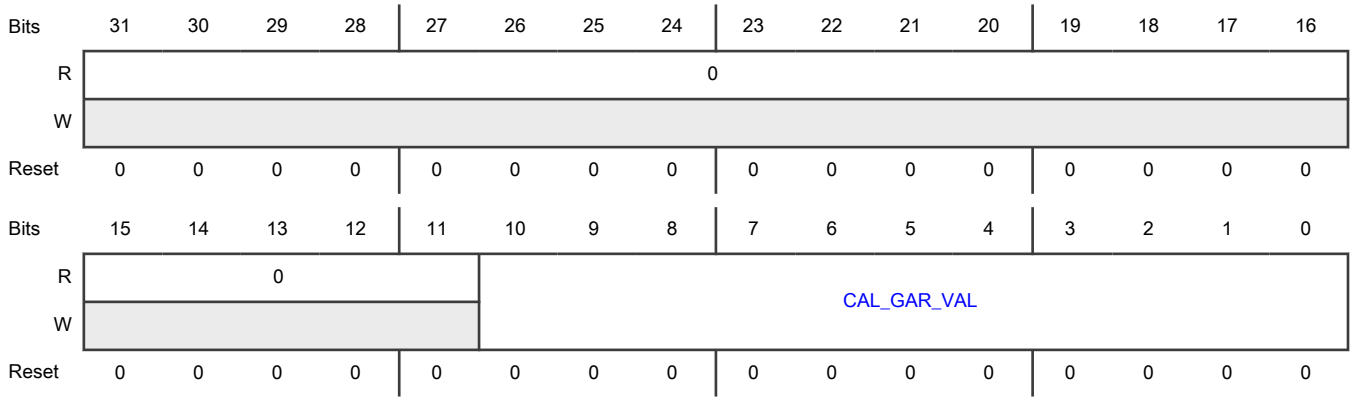
47.6.1.26 Calibration General A-Side Registers (CAL_GAR32)

Calibration register CAL_GAR32. For more detail, see Calibration General A-Side Registers (CAL_GAR0).

Offset

Register	Offset
CAL_GAR32	480h

Diagram



Fields

Field	Description
31-11 —	Reserved
10-0 CAL_GAR_VAL	Calibration General A Side Register Element

47.6.1.27 Calibration General B-Side Registers (CAL_GBR0)

Corrects for linearity errors of the B-side converter. All 33 B-side general calibration value registers (CAL_GBR0-CAL_GBR32) contain calibration information that is automatically updated during the self-calibration sequence. See Calibration functions for more information on completing ADC calibration steps.

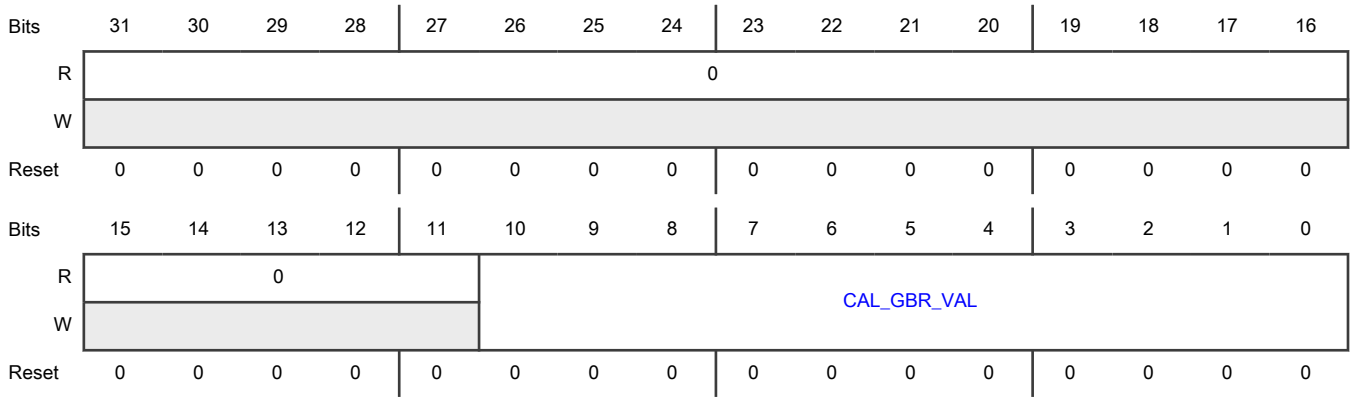
The calibration values in the CAL_GBR registers affect the conversion result by conditionally being subtracted from the conversion before the result is transferred into the FIFOs. Calibration must be run each time ADC is powered down or a hard reset is issued. To reduce the latency required to run calibration, the CAL_GBR values can be stored in non-volatile memory after an initial calibration. These values can be recovered prior to the first ADC conversion. If these registers are set to values not generated by the calibration function, the linearity error specifications may not be met.

The CAL_GBR registers are only read-and-write accessible when ADC is disabled with CTRL[ADCEN] = 0. Access time when writing to these registers is larger than three ADC clock cycles. Wait states are inserted on the bus to meet synchronization timing to the associated CAL_GBR register. The width of each register in this array is non-uniform. The exact width of each register is summarized in Calibration General A-Side and B-Side Widths.

Offset

Register	Offset
CAL_GBR0	500h

Diagram



Fields

Field	Description
31-11 —	Reserved
10-0 CAL_GBR_VAL	Calibration General B Side Register Element

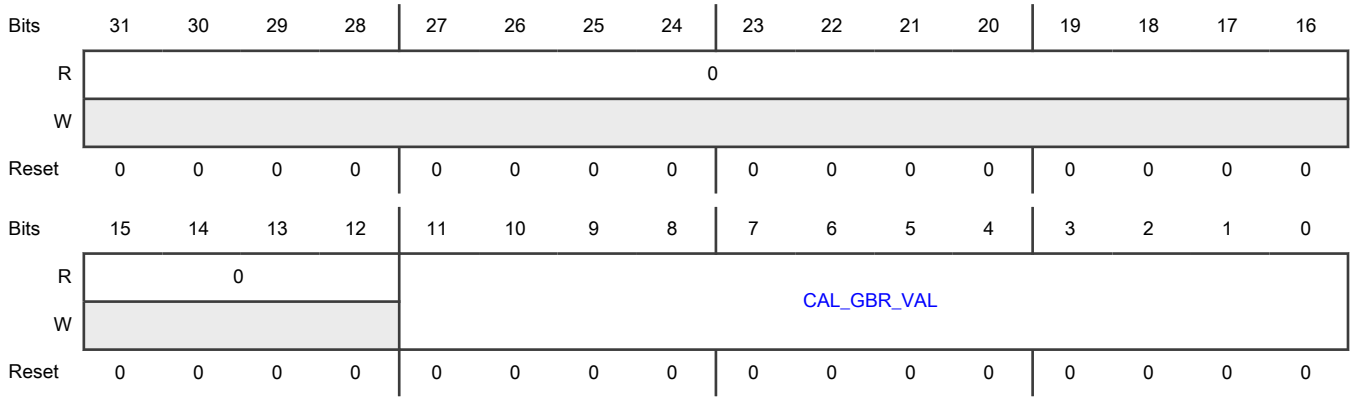
47.6.1.28 Calibration General B-Side Registers (CAL_GBR1)

Calibration register CAL_GBR1. For more detail, see Calibration General B-Side Registers (CAL_GBR0).

Offset

Register	Offset
CAL_GBR1	504h

Diagram



Fields

Field	Description
31-12 —	Reserved
11-0 CAL_GBR_VAL	Calibration General B Side Register Element

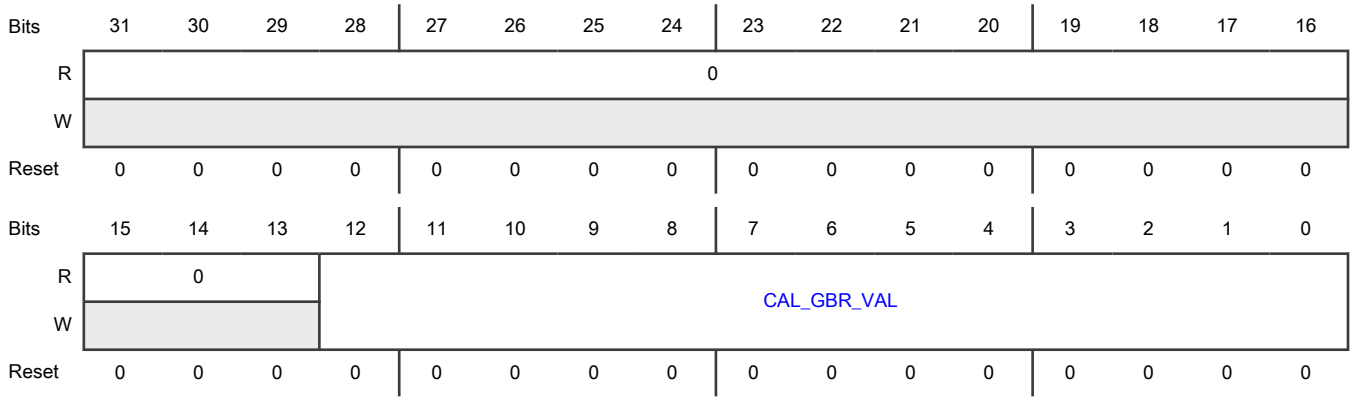
47.6.1.29 Calibration General B-Side Registers (CAL_GBR2 - CAL_GBR3)

Calibration registers CAL_GBR2 and CAL_GBR3. For more detail, see Calibration General B-Side Registers (CAL_GBR0).

Offset

Register	Offset
CAL_GBR2	508h
CAL_GBR3	50Ch

Diagram



Fields

Field	Description
31-13 —	Reserved
12-0 CAL_GBR_VAL	Calibration General B Side Register Element

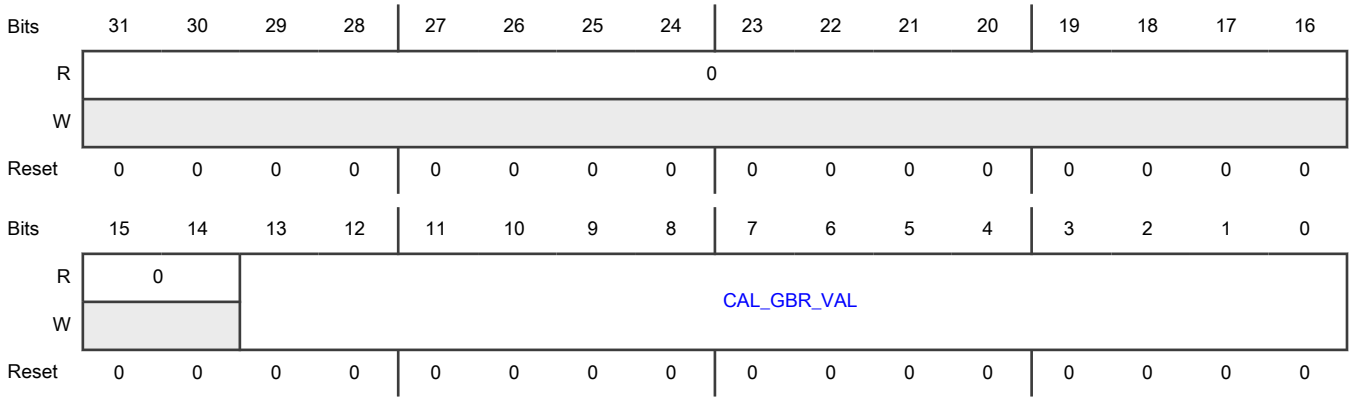
47.6.1.30 Calibration General B-Side Registers (CAL_GBR4 - CAL_GBR7)

Calibration registers CAL_GBR4 through CAL_GBR7. For more detail, see Calibration General B-Side Registers (CAL_GBR0).

Offset

Register	Offset
CAL_GBR4	510h
CAL_GBR5	514h
CAL_GBR6	518h
CAL_GBR7	51Ch

Diagram



Fields

Field	Description
31-14 —	Reserved
13-0 CAL_GBR_VAL	Calibration General B Side Register Element

47.6.1.31 Calibration General B-Side Registers (CAL_GBR8 - CAL_GBR15)

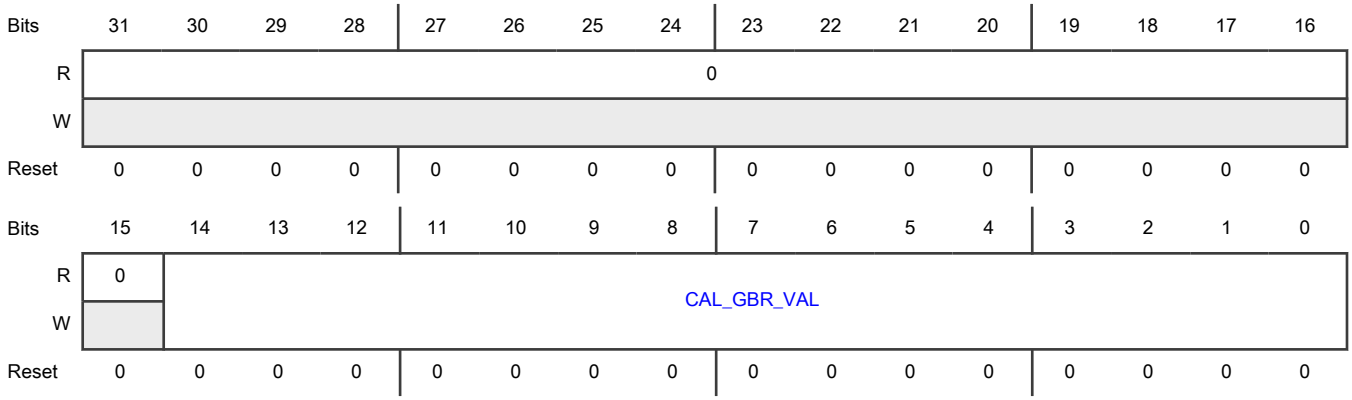
Calibration registers CAL_GBR8 through CAL_GBR15. For more detail, see Calibration General B-Side Registers (CAL_GBR0).

Offset

For a = 8 to 15:

Register	Offset
CAL_GBRa	500h + (a × 4h)

Diagram



Fields

Field	Description
31-15 —	Reserved
14-0 CAL_GBR_VAL	Calibration General B Side Register Element

47.6.1.32 Calibration General B-Side Registers (CAL_GBR16 - CAL_GBR31)

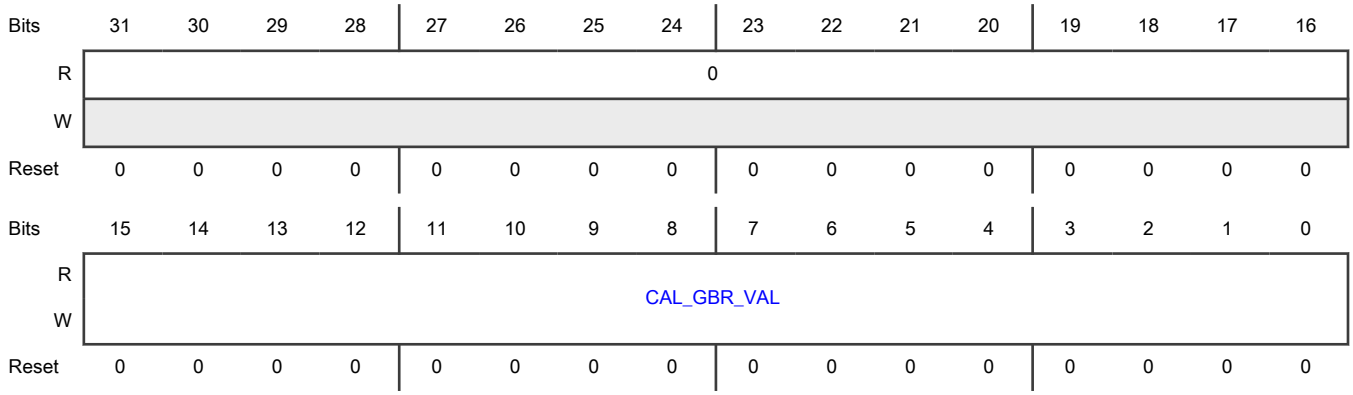
Calibration registers CAL_GBR16 through CAL_GBR31. For more detail, see Calibration General B-Side Registers (CAL_GBR0).

Offset

For a = 16 to 31:

Register	Offset
CAL_GBRa	500h + (a × 4h)

Diagram



Fields

Field	Description
31-16 —	Reserved
15-0 CAL_GBR_VAL	Calibration General B Side Register Element

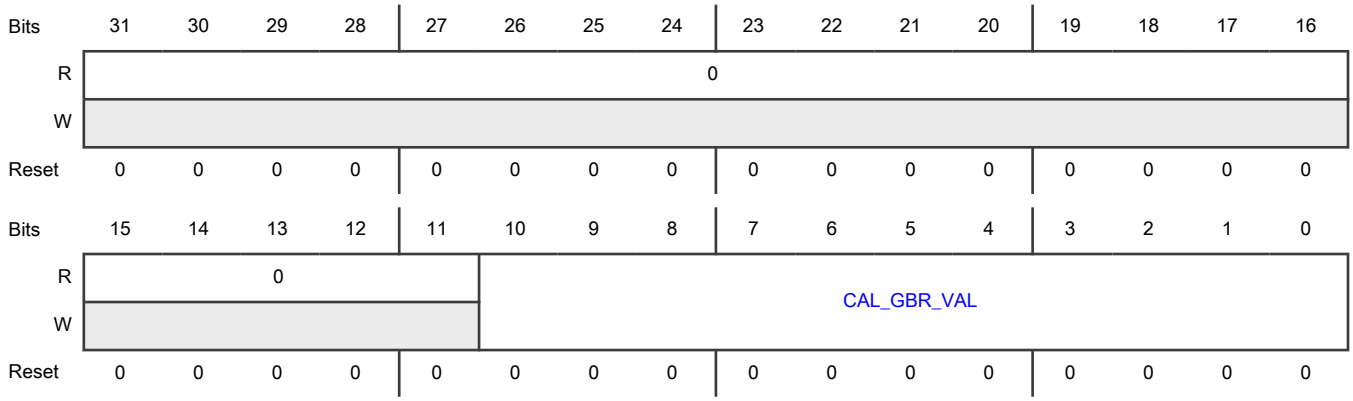
47.6.1.33 Calibration General B-Side Registers (CAL_GBR32)

Calibration register CAL_GBR32. For more detail, see Calibration General B-Side Registers (CAL_GBR0).

Offset

Register	Offset
CAL_GBR32	580h

Diagram



Fields

Field	Description
31-11 —	Reserved
10-0 CAL_GBR_VAL	Calibration General B Side Register Element

Chapter 48
12-bit Digital-to-Analog Converter (DAC)

48.1 Chip-specific DAC information

Table 404. Reference links to related information

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

NOTE

GCR[RCV_TRG] bitfield is not available on this device as multiple DACs are not using the same RCLK.

48.1.1 Module instances

This device has three instances of the DAC module, DAC0, DAC1 and DAC2.

48.1.2 DAC reference options

Table 405. DAC reference options

DACRFS[2:1]	Description
0	VREFH1 = VDDA supply rail
1	VREFH2 = VREF_OUT output
2	VREFH3 = VREFP reference pin

48.1.3 DAC trigger inputs

DAC trigger sources get routed through the Input Multiplexer (INPUTMUX). See the INPUTMUX chapter for available trigger sources. See registers DAC0_TRIG - DAC2_TRIG.

48.1.4 DAC Chip-specific initialization

To initialize the DAC module:

- Set the clock divider for the DAC with DACnCLKDIV[DIV] bits in SYSCON register.
- Enable the clock to the DAC (AHBCLKCTRLn[DACn] = 1) in SYSCON registers. This enables the register interface and the peripheral function clock.
- Enable VREF module (PDRUNCFGCLR0[PDEN_VREF] = 1) in PMC register. DAC PTAT and ZTC current sources comes from VREF module.
- Power up the DAC (PDRUNCFGCLR1[PDEN_DACn] = 1) in PMC register.
- Reset and release the DAC peripheral reset by setting the PRESETCTRLSETn[DACn_RST] bit and then set PRESETCTRLCLRn[DACn_RST] bit.
- Reset the DAC logic by setting and clearing bit RCR[SWRST].

- Reset the DAC FIFO by setting and clearing bit RCR[FIFORST].
- If needed, initialize the DAC reference source before selecting the DAC reference with GCR[DACRFS].
  - If using VREF_OUT as reference, see Initialization/Application Information.
- Configure the DAC as described in section Initialization

48.2 Overview

The 12-bit general-purpose digital-to-analog converter is a low-power DAC. The output of the DAC can be placed on an external pin or set as one of the inputs to the analog comparator or ADC.

48.2.1 Block diagram

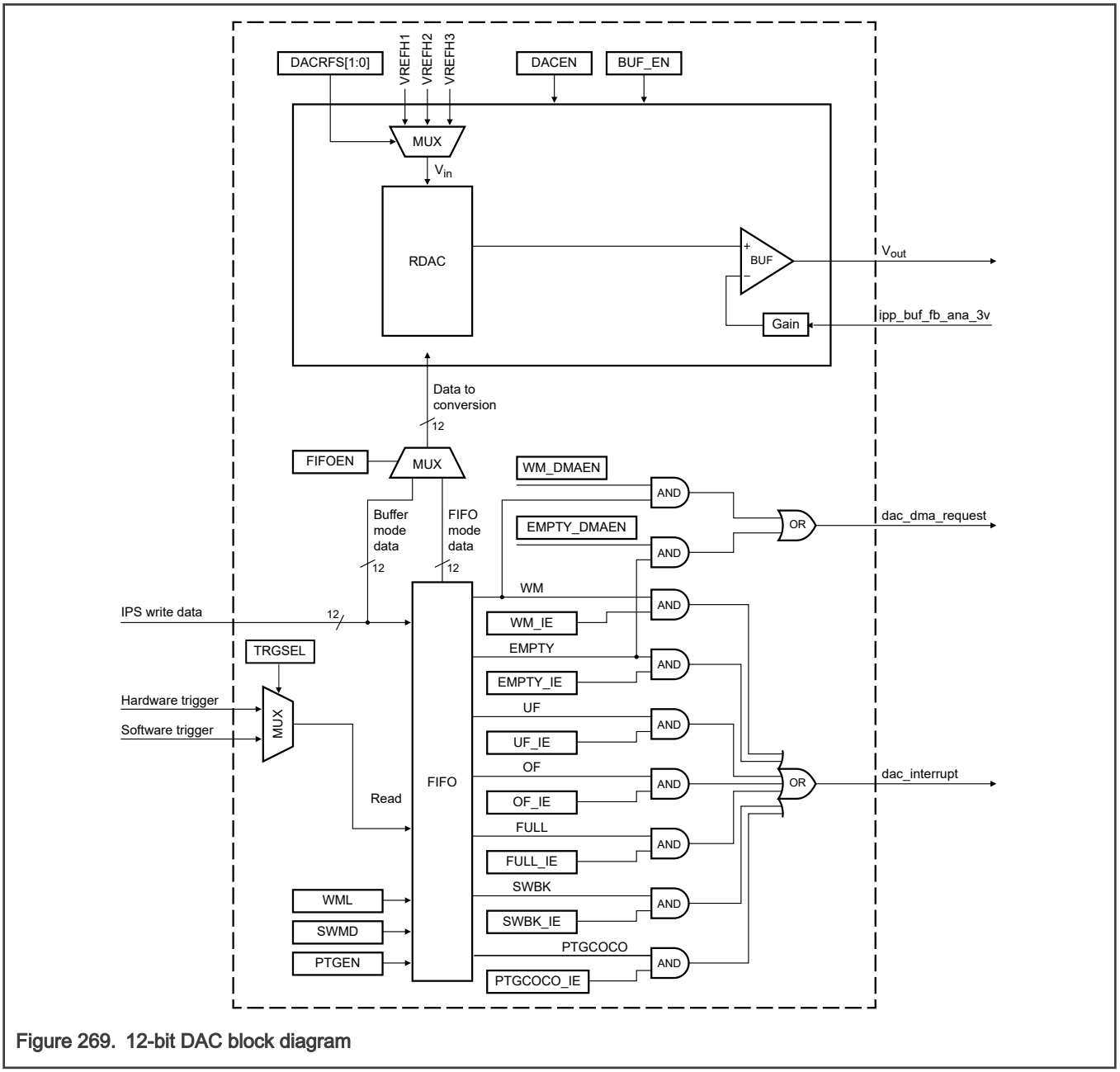


Figure 269. 12-bit DAC block diagram

48.2.2 Features

- Includes an on-chip programmable reference generator output. The voltage output ranges from 1/4096 V_{in} to V_{in}, and the step is 1/4096 V_{in}, where V_{in} is the input reference voltage.
- Selects V_{in} from three reference sources.
- Supports 16-word depth FIFO with the configurable watermark.
- Uses multiple operation modes.
  - Buffer mode
  - FIFO mode
  - Swing Back mode
  - Periodic Trigger mode
- Supports software trigger and hardware trigger.
- Provides selectable performance levels: Low-Power mode(LP) and Lower Low-Power mode(LLP).
- Provides interrupt and DMA support.

48.3 Functional description

12-bit DAC can select one of the three reference inputs: VREFH1, VREFH2 and VREFH3 as the reference voltage, by GCR[DACRFS]. See the chip-specific 12-bit DAC information to determine the source options for VREFH1, VREFH2 and VREFH3.

12-bit DAC converts a 12-bit data to a stepped analog output voltage when enabled. The output voltage range is from V_{in} to V_{in}/4096, and the step is V_{in}/4096.

You can configure the 12-bit DAC to Buffer mode, FIFO mode, Swing Back mode, or Periodic Trigger mode.

NOTE

You must configure 12-bit DAC before any conversion. During conversion or FIFO write operation, you must not change the configuration or disable DACEN. If you switch to another mode, you must first write a 1 to SWRST to reset 12-bit DAC or write 0 to DACEN.

48.3.1 Buffer mode

You can enable Buffer mode with DACEN.

In Buffer mode, any write to DATA pushes the data to the analog conversion without trigger support.

NOTE

The data write speed must not be higher than the analog conversion speed.

48.3.2 FIFO mode

You can enable FIFO mode with FIFOEN and DACEN.

In FIFO mode, for a 32-/16-bit write to DATA, put the data into the FIFO and the write pointer increments automatically. For an 8-bit write, only if you write both the bytes, the write pointer increases. You must ensure that an 8-bit write is done in pair and write both the upper and lower bytes. There is no priority, which byte to write first.

When a software or hardware trigger comes, the read pointer pointed data in FIFO pushes to the analog conversion, and the read pointer increments automatically. If the trigger comes when FIFO is empty, the data to analog remains unchanged, and an underflow error flag sets.

48.3.3 Swing Back mode

You can enable Swing Back mode with FIFOEN, SWMD, and DACEN.

The read pointer swings between the writer pointer and zero in Swing Back mode. The trigger increases the read pointer till the writer pointer and decreases the read pointer till zero, and so on. EMPTY, FULL, and WM have no meaning and you should ignore them during Swing Back mode.

NOTE

In Swing Back mode, the FIFO should first write at least two data; otherwise, this mode will not work correctly. You should not write the FIFO when Swing Back mode is in progress.

48.3.4 Periodic Trigger mode

You can enable Periodic Trigger mode with FIFOEN, PTGEN, and DACEN.

Due to the trigger (hardware or software) synchronization logic, there is a delay time variation between each conversion. If there is a need to trigger DAC periodically, NXP recommends you to use Periodic Trigger mode.

In Periodic Trigger mode, you only send the first trigger. Then after every [PTG_PERIOD+1] read clock (RCLK) cycle, an internal trigger automatically triggers the DAC. There are [PTG_NUM] internal triggers, therefore, in total [PTG_NUM+1] conversions, including the first trigger that you send. Write 0 to PTGEN to terminate the current conversion queue. Then, after the current conversion completes, the conversion terminates, and the PTGCOCO sets. If PTG_NUM becomes zero, the infinite triggers follow the first hardware/software trigger until you write 0 to PTGEN. In any case, FIFORST or SWRST can terminate the conversion.

During the periodic trigger mode conversion, you can ignore the additional hardware/software triggers. You can start another queue after PTGCOCO becomes 1.

During Periodic Trigger Mode, WM feeds data to FIFO to avoid underflow.

You can combine Periodic Trigger mode with SWMD to periodically trigger DAC and swing back.

48.3.5 Low-power mode operation

If the RCLK remains enabled, 12-bit DAC can be functional with FIFO mode or Swing Back mode or Periodic Trigger mode during Stop and Wait modes, using the hardware trigger initiating the conversion.

48.3.6 DMA

After the corresponding DMA enable register field becomes 1, you can automatically generate DMA requests when WM or EMPTY becomes 1. When you enable DMA, the error flags can only generate an interrupt.

48.3.7 Clocks

12-bit DAC block requires the following clock(s):

- Bus clock for register access and writing data to FIFO or Buffer.
- Read clock to push data from FIFO or Buffer to the analog DAC.

48.3.8 Resets

The following are the 12-bit DAC resets:

1. Global reset fed into the block. This global reset resets everything.
2. SWRST: resets all 12-bit DAC registers and internal logic.
3. FIFORST: resets the FIFO pointers in DAC FIFO Pointer (FPR) and flags in FIFO Status (FSR).

48.3.9 Interrupts

After the corresponding interrupt enable register field becomes 1, warning flags and error flags can generate interrupt. The warning flags are EMPTY, FULL, WM, SWBK, and PTGCOCO. The error flags are OF and UF.

48.4 DAC external signal descriptions

Table 406. DAC external signal descriptions

Signal	Description	I/O
VREFH1	DAC reference high voltage 1	I
VREFH2	DAC reference high voltage 2	I
VREFH3	DAC reference high voltage 3	I
Vout	DAC output	O

48.5 Initialization

To initialize 12-bit DAC:

1. Configure the read clock at chip level.
2. Configure Work mode and GCR enables 12-bit DAC.
3. Configure Interrupt Enable (IER) if required.
4. Configure DMA Enable (DER) if required.
5. Configure Periodic Trigger Control (PCR) if required.
6. Write data to DATA to fill the FIFO, if FIFO mode.
7. Wait analog start-up time
8. Use hardware or software trigger to push data to the analog DAC to convert, if FIFO mode.
9. Write data directly to DATA to push data to the analog DAC to convert, if Buffer mode.

48.6 Application Information

Digital delay

Due to the resynchronization and data latch process, there is a delay between the trigger posedge and analog voltage settling. The delay is (2+ LATCH_CYC)*T to (3+ LATCH_CYC)*T, where T is the RCLK period, and LATCH_CYC*T should not be smaller than 40 ns. So the delay has one T uncertainty. The fast RCLK can achieve a small delay.

RCLK frequency

For discrete conversion, the RCLK frequency has no low limit. But slow RCLK involves large digital delay and uncertainty.

For continuous conversion, the RCLK frequency should be at least 5 times the trigger speed.

For continuous conversion, due to the one cycle uncertainty RCLK, the trigger period should not less than (1000ns + T). Where 1000 ns period means the maximum analog conversion speed 1 Msps.

To achieve the full speed of DAC analog, you must configure to the internal periodic trigger mode.

For Internal Periodic Trigger mode, RCLK frequency should be at least two times the conversion speed. For example, to achieve 1 Msps conversion speed, at least 2 MHz RCLK requires in Periodic Trigger mode.

48.7 Memory map and register definition

All registers are accessible in 8-, 16-, or 32-bit accesses. For efficiency, 32-bit accesses are recommended. Access to an address outside the valid memory map generates a bus error.

48.7.1 12-bit DAC register descriptions

Access to an address outside the valid memory map generates a bus error.

48.7.1.1 DAC memory map

DAC0 base address: 400B_2000h

DAC1 base address: 400B_6000h

DAC2 base address: 400B_9000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Version Identifier (VERID)	32	RO	0000_0000
4	Parameter (PARAM)	32	See section	0000_0003
8	Data (DATA)	32	See section	0000_0000
C	Global Control (GCR)	32	See section	0000_0100
10	DAC FIFO Control (FCR)	32	See section	0000_0000
14	DAC FIFO Pointer (FPR)	32	See section	0000_0000
18	FIFO Status (FSR)	32	See section	0000_0002
1C	Interrupt Enable (IER)	32	See section	0000_0000
20	DMA Enable (DER)	32	See section	0000_0000
24	Reset Control (RCR)	32	See section	0000_0000
28	Trigger Control (TCR)	32	See section	0000_0000
2C	Periodic Trigger Control (PCR)	32	RW	0000_0000

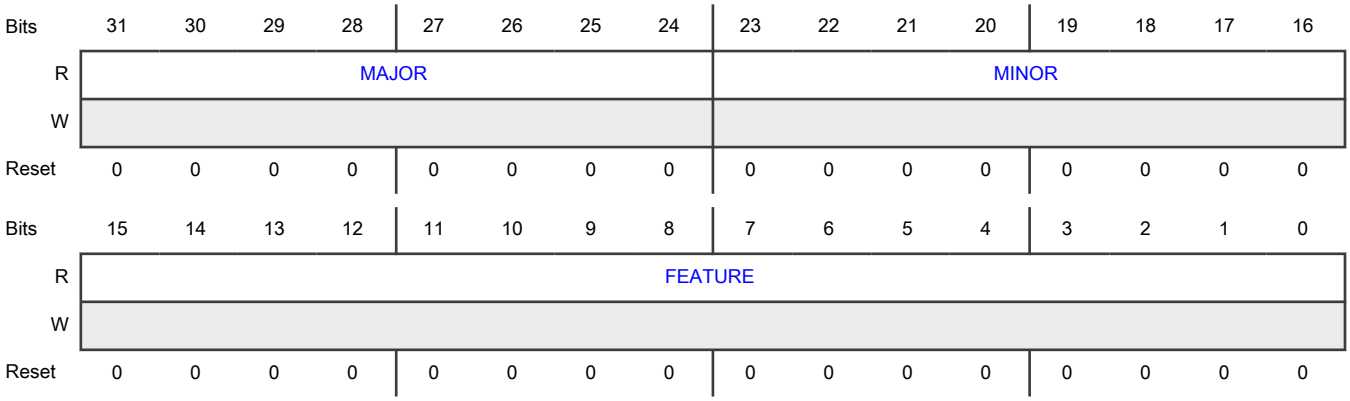
48.7.1.1.1 Version Identifier (VERID)

Indicates the version integrated for this instance on the chip.

Offset

Register	Offset
VERID	0h

Diagram



Fields

Field	Description
31-24 MAJOR	Major Version Number Returns the major version number for the module specification.
23-16 MINOR	Minor Version Number Returns the minor version number for the module specification.
15-0 FEATURE	Feature Identification Number Returns the feature set number.

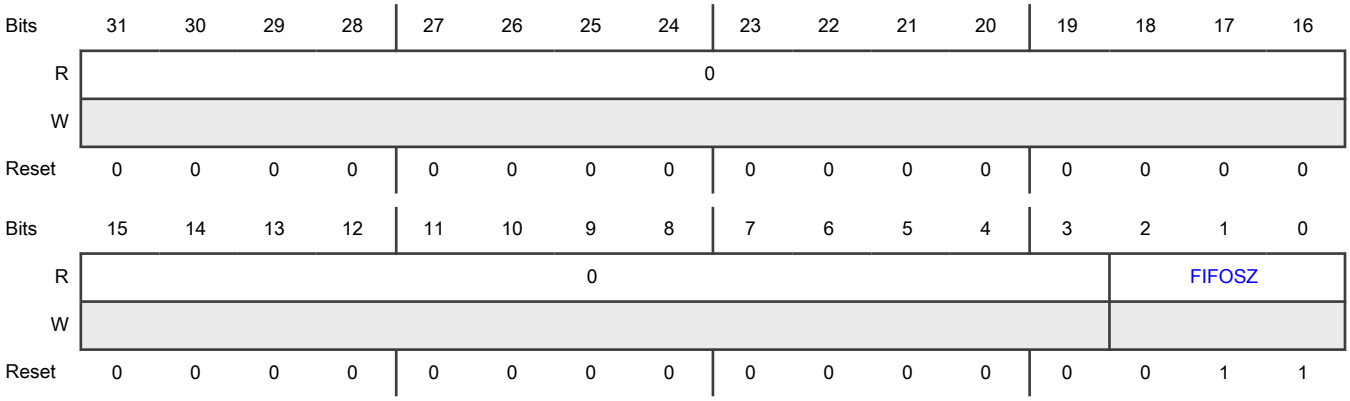
48.7.1.1.2 Parameter (PARAM)

Indicates the feature parameters for this instance on the chip.

Offset

Register	Offset
PARAM	4h

Diagram



Fields

Field	Description
31-3 —	Reserved
2-0 FIFOSZ	FIFO Size Indicates that the FIFO depth is 2^(FIFOSZ+1) words. 000 - Reserved 001 - FIFO depth is 4 010 - FIFO depth is 8 011 - FIFO depth is 16 100 - FIFO depth is 32 101 - FIFO depth is 64 110 - FIFO depth is 128 111 - FIFO depth is 256

48.7.1.1.3 Data (DATA)

Holds the FIFO entry in FIFO mode and serves as the buffer in Buffer mode.

Offset

Register	Offset
DATA	8h

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											
W					DATA											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-12 —	Reserved
11-0 DATA	FIFO Entry or Buffer Entry Specifies that in FIFO mode, Swing Back mode, or Periodic Trigger mode, this is the FIFO data entry. In Buffer mode, write to this field pushes the data to analog without trigger support.

48.7.1.1.4 Global Control (GCR)

Contains configuration options for DAC function, such as mode select, trigger select, and so on. It also includes the reference current select for the analog DAC.

Offset

Register	Offset
GCR	Ch

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0								BUF_SPD...	0	IREF_ZT...	IREF_PT...	0		BUF_EN	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				LATCH_CYC				0	PTGEN	TRGSEL	SWMD	FIFOEN	DACRFS		DACEN
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Fields

Field	Description
31-24 —	Reserved
23 BUF_SPD_CTRL	OPAMP as Buffer, Speed Control Signal **NOTE** See the DAC electrical characteristics of the chip datasheet for more information on the impact of the modes below. 0 - Lower Low-Power mode 1 - Low-Power mode
22 —	Reserved
21 IREF_ZTC_EXT_SEL	External On-Chip ZTC Current Reference Select Selects the external ZTC current reference, which is from the VREF module. See the Chip-specific section for more information. 0 - Not selected 1 - Selected
20 IREF_PTAT_EXT_SEL	External On-Chip PTAT Current Reference Select Selects the external PTAT current reference, which is from the VREF module. See the Chip-specific section for more information. 0 - Not selected 1 - Selected
19-18 —	Reserved
17 BUF_EN	Buffer Enable Uses OPAMP as the DAC analog buffer. 0 - Not used 1 - Used
16-12 —	Reserved
11-8 LATCH_CYC	RCLK Cycles Before Data Latch Configures the DAC sync cycles, which reduces the glitch on the output. The sync time is (LATCH_CYC+1) RCLK cycles. You must configure this register per the RCLK frequency. The recommended sync time is at least 40 ns.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	The recommended LATCH_CYC configuration is: Sync time is (LATCH_CYC+1) RCLK cycles, when (25×LATCH_CYC) MHz < RCLK ≤ (25×LATCH_CYC+25) MHz. Example 1, if f_{RCLK} is 48 MHz (period is 20.8 ns), you can configure LATCH_CYC to 0x1 to generate a 41.6 ns sync time. Example 2, if f_{RCLK} is 6 MHz (period is 166.7 ns), you can configure LATCH_CYC to 0x0 to generate a 166.7 ns sync time. See the "Clocking" chapter or chip-specific DAC information for more on the RCLK frequency. **NOTE** 12-bit DAC adds (LATCH_CYC+1) RCLK cycles before the analog starts to set up the output. Therefore, when you write to this field, NXP recommends using the smallest possible value that still meets the requirement of 40 ns sync time.
7 —	Reserved
6 PTGEN	DAC Periodic Trigger Mode Enable Enables Periodic Trigger Mode when FIFOEN becomes 1. 0 - Disables 1 - Enables
5 TRGSEL	DAC Trigger Select Selects the trigger source for FIFO mode. 0 - Hardware trigger 1 - Software trigger
4 SWMD	Swing Back Mode Specifies whether Swing Back mode is enabled. Selects the Swing Back mode when FIFOEN becomes 1. 0 - Disables 1 - Enables
3 FIFOEN	FIFO Enable Selects the FIFO mode or Buffer mode. 0 - Enables FIFO mode and disables Buffer mode. Any data written to goes to buffer then goes to conversion. 1 - Enables FIFO mode. Data will be first read from FIFO to buffer and then goes to conversion.
2-1 DACRFS	DAC Reference Select Selects the source of reference voltage high.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	NOTE Refer to table DAC reference options in chip specific chapter for the bit field values.
0 DACEN	DAC Enable Enables the DAC system. Starts the programmable reference generator operation. **NOTE** For an 8-bit or 16-bit register write, enable this field after configuring other bytes of GCR. You must configure Periodic Trigger Control (PCR) and DAC FIFO Control (FCR) before enabling this field. 0 - Disables 1 - Enables

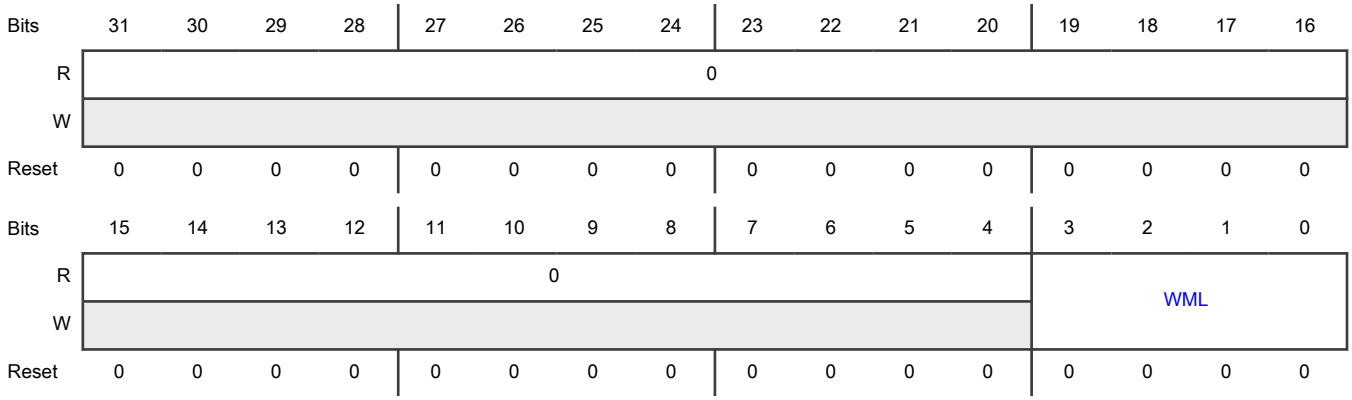
48.7.1.1.5 DAC FIFO Control (FCR)

Contains watermark level control for the FIFO.

Offset

Register	Offset
FCR	10h

Diagram



Fields

Field	Description
31-4	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
3-0 WML	Watermark Level Specifies that when the number of data that remains in FIFO is equal to or less than this level, WM becomes 1. **NOTE** To enable the watermark feature, do not write 0 to this field.

48.7.1.1.6 DAC FIFO Pointer (FPR)

Contains the read pointer and write pointer of the FIFO.

Offset

Register	Offset
FPR	14h

Diagram

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

Fields

Field	Description
31-20 —	Reserved
19-16 FIFO_WPT	FIFO Write Pointer Shows the current write pointer of the FIFO.
15-4	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
3-0 FIFO_RPT	FIFO Read Pointer Shows the current read pointer of the FIFO.

48.7.1.1.7 FIFO Status (FSR)

Contains the status flags of the FIFO.

Offset

Register	Offset
FSR	18h

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							PTGCOCO	UF	OF	0		SWBK	WM	EMPTY	FULL
W								W1C	W1C	W1C			W1C			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Fields

Field	Description
31-9 —	Reserved
8 PTGCOCO	Period Trigger Mode Conversion Complete Flag Indicates that PTG mode conversion is completed. This flag becomes 0 by writing a 1 to it. 0 - Not completed or not started 1 - Completed
7 UF	FIFO Underflow Flag

Table continues on the next page...

Table continued from the previous page...

Field	Description
	Specifies that there is a new trigger after EMPTY becomes 1. The FIFO read pointer do not increase in this case and the data sent to DAC analog conversion will not changed. This flag becomes 0 by writing a 1 to it. 0 - No underflow has occurred since the last time the flag was cleared 1 - At least one trigger underflow has occurred since the last time the flag was cleared
6 OF	FIFO Overflow Flag Indicates that data is intended to write into FIFO after FULL becomes 1. The writer pointer do not increase in this case. Do not write the extra data into the FIFO. This flag clears by writing a 1 to it. 0 - No overflow has occurred since the last time the flag was cleared 1 - At least one FIFO overflow has occurred since the last time the flag was cleared
5-4 —	Reserved
3 SWBK	Swing Back One Cycle Complete Flag Indicates that the DAC has completed one period of conversion in Swing Back mode. The read pointer increases to the top (write pointer) once and then decreases to zero. For example, after writing three data to FIFO, the writer pointer is now 3. Then, if continually triggered, the read pointer will swing like: 0-1-2-1-0-1-2-, and so on. After the fourth trigger, the flag sets. Write a 1 to clear this flag. 0 - No swing back cycle has completed since the last time the flag was cleared 1 - At least one swing back cycle has occurred since the last time the flag was cleared
2 WM	FIFO Watermark Status Flag Write 1 to this field if the remaining data in FIFO is less than or equal to WML. By writing data into FIFO by DMA or CPU, this flag clears automatically when the data in FIFO is more than WML. 0 - Data in FIFO is more than watermark level 1 - Data in FIFO is less than or equal to watermark level
1 EMPTY	FIFO Empty Flag Indicates whether the FIFO is empty or not. 12-bit DAC automatically clears this flag when CPU or DMA writes data to the FIFO. 0 - Not empty 1 - Empty
0 FULL	FIFO Full Flag Indicates whether the FIFO is full or not. FIFO read with software trigger or hardware trigger automatically clears this flag. 0 - Not full 1 - Full

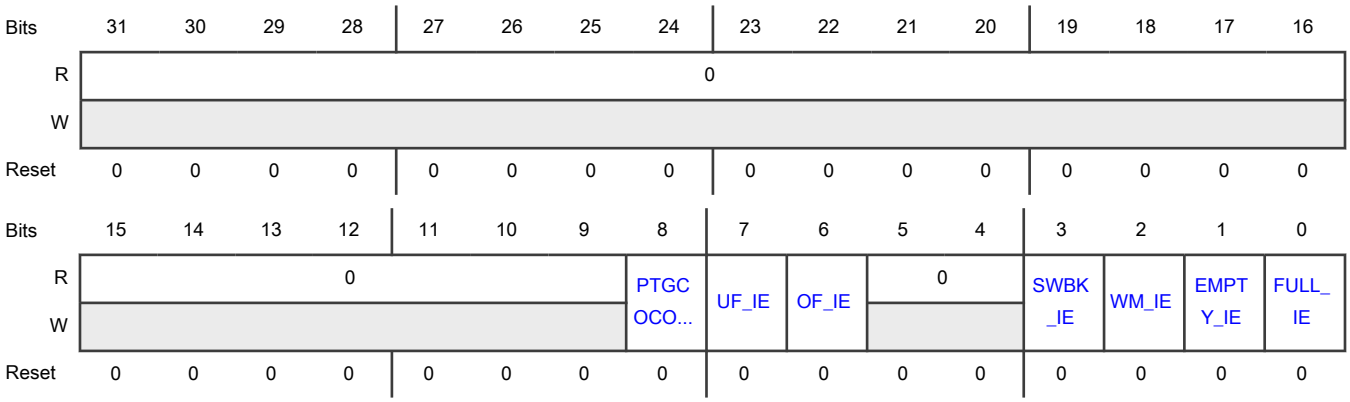
48.7.1.1.8 Interrupt Enable (IER)

Contains the interrupt enable fields.

Offset

Register	Offset
IER	1Ch

Diagram



Fields

Field	Description
31-9 —	Reserved
8 PTGCOCO_IE	PTG Mode Conversion Complete Interrupt Enable Enables interrupt after the PTG mode conversion completes. 0 - Disables 1 - Enables
7 UF_IE	FIFO Underflow Interrupt Enable Enables interrupt when FIFO underflows. 0 - Disables 1 - Enables
6 OF_IE	FIFO Overflow Interrupt Enable Enables interrupt when FIFO overflows. 0 - Disables 1 - Enables
5-4	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
3 SWBK_IE	Swing Back One Cycle Complete Interrupt Enable Enables interrupt after the swing back one cycle completes. 0 - Disables 1 - Enables
2 WM_IE	FIFO Watermark Interrupt Enable Enables interrupt when data in FIFO is less than or equal to watermark level. 0 - Disables 1 - Enables
1 EMPTY_IE	FIFO Empty Interrupt Enable Enables interrupt when FIFO is empty. 0 - Disables 1 - Enables
0 FULL_IE	FIFO Full Interrupt Enable Enables interrupt when FIFO is full. 0 - Disables 1 - Enables

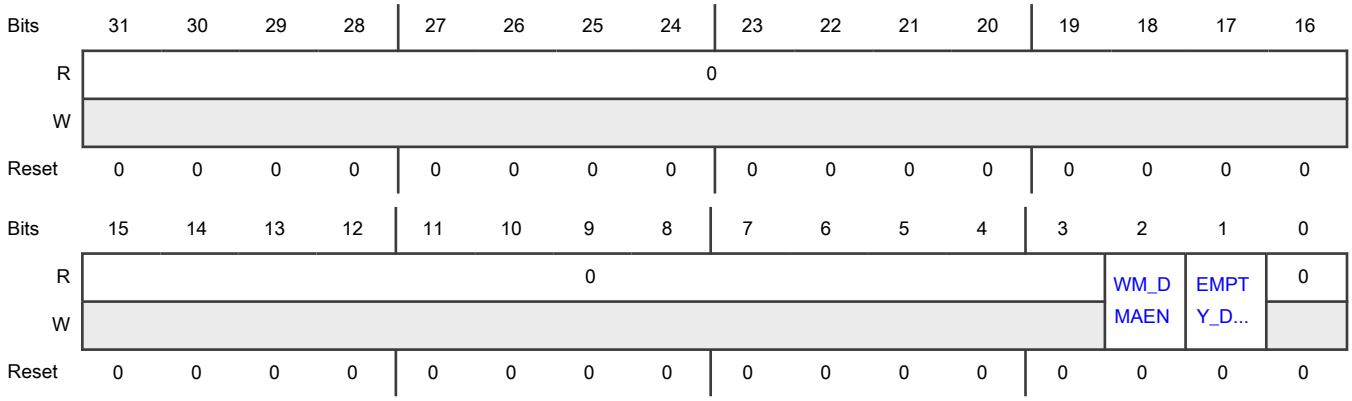
48.7.1.1.9 DMA Enable (DER)

Contains the DMA enable fields.

Offset

Register	Offset
DER	20h

Diagram



Fields

Field	Description
31-3 —	Reserved
2 WM_DMAEN	FIFO Watermark DMA Enable Enables DMA request when data in FIFO is less than or equal to watermark level. 0 - Disables 1 - Enables
1 EMPTY_DMAEN	FIFO Empty DMA Enable Enables DMA request when FIFO is empty. 0 - Disables 1 - Enables
0 —	Reserved

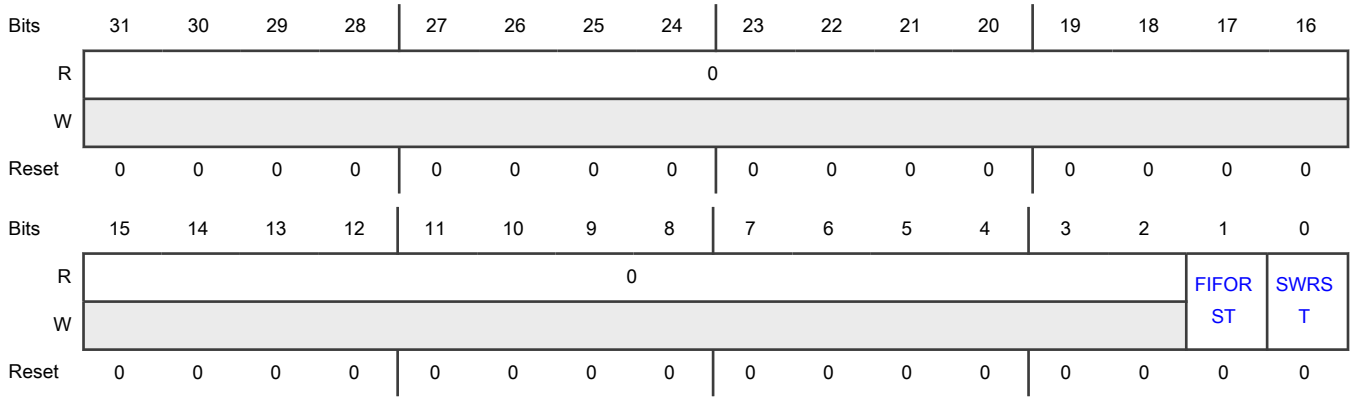
48.7.1.1.10 Reset Control (RCR)

Contains the software reset and FIFO reset fields.

Offset

Register	Offset
RCR	24h

Diagram



Fields

Field	Description
31-2 —	Reserved
1 FIFORST	FIFO Reset Resets the FIFO pointers and flags in FIFO Status (FSR). Remains 1 until you write 0 to it. 0 - No effect 1 - FIFO reset
0 SWRST	Software Reset Resets all DAC registers and internal logic. Remains 1 until you write 0 to it. 0 - No effect 1 - Software reset

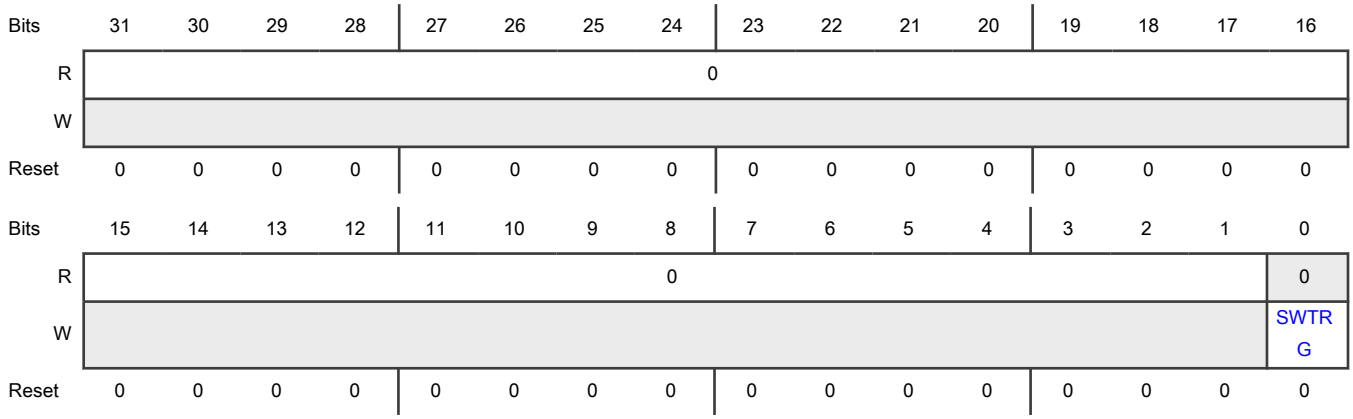
48.7.1.1.11 Trigger Control (TCR)

Contains the software trigger field.

Offset

Register	Offset
TCR	28h

Diagram



Fields

Field	Description
31-1 —	Reserved
0 SWTRG	Software Trigger Indicates whether the DAC soft trigger is valid. Active high. This is a write-only field, which always reads 0. If TRGSEL is 1 and FIFO is enabled and not empty, writing 1 to this field advances the FIFO read pointer once. **NOTE** During continuous conversions, do not adopt hardware trigger and software trigger in a mixed use. 0 - Not valid 1 - Valid

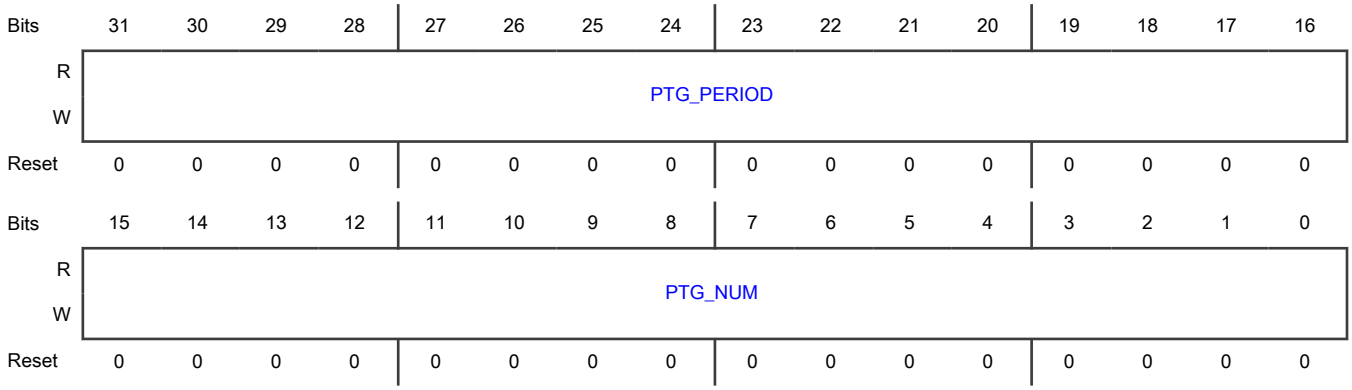
48.7.1.1.12 Periodic Trigger Control (PCR)

Contains the periodic trigger period and periodic trigger number fields.

Offset

Register	Offset
PCR	2Ch

Diagram



Fields

Field	Description
31-16 PTG_PERIOD	Periodic Trigger Period Width Uses only in Periodic Trigger mode. It controls the periodic trigger frequency. There are [PTG_PERIOD+1] RCLK cycles between each periodic trigger. See the chip-specific DAC information for more on the RCLK frequency. Example 1, if f_{RCLK} is 48 MHz and the conversion speed is 0.5 MHz, there should be 96 RCLK cycles between each periodic trigger. So you must configure the PTG_PERIOD as 0x5F(95). Example 2, if f_{RCLK} is 6 MHz and the conversion speed is 0.5 MHz, there should be 12 RCLK cycles between each periodic trigger. So you must configure the PTG_PERIOD as 0xB(11). **NOTE** The PTG_PERIOD should not be less than 0x1. **NOTE** The periodic trigger frequency should not be larger than the analog conversion speed.
15-0 PTG_NUM	Periodic Trigger Number Uses only in Periodic Trigger mode. There are [PTG_NUM] internal triggers following the first hardware/software trigger. So, there are [PTG_NUM+1] conversions in total. **NOTE** If you write 0 to it, there will be infinite triggers following the first hardware/software trigger, until PTGEN becomes 0.

Chapter 49
Analog Comparator (ACMP)

49.1 Chip-specific Analog comparator information

Table 407. Reference links to related information

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

49.2 Overview

This section provides an introduction to the Analog Comparator module.

49.2.1 Block diagram

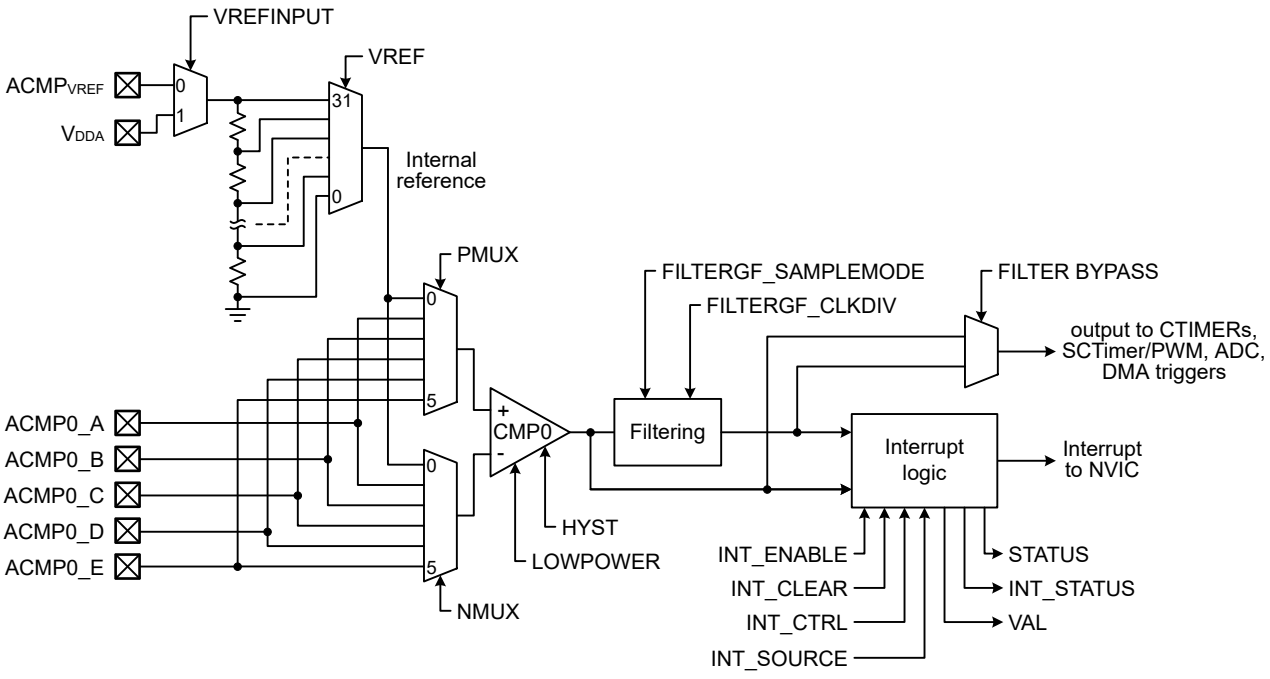


Figure 270. Analog comparator block diagram

49.2.2 Features

- Selectable external inputs can be used as either the positive or negative input of the comparator.

- Voltage ladder source selectable between the supply, multiplexing between internal VDDA and ACMP0VREF.
- Voltage ladder can be separately powered down when not required (vref_int block is automatically enabled - comp_vref_enable = 1 - as soon as PMUX or NMUX input 0 is selected).
- 32-stage voltage ladder can be used as either the positive or negative input of the comparator.
- Interrupt capability. Can be a wake up source in deep-sleep and power-down low power modes.

49.3 Functional description

The analog comparator can compare voltage levels on external pins and internal voltages.

The comparator has 5 inputs multiplexed separately to its positive and negative inputs. The multiplexers are controlled by the comparator register. See Memory Map and register definition.

49.3.1 Comparator modes

The analog comparator supports both Standard and Low Power mode. Comparator mode can be selected by LOWPOWER value.

In Standard mode (LOWPOWER = 0), comparator delay is typically 15 ☐s in low overdrive configuration (inputs voltage difference of 10 mV). Overdrive mode refers to the comparator input voltage difference.

In Low Power mode (LOWPOWER = 1), the comparator current consumption can be reduced to 360 nA (typical, in case voltage ladder source is not selected) at the expense of higher comparator delay (95 us typical in low overdrive configuration). This last mode is suitable for IC low power modes.

Typical comparator delay is 10 us in large overdrive mode with any LOWPOWER setting.

49.3.2 Comparator outputs

The comparator output can be routed to an external pin.The comparator can be used with the bus clock disabled, see Memory map and register definition to save power if the control registers are not required to be written.

The status of the comparator output can be observed through the comparator status register bit (COMP_INT_STATUS). Comparator outputs are connected to the I/O pad and can also be used as trigger inputs to various on-chip peripherals (for example, CTimers, SCTimer/PWM, ADC, DMA controllers).

49.3.3 Settling times

After the voltage ladder is powered on, it requires stabilization time until comparisons using it are accurate. Much shorter settling times apply after the VREFINPUT value is changed and when either or both voltage sources are changed. Software can deal with these factors by repeatedly reading the comparator output until a number of readings yield the same result.

49.3.4 Reference voltages

The voltage ladder can use two reference voltages (VBAT_PMU or ACMPVREF). The voltage ladder selects one of 32 steps between the pin voltage and V_{ss} inclusive.

49.3.5 Interrupts

Interrupt management is set with COMP_INT_CTRL and COMP_INT_STATUS registers.

The interrupt output comes from edge detection circuitry in this module. Rising edges, falling edges, or both edges can be set in the INT_CTRL field. Interrupt requests are cleared when software writes a 1 to INT_CLEAR. The source can also be selected with INT_SOURCE to use a filtered or unfiltered comparator output.

49.4 Signals

The analog comparator reference voltage, the inputs, and the output are assigned to external pins through IOCON. The comparator inputs and the reference voltage are fixed-pin functions that must be enabled through IOCON and can only be assigned to special pins on the package.

See General Purpose I/O (GPIO) to assign the analog comparator output to any pin.

See the table below to enable the analog comparator inputs and the reference voltage input.

Table 408. Pin description

Function	Type	Pin	Description	SWM register
CMP0_A	I	PIO0_0	Comparator input A	COMP
CMP0_B	I	PIO0_9	Comparator input B	COMP
CMP0_C	I	PIO0_18	Comparator input C	COMP
CMP0_D	I	PIO1_14	Comparator input D	COMP
CMP0_E	I	PIO2_23	Comparator input E	COMP
CMP0_OUT	O	PIO0_1, PIO0_29	Comparator output	-
VREF_COMP	I	PIO1_13	External reference voltage source for 32-stage Voltage Ladder.	-

49.5 Memory map and register description

See COMP, COMP_INT_CTRL and COMPINT_STATUS.

Chapter 50
High Speed Comparator (HSCMP)

50.1 Chip-specific HSCMP information

Table 409. Reference links to related information

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

50.1.1 Module instances

This device has three instances of the HSCMP module, HSCMP0, HSCMP1 and HSCMP2.

50.1.2 Input connections

Table 410. Input connections

Index	CMP0	CMP1	CMP2	Description
0	HSCMP0_IN0 PIO0_24	HSCMP1_IN0 PIO0_7	HSCMP2_IN0 PIO0_17	Standard Muxed, VDD_IO domain
1	HSCMP0_IN1 PIO1_12	HSCMP1_IN1 DAC0_OUT PIO1_22	HSCMP2_IN1 PIO1_23	Standard Muxed, VDD_IO domain
2	unconnected	unconnected	unconnected	Standard Muxed, VDD_IO domain
3	HSCMP0_IN3 PIO1_5	HSCMP1_IN3 PIO1_10	unconnected	Standard Muxed, VDD_IO domain
4	HSCMP0_IN4 OPAMP0_OUT PIO1_9	OPAMP1_OUT	OPAMP2_OUT	GPIO muxed, ch4 in VDDA domain
5	DAC0_OUT PIO1_22	HSCMP1_IN5 DAC1_OUT PIO1_19	DAC2_OUT	GPIO muxed, ch5 in VDDA domain

50.1.3 Input references

In the DAC Control Register, when:

- VRSEL=0, VREFH0 is selected, which is VDD_MAIN supply rail

- VRSEL=1, VREFH1 is selected, which is VREF_OUT

50.1.4 HSCMP trigger inputs

HSCMP Trigger sources get routed through the Input Multiplexer (INPUTMUX). See the INPUTMUX chapter for available trigger sources, in registers HSCMP0_TRIG - HSCMP2_TRIG. The selected trigger drives the WINDOW/SAMPLE signal of the HSCMP.

50.2 Overview

The High Speed Comparator (HSCMP) module provides a circuit for comparing two analog input voltages. It comprises a comparator (CMP), a DAC and an analog mux (ANMUX). See Block diagram for details.

CMP 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. The 256-tap resistor ladder network divides the supply reference V_{in} into 256 voltage levels. A 8-bit digital signal input selects the output voltage level, which varies from V_{in} to $V_{in}/256$. V_{in} can be selected from two voltage sources, vrefh0 and vrefh1. See the chip-specific HSCMP information for the source of vrefh0 and vrefh1.

NOTE

The HSCMP's internal DAC output is available as an on-chip internal signal only and is not available for an external device pin.

ANMUX allows software to select an analog input signal from among eight channel options. One channel option is the DAC output. Other device resources are connected to the other channels; see the chip-specific HSCMP information section for details. ANMUX can operate across the full range of the supply voltage.

50.2.1 Block diagram

The following figure shows the HSCMP block diagram includes the CMP, DAC, and ANMUX modules.

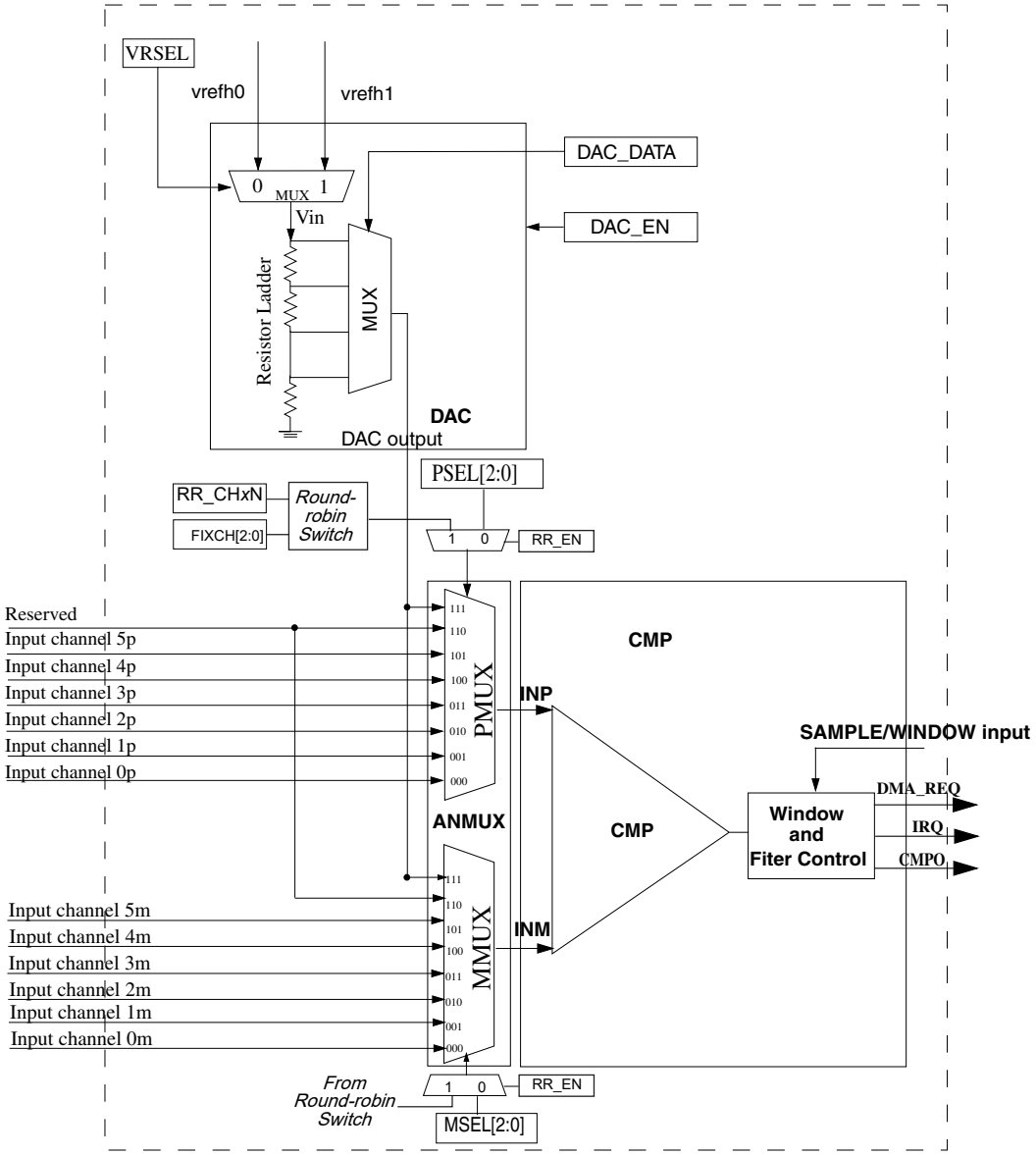


Figure 271. Block Diagram

50.2.2 Features

The features of the HSCMP module include:

- Two 8-to-1 channel MUXes to select input signal from eight channels
- 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
- Advanced feature for window and sample
  - WINDOW/SAMPLE signal can be inverted

— Window can be closed by COUT (comparator output) rising, falling or both edges

— User can define COUT level, when window is closed

- Selectable performance levels: nano power mode, low power (speed) mode, high power (speed) mode
- Programmable hysteresis control
- Selectable inversion on comparator output
- External hysteresis can be used at the same time that the output filter is used for internal functions
- Interrupt and DMA support
- Trigger mode
- Includes a 8-bit resolution DAC
- Selectable supply reference source for DAC
- Configurable low power (speed) mode or high power (speed) mode for DAC

50.3 Functional Description

50.3.1 Functional Block Diagram

The following figure shows the functional block diagram.

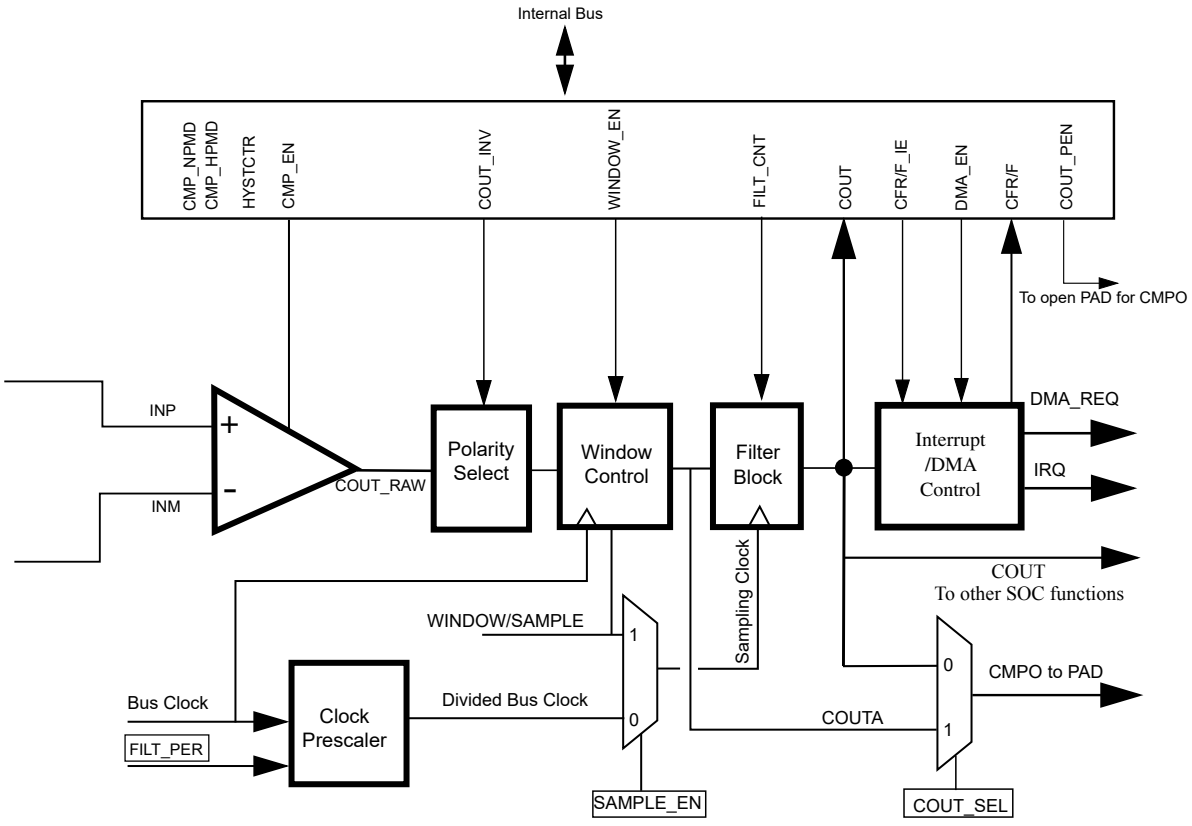


Figure 272. Functional Block Diagram

In the block diagram:

- Two analog input voltages applied to INP and INM are compared. COUT_RAW is high when the INP input voltage is greater than the INM input, and is low when the INP input voltage is less than the INM input.

- This COUT_RAW signal can be inverted if CCR1[COUT_INV] is enabled.
- If CCR1[WINDOW_EN] is enabled, the optionally inverted comparator output COUT_RAW is sampled on every bus clock when window is enabled to generate COUTA. Sampling does NOT occur when window is disabled. This feature can be used to ignore the comparator output during time periods in which the input voltages are not valid. This is especially useful when implementing zero-crossing-detection for certain PWM applications.
- The window control block can be bypassed if CCR1[WINDOW_EN] is disabled.
- The filter block acts as a simple sampler if CCR1[FILT_CNT] is set to 0x01.
- The filter block acts as a filter based on multiple samples when CCR1[FILT_CNT] is set greater than 0x01.
  - If CCR1[SAMPLE_EN] = 1, the external SAMPLE input is used as the sampling clock.
  - If CCR1[SAMPLE_EN] = 0, the divided bus clock is used as the sampling clock.
- The filter block can be bypassed when 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 the CCR1[COUT_SEL] bit and are used for different purposes within the system.
- The optionally filtered COUT can be read directly through register bit CSR[COUT].
- The SAMPLE/WINDOW signal can be inverted by setting CCR1[WINDOW_INV].
- The SAMPLE/WINDOW signal can be closed by COUT's falling edge or rising edge or both edge by setting CCR1[WINDOW_CLS] in window mode.
- In the window mode, when window is closed, COUTA value can be defined as CCR1[COUTA_OW] by setting CCR1[COUTA_OWEN]. If CCR1[COUTA_OWEN] is not set, COUTA holds the last sampled value when window is closed

NOTE

See the chip configuration section for the source of SAMPLE/WINDOW input.

50.3.2 Functional Modes

The comparator window and filter features can be combined as shown in the following table. Individual modes are discussed below the table.

Table 411. 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 & 2B).	T_{PD}
2B	1	0	0	X	0x00		
3A	1	0	1	0x01	X	See the Sampled, Non-Filtered Mode (#3A & 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 411. 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 & 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 & 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, 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.

50.3.2.1 Disabled Mode (#1)

In disabled mode, the analog comparator is non-functional and consumes no power. CSR[COUT] and CMPO is same as CCR1[COUT_INV] in this mode.

50.3.2.2 Continuous Mode (#2A & 2B)

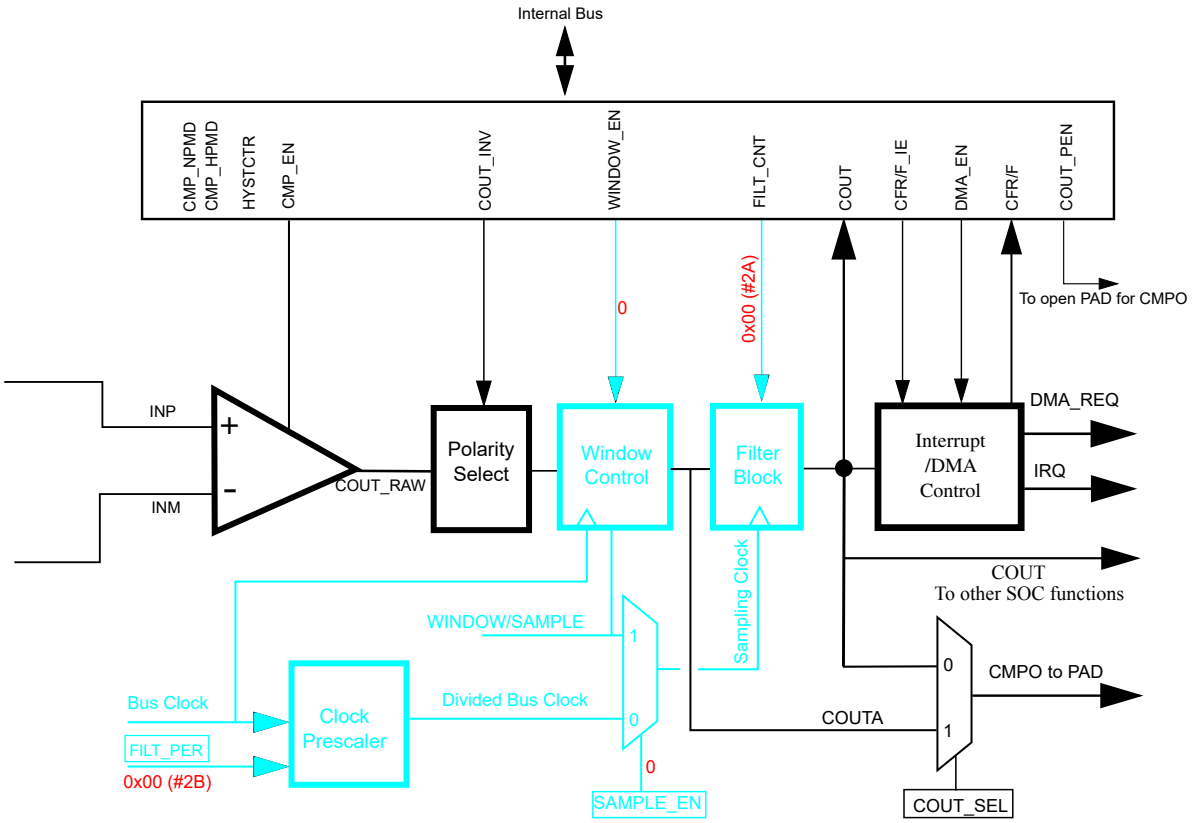


Figure 273. Comparator Operation in Continuous Mode

In this mode, COUT_RAW may be optionally inverted, but is not subject to external sampling or filtering. Both window control and filter blocks are completely bypassed. CSR[COUT] is updated continuously. The path from comparator input pins to output pin is operating in combinational(unclocked) mode. COUT and COUTA are identical.

For cases where a comparator is used to drive a fault input, for example, for a motor-control module such as FTM, it should generally be configured to operate in continuous mode so that an external fault can immediately pass through the comparator to the target fault circuitry.

50.3.2.3 Sampled, Non-Filtered Mode (#3A & 3B)

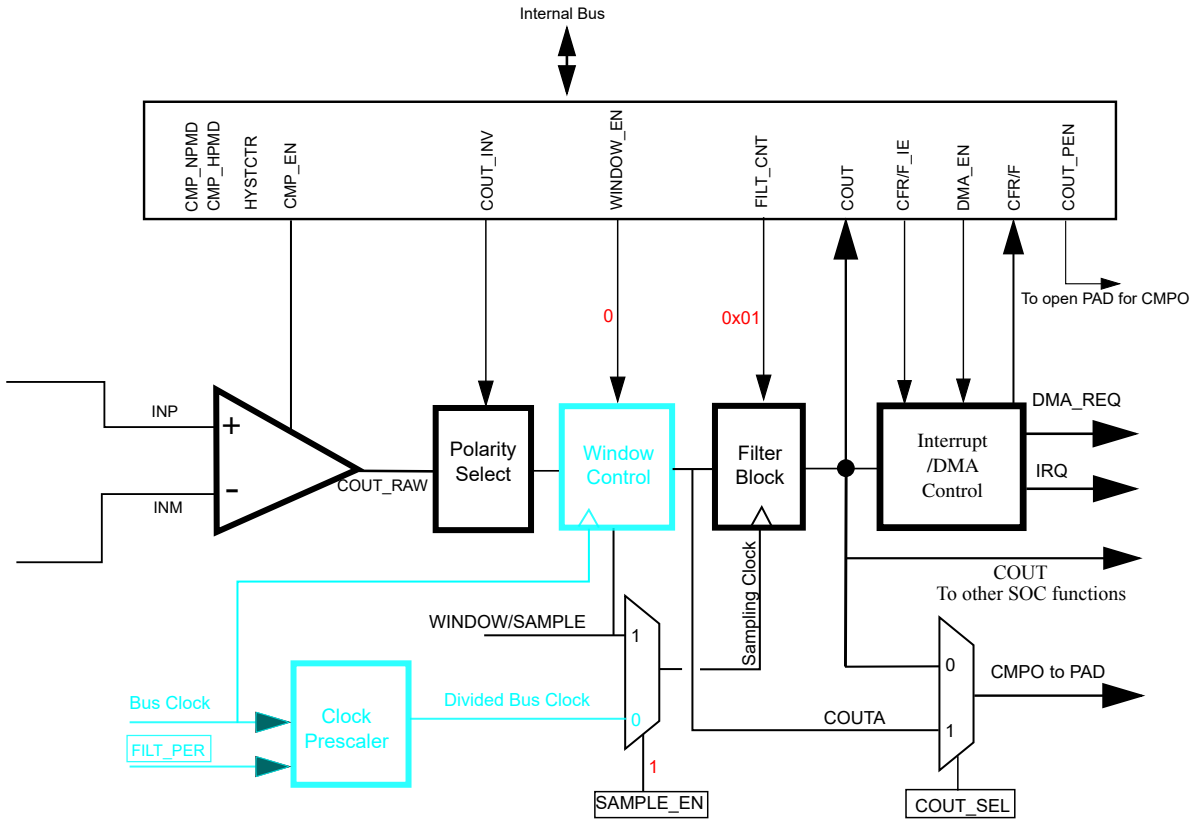


Figure 274. Sampled, Non-Filtered (#3A): Sampling Point Externally Driven

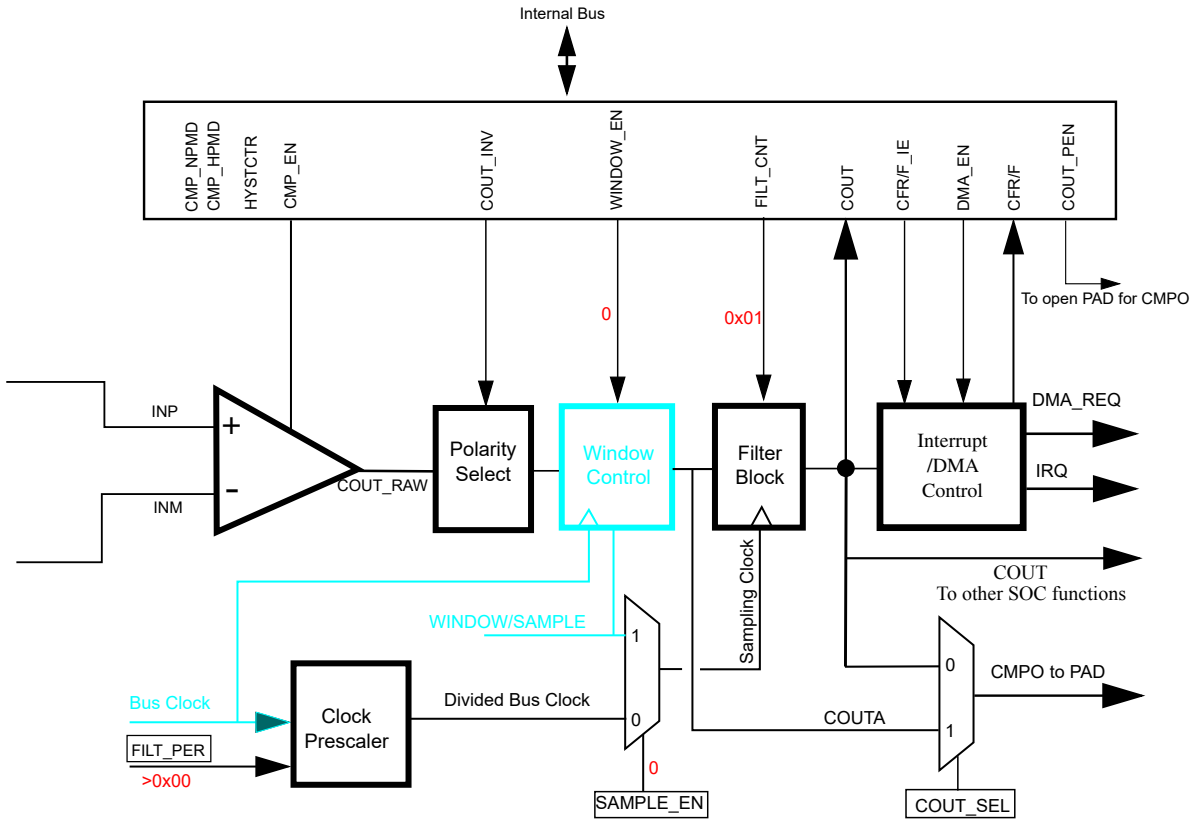


Figure 275. Sampled, Non-Filtered (#3B): Sampling Interval Internally Derived

In this mode, the path from analog inputs to COUTA is combinational(unclocked). Windowing control is completely bypassed. COUTA is sampled whenever a rising edge is detected on the sampling clock.

The only difference in operation between sampled, non-filtered (#3A) and sampled, non-filtered (#3B) is in how the clock to the filter block is derived. In #3A, the clock to filter block is externally derived while in #3B, the clock to filter block is internally derived.

The filter block has no other function than sample/hold of the comparator output in this mode.

The following figure illustrates comparator operation in this mode, assuming that the polarity select is set to non-inverting state.

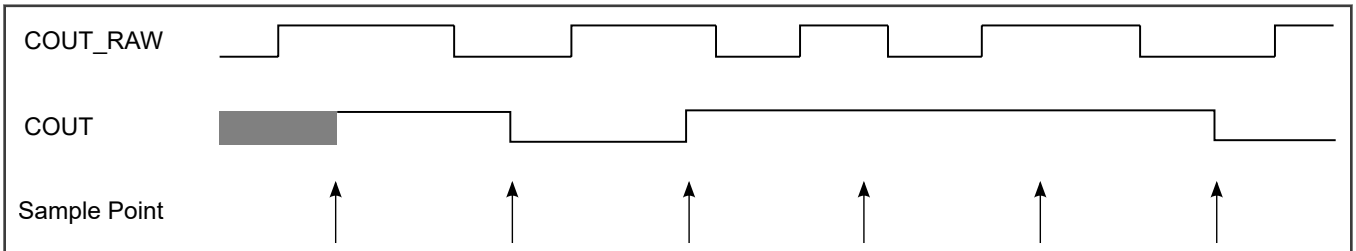


Figure 276. Sampled, Non-Filtered Mode Timing Diagram

50.3.2.4 Sampled, Filtered Mode (#4A & 4B)

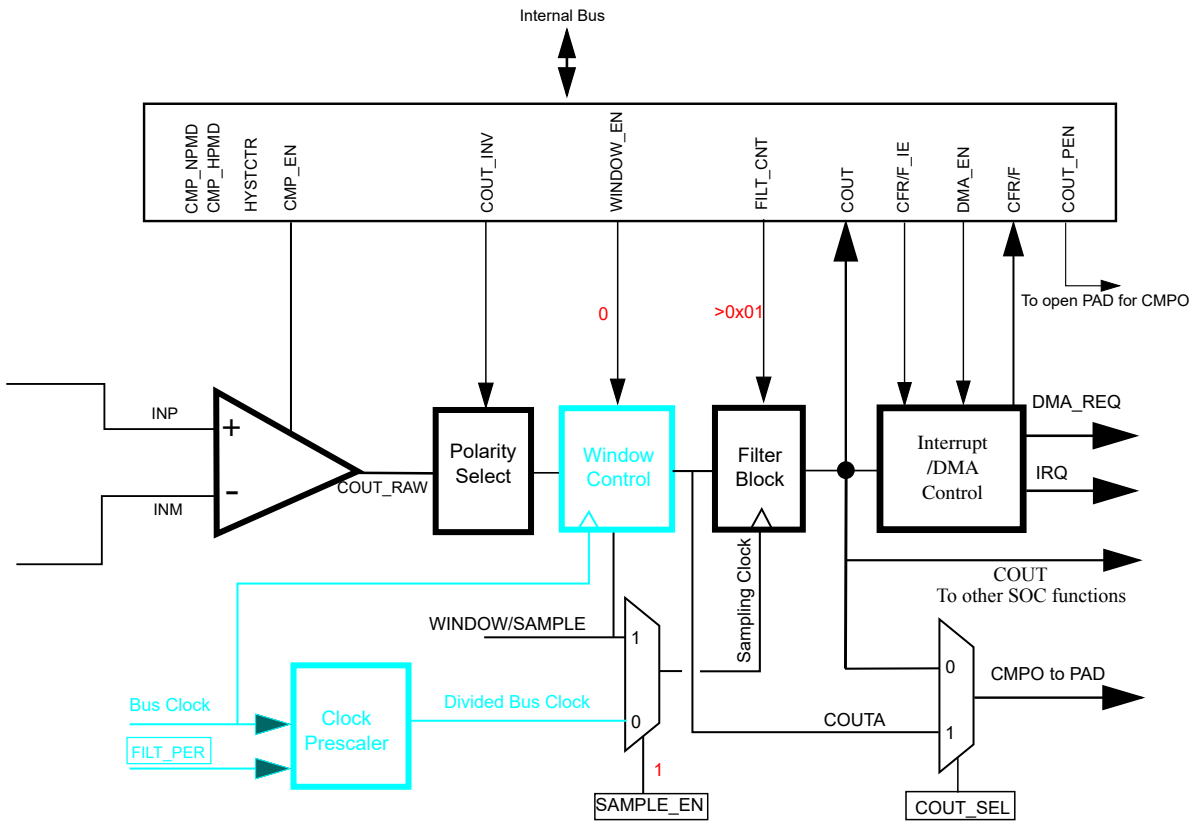


Figure 277. Sampled, Filtered (#4A): Sampling Point Externally Driven

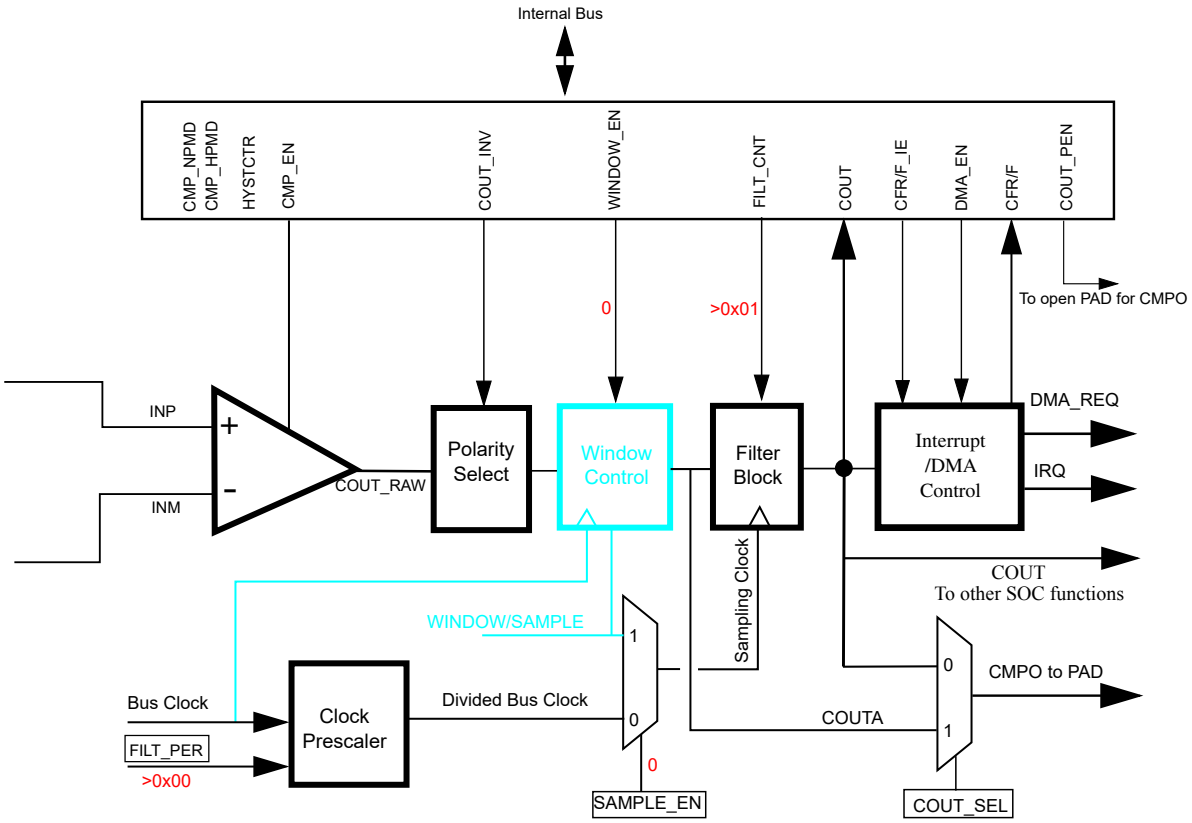


Figure 278. Sampled, Filtered (#4B): Sampling Point Internally Derived

In this mode, the path from analog inputs to COUTA is combinational(unclocked). Windowing control is completely bypassed. COUTA is sampled whenever a rising edge is detected on the sampling clock.

The only difference in operation between sampled, non-filtered (#3A) and sampled, filtered (#4A) is that, now, CCR1[FILT_CNT]>1, which activates filter operation.

The only difference in operation between sampled, non-filtered (#3B) and sampled, filtered (#4B) is that now, CCR1[FILT_CNT]>1, which activates filter operation.

50.3.2.5 Windowed Mode (#5A & 5B)

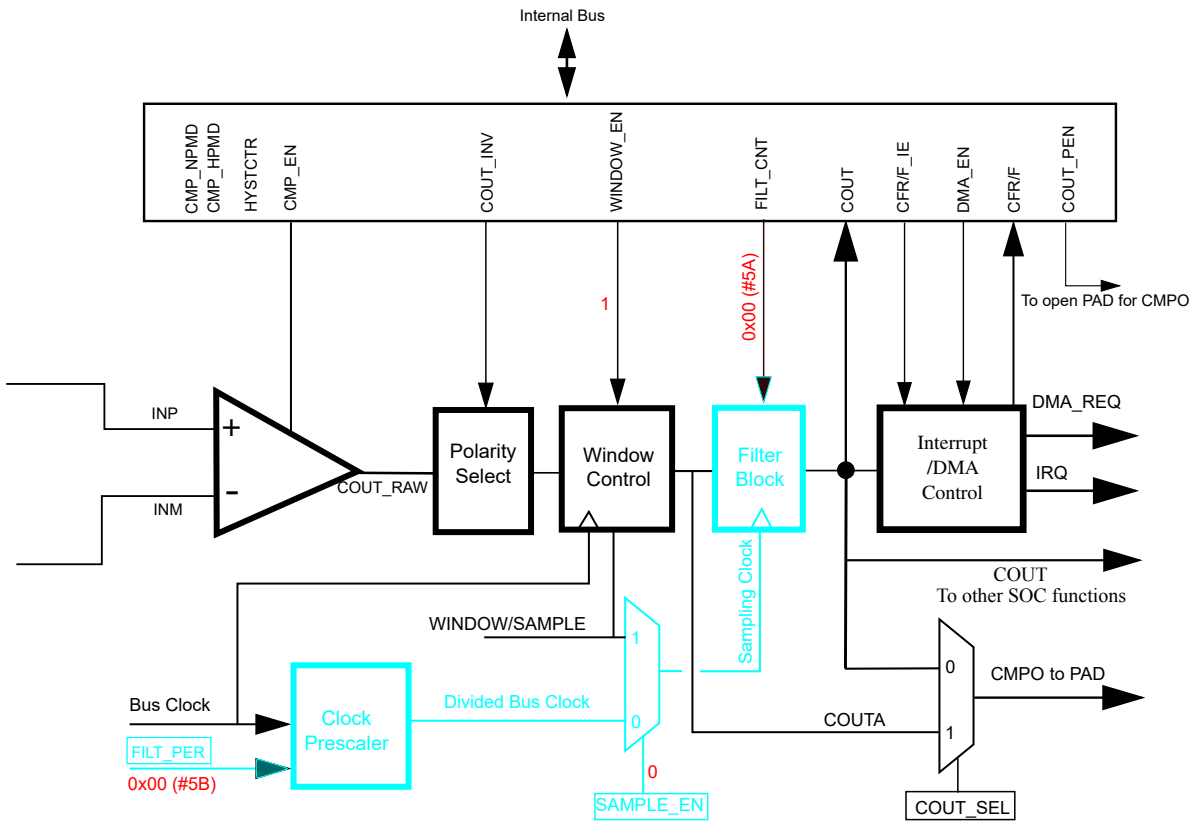


Figure 279. Windowed Mode

In this mode, COUTA is clocked by the bus clock whenever window is enabled. The last latched value is held when window is disabled. The filter block is bypassed.

The following figure illustrates comparator operation in this mode, ignoring latency of the analog comparator, polarity select, and window control block. It also assumes that the polarity select is set to non-inverting state.

In actual operation, COUTA may lag the analog inputs by up to two bus clock cycles plus the combinational path delay through the comparator and polarity select logic.

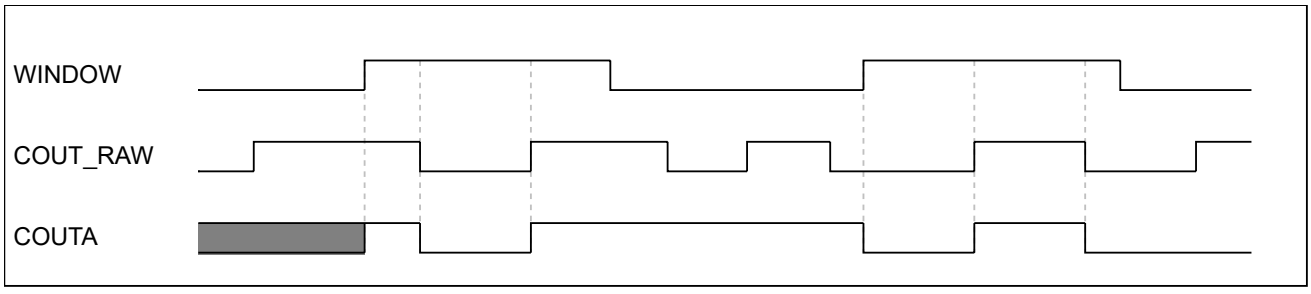


Figure 280. Windowed Mode Timing Diagram

If CCR1[COUTA_OWEN] is set, user can define COUTA level as CCR1[COUTA_OW], when window is closed. Following figures show this case.

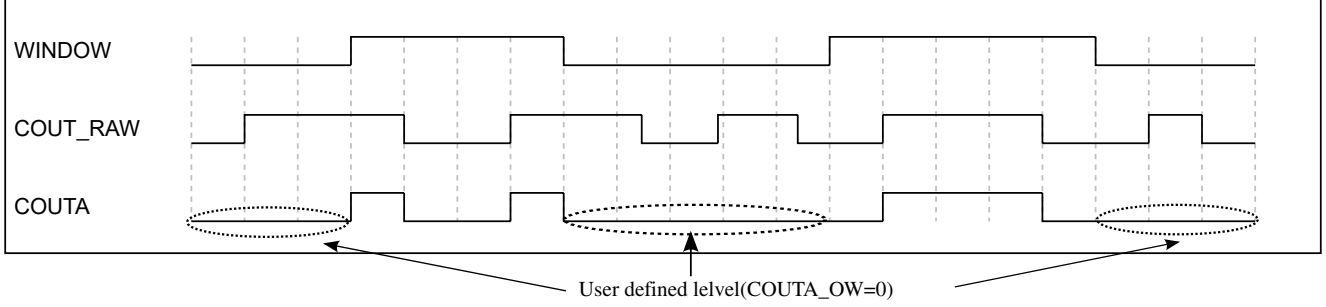


Figure 281. Windowed Mode Timing Diagram With User Defined Value 0 outside WINDOW

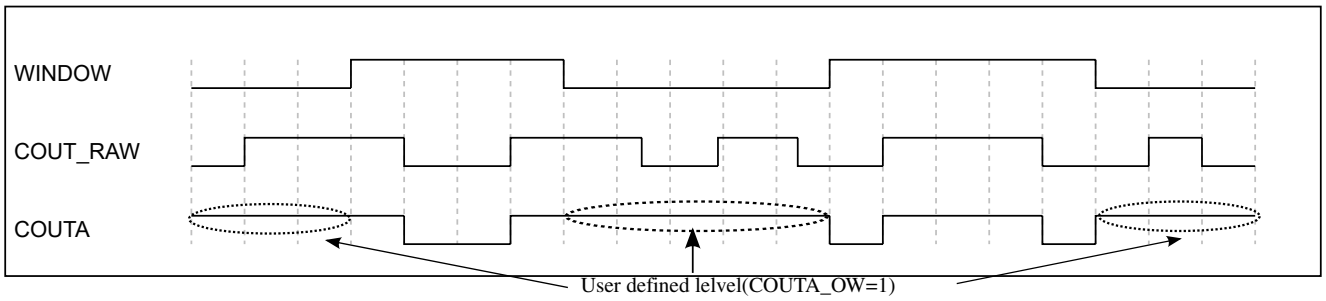


Figure 282. Windowed Mode Timing Diagram With User Defined Value 1 outside WINDOW

If CCR1[WINDOW_CLS] is set, user can define the COUT event (rising, falling or both edges, selected by CCR1[EVT_SEL]) to close the window. The external WINDOW signal has to go to zero and back to one to enable the internal window again. Following figure shows an example that COUT rising edge close the internal window.

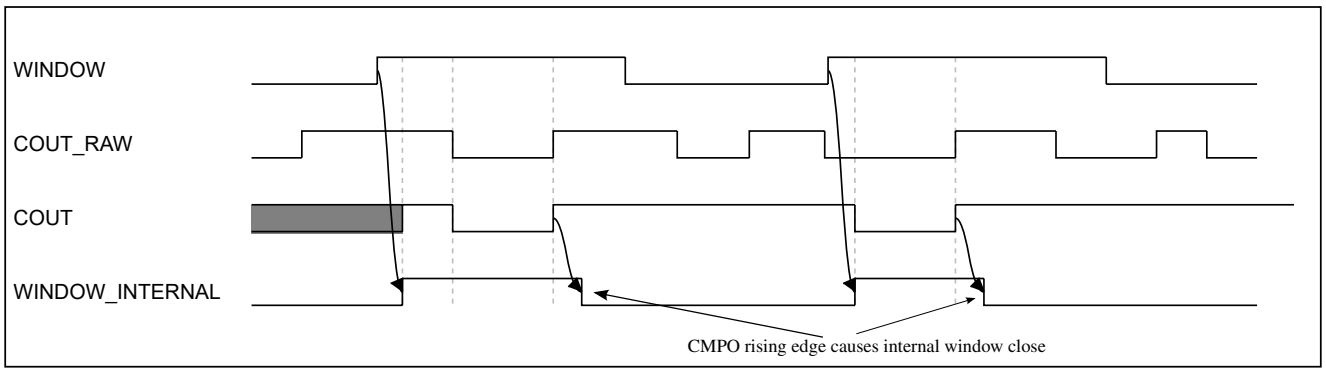


Figure 283. Windowed Mode Timing Diagram With COUT Rising Edge Close Window

If CCR1[WINDOW_INV] is set, user can invert the window signal before it is used. The following figure shows this case.

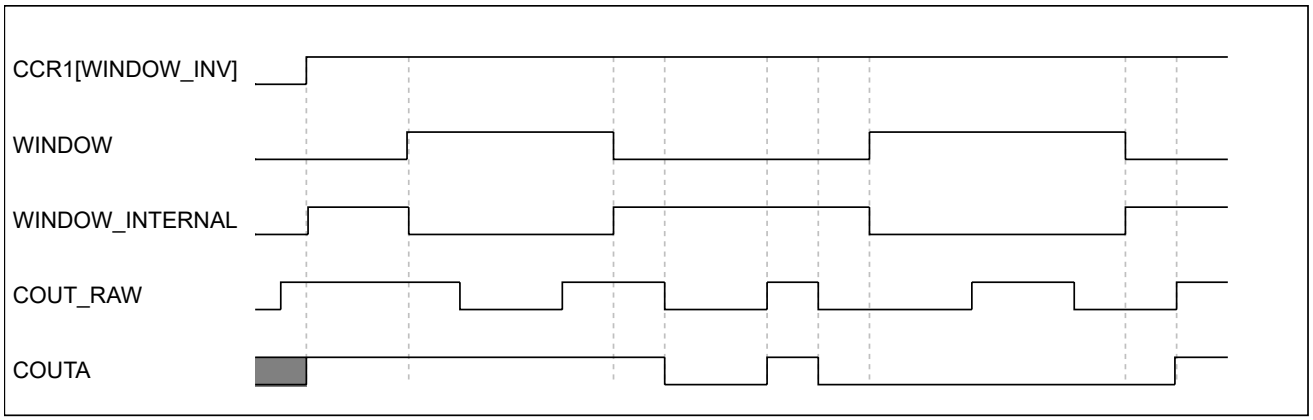


Figure 284. Windowed Mode Timing Diagram With WINDOW Signal Inverted.

50.3.2.6 Windowed/Resampled Mode (#6)

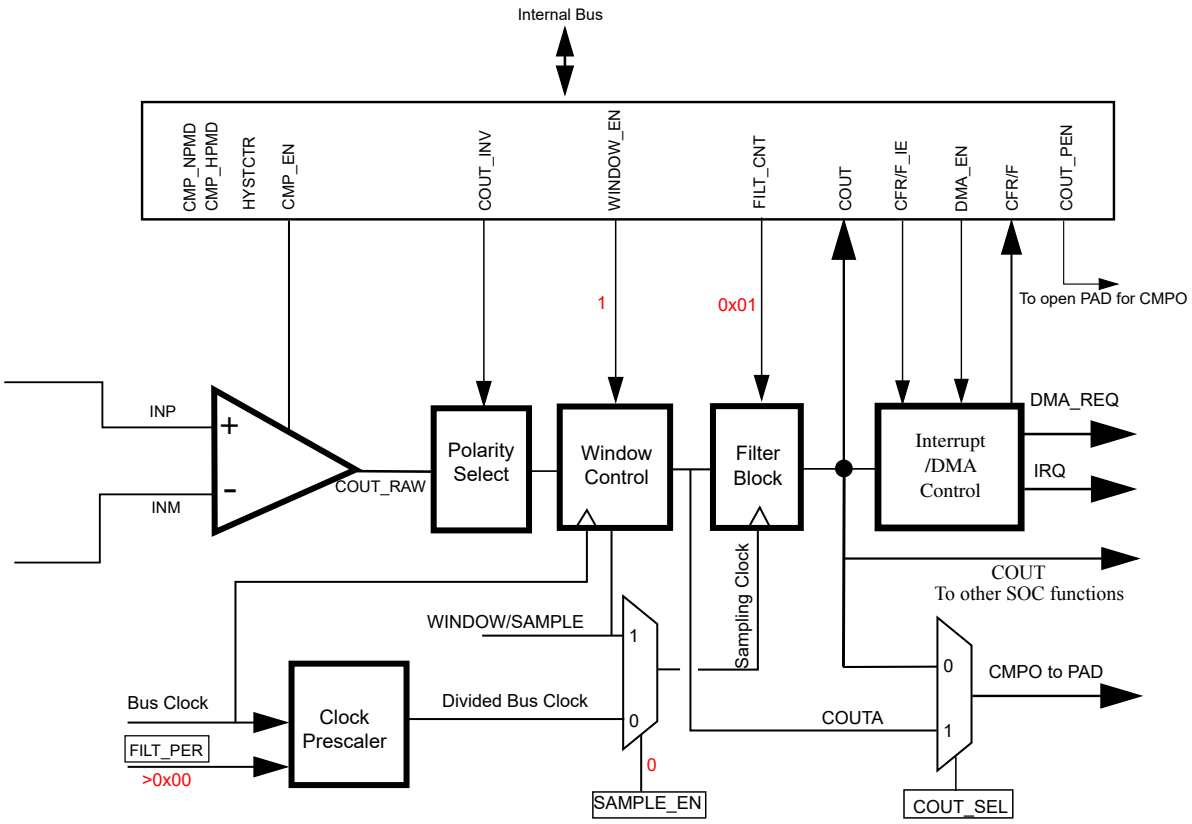


Figure 285. Windowed/Resampled Mode

This mode of operation results in an unfiltered string of comparator samples where the interval between the samples is determined by CCR1[FILT_PER] and the bus clock rate. Configuration for this mode is virtually identical to that for the windowed/filtered mode shown in the next section. The only difference is that the value of CCR1[FILT_CNT] in this mode must be 1.

The following figure uses the same input stimulus shown in Figure 280, and adds resampling of COUTA to generate COUT. Samples are taken at the time points indicated by the arrows in the figure. Again, prop delays and latency are ignored for the sake of clarity.

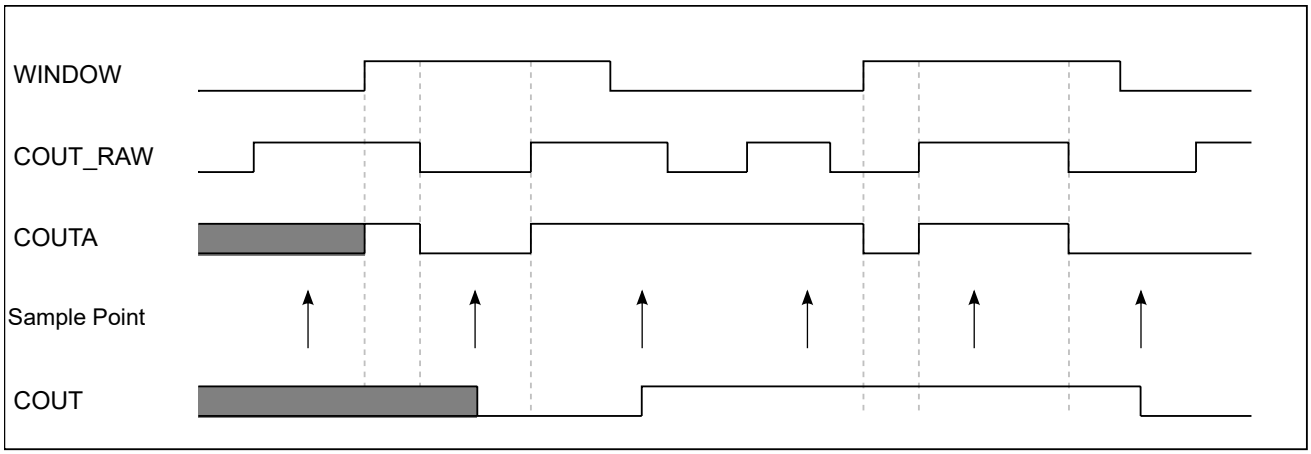


Figure 286. Windowed/Resampled Mode Operation

This example was generated solely to demonstrate operation of the comparator in windowed/resampled mode, and does not reflect any specific application. Depending upon the sampling rate and window placement, COUT may not see zero-crossing events detected by the analog comparator. Sampling period and/or window placement must be carefully considered for a given application.

50.3.2.7 Windowed/Filtered Mode (#7)

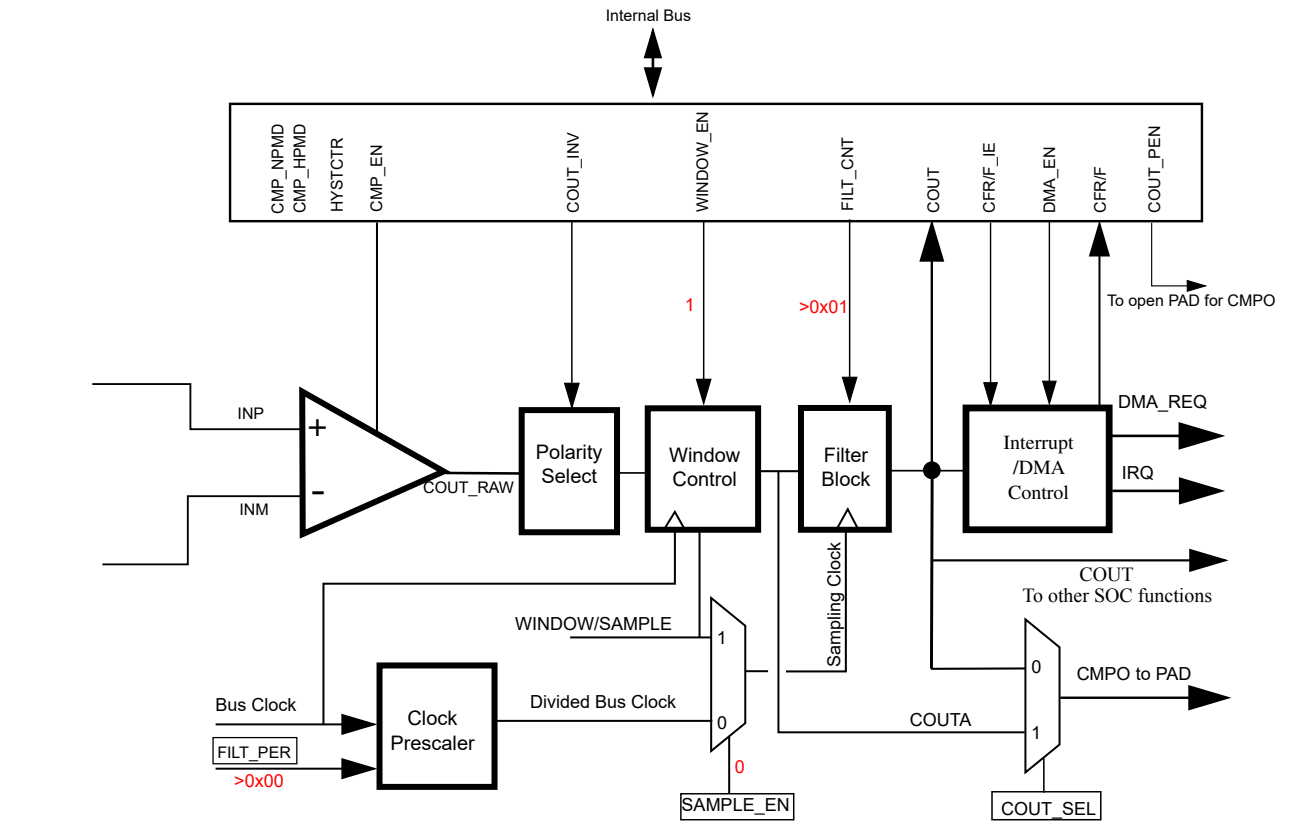


Figure 287. Windowed/Filtered Mode

The only difference in operation between Windowed/Resampled Mode (#6) and Windowed/Filtered Mode (#7) is that, now, CCR1[FILT_CNT]>1, which activates filter operation.

This 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 can be approximated: 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.

50.3.3 Round Robin Trigger mode

The Round Robin Trigger mode is enabled with RRCR0[RR_EN]. A trigger event initiates a comparison sequence. The next trigger event should not come before the current sequence completes.

The reference channel for either the plus side MUX or the minus side MUX is selected by RRCR1[RR_FIXP] and RRCR1[FIXCH]. The active channels are selected by RRCR1[RR_CH*n*EN].

When a trigger comes, the analog comparator is enabled. After the comparison sequence completes, the analog comparator is disabled again. The analog stabilization time is controlled by RRCR0[RR_INITMOD]. Make sure that the (RR_INITMOD*round robin clock period) should be longer than the initialization delay in the chip data sheet.

When the stabilization process completes, the round robin manner comparison sequence begins. After the configurable number of operation clocks defined by RRCR0[RR_NSAM], the comparison result is sampled for the selected active channel.

After all the active channels are sampled/compared, if the comparison result changes from its pre-programmed state, the corresponding flag in RRSR[RR_CH*n*F] is set. The pre-programmed state for each channel is configured by writing to RRCSR[RR_CH*n*OUT] field. RRCSR[RR_CH*n*OUT] is updated to store the last comparison result for each channel. If any flag in RRSR[RR_CH*n*F] is set, flag CSR[RRF] will be set. If IER[RRF_IE] is set, an asynchronous interrupt is asserted. 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 set, so channels #1, #3, and #4 are selected for round robin depended on their priority setting. RRCR0[RR_NSAM] is set to 2'b01, so one clock later the comparison result of the selected channel is sampled. When channel #4 is compared, 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]), an interrupt is generated. Software can then poll the RRSR[RR_CH*n*F] to see which channel input(s) changed value.

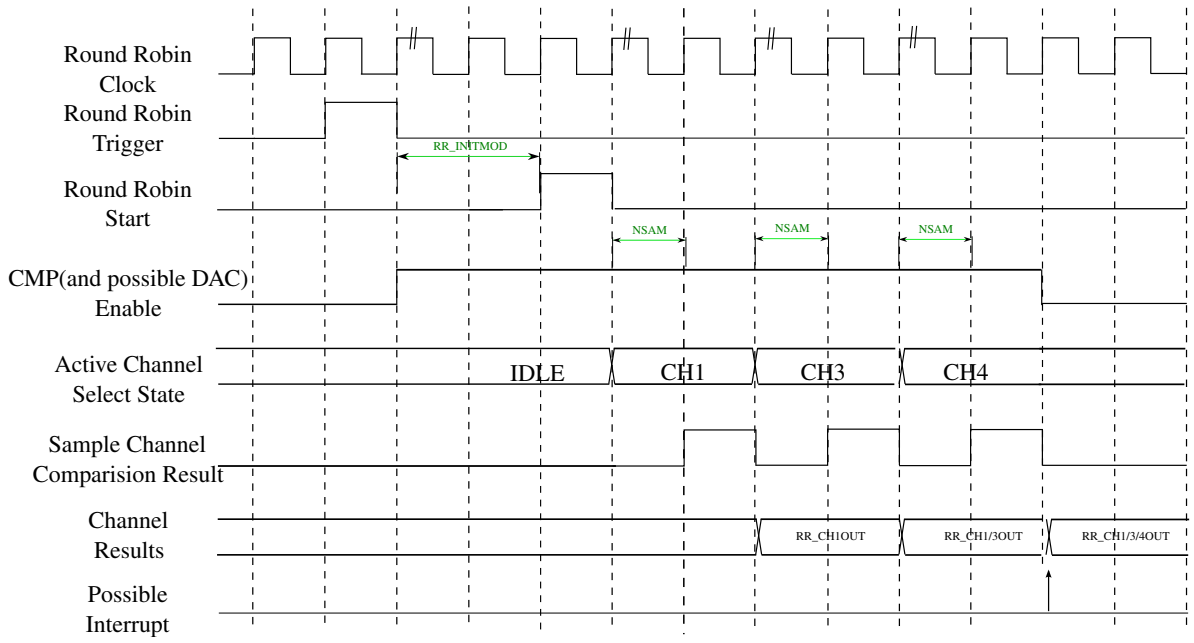


Figure 288. Trigger Mode

The table below shows the channel decode in both functional mode and trigger mode. Other cases not in the table are illegal.

Table 412. CMP Channel Decode in Functional Mode and 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]	0~7	0	1	x	x	x	DAC	Channel decoded from MSEL[2:0]	Channel 0~7 can be compared with DAC
		0~7	x	1	0	x	x	x	Channel decoded from PSEL[2:0]	DAC	Channel 0~7 can be compared with DAC
		0~7	0~7	1	1	x	x	x	Channel decoded from PSEL[2:0]	Channel decoded from MSEL[2:0]	Channel 0~7 can be compared with channel 0~7[2]
Trigger Mode	1	x	x	0	1	0	x	0~7	DAC	Channel sweep (RR_CHxN)	Channel 0~7 can be swept with DAC
		x	x	1	0	1	x	0~7	Channel sweep (RR_CHxN)	DAC	Channel 0~7 can be swept with DAC
		x	x	1	1	0	0~7	0~7	Channel fixed by FIXCH[2:0]	Channel sweep (RR_CHxN)	Channel 0~7 can be swept with a fixed channel(0~7)[3]
		x	x	1	1	1	0~7	0~7	Channel sweep (RR_CHxN)	Channel fixed by FIXCH[2:0]	Channel 0~7 can be swept with a fixed channel(0~7)[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.

Trigger Mode Programming Recommendation

Configure the Trigger Mode as follows. Register fields in the same register can be configured at the same time.

1. Configure the comparison cycles by RRCR0[RR_NSAM]. Note: It is a mandatory request that the round robin cycling period must be 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 the RRCR0[RR_INITMOD] registers, the RR_INITMOD x round robin clock period must be longer than the initialization delay, which can be referred from the chip data sheet.
3. If using internal trigger, enable the RRCR2[RR_TIMER_EN] and configure RRCR2[RR_TIMER_RELOAD] according to the Round Robin Clock frequency.
4. Configure the RRCR1[FIXP] to select the fixed port of CMP and
    a. If using one input channel to compare with other channels, configure the RRCR1[FIXCH] to select the fixed channel
    b. If using DAC output to compare with input channels, configure the CCR2[INPSEL] or CCR2[INMSEL](according to RRCR1[FIXP])to select the DAC output
5. Configure channels for comparison by RRCR1[RR_CHnEN]
6. Write RRCSR[RR_CHnOUT] to define the pre-set state of channel n.
7. Clear channel flags RRSR[RR_CHnF]
8. Enable round robin interrupt by IER[RRF_IE] (disable IER[CFR_IE] and IER[CFF_IE])
9. Enable round robin mode by RRCR0[RR_EN]

50.3.4 Low-Pass Filter

The low-pass filter operates on the unfiltered, optionally inverted comparator output COUTA and generates the filtered and synchronized output COUT. Both COUTA and COUT can be configured as module outputs and are used for different purposes within the system.

Synchronization and edge detection are always used to determine status register bit values. They also apply to COUT for all sampling and windowed modes. Filtering can be performed using an internal timebase defined by CCR1[FILT_PER], or using an external SAMPLE input to determine sample time.

The need for digital filtering and the amount of filtering is dependent on user requirements. Filtering can become more useful in the absence of an external hysteresis circuit. Without external hysteresis, high-frequency oscillations can be generated at COUTA when the selected INM and INP input voltages differ by less than the offset voltage of the differential comparator.

50.3.4.1 Enabling Filter Modes

Filter modes can be enabled by:

- Setting CCR1[FILT_CNT] > 0x01 and
- Setting CCR1[FILT_PER] to a nonzero value or setting CCR1[SAMPLE_EN]=1

If using the divided bus clock to drive the filter, it samples COUTA every CCR1[FILT_PER] bus clock cycles.

If CCR1[SAMPLE_EN]=1, the 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.

50.3.4.2 Latency issues

Program the value of CCR1[FILT_PER] or SAMPLE period such that the sampling period is just longer than the period of the expected noise to ensure that a given noise spike corrupts only one sample. The value of CCR1[FILT_CNT] must be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the power of CCR1[FILT_CNT].

The values of CCR1[FILT_PER] or SAMPLE period and CCR1[FILT_CNT] must also be traded off against the desire 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 411 summarizes maximum latency values for the various modes of operation in the absence of noise. Filtering latency is restarted each time an actual output transition is masked by noise.

50.3.5 Low Power Mode Operation

The following table shows the WAIT mode and the STOP mode operation of the HSCMP module.

Table 413. Low Power Mode Operation

Mode of Operation	Description
WAIT	HSCMP can operate normally with window and filter function.
STOP	HSCMP can operate only in continuous mode or trigger mode.

50.3.6 DMA

When DMA support is enabled by setting CCR1[DMA_EN] and the interrupt is enabled by setting IER[CFR_IE], IER[CFF_IE], or both, the corresponding change on COUT forces a DMA transfer request rather than a CPU interrupt instead. When the DMA has completed the transfer, it sends a transfer completing indicator signal that de-asserts 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 force another DMA request.

The comparator can remain functional in STOP modes if CCR0[CMP_STOP_EN] is set. A DMA transfer request will wake up the system from STOP modes. After the data transfer has finished, the system will go back again to STOP modes. Refer to the DMA chapters in the device reference manual for the asynchronous DMA function for details.

50.3.7 Clocking

HSCMP requires the following clocks to operate:

- Bus clock is used for register access and window/filter function.
- Round robin clock for trigger mode.

50.3.8 Resets

The global chip reset signal resets HSCMP.

50.3.9 Interrupts

When corresponding interrupt enable register bit is set, flags CSR[CFR], CSR[CFF], CSR[RRF] can generate interrupt, assuming the DMA enable bit is not set. The interrupt is de-asserted by clearing either the flag or the interrupt enable register bit.

50.4 External Signals

Table 414. External Signal Descriptions

Signal	Description	I/O
CMPO	Filtered or unfiltered comparator output	O
Input_Analog_Channels	Analog input channels(Refer to the chip-specific information for the connections)	I
VREFH_EXT	External reference voltage for the CMP-DAC(Refer to the chip-specific information for the connections)	I

50.5 Initialization

To enable HSCMP, configure the control registers (CCR1, CCR2, DCR, and so on) before setting CCR0[CMP_EN]. To disable the module, clear CCR0[CMP_EN]. Switching operation modes or changing control register bits on the fly (when CCR0[CMP_EN] is 1) may cause noise on the COUT or COUTA signals. To avoid unwanted signal noise, be sure to disable the module before switching modes or changing control bits.

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 data sheet for power-on delays of the comparators. The delay caused by windowing and filter function is specified in the Table 411.

During operation, the propagation delay of the selected data paths must always be considered. 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.

50.6 Application Information

50.6.1 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 the initialization time.)
   - Register field RR_INITMOD provides a maximum 63 RCLK cycles for the analog CMP and DAC initialization.
   - Rclk must be slow enough to satisfy: 63 * (1/f_{RCLK}) > $T_{inialization}$, where f_{RCLK} is in MHz, and $T_{inialization}$is in microsecond.
   - so f_{RCLK} < 63 / $T_{inialization}$
   - Example: $T_{inialization}$ = 40 microsecond, then f_{RCLK} should be smaller than 1.575MHz.
2. The analog CMP propagation delay(See the chip datasheet for the CMP propagation delay.)
   - Register field NSAM field provides a maximum 4 RCLK cycles for the analog CMP propagation delay.
   - Rclk must be slow enough to satisfy: 4 * (1/f_{RCLK}) > $T_{propagation}$, where f_{RCLK} is in MHz, and $T_{propagation}$ is in microsecond.
   - so f_{RCLK} < 4 / $T_{propagation}$
   - Example: $T_{propagation}$ = 0.1 microsecond, then f_{RCLK} should 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.

50.7 HSCMP register descriptions

The memory map comprises of 32-bit aligned registers which can be accessed 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 are terminated with an error. Read access to reserved address will also generate a transfer error and the read data bus will show all 0s.

50.7.1 HSCMP memory map

HSCMP0 base address: 400B_3000h

HSCMP1 base address: 400B_7000h

HSCMP2 base address: 400B_A000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Version ID Register (VERID)	32	RO	0100_0000
4	Parameter Register (PARAM)	32	See section	0000_0002
8	Comparator Control Register 0 (CCR0)	32	See section	0000_0002
C	Comparator Control Register 1 (CCR1)	32	See section	0000_0000
10	Comparator Control Register 2 (CCR2)	32	See section	0000_0000
18	DAC Control Register (DCR)	32	See section	0000_0000
1C	Interrupt Enable Register (IER)	32	See section	0000_0000
20	Comparator Status Register (CSR)	32	See section	0000_0000
24	Round Robin Control Register 0 (RRCR0)	32	See section	0000_0000
28	Round Robin Control Register 1 (RRCR1)	32	See section	0000_0000
2C	Round Robin Control and Status Register (RRCSR)	32	See section	0000_0000
30	Round Robin Status Register (RRSR)	32	See section	0000_0000
38	Round Robin Control Register 2 (RRCR2)	32	See section	0000_0000

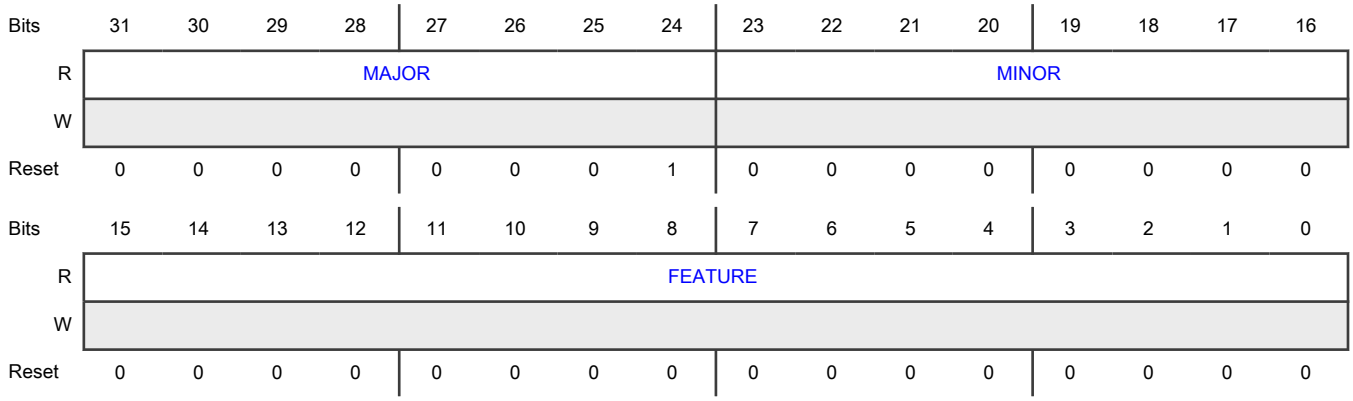
50.7.1.1 Version ID Register (VERID)

Contains version numbers for the module design and feature set.

Offset

Register	Offset
VERID	0h

Diagram



Fields

Field	Description
31-24 MAJOR	Major Version Number Returns the major version number for the module design.
23-16 MINOR	Minor Version Number Returns the minor version number for the module design.
15-0 FEATURE	Feature Specification Number Returns the feature set number. 0000_0000_0000_0001 - Round robin feature

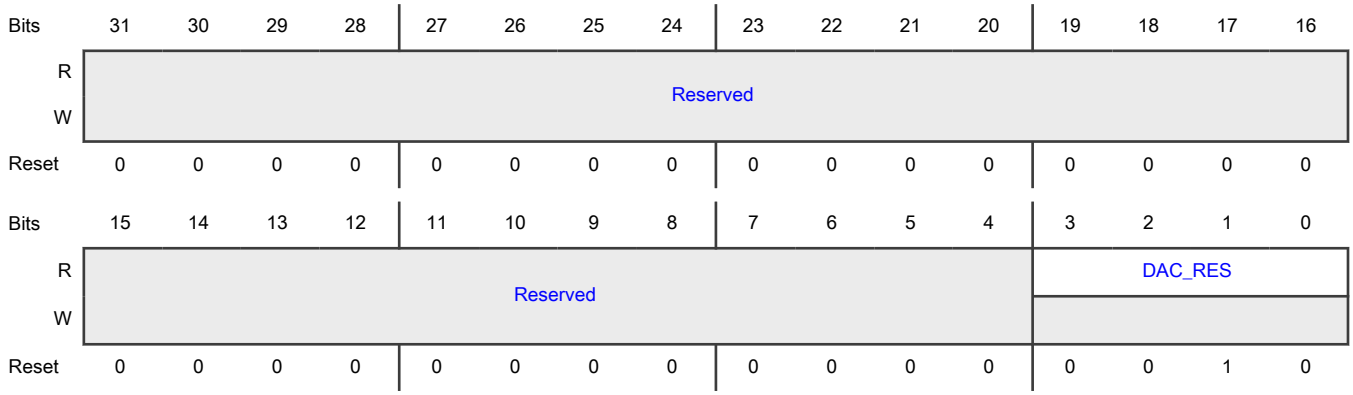
50.7.1.2 Parameter Register (PARAM)

Contains parameter values that are implemented in the module.

Offset

Register	Offset
PARAM	4h

Diagram



Fields

Field	Description
31-4 —	Reserved
3-0 DAC_RES	DAC Resolution Indicates the DAC resolution supported by this implementation. 0000 - 4 bit DAC 0001 - 6 bit DAC 0010 - 8 bit DAC 0011 - 10 bit DAC 0100 - 12 bit DAC 0101 - 14 bit DAC 0110 - 16 bit DAC

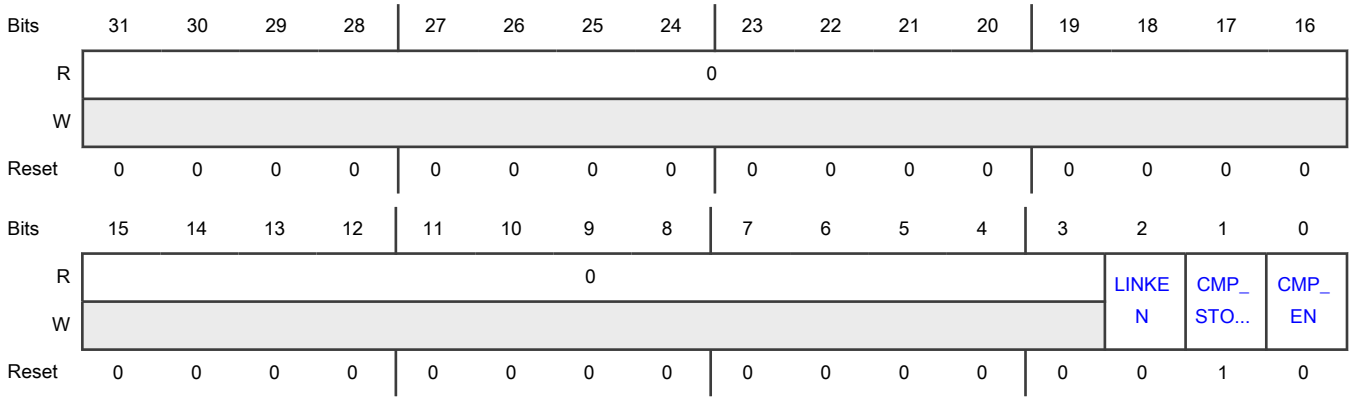
50.7.1.3 Comparator Control Register 0 (CCR0)

Contains configuration options for enabling the analog comparator and the DAC.

Offset

Register	Offset
CCR0	8h

Diagram



Fields

Field	Description
31-3 —	Reserved
2 LINKEN	CMP-to-DAC Link Enable Controls the link from the CMP enable to the DAC enable. 0 - Disable the CMP-to-DAC link: enabling or disabling the DAC is independent from enabling or disabling the CMP. 1 - Enable the CMP-to-DAC link: the DAC enable/disable is controlled by the CMP_EN bit instead of DCR[DAC_EN].
1 CMP_STOP_EN	Comparator STOP Mode Enable Allows software to enable the analog comparator or the DAC when the device is in STOP mode. **NOTE** CMP_STOP_EN has no effect in trigger mode. 0 - Disable the analog comparator regardless of CMP_EN. 1 - Allow the analog comparator to be enabled by CMP_EN.
0 CMP_EN	Comparator Enable Enables the analog comparator. **NOTE** When CCR0[LINKEN]=1, CMP_EN also controls the enabling/disabling of the DAC instead of DCR[DAC_EN]. 0 - Disable (The analog logic remains off and consumes no power.) 1 - Enable

50.7.1.4 Comparator Control Register 1 (CCR1)

Contains configuration options for comparator operation, such as enabling Sampling or Windowing mode.

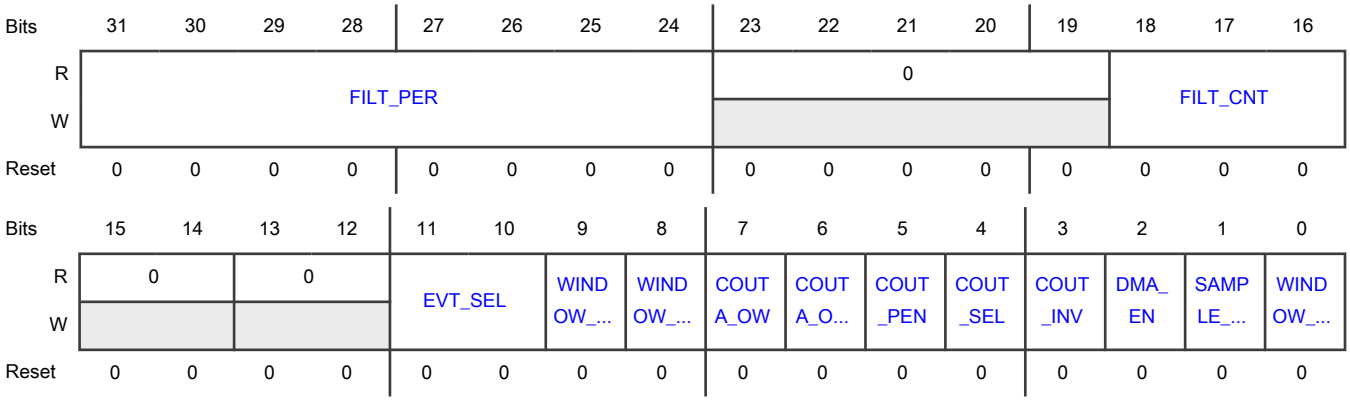
NOTE

Sampling and Windowing modes cannot both be enabled at the same time. Sampling mode takes precedence over Windowing mode. If software attempts to set both SAMPLE_EN and WINDOW_EN, only SAMPLE_EN is set.

Offset

Register	Offset
CCR1	Ch

Diagram



Fields

Field	Description
31-24 FILT_PER	Filter Sample Period Specifies the sampling period (in bus clock cycles) of the comparator output filter. Programming FILT_PER to 0x00 bypasses the filter. Filter programming and latency details are provided in the functional description section. **NOTE** FILT_PER has no effect in Sampling mode (CCR1[SAMPLE_EN]=1).
23-19 —	Reserved
18-16 FILT_CNT	Filter Sample Count Specifies the number of consecutive samples that must agree before the comparator ouput filter accepts the sample as a new valid output state. For information regarding filter programming and latency, see the functional description section. 000 - Filter is bypassed: COUT = COUTA 001 - 1 consecutive sample (Comparator output is simply sampled.) 010 - 2 consecutive samples 011 - 3 consecutive samples

Table continues on the next page...

Table continued from the previous page...

Field	Description
	100 - 4 consecutive samples 101 - 5 consecutive samples 110 - 6 consecutive samples 111 - 7 consecutive samples
15-14 —	Reserved
13-12 —	Reserved
11-10 EVT_SEL	COUT Event Select Selects which COUT signal edge (rising, falling, or both) defines a COUT event. **NOTE** Valid only in Windowing mode. 00 - Rising edge 01 - Falling edge 1x - Both edges
9 WINDOW_CLS	COUT Event Window Close Enables a COUT event (defined as a COUT rising edge, falling edge, or both) to close an active window. See the EVT_SEL field to configure the COUT event. **NOTE** The WINDOW signal has to go to zero and back to one again to re-activate the window. Valid only in Windowing mode. 0 - COUT event cannot close the window 1 - COUT event can close the window
8 WINDOW_INV	WINDOW/SAMPLE Signal Invert Inverts the WINDOW/SAMPLE signal. 0 - Do not invert 1 - Invert
7 COUTA_OW	COUTA Output Level for Closed Window Defines the COUTA signal value when the window is closed. **NOTE** Valid only in Windowing mode and when COUTA_OWEN=1. 0 - COUTA is 0

Table continues on the next page...

Table continued from the previous page...

Field	Description
	1 - COUTA is 1
6 COUTA_OWEN	COUTA_OW Enable Enables the COUTA signal value to be defined by the COUTA_OW bit when the window is closed. **NOTE** Valid only in Windowing mode. 0 - COUTA holds the last sampled value 1 - COUTA is defined by the COUTA_OW bit
5 COUT_PEN	Comparator Output Pin Enable Enables the comparator output to become an available signal option for a selected package pin. 0 - Not available 1 - Available
4 COUT_SEL	Comparator Output Select Selects which comparator output option, COUT or COUTA, to use for CMPO. 0 - Use COUT (filtered) 1 - Use COUTA (unfiltered)
3 COUT_INV	Comparator Invert Selects the polarity of the analog comparator function, affecting the value driven to the COUT output (on both the device pin and as CSR[COUT]) when CCR0[CMP_EN] is 0. **NOTE** COUT_INV has no effect in trigger mode. 0 - Do not invert 1 - Invert
2 DMA_EN	DMA Enable Enables DMA transfers triggered from the HSCMP module. When DMA_EN and the corresponding interrupt enable bit are set, a DMA request is asserted when CFR or CFF is set. 0 - Disable 1 - Enable
1 SAMPLE_EN	Sampling Enable Enables Sampling mode. 0 - Disable 1 - Enable

Table continues on the next page...

Table continued from the previous page...

Field	Description
0 WINDOW_EN	Windowing Enable Enables Windowing mode. **NOTE** Valid only when SAMPLE_EN=0. 0 - Disable 1 - Enable

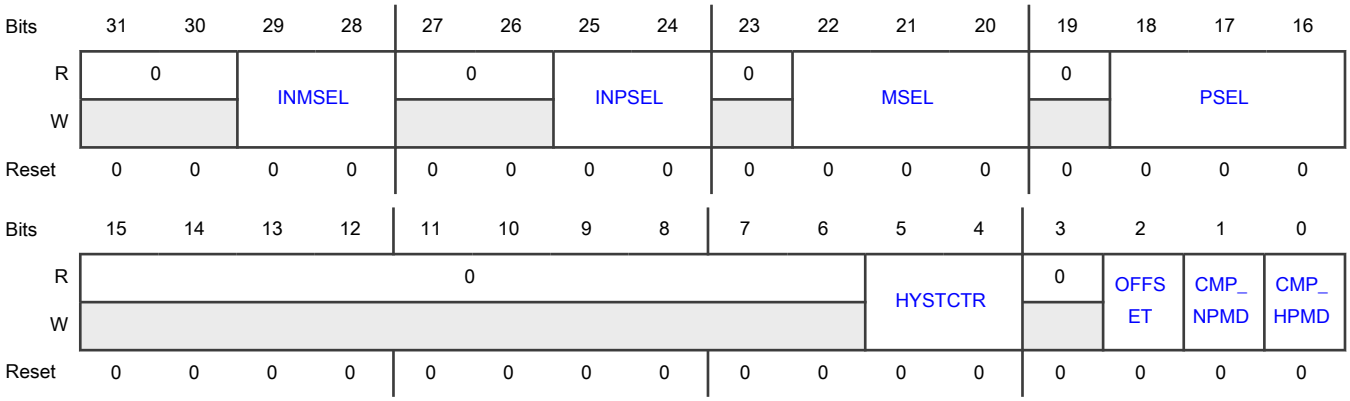
50.7.1.5 Comparator Control Register 2 (CCR2)

Contains configuration options for comparator operation, such as selecting the plus and minus comparator inputs and the hysteresis levels.

Offset

Register	Offset
CCR2	10h

Diagram



Fields

Field	Description
31-30 —	Reserved
29-28 INMSEL	Input Minus Select Selects the minus input of the comparator.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	NOTE These selections connect directly to the minus input of the comparator. 00 - IN0: from the 8-bit DAC output 01 - IN1: from the analog 8-1 mux 10 - Reserved 11 - Reserved
27-26 —	Reserved
25-24 INPSEL	Input Plus Select Selects the plus input of the comparator. **NOTE** These selections connect directly to the plus input of the comparator. 00 - IN0: from the 8-bit DAC output 01 - IN1: from the analog 8-1 mux 10 - Reserved 11 - Reserved
23 —	Reserved
22-20 MSEL	Minus Input MUX Select Selects the input used for the negative mux. See the chip-specific HSCMP information to get the detailed connections. **NOTE** MSEL has no effect in trigger mode. 000 - Input 0m 001 - Input 1m 010 - Input 2m 011 - Input 3m 100 - Input 4m 101 - Input 5m 110 - Reserved 111 - Internal DAC output
19	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
18-16 PSEL	Plus Input MUX Select Selects the input used for the positive mux. See the chip-specific HSCMP information to get the detailed connections. **NOTE** PSEL has no effect in trigger mode. 000 - Input 0p 001 - Input 1p 010 - Input 2p 011 - Input 3p 100 - Input 4p 101 - Input 5p 110 - Reserved 111 - Internal DAC output
15-6 —	Reserved
5-4 HYSTCTR	Comparator Hysteresis Control Selects the level of internally generated hysteresis for the comparator output. See the chip data sheet to get the specific values for each hysteresis level. 00 - Level 0 01 - Level 1 10 - Level 2 11 - Level 3
3 —	Reserved
2 OFFSET	Comparator Offset Control 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. 0 - Level 0: The hysteresis selected by HYSTCTR is valid for both directions (rising and falling). 1 - 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	CMP Nano Power Mode Select

Table continues on the next page...

Table continued from the previous page...

Field	Description
CMP_NPMD	Enables Nano Power mode for the comparator. 0 - Disable (Mode is determined by CMP_HPMD.) 1 - Enable
0 CMP_HPMD	CMP High Power Mode Select Selects Low or High Power(Speed) mode for the comparator. **NOTE** Valid only when not in Nano Power mode (CMP_NPMD=0). 0 - Low power(speed) comparison mode 1 - High power(speed) comparison mode

50.7.1.6 DAC Control Register (DCR)

Contains configuration options for enabling the DAC.

Offset

Register	Offset
DCR	18h

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0								DAC_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	DACOE	0						VRSEL	0						DAC_HPMD	DAC_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-24 —	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
23-16 DAC_DATA	DAC Output Voltage Select Selects the DAC output (DACO) voltage from one of 256 distinct levels. The DACO range is from Vin/256 to Vin: DACO = (Vin/256) * (DAC_DATA + 1)
15 DACOE	DAC Output Enable Enables the DAC output to be available for other on-chip peripherals. 0 - Disable 1 - Enable
14-9 —	Reserved
8 VRSEL	DAC Reference High Voltage Source Select Selects the high voltage reference source for the Vin supply of the DAC's resistor ladder network. See the chip-specific HSCMP information for the source of vrefh0 and vrefh1. 0 - vrefh0 1 - vrefh1
7-2 —	Reserved
1 DAC_HPMD	DAC High Power Mode Select Enables the DAC high power mode. 0 - Disable 1 - Enable
0 DAC_EN	DAC Enable Enables the DAC. When disabled, the DAC is powered down to conserve power. 0 - Disable 1 - Enable

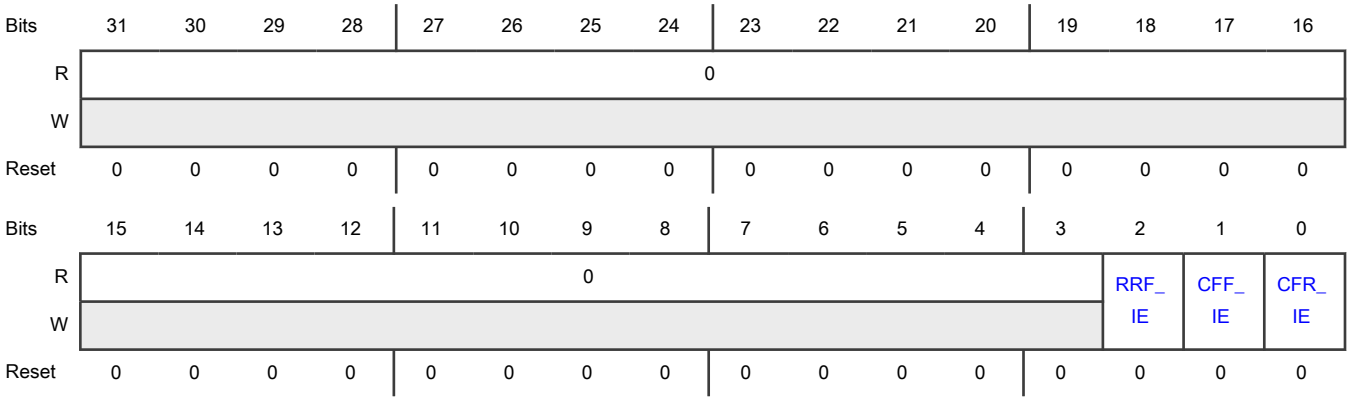
50.7.1.7 Interrupt Enable Register (IER)

Provides enable bits for the comparator and round-robin flag interrupts.

Offset

Register	Offset
IER	1Ch

Diagram



Fields

Field	Description
31-3 —	Reserved
2 RRF_IE	Round-Robin Flag Interrupt Enable Enables the Round-Robin Flag interrupt. 0 - Disable 1 - Enable: Assert an interrupt when the comparison result changes for a given channel.
1 CFF_IE	Comparator Flag Falling Interrupt Enable Enables the Comparator Flag Falling interrupt. 0 - Disable 1 - Enable: Assert an interrupt when CFF is set.
0 CFR_IE	Comparator Flag Rising Interrupt Enable Enables the Comparator Flag Rising interrupt. 0 - Disable 1 - Enable: Assert an interrupt when CFR is set.

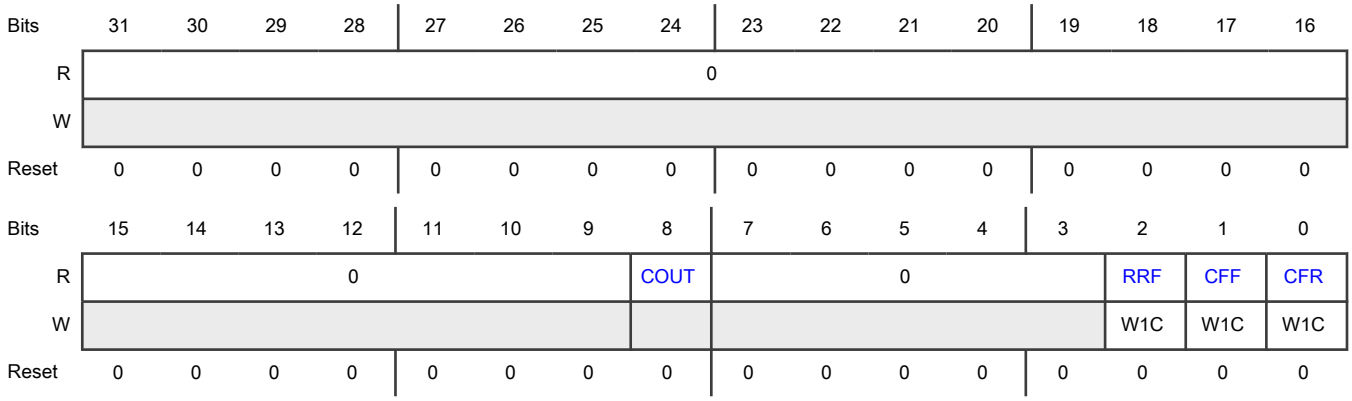
50.7.1.8 Comparator Status Register (CSR)

Provides the COUT, and CFF, CFR, RRF flags.

Offset

Register	Offset
CSR	20h

Diagram



Fields

Field	Description
31-9 —	Reserved
8 COUT	Analog Comparator Output Returns the current value of the analog comparator output, when read. The field is reset to 0 and reads as CCR1[COUT_INV] when the analog comparator module is disabled, that is, when CCR0[CMP_EN] = 0. Writing to this field is ignored.
7-3 —	Reserved
2 RRF	Round-Robin Flag Detects when any channel's last comparison result is different from the pre-set value in trigger mode. To clear RRF, write 1 to it. RRF is cleared when CCR0[CMP_EN] or RRCR0[RR_EN] is not set. 0 - Not detected 1 - Detected
1 CFF	Analog Comparator Flag Falling Detects when a falling edge on COUT occurs. CFF is cleared by writing 1 to it when CCR1[DMA_EN] is disabled. If CCR1[DMA_EN] is enabled, the flag is automatically cleared when DMA is done. CFF is cleared when CCR0[CMP_EN] is not set. 0 - Not detected 1 - Detected
0 CFR	Analog Comparator Flag Rising Detects when a rising edge on COUT occurs. CFR is cleared by writing 1 to it when CCR1[DMA_EN] is disabled. If CCR1[DMA_EN] is enabled, the flag is automatically cleared when DMA is done. CFR is cleared when CCR0[CMP_EN] is not set. 0 - Not detected 1 - Detected

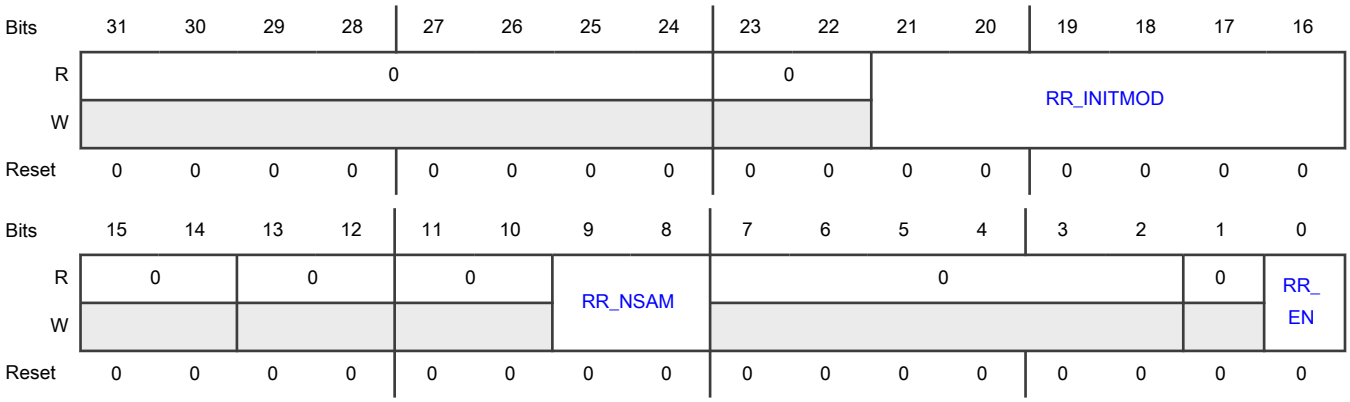
50.7.1.9 Round Robin Control Register 0 (RRCR0)

Contains configuration options for round robin operation, such as enabling it and specifying the initialization delay.

Offset

Register	Offset
RRCR0	24h

Diagram



Fields

Field	Description
31-24 —	Reserved
23-22 —	Reserved
21-16 RR_INITMOD	Initialization Delay Modulus Specifies the number of round-robin clock cycles used to determine the comparator and DAC initialization delay as specified in the chip datasheet. The initiallization delay is calculated as: RR_INITMOD * (round-robin clock period). For example, if the initialization delay is 80us and the round-robin clock is 100kHz, program RR_INITMOD to be 80us/10us = 8. 00_0000 - 63 cycles (same as 111111b) 00_0001-11_1111 - 1 to 63 cycles
15-14 —	Reserved
13-12	Reserved

Table continues on the next page...

Table continued from the previous page...

Field	Description
—	
11-10 —	Reserved
9-8 RR_NSAM	Number of Sample Clocks Specifies how many round-robin clock cycles to wait after the active channel is scanned before sampling the channel's comparison result. After the next cycle of the round-robin clock, the sampling takes place RR_NSAM clocks later. 00 - 0 clocks 01 - 1 clocks 10 - 2 clocks 11 - 3 clocks
7-2 —	Reserved
1 —	Reserved
0 RR_EN	Round-Robin Enable Enables round-robin operation. 0 - Disable 1 - Enable

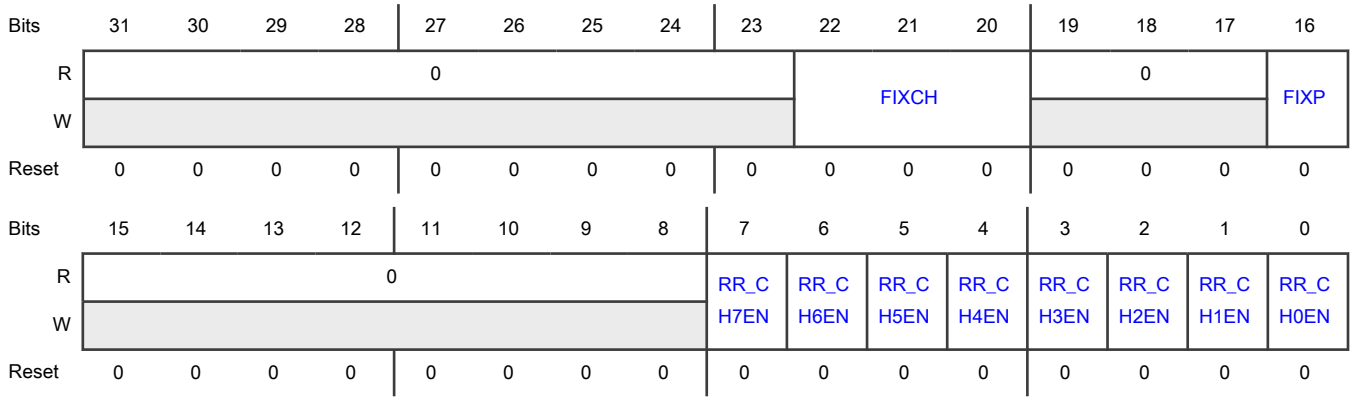
50.7.1.10 Round Robin Control Register 1 (RRCR1)

Contains configuration options for round robin operation, such as enabling individual channels to participate.

Offset

Register	Offset
RRCR1	28h

Diagram



Fields

Field	Description
31-23 —	Reserved
22-20 FIXCH	Fixed Channel Select Selects which channel in the mux port to fix for a given trigger mode application. 000 - Channel 0 001 - Channel 1 010 - Channel 2 011 - Channel 3 100 - Channel 4 101 - Channel 5 110 - Channel 6 111 - Channel 7
19-17 —	Reserved
16 FIXP	Fixed Port Fixes (locks) an analog mux port (Plus or Minus) for trigger mode. The inputs to the non-fixed port are swept during each round. 0 - Fix the Plus port. Sweep only the inputs to the Minus port. 1 - Fix the Minus port. Sweep only the inputs to the Plus port.
15-8 —	Reserved
7-0	Channel n Input Enable in Trigger Mode

Table continues on the next page...

Table continued from the previous page...

Field	Description
RR_CHnEN	Enables channel n of the non-fixed mux port to have its voltage value checked when in trigger mode. 0 - Disable 1 - Enable

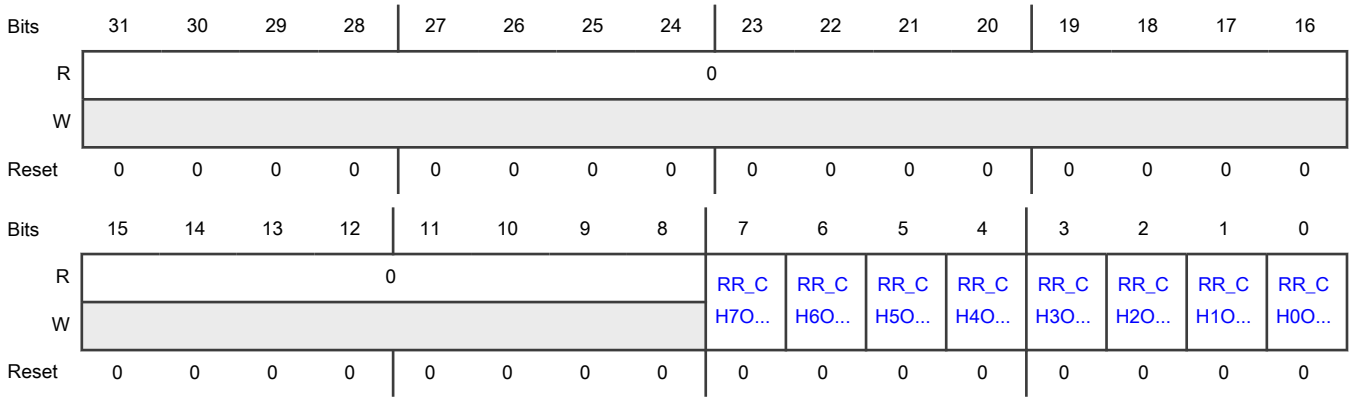
50.7.1.11 Round Robin Control and Status Register (RRCSR)

Contains the latest comparison results of the individual channels with the fixed mux port. It also allows software to define the pre-set state for each channel.

Offset

Register	Offset
RRCSR	2Ch

Diagram



Fields

Field	Description
31-8 —	Reserved
7-0 RR_CHnOUT	Comparison Result for Channel n Reading RR_CHnOUT returns the latest comparison result for channel n. Writing RR_CHnOUT defines the pre-set state for channel n.

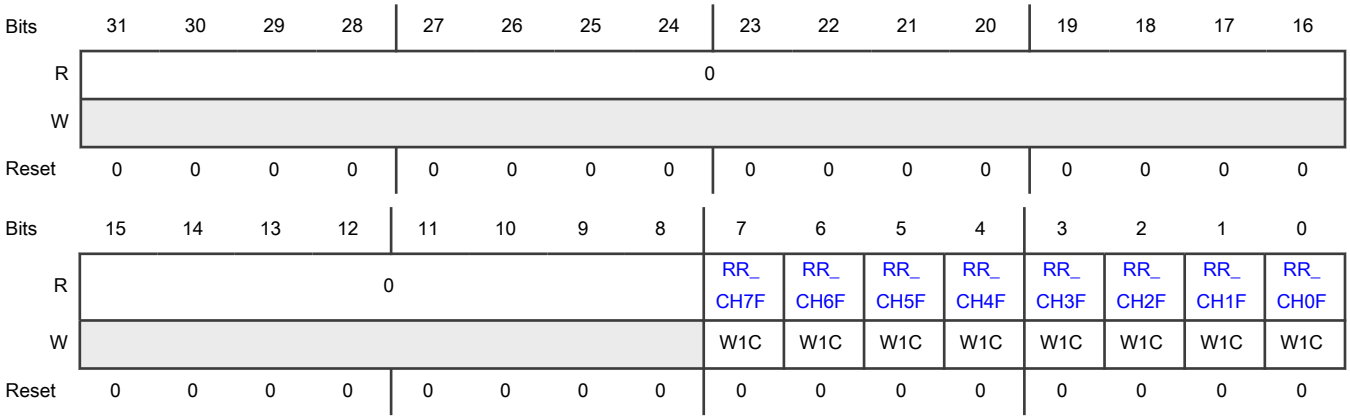
50.7.1.12 Round Robin Status Register (RRSR)

Contains individual channel flags indicating when a channel's last comparison result is different from its pre-set value.

Offset

Register	Offset
RRSR	30h

Diagram



Fields

Field	Description
31-8 —	Reserved
7-0 RR_CHnF	Channel n Input Changed Flag Indicates when the corresponding channel's last comparison result is different from its pre-set value. **NOTE** To clear a flag, write a 1 to it. 0 - Not different 1 - Different

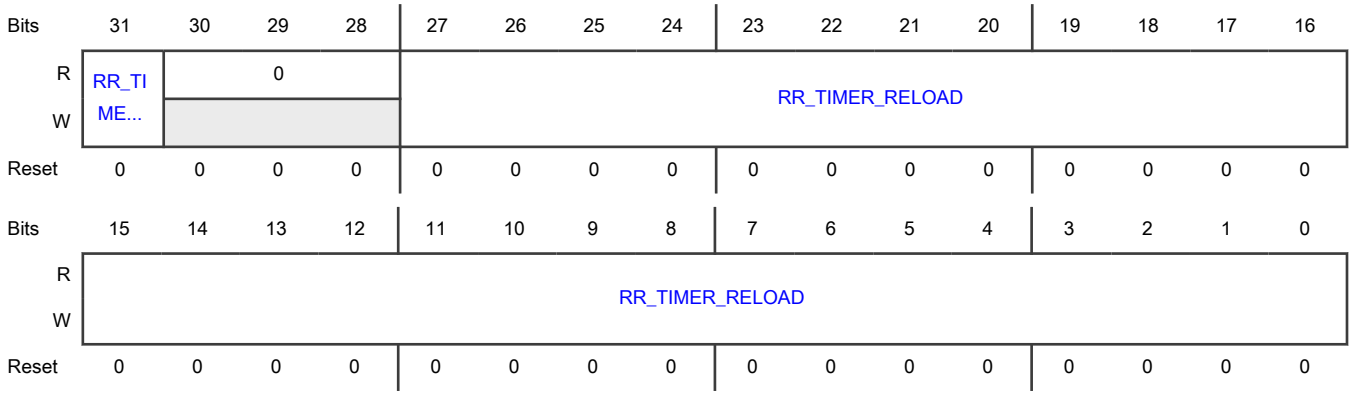
50.7.1.13 Round Robin Control Register 2 (RRCR2)

Provides the controls for Round-Robin internal trigger generation.

Offset

Register	Offset
RRCR2	38h

Diagram



Fields

Field	Description
31 RR_TIMER_EN	Round-Robin internal timer enable. 0 - Round-Robin internal timer is disabled. 1 - Round-Robin internal timer is enabled.
30-28 —	Reserved
27-0 RR_TIMER_RELOAD	Number of sample clocks This field establishes the repetitive count rate for the Round-Robin internal timer. Each time the timer counts down to zero it is reloaded with this value. The Round-Robin trigger signal will be generated at a rate of (RR_TIMER_RELOAD + 1) times the Round-Robin clock period.

Chapter 51
Operational Amplifier (OPAMP)

51.1 Chip-specific OPAMP information

Table 415. Reference links to related information

Topic	Related module	Reference
Full description	OPAMP	OPAMP
System memory map		System memory map
Clocking		Clock generation
Power management		Power management
Signal multiplexing	Port control	Signal multiplexing

51.1.1 Module instances

This device has three instances of OPAMP, OPAMP0, OPAMP1 and OPAMP2.

51.1.2 Positive Reference Voltage options

In the OPAMP_CTR Register, when:

- PREF = 0b00, VREFH3 is selected, which is DACx_OUT
- PREF = 0b01, VREFH0 is selected, which is VDDA supply rail
- PREF = 0b10, VREFH1 is selected, which is VREFO output
- PREF = 0b11, Floating, disables Positive Reference selection mux

NOTE

For DACx_OUT, OPAMP0 uses DAC0_OUT, OPAMP1 uses DAC1_OUT, and OPAMP2 uses DAC2_OUT.

51.2 About this module

51.2.1 Introduction

The OPAMP supports PGA amplifier.Positive and negative inputs of amplifier connect to internal channels.Selecting different gains by configuring registers which contains optional non-inverting or inverting gain application.The module is applicable to the signal processing stage before SARADC.

51.2.2 Features

The OPAMP includes the following features:

- Low noise and highspeed mode compatible
- Buffer mode
- Programmable gain
- Applicable to the signal processing stage before SARADC

51.2.3 Block diagram

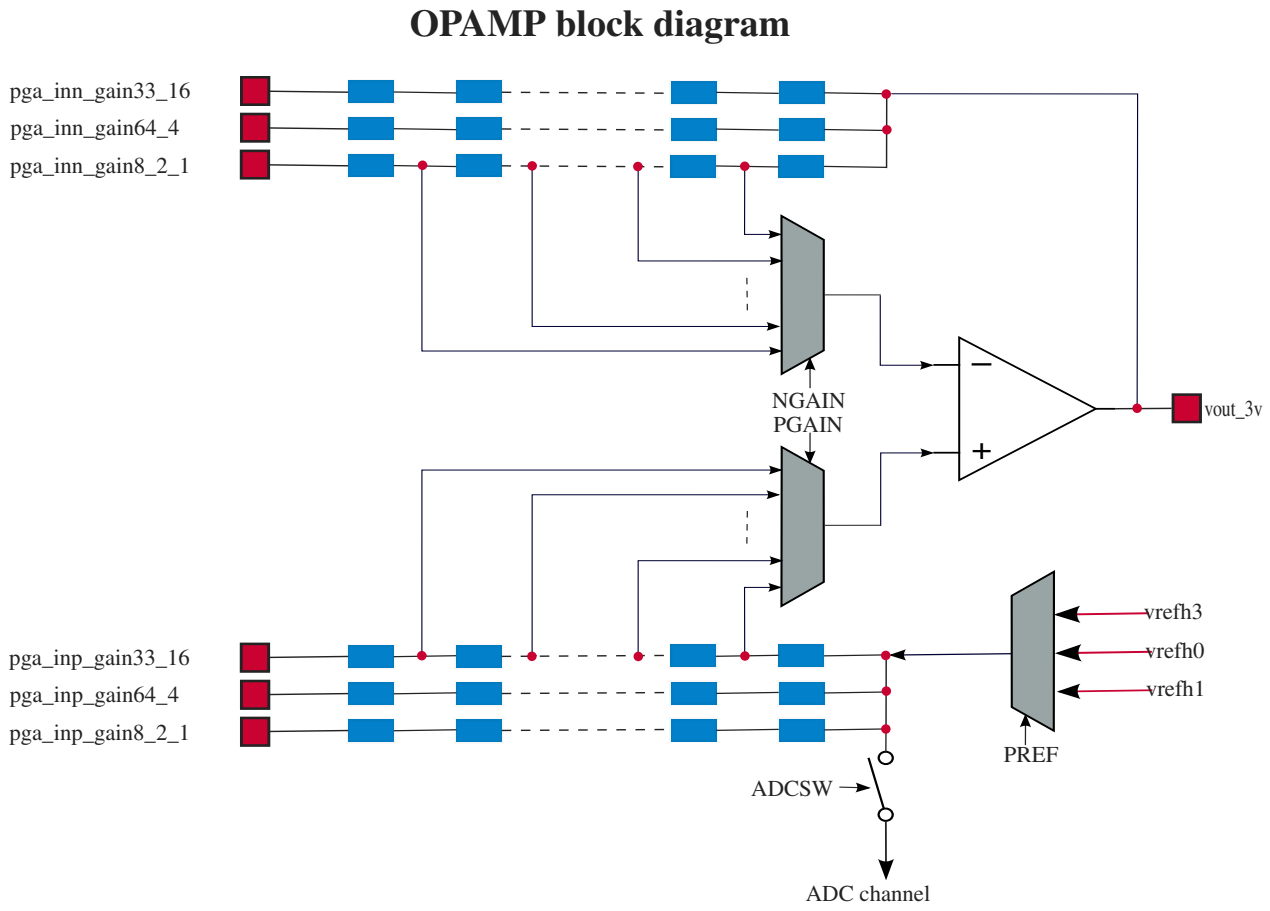
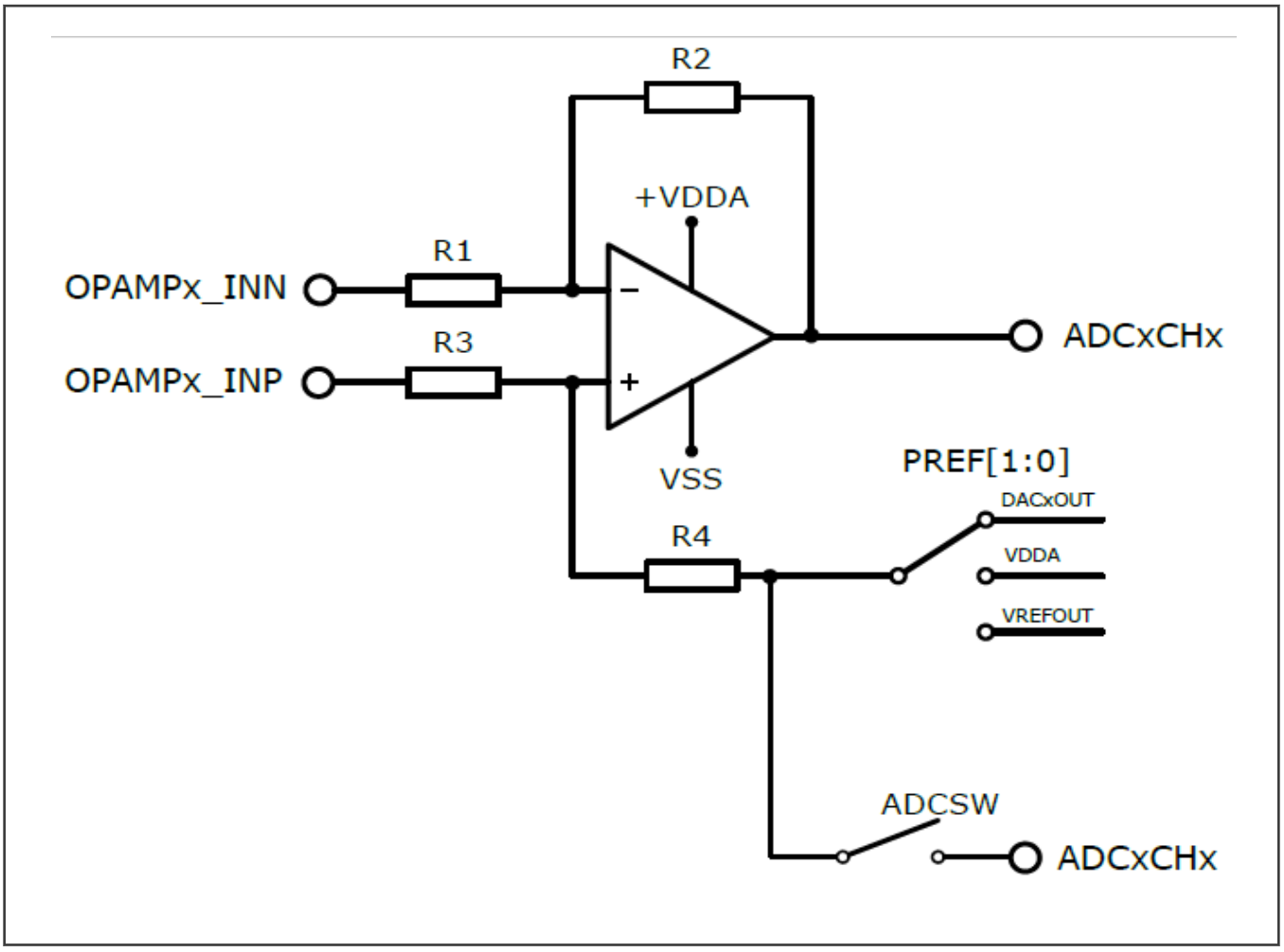


Figure 289. OPAMP block diagram

51.3 Functional description

This module supports optional gains.Selecting different gains by configuring register bits OPAMP_CTR[NGAIN] or OPAMP_CTR[PGAIN] which contains optional inverting or non-inverting gain application.

See below for calculation of OPAMP_OUT value from OPAMP_INP, OPAMP_INN and VPREF.



R2/R1 = NGAIN setup

R4/R3 = PGAIN setup

if R1=R3 and R2=R4

OUT = (R2/R1) * (IN+ - IN-) + offset

else

OUT = 1/R1*(((R1+R2)/(R3+R4)) * R4*IN+ - R2*IN-) + offset

OPAMP_CTR[NGAIN]	Description
3'b000	Buffer mode
3'b001	Ratio 1:1
3'b010	Ratio 1:2
3'b011	Ratio 1:4
3'b100	Ratio 1:8
3'b101	Ratio 1:16
3'b110	Ratio 1:33
3'b111	Ratio 1:64

OPAMP_CTR[PGAIN]	Description
3'b000	Reserved
3'b001	Ratio 1:1
3'b010	Ratio 1:2
3'b011	Ratio 1:4
3'b100	Ratio 1:8
3'b101	Ratio 1:16
3'b110	Ratio 1:33
3'b111	Ratio 1:64

51.3.1 Mode

This module supports low noise and high speed modes selected by configuring OPAMP_CTR[MODE].

51.4 Memory Map and register definition

This section includes the OPAMP module memory map and detailed descriptions of all registers.

51.4.1 OPAMP register descriptions

51.4.1.1 OPAMP memory map

OPAMP0 base address: 400B_4000h

OPAMP1 base address: 400B_8000h

OPAMP2 base address: 400B_B000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Version ID Register (VERID)	32	RO	0000_0000
4	Parameter Register (PARAM)	32	See section	0000_0001
8	OPAMP control register (OPAMP_CTR)	32	See section	0000_0000

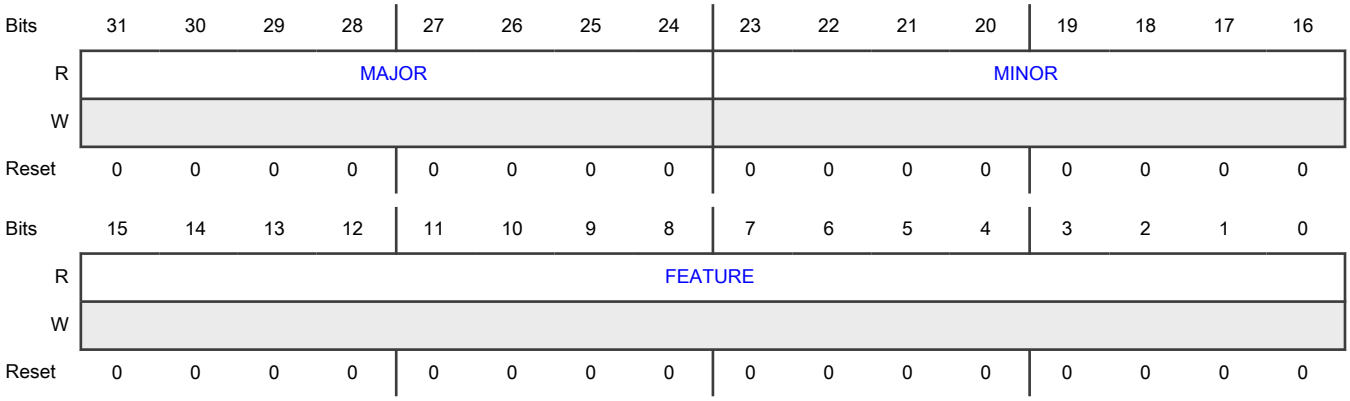
51.4.1.1.1 Version ID Register (VERID)

The Version ID register indicates the version integrated for this instance on the device and also indicates inclusion/exclusion of several optional features.

Offset

Register	Offset
VERID	0h

Diagram



Fields

Field	Description
31-24 MAJOR	Major Version Number This read only field returns the major version number for the specification.
23-16 MINOR	Minor Version Number This read only field returns the minor version number for the specification.
15-0 FEATURE	Feature Specification Number This read only field returns the feature set number.

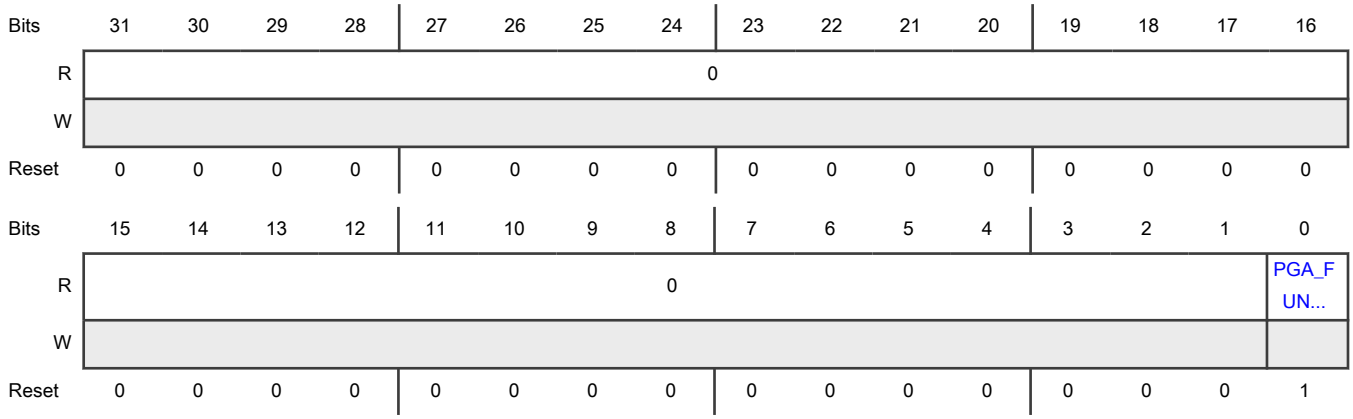
51.4.1.1.2 Parameter Register (PARAM)

The Parameter register indicates optional PGA functions or core ampilier for this instance on the device.

Offset

Register	Offset
PARAM	4h

Diagram



Fields

Field	Description
31-1 —	Reserved
0 PGA_FUNCTION	PGA Function Option This field is used to select different function. 0 - Core amplifier is enabled. 1 - PGA function is enabled.

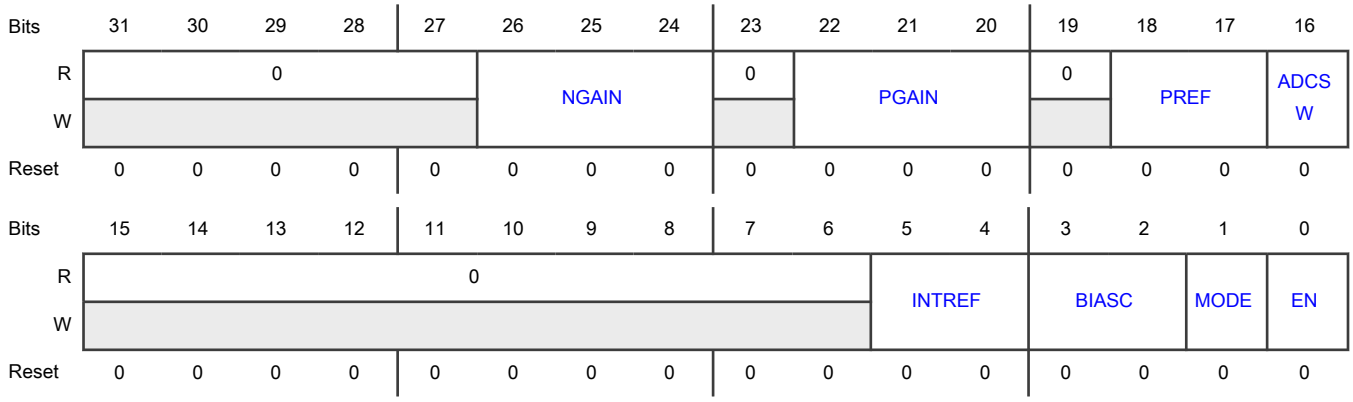
51.4.1.1.3 OPAMP control register (OPAMP_CTR)

OPAMP_CTR is used to configure the OPAMP module.

Offset

Register	Offset
OPAMP_CTR	8h

Diagram



Fields

Field	Description
31-27 —	Reserved
26-24 NGAIN	Negative PGA selection Negative programmable gain selection. 000 - Buffer. 001 - Ratio 1:1 010 - Ratio 1:2 011 - Ratio 1:4 100 - Ratio 1:8 101 - Ratio 1:16 110 - Ratio 1:33 111 - Ratio 1:64
23 —	Reserved
22-20 PGAIN	Positive PGA Selection. Positive programmable gain selection. 000 - Reserved. 001 - Ratio 1:1 010 - Ratio 1:2 011 - Ratio 1:4 100 - Ratio 1:8 101 - Ratio 1:16 110 - Ratio 1:33 111 - Ratio 1:64
19 —	Reserved
18-17 PREF	Positive Reference Voltage Selection Positive part reference voltage selection. 00 - Select vrefh3. 01 - Select vrefh0. 10 - Select vrefh1. 11 - Reserved.

Table continues on the next page...

Table continued from the previous page...

Field	Description
16 ADCSW	ADC Channel Switch Positive part reference voltage switch to ADC channel.Active in high.
15-6 —	Reserved
5-4 INTREF	Internal Reference Voltage Selection Internal reference voltage selection. 00 - Select vdda/2. 01 - Select vdda_3v. 10 - Select vssa_3v. 11 - Not allowed.
3-2 BIASC	Bias Current Trim Selection Bias current trim selection. 00 - Default. 01 - Increase current. 10 - Decrease current. 11 - Further decrease current.
1 MODE	Mode Selection Select one mode between low noise and high speed. 0 - Low noise mode. 1 - High speed mode.
0 EN	OPAMP Enable OPAMP is enabled or disabled. 0 - OPAMP is disabled 1 - OPAMP is enabled

Chapter 52
Voltage Reference (VREF)

52.1 Chip-specific VREF information

Table 416. Reference links to related information

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

NOTE

For either an internal or external reference if the VREF_OUT functionality is being used, VREF_OUT signal must be connected to an output load capacitor. Refer the device data sheet for more details.

If it is desired to use the VREF regulator and/or the chop oscillator in low power modes, the bandgap voltage reference must be enabled in these modes. CSR[HCGBEN] bit must be set in this device.

52.1.1 Module instances

This device has one instance of the VREF module, VREF0.

52.1.2 VREF Chip-specific initialization

- Enable the clock to the VREF (AHBCLKCTRLn[VREFn] = 1) in SYSCON registers. This enables the register interface and the peripheral function clock.
- Power up the VREF (PDRUNCFGCLR0[PDEN_VREFn] = 1) in PMC register.
- Configure the VREF as described in section Initialization/Application Information
  - UTRIM data are copied into VREF during ROM booting to achieve the performance stated in the device data sheet.

52.2 Introduction

The VREF can be used in applications to provide a reference voltage to external devices, or used internally in the device as a reference to analog peripherals (such as the ADC, DAC, or CMP). The Voltage Reference (VREF) can supply an accurate voltage output that can be trimmed in coarse or fine voltage steps. The voltage reference has 3 operating modes that provide different levels of supply rejection and power consumption.

52.2.1 Overview

The Voltage Reference provides a buffered reference voltage for use as an external reference, which can be set to from 1.0 V to 2.1 V. In addition, the buffered reference is available internally for use with on-chip peripherals (such as ADCs and DACs); see the chip-specific VREF information for details. When the VREF is enabled, the reference voltage is placed on a dedicated output pin.

- For fine tuning, use UTRIM[VREFTRIM] to trim the Voltage Reference output with (0.5 x VREF_OUT) mV resolution, where VREF_OUT is the value of the Voltage Reference output (as measured in volts). For example, when VREF_OUT = 1 V, the fine step resolution is: (0.5 x 1) = 0.5 mV.
- For coarse tuning, use UTRIM[TRIM2V1] to trim the Voltage Reference output with 100 mV resolution.

52.2.2 Features

- Programmable trim register with (0.5 x VREF_OUT) mV steps in fine-tuning and 100 mV steps in coarse-tuning. Upon reset, the trim register is automatically loaded with a factory-trimmed value.
- Programmable buffer mode selection:
  - Off
  - Bandgap enabled/standby (output buffer disabled)
  - Low power buffer mode (output buffer enabled)
  - High power buffer mode (output buffer enabled)
- Low-Power bandgap and High-Accurate bandgap selection
- Dedicated output pin, VREF_OUT

52.2.3 Low power modes

VREF can run when the chip is in low power modes. If it is desired to use the VREF regulator and/or the chop oscillator in low power modes, the system reference voltage (also referred to as the bandgap voltage reference) must be enabled in these modes. See the chip-specific VREF information for the details on enabling this mode of operation.

Having the VREF regulator enabled increases current consumption. In low power modes it may be desirable to disable the VREF regulator to minimize current consumption; however, the accuracy of the output voltage is reduced (by as much as several mVs) when the VREF regulator is not used.

NOTE

The assignment of module modes to core modes is chip-specific. For module-to-core mode assignments, see the chapter that describes power modes, such as a power management chapter.

52.2.4 VREF Signal Descriptions

The following table shows the Voltage Reference signals properties.

Table 417. VREF Signal Descriptions

Signal	Description	I/O
VREF_OUT	Internally-generated Voltage Reference output	O

NOTE

When the VREF output buffer is disabled, the status of the VREF_OUT signal is high-impedence.

52.3 Memory Map and Register Definition

52.3.1 VREF register descriptions

52.3.1.1 VREF memory map

VREF base address: 400B_5000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	VREF Version ID (VERID)	32	RO	0100_0000
4	VREF Parameter (PARAM)	32	See section	0000_0000
8	VREF Control and Status Register (CSR)	32	See section	0000_0000
10	VREF User Trim (UTRIM)	32	See section	0000_0000
1C	Unlock test registers (TEST_UNLOCK)	32	See section	0000_0000
24	VREF Test Trim 0 (TRIM0)	32	See section	0000_0000
28	VREF Test Trim 1 (TRIM1)	32	See section	0000_0000

52.3.1.1.1 VREF Version ID (VERID)

Offset

Register	Offset
VERID	0h

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	0	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	0	0	0	0

Fields

Field	Description
31-24	MAJOR

Table continues on the next page...

Table continued from the previous page...

Field	Description
MAJOR	Major Version Number: This read only field returns the major version number for the module specification
23-16 MINOR	MINOR Minor Version Number. This read only field returns the minor version number for the module specification
15-0 FEATURE	FEATURE Feature Specification Number This read only field returns the feature set number

52.3.1.1.2 VREF Parameter (PARAM)

Offset

Register	Offset
PARAM	4h

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																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-0 —	Reserved

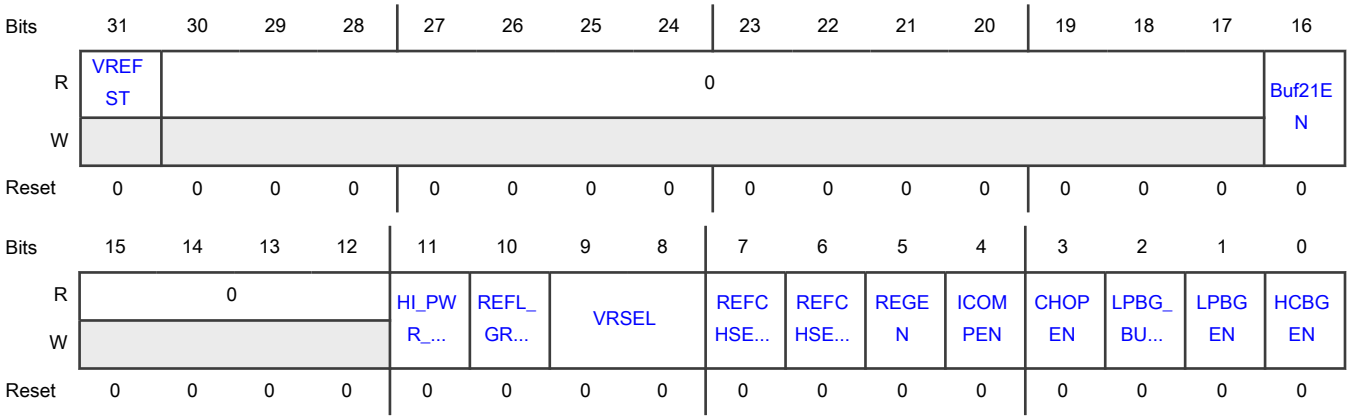
52.3.1.1.3 VREF Control and Status Register (CSR)

This register contains the control and status bits of VREF

Offset

Register	Offset
CSR	8h

Diagram



Fields

Field	Description
31 VREFST	Internal HC Voltage Reference stable Indicates that the bandgap reference within the Voltage Reference module has finished its startup and stabilization. **NOTE** VREFST bit is valid only when the chop oscillator is not being used. 0 - The module is disabled or not stable. 1 - The module is stable.
30-17 —	Reserved
16 Buf21EN	Internal buf21 Enable Programmable output voltage from 1.0v to 2.1v 0 - buf21 is disabled 1 - buf21 is enabled
15-12 —	Reserved
11 HI_PWR_LV	Buffer21 mode control

Table continues on the next page...

Table continued from the previous page...

Field	Description
10 REFL_GRD_SEL	Bit for the ground select 0 - vrefl_3v 1 - vssa
9-8 VRSEL	Control bits for voltage reference selection 00 - Internal bandgap 01 - Low power buffered 1v 10 - Buffer 2.1v output. Low-power buffer mode enabled
7 REFCHSELP_EN	Positive channel to ADC select enable. 0 - disable 1 - enable
6 REFCHSELN_EN	Negative channel to ADC select enable 0 - disable 1 - enable
5 REGEN	Regulator enable Enables the internal 1.75 V regulator to produce a constant internal voltage supply in order to reduce the sensitivity to external supply noise and variation. If it is desired to keep the regulator enabled in very low power modes, refer to the Chip Configuration details to see how this can be achieved. REGEN bit should be written to 1 to achieve the performance stated in the data sheet. **NOTE** See section "Internal voltage regulator" for details on the required sequence to enable the internal regulator. 0 - Internal 1.75 V regulator is disabled. 1 - Internal 1.75 V regulator is enabled.
4 ICOMPEN	Second order curvature compensation enable ICOMPEN bit should be written to 1 to achieve the performance stated in the data sheet. 0 - Disabled 1 - Enabled
3 CHOPEN	Chop oscillator enable. When set, the internal chopping operation is enabled and the internal analog offset will be minimized. If CHOPEN bit is set. Then the bit FLIP of TRIM0 has no use If the internal voltage regulator is being used (REGEN bit is set to 1), then the chop oscillator must also be enabled.

Table continues on the next page...

Table continued from the previous page...

Field	Description
	If the chop oscillatorwill be used in very low power modes, then the HC bandgap voltage reference must also be enabled. See the chip-specific VREF information (also known as "chip configuration" details) to see how this can be achieved. 0 - Chop oscillator is disabled. 1 - Chop oscillator is enabled.
2 LPBG_BUF_EN	Low power buffer enable for lpbg with latch function enable. NOTE: This bit can only be enabled when LPBG_EN =1. 0 - disable 1 - enable
1 LPBGEN	Low Power Bandgap enable This bit is used to enable the Low Power Bandgap. 0 - LP Bandgap is disabled 1 - LP Bandgap is enabled
0 HCBGEN	HC Bandgap enabled This bit is used to enable the HC Bandgap. NOTE: This bit can only be enabled when LPBG_EN =1. 0 - HC Bandgap is disabled 1 - HC Bandgap is enabled

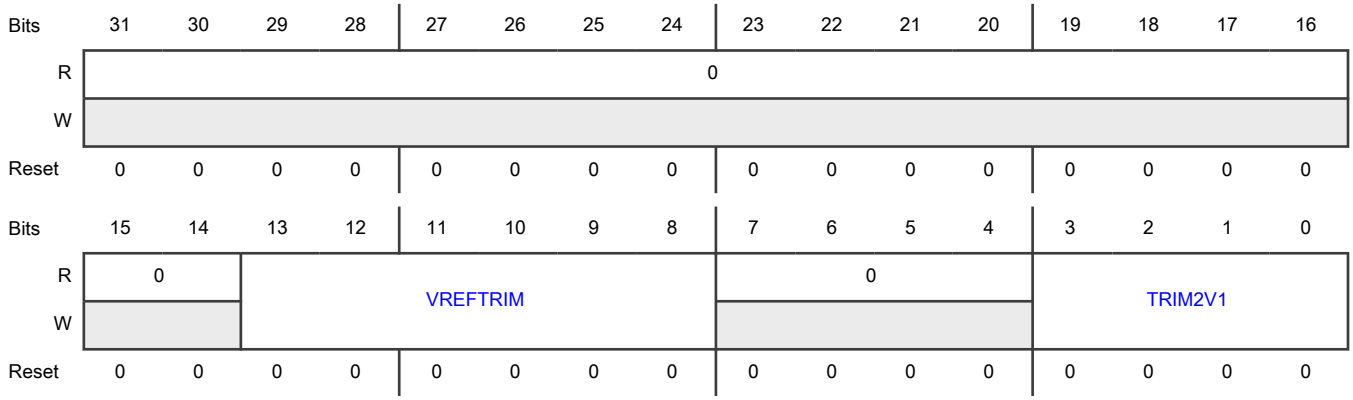
52.3.1.1.4 VREF User Trim (UTRIM)

This register contains the trim data can be read/write in USER mode

Offset

Register	Offset
UTRIM	10h

Diagram



Fields

Field	Description
31-14 —	Reserved
13-8 VREFTRIM	VREF Trim bits These bits change the resulting VREF by approximately ± (0.5 x VREF_OUT) mV for each step. The trim value is stored in IFR and loaded out of reset.
7-4 —	Reserved
3-0 TRIM2V1	VREF 2.1V Trim Bits These bits supply configurable bits for vref output. Change the resulting VREF output by approximately ± 100 mV for each step. The trim value is stored in IFR and loaded out of reset.

52.3.1.1.5 Unlock test registers (TEST_UNLOCK)

Unlocks read/write for test registers. After reset, SW can not read/write into test register. After SW write into bit[15:1] value 15'h5aa5, the read/write is unlocked.

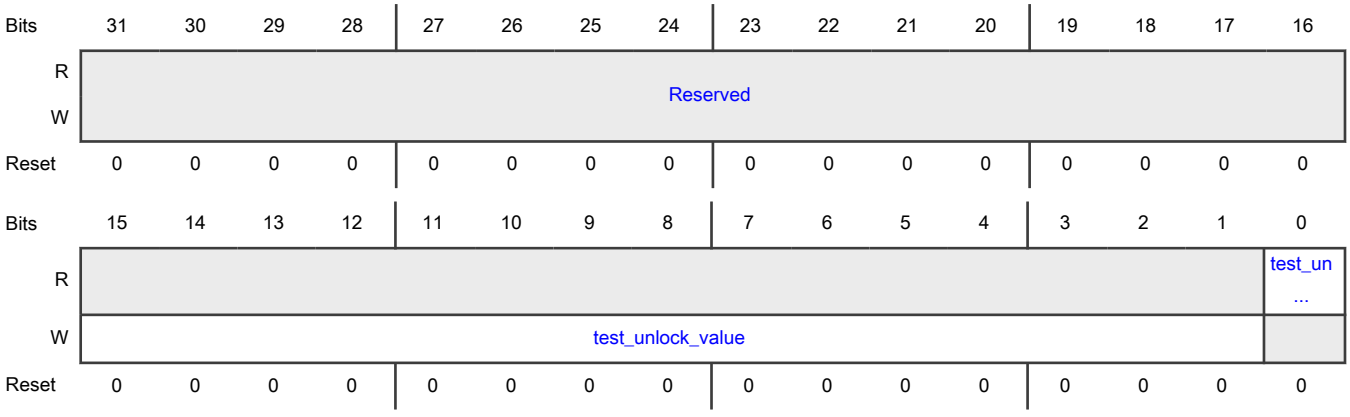
NOTE

Do not modify this register. This register is configured by the SDK drivers. Changes to this register can cause application failure. NXP is not responsible for any change to this register and is not obligated to provide support.

Offset

Register	Offset
TEST_UNLOCK	1Ch

Diagram



Fields

Field	Description
31-16 —	Reserved
15-1 test_unlock_value	Test unlock value After reset, SW can not read/write into test registers, if SW write value 15'h5aa5 into this 15 bit field. The read/write will be unlocked.
0 test_unlock	Test_unlock status bit This bit will be set "1", if SW write 15'h5aa5 into test_unlock_value. 0 - Lock read/write into test register 1 - Unlock read/write into test register

52.3.1.1.6 VREF Test Trim 0 (TRIM0)

This register contains the trim values read/writen in func test mode.

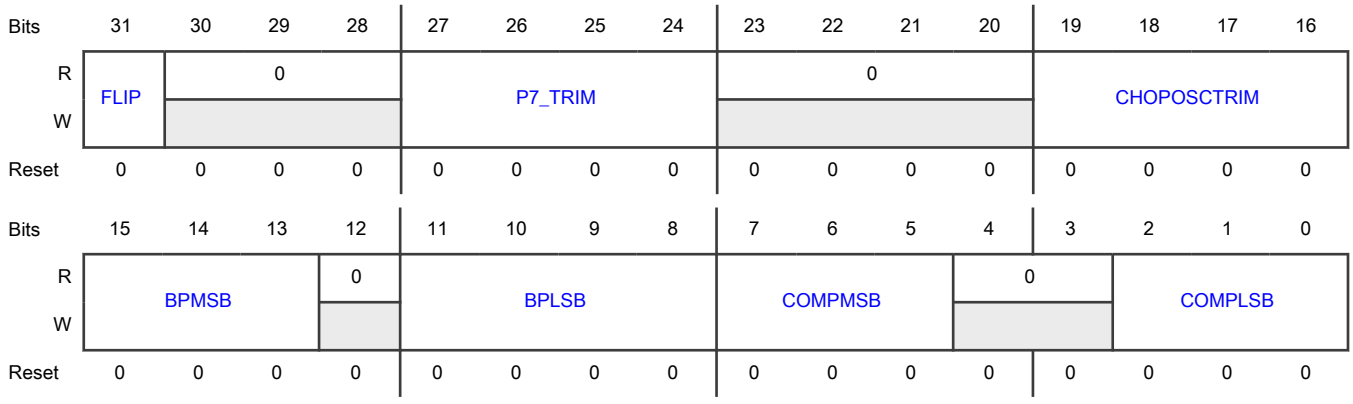
NOTE

Do not modify this register. This register is configured by the SDK drivers. Changes to this register can cause application failure. NXP is not responsible for any change to this register and is not obligated to provide support.

Offset

Register	Offset
TRIM0	24h

Diagram



Fields

Field	Description
31 FLIP	Amplifier Polarity Reverses the amplifier polarity. FLIP can be used to test bandgap amplifier offset. The trim value is stored in IFR and loaded out of reset.
30-28 —	RESERVED
27-24 P7_TRIM	P7_TRIM P7TRIM bits are used for trimming the 0.7 V for V to I at 3 mV/bit. The trim value is stored in IFR and loaded out of reset. 0000 - VREF 2.1V is enabled 0001 - VREF 2.1V is disabled
23-20 —	RESERVED
19-16 CHOPOSCTRIM	CHOPOSCTRIM CHOPOSCTRM bits trim the chop oscillator clock. The center frequency of the chop oscillator is 250 kHz at the trim value of 4'b1000. The trim step size is 5% time domain when chop oscillator is enabled.
15-13 BPMSB	BPMSB BPMSB bits are used for thin trimming HC bandgap temperature curvature. BPMSB bits are active low and will shift the bandgap inflection point ~15 °C/bit. The trim value is stored in IFR and loaded out of reset.
12 —	RESERVED
11-8 BPLSB	BPLSB BPLSB bits are used for thin trimming HC bandgap temperature curvature. BPLSB bits are active low and will shift the bandgap inflection point ~2 °C/bit. The trim value is stored in IFR and loaded out of reset.
7-5 COMPMSB	COMPMSB COMPMSB bits are used for trimming the curvature used for the second-order temperature compensation. The trim value is stored in IFR and loaded out of reset.
4-3 —	RESERVED
2-0 COMPLSB	COMPLSB COMPLSB bits are used for trimming the curvature used for the second-order temperature compensation. The trim value is stored in IFR and loaded out of reset.

52.3.1.1.7 VREF Test Trim 1 (TRIM1)

This register contains the trim values read/writen in func test mode.

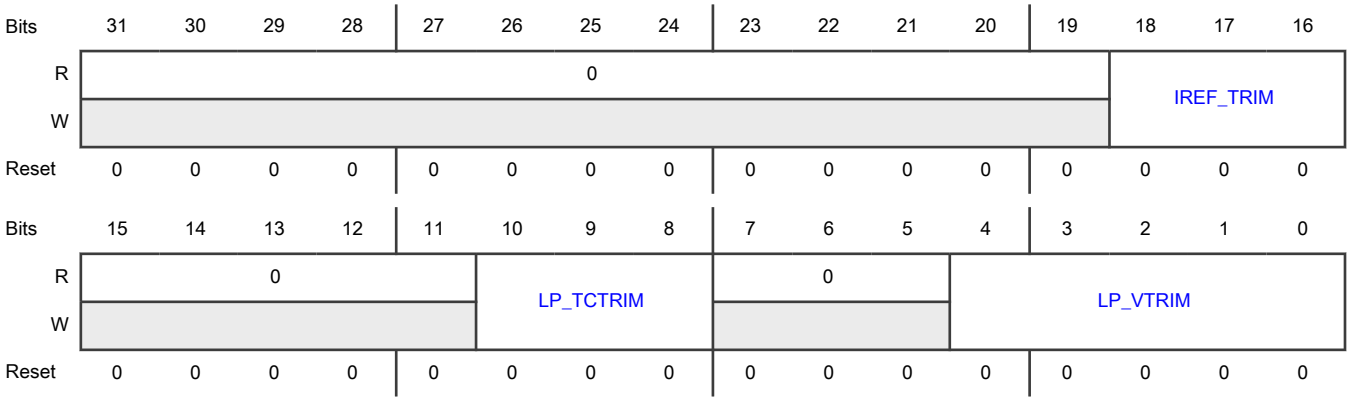
NOTE

Do not modify this register. This register is configured by the SDK drivers. Changes to this register can cause application failure. NXP is not responsible for any change to this register and is not obligated to provide support.

Offset

Register	Offset
TRIM1	28h

Diagram



Fields

Field	Description
31-19 —	RESERVED
18-16 IREF_TRIM	IREF_TRIM Iref trim is the trimming bits for current bias for Low Power Bandgap
15-11 —	RESERVED
10-8 LP_TCTRIM	LP_TCTRIM TC Trim bits for low power bandgap
7-5 —	RESERVED
4-0 LP_VTRIM	LP Bandgap Voltage Trim Bits for low power voltage reference trim

52.4 Functional Description

The Voltage Reference is a bandgap buffer system. Unity gain amplifiers are used.

The VREF_OUT signal can be used by both internal and external peripherals in low and high power buffer mode. A 220 nF capacitor must always be connected between VREF_OUT and VSSA if the VREF is being used.

When CSR[HCBGEN] = 1, the Voltage Reference is enabled and diferent modes should be set by the bits CSR[Buf2V1EN] and CSR[HI_PWR_LV]

The following table shows all possible function configurations of the Voltage Reference.

Table 418. VREF_OUT value related to the TRIM2V1 value

UTRIM[TRIM2V1]	VREF_OUT
0000	1.0
0001	1.1
0010	1.2
0011	1.3
0100	1.4
0101	1.5
0110	1.6
0111	1.7
1000	1.8
1001	1.9
1010	2.0
1011	2.1
X	1.0

52.4.1 Voltage Reference Disabled

When (CSR[LPBG] = 0)and(CSR[HCBG] = 0)and(CSR[Buf21EN] = 0), the Voltage Reference is disabled, the VREF bandgap and the output buffers are disabled. The Voltage Reference is in off mode.

52.4.2 Voltage Reference Enabled

When Voltage Reference Enabled,we have following case

1. Enable the CSR[LPBG] or ADC send signal adc_en_vref to enable LPBG supply current bias for ADC use
2. Enable (CSR[LPBG] = 1)and (CSR[HCBG] = 1), this configuration enable VREF supply voltage on VREF_OUT

52.4.2.1 CSR[Buf21EN]=0,CSR[HI_PWR_LV]=X

The internal VREF bandgap is enabled to generate an accurate 1.0 V output that can be trimmed with the UTRM register's UTRIM[13:8] bitfield. The bandgap requires some time for startup and stabilization. CSR[VREFST] can be monitored to determine if the stabilization and startup is complete when the chop oscillator is not enabled.

The output buffer is disabled in this mode, and there is no buffered voltage output. The Voltage Reference is in standby mode. If this mode is first selected and the low power or high power buffer mode is subsequently enabled, there will be a delay before the buffer output is settled at the final value. This is the buffer start up delay (Tstup or t_{st}) and the value is specified in the appropriate device data sheet.

52.4.2.2 CSR[Buf21EN]=1,CSR[HI_PWR_LV]=1

The internal VREF bandgap is on. The high power buffer is enabled to generate a buffered voltage to VREF_OUT which can be programmable with 0.1v each step from 1.0-2.1 v with UTRIM[TRIM2V1]. It can also be used as a reference to internal analog peripherals such as an ADC channel or analog comparator input.

If this mode is entered from the standby mode (CSR[Buf21EN]=0) there will be a delay before the buffer output is settled at the final value. This is the buffer start up delay (Tstup) and the value is specified in the appropriate device data sheet. If this mode is entered when the VREF module is enabled then you must wait the longer of Tstup or until CSR[VREFST] = 1 when the chop oscillator is not enabled. If the chop oscillator is being used, you must wait the time specified by Tchop_osc_stup (chop oscillator start up time) to ensure the VREF output has stabilized.

In this mode, a 220 nF capacitor is required to connect between the VREF_OUT pin and VSSA.

52.4.2.3 CSR[Buf21EN]=1,CSR[HI_PWR_LV]=0

The internal VREF bandgap is on. The low power buffer is enabled to generate a buffered voltage to VREF_OUT which can be programmable with 0.1v each step from 1.0-2.1 v with UTRIM[TRIM2V1]. It can also be used as a reference to internal analog peripherals such as an ADC channel or analog comparator input.

If this mode is entered from the standby mode (CSR[Buf21EN]=0) there will be a delay before the buffer output is settled at the final value. This is the buffer start up delay (Tstup) and the value is specified in the appropriate device data sheet. If this mode is entered when the VREF module is enabled then you must wait the longer of Tstup or until CSR[VREFST] = 1 when the chop oscillator is not enabled. If the chop oscillator is being used, you must wait the time specified by Tchop_osc_stup (chop oscillator start up time) to ensure the VREF output has stabilized.

In this mode, a 220 nF capacitor is required to connect between the VREF_OUT pin and VSSA.

52.4.3 Internal voltage regulator

The VREF module contains an internal voltage regulator that can be enabled to provide additional supply noise rejection. And internal oscillator which provide the chop clock.It is recommended that when possible, this regulator and oscillator be enabled to provide the optimum VREF performance.

If the chop function is being used, and internal voltage regulator must also be enabled. A specific sequence must be followed when enabling the internal regulator as follows:

1. Enable the internal regulator by setting CSR[REGEN] = 1)
2. Enable the chop oscillator (CSR[CHOPEN] = 1)
3. Enable the (CSR[LPBGEN]=1) and (CSR[HCBGEN]=1)

52.5 Initialization/Application Information

The Voltage Reference requires some time for startup and stabilization. After CSR[HCBG] = 1, CSRs[VREFST] can be monitored to determine if the stabilization and startup is completed when the chop oscillator is not enabled. When the chop oscillator is enabled, the settling time of the internal bandgap reference is defined by Tchop_osc_stup (chop oscillator start up time). You must wait this time (Tchop_osc_stup) after the internal bandgap has been enabled to ensure the VREF internal reference voltage has stabilized.

When the Voltage Reference is already enabled and stabilized, changing CSR[HI_PWR_LV] will not clear CSR[VREFST] but there will be some startup time before the output voltage at the VREF_OUT pin has settled. This is the buffer start up delay (Tstup) and the value is specified in the appropriate device data sheet. Also, there will be some settling time when a step change of the load current is applied to the VREF_OUT pin. When the 1.75V VREF regulator is disabled, the VREF_OUT voltage will be more sensitive to supply voltage variation. It is recommended to use this regulator to achieve optimum VREF_OUT performance.

The CSR[CHOPEN], CSR[REGEN] and CSR[ICOMPEN] bits must be written to 1 to achieve the performance stated in the device data sheet.

NOTE

See section "Internal voltage regulator" for details on the required sequence to enable the internal regulator.

Chapter 53
Debug Subsystem

53.1 Chip-specific Debug information

Table 419. Reference links to related information

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

53.1.1 Module instances

This device has one instance of the DEBUG module, DEBUGGER_MAILBOX0.

53.2 Overview

The Debug Subsystem chapter outlines the debug capabilities implemented in this chip. Debugging is supported through Arm's Serial Wire Debug (SWD) interface. This chapter is intended for debug tool developers and assumes that you know Arm's SWD interface and Coresight (TM) debug and trace technology.

NOTE

JTAG is used on this device for boundary scan and production test only and cannot be used for debugging purposes.

53.2.1 Block diagram

This figure shows top-level debug ports and connections in this device.

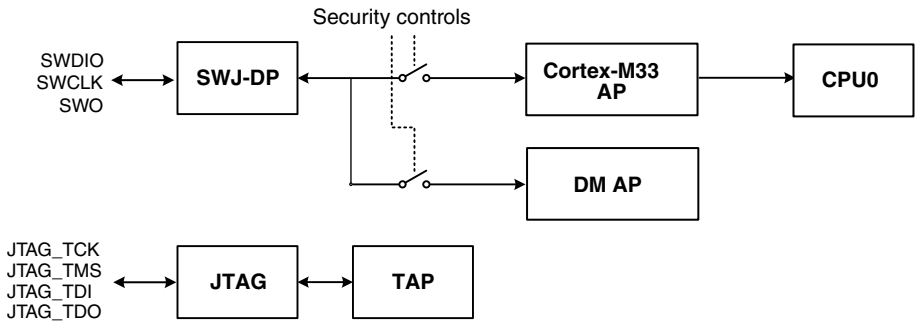


Figure 290. Serial Wire Debug (SWD) internal connections

Blocks SWJ-DP and DM-AP are always enabled and are accessible through the SWD interface. The remaining blocks are enabled or disabled under hardware state machine and software control.

- DAP: Debug access port with Serial Wire port (SWJ-DP) which interprets incoming data and routes to appropriate Access Port (AP).
- CPU0 AP: Debug access port for Cortex-M33 core instantiated as CPU0.
- DM-AP: Debug access port for debug mailbox. Debug Mailbox sends and receives messages from code executing from the ROM.
  - DM-AP is always enabled and the external world can send data to and receive data from the ROM.
  - DM-AP implements NXP debug authentication protocol version 1.0.

53.2.2 Features

- Supports Arm SWD mode.
- Includes. Cortex-M33 CPU instruction trace capability via trace port, and output via a Serial Wire Viewer.
- Provides direct debug access to all memories, registers, and peripherals.
- No target resources required for debugging session.
- Can set instruction breakpoints.
- Can set data watchpoints, which can be used as triggers.
- Instrumentation Trace Macrocell (ITM) allows additional software-controlled trace for the CPU.

53.2.3 Glossary

Table 420. Glossary

Abbreviation	Term	Description
RoT	Root of Trust	Vendor-owned key pair that authorizes data assets via cryptographic signatures. The public part of the key is typically pre-configured in products so that data from untrusted sources can be cryptographically verified. The vendor public key used by the device to verify the signature of this DC. (The corresponding private key was used to sign the DC).
RoTpub	RoT Public Key	The vendor public key used by the device to verify the signature of this DC. (The corresponding private key was used to sign the DC.)
RoTID	RoT Identifier	RoTID allows the debugger to infer which RoT public keys are acceptable to the device. If the debugger does not provide such a credential, the authentication process fails.
RoTMETA	RoT metadata	The RoT metadata required by the device to corroborate: the ROTID sent in the DAC, the field in this DC, and any additional RoT state not stored within the device. This metadata allows different RoT identification, management, and revocation solutions to be handled.
SoCC	SoC Class	A unique identifier for a set of SoCs that require no SoC-specific differentiation in their debug authentication. The main usage is to allow a different set of debug domains and options to be negotiated between the device configuration and credentials. If the granularity of debug control warrants it, a class can contain a single revision of a single SoC model.

Table continues on the next page...

Table 420. Glossary (continued)

Abbreviation	Term	Description
DCK	Debug Credential Key	A user-owned key pair. The public part of the key is associated with a DC, the private part is held by the user and used to produce signatures during authentication.
DC	Debug Credential	A user public key and associated attributes, bound together and signed by a RoT, serves as an *identity*.
CC	Credential Constraint	In product configuration, CCs are limitations on the DCs that the device accepts for authentication. In DCs, CCs are vendor/RoT-authorized usages of the DC, as well as inputs to the desired debug behavior.
VU	Vendor Usage	A CC (constraint) value that is opaque to the debug authentication protocol itself but which can be leveraged by vendors in product-specific ways.
SoCU	SoC Usage	A CC (constraint) value that is a bit mask, and whose bits are used in an SoC-specific manner. These bits are typically used for controlling which debug domains are accessed via the authentication protocol. Device-specific debug options can also be managed in this way.
CB AB	Credential Beacon Authentication beacon	A value that is passed through the authentication protocol, which is not interpreted by the protocol but is instead made accessible to the application being debugged. A credential beacon is associated with a DC and is therefore vendor/RoT-signed. An authentication beacon is provided and signed by the debugger during the authentication process.
DCFG_*	Debug Config	Refers to device configuration settings stored in OTP
CMPA		Customer manufacturing programmable area
CFPA		Customer field-programmable area

53.3 Functional description

53.3.1 Basic configuration

This device supports Arm's Serial wire Debug (SWD) interface. SWD is the default function for pins PIO0_0 (SWCLK) and PIO0_9 (SWDIO) after a reset. If Serial Wire Output (SWO) is to be used, it must be enabled in the application code by selecting the SWO function on either PIO0_8 or PIO0_18 and enabling the trace clock (see External signals).

Debug access via remote host is controlled by the ROM and is only enabled when permitted through the device configuration and when the correct protocol is followed to initiate a debug session. If the device has been configured for debug authentication, then a debug session must be initiated following the correct authentication sequence. When a device is in the development life-cycle state, the debug session protocol described in Debug session protocol should be used.

53.3.2 Debug Access Port (DAP)

The DAP with a Serial Wire Port (SWJ-DP) interprets incoming data and routes it to the appropriate Access Port (AP). External I/O pins that interface with DAP are described in External signals. The DAP block is always enabled but the I/O pins that provide access to the SWD signals may be used for other functions controlled byhttps://nxp.acrolinx.cloud/ software.

53.3.3 Arm Cortex-M33 Access Port

The Debug Access Port (DAP) for the Arm Cortex-M33 processor is disabled during power-on-reset or during the assertion of the reset pin. The DAP is enabled by the ROM when the correct debug initiation procedures are followed. If DAP is not being used, the debug enablement protocol can be used to initiate a debug session. The debug authentication process allows control of the DBGEN, NIDEN, SPIDEN and SPNIDEN signals generated by the Cortex-M33 as described below.

- **DBGEN:** Invasive debugging of TrustZone for Armv8-M architecture defined non-secure domain
  - Breakpoints and watch points halt the processor on a specific activity.
  - A debug connection examines and modifies registers and memory, and it provides single-step execution.
- **NIDEN:** Noninvasive debugging of TrustZone for a defined non-secure domain
  - Collects information on instruction execution and data transfers.
  - Delivers trace to off-chip in real time to tools to merge data with source code on a development workstation for future analysis.
- **SPIDEN:** Invasive debugging of TrustZone for Armv8-M architecture defined secure domain
- **SPNIDEN:** Noninvasive debugging of TrustZone for Armv8-M architecture defined secure domain

53.3.4 Debugger Mailbox Access Port (DM-AP)

The Debugger Mailbox Access Port (DM-AP) offers a register-based mailbox accessible by CPU0 and the device debug port (DP) of the MCU. This port is always enabled and an external host communicating through the SWD interface can exchange messages and data with the boot code executing from ROM on CPU0. This port is used to implement the NXP Debug Authentication Protocol.

53.4 Debug session protocol

The boot ROM implements debug mailbox protocol to interact with tools via the SWD interface.

The protocol has following features:

- Based on requests and responses, where requests and responses use the same basic structure.
- Supports relatively large command and response data.
- All commands and responses are 32-bit word aligned.
- Supports data above 32-bits by using an ACK_TOKEN that moderates the transfer in 32-bit value chunks.

The ROM provides three debug methods/mechanisms to attach the debugger in a predictable manner:

- Start debug session
- Debug access trigger
- Debug authentication

Debug authentication is disabled by default. If it is required, it is set up during development, or when provisioning a device during the production phase of a product using the device.

When instructions are fetched from the ROM address range during boot, the DAP of CPU0 is disabled, regardless of device life-cycle state or DCFG_SOCU settings. This mechanism is referred as Boot-ROM protection in this document. The method to initiate a debug session varies depending on the device state and intended debug scenario. The scenarios described in the rest of these sections are:

- Debug session with uninitialized/invalid flash image or ISP mode. Flash is uninitialized (as with a new part from the NXP factory) or does not contain a valid, bootable image. If the flash contains no valid image, the ROM proceeds into In-System Programming (ISP) mode and waits to be booted via one of its serial interfaces. In ISP mode, the debug interface is disabled.
- Debug session with valid application in flash. Program control is in ISP mode, initiated because the ISP pin was asserted on the device at reset.

- Debug session attaching to a running target. Connecting to a device running a valid application in flash, with the intent to debug without updating flash (also called a "debug attach").
- Halting execution immediately following ROM execution. Connecting to a device running a valid application, with the intent to update flash with a new application.

53.4.1 Debug session with uninitialized/invalid flash image or ISP mode

When the device boots, there might not be a valid image in the boot media. In this case, the ROM-based program control enters In-System Programing (ISP) mode, and debug access is disabled for security reasons. The device might also be placed into ISP mode because the ISP has been asserted as the device leaves reset. This section describes how to establish a debug session for these scenarios.

To ensure that the state machine controlling debug mailbox commands is in a known state, the debugger can reset this logic. This reset is done by setting CSW[RESYNCH_REQ] to 1. The debugger must then reset the device by writing a 1 to CSW[CHIP_RESET_REQ]. After requesting resynchronization and resetting the device, the debugger reads the CSW register. The debugger must wait until the device has completed resynchronization, by reading a value of 0 from the lower 16 bits of the CSW register.

To start a debug session and control the exchange of debug information, use the DM-AP commands. Following a successful initial resynchronization, the debugger communicates with the device via 32-bit DM-AP commands to the REQUEST register in the Debug Mailbox. The debugger can read results via the RETURN register. The debugger should poll the RETURN register as it polled the CSW register following a resynchronization request, to ensure that transactions have completed successfully. Results are shown in DM-AP response codes.

To initiate a debug session over SWD while debug is disabled, the debug system must issue a Start Debug Session command to the ROM-based boot code. This command must follow debug mailbox protocol. Upon receiving the command, the boot code disables any unwanted peripheral and manages secrets before enabling debug access. After enabling debug access, the ROM enters a while (1) loop.

Once the Start Debug Session command and device reset have executed successfully, the AP for CPU0 is accessible. It can be used to set breakpoints and perform other tasks, as with other Arm Cortex-M devices.

The code sample below shows how a debug session is initiated for the scenarios described above:

```
// Pseudo Code Syntax
// -------------------------------------------------------------------------
// WriteDP "register" "value"
// value = ReadDP "register"
// AP transactions presume the DM AP is selected
// WriteAP "register" "value"
// value = ReadAP "register" "value"
// -------------------------------------------------------------------------
```

```
// Read AP ID register to identify DM AP at index 2
WriteDP 2 0x020000F0
// The returned AP ID should be 0x002A0000
value = ReadAP 3
print "AP ID: ", value
```

```
// Select DM AP index 2
WriteDP 2 0x02000000
// Write DM RESYNC_REQ + CHIP_RESET_REQ
WriteAP 0 0x21
// Poll CSW register (0) for zero return, indicating success
value = -1
while value != 0 {
value = ReadAP 0
}
print "RESYNC_REQ + CHIP_RESET_REQ: ", value
```

```
// Write DM START_DBG_SESSION to REQUEST register (1)
WriteAP 1 7
// Poll RETURN register (2) for zero return
value = -1
while value != 0 {
value = (ReadAP 2 & 0xFFFF)
}
print "DEBUG_SESSION_REQ: ", value
```

Following a successful debug connection, a flashloader is loaded into RAM to program the application to be debugged and to set required breakpoints in the code. After this process, a SYSRESET_REQ command should be issued, verifying that the ROM fully executes (as a deployed end application would) before reaching the downloaded application.

53.4.2 Debug session with valid application in flash

In this case, it is assumed that debug has not been disabled by an application in flash. The CPU0 AP is accessible and breakpoints can be set without resynchronizing the Mailbox hardware or issuing a Debug Session Request. The methods described in Debug session with uninitialized/invalid flash image or ISP mode may be used to simplify debug support implementations.

53.4.3 Debug session attaching to a running target

In this scenario, the device has booted and is running an application that has not disabled debug. The host system is attempting to connect to the device without resetting it and without updating flash memory. In this case, the CPU0 AP should be accessible and breakpoints can be set without the need to resynchronize the Mailbox hardware.

53.4.4 Halting execution immediately following ROM execution

Traditionally, debug systems may set a vector catch at the reset vector to break code execution, but the chip ROM prevents this method. To allow the debug system to halt execution immediately after the ROM completes preparations following a debug session request, set a watchpoint on a read access to address 50000040h.

The debugger should use this reset sequence:

1. Set the data watchpoint to halt the core on a read of address location 50000040h.
2. If all data watchpoint comparators are occupied, back-up one of the watchpoint settings and replace it with the above watchpoint location.
3. Reset the core and peripherals by setting AIRCR[SYSRESETREQ].
4. Wait for 100 msec to allow ROM to re-enable debug access.
5. Verify that the core has halted at the watchpoint by checking the DHCSR register.
6. If DHCSR read times out or returns an error response, the ROM has entered an ISP command-handling loop due to an invalid image in boot media.
    a. Execute start debug session sequence described in Debug session with uninitialized/invalid flash image or ISP mode.
    b. Wait for 10 msec.
    c. Enable the Cortex-M33 AP.
    d. Read DHCSR and issue HALT to Cortex-M33 AP.
7. Clear data watchpoint added in step 1. If watchpoint was set as described in step 2, then restore the watchpoint configuration.

53.5 Reset handling

The debug domain (DP, Cortex-M33 AP, DM-AP) is reset upon Power On Reset (POR) or pin reset (assertion of external nRESET). On other resets, the debug domain retains its state. The defined breakpoints and watchpoints persist even when the debugging tool issues a reset to the device.

53.6 Mailbox commands

This section describes the Request and Response message formats and available mailbox commands.

53.6.1 Request packet layout

The first word transmitted in a request is a header word containing the command ID and the number of following data words. The command packet is sent to the device by writing 32 bits at a time to the REQUEST register. When sending command packets greater than 32 bits, the debugger should read an ACK_TOKEN in the RETURN register before writing the next 32 bits.

The 32-bit words quantified by the header follow the header itself.

Table 421. Request register byte description

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	commandID[7:0]	commandID[15:8]	dataWordCount[7:0]	dataWordCount[15:8]
1	*data*			

The C structure definition for a request is:

```
struct dm_request {
        uint16_t commandID;
        uint16_t dataWordCount;
        uint32_t data[ ];
        };
```

53.6.1.1 DM-AP commands

DM-AP commands are written to the REQUEST register. An Exit DM_AP command follows one or more of the DM-AP commands in the table below to resume the normal device boot flow. This sequence does not occur after the Enter ISP mode and Start Debug Session commands.

Table 422. DM-AP commands

Name	Command code	Parameter/ Response	Description
Start DM-AP (legacy command)	01h	Parameters: None Response: 32-bit status	Causes the device to enter DM-AP command mode. Must be done prior to sending other commands. This command is provided for backward compatibility and does not need to be used.
Get CRP Level	02h	Parameters: None Response: 32-bit status	Returns CRP levels
Bulk Erase (legacy command)	03h	Parameters: None Response: 32-bit status	Erase the entire on-chip flash memory, excluding Protected Flash Regions (PFR)

Table continues on the next page...

Table 422. DM-AP commands (continued)

Name	Command code	Parameter/ Response	Description
Exit DM_AP (legacy command)	04h	Parameters: None Response: 32-bit status	Causes the device to continue normal debug flow.
Enter ISP Mode	05h	Parameters: dataWordCount: 1h data[0]: ISP mode enum. FFFFFFFFh - Auto detection 1h - UART 2h - I2C 4h - SPI 10h - USB-HID Others - Reserved Response: 32-bit status	Enter specified ISP mode. By default, ISP mode entry is determined by the state of the ISP boot selection pins at reset time. Usually this functionality is disabled through PFR configuration prior to field deployment. This command can be used to enter ISP mode in those situations or when ISP boot selection pins are used for some other board function.
Set FA Mode	06h	Parameters: None Response: 32-bit status	This command sets the part permanently in Fault Analysis (FA) mode to return to NXP factory. Upon receiving this command boot code in ROM, customer-sensitive assets (key codes) stored in PFR are erased. Also, the FA or RMA mode bit in the PFR field is set so suspect parts can be sent to NXP for FA/RMA testing.
Start DBG Session	07h	Parameters: None Response: 32-bit status	Start Debug Session Program control is in ROM memory context when instructions are fetched from a ROM memory address range. When program control is in ROM memory context during boot, the debug access is disabled regardless of the device life-cycle state or DCFG_SOCU settings. This command instructs the boot code in ROM to clean up memory and peripherals initialized by boot code: SysTick, UART, SPI, I2C, QSPI, SD/MMC, USB, MPU, and SAU. It also instructs the boot code to enter safe processor execution context for external debuggers to attach. When no valid image is in boot media, the program control enters ISP command handler loop (still in ROM memory context) and debug access is disabled. As a result, an external debug system must use this command to indicate to the ROM its intention of connecting to the debugger.

Table continues on the next page...

Table 422. DM-AP commands (continued)

Name	Command code	Parameter/ Response	Description
			Upon receiving the command, the ROM cleans all peripheral configurations and secrets before enabling debug access. After enabling debug access, the ROM enters a while(1) loop.
Debug Auth. Start	10h	Parameters: None Response: dataWordCount: 12Ch or 22Ch data[]: DAC	Start Debug Authentication Protocol. ROM responds to debugger with *DAC (Debug Authentication Challenge)* message
Debug Auth. Response	11h	Parameters: dataWordCount: 12Ch data[]: DAR Response: 32-bit status	Debug Authentication Response

53.6.2 Response packet layout

The first word transmitted in a response is a header word containing the command status and number of following data words. The command response can be read 32 bits at a time through the RETURN register. The initial 32 bits contains the response header as shown in the table below. When reading a response longer than 32 bits, the debugger should write the ACK_TOKEN to the REQUEST register after every read until the full response packet is received.

NOTE

To support legacy LPC command and response values, Bit_31 in the header indicates that the response follows new protocol structure. This bit is set when the new protocol is used.

Table 423. Response register byte description

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	bits[7:0]:commandStatus[7:0]	bits[7:0]:commandStatus[15:8]	bits[7:0]:dataWordCount[7:0]	bits[6:0]:dataWordCount[14:8] bits[7]:new_protocol[15]
1	*data*			

The C structure definition for a response is:

```
struct dm_response {
        uint16_t commandStatus;
        uint16_t dataWordCount;
        uint32_t data[];
        };
```

53.6.2.1 DM-AP response codes

Response codes for DM-AP commands are listed below. These commands can be read from the RETURN register. The upper 15 bits (bits 30:16) of the response code only contain relevant data (data word count related to the command) when the command is successful.

Table 424. DM-AP response codes

Return code (bits 15:0)	
0000h	Command succeeded
0001h	Debug mode not entered. This code is returned if other commands are sent prior to the Enter DM-AP command.
0002h	Command not recognized. A command was received other than the ones defined above.
0003h	Command failed

53.6.3 ACK_TOKEN

The ACK_TOKEN provides an acknowledgment and is used in the following ways:

- When a command has parameters, the debugger waits for ACK_TOKEN (sent through RETURN register) before sending next 32-bit value.
- When a response packet has data to send back to the debugger, the boot ROM waits for the debugger to send an ACK_TOKEN (through REQUEST register) before sending the next 32-bit value.

The upper 16 bits are set by the receiving end with the number of remaining words expected. The lower 16 bits are always set to A5A5h.

Table 425. ACK_TOKEN register byte description

Word	Byte 0	Byte 1	Byte 2	Byte 3
0	A5h	A5h	remainCount[7:0]	remainCount[15:8]

The C structure definition for a ACK_TOKEN is:

```
struct dm_ack_token {
        uint16_t token; /* always set to A5A5h */
        uint16_t remainCount; /* count of remaining word */
        };
```

53.6.4 Error handling

When an overrun occurs from either side of the communication, the appropriate error flag is set in the CSW register. The state machine hardware prevents further communication in either direction. The debugger must start with a new resynchronization request to clear the error flag.

53.7 Debug Credential Certificate (DC)

By prior construction, the debugger should already have a Debug Credential Key (DCK). The public key part of this key pair represents the identity of the debugger through the creation of a DC. It binds that public key to usage attributes, and is then authorized or signed by the vendor's RoT key.

Total DC size depends on protocol version and number of used root keys.

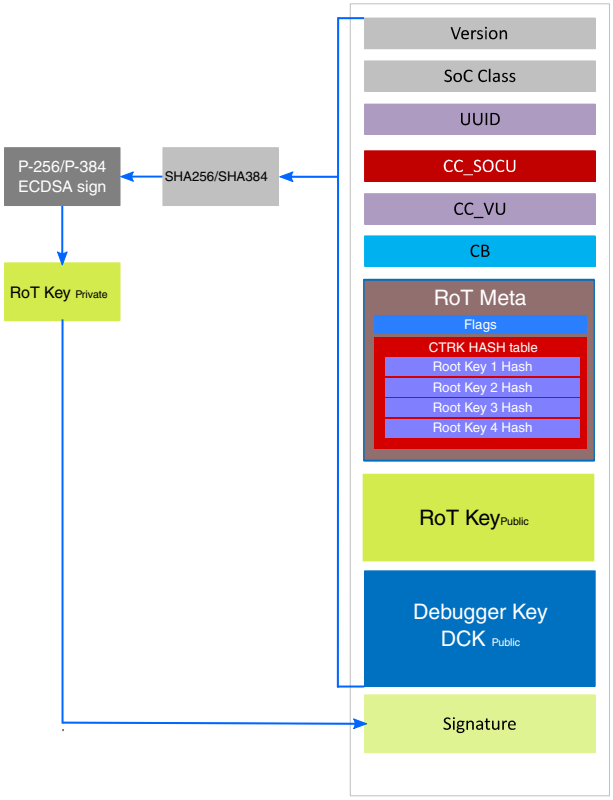


Figure 291. Debug Credential certificate fields

The data structure is a represented as a packed binary concatenation of its component fields as shown in the list below.

Table 426. Debug Credential Certificate fields

Name	Offset	Description	Size in bytes
VERSION	0000h	Identifies the Debug Authentication protocol version Version determined by ENF_CNSA field (CMPA) • 00020000h for v2.0, which uses ECDSA P-256 (ENF_CNSA = 00) • 00020001h for v2.1, which uses ECDSA P-384 (ENF_CNSA = 01, 10, 11)	4
SOCC	0004h	SoC class specifier Always set this field to 00000004h	4
UUID	0008h	Unique device identifier, if this certificate is used on a device configured for UUID-matching. If UUID matching is enabled, the certificate is restricted to a specific device, otherwise the certificate is enabled for whole SoC Class.	16
CC_SOCU	0018h	SoC specific Credential Constraints Specifies the debug access rights allowed for this certificate holder.	4
CC_VU	001Ch	Vendor Usage	4

Table continues on the next page...

Table 426. Debug Credential Certificate fields (continued)

Name	Offset	Description	Size in bytes
		Should match DCFG_VENDOR_USAGE field in Device Configuration for Credential Constraints (DCFG_CC). It can be used to revoke Debug Certificates.	
CB	0020h	Credential beacon that vendor has associated with this DC. Only the lower 16-bits of this field are effective. This field can extend the Debug Authentication process. When a nonzero value is used in this field, ROM defers opening debug access to the user application. The result of the authentication process is written to DBG_FEATURES register. The user application, after performing extended processing such as cleaning up critical keys and secrets, should copy the value to DBG_FEATURES_DP register to enable debug access. To aid user application, ROM stores beacon values in DEBUG_AUTH_BEACON register.	4
ROTMETA_FLAGS	0024h	See Table 427	4
ROTMETA_CTRK_HASH_TABLE	0028h	SHA-256 or SHA-384 of root public key calculated based on root certificate EC size. Between two and four root certificates can be specified. In the case where only one root certificate is used, ctrkHashTable is not present.	variable (0 - 192)
ROT_KEY_PUB	variable	X and Y coordinates of Root of Trust vendor public key used for signing this certificate.	64 / 96
DCK_KEY_PUB	variable	X and Y coordinates of Debugger public key	64 / 96
SIGNATURE	variable	R and S portion of ECDSA cryptographic signature (P-256 or P-384) by the RoT over the previous fields. This signature ensures that the DC is approved by the vendor for use by the debugger.	64 / 96

Table 427. ROTMETA_FLAGS

Field	Description
3:0	Reserved
7:4	Number of records in ctrkHashTable [1-4]. When equal to 1, ctrkHashTable is not present and the device computes hash from ROT_KEY_PUB and compares it to RKTH stored in CMPA
11:8	Determine which root cert number [0-3] was used for signing. Used only when more than one root certificate is specified.
30:12	Reserved
31	CA flag Should be always set to 1b in DC

53.7.1 Debug Authentication Challenge (DAC)

The debug authentication protocol begins with a DAC message, issued by the device to the debugger. The total message size is 104 bytes.

From a protocol perspective, the debugger must select a Debug Credential (DC) certificate that successfully authenticates based on the constraints provided in the DAC. This DC provides the required debug behavior post-authentication (for example, whether to debug secure world, with the desired credential beacon). If such a credential cannot be found, the debugger should report an error to the user.

NOTE

The debugger must also be able to produce signatures using the private key corresponding to the selected DC. This requirement exists so that any credential store can manage this association between credentials and corresponding private keys.

The named elements of this message are shown below.

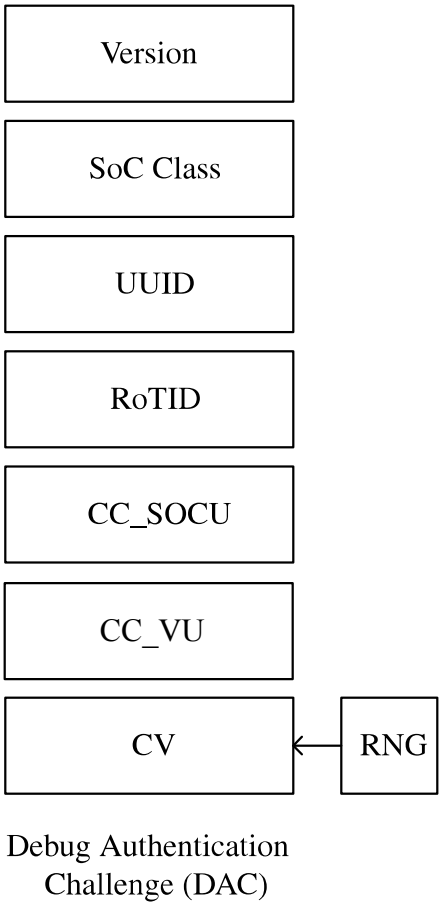


Figure 292. DAC fields

Table 428. DAC fields

Name	Offset	Description	Size in bytes
VERSION	000h	Identifies the Debug Authentication protocol version. Version is determined by ENF_CNSA field (CMPA)	4

Table continues on the next page...

Table 428. DAC fields (continued)

Name	Offset	Description	Size in bytes
		• 00020000h for v2.0, which uses ECDSA P-256 (ENF_CNSA = 00) • 00020001h for v2.1, which uses ECDSA P-384 (ENF_CNSA = 01, 10, 11)	
SOCC	004h	SoC class specifier This field is always set to 00000004h.	4
UUID	008h	Unique device identifier If UUID matching is enabled in DCFG_CC_SoCU, then the device includes its UUID in the challenge packet. Otherwise, this field is set to zeroes.	16
ROTID	018h	Metadata used to authenticate Certificate Authority key (RoT key) for signing DC certificate. SHA256 or SHA384 hashes of four RoT public keys are set in this field. **NOTE** On this device, the same RoT keys are used for certifying image signing keys and debug keys. 32-byte or 48-byte RKTH value. 4 bytes containing revocation flags. Only least significant 4 bits are used.	36 or 52
CC_SOCU and CC_VU	3Ch or 4Ch	Credential Constraints Specifies the debug access rights for this certificate holder. A 32-bit SOCU_PIN value. A 32-bit SOCU_DFLT value. A 32-bit DCFG_VENDOR_USAGE value.	12
CV	48h or 58h	Challenge Vector generated by the device. Device provides a random 32-byte value generated using the TRNG block.	32

53.7.2 Debug Authentication Response (DAR)

Before the debugger can formulate a response to the challenge, it should perform some checks to confirm the correctness of VER, SoCC, UUID, RoTID, and CC. It should find a matching DC.

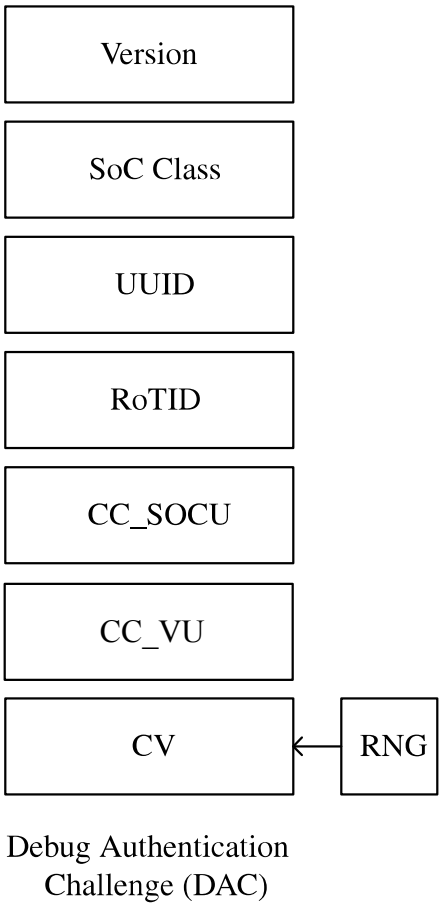


Figure 293. DAR fields

Table 429. DAR fields

Name	Description	Size in bytes
DC	Provides the credential, RoT public key, and more for the debugger as described in Debug Credential Certificate (DC).	variable
AB	The Authentication Beacon provided and signed by the debugger during the authentication process. See the Credential Beacon (CB) field in Debug Credential Certificate (DC) structure for details.	4
SIG	A cryptographic signature by the debugger that binds the previous two fields with the challenge vector from the DAC. SIG = ECDSA_SIGN (DCKpriv,SHA(DAR::DC II DAR::AB II DAR::CV)) • Uses the private key corresponding to the public key (DCK) of the selected DC, proving that debugger has possession of debugger private key. • Depending on which Debug Authentication protocol is used, ECDSA secp256r1 or secp384r1 is used for SIG function. SHA is either SHA256 or SHA384 based on the ECDSA curve type.	64 or 96

53.7.3 Device processing the DAR

The device Boot ROM processes the DAR received from the debugger. As part of the validation process, the device will:

- Verify the DC by validating the DC version, SoCC, UUID, RoT, VU, and DC signature.
- Verify that the DAR has a valid signature that binds it to the CV from the DAC.

If valid, it means that:

- The debugger possesses the private key corresponding to the vendor or RoT-signed credential.
- The credential satisfies the constraints specified in the device configuration.
- The response of the debugger to the challenge is produced and signed in response to the challenge (because of its cryptographic dependency on the challenge vector). The response is not replayed from a previous authentication where a different challenge vector is used.

After completion of processing the DAR:

- If authentication succeeds, debug access is granted.
- If authentication fails, no special response is issued. Further debug access requests are ignored and device enters a failure loop (an infinite loop waiting for debug attach).

53.7.3.1 Successful authentication

The ROM executes the following steps after successful debug authentication:

1. The ROM determines the final enable states of the debug ports based on pinned state from DCFG_CC_SOCU and the DC::CC fields.
2. The ROM evaluates part enables using the following logic:
   - Uses pinned states based on CC_SOCU_PIN (CMPA) and DCFG_CC_SOCU_NS_PIN (CFPA).
   - Evaluates socu_pinned and socu_default.
   - Evaluates debug port enables for ports which are not pinned using authentication protocol.
   - Debug_State = (SOCU_PIN & SOCU_DFLT) | (!SOCU_PIN & DC:: CC_SOCU)
   - Enables debug ports for bits set in the above evaluation.
3. The Debug Mailbox handler allows the following commands only if the enable bit is set in the final evaluation of DCFG_CC_SOCU.
   - ENTER_ISP_MODE command, only if default ISP_CMD_EN bit is set in DCFG_CC_SOCU.
   - SET_FA_MODE and ERASE_FLASH commands, only if default FA_CMD_EN bit is set in DCFG_CC_SOCU.
4. The ROM stores the beacons in the write-lockable register DBG_AUTH_SCRATCH.
   - DBG_AUTH_SCRATCH [15:0] = DAR::DC::CB[15:0]
   - DBG_AUTH_SCRATCH [31:16] = DAR::AB[15:0]
5. On receiving an EXIT_DBG_MB command, the ROM exits the debug mailbox handler loop and continues normal boot flow.

53.7.4 Fault Analysis (FA) mode

The ROM on this device has an FA mode command (SET_FA_MODE) to enable deletion of sensitive information (for example, keys) before giving the device to NXP for fault analysis. The ROM allows the SET_FA_MODE command only when a corresponding flag in Debug_State is set.

The FA_MODE sequence when activated by the boot ROM is:

1. Create a new version of Customer Field Programmable (CFPA) page.
2. Set ENABLE_FA_MODE word in the page to value C33CA55Ah.
3. Erase all KEYS and IVs in KEYSTORE Flash page.

4. Flush all temporary key registers.
5. Block PUF indexes.
6. Open all debug ports.
7. Enter while(1) loop.

53.7.5 Debug authentication use cases

This section describes use cases for debug authentication.

53.7.5.1 Return Material Analysis (RMA) use case

The diagram shows the RMA flow using debug authentication, where a debug credential certificate is issued for each field technician.

1. Vendor generates RoT key pairs and programs the device with a hash of RoT public key hashes before shipping.
2. Field technician generates own key pair and provides public key to vendor for authorization.
3. Vendor attests public key for field technician. In the debug credential certificate, vendor assigns access rights.
4. End customer with a locked product takes it to field technician.
    - Field technician uses credentials to authenticate with device and unlocks the product for debugging.

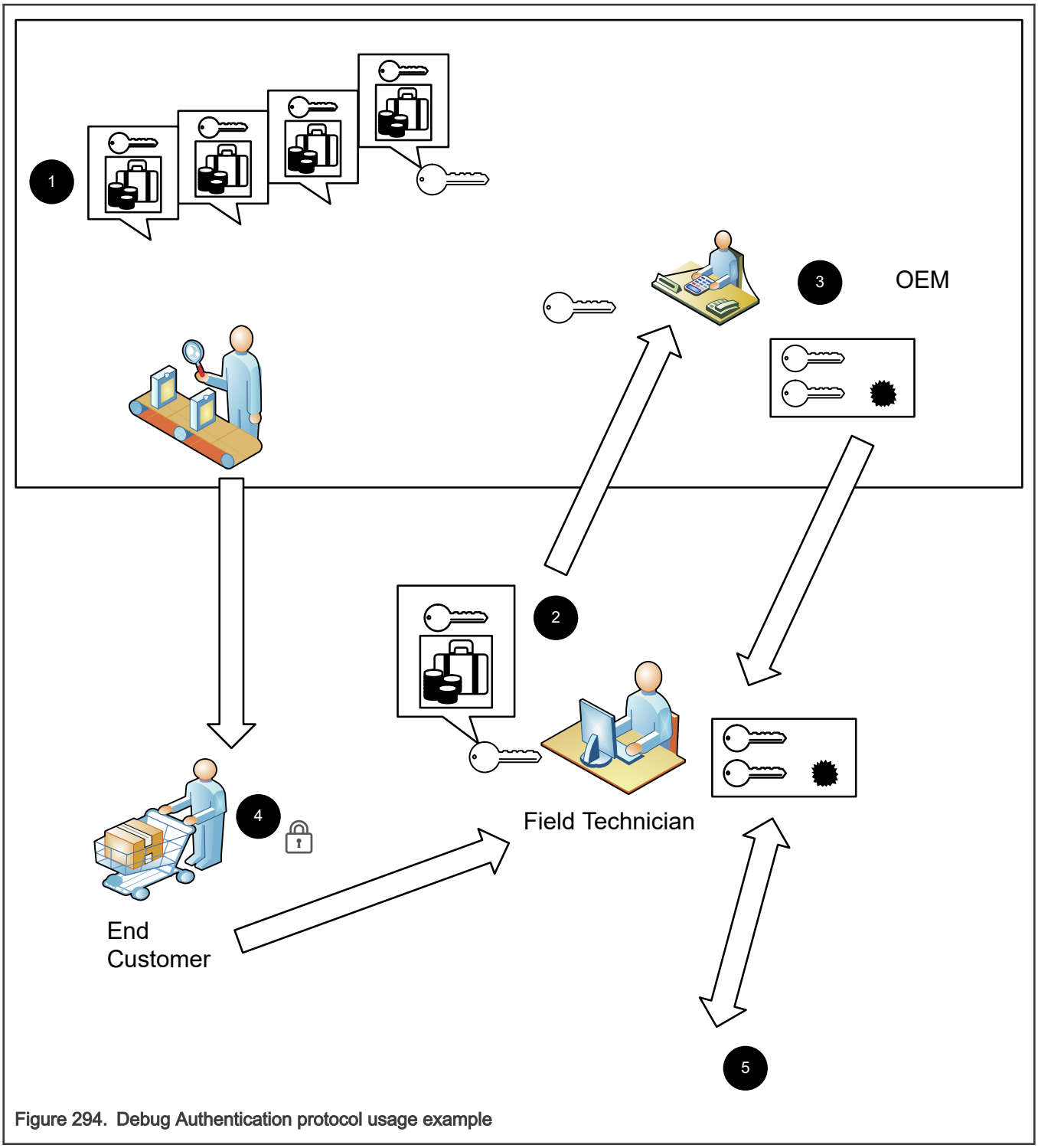


Figure 294. Debug Authentication protocol usage example

53.7.5.2 Module use case with Develop and Develop 2 Lifecycle states

CC_SOCU_PIN_NS and CC_SOCU_DFLT_NS allow module makers and OEMs to implement a tiered protection approach.

- The module maker, a tier-1 developer, develops secure code and define access rights to the module using CC_SOCU_PIN and CC_SOCU_DFLT.
- The code is configured to block access to the secure module but allow access to non-secure debug.

- Once the module is ready, the tier-1 developer releases the module to a tier-2 developer (for example, an OEM). The developer blocks debug access to secure mode and enables debug access to non-secure mode.
- A tier-2 developer develops a non-secure module and extends access rights configuration to that module using CC_SOCU_DFLT_NS and CC_SOCU_PIN_NS.

53.8 External signals

Table 430 shows the signals related to the debug process. A trace using the Serial Wire Output has limited bandwidth.

Table 430. Serial wire debug signals

Signal	I/O	Description
SWCLK	In	**Serial Wire Clock.** Provides the clock for the SWD debug logic when in Serial Wire Debug mode (SWD). SWCLK is the default signal of its pin. At the release of reset, the pin is pulled down internally.
SWDIO	I/O	**Serial Wire Debug Data input/output.** Used by an external debug tool to communicate with and control the part. SWDIO is the default signal of its pin. At the release of reset, the pin is pulled up internally.
SWO	Out	**Serial Wire Output.** Optionally provides data from the Instrumentation Trace Macrocell (ITM) for an external debug tool to evaluate. See the chip-specific Debug Subsystem information for the clocking required to enable SWO.

53.9 Memory map and register definition

53.9.1 Debug register descriptions

53.9.1.1 Debug memory map

DEBUGGER_MAILBOX0 base address: 4009_C000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
0	Command and status word (CSW)	32	See section	0000_0000
4	Request Value (REQUEST)	32	RW	0000_0000
8	Return Value (RETURN)	32	RW	0000_0000
FC	Identification (ID)	32	RO	002A_0000

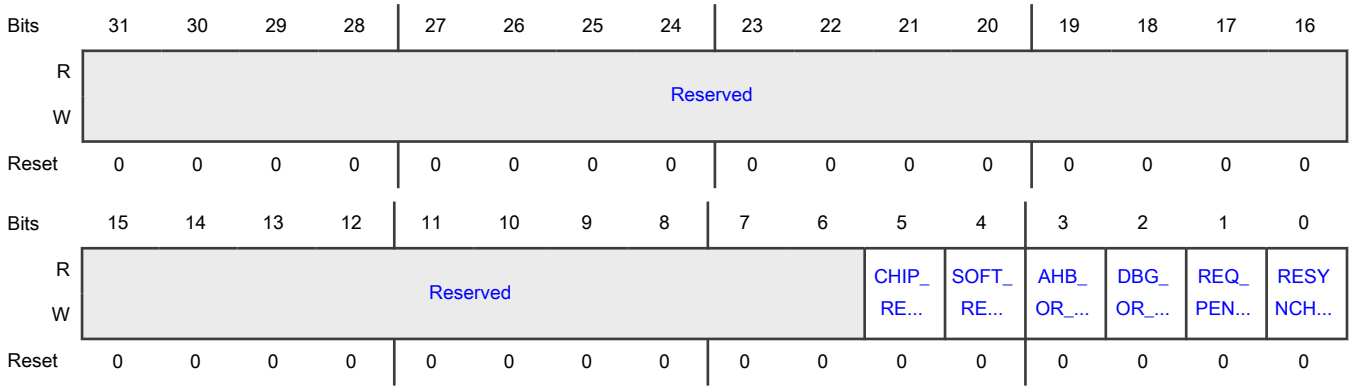
53.9.1.1.1 Command and status word (CSW)

The CSW register contains command and status bits to facilitate communication between the debugger and the device.

Offset

Register	Offset
CSW	0h

Diagram



Fields

Field	Description
31-6 —	Reserved
5 CHIP_RESET_REQ	Chip Reset Request This write-only bit causes the device (but not the DM-AP) to be reset by generating SYSRESET_REQ.
4 SOFT_RESET	Soft Reset The SOFT_RESET bit is write-only by the device and resets the DM-AP.
3 AHB_OR_ERR	AHB Overrun Error Indicates whether an AHB overrun has occurred: a RETURN value has been overwritten by the device before the RETURN value was read by the debugger. 0 - No AHB Overrun Error 1 - AHB Overrun Error. An AHB overrun occurred.
2 DBG_OR_ERR	Debug Overrun Error Indicates whether a debug overrun has occurred: a REQUEST value has been overwritten by the debugger before the REQUEST value was read by the device. 0 - No Debug Overrun error 1 - Debug overrun error. A debug overrun occurred.
1 REQ_PENDING	Request Pending A request is pending for the debugger: a value is waiting to be read from the REQUEST register. 0 - No request pending 1 - Request for resynchronization pending
0	Resynchronization Request

Table continues on the next page...

Table continued from the previous page...

Field	Description
RESYNCH_REQ	The debugger sets RESYNCH_REQ to request a resynchronization. 0 - No request 1 - Request for resynchronization

53.9.1.1.2 Request Value (REQUEST)

The REQUEST register is used by a debugger to send action requests to the device.

Offset

Register	Offset
REQUEST	4h

Diagram

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

Fields

Field	Description
31-0 REQUEST	Request Value Reads as 0 when no new request is present. Cleared by the device. Can be read back by the debugger in order to confirm communication.

53.9.1.1.3 Return Value (RETURN)

The RETURN register provides the responses from the device to the debugger.

Offset

Register	Offset
RETURN	8h

Diagram

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

Fields

Field	Description
31-0	Return Value
RET	This value is any response from the device to the debugger. If no new data is present, a debugger read is stalled until new data is available.

53.9.1.1.4 Identification (ID)

The ID register provides an identification of the DM-AP interface.

Offset

Register	Offset
ID	FCh

Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ID															
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0

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

Fields

Field	Description
31-0 ID	Identification Value

Appendix A
Revision History

A.1 Reference manual changes

Label	Date	Changes
Rev 2	11/2022	• In Chip-specific SYSCON information, added section DEVICE_ID vlaue. • In SYSCON: — Removed the Device ID register. — In Chip revision ID and Number register removed the bitfield MCO_NUM_IN_DIE_ID. • In FlexSPI: — Added Inline PRINCE encryption and decryption (IPED) CTR — Added AHBCR[CLRAHBRXBUF] — Added AHBCR[RESUMEDISABLE] — Added IPEDCTRL — Updated IPEDCTRL[6-8] to Reserved — Added IPEDCTXCTRL0 — Added IPEDCTX0IV1 — Added IPEDCTX0START — Added IPEDCTX0END — Updated IPEDCTXxSTART[GCM] to Reserved • In Power Mode API chapter removed the reference of High Power and Low Power modes. • In Chip-specific OPAMP information removed the table Resistor Values. • Removed the Device type register.

Table continues on the next page...

Table continued from the previous page...

Label	Date	Changes
		• Removed the bit DIEID[MCO_NUM_IN_DIE_ID] from the Chip revision ID and Number reigster. • Added a new section 'DEVICE_ID value' in Chip-specific SYSCON information. • Updated the description of DIEID[REV_ID]. • Added the register SWTATAB1 in Improved Inter-Integrated Circuit (I3C) chapter. • Added a bullet 'For proper ADC operation...the ADC peripheral' in the Chip specific section 'ADC subsystem Introduction'.
Rev 3	07/2023	• In Features and benefits section updated the bullet "Anti-rollback check during...monotonic counter, in CFPA" and added bullet "Image key certificate revocation...constraint field from ISK certificate". • In SYSCON: — Updated the register description of: ◦ Enable protection ◦ CDPA base address ◦ Flash hiding lockout address ◦ Flash hiding base address ◦ Flash hiding base DP address ◦ Hiding size DP address ◦ Hiding size address — In Clock generation chip specific chapter in Syscon added a note "The maximum frequency of OSTimer clock is 100 MHz".

Table continued from the previous page...

Label	Date	Changes
		— In Start-up behavior section updated the note. • In Analog Controller: — In section Features changed the frequency from 12 MHz to 16 MHz in the text "High-speed Crystal oscillator module control and status (from 12 MHz to 32 MHz)". — In ANACTRL[LDO_XO32M] changed the frequency from 12 MHz to 16 MHz in the bit field description "High-speed Crystal oscillator module control and status (from 12 MHz to 32 MHz)" • In Features and Benefits section updated Crystal oscillator with an operating frequency from 12 to 16 MHz. • In Non-Secure BootROM: — Fixed multiple typos. — In SPI flash recovery section changed "SPI_FLASH_CFG (0x9E404)" to "SPI_CAN_CFG (0x3E204)" and changed "SECURE_BOOT_CFG (0x9E41C)" to "SECURE_BOOT_CFG (0x3E21C)". — Added note in sections "Program 1-bit serial NOR flash device via SPI NOR Configuration Option Block", "Program Serial NOR Flash device using FlexSPI NOR Configuration Option" and "OTP-eFUSE definitions". — In Flash programming constraint removed the note "Flash ERASE and PROGRAM...or equal to 100 MHz".

Table continued from the previous page...

Label	Date	Changes
		— In Recovery boot section, changed the protected flash from "SPI_FLASH_CFG (0x3E204)" to "SPI_RECOVERY_BOOT_EN (0x3E204 bits 3:0)". — In Recovery boot updated the text "SPI_RECOVERY_BOOT_EN bits (SPI_CAN_CFG <1:0>) at address 0x3E204 of CMPA". • In ISP and IAP: — In RAM used by the ISP command handler section changed the RAM region from "0x3000_1000-0x3000_57FF" to "0x3000_1000-0x3000_7fff" and added "0x3000_4000 - 0x3000_7fff (For the USB HID RAM use)". — In Bootloader Status Error Codes section removed reference of SKBOOT from the table. • In ROM API — In the figure ROM API structure changed the name form "romapi_rng_generation_random" to "romapi_rng_generate_random". — In ROM API structure change the address from "0x1302fc00" to 0x0302fc00 — In the code changed the address from "0x1302fc00" to "0x1302FC00U". — In Version sections updated the code. — In flash_init section added a note. — In ffr_init section added a note.

Table continued from the previous page...

Label	Date	Changes
		— In Flash APIs Example section updated the code in Example 1. — In otp_program section updated the code and changed index 0 to index 7 (CUSTOMER_WORD0). — In General description section removed the kp_api_init_param_t definition and removed the table Description of the structures in the core context. • In ADC chapter added section "Cycle per conversion". • In OSTIMER chip specific information: — In section OSTIMER configuration addd note "AHB_clk clock source for OSTimer is limited to maximum frequency of 100 MHz". • In Chip specific Analog-to-Digital Converter (ADC) section ADC input connections: — In the table ADC analog channel input connections changed: ◦ Analog Channel Input from ADC0IN5B/VREFN to VREFN, changed Input type from Standard Dedicated with internal (VSS) and removed ADC0IN5B — In the ADC trigger inputs section added text "There are four trigger sources for each ADC module".

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 http://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 data sheet 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.

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.

Airfast — is a trademark of NXP B.V.

CodeWarrior — is a trademark of NXP B.V.

ColdFire — is a trademark of NXP B.V.

ColdFire+ — is a trademark of NXP B.V.

CoolFlux — is a trademark of NXP B.V.

CoolFlux DSP — is a trademark of NXP B.V.

EdgeLock — is a trademark of NXP B.V.

I2C-bus — logo is a trademark of NXP B.V.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 07/2023
Document identifier: LPC553xRM